

The logo for SystemC, featuring the word "SYSTEMC" in a white, sans-serif font inside a dark blue, semi-circular shape. The background is split into a yellow top half and a dark blue bottom half, with a horizontal line of thin white lines separating them.

SYSTEMC™

Transaction Level Modeling with SystemC

Thorsten Grötter
Engineering Manager
Synopsys, Inc.

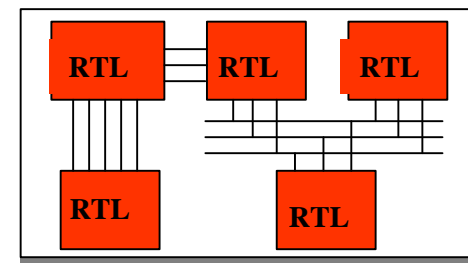
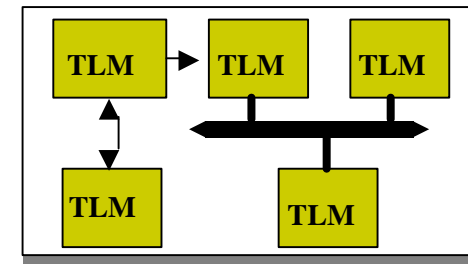
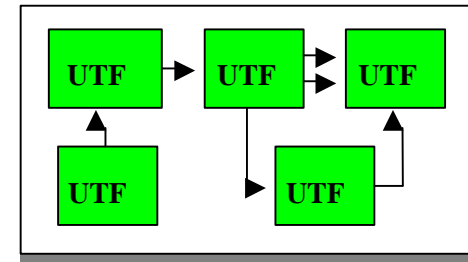
Outline

- Abstraction Levels
- SystemC Communication Mechanism
- Transaction Level Modeling of the AMBA AHB/APB Protocol
- Generic Transaction Level Channel

Abstraction Levels:

■ Transaction Level

- Layer 3: Message Layer
 - ◆ Model (un-)timed functionality
 - ◆ Point-point communication
- Layer 2: Transaction Layer
 - ◆ Analyze SoC architecture, early SW development
 - ◆ Estimated timing
- Layer 1: Transfer Layer
 - ◆ Cycle true but faster than RTL
 - ◆ Detailed analysis, develop low-level SW



■ Pin Level

- Layer 0: Register Transfer Level

SystemC Language Architecture

Methodology-specific channels

Elementary Channels

Signal, Timer, Mutex, Semaphore, FIFO, etc.

- Time
- Concurrency
- Modules
- Processes
- Interfaces
- Ports
- Channels
- Events
- Event-driven sim. kernel

Data Types

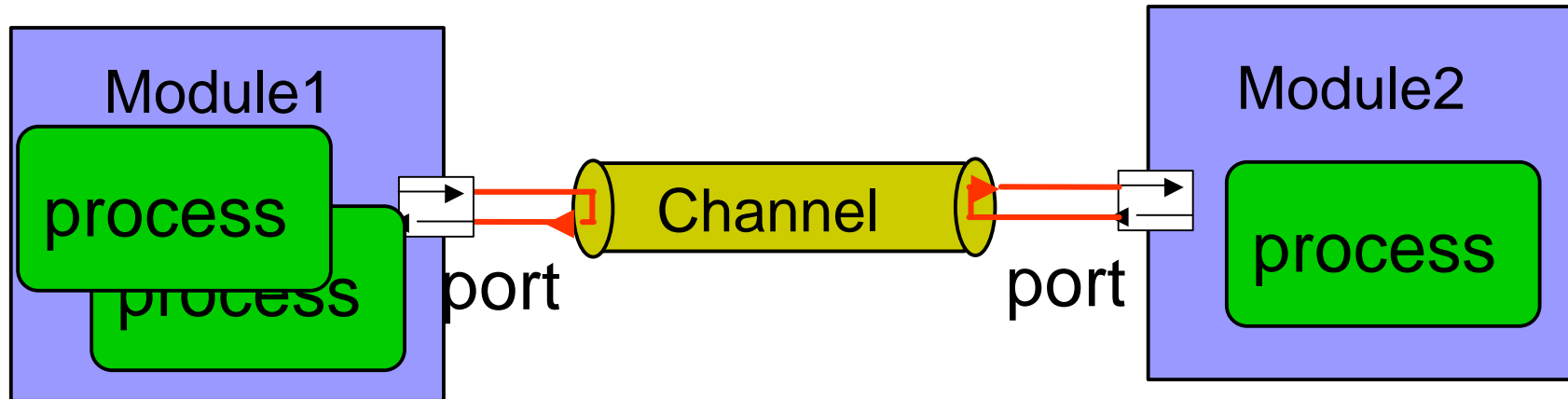
- 4-valued logic (0, 1, X, Z)
- 4-valued logic-vectors
- Bits and bit-vectors
- Arbitrary-precision integers
- Fixed-point numbers
- C++ user-defined types
- C++ built-in types (int, char...)

C++ Language Standard

SystemC Communication Mechanism: How?

- Separate Functionality from Communication
 - Functionality: implemented in **Modules**
 - Communication: implemented in **Channels**
- Interface Method Calls (IMC)
 - The collection of a fixed set of communication **Methods** is called an **Interface**
 - **Channels** implement one or more **Interfaces**
 - **Modules** can be connected via their **Ports** to those **Channels** which implement the corresponding **Interface** (`sc_port<interface>`)

Primitive Channel

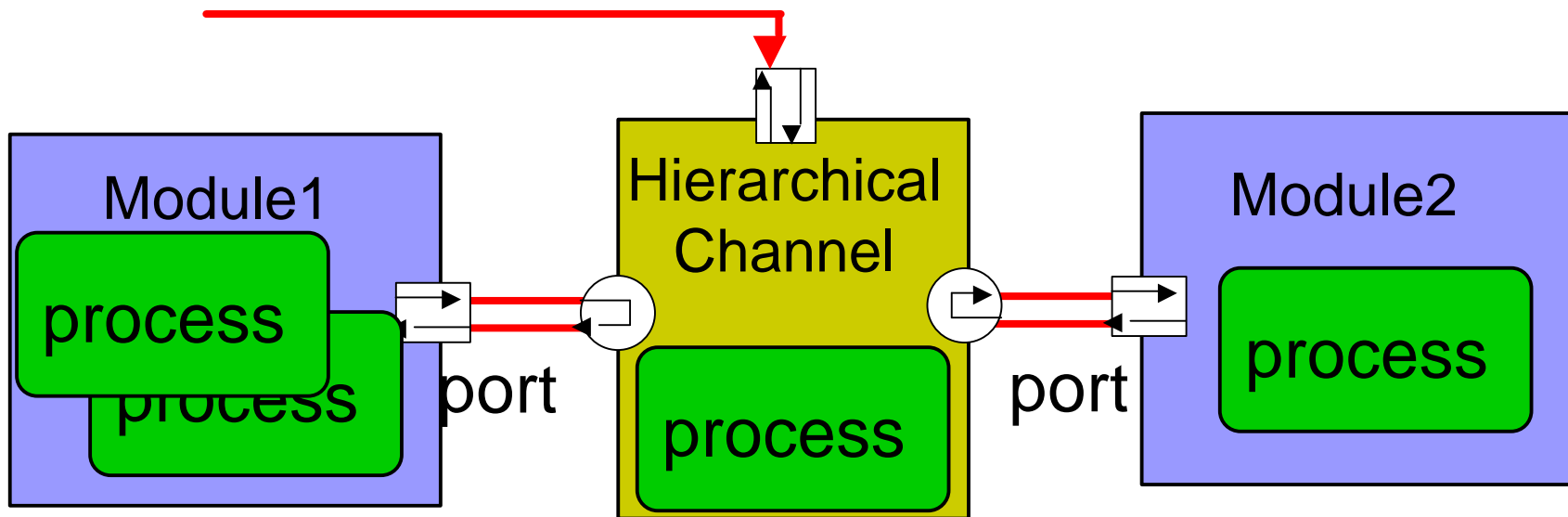


```
Module1::process()  
{  
  port->write(42);  
}
```

```
Module2::process()  
{  
  x = port->read();  
}
```

Hierarchical Channel

Channels can be hierarchical, i.e. they can contain processes, ports, modules and channels.



Outline

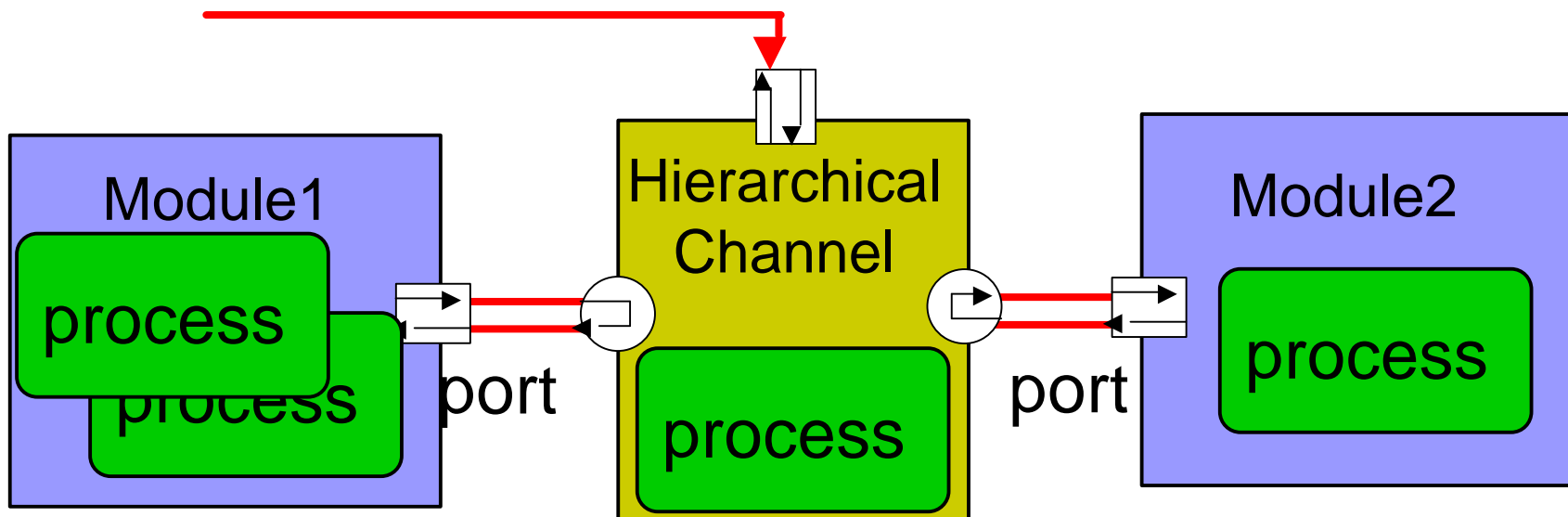
- **Abstraction Levels**
- **SystemC Communication Mechanism**
- **Transaction Level Modeling of the AMBA AHB/APB Protocol**
- **Generic Transaction Level Channel**

Transaction-Level Modeling of the AMBA AHB/APB Protocol

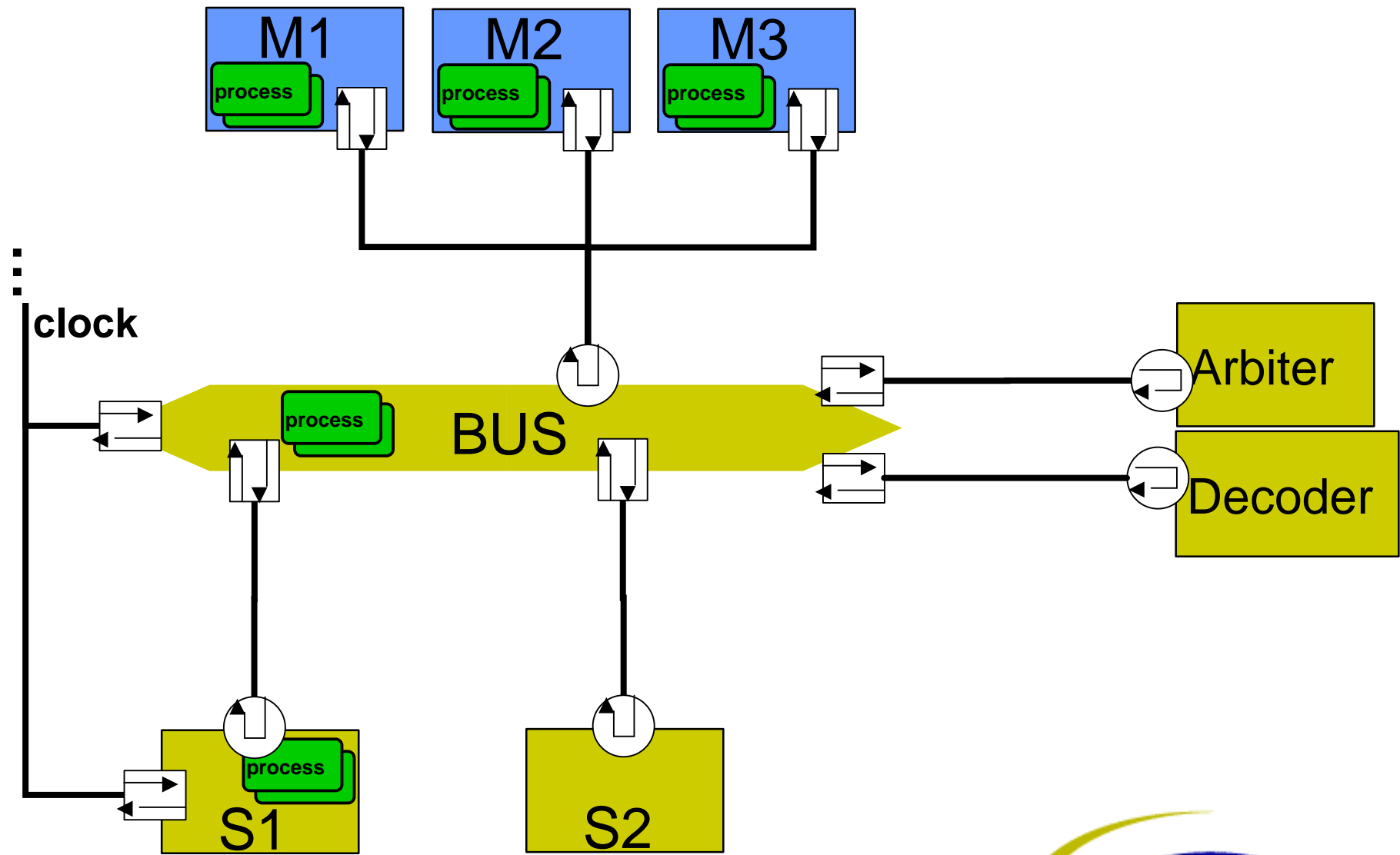
- Background
 - implemented in the spirit of “simple_bus” (transfers modeled via dedicated interface methods)
- Requirements
 - cycle-accurate (Layer 1)
 - optimal simulation speed
 - ease of use (blocking interface)
 - integration of low-level models (non-blocking interface)
 - enable SW debugging (debug interface)

Hierarchical Channel

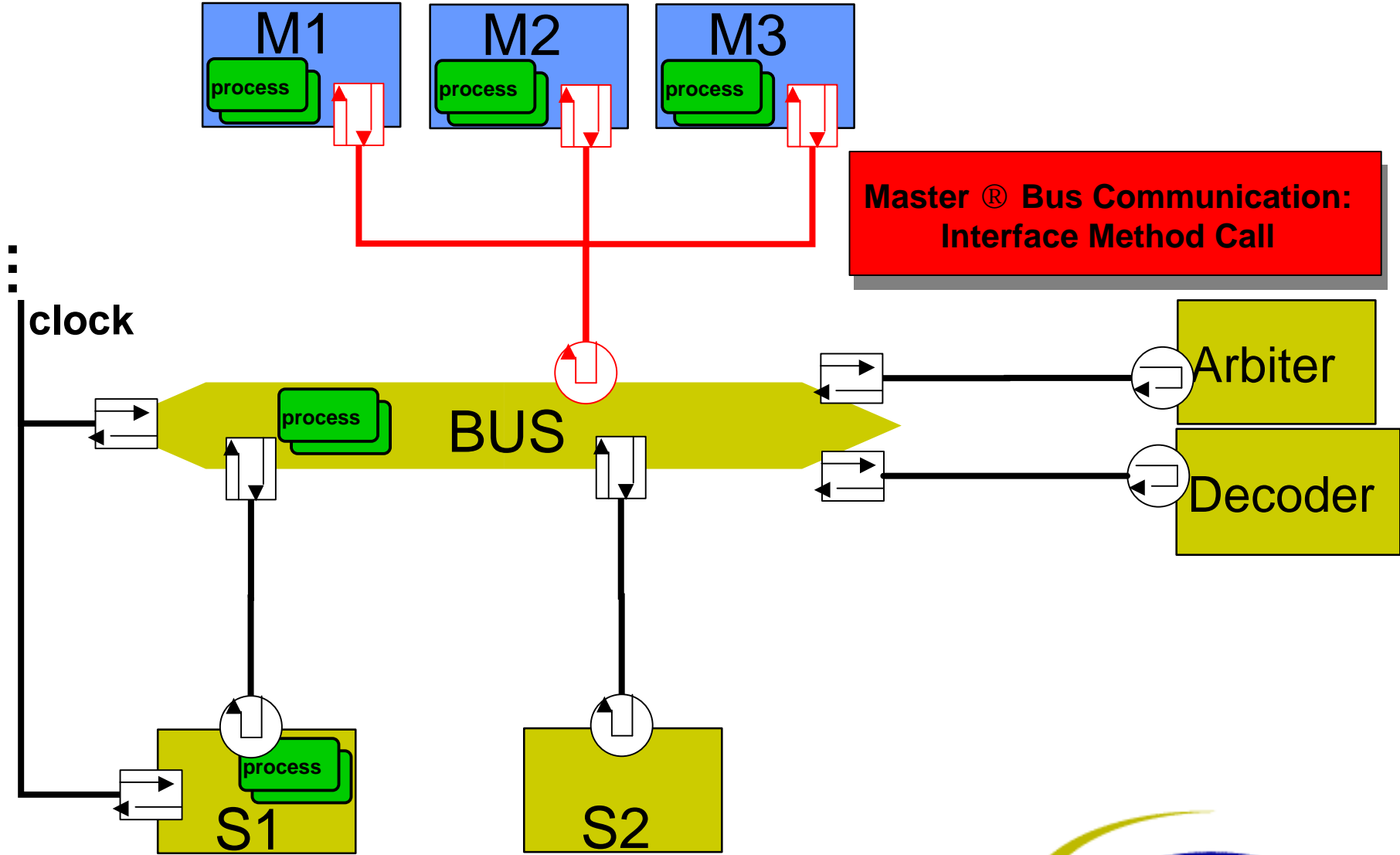
Channels can be hierarchical, i.e. they can contain processes, ports, modules and channels.



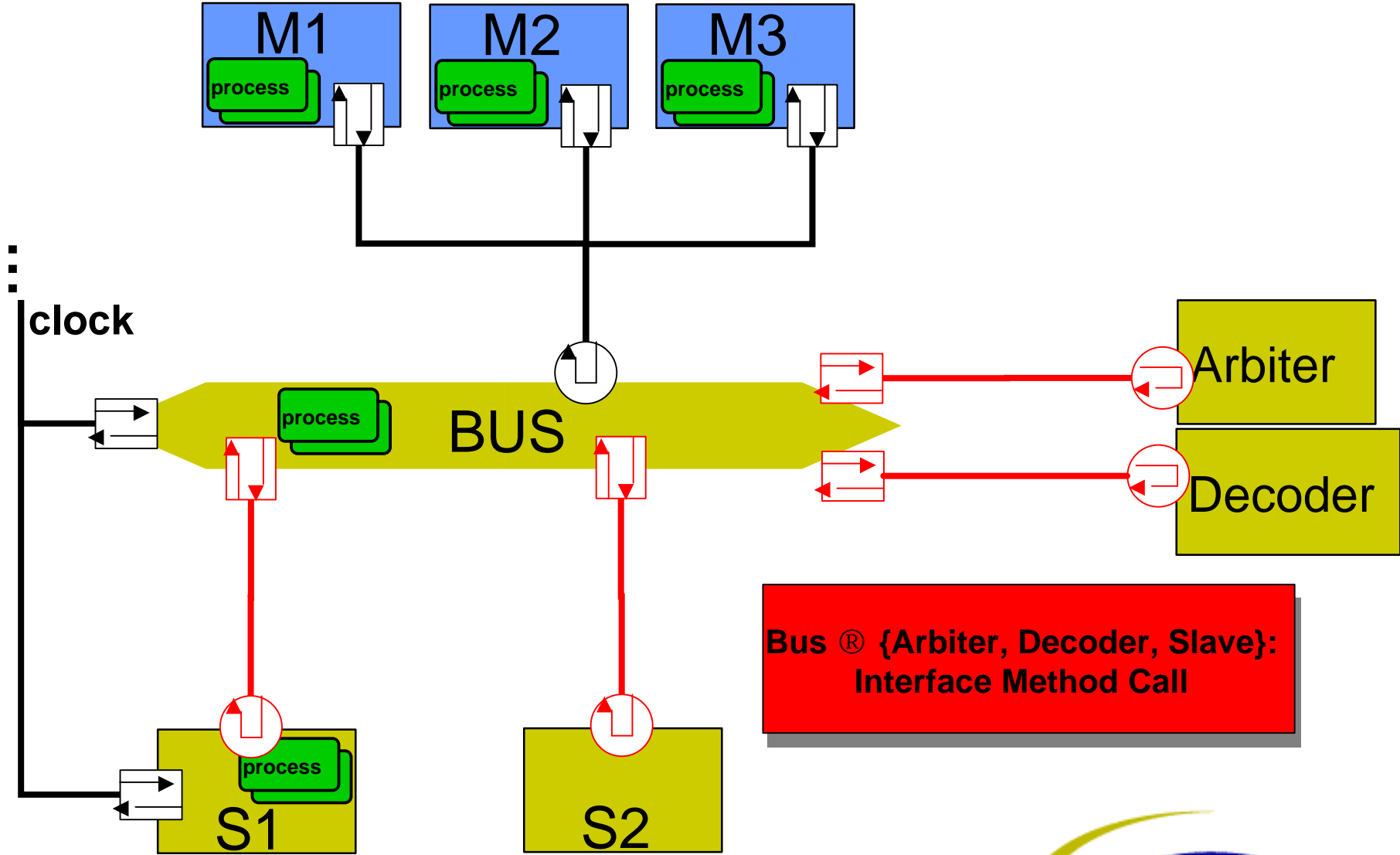
TLM Bus Model (Layer 1, simplified)



TLM Bus Model (Layer 1, simplified)



TLM Bus Model (Layer 1, simplified)



AHB Master-Bus Interface

- **Blocking**
 - `bool burst_read(id, data, start_address, burst_length, burst_mode, num_bytes);`
 - `bool burst_write(id, data, start_address, burst_length, burst_mode, num_bytes);`
- **Non-Blocking**
 - `void request(id);`
 - `bool has_grant(id);`
 - `void init_transaction(id, read_mode, address, ...);`
 - `void set_data(id, data);`
 - `bool response(id, status);`
 - ...

Note that such interfaces are always protocol-specific

AHB Interfaces (cont'd)

- Direct Bus Interface (Debugging)
 - bool direct_read(address, data, num_bytes);
 - bool direct_write(address, data, num_bytes);
- Bus-Slave Interface
 - void set_data(data);
 - void control_info(burst_mode, transfer_type, ...);
 - void read(address, num_bytes);
 - void write(address, num_bytes);
 - bool response(status);
 - ...

AMBA AHB Components

- AHB bus with external arbiter and decoder
- Interconnection matrix for AMBA multilayer support
- AHB bus monitor
- Processors (ARM 926, ARM 946)
- Pin Level adaptors
- Example masters and slaves
- Example AHB platforms

AMBA APB Components

- AHB-APB bridge with APB bus
- APB bus monitor
- Pin Level adaptors
- APB peripherals
 - Remap and pause controller
 - Interrupt controller
 - APB timer
- Example AHB/APB platforms

Outline

- Abstraction Levels
- SystemC Communication Mechanism
- Transaction Level Modeling of the AMBA AHB/APB Protocol
- Generic Transaction Level Channel

Generic Transaction-Level Channel

- Background

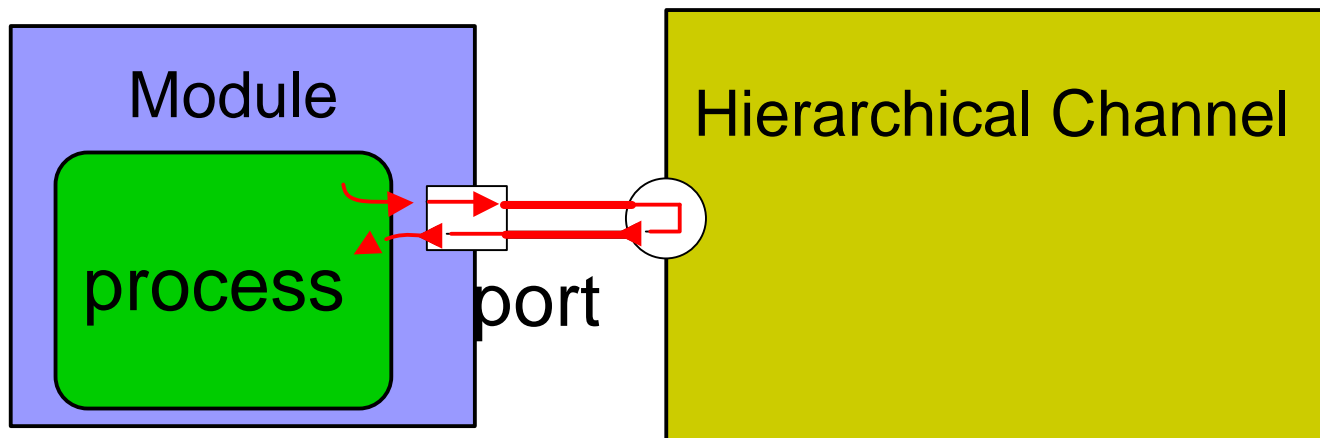
- Developed by Nokia and Synopsys in cooperation with TI and Sonics for OCP

- Requirements

- Point-to-point connection
(module « generic channel « module)
 - ◆ Data passing and synchronization
- Interface independent of communication protocol
 - ◆ Protocol to be implemented by the user
 - ◆ Can be used for Layers 1, 2, and 3

Generic TL Channel: A Building Block for easy Transaction-Level Modeling

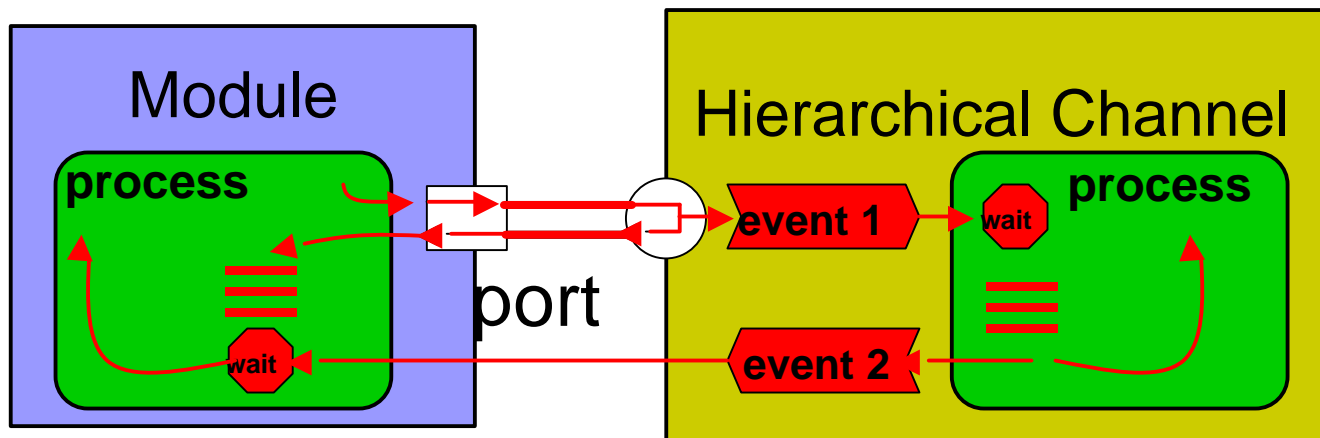
Recap interface method calls (IMCs)



Interface methods are executed in the context of the calling process.

Generic TL Channel: A Building Block for easy Transaction-Level Modeling

How to implement concurrent behavior (pipelining, split transfers, ...) with IMCs?



More events are needed when

- module needs to wait for channel to be ready
- channel needs to wait for module to collect data

Generic TL Channel: A Building Block for easy Transaction-Level Modeling

The generic channel is a building block that eases implementing data passing and synchronization of initiator and target.



It is not a ready-made implementation of any communication protocol.

Generic Channel Interface (simplified)

- Initiator ("Master")
 - TdataCI* GetDataCI();
 - bool MputRequest*();
 - bool MgetResponse*(bool release);
 - void Mrelease(time);
- Target ("Slave")
 - TdataCI* GetDataCI();
 - bool SgetRequest*(bool release);
 - bool SputResponse*();
 - void Srelease(time);

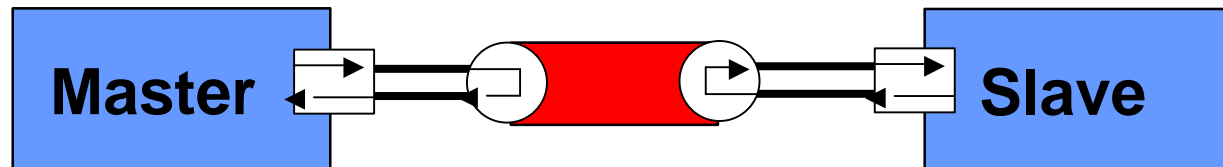
TdataCI:

- template parameter
- depends on application and communication protocol being used

Flow of Events

```
data = port->GetDataCI();
```

```
data = port->GetDataCI();
```



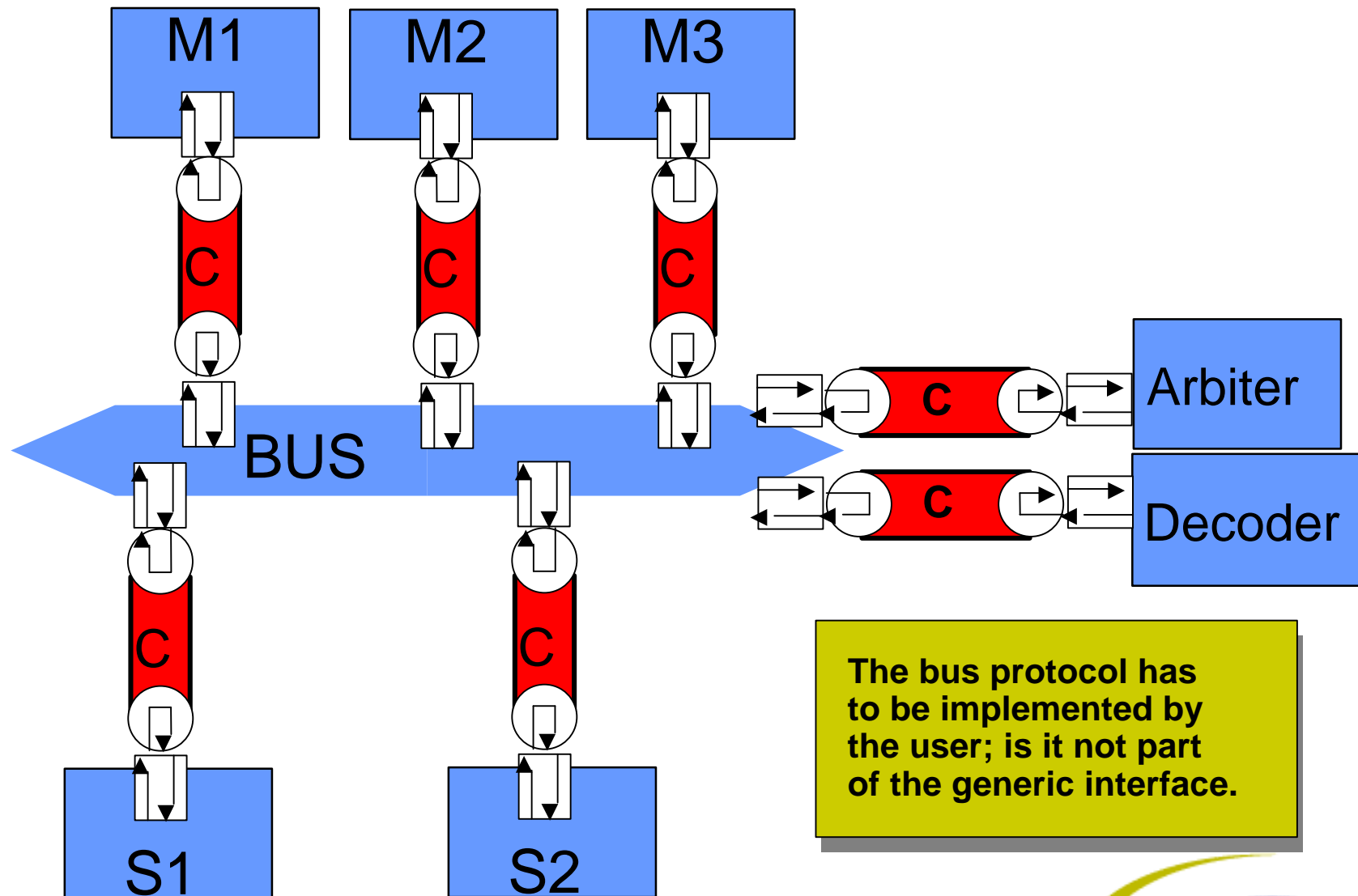
```
data->input = 42;  
MputWriteRequestBlocking();
```

data type is likely to depend on the protocol

```
MgetResponseBlocking();  
if (data->response) ...
```

```
SgetRequestBlocking(true);  
int mem = data->input;  
if (mem != 0)  
    data->response = true;  
else  
    data->response = false;  
SputResponseBlocking();
```

Bus-based System w/ Generic Channels



Summary

- **Abstraction Levels**

 - Transaction Level comprises multiple Layers
(Message, Transaction, Transfer)

- **SystemC Communication Mechanism**

 - Interface Method Calls (IMC)

- **Transaction Level Modeling of the AMBA Protocol**

 - Efficient implementation based on IMCs

- **Generic Transaction Level Channel**

 - Building Block for easy Transaction Level Modeling

Further information

- **SystemC and Transaction-Level Modeling**
 - OSCI website: www.systemc.org
 - *“System Design with SystemC”*
- **Generic Transaction-Level Channel White Paper**
 - www.ocpip.org/data/systemc.pdf
- **AMBA models**
 - www.synopsys.com

