# TLM

Transfer-Level Modeling in SystemC

# Context

# SystemC

Model hardware using C++

- Modules

- Threads and Methods

### Levels of abstraction

RTL:

- No gates, no wiring

- Timing (clocks)

- Logic (registers)

- Signals (pins)

TL:

- No gates, no wiring

- No timing

- Behavioral model

- Transaction

# Usage

### Use cases

- Software development
- Software performance
- Architecture analysis
- Hardware verification

# Requirements

- Fast

- Early

- Accurate

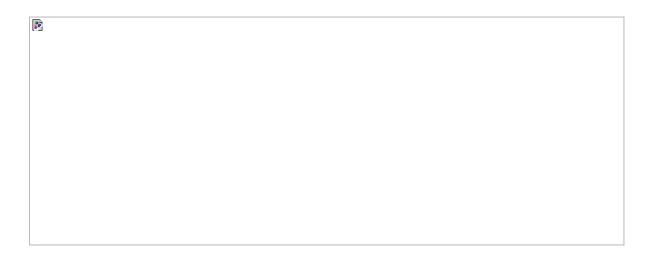
# Overview

## Main concepts

IPs → Behavioral models

IOs → Function calls

Information → Objects



### **TLM 2.0**

Memory-mapped buses and Interoperability

- Standard set of APIs
- Generic utility classes
- Coding style directives

# Implementation

## Target

```
SC_MODULE(Target) {
    /* Simple target socket
     * Parameters:
     * - module name (mandatory);
     * - bus width (optional, defaults to 32);
     * - protocol types, which mainly concerns the type of payload (optional,
          defaults to `tlm_base_protocol_types').
     */
    tlm_utils::simple_target_socket<Target, 32, tlm::tlm_base_protocol_types> socket;
    SC_CTOR(Target) : socket("socket") {
        . . .
    . . .
};
```

### Initiator

```
SC_MODULE(Initiator) {
   tlm_utils::simple_initiator_socket<Initiator, 32, tlm::tlm_base_protocol_types> socket;

SC_CTOR(Initiator) : socket("socket") {
    ...
}
...
};
```

### Transaction

- command: read or write

- address: destination

- data: buffer

- length: buffer size

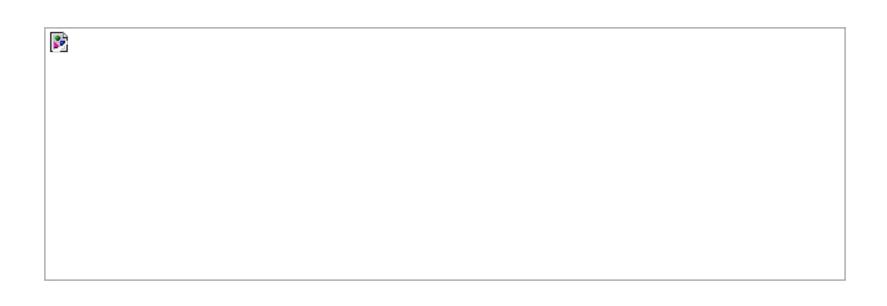
- response\_status: for errors

- streaming\_width: for bursts

byte\_enables: to select bytes

- dmi\_allowed: set by the target

extensions: to extend the protocol



## Coding styles

### **Loosely-timed**:

- Blocking transport
- Time decoupling

### **Approximately-timed**:

- Non-blocking transport
- Events

# Questions?

## Complex protocols extension

- Sockets: base classes extension.
- Transactions: use of the extension field.
- Transport: keep the usual functions (blocking or not) to respect the coding styles.

Preserve the interoperability!

