

Practical Co-simulation with Cycle-callable Models

Hervé Alexanian



Problem Statement

Systematic Pin-Level Wrapping

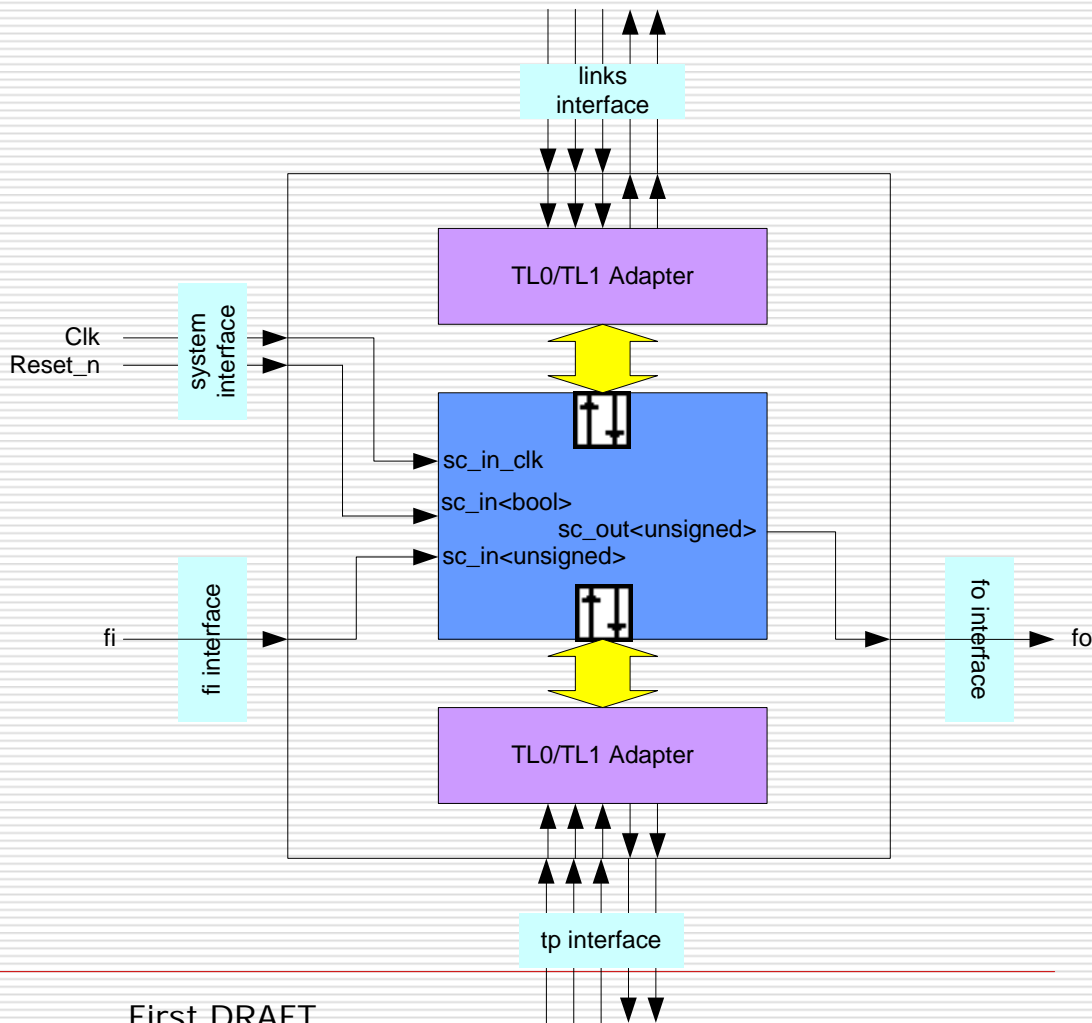
- Protocol Abstraction
 - Standard on-chip protocol interfaces (OCP, AMBA...)
 - Signal grouping (MCmd/MAddr/MThreadID/MBurst)
 - Not all signal transitions are interesting
- Configurable Protocols
 - Protocols define parameters
 - Results in configurable models
- Need Variable Pin Lists
 - Protocol parameters define pin lists
 - Variable widths
- Configurable and Reusable Adapters

Definitions

- Cycle-callable (TL1)
 - Clocked modeling
 - Abstracted protocol data
 - Transfers represented cycle accurately
- Pin-level (TL0)
 - SystemC ports to bit or integer types
 - `sc_bv<N>`, `bool`, `unsigned int`
- Co-simulation
 - Using commercial HDL simulator
 - Modules implemented as SystemC within hierarchy
 - SystemC port to HDL mapping
 - Top-level: HDL or SystemC

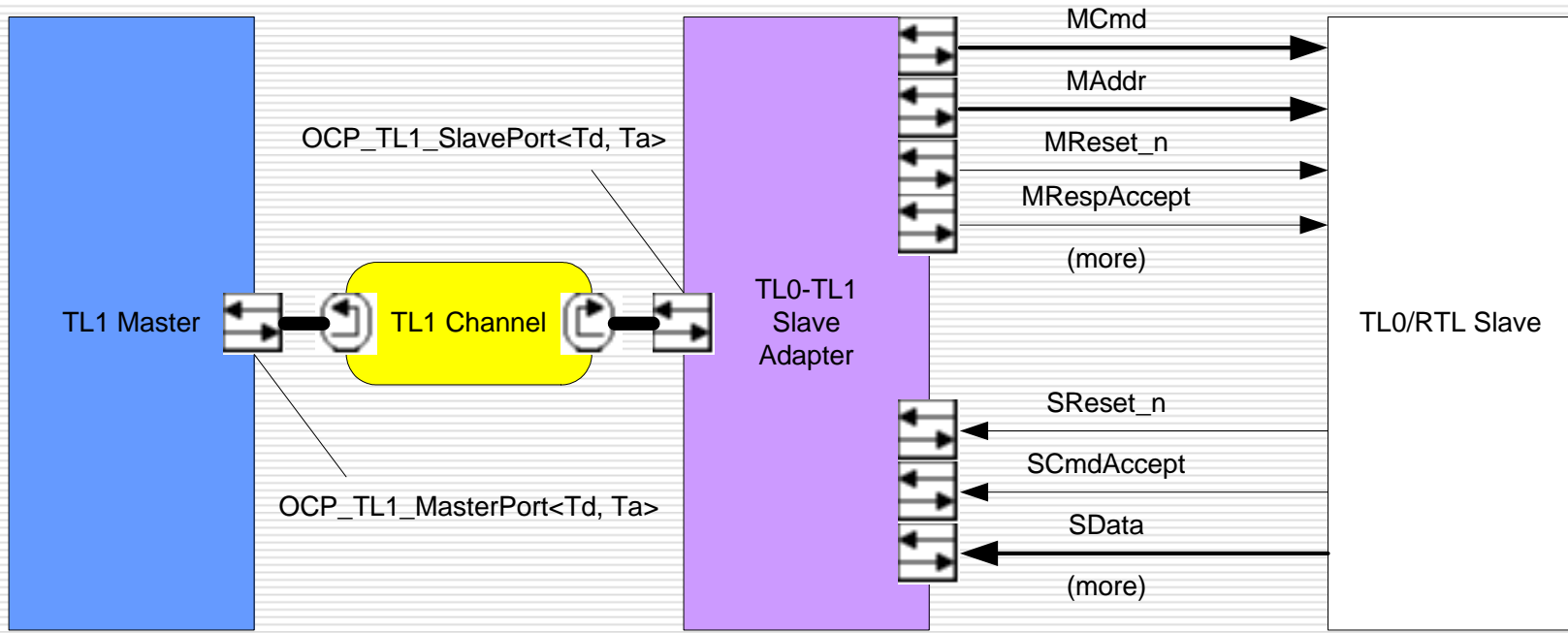
Co-simulation

- Able to write Cycle-Callable Model
 - Protocol CC ports
 - Plain clock and signal ports
- Systematic Pin-Level Wrapping
 - Automate protocol port creation
 - Automate binding with adapters
- In Place of RTL or as Reference Model



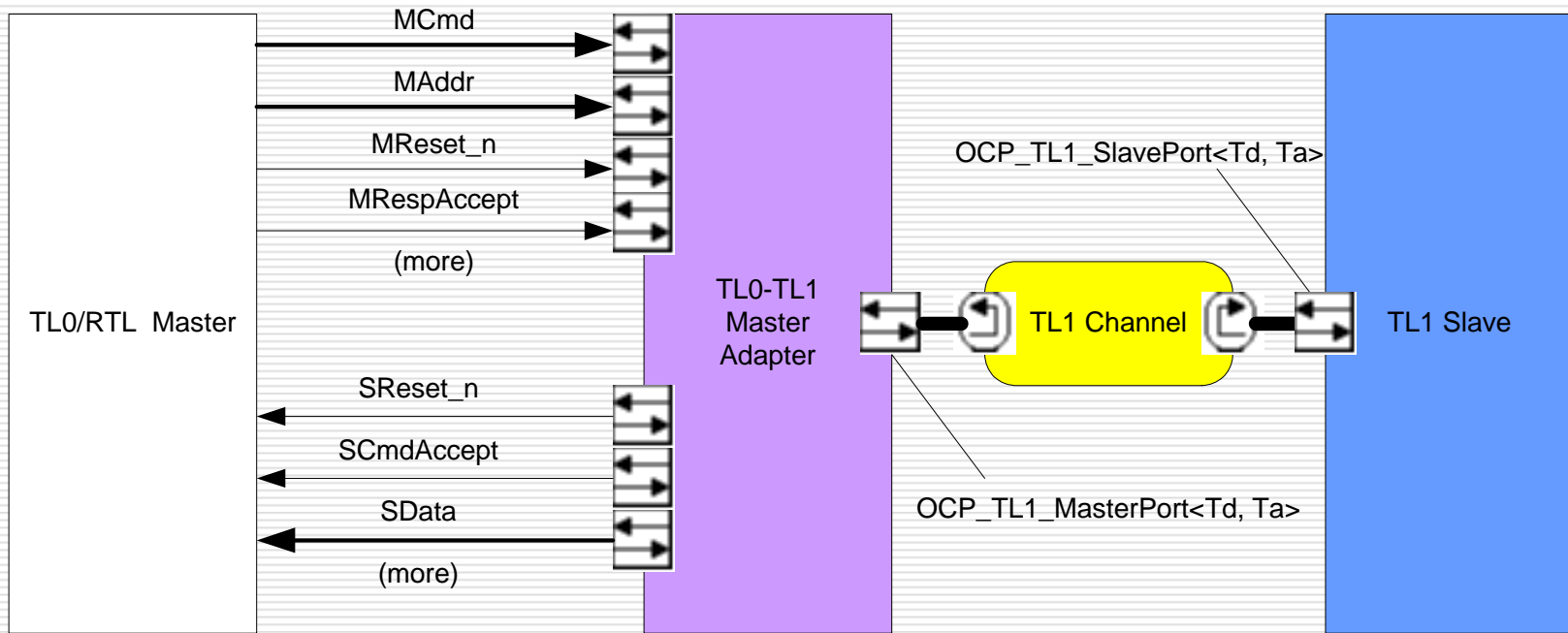
Pin/Cycle-Callable Adapters (OCP Example)

- ❑ Cycle-callable Interface Port on one End
- ❑ Variable List of `sc_in`, `sc_out` ports on other End
- ❑ Master Pins/Slave Cycle-Callable



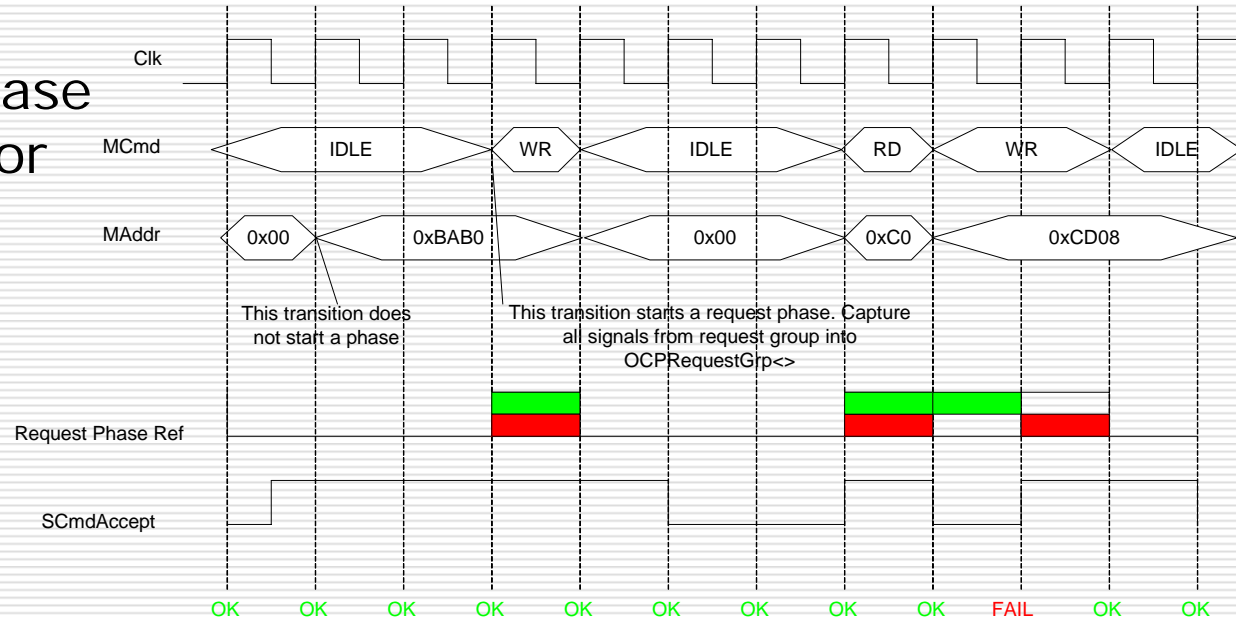
Pin/Cycle-Callable Adapters (OCP Example)

- Dual case: Slave
Pins/Master Cycle-Callable



Pin/Cycle-Callable Adapter

- Precise protocol phase timing definitions for phase-accurate modeling



- Phase Translation

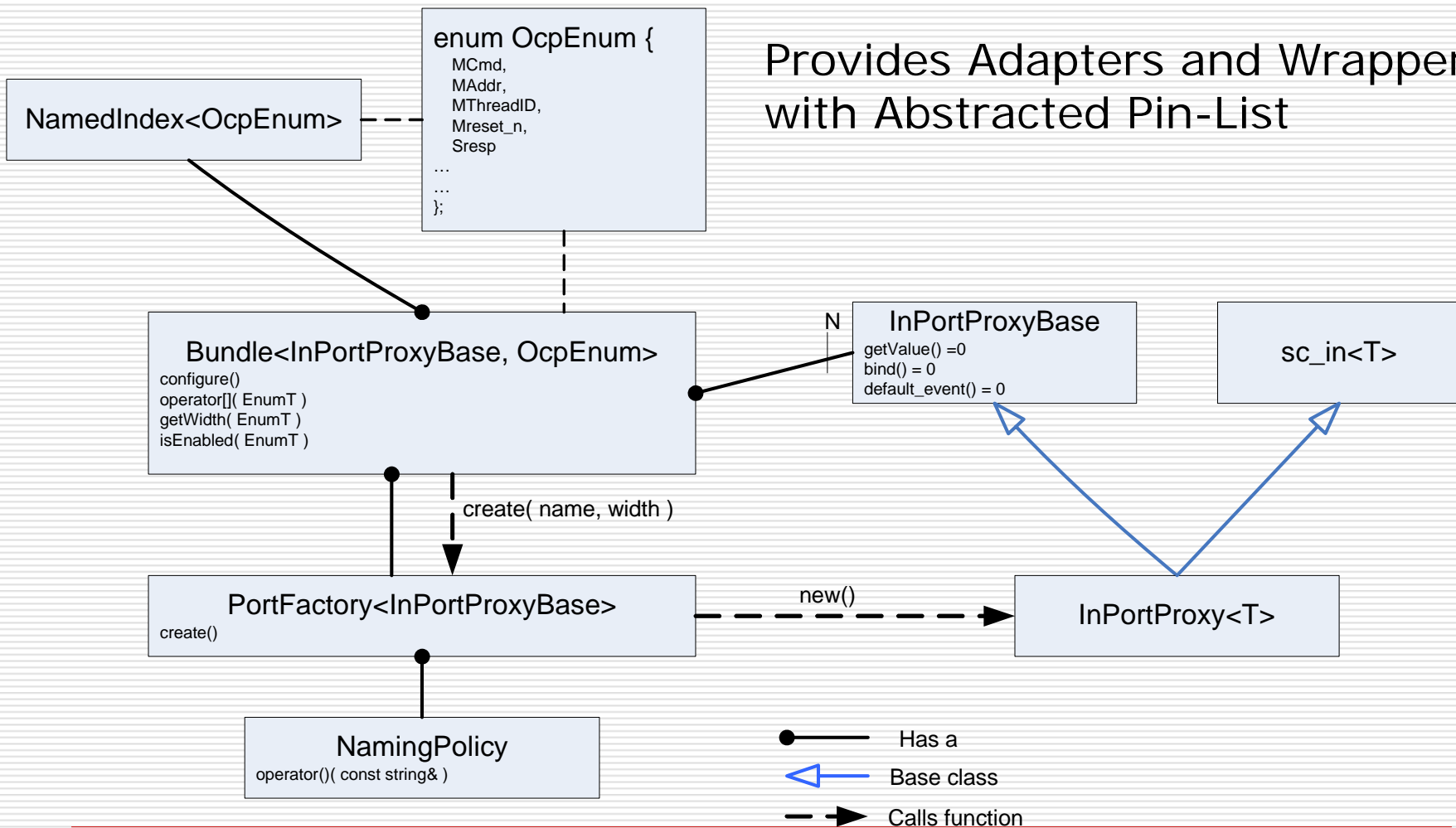
- MCmd non-idle starts request phase
- SCmdAccept == 1 with request started ends request phase

Beginning Cycle of Phase

Ending Cycle of Phase

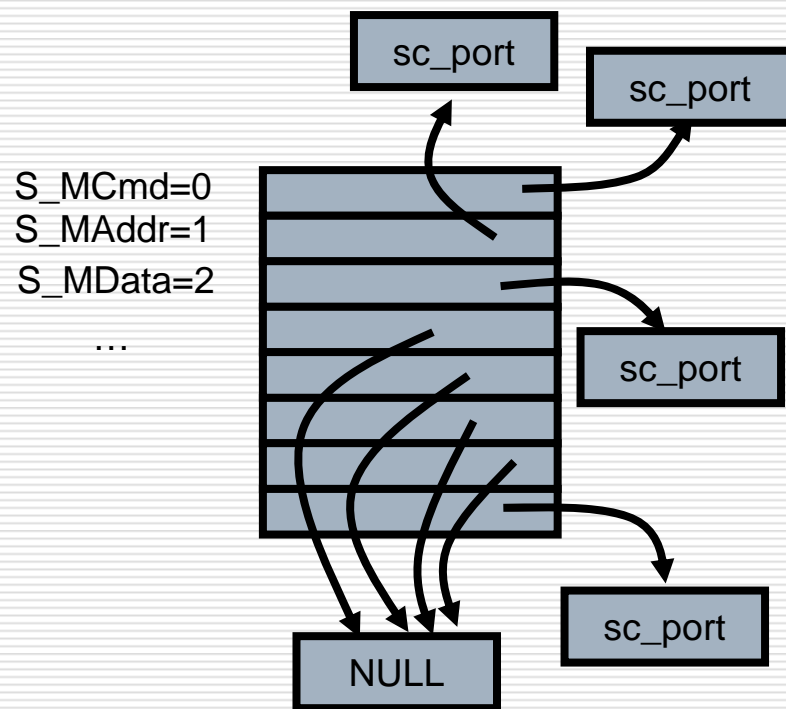
Bundle Access Class

Provides Adapters and Wrappers with Abstracted Pin-List



Bundle Access Class

- Templated Class Provides Access to Configurable Pin Bundle
 - Random access to signals
 - SignalT& operator[]
 - Configuration at elab time
 - Signals enabled or not
 - Factory class creates actual nodes
 - Based on name, width, direction
 - Support for reference ports
- Pin-level Wrappers and Adapters use Bundle Instances



Bundle Abstraction Example

```
template <typename Request, typename Response>
void
OcpXT10T11SlaveAdapter<Request, Response>::responseT10Sample()
{
#define ASSIGN_RESP_FIELD( field ) \
    if ( m_t10In.isEnabled( S_##field ) ) \
        m_currentResponse.field = m_t10In[ S_##field ].getValue();

    SRespType resp = static_cast<SRespType>( m_t10In[S_SResp].getValue() );
    if ( resp != SRESP_NULL && !m_responsePendingAccept ) {
        m_currentResponse = m_slaveP->createResponse();
        m_currentResponse.SResp = resp;
        ASSIGN_RESP_FIELD( SData );
        ASSIGN_RESP_FIELD( SDataInfo );
        ASSIGN_RESP_FIELD( SRespInfo );
        ASSIGN_RESP_FIELD( SThreadID );
        ASSIGN_RESP_FIELD( SDestID );

        m_slaveP->startResponse( m_currentResponse );
        m_responsePendingAccept = true;
    }
}
```

Varying Pin Widths

- Determine Pin Widths
 - `sc_bv<N>`: compile time determination
 - Use integer types: `sc_uint`, `sc_biguint`

- Simulator may require exact match
 - ModelSim
 - NCSim tolerates integer types and gives warnings

- Would Like to Ship Binary Models
 - Protect IP
 - Avoid tedious compile of generated wrappers

Managing Run-time Data Width

□ Did you know contexts?

- `sc_length_context`
- `sc_length_param`

□ Use base integer classes

- `sc_uint_base`
- `sc_unsigned`

```
class InPortProxyBase : public PortProxyGetIf {
public:
    InPortProxyBase( int len = 0 ) :
        m_context( len ? sc_length_param( len ) : sc_length_param() ) {}
    virtual void bind( InPortProxyBase& ) = 0;
    virtual void bind( sc_interface& ) = 0;
    virtual void bind( sc_port_base& ) = 0;

    template <typename T> void bindTo( sc_in<T>& in );
    virtual ~InPortProxyBase() {}
protected:
    sc_length_context          m_context;
};

template <typename T>
class InPortProxy: public InPortProxyBase, public sc_in<T>
{
public:
    InPortProxy( const char* szName, int size ) :
        InPortProxyBase( size ),
        sc_in<T>( szName ) {
            m_context.end();
        }

    virtual void getLongValue( unsigned int& ) const;
    virtual unsigned int getValue() const;
    virtual sc_event_finder& defaultEvent() const;
    ...
};
```

Overview of Co-simulation Use Model

- Cadence NCSim
 - Version 5.6, 5.7, 5.8
 - SystemC Models require Verilog wrapper
 - Name-based port matching
 - Can use `sc_uint_base` with length context

- Mentor Graphics ModelSim Questa
 - Version 6.1e, 6.2a
 - No Verilog Wrapper
 - SystemC to be compiled with `sccom`
 - Can use `sc_unsigned` with length context
 - Needs data members for ports

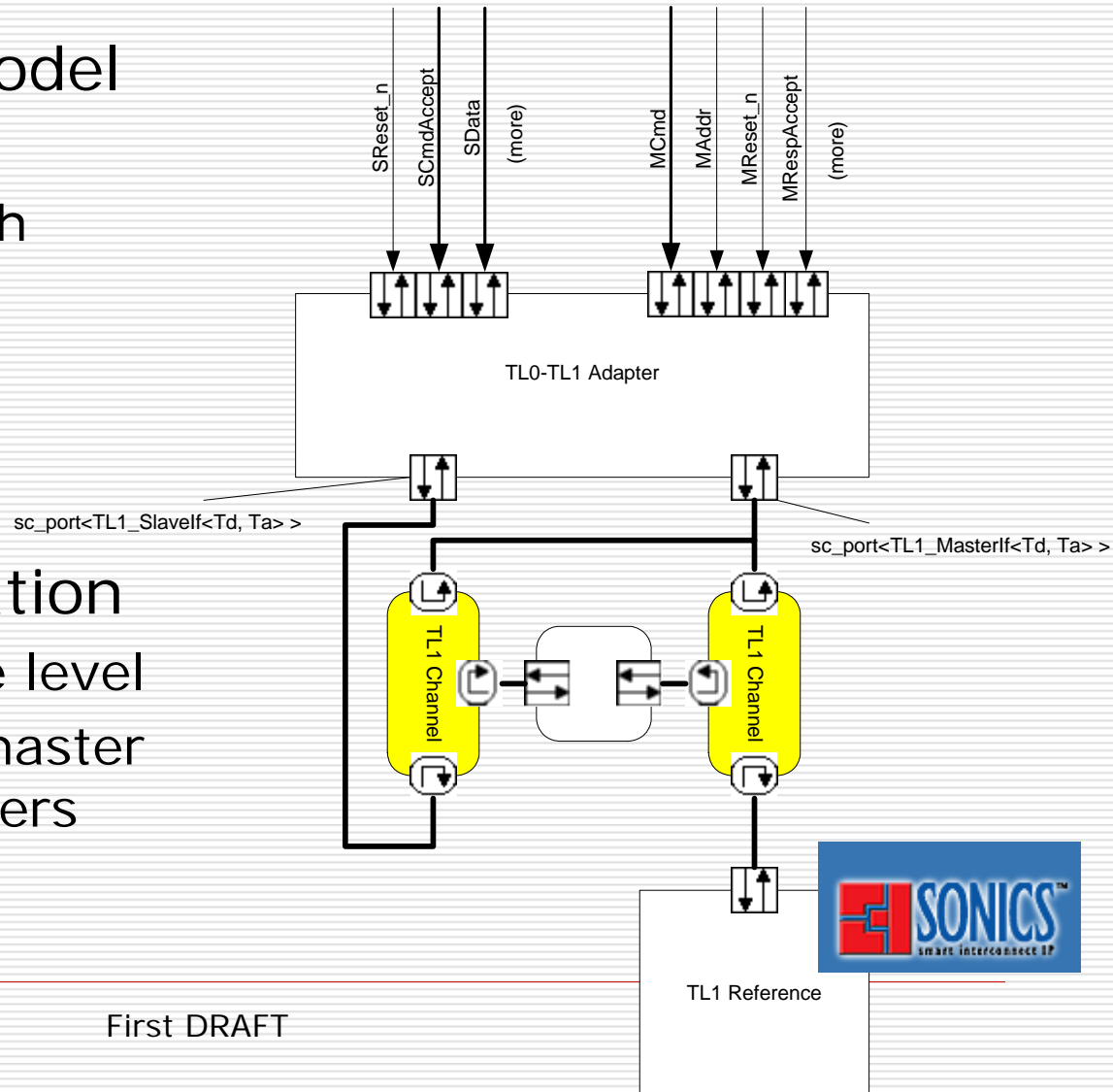
Sonics Reference Model Flow

□ CC Reference Model

- For each unit
- Instantiated with Verilog

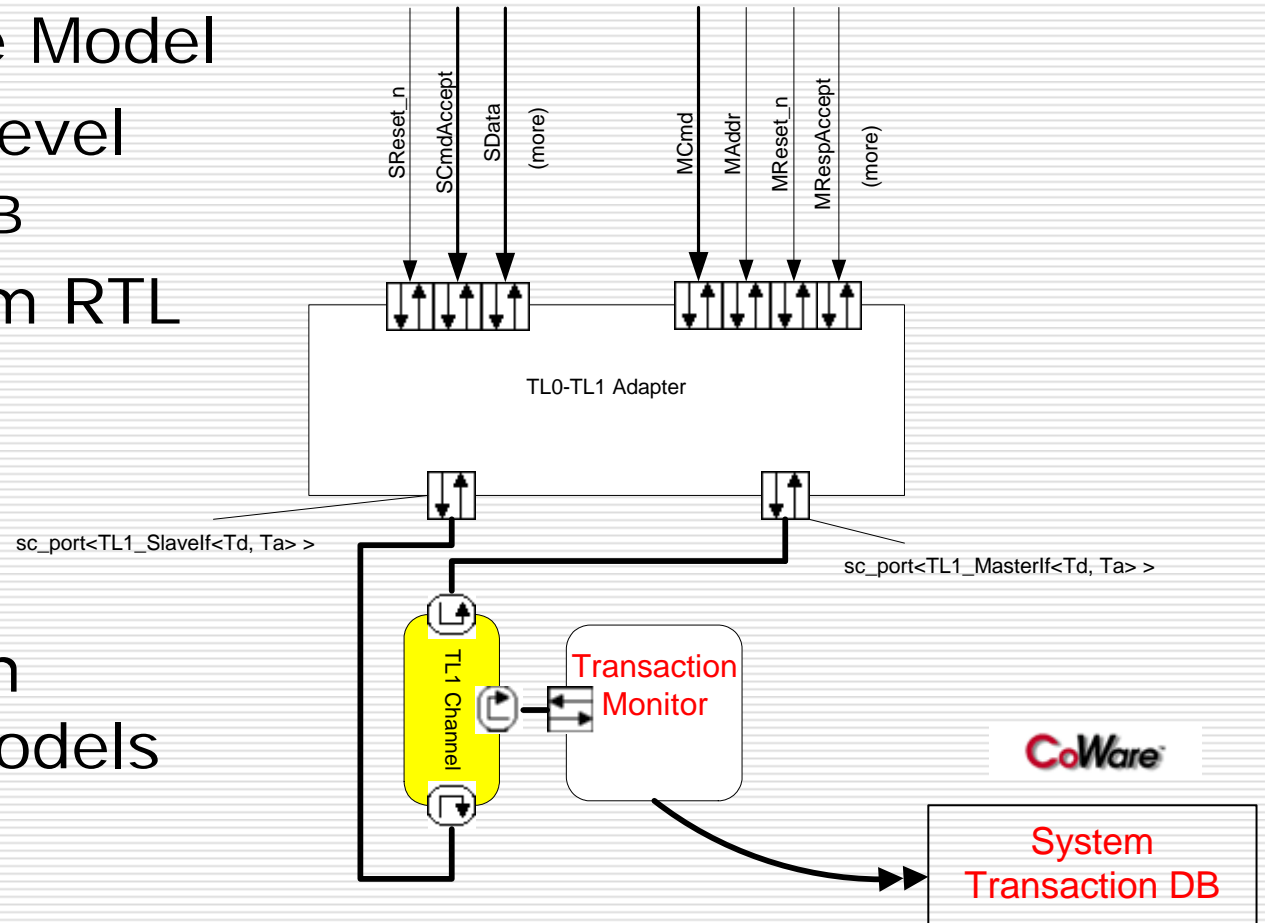
□ Runtime Correlation

- At cycle-callable level
- Requires both master and slave adapters



ESL Integration

- ❑ Drop Reference Model
- ❑ Monitor at CC-level
 - Into Coware DB
- ❑ ESL Access from RTL



- ❑ Correlation with Higher-Level Models