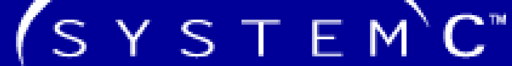


TLM and Verification

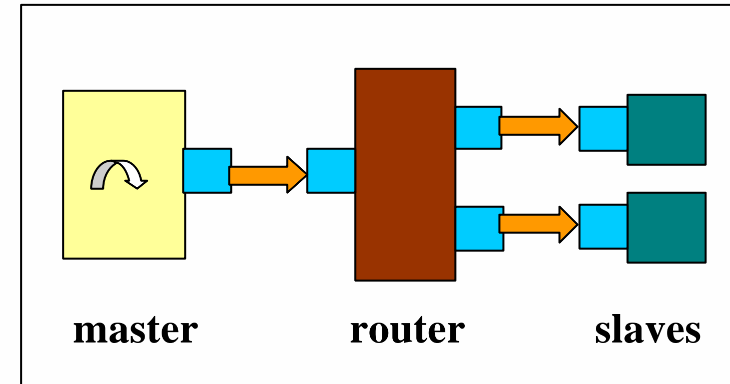
Adam Rose

March 2004

SystemC Transaction Level Modeling

- *What is TLM?*
  - Communication uses function calls

    burst_read(char* buf, int addr, int len);
- *Why is TLM interesting?*
  - Fast and compact
  - Integrate HW and SW models
  - Early platform for SW development
  - Early system exploration and verification
  - Verification reuse



SYSTEMC™

SystemC Transaction Level Modeling

- *How is TLM being adopted?*
  - Widely used for verification
  - TLM for design is starting at major electronics companies
- *Is it really worth the effort?*
  - Yes, particularly for platform-based design and verification
- *What will help proliferate TLM?*
  - Standard TLM APIs and guidelines
  - Availability of TLM platform IP
  - Tool support

➢ SystemC TLM Standard

SYSTEM C™

Endorsements of the OSCI TLM API

“We are excited about the TLM API proposal that is currently being reviewed by the OSCI TLM working group. This proposal satisfies the technical requirements of the TLM-API WG. We believe it can provide the standard foundation that enables transaction level SystemC IP to be developed and reused quickly and efficiently.”

- Adam Donlin, Xilinx
- Frank Ghenassia, ST Microelectronics, Chairman of OSCI TLM WG
- Serge Goossens, CoWare
- Anssi Haverinen, Nokia, Chairman of OCP-IP TLM Working Group
- Mike Meredith, Forte Design Systems
- Stuart Swan, Cadence Design Systems

SYSTEMC™

TLM API Goals

- Support design & verification IP reuse
- Provide common TLM recipe
- Usability
- Safety
- Speed
- Generality
  - Abstraction Levels
  - HW / SW
  - Different communication architectures (bus, packet, NOC, ...)
  - Different protocols

SYSTEMC™

Key Concepts

- Focus on SystemC interface classes
  - Define small set of generic, reusable TLM interfaces
  - Different components implement same interfaces
  - Same interface can be implemented
    - directly within a C/C++ function, or
    - via communication with other modules/channels in system
- Object passing semantics
  - Similar to sc_fifo, effectively pass-by-value
  - Avoids problems with raw C/C++ pointers
  - Leverage C++ smart pointers and containers where needed

Key Concepts (cont.)

- Unidirectional vs. bidirectional dataflow
  - Unidirectional interfaces are similar to sc_fifo
  - Bidirectional can be easily and cleanly layered on unidirectional
  - Separates requests from responses
- Blocking vs. nonblocking
- Use sc_port & sc_export

SYSTEMC™

Layered TLM API Architecture

User Layer Protocol-specific "convenience" API Targeted for embedded SW engineer Typically defined and supplied by IP vendors	amba_bus->burst_read(buf, adr, n);

Protocol Layer
Protocol-specific code
Adapts between user layer and transport layer
Typically defined and supplied by IP vendors
	req.addr = adr; req.num = n;
	rsp = transport(req);
	return rsp.buf;

Transport Layer Uses generic data transport APIs and models Facilitates interoperability of models Key focus of TLM standard May use generic fifos, arbiters, routers, xbars, pipelines, etc.	sc_port<tlm_transport_if<REQ, RSP> > p;

SYSTEMC™

Unidirectional TLM Interfaces

Blocking Put Interface

```
template < typename T >
class tlm_blocking_put_if : public virtual sc_interface
{
public:
  virtual void put( const T &t ) = 0;
};
```

Nonblocking Put Interface

```
template < typename T >
class tlm_nonblocking_put_if : public virtual sc_interface
{
public:
  virtual bool nb_put( const T &t ) = 0;
  virtual bool nb_can_put() const = 0;
  virtual const sc_event &ok_to_put() const = 0;
};
```

Blocking Get Interface

```
template < typename T >
class tlm_blocking_get_if : public virtual sc_interface
{
public:
  virtual T get() = 0;
  virtual void get( T &t ) { t = get(); }

};
```

Nonblocking Get Interface

```
template < typename T >
class tlm_nonblocking_get_if : public virtual sc_interface
{
public:
  virtual bool nb_get( T &t ) = 0;
  virtual bool nb_can_get() const = 0;
  virtual const sc_event &ok_to_get() const = 0;
};
```

Unidirectional TLM Interfaces

Blocking Put Interface

```
tlm_port< tlm_put_if< ethernet_packet > > p;

ethernet_packet e;

p->put( e );
```

Nonblocking Put Interface

```
tlm_port< tlm_put_if< ethernet_packet > > p;

ethernet_packet e;

if( p->nb_can_put() )
  p->nb_put( e );
```

Blocking Get Interface

```
tlm_port< tlm_get_if< ethernet_packet > > p;

ethernet_packet e = p->get();
```

Nonblocking Get Interface

```
tlm_nonblocking_get_port< ethernet_packet > p;

SC_METHOD( get_packet );
sensitive << p.ok_to_get();

void get_packet() {
  ethernet_packet e;
  p->nb_get( e );
  ...
}
```

Bidirectional TLM Interface

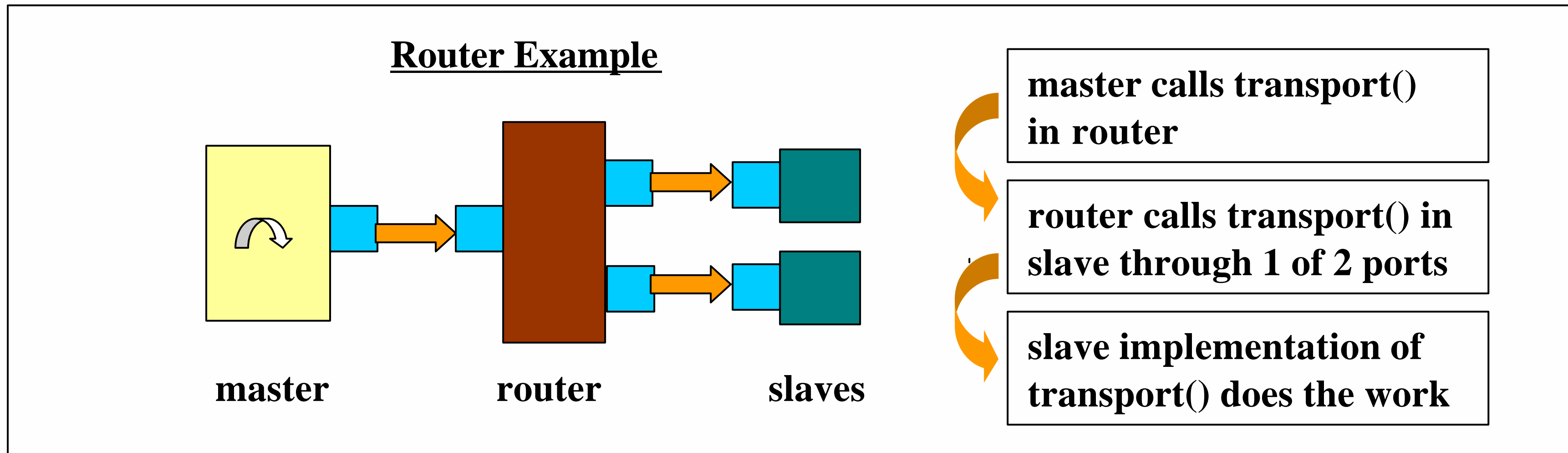
Bidirectional Blocking Interface

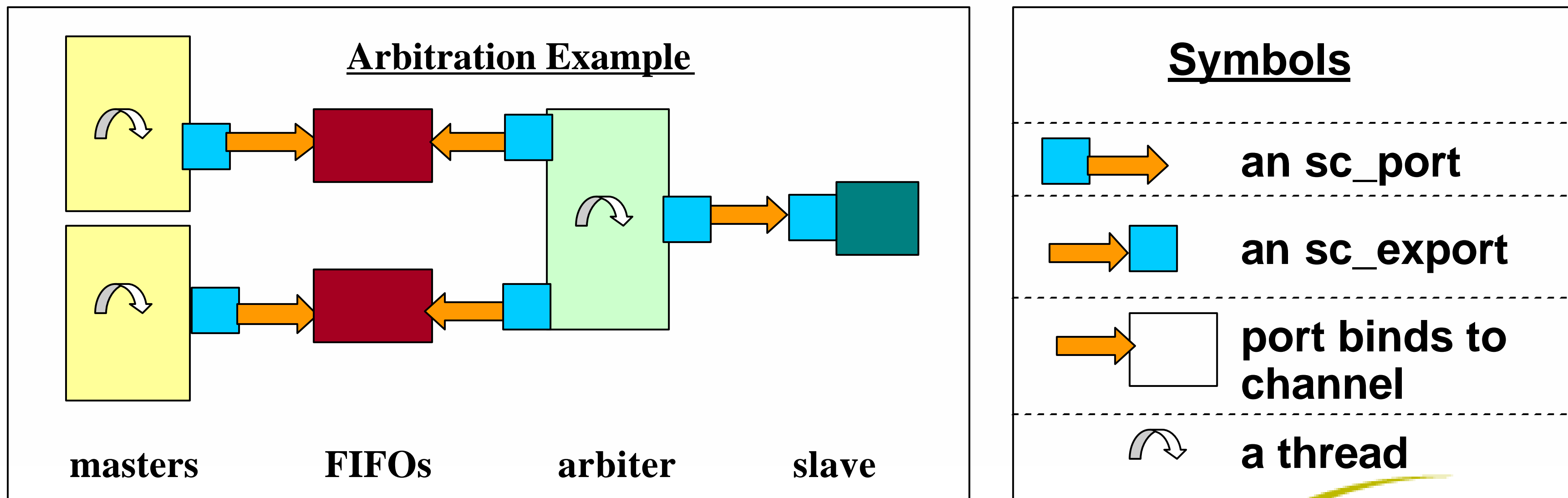
```
template < typename REQ , typename RSP >
class tlm_transport_if : public virtual sc_interface
{
public:
  virtual RSP transport( const REQ & ) = 0;

};
```

```
sc_port < tlm_transport_if < AMBA_REQ , AMB_RSP > > p;

AMBA_REQ req( READ , 0x1000 );
AMBA_RSP rsp;

rsp = p- >transport( req );
```


SYSTEMC™

Transaction Level Modeling with the TLM API

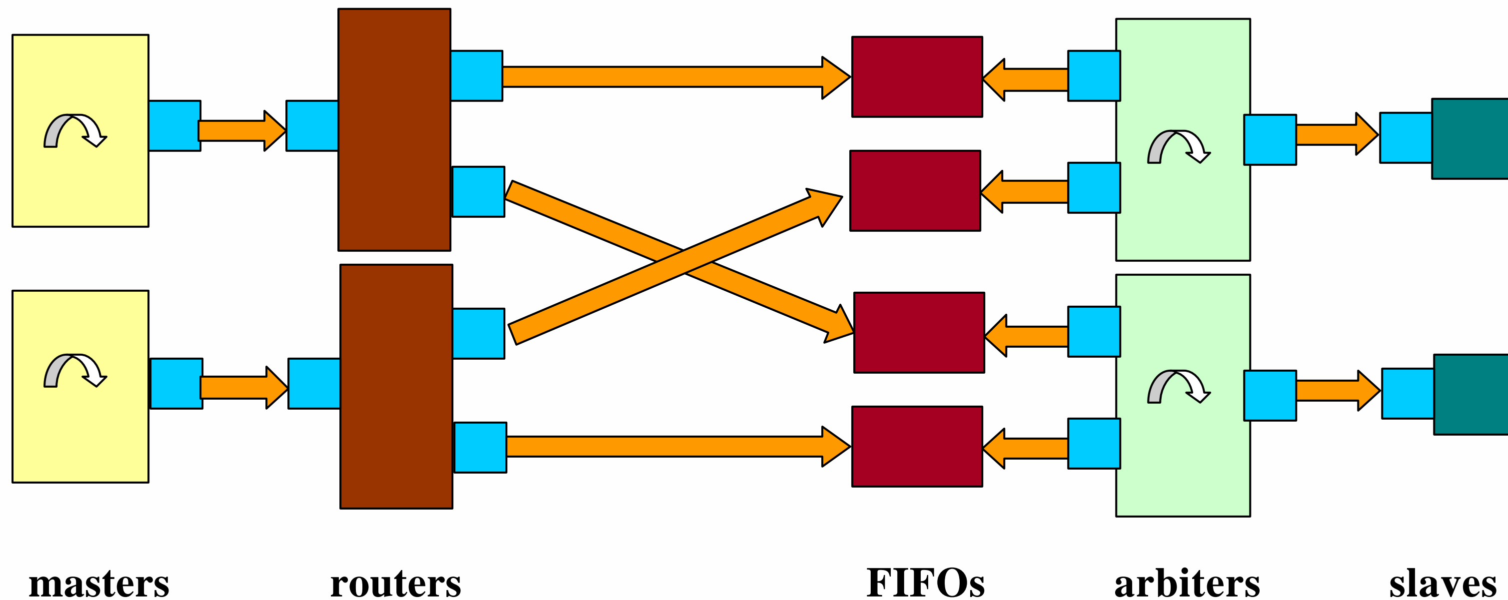




SYSTEMC™

Transaction Level Modeling – Cross Bar

- Uses the same components on the previous slide connected in different ways



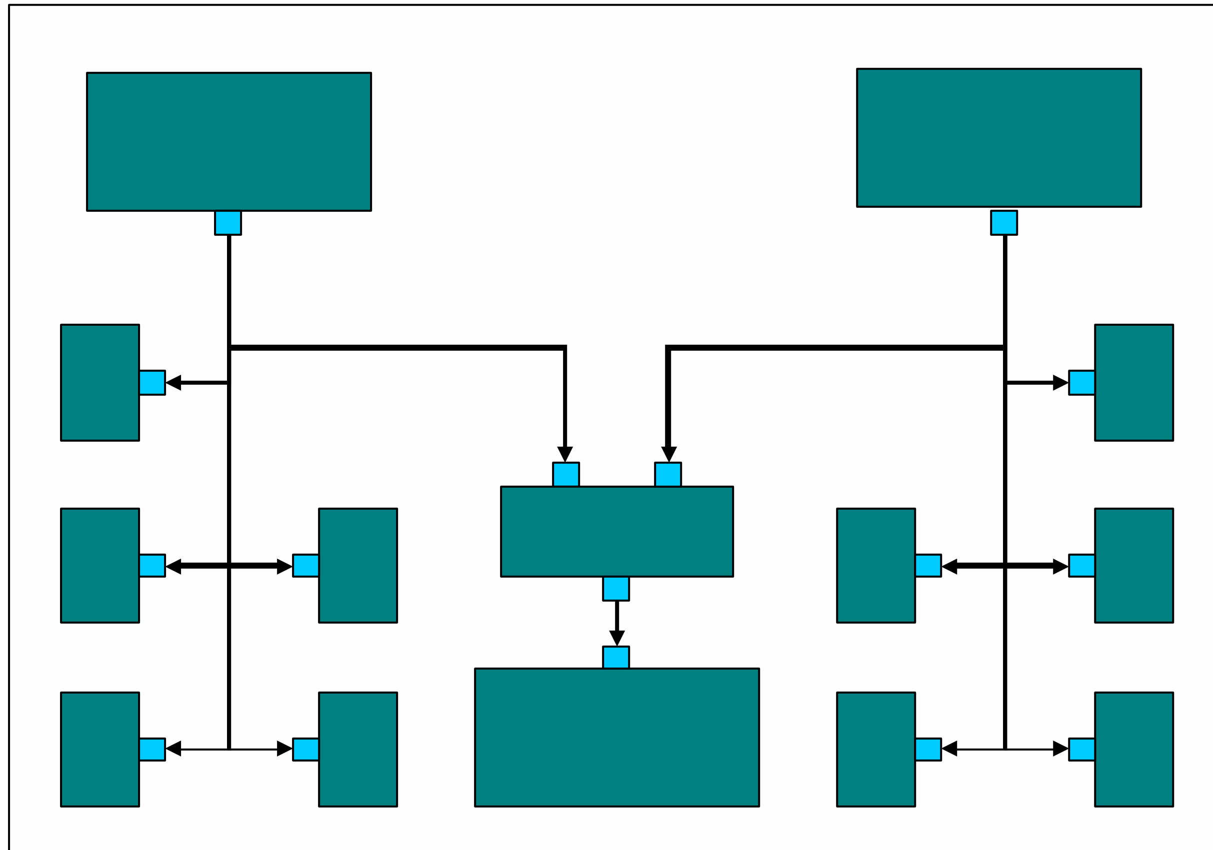
Cross Bar Switch

SYSTEMC™

Transaction Level Modeling and Verification

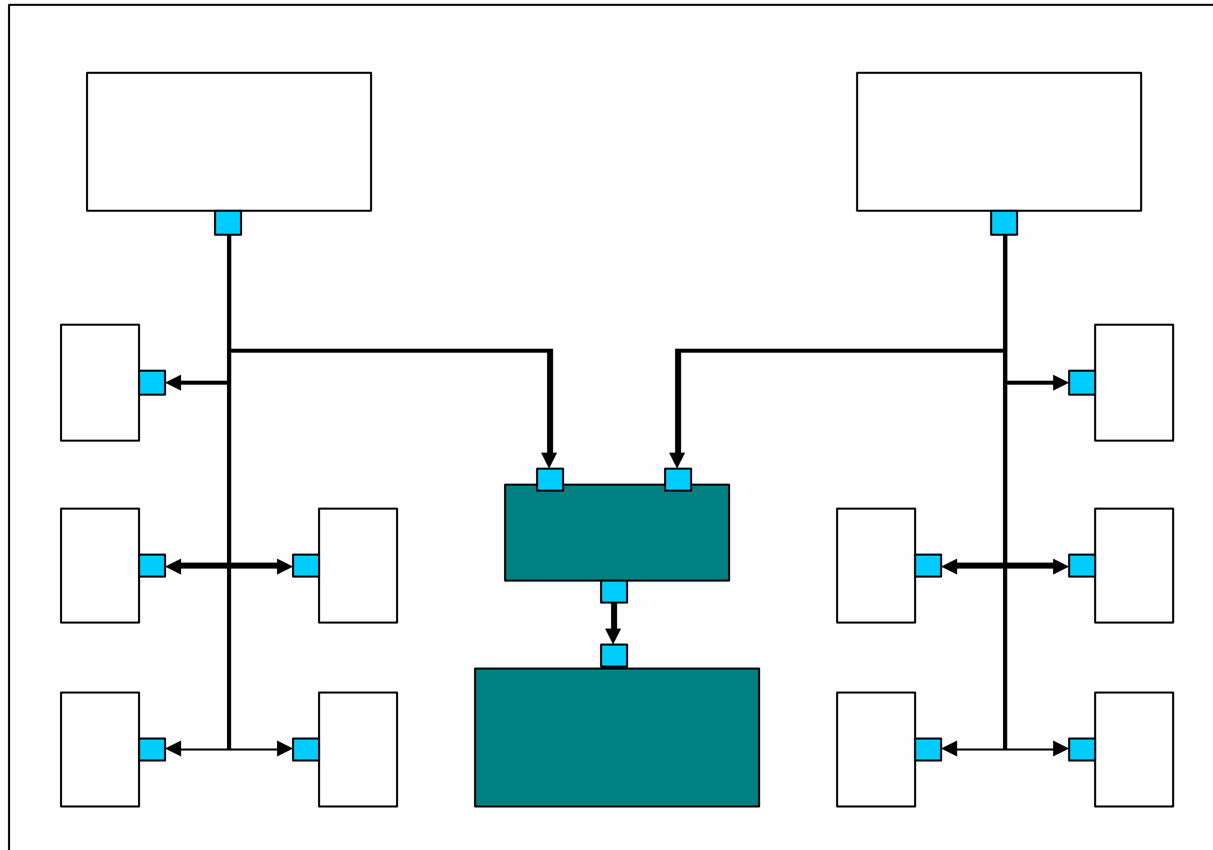
- Verification is best done at the transaction level
  - Enables reuse between block level and system level tests, and between simulation and emulation.
- TLMs are used to increase execution speed of DUT
- A testbench is a TLM of the environment of the DUT
- TLMs are used as golden models to predict behaviour of DUT

TLM used to increase execution speed



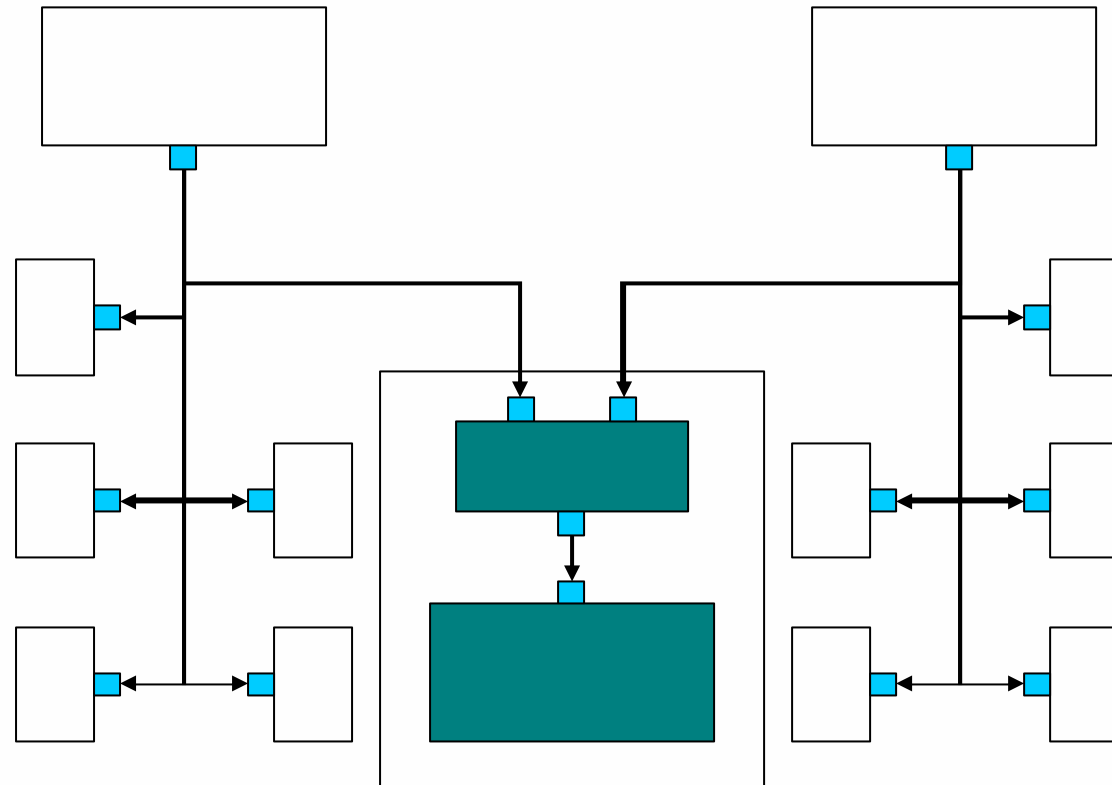
SYSTEM C™

TLM used to increase execution speed



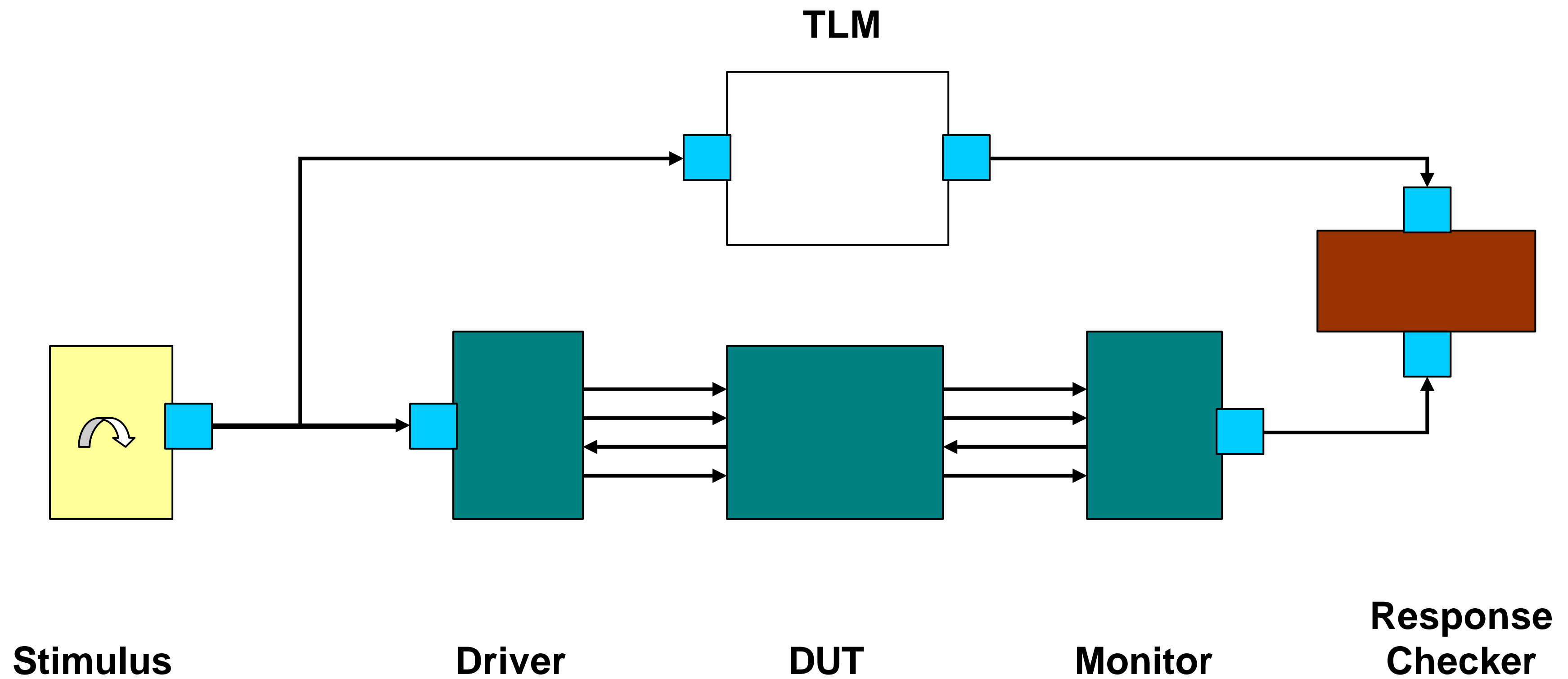
SYSTEMC™

TLM used to model the DUT's environment



SYSTEMC™

TLM as a Golden Model



SYSTEMC™

SCV, TLM and Verification

- TLM and SCV are complementary
  - The TLM API provides a standardized communication mechanism for both TLM and verification
  - The fifo based unidirectional interfaces in particular are very well suited to verification
  - SCV provides constrained randomization and transaction recording
- The most interesting new developments in OSCI, ESL EDA, and SystemC Users are on the boundary between TLM and Verification
- The OSCI TLM is available for public review
  - Download www.systemc.org