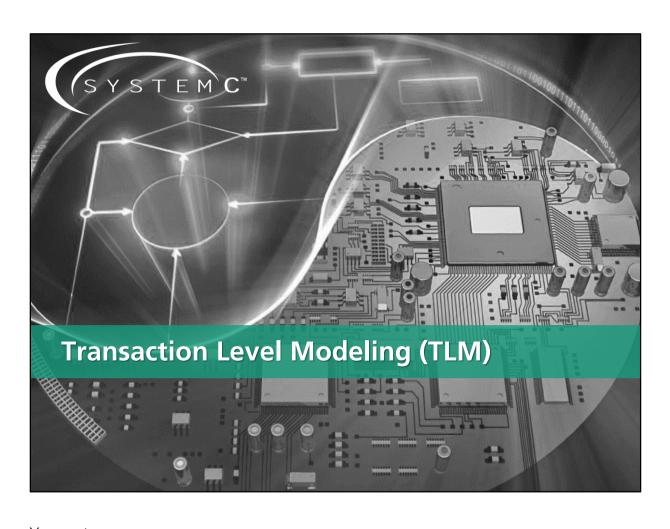
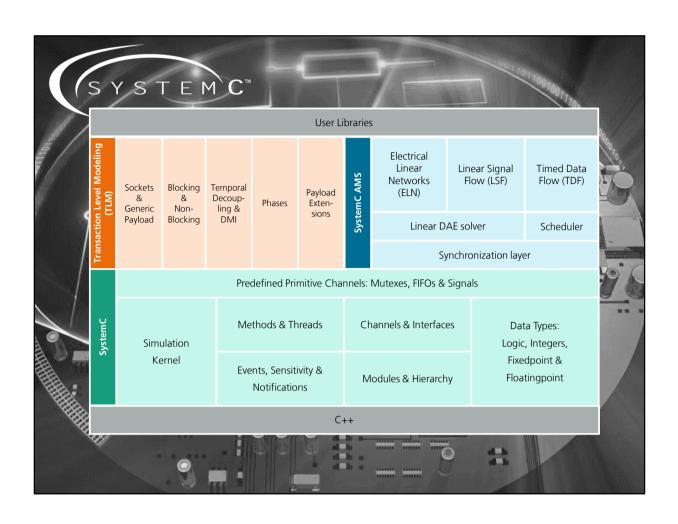


our notes:	



Your notes:			
_			



Your notes:	

	1			પ C™		User Li	ibraries			11/1/1
	Transaction Level Modeling (TLM)	Sockets & Generic	Blocking & Non-	Temporal Decoup- ling &	Phases	Payload Exten-	SystemC AMS	Electrical Linear Networks (ELN)	Linear Signal Flow (LSF)	Timed Data Flow (TDF)
1	saction	Payload	Blocking	DMI		sions	Syste	Linear D	AE solver	Scheduler
	Tran							Sy	nchronization lay	er
	Predefined Primitive Channels: Mutexes, FIFOs & Signals									
	SystemC	Simı	ulation	Me	thods & Th	nreads	Ch	annels & Interfac		ta Types: ., Integers,
	ν.	Kε	ernel		nts, Sensit Notificatio		М	odules & Hierarch		dpoint & tingpoint
						C	++			

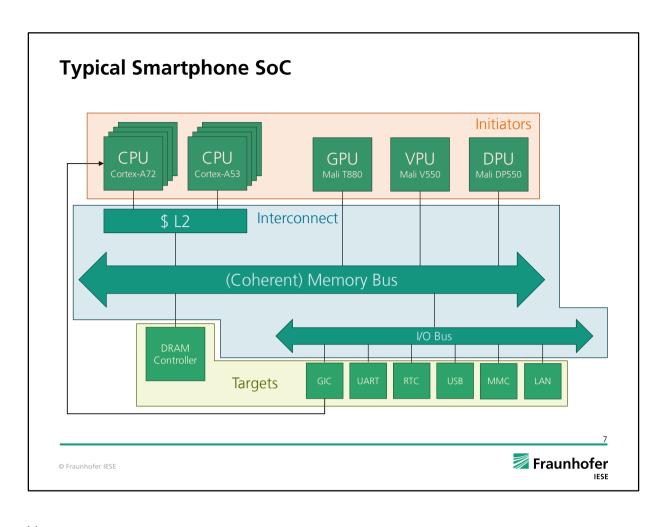
Your notes:	

How to build a virtual Smartphone? ■ CoWare Virtual AT Keyboard, PS/2 Mouse/Touch Linux Connectivity Do File Simulation Debug Analysis View Windows 📶 🔲 1:13 PM Item Implication of the property of the prope Address : Core : [00000000] e1a00000 [00000004] e1a00000 [00000008] e1a00000 [00000000] e1a00000 [00000010] e1a00000 [00000014] e1a00000 0:00:15.605 017 430 0 Messaging Opera Mini Contacts Browser Google Ma 0 - Un-compressed kernel @0x30000000 - Relocate pre-mmu symbols by @0xc0000000 Please wait while preparing symbols... Simulation has been resetted. Please run the simulation now via 'F5'. MENU × Views Overview Sisassembly 6 6

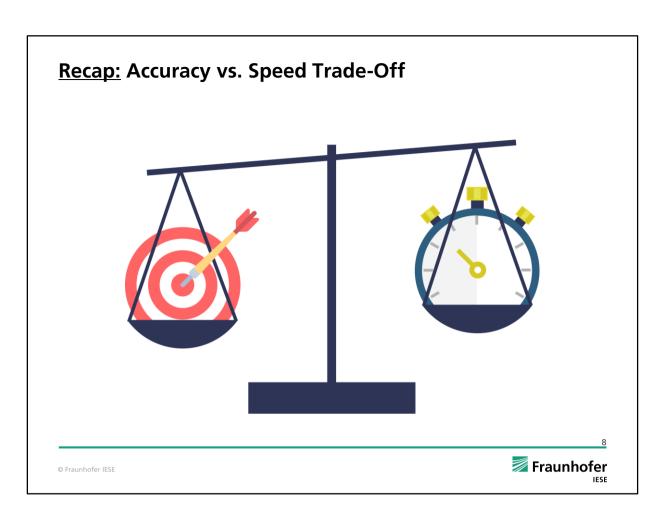
Your notes:		

© Fraunhofer IESE

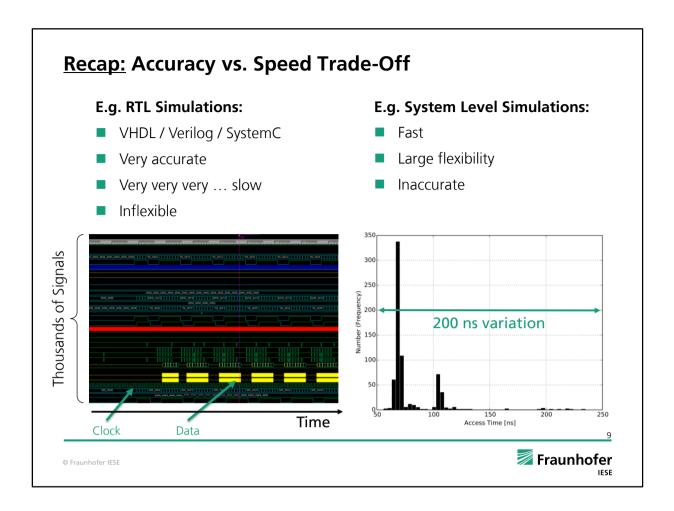
Fraunhofer



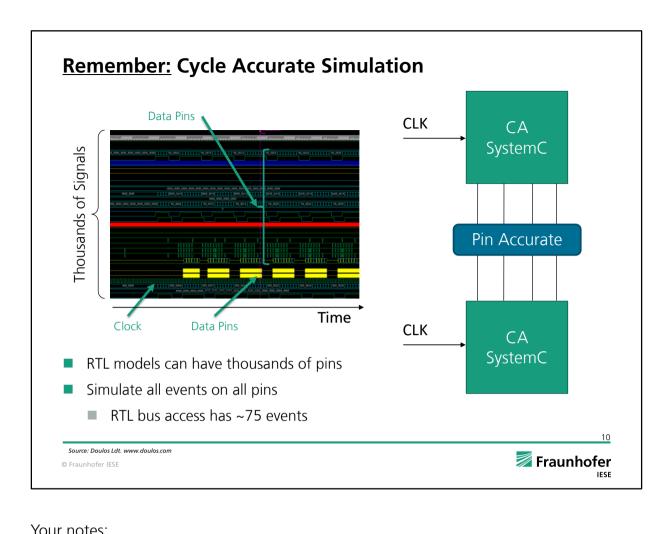
Your notes:	



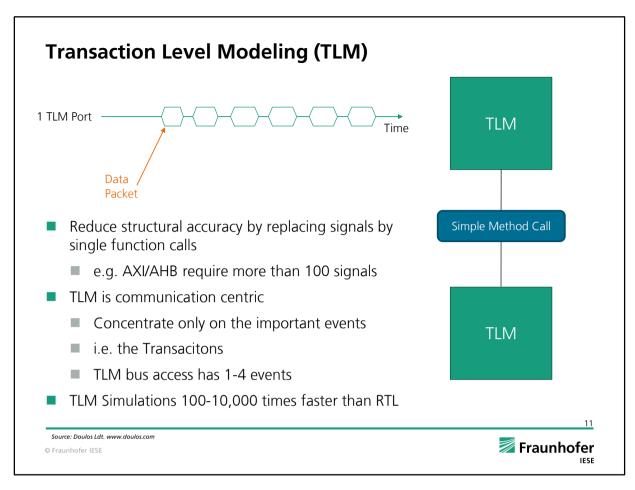
Your notes:			



Your notes:	



Tour Hotes.	



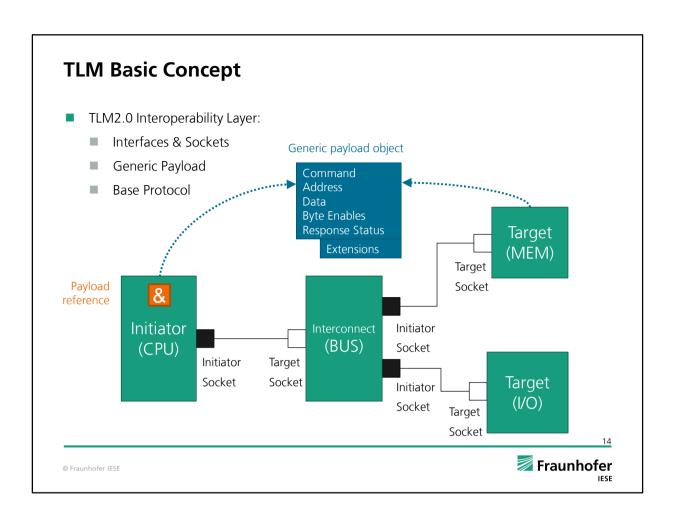
Transaction level modeling is speeding up simulation by replacing all pin-level events with a single method call. In an RTL simulation the communication requires multiple events at the level of individual bits. In a transaction-level model, complete bus cycles can be modeled with one method call. Therefore, TLM models can be 100 to 10000 times faster than their RTL counter parts. However, there is no free lunch: the increased speedup of simulation is payed with reduced accuracy of the results.

Your notes:			

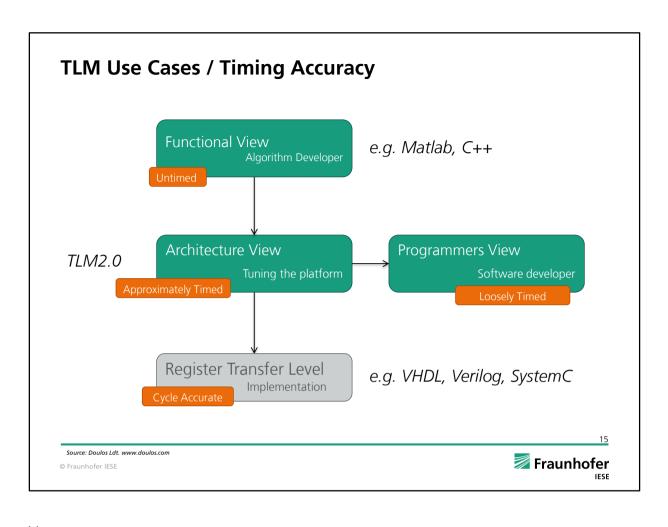
Transaction: A custom TLM Implementation

```
#include <iostream>
#include <systemc.h>
                                                          Try code on github:
#include <queue>
                                                  https://github.com/TUK-
SCVP/SCVP.artifacts/tree/master/custom_tlm
using namespace std;
enum cmd {READ, WRITE};
struct transaction {
    unsigned int data;
     unsigned int addr;
                                                                         Consumer Module as
     cmd command;
                                                                         Hierarchical Channel
};
class transactionInterface : public sc_interface {
     public:
     virtual void transport(transaction &trans) = 0;
                                                                            Fraunhofer
© Fraunhofer IESE
```


```
Transaction: A custom TLM Implementation
   SC_MODULE(PRODUCER)
                                                         sc_port< transactionInterface > master;
       SC_CTOR(PRODUCER)
                                                              private:
                                                              unsigned int memory[1024];
           SC_THREAD(process);
                                                              public:
                                                              SC_CTOR(CONSUMER) {
                                                                  for(unsigned int i=0; i < 1024; i++) {
    memory[i] = 0; // Initialize memory</pre>
       void process()
           for(unsigned int i=0; i < 4; i++) {</pre>
               wait(1,SC_NS);
               transaction trans;
               trans.addr = i;
trans.data = rand();
                                                              void transport(transaction &trans) {
Write
                                                                  if(trans.command == cmd::WRITE) {
                                                                      memory[trans.addr] = trans.data;
               trans.command = cmd::WRITE;
               master->transport(trans);
                                                                  else /* cmd::READ */ {
                                                                      trans.data = memory[trans.addr];
           for(unsigned int i=0; i < 4; i++) {</pre>
               wait(1,SC_NS);
                                                              }
                                                         };
               transaction trans;
               trans.addr = i;
trans.data = 0;
                                                         int sc_main(...) {
    PRODUCER cpu("cpu");
    CONSUMER mem("memory");
Read
               trans.command = cmd::READ;
                                                                                               The Produce is
               master->transport(trans);
                                                                                               active, the
                                                              cpu.master.bind(mem);
               cout << trans.data << endl;</pre>
                                                                                               consumer is passive
                                                              sc_start();
                                                              return 0;
                                                         }
                                                                                                   Fraunhofer
```



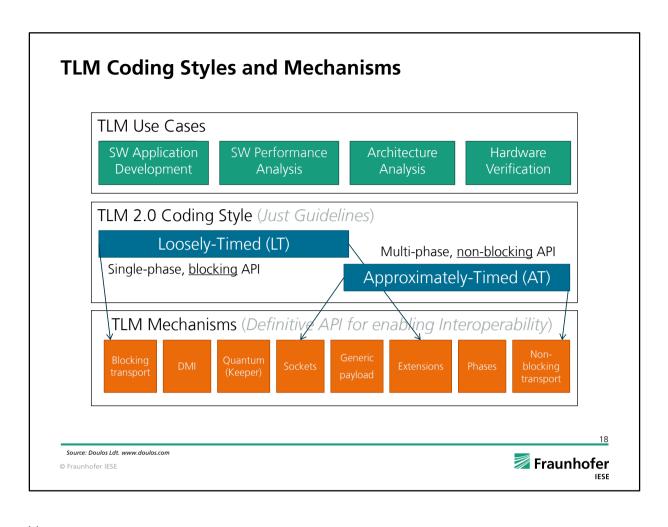
Your notes:			
_			



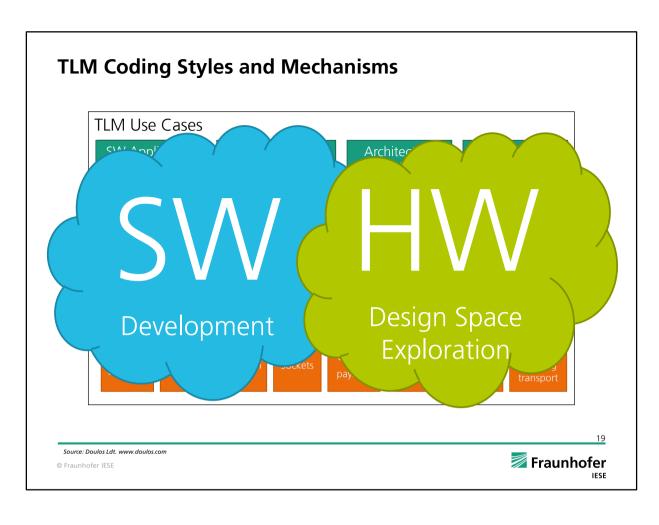
Your notes:			

			User Li	braries			\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
Transaction Level Modeling (TLM)	Sockets Blocking & & Generic Non-	Temporal Decoup- Ph ling &	Payload ases Exten-	SystemC AMS	Electrical Linear Networks (ELN)	Linear Signal Flow (LSF)	Timed Data Flow (TDF)
saction (7	Payload Blocking	DMI	sions	Syste	Linear D.	AE solver	Scheduler
Tran					Sy	nchronization lay	er
		Predefin	ed Primitive Chai	nnels:	Mutexes, FIFOs &	Signals	
SystemC	Simulation	Method	s & Threads	Ch	annels & Interface	-]	ta Types: c, Integers,
Š	Kernel		Events, Sensitivity & Notifications		odules & Hierarch		edpoint & atingpoint
			C-	++			

Your notes:		



Your notes:			



Your notes:			

Coding Styles in TLM

- Loosely-Timed (LT):
- SW
- As fast as possible
- Sufficient timing detail to boot OS and run multicore systems and to develop SW or drivers
- Processes can run ahead of simulation time (temporal decupling)
- Each transaction completes in one <u>blocking</u> function call
- Usage of Direct Memory Interface (DMI) e.g. for boot process

Approximately-Timed (AT):



- Accurate enough for performance modelling
- Sufficient for architectural HW design space exploration
- Processes run in lockstep with simulation time
- Each transaction has usually 4 timing points i.e. 4 function calls (extensible if required, also less possible); non-blocking behavior
- More detailed than LT and therefore also slower than LT

20

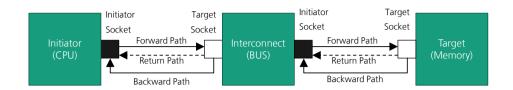


Your notes:	

			The section of the section is		User L	ibraries				<u> </u>
Transaction Level Modeling (TLM)	Sockets & Generic	Blocking & Non-	Temporal Decoup- ling &	Phases	Payload Exten-	SystemC AMS	Electrical Linear Networks (ELN)	Linear Si Flow (L	gnal SF)	Timed Data Flow (TDF)
saction (T	Payload	Blocking	DMI		sions	Syste	Linear D	AE solver		Scheduler
Trans							Sy	nchronizat	ion lay	er
			Pred	defined Pri	mitive Cha	nnels:	Mutexes, FIFOs &	Signals		
SystemC	Simı	ulation	Me	thods & Th	nreads	Cł	annels & Interfac	es		ta Types: c, Integers,
S	Kϵ	ernel		Events, Sensitivity & Notifications		Modules & Hierarchy				edpoint & atingpoint
					C-	++				

Your notes:			

Initiators, Targets and Interconnect



- TLM components are divided into Initiators, Targets and Interconnect components:
 - A initiator initiates (construct and send) new transactions
 - A target is a module that acts as the end point for a transaction. It executes requests from an initiator and send responses
 - An interconnect component forwards and routes transaction objects between initiators and targets
- Transactions are sent through initiator sockets (■) and received through target sockets (□)
- References to the object are passed along the forward and backward paths:
 - LT uses Forward and Return Path
 - AT uses Forward, Backward and Return Path



Your notes:			

Initiator and Target Sockets class tlm_fw_transport_if<...> Target Initiator Socket Socket virtual ... b_transport(...) = 0; virtual ... nb_transport_fw(...) = 0; (CPU) virtual ... get_direct_mem_ptr(...) = 0; virtual ... transport_dbg(...) = 0; }; class tlm_bw_transport_if<...> virtual ... nb_transport_bw(...) = 0; virtual ... invalidate_direct_mem_ptr(...) = 0; Target port must implement tlm_fw_transport_if methods Initiator must implement tlm_bw_transport_if methods b transport and mem ptr functions are used for LT modeling nb_transport functions are used for AT modeling

Your notes:			

© Fraunhofer IESE

Fraunhofer

			The section of the se		User L	ibraries				<u> </u>
Transaction Level Modeling (TLM)	Sockets & <u>Generic</u>	Blocking & Non-	Temporal Decoup- ling &	Phases	Payload Exten-	SystemC AMS	Electrical Linear Networks (ELN)	Linear : Flow	Signal (LSF)	Timed Data Flow (TDF)
saction (<u>Payload</u>	Blocking	DMI		sions	Syste	Linear D	AE solver		Scheduler
Trans							Sy	nchroniza	ation laye	er
			Pred	defined Pri	mitive Cha	nnels:	Mutexes, FIFOs &	Signals		
SystemC	Simı	ulation	Me	thods & Th	nreads	Cł	nannels & Interfac	es		ta Types: ., Integers,
Ś	Kε	ernel		nts, Sensit Notificatio		М	odules & Hierarch	ıy		dpoint & tingpoint
					C-	++				

Your notes:			

Generic Payload

- The generic payload is designed to include typical attributes of memory mapped busses (e.g. AXI, AHB, etc.)
- It can supports
 - READ and WRITE transactions
 - Byte enables
 - Single word transfer
 - Burst transfer
 - Streaming transfer
- Extensions can be used to carry further metadata or to model more complex bus and NoC protocols and maintain 100% compatibility because they are ignorable
- Very efficient implementation for simulation speed

27

© Fraunhofer IESE



Generic payload object

Response Status

Extensions

Command Address

Data <u>Byte</u> Enables

Your notes:		

Generic Payload Attributes

Attribute	Туре	Modifiable?
Command	tlm_command	No
Address	uint64_t	Interconnect Only
Data Pointer	unsinged char *	No (array yes)
Data Length	unsigned int	No
Byte Enable Pointer	unsigned char *	No
Byte Enable Length	unsinged int	No
Streaming Width	unsigned int	No
DMI Hint	bool	Yes
Response Status	tlm_response_status	Target Only
Extensions	<pre>(tlm_extension_base*)[]</pre>	Yes

- Initiator should initialize attributes before sending the transaction object
- Set-Methods like set_address() etc. are provided
- The majority of the attributes must not be changed by Interconnects & Targets



Your notes:	

					User L	braries			- 7- 6 YEO O VURN (1890)	X 1/1/1
Transaction Level Modeling (TLM)	Sockets & Generic	Blocking & Non-	Temporal Decoup- ling &	Phases	Payload Exten-	SystemC AMS	Electrical Linear Networks (ELN)	Linea Flov	r Signal v (LSF)	Timed Data Flow (TDF)
saction (1	Payload	Blocking	DMI		sions	Syste	Linear D	AE solve	er	Scheduler
Trans							Sy	/nchroni	ization laye	er
			Pred	defined Pri	mitive Cha	nnels:	Mutexes, FIFOs &	Signals		
SystemC	Simi	ulation	Me	thods & Th	nreads	Cł	annels & Interface	es		ta Types: r, Integers,
Š	Kϵ	ernel		nts, Sensit Notificatio		М	odules & Hierarch	ny		dpoint & tingpoint
					C-	++				

Your notes:		

```
SW
Building a Blocking (LT) Initiator
class Initiator: sc_module, tlm::tlm_bw_transport_if<> {
    // Dummy method:
                                                                             void invalidate_direct_mem_ptr(
                                                                                 sc_dt::uint64 start_range,
                                                                                  sc_dt::uint64 end_range)
                                                                                                                 Must be
                                                                                                                 implemented
                                                                             void process() {
    for (int i = 0; i < 4; i++) {</pre>
                                              Write to memory
             tlm::tlm_generic_payload trans;
                                                                                      tlm::tlm_phase& phase,
             unsigned char data = rand();
                                                                                      sc_time& delay)
             trans.set_address(i);
             trans.set_data_length(1);
             trans.set_data_length(1);
trans.set_command(tlm::TLM_WRITE_COMMAND);
trans.set_data_ptr(&data);
                                                                                  return tlm::TLM ACCEPTED;
             sc_time delay = sc_time(0, SC_NS);
iSocket->b_transport(trans, delay);
                                                                         };
             wait(delay);
                                             Read from memory
         for (int i = 0; i < 4; i++) {
             tlm::tlm_generic_payload trans;
             unsigned char data;
             trans.set_address(i);
             trans.set_data_length(1);
trans.set_command(tlm::TLM_READ_COMMAND);
             trans.set_data_ptr(&data);
             sc_time delay = sc_time(0, SC_NS);
iSocket->b_transport(trans, delay);
                                                                                             Try code on github:
             wait(delay);
                                                                               https://github.com/TUK-
SCVP/SCVP.artifacts/tree/master/tlm_lt_initiator_target
    }
                                                                                                           Fraunhofer
© Fraunhofer IESE
```


Building a Blocking (LT) Target class Target:sc_module, tlm::tlm_fw_transport_if<> { private: unsigned char mem[1024]; // Dummy method virtual tlm::tlm_sync_enum nb_transport_fw(tlm::tlm_generic_payload& trans, tlm::tlm_target_socket<> tSocket; tlm::tlm_phase& phase, sc_time& delay) Must be SC_CTOR(Target) : tSocket("tSocket") { tSocket.bind(*this); return tlm::TLM_ACCEPTED; implemented // Dummy method bool get_direct_mem_ptr(void b_transport(tlm::tlm_generic_payload &trans, sc_time &delay) tlm::tlm_generic_payload& trans, tlm::tlm_dmi& dmi_data) if(trans.get_address() >= 1024){ SC_REPORT_FATAL(this->name(),"Out of Range"); if(trans.get_command() == tlm::TLM_WRITE_COMMAND) // Dummy method unsigned int transport_dbg(memcpy(mem+trans.get_address(), // destination tlm::tlm_generic_payload& trans) trans.get_data_ptr(), trans.get_data_length()); // size } else { return 0; }; delay = delay + sc_time(40, SC_NS); Fraunhofer

Binding Target and Initiator



```
int sc_main (...)
{
    Initiator * cpu = new Initiator("cpu");
    Target * memory = new Target("memory");
    cpu->iSocket.bind(memory->tSocket);
    sc_start();
    return 0;
}
```

Summary:

- Initiator and target ports are derived from sc_port and sc_export
- b_transport uses call by reference to transfer the transaction object (from tlm_generic_payload class)
- The key idea of timing annotation is that the recipient is obliged to behave as if it had received the transaction at time sc_time_stamp() + delay
- All virtual methods must be implemented
- Transaction objects should be reused to avoid always new allocations

Try code on github:

https://github.com/TUKSCVP/SCVP.artifacts/tree/master/tlm_lt_initiator_target

33



Your notes:			

Error Handling for b_transport

enum tlm_response_status	Meaning
TLM_OK_RESPONSE	Successful transmission
TLM_INCOMPLETE_RESPONSE	Transaction not delivered to the target (default)
TLM_ADDRESS_ERROR_RESPONSE	Unable to work with given address
TLM_COMMAND_ERROR_RESPONSE	Unable to execute command (e.g write to ROM)
TLM_BURST_ERROR_RESPONSE	Unable to work with given datalength
TLM_BYTE_ENABLE_ERROR_RESPONSE	Unabel to work with byte enable
TLM_GENERIC_ERROR_RESPONSE	Any other error

- Initiator should set response status to TLM_INCLOMPLETE_RESPONSE (default)
- Targets modify the response status
- Initiator checks status of transaction when it is completed (e.g. after b_transport)

34



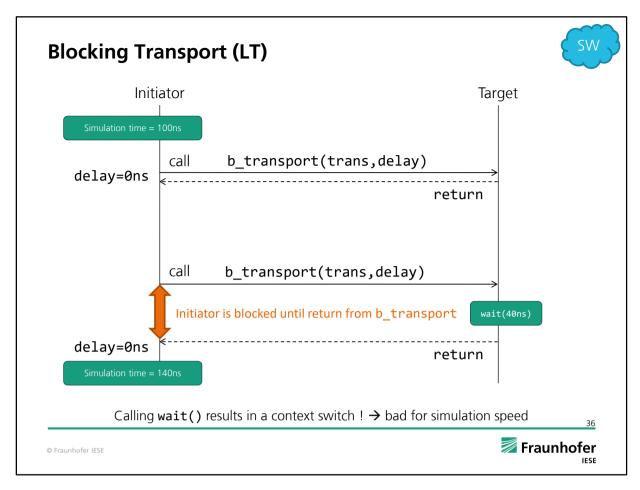
Error Handling for b_transport

```
SW
```

```
void b transport (... &trans, ... &delay)
      if (trans.get_address() >= 512) {
          trans.set_response_status(
             tlm::TLM_ADDRESS_ERROR_RESPONSE );
      if (trans.get_data_length() != 4) {
          trans.set_response_status(
             tlm::TLM_BURST_ERROR_RESPONSE );
           return:
      if (byt) {
          trans.set response status (
             tlm::TLM BYTE ENABLE ERROR RESPONSE );
      if(trans.get_command() == tlm::TLM_WRITE_COMMAND){
        memcpy(...);
      else (
        memcpy(...);
     delay = delay + sc time(40, SC NS);
      trans.set response status ( tlm::TLM OK RESPONSE );
};
```

Fraunhofer

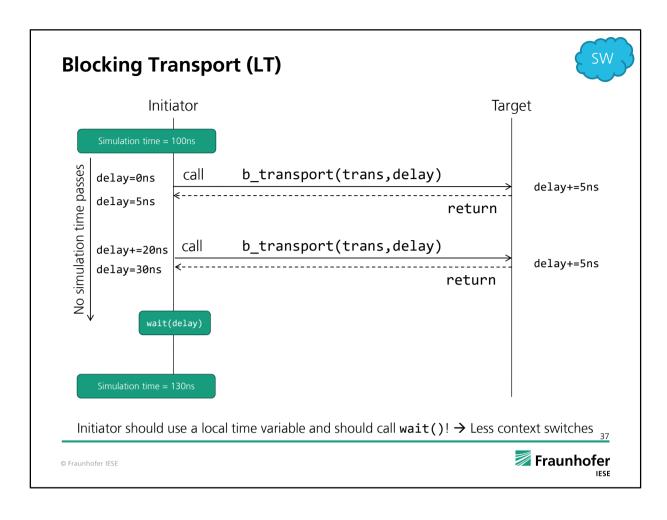
Your notes:



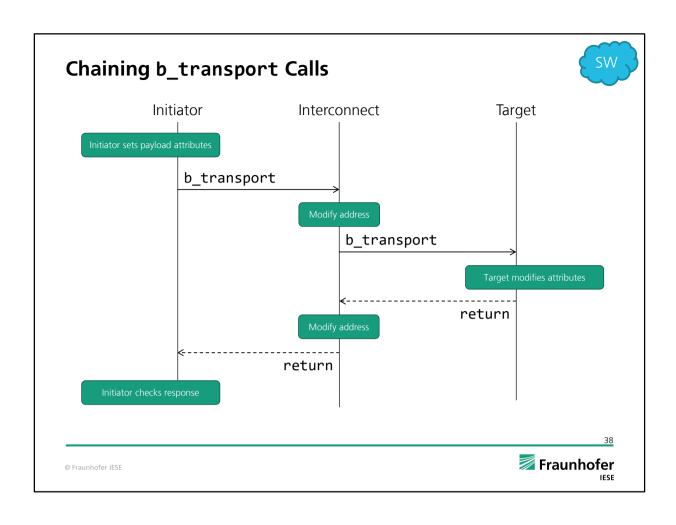
An initiator, and only an initiator, may call **b_transport**. The target can return immediately, or can suspend (by calling **wait**) and return later. However, calling **wait** statements in targets should be avoided because a wait statement will result in a context switch of the simulator, which decreases the simulation speed.

The same transaction object could be reused from one call to the next, but the two calls would count as separate transactions.

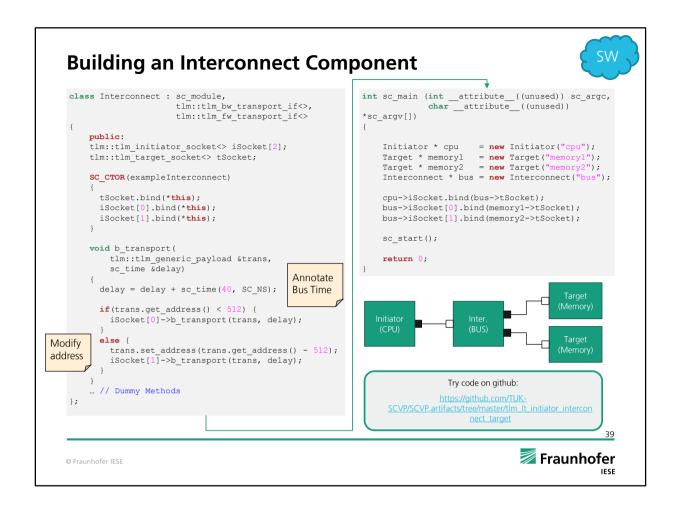
Your notes:			



Your notes:			



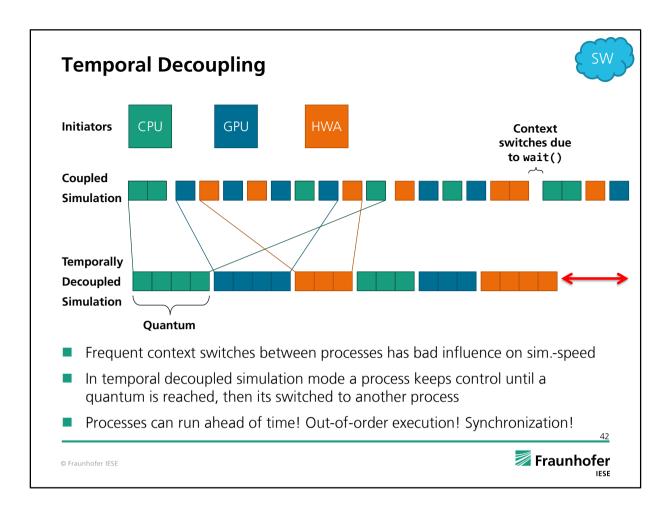
Your notes:	



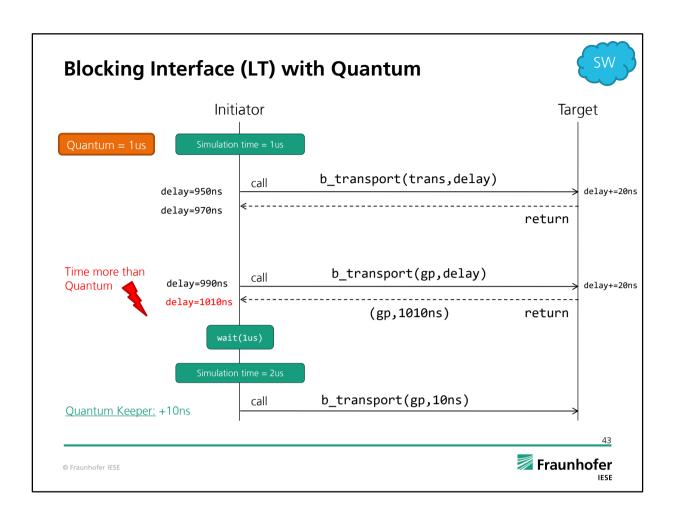
Your notes:	

					User Li	braries	- Annual Control of the Control of t	va vy valendi.		\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
Transaction Level Modeling (TLM)	Sockets & Generic	Blocking & Non-	Temporal Decoup- ling &	Phases	Payload Exten-	SystemC AMS	Electrical Linear Networks (ELN)	Line Flo	ar Signal w (LSF)	Timed Data Flow (TDF)
saction (1	Payload Blocking	DMI		sions	Syste sions	Linear D	DAE solver		Scheduler	
Trans							Sy	nchror	nization laye	er
			Predefined Primitive Channels: Mutexes, FIFOs & Signals							
SystemC	Methods & Threa					eads Channels & Interfaces			Data Types: Logic, Integers,	
	Kε	ernel		Events, Sensitivity & Notifications						dpoint & tingpoint
					C-	++				

Your notes:			

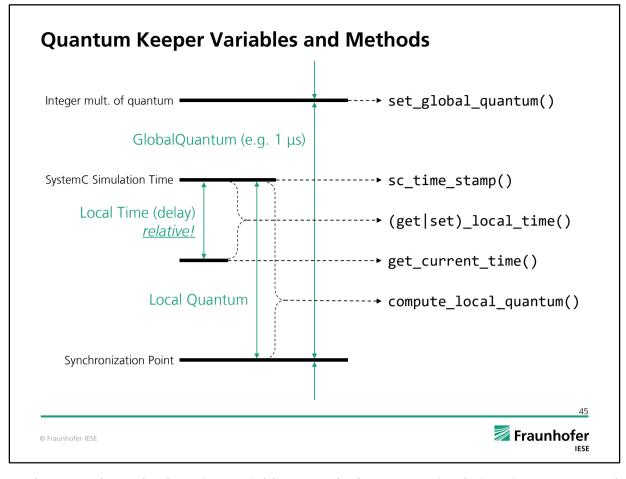


Your notes:			



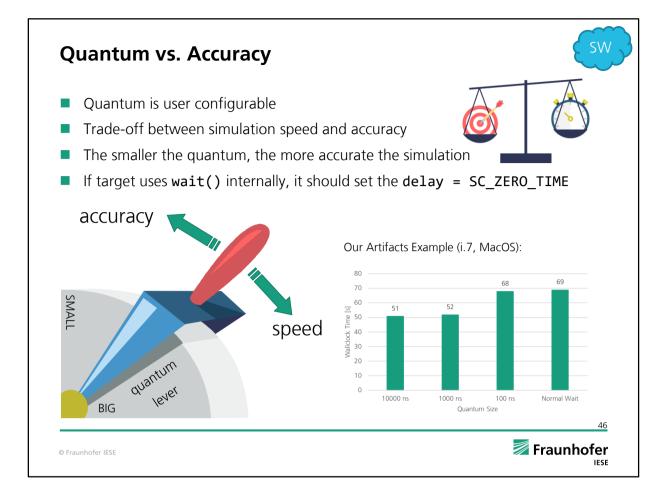
Your notes:			

```
SW
Quantum Keeper
iSocket->b_transport(trans, delay);
// Anotate the time of the target
                                                                                  quantumKeeper.set(delay);
   tlm_utils::tlm_quantumkeeper quantumKeeper;
                                                                                  // Consume internal computation time
                                                                                  quantumKeeper.inc(sc time(10,SC NS));
   tlm::tlm_initiator_socket<> iSocket;
SC_CTOR(exampleInitiator) : iSocket("iSocket")
                                                                                  if(quantumKeeper.need_sync())
     iSocket.bind(*this);
SC_THREAD(process);
                                                                                    quantumKeeper.sync();
                                                                                                                      Calls wait()
                                                                                                                     internally
     quantumKeeper.set_global_quantum(
   sc_time(1,SC_US)
                                              Static Method
     quantumKeeper.reset();
                                                                             // Dummy methods ...
   void process()
     // Write to memory:
for (int i = 0; i < 1024; i++) {
       tlm::tlm_generic_payload trans;
       unsigned char data = rand();
trans.set_address(i);
       trans.set_data_length(1);
trans.set_command(tlm::TLM_WRITE_COMMAND);
trans.set_data_ptr(&data);
                                                                                                Try code on github:
       sc time delay = quantumKeeper.get local time();
                                                                                              https://github.com/TUK-
                                                                                  SCVP/SCVP.artifacts/tree/master/tlm_quantum_keeper
                                                                                                                  Fraunhofer
© Fraunhofer IESE
```

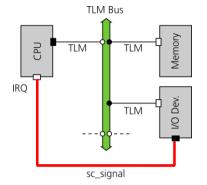
- There are three absolute time variables, namely the current simulation time as returned by sc_time_stamp(), the current time decoupled from simulation time, and the time of the next synchronization point.
- The global quantum is the time between two sync points.
- The local time is the time offset of the current time (as used by a temporally decoupled initiator) from the SystemC simulation time.
- The current time is the absolute time value as used by a temporally decoupled initiator.

Your notes:			



Your notes:		

A Closer Look on Functional Simulation Errors

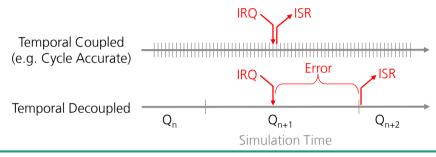


Temporal Coupled (e.g. Cycle Accurate)

- I/O Device makes an Interrupt Request (IRQ)
- The Interrupt Service Routine (ISR) will be called a few cycles later

Temporal Decoupled Simulation

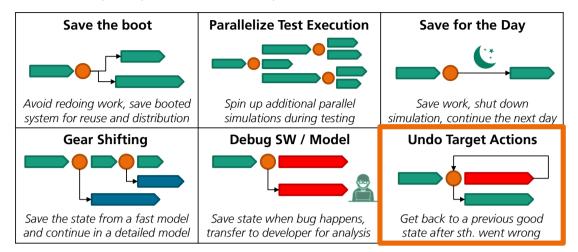
- I/O Device makes an IRQ
- The ISR will be called in the next Quantum



© Fraunhofer IESE

Checkpointing

The ability to save the state of a simulation and later pick up at the exact same point in time.



 Gläser et al. [4] presented in 2015 the idea of using checkpointing in order to rollback in simulation time and force an earlier synchronization to correct the occurred errors.

Source: Jacob Engblom, Intel, SystemC Evolution Day, 2017

[4] Temporal decoupling with error-bounded predictive quantum control. Georg Glaeser, Gregor Nitsche, Eckhard Hennig. Published in Forum on Specification and Design Languages (FDL) 2015



The Good Old fork()

- Parent Process

 fork()

 Parent Child
- fork() is a system call that allows a process in the OS to create a one-to-one copy of itself, called child.
- Supported by all OS: Linux, FreeBSD, macOS, ...
- Modern OS do not duplicate the complete memory space of a process
- Instead they use the Copy-on-Write semantic:
 - The copy operation is deferred to the first write to a memory page
 - In other words: a memory page is only copied in the moment of the change

Can fork() be used as an efficient way for checkpointing in order to get an error free temporal decoupled simulation?

49

© Fraunhofer IESE



Speculative Temporal Decoupling using fork()

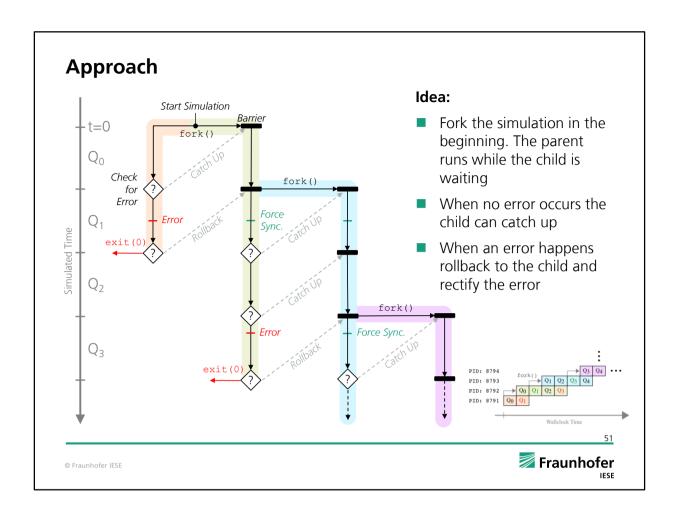
- Idea:
 - Use fork() to backup the simulation state
 - Execute the next quantum speculatively
 - In case of an error rollback in simulation time
 - Correct the timing error e.g. by temporary decreasing the quantum size
- Two approaches investigated:
 - **1. Naïve Approach** (Forking at each quantum)
 - 2. Lockstep Approach (Forking only in case of an error)
- Synchronization of parent and child is done with pipe (acts like a barrier)
- Details of the implementation are in the paper

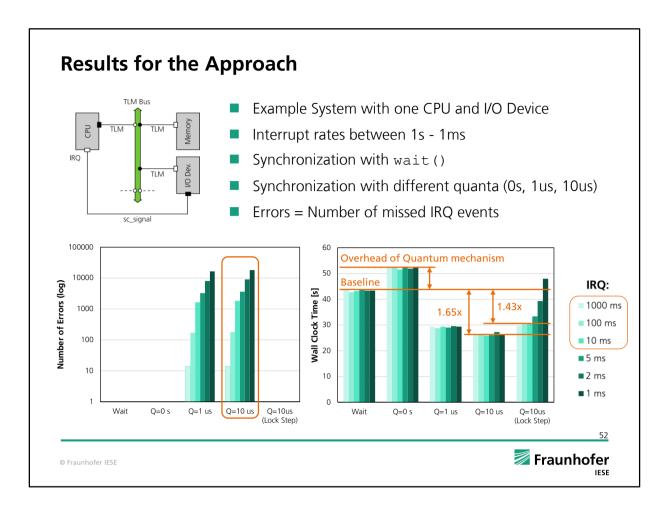
© Fraunhofer IESE



Undo Target Actions

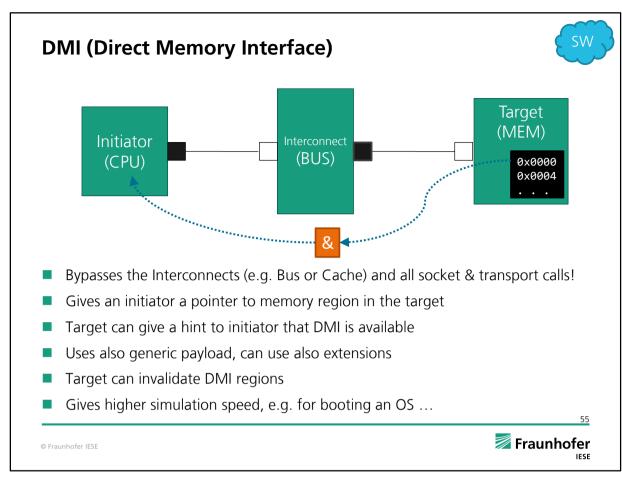






			S. CONTROL CAMPBELL		User L	braries				\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \			
Transaction Level Modeling (TLM)	Sockets & Generic	Blocking & Non-	Temporal Decoup- ling &	Phases	Payload Exten-	SystemC AMS	Electrical Linear Networks (ELN)	Linear S Flow (ignal _SF)	Timed Data Flow (TDF)			
saction (Payload	Blocking	<u>DMI</u>				sions	sions	Syste	Linear D	AE solver		Scheduler
Trans							Sy	nchroniza	tion lay	er			
		Predefined Primitive Channels: Mutexes, FIFOs & Signals											
SystemC	Simulation		Me	Methods & Threads			Channels & Interfaces			Data Types: Logic, Integers,			
Sys	Kϵ	Kernel Events, Sensitivity & Notifications				Fix				dpoint & tingpoint			
					C-	++							

Your notes:			



The main reason for DMI is simulation speed. DMI allows the transport interface to be bypassed by giving an initiator a direct pointer to an area of memory in a target. Instead of calling transport for each read and write operation, the initiator can simply access the memory directly. For operations other than read or write, it is possible to add extensions to the DMI transaction using the standard generic payload extension mechanism. It is the responsibility of the initiator to honor any invalidation calls received from the target

Your notes:			

DMI (Direct Memory Interface) bool get_direct_mem_ptr(tlm_generic_payload trans, tlm_dmi dmiData); Initiator Target Initiator Socket Socket Socket Socket Forward Path Forward Path (CPU) Return Path Return Path Backward Path Backward Path void invalidate_direct_mem_ptr(uint64 start, uint64 end); Same routing as e.g. b_transport

- Class tlm dmi:
 - unsigned char* dmi_ptr
 - uint64 start_address
 - uint64 end_address
- dmi_access_e granted_access
- sc_time read_latency
- sc_time write_latency

Fraunhofer

Your notes:	
	·

```
DMI Initiator
class Initiator: sc_module, tlm::tlm_bw_transport_if<> {
  bool dmi // set to false in the constructor;
  tlm::tlm_dmi dmiData;
                                                                                                                         Normal b_transport
  void process() {
  for (int i = 0; i < 16; i++)</pre>
                                                                                   } else {
                                                                                     iSocket->b_transport(trans, delay);
                                                                                                                                   Get DMI Hint!
       tlm::tlm generic payload trans;
                                                                                     if(trans.is dmi allowed() == true) {
       unsigned char data = rand();
                                                                                       dmiData.init(); // Reset DMI descriptor
       trans.set_address(i);
                                                                                       dmi = iSocket->get_direct_mem_ptr(trans, dmiData);
       trans.set_data_length(1);
       trans.set_command(tlm::TLM_WRITE_COMMAND);
       trans.set_data_ptr(&data);
                                                                                  wait(delay);
       sc_time delay = sc_time(0, SC_NS);
                                                    DMI start here
                                                                                } // end for
                                                                              sc_stop();
} // end process
       if ( dmi == true
            && i >= dmiData.get_start_address()
&& i <= dmiData.get_end_address())</pre>
                                                                              void invalidate_direct_mem_ptr(sc_dt::uint64 start_range,
         if( trans.get_command() == tlm::TLM_READ_COMMAND
    && dmiData.is_read_allowed())
                                                                                                                     sc dt::uint64 end range)
                                                                                dmi = false;
           memcpy(&data,
           dmiData.get_dmi_ptr() + i
                                                                              // Dummy methods
                - dmiData.get_start_address(),
           trans.get_data_length());
                                                                           };
           delay += dmiData.get_read_latency();
         else if( trans.get_command()==tlm::TLM_WRITE_COMMAND
    && dmiData.is_write_allowed())
                                                                                                        Try code on github:
           memcpy(dmiData.get_dmi_ptr() + i
                 - dmiData.get_start_address(),
                                                                                  https://github.com/TUK-SCVP/SCVP.artifacts/tree/master/tlm_lt_dmi
           trans.get_data_length());
           delay += dmiData.get_write_latency();
                                                                                                                      Fraunhofer
```


DMI Interconnect

© Fraunhofer IESE



```
class exampleInterconnect : sc_module, tlm::tlm_bw_transport_if<>, tlm::tlm_fw_transport_if<>
    tlm::tlm_initiator_socket<> iSocket;
    tlm::tlm_target_socket<> tSocket;
    SC_CTOR(exampleInterconnect) {
         tSocket.bind(*this); iSocket.bind(*this);
    void b_transport(tlm::tlm_generic_payload &trans, sc_time &delay) {
         delay = delay + sc_time(40, SC_NS);
iSocket->b_transport(trans, delay);
    bool get_direct_mem_ptr(tlm::tlm_generic_payload& trans, tlm::tlm_dmi& dmi_data) {
         bool dmi = iSocket->get_direct_mem_ptr(trans, dmi_data);
         dmi_data.set_read_latency( dmi_data.get_read_latency() + sc_time(40, SC_NS));
dmi_data.set_write_latency( dmi_data.get_write_latency() + sc_time(40, SC_NS));
                                                                                                               Forwarding DMI
                                                                                                               request on
         return dmi;
                                                                                                               forward and
    }
                                                                                                               backward path
    void invalidate_direct_mem_ptr(sc_dt::uint64 start_range, sc_dt::uint64 end_range)
         tSocket->invalidate_direct_mem_ptr(start_range, end_range);
    // Dummy methods ...
                                                                                                                Fraunhofer
```

```
DMI Target
class exampleTarget : sc_module, tlm::tlm_fw_transport_if<> {
     unsigned char mem[512];
     public:
     tlm::tlm_target_socket<> tSocket;
     SC_CTOR(exampleTarget) : tSocket("tSocket") {
    tSocket.bind(*this);
          SC_THREAD(invalidateProcess);
     }
                                                                                            In this example the DMI
     void invalidateProcess() {
                                                                                            access is invalidated every
          while(true) {
   wait(500, SC_NS);
   tSocket->invalidate_direct_mem_ptr(0,511);
                                                                                            500 ns, which is just an
                                                                                            artificial example
     void b_transport(tlm::tlm_generic_payload &trans, sc_time &delay) {
          trans.set_dmi_allowed( true );
                                                                                            Give Initiator a hint that DMI is possible
     }
     bool get_direct_mem_ptr(tlm::tlm_generic_payload& trans, tlm::tlm_dmi& dmi_data) {
          std::cout <<
                                        mem_ptr called" << std::endl;</pre>
          dmi_data.set_dmi_ptr(mem);
dmi_data.set_start_address(0);
                                                                                             Configure DMI object
          dmi_data.set_end_address(511);
dmi_data.set_read_latency(sc_time(40, SC_NS));
dmi_data.set_write_latency(sc_time(40, SC_NS));
                                                                                            with all relevant
                                                                                             information
          dmi_data.allow_read_write();
          return true;
     // Dummy methods ...
};
                                                                                                                    Fraunhofer
© Fraunhofer IESE
```


					User L	braries			- 7- 6 YEO O VURN (1890)	X 1/1/1		
Transaction Level Modeling (TLM)	Sockets & Generic	Blocking & Non-	Temporal Decoup- ling &	Phases	Payload Exten-	SystemC AMS	Electrical Linear Networks (ELN)	Linea Flov	r Signal v (LSF)	Timed Data Flow (TDF)		
saction (1	Payload	Blocking	-	DMI		sions	sions	Syste	Linear D	AE solve	er	Scheduler
Trans							Sy	/nchroni	ization laye	er		
		Predefined Primitive Channels: Mutexes, FIFOs & Signals										
SystemC	Simulation		Me	Methods & Threads			Channels & Interfaces			ta Types: r, Integers,		
Sys	Kϵ	Kernel Events, Sensitivity & Notifications				Fix				dpoint & tingpoint		
					C-	++						

Your notes:			

Debug Transport transport_dbg unsigned int transport_dbg(tlm_generic_payload trans); Initiator (CPU) Gives an initiator debug access to memory in a target Similar to b_transport Different: delay free, no waits, no event notifications Uses generic payload Same routing as b_transport Used for initialization, e.g. for bootloading

© Fraunhofer IESE

Your notes:			

Fraunhofer

```
Debug Transport transport_dbg
class Initiator : sc_module, tlm::tlm_bw_transport_if<> {
                                                                   class Target : sc_module, tlm::tlm_fw_transport_if<>
    void process() {
    ... // End of simulation:
                                                                     void b transport(... &trans, ... &delay)
         dumpMemory();
                                                                     {
    void dumpMemory()
                                                                     unsigned int transport_dbg(&trans)
        unsigned char buffer[64];
                                                                       if (trans.get_address() >= 1024) {
        tlm::tlm_generic_payload trans;
                                                                         return 0;
        trans.set_address(0);
        trans.set_read();
trans.set_data_length(64);
                                                                       if(trans.get_command() == tlm::TLM_WRITE_COMMAND)
        trans.set_data_ptr(buffer);
                                                                          memcpy(&mem[trans.get_address()],
        unsigned int n = iSocket->transport_dbg(trans);
                                                                                 trans.get_data_ptr(),
                                                                                 trans.get_data_length());
                                                                       } else /* tlm::TLM_READ_COMMAND */ {
         for(unsigned int i = 0; i < n; i++) {</pre>
            std::cout << std::hex
                                                                         memcpy(trans.get_data_ptr(),
                       << std::setfill('0')
                                                                                &mem[trans.get_address(),
                       << std::setw(2)
<< (unsigned int)buffer[i];</pre>
                                                                                trans.get_data_length());
                                                                       return trans.get_data_length();
                                                                  };
            if((i+1)%8 == 0) {
    std::cout << std::endl;</pre>
       }
   }
                                                                                       Try code on github:
};
                                                                                     https://github.com/TUK-
                                                                          SCVP/SCVP.artifacts/tree/master/tlm_lt_debug_transport
                                                                                                       Fraunhofer
```

our notes: