Using the new TLM-2.0 Standard for the Creation of Virtual Platforms for ESL Design

Dr. Tim Kogel

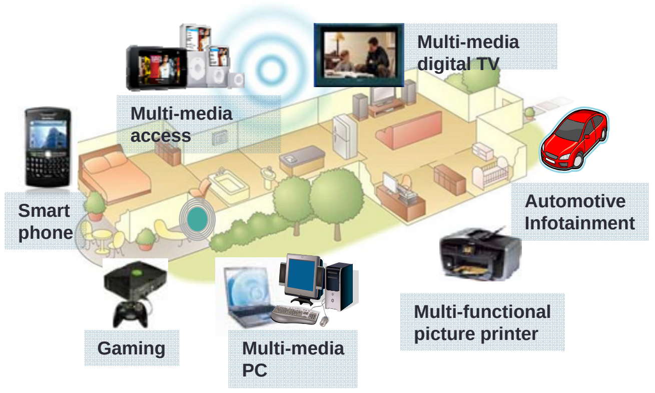
Office of the CTO
CoWare, Inc.

Overview

- MP-SoC Trends and Challenges

- ESL Design Solutions
  - Design Tasks and Requirements
  - Enabling technologies

Design Trends



Device convergence

Wireless connectivity anytime, anywhere

High definition Imaging anywhere

HW Centric	→	SW and HW Centric
Local memory subsystem		Complex memory hierarchy
Local, shared bus		Intelligent interconnect (NoC)
Single processor		Multiple processor
Single SW stack		Multiple, dependent SW stacks

Transition from ASIC to MPSoC

SW Driven Design

- Exploding SW content?
- Higher clock frequency?
- Increased memory?

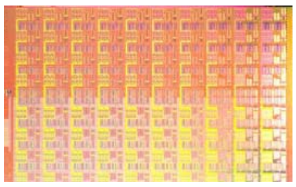
ASIC Cost ←

Complex ASIC

- High Definition
- Convergence
- Wireless Everywhere

Power ↓

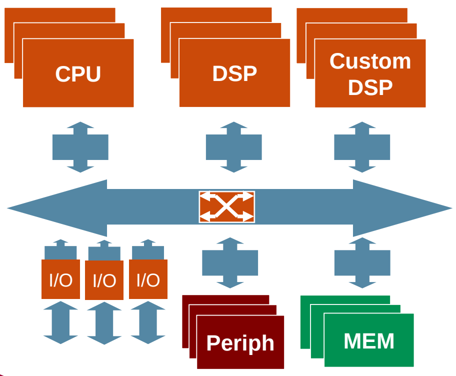
Multi Core SoC



- Portable devices?

Energy Efficiency →

MP-SoC



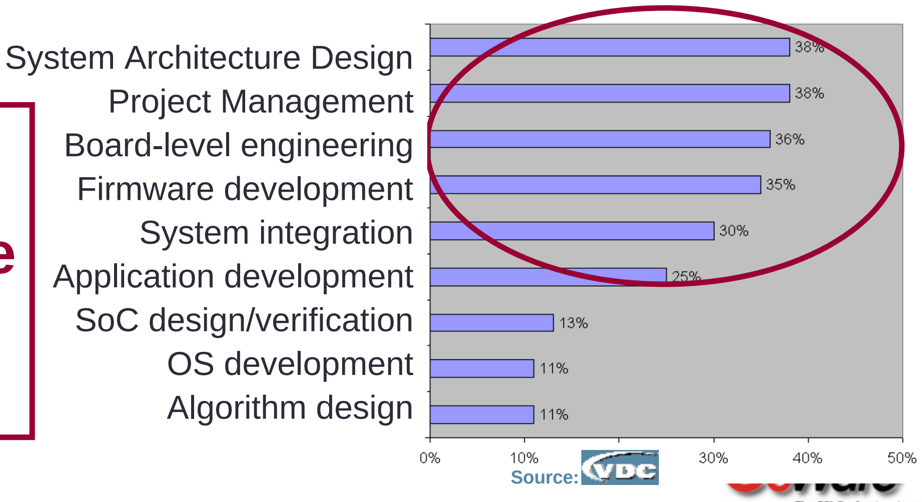
Design Challenges

- **24%** of projects canceled due to schedule slip
- **54%** of SW designs completed **behind schedule**
- **33%** of devices **miss functionality/performance**
- **80%** of effort to correct **errors discovered late**

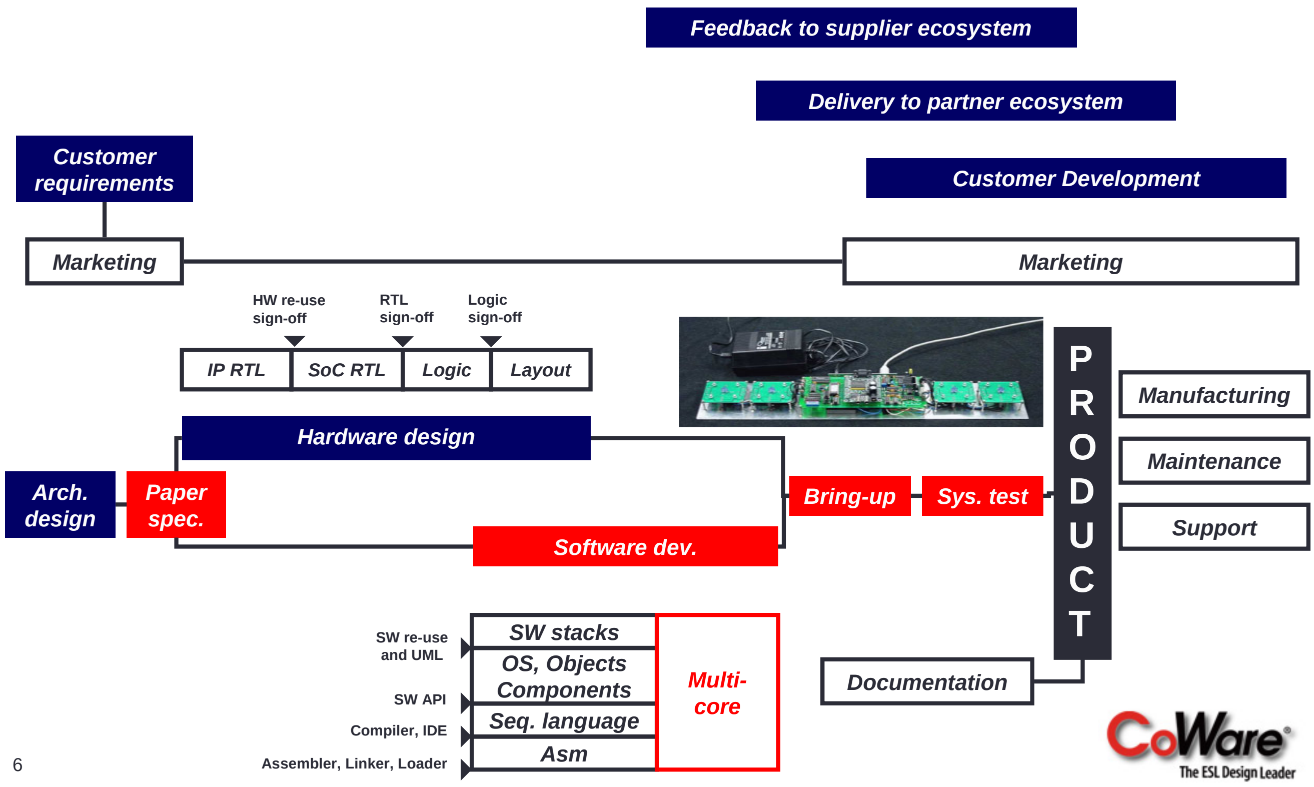
Top issues

Architecture
Software
Integration

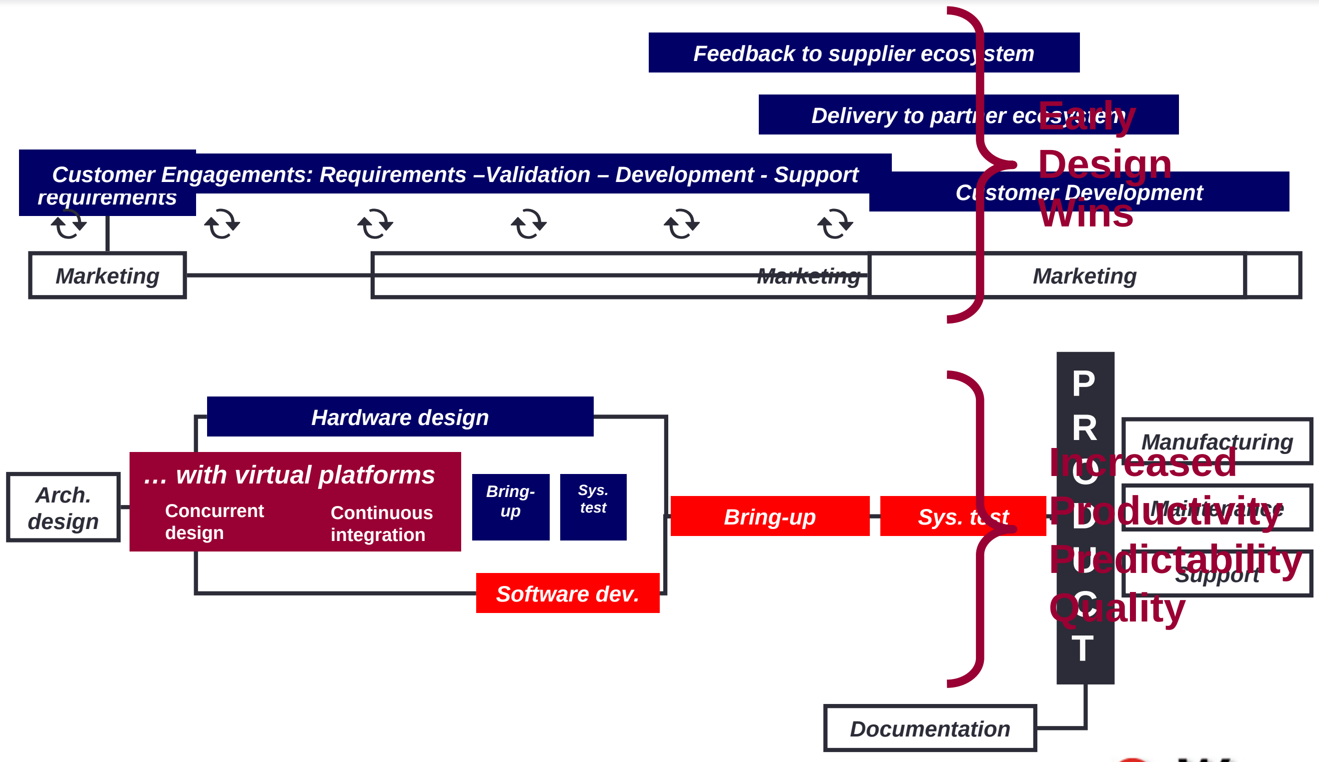
Engineering Tasks Believed To Be the Cause of Schedule Delay on the Current Project



MP-SoC Design Flow Challenges



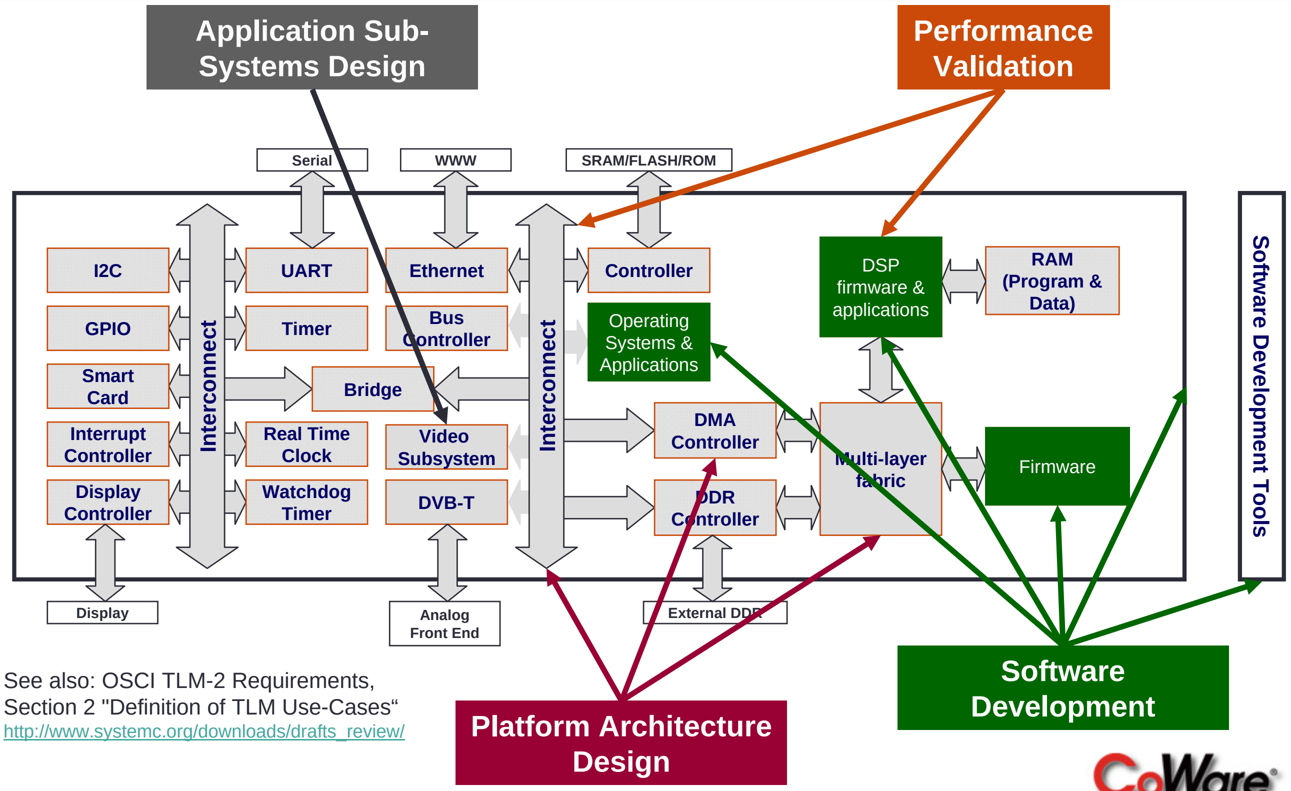
Solution: ESL Design



Overview

- MP-SoC Trends and Challenges
- ESL Design Solutions
  - Design Tasks and Requirements
  - Enabling technologies

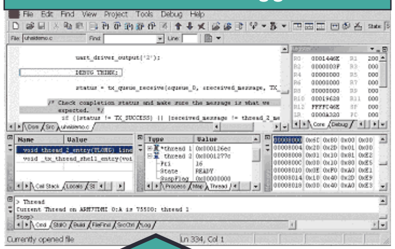
Need virtual platforms for …



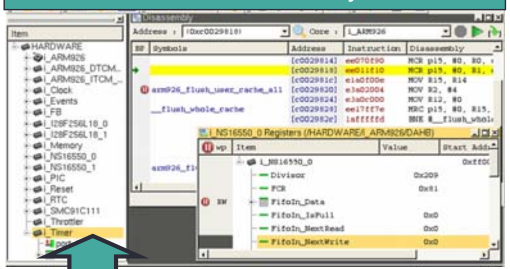
See also: OSCI TLM-2 Requirements, Section 2 "Definition of TLM Use-Cases“
http://www.systemc.org/downloads/drafts_review/

Software Application Development

Software Debugger

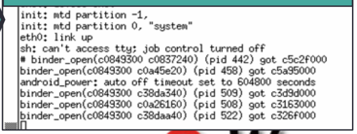


Virtual Platform Analyzer



Keypad/Display Device

Console



```
init: mtd partition -1,
init: mtd partition 0, "system"
eth0: link up
sh: can't access tty; job control turned off
# binder_open(c0849300 c0837240) (pid 442) got c5c2f000
binder_open(c0849300 c0a45e20) (pid 458) got c5a95000
android_power: auto off timeout set to 604800 seconds
binder_open(c0849300 c38da340) (pid 509) got c3d9d000
binder_open(c0849300 c0a26160) (pid 508) got c3163000
binder_open(c0849300 c38daa40) (pid 522) got c326f000
```

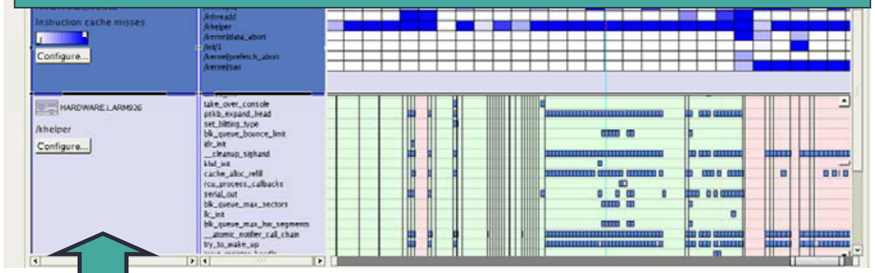
Requirements

- Sufficient simulation speed (10-50% real-time)
- Functional completeness and register accuracy
- Timing accuracy: software synchronization
- Controllability and observability
- Integration with Software IDEs
- External connectivity

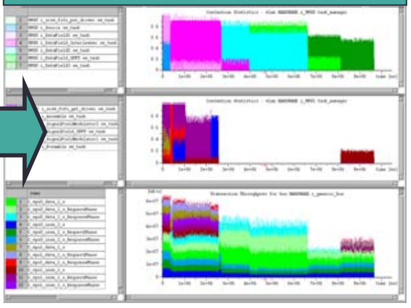
SystemC Virtual Platform

Software Performance Analysis

Software Performance Analysis



Hardware Performance Analysis



Requirements

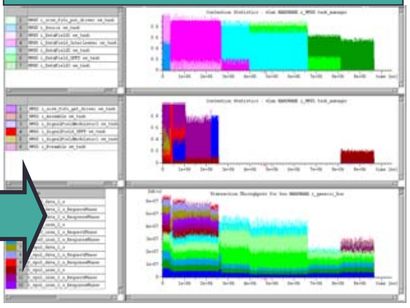
- Sufficient simulation speed (1-10% real-time)
- Functional completeness and register accuracy
- Timing accuracy: 80% (interval: ~100k cycles)
- Hardware and software performance analysis views
- External connectivity

SystemC Virtual Platform

Architecture Analysis

- Workload modeling options:
  - Trace-driven File Reader Bus Master
  - Task-graph driven Virtual Processing Unit

Hardware Performance Analysis



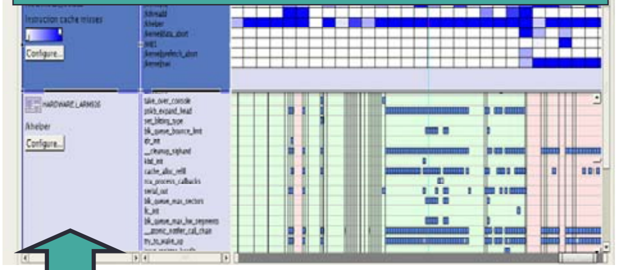
Requirements

- Sufficient simulation speed (100-1000 x RTL)
- Cycle-accurate models of critical components
  - Interconnect, memory subsystem
- Same level of configurability as real IP
- Timing accuracy: 95% (interval: 1-10 cycles)
- Hardware performance analysis views

SystemC Virtual Platform

Example: Performance Validation

Software Performance Analysis

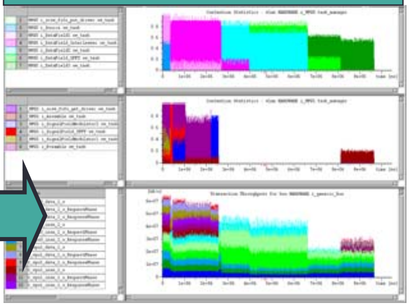


Requirements

- Sufficient simulation speed (50-500 x RTL)
- Cycle-accurate models of critical components
  - Processor, interconnect, memory subsystem
- Functional completeness and register accuracy
- Timing accuracy: 95% (interval: 1-10 cycles)
- Hardware and software performance analysis views

SystemC Virtual Platform

Hardware Performance Analysis



Overview

- **MP-SoC Trends and Challenges**
- **ESL Design Solutions**
  - **Design Tasks and Requirements**
  - **Enabling technologies**

Outline

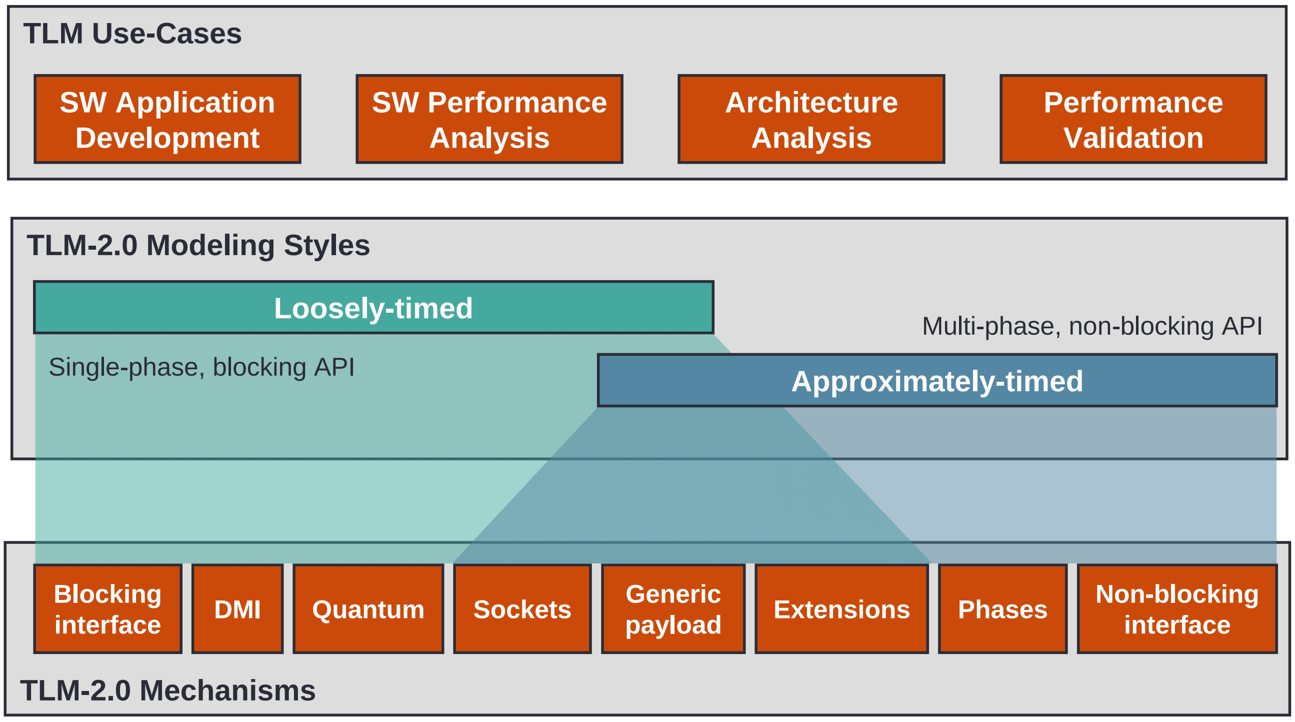
- TLM-2.0 Standard Overview
  - Concepts and APIs
  - The Loosely Timed Modeling Style
  - The Approximately Timed Modeling Style
- Effective Creation of TLM-2.0 Peripheral Models
- Creating TLM-2.0 based Virtual Platforms

OSCI TLM WG

Source: OSCI SystemC Community Update, DATE 2007

- 120 individuals from 27 organizations
- ~20 individuals from ~17 organizations participate regularly in weekly 2-hour teleconference

TLM-2.0 Overview



Generic Payload

- Typical set of memory mapped bus attributes

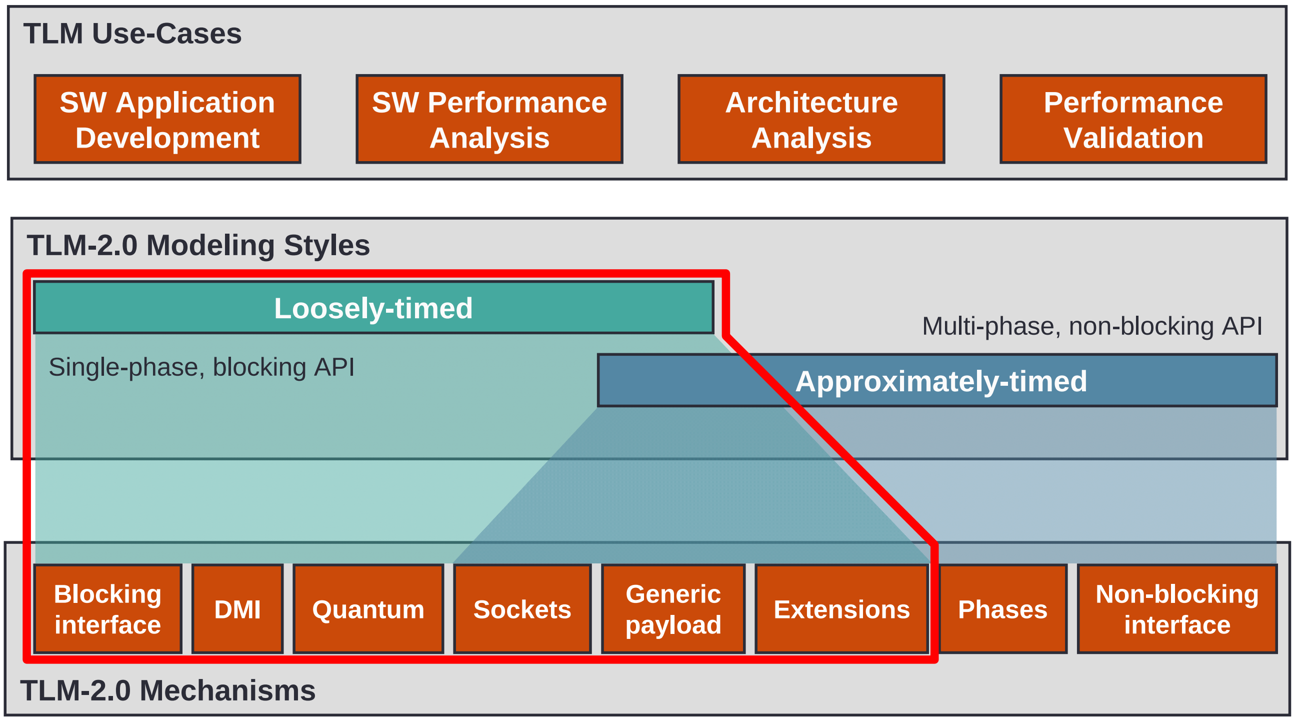
command	: enum,	READ, WRITE, IGNORE
address	: uint64,	byte address
data	: unsigned char*,	pointer to storage
length	: unsigned int,	number of bytes in the data array
byte_enable	: unsigned char*,	species sub-word accesses
byte_enable_length	: unsigned int,	number of elements in byte_enable
streaming_width	: unsigned int,	defines a streaming burst
response_status	: enum,	INCOMPLETE, OK, ERROR-code

- Extension mechanism
  - Array of pointers to user defined payload extensions
  - Defines rules for ignorable and mandatory extensions

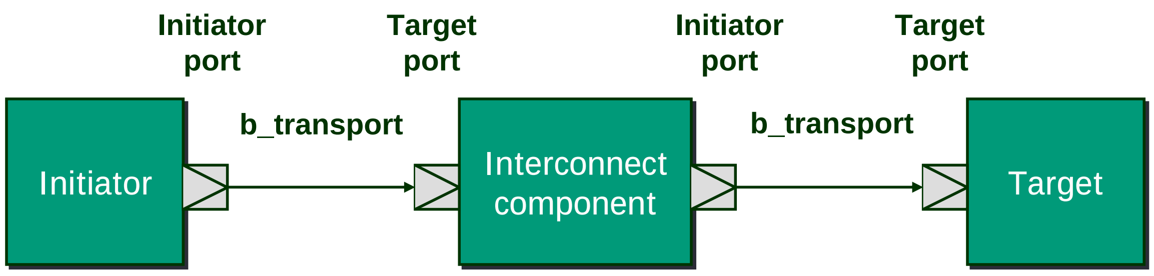
- Memory Management
  - Reference counting mechanism
  - Mandatory for AT, optional for LT

- Helper functions for endianness conversion

TLM-2.0 Overview



Blocking Transport

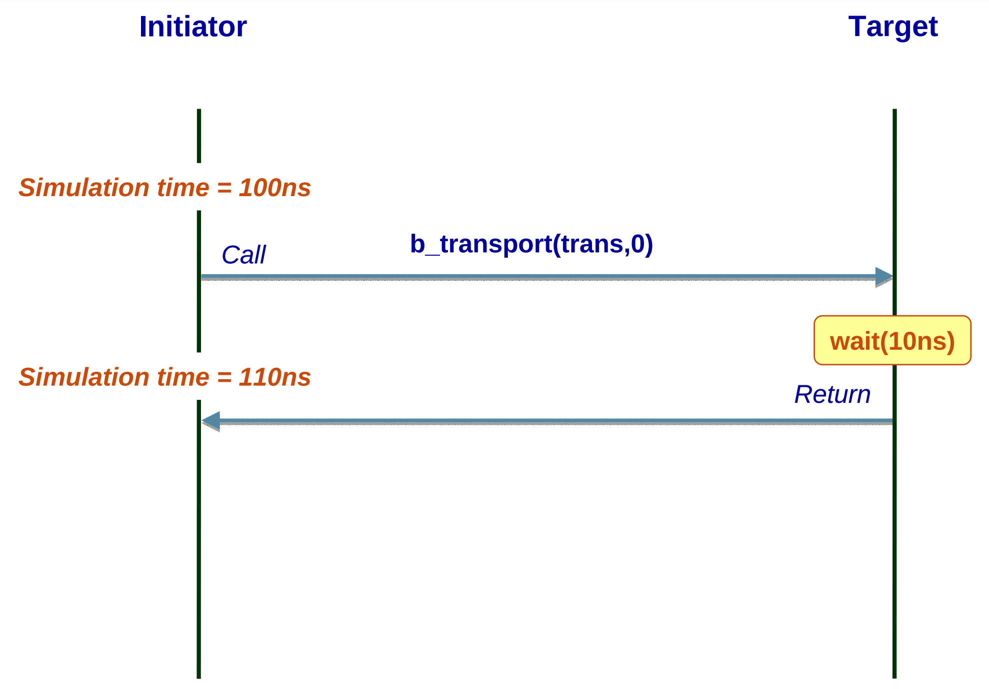


```
tlm_blocking_transport_if {
    void   b_transport ( TRANS& trans ,
                         sc_core::sc_time& t );
};
```

Simple API, support for timing annotation, addressing all SW related ESL Design tasks

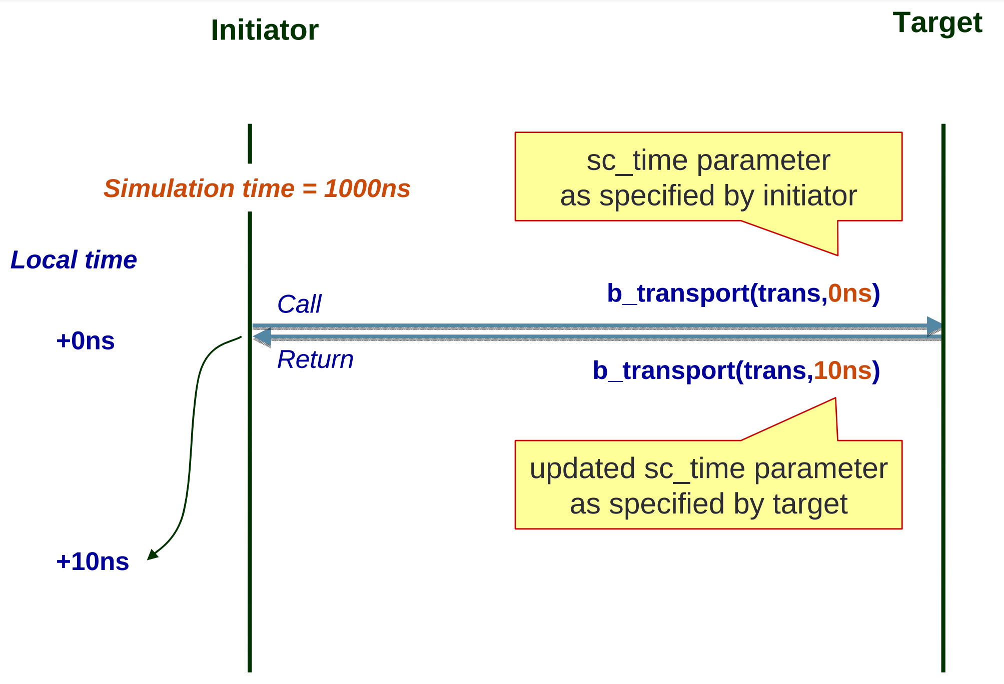
Sources: OSCI and CoWare (adapted from the TLM-2 Draft 2 manual)

Blocking Transport



Initiator is blocked until return from b_transport

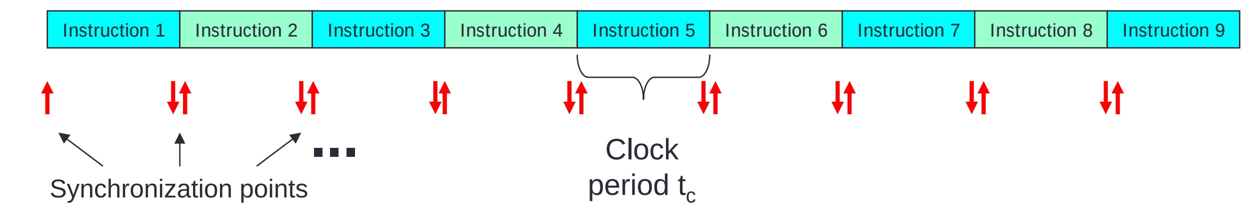
Loosely-timed with Timing Annotation



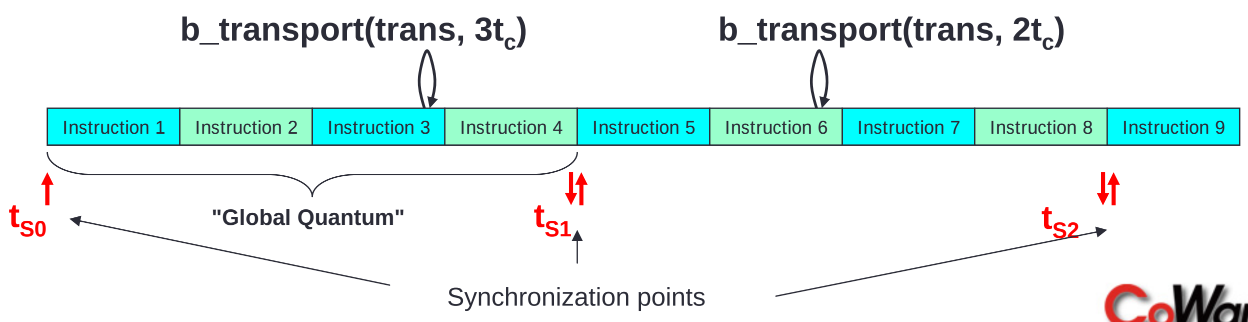
Transaction completed immediately with timing annotation

Temporal Decoupling

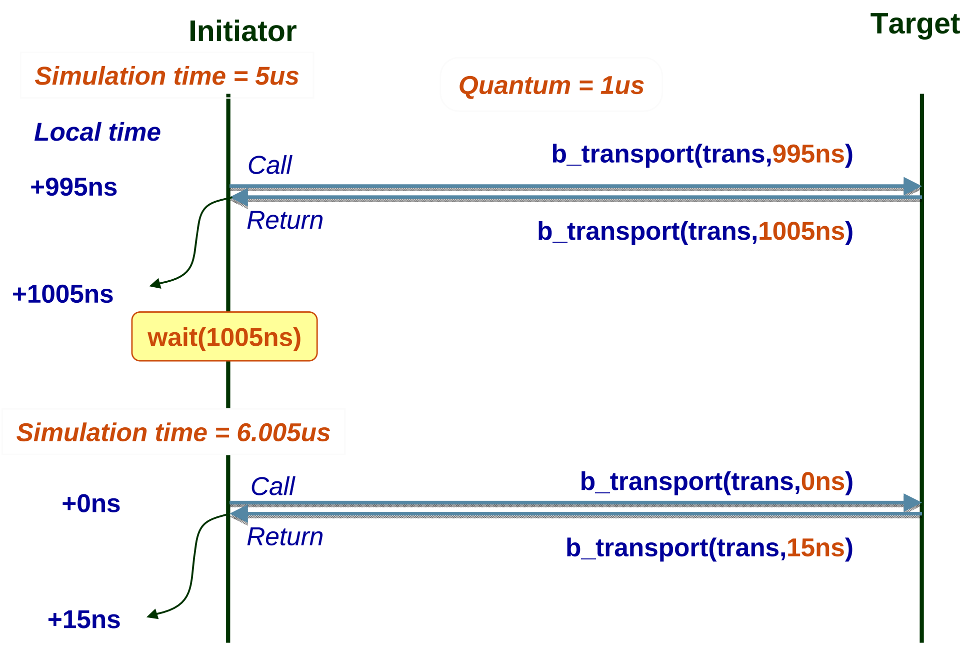
Clock-driven Modeling Style



Loosely Timed Modeling Style

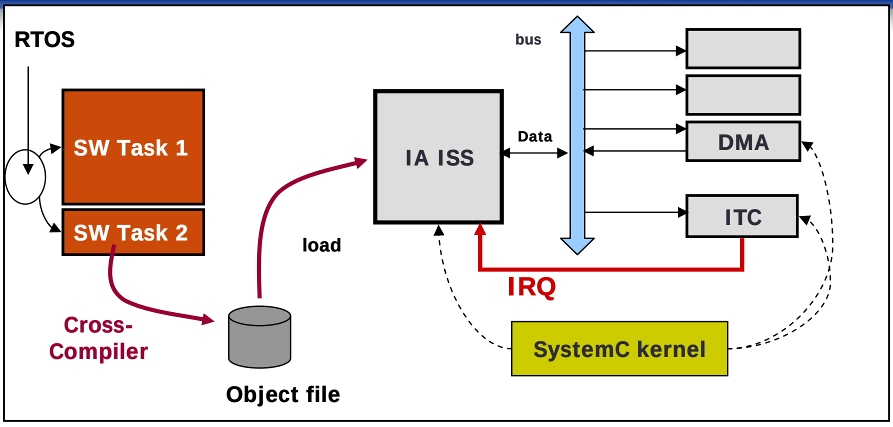


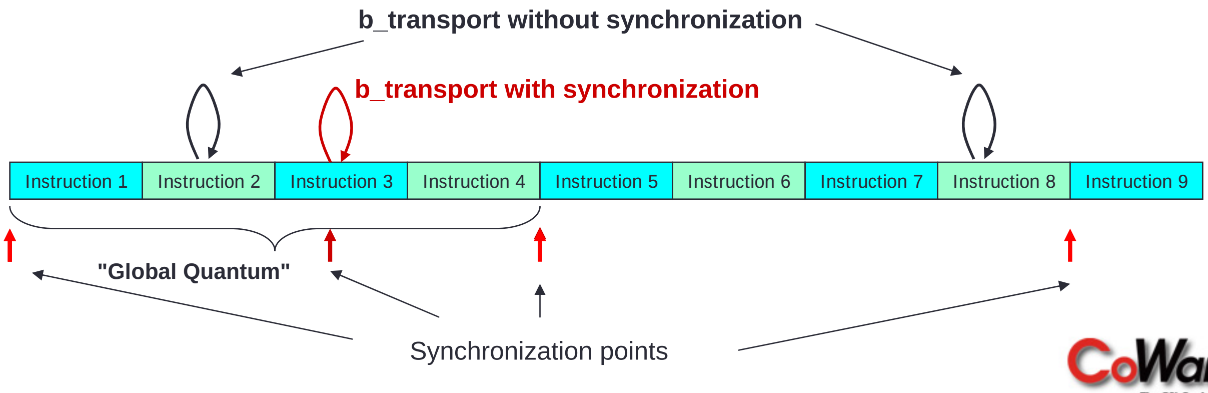
The Time Quantum



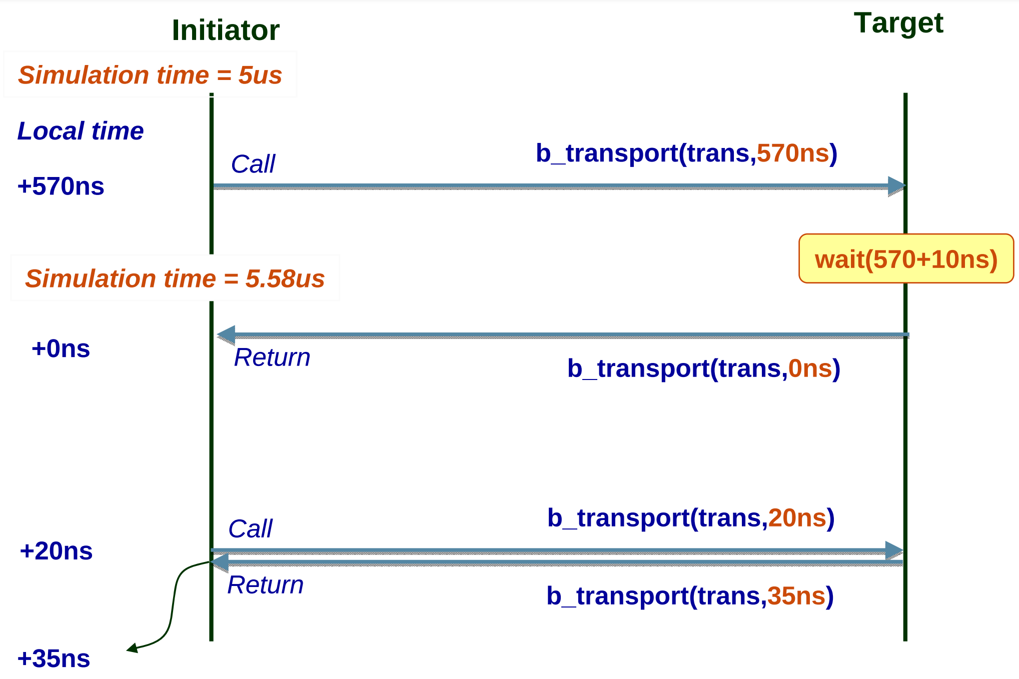
Initiator waits when local time exceeds the quantum

"Synchronization on Demand"

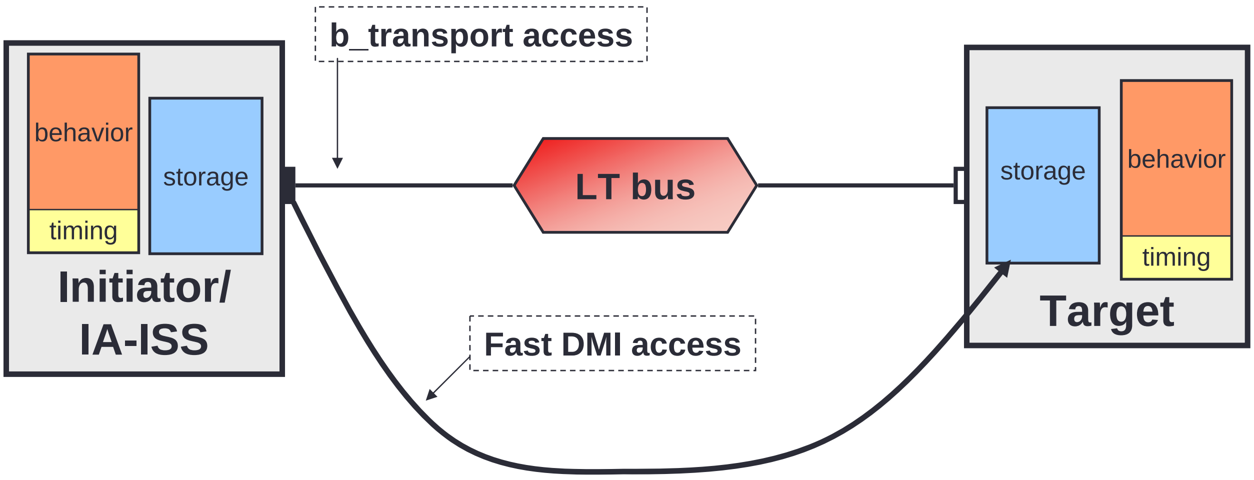




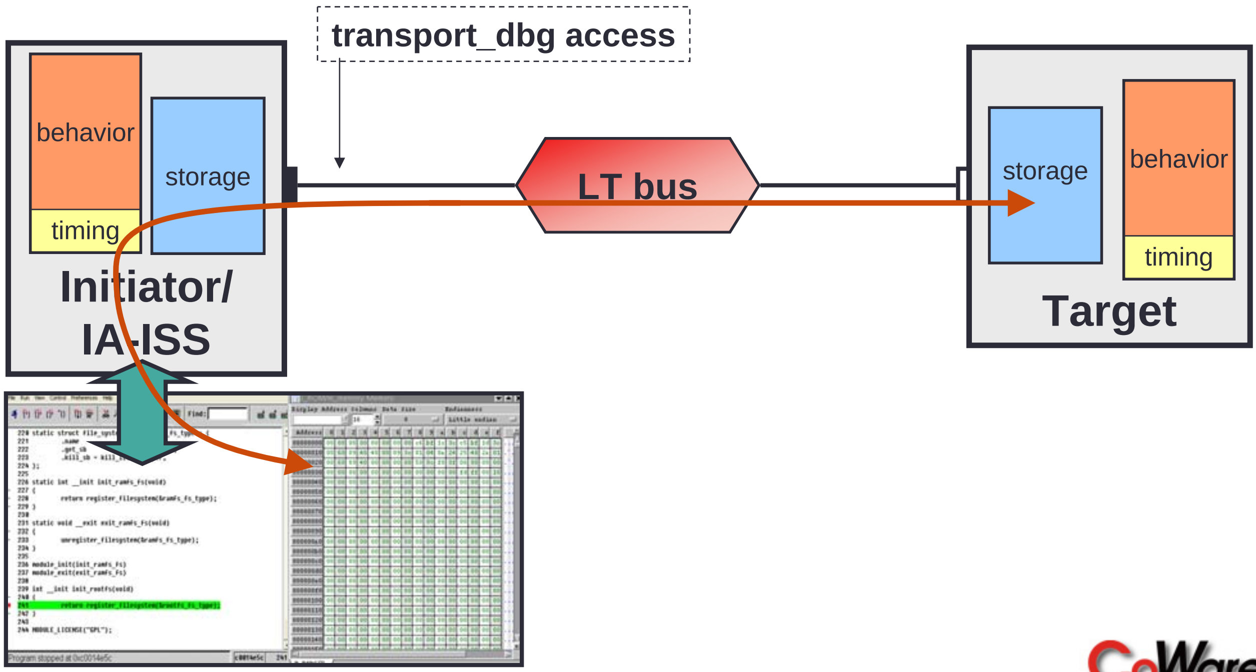
Temporal Decoupling with Synchronization



Direct Memory Interface

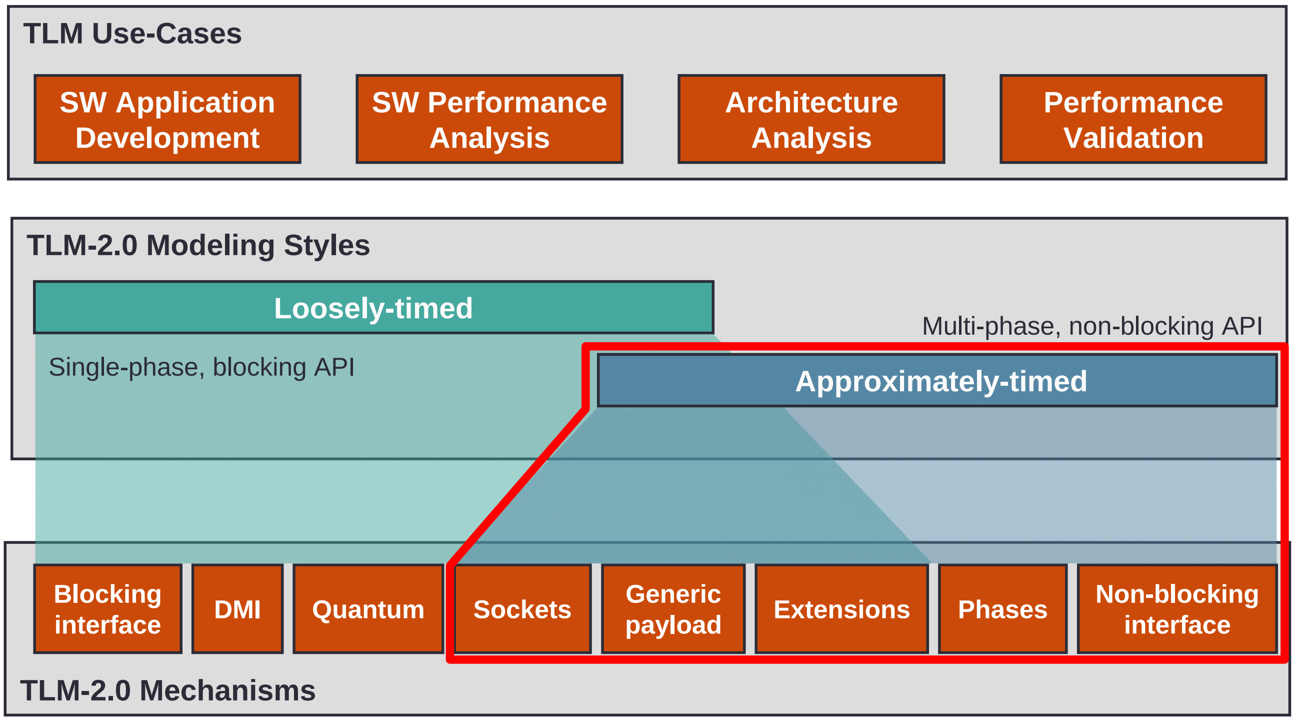


Debug Transport

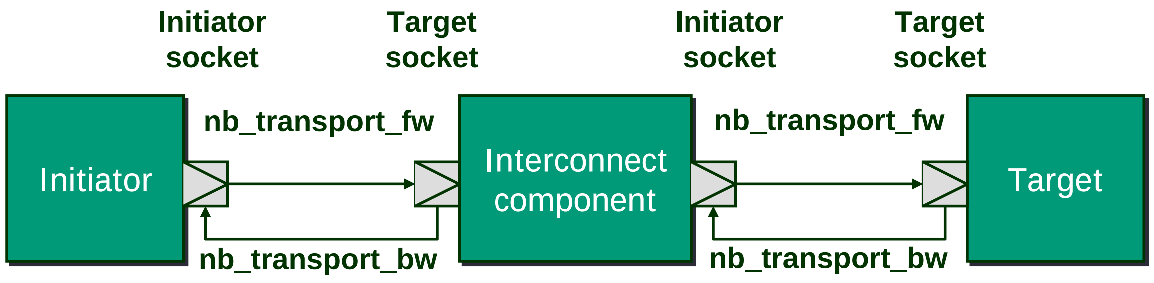


```
static struct file_syste                 fs_type = {
        .name
        .get_sb
        .kill_sb = kill_
};

static int __init init_ramfs_fs(void)
{
        return register_filesystem(&ramfs_fs_type);
}

static void __exit exit_ramfs_fs(void)
{
        unregister_filesystem(&ramfs_fs_type);
}

module_init(init_ramfs_fs)
module_exit(exit_ramfs_fs)

int __init init_rootfs(void)
{
        return register_filesystem(&rootfs_fs_type);
}

MODULE_LICENSE("GPL");
```

TLM-2.0 Overview



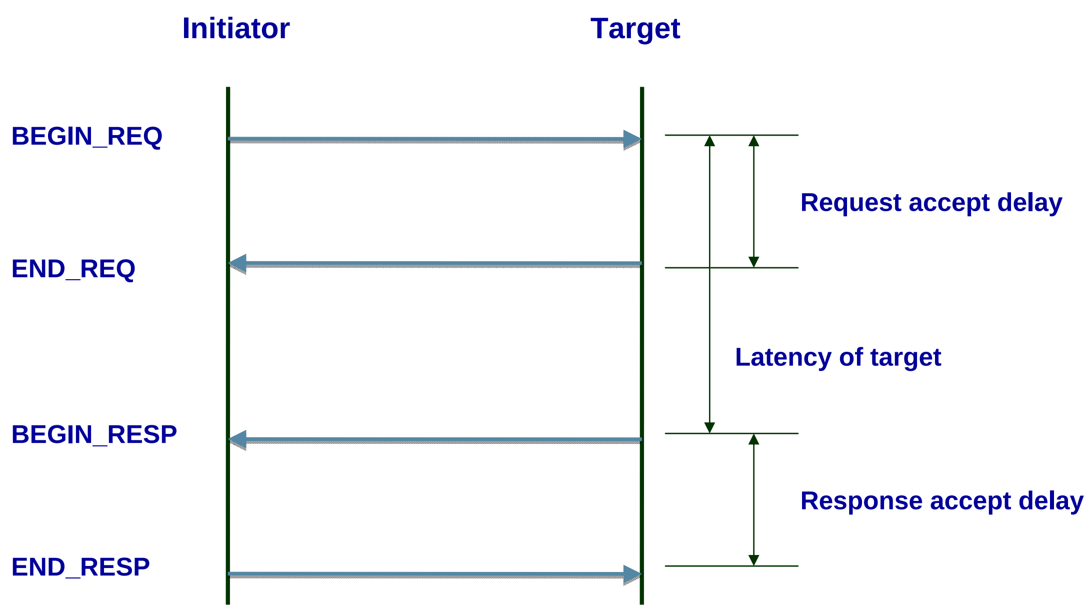
Non-Blocking Transport



```
template <   typename  TRANS  =  tlm_generic_payload,
             typename  PHASE  =  tlm_phase>

class  tlm_fw_nonblocking_transport_if : public  virtual  sc_core::sc_interface  {
public:
   virtual   tlm_sync_enum   nb_transport(  TRANS&  trans,
                                            PHASE&  phase,
                                            sc_core::sc_time&  t )  =  0;
};
```

Approximately-timed Timing Parameters

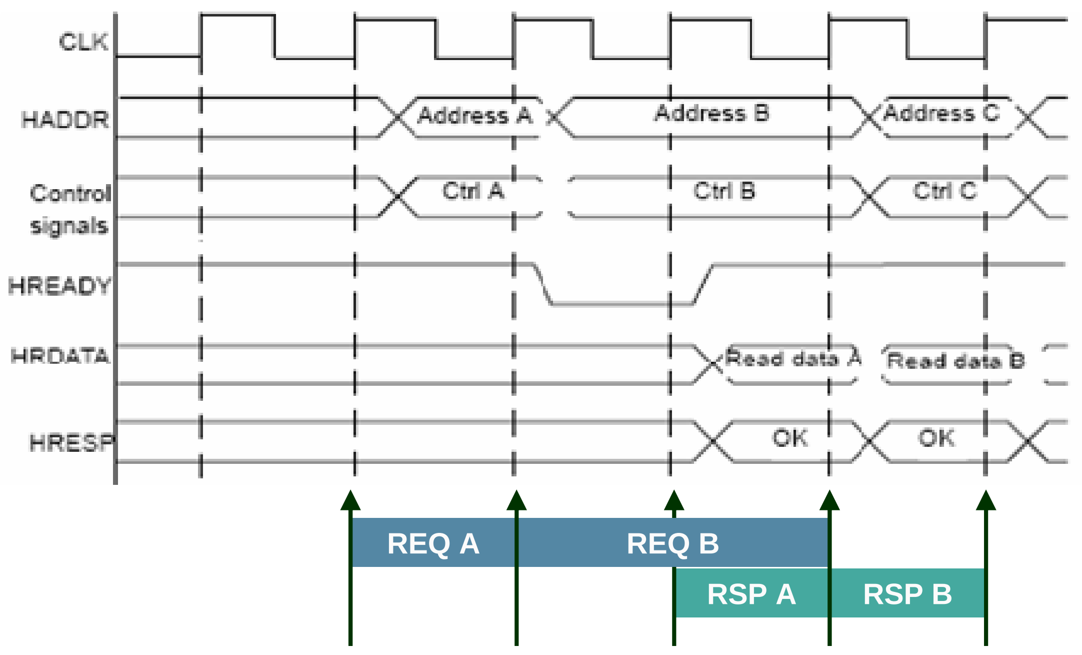
TLM 2.0 Base Protocol



BEGIN_REQ must wait for previous END_REQ, BEGIN_RESP for END_RESP

Mapping AT to Real Bus Protocols

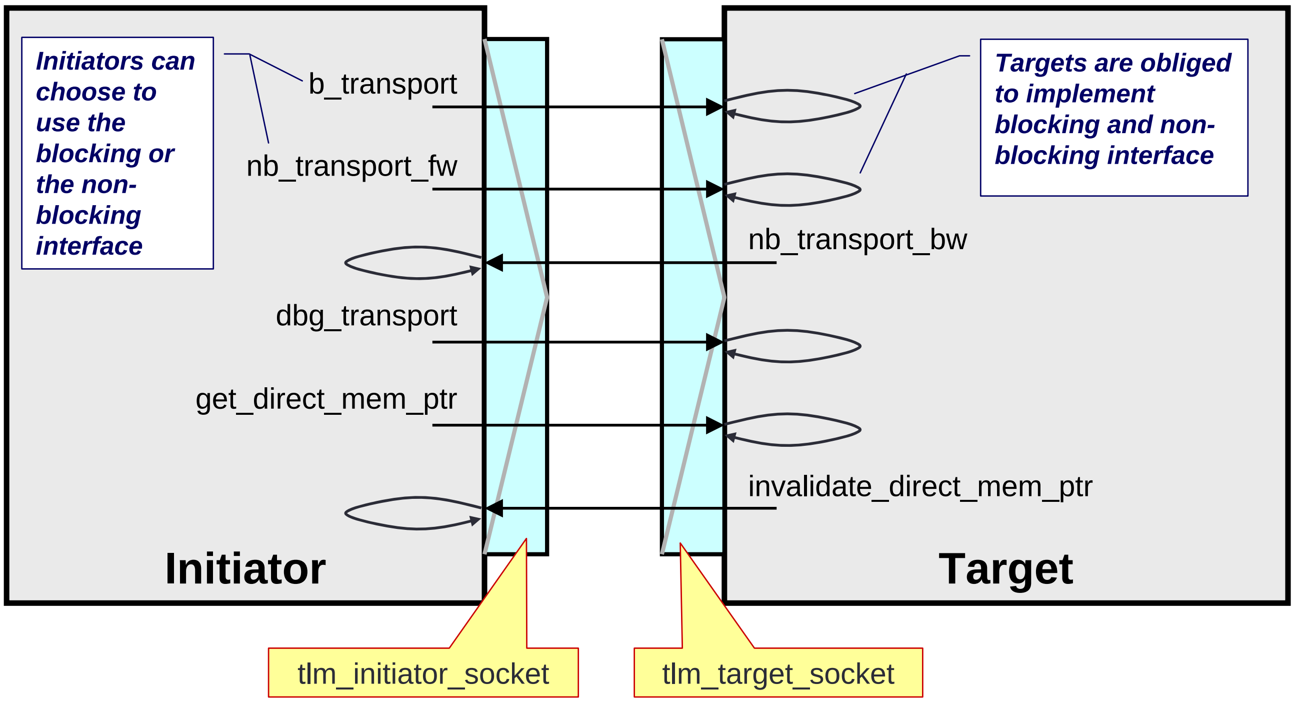
- Timing of the AHB Initiator Protocol



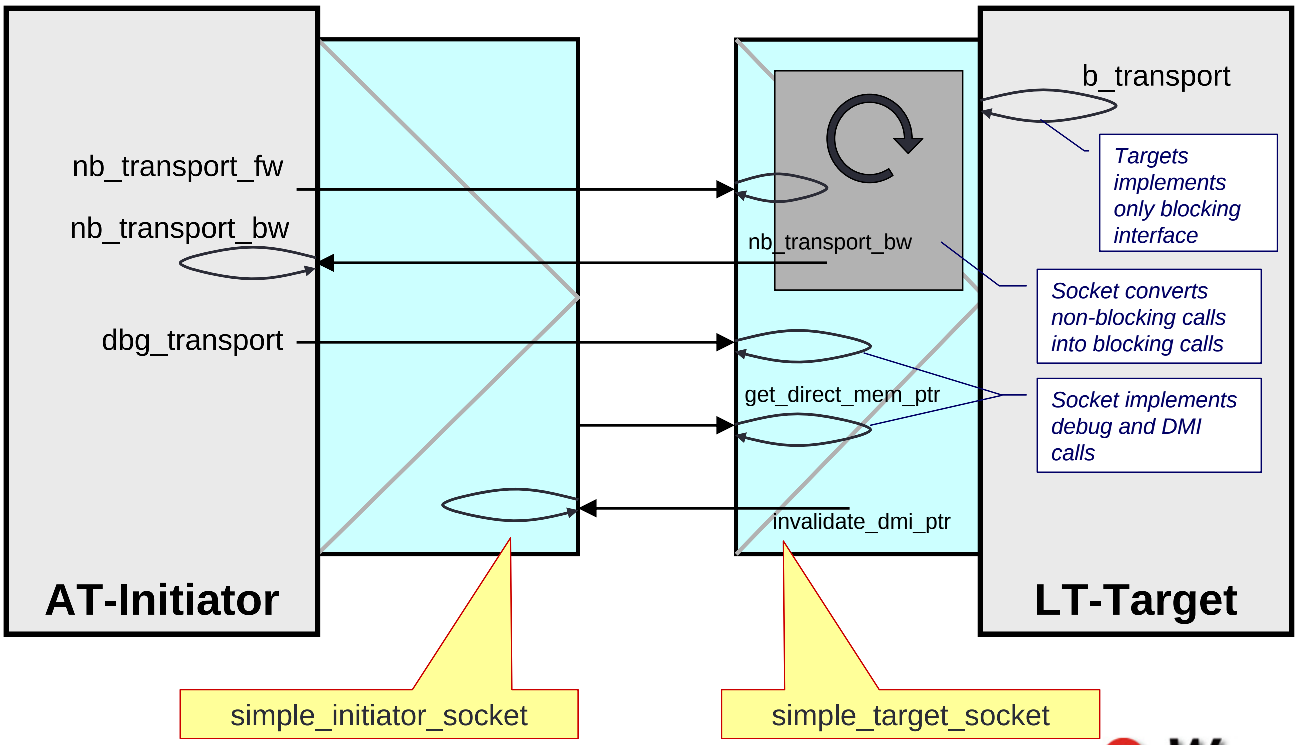
What are the Limitations?

- Goal of Base Protocol:
  - Mimic performance of real IP with generic AT models
  - Bridge TLM-2.0 with protocol-specific CA models
- Limitations:
  - Base Protocol does not represent the specifics of all protocols
  - E.g. no out-of-order transactions, no interleaving of bursts
- Strategy for refinement
  - Use TLM-2.0 extension mechanism for payload and phases to enhance accuracy
  - Owners of standard protocols (ARM, OCP-IP) are expected to define protocol specific TLM-2.0 extension kits

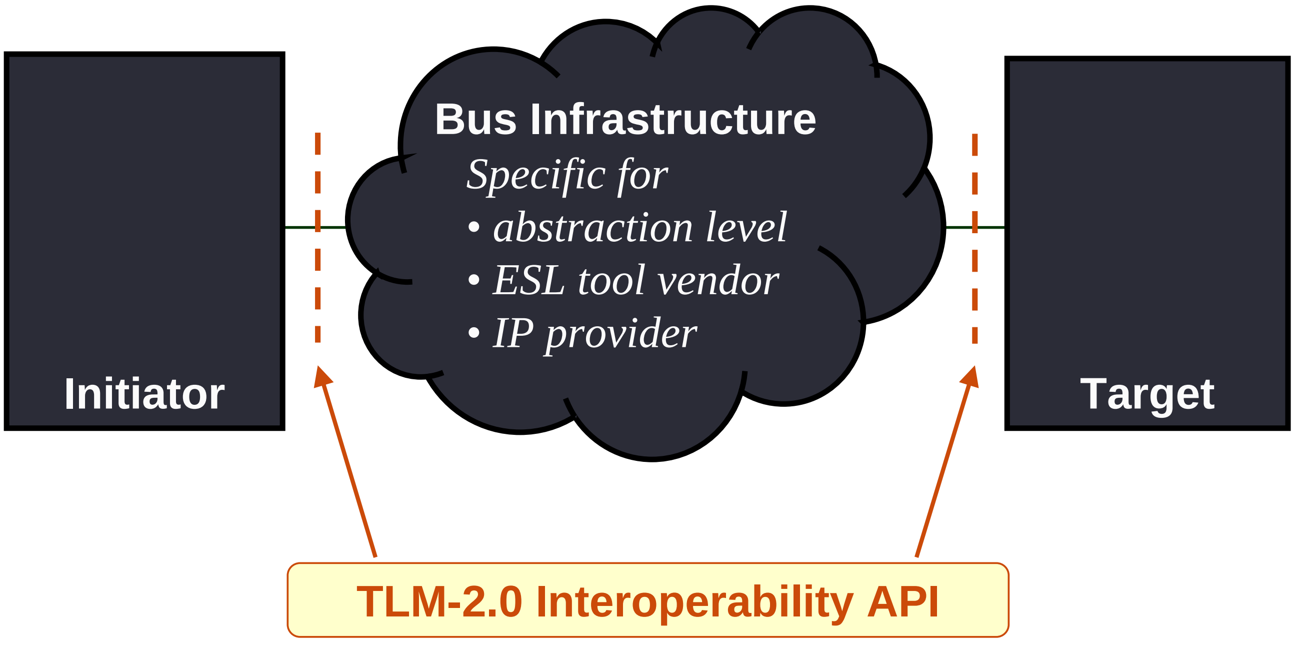
TLM-2.0 Standard Sockets


Initiators can choose to use the blocking or the non-blocking interface
b_transport
nb_transport_fw
dbg_transport
get_direct_mem_ptr
Initiator
Targets are obliged to implement blocking and non-blocking interface
nb_transport_bw
invalidate_direct_mem_ptr
Target
tlm_initiator_socket
tlm_target_socket

"Simple" TLM-2.0 Utility Sockets



TLM-2.0 Model Interoperability



Outline

- TLM-2.0 Standard Overview
- Effective Creation of TLM-2.0 Peripheral Models
  - ... using the CoWare SystemC Modeling Library
- Creating TLM-2.0 based Virtual Platforms

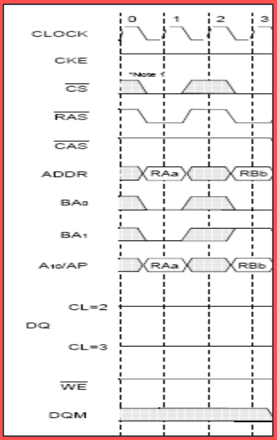
CoWare's SCML Methodology

Bus interface
(re-target communication to protocol)

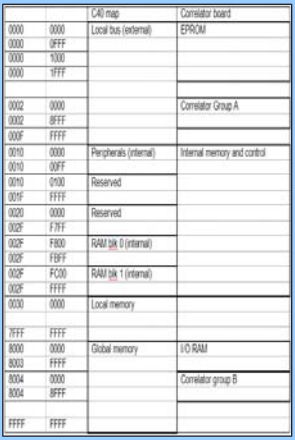
Register interface
(re-target algorithm to platform)

Behavior
(re-usable algorithm)

OCP, AMBA, CoreConnect, ...

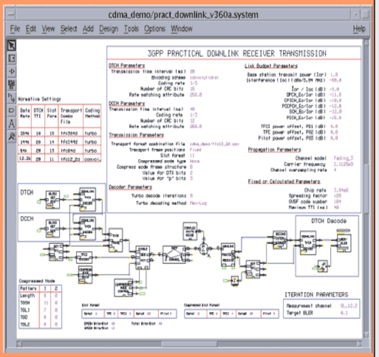


Address, access size, burst, ...
Read / write ahead buffer, ...



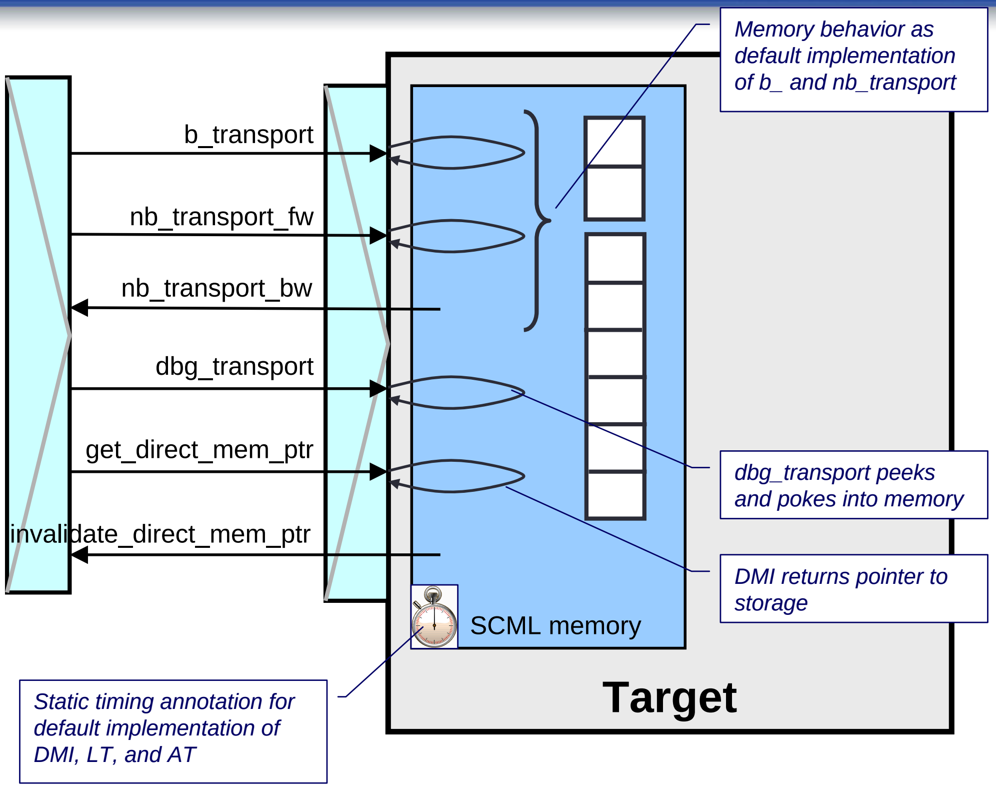
		C40 map	Correlator board
0000	0000	Local bus (external)	EPROM
0000	0FFF		
0000	1000		
0000	1FFF		
0002	0000		Correlator Group A
0002	8FFF		
000F	FFFF		
0010	0000	Peripherals (internal)	Internal memory and control
0010	00FF		
0010	0100	Reserved	
001F	FFFF		
0020	0000	Reserved	
002F	F7FF		
002F	F800	RAM blk 0 (internal)	
002F	FBFF		
002F	FC00	RAM blk 1 (internal)	
002F	FFFF		
0030	0000	Local memory	
7FFF	FFFF		
8000	0000	Global memory	I/O RAM
8003	FFFF		
8004	0000		Correlator group B
8004	8FFF		
FFFF	FFFF		

Algorithm, Timer, DMA, ...

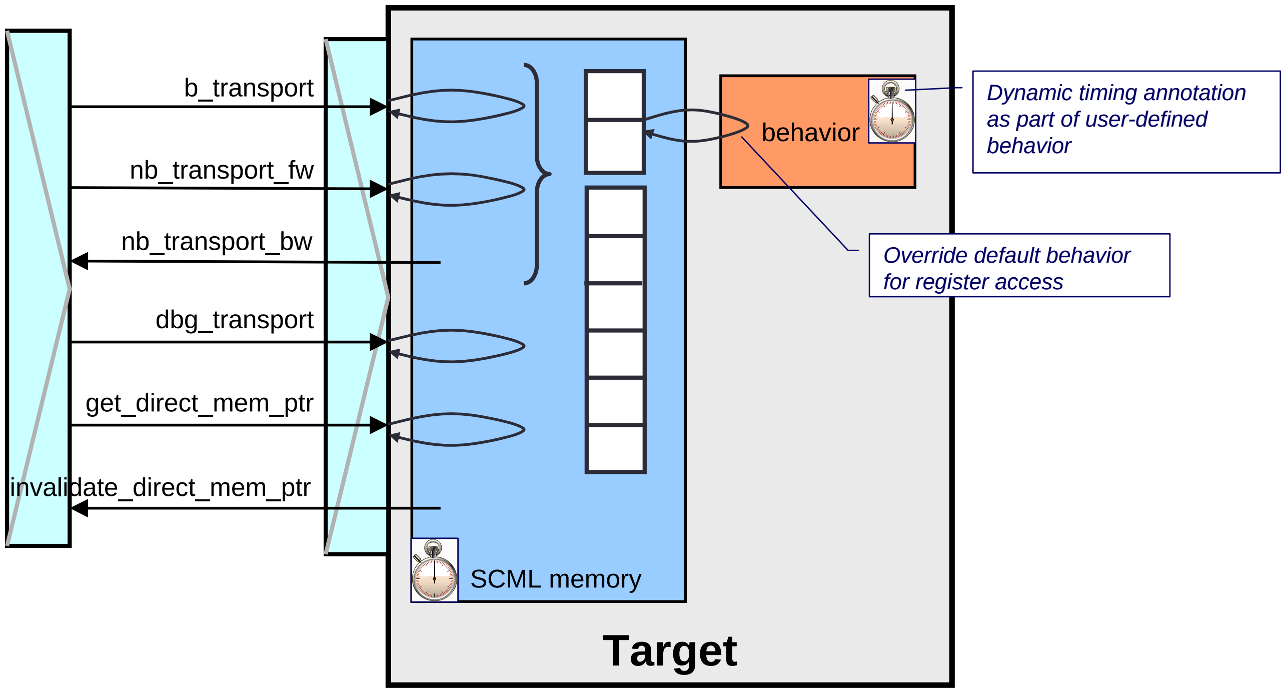

cdma_demo/pract_downlink_v360a.system
File Edit View Select Add Design Tools Options Window
Help
3GPP PRACTICAL DOWNLINK RECEIVER TRANSMISSION
DTCH
DCCH
DTCH Decode
ITERATION PARAMETERS

Maximize code reuse through orthogonalization

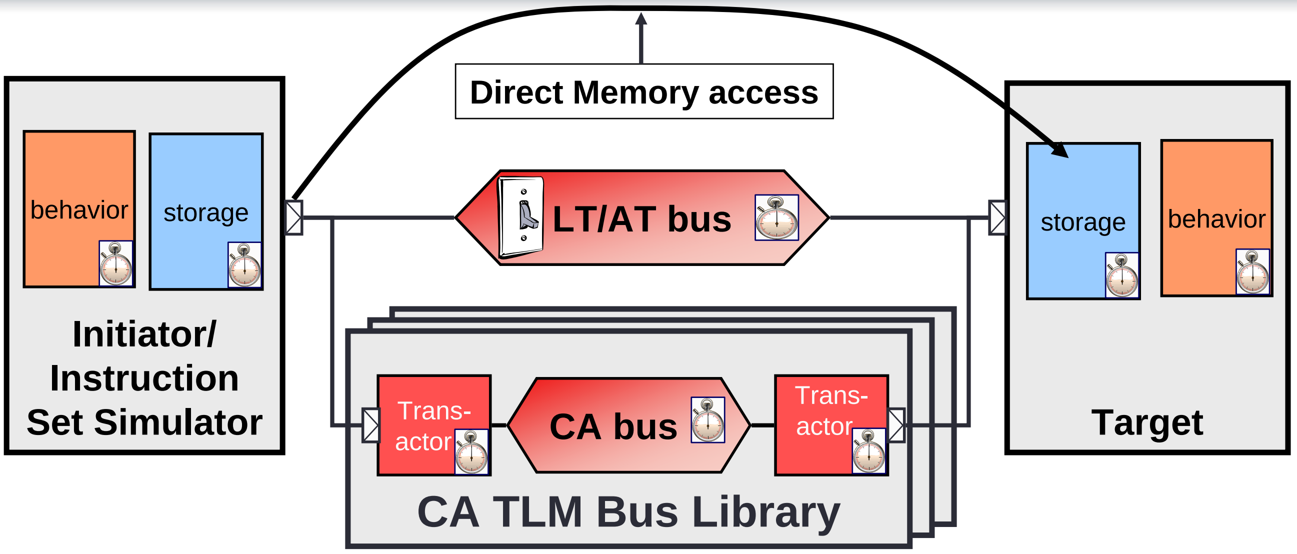
SCML Memory


b_transport
nb_transport_fw
nb_transport_bw
dbg_transport
get_direct_mem_ptr
invalidate_direct_mem_ptr
SCML memory
Target
Memory behavior as default implementation of b_ and nb_transport
dbg_transport peeks and pokes into memory
DMI returns pointer to storage
Static timing annotation for default implementation of DMI, LT, and AT

SCML Memory


b_transport
nb_transport_fw
nb_transport_bw
dbg_transport
get_direct_mem_ptr
invalidate_direct_mem_ptr
behavior
SCML memory
Target
Dynamic timing annotation as part of user-defined behavior
Override default behavior for register access

Re-using TLM Peripheral Models



- TLM2.0 is coding style and abstraction level agnostic
- Separation of behavior, communication and timing
- Re-use TLM peripheral models for multiple design tasks
- Modular and compositional modeling of timing
- Supported by standards based SystemC Modeling Library

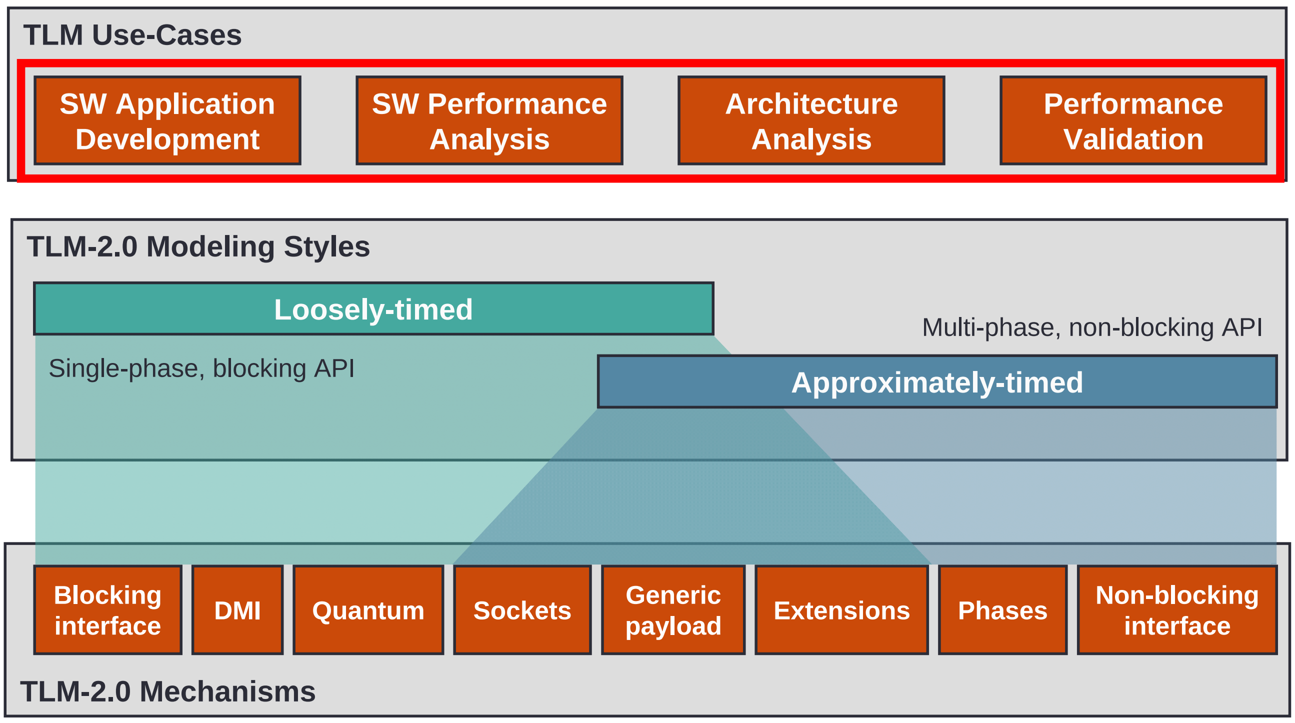
CoWare The ESL Design Leader

Outline

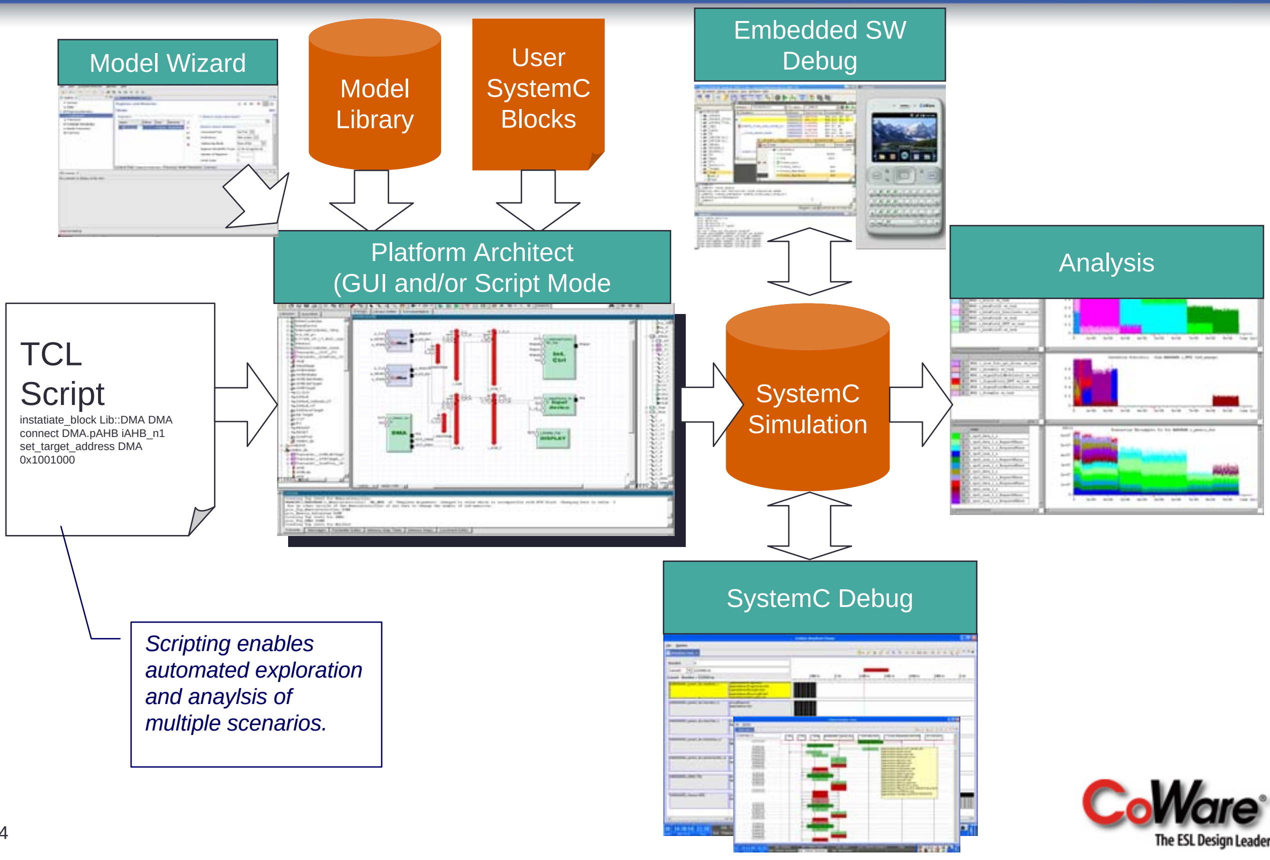
- TLM-2.0 Standard Overview
- Effective Creation of TLM-2.0 Peripheral Models
- Creating TLM-2.0 based Virtual Platforms
  - Loosely Timed virtual platforms for software development
  - Approximately Timed virtual platforms for architecture design

CoWare The ESL Design Leader

TLM-2.0 Overview



ESL Design Tools



```
instatiate_block Lib::DMA DMA
connect DMA.pAHB iAHB_n1
set_target_address DMA
0x1001000
```

Scripting enables automated exploration and anaylsis of multiple scenarios.

CoWare Ecosystem

EDA and FPGA
Mentor Graphics, cadence, Synplicity, SYNOPSYS, FORTE DESIGN SYSTEMS, eve, NOVAS, Javelin, chipvision The Low Power Specialists, ESTEREL Technologies, carbon design systems, JEDA Technologies, XILINX, ACE, ALTERA, Synfora, criticalblue

ESW
LAUTERBACH, altia, Green Hills SOFTWARE, INC., WIND RIVER, ARM, montavista, Microsoft Windows Embedded Partner, eclipse

IP
VeriSilicon, MeP, CEVA, IBM, ARM, MIPS TECHNOLOGIES, SONICS smart interconnect IP, Arteris, Silistix, tensilica, denali Memory solutions defined., ARC, TEXAS INSTRUMENTS

CoWare

Training
WILLAMETTE HDL language based design experts, hdlLab, ESLX, DOULOS

Standards
The SPIRIT Consortium, OCP International Partnership, SYSTEM C

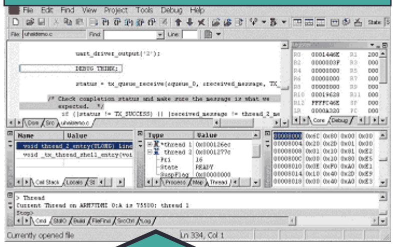
DSP Tools
The MathWorks, TEXAS INSTRUMENTS, Catalytic INC.

Services
HCL TECHNOLOGIES HCL, BDTi, Newfield

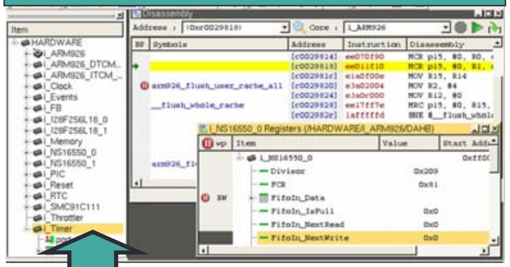
CoWare The ESL Design Leader

Software Application Development

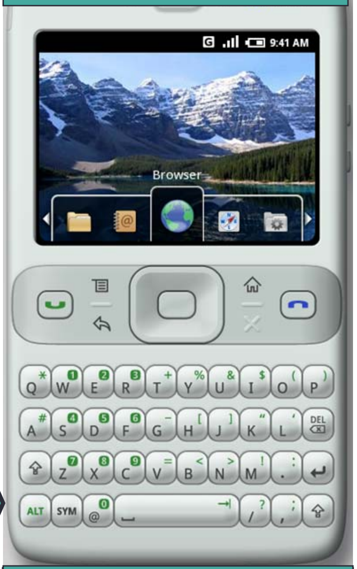
Software Debugger



Virtual Platform Analyzer



Keypad/Display Device



Requirements

- Sufficient simulation speed (10-50% real-time)
- Functional completeness and register accuracy
- Timing accuracy: software synchronization
- Controllability and observability
- Integration with Software IDEs
- External connectivity

SystemC TLM-2.0 based Virtual Platform

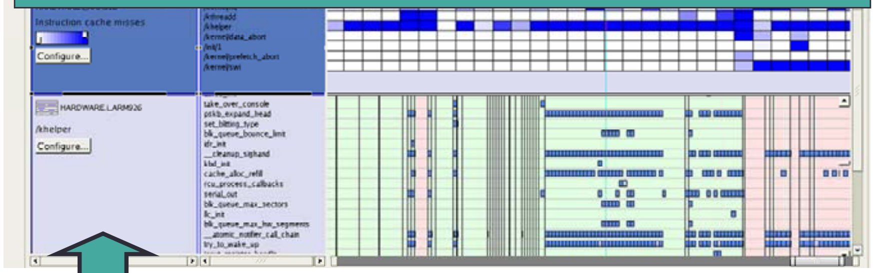
Console

```
init: mtd partition -1,
init: mtd partition 0, "system"
eth0: link up
sh: can't access tty; job control turned off
# binder_open(c0849300 c0837240) (pid 442) got c5c2f000
binder_open(c0849300 c0a45e20) (pid 458) got c5a95000
android_power: auto off timeout set to 604800 seconds
binder_open(c0849300 c38da340) (pid 509) got c3d9d000
binder_open(c0849300 c0a26160) (pid 508) got c3163000
binder_open(c0849300 c38daa40) (pid 522) got c326f000
```

CoWare The ESL Design Leader

Software Performance Analysis

Software Performance Analysis

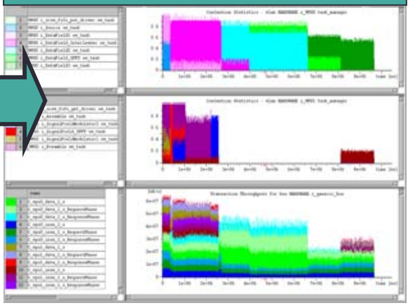


Requirements

- Sufficient simulation speed (1-10% real-time)
- Functional completeness and register accuracy
- Timing accuracy: 80% (interval: ~100k cycles)
- Hardware and software performance analysis views
- External connectivity

SystemC TLM-2.0 based Virtual Platform

Hardware Performance Analysis



A Real SystemC based TLM Platform

- Results based on CoWare's pre-TLM-2.0 SystemC TLM Environment
- Platform originally modeled at PV for Application SW Development
  - 55 unique models (95 instances)
  - Runs the actual, unmodified software for the phone
- Updated platform reuses TLM peripheral models with timing information in the memory sub-system for SW Performance Analysis
  - 4 models within memory sub-system enabled with timing annotation

	Silicon	CoWare VP (at PV)	CoWare VP (w/ PV+T)
Phone OS Booted	2 sec	20 sec	31 sec
GSM Network Registration	8 sec	66 sec	476 sec
Idle execution	1x	3.5x	3.5x
Accuracy	100%	50%	85-99%

Architecture Analysis

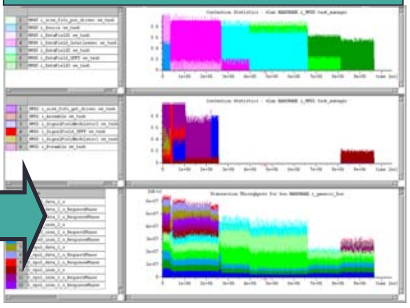
- Workload modeling options:
  - Trace-driven File Reader Bus Master
  - Task-graph driven Virtual Processing Unit
- Using partial virtual platforms and non-functional workload models
  - Reduced effort to capture platform
  - Requires profiling information, but porting of real SW not required
  - Ideal for performance optimization of SoC backbone (interconnect/memory)

Requirements

- Sufficient simulation speed (100-1000 x RTL)
- Cycle-accurate models of critical components
  - Interconnect, memory subsystem
- Same level of configurability as real IP
- Timing accuracy: 95% (interval: 1-10 cycles)
- Hardware performance analysis views

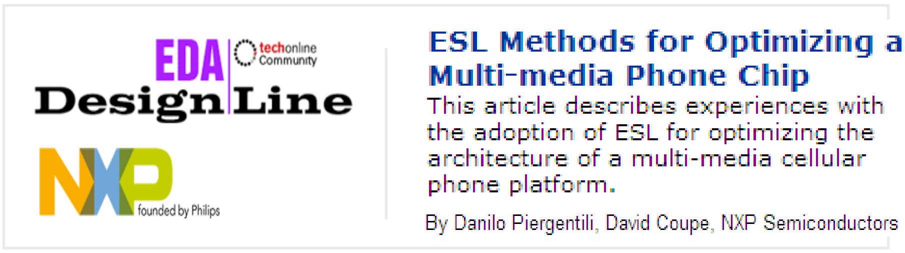
SystemC TLM-2.0 based Virtual Platform

Hardware Performance Analysis



CoWare The ESL Design Leader

Example: NXP



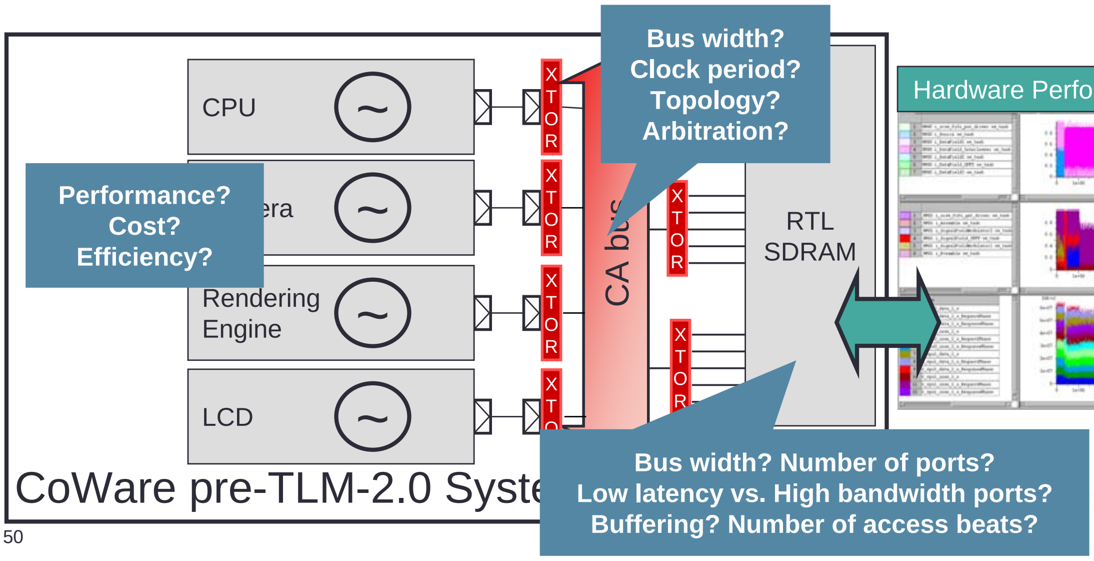
EDA DesignLine techonline Community

NXP founded by Philips

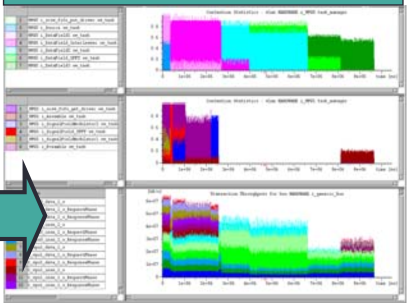
ESL Methods for Optimizing a Multi-media Phone Chip

This article describes experiences with the adoption of ESL for optimizing the architecture of a multi-media cellular phone platform.

By Danilo Piergentili, David Coupe, NXP Semiconductors


CPU
era
Rendering Engine
LCD
XTOR
CA bus
RTL SDRAM
Bus width? Clock period? Topology? Arbitration?
Performance? Cost? Efficiency?
Bus width? Number of ports? Low latency vs. High bandwidth ports? Buffering? Number of access beats?
CoWare pre-TLM-2.0 Syste

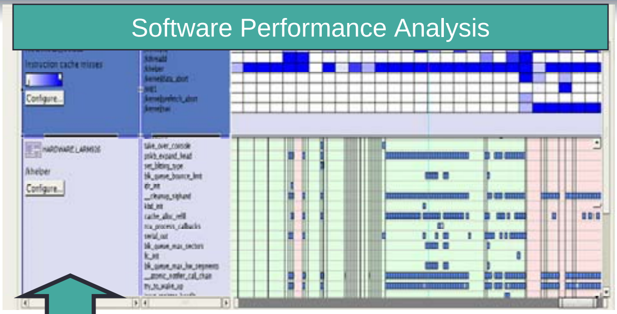
Hardware Performance Analysis



CoWare The ESL Design Leader

Performance Validation

Software Performance Analysis



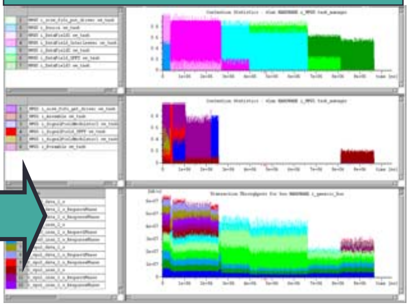
- Using complete virtual platforms and cycle-accurate ISSes running real SW
  - Realistic performance results from execution of real SW
  - Modeling effort of cycle-accurate IP can be mitigated by means of RTL co-simulation, Co-emulation, or synthesis of fast SystemC models from RTL using Carbon

Requirements

- Sufficient simulation speed (50-500 x RTL)
- Cycle-accurate models of critical components
  - Processor, interconnect, memory subsystem
- Functional completeness and register accuracy
- Timing accuracy: 95% (interval: 1-10 cycles)
- Hardware and software performance analysis views

SystemC TLM-2.0 based Virtual Platform

Hardware Performance Analysis



Summary

What does TLM-2.0 enable for ESL Users?

- **Well defined Use-cases, Modeling Styles, and TLM APIs**
  - **Model interoperability**
  - ⇒ **Model availability**
- **High speed simulation for SystemC based Virtual Platforms**
  - **Temporal decoupling, Direct Memory Interface, synchronization on demand**
- **Model re-use for multiple ESL design tasks**
  - **LT models interoperate with and can be refined to AT models**
  - **LT and AT models can be connected to cycle accurate models by means of transactors**

Thank You!