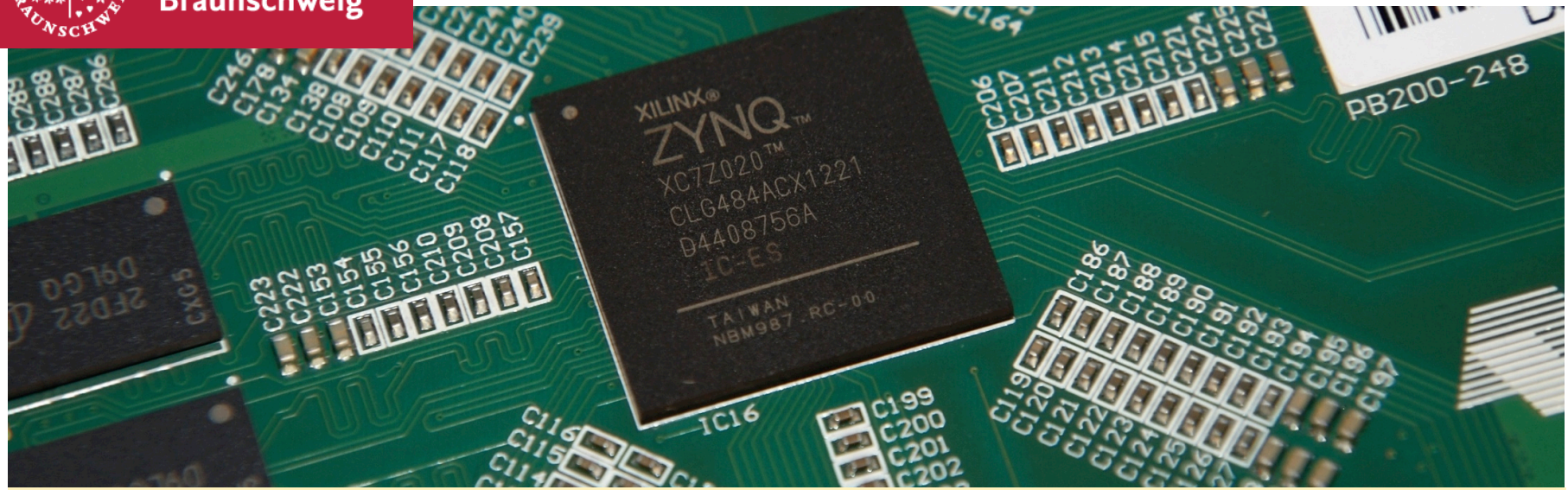
Technische Universität Braunschweig

Chair for Chip Design for Embedded Computing



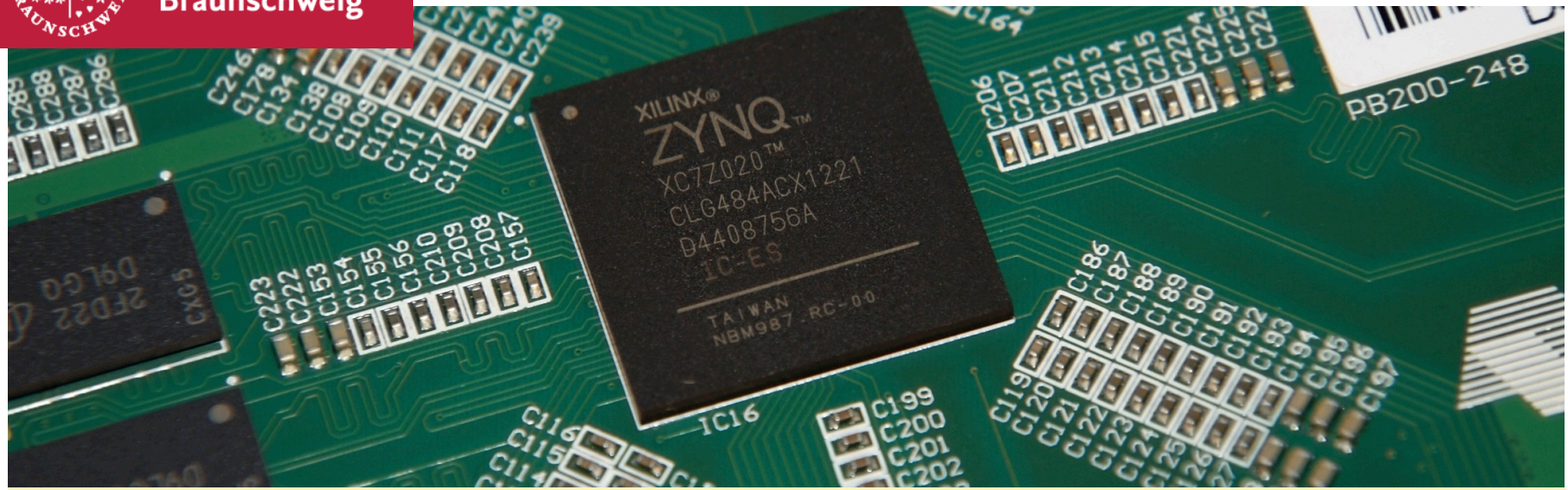
SoCRocket 101

HW/SW Co-Design for Space Applications

Bastian Farkas, Rolf Meyer

Technische Universität Braunschweig

Chair for Chip Design for Embedded Computing



SoCRocket 101

SoCRocket is available online:

https://socrocket.github.io

SoC Rocket

Outline

Overview

- **Why do we need Virtual Platforms?**
- Where do Virtual Platforms fit in?
- How do we build Virtual Platforms?

SoCRocket: a framework solution

- The Motivation
- Tools & Techniques
- Modeling of SoCRocket IP
- High-Level DSE demonstration
- Current activities

Demo

The space domain challenge

Embedded Systems in the Space Domain

High costs and risks require conservative design methods

source: upv.es

Rugged against vibrations and physical stress

Shielded against radiation

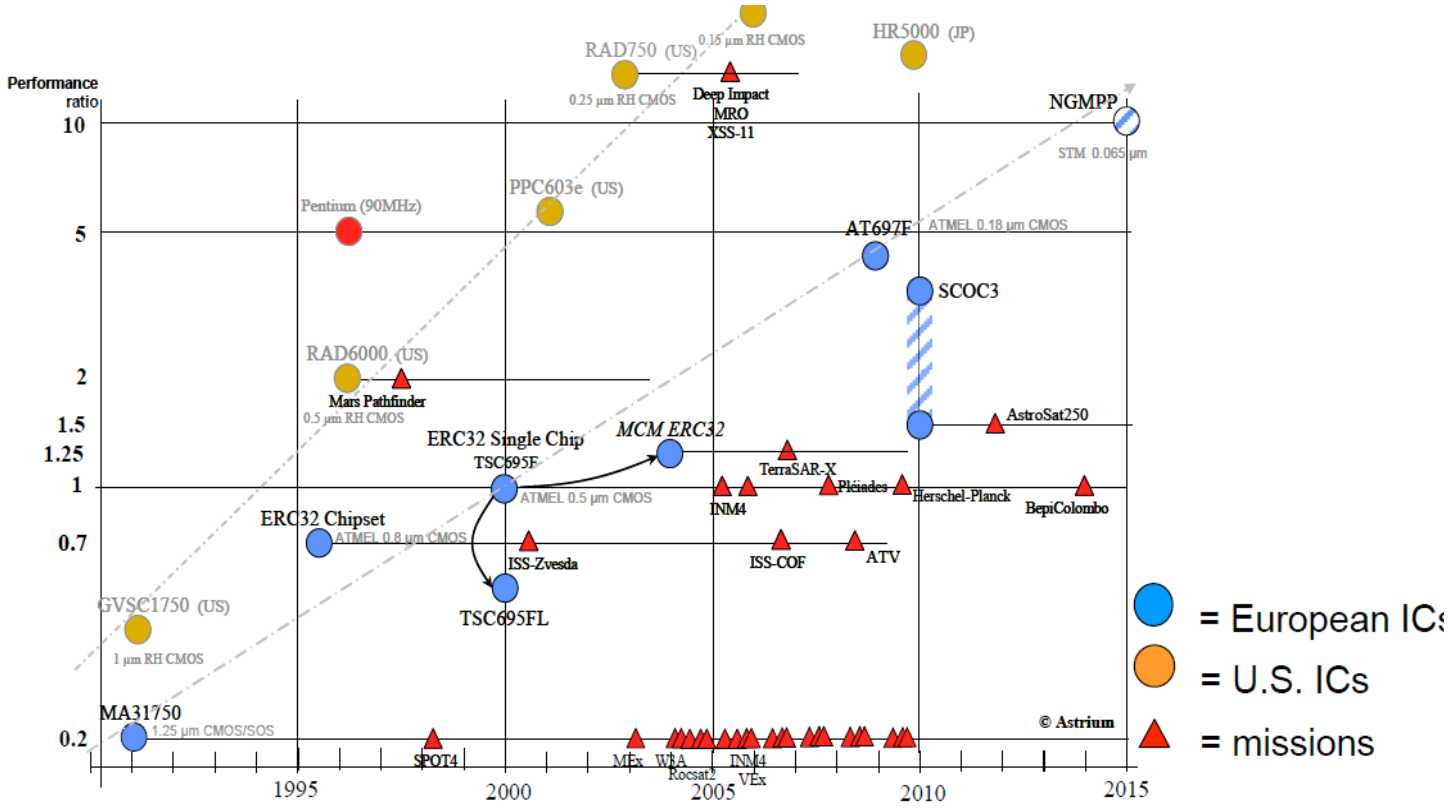
source: blog.ametsoc.org

Fail safe, durable, low maintenance

Electronics must pass complex verification and certification process

Performance Trends

Data-systems and Onboard Computers



source: European Space Agency: Technology Dossier on Data Systems and Onboard Computers, TEC-EDD/2011.109/GM, 2011

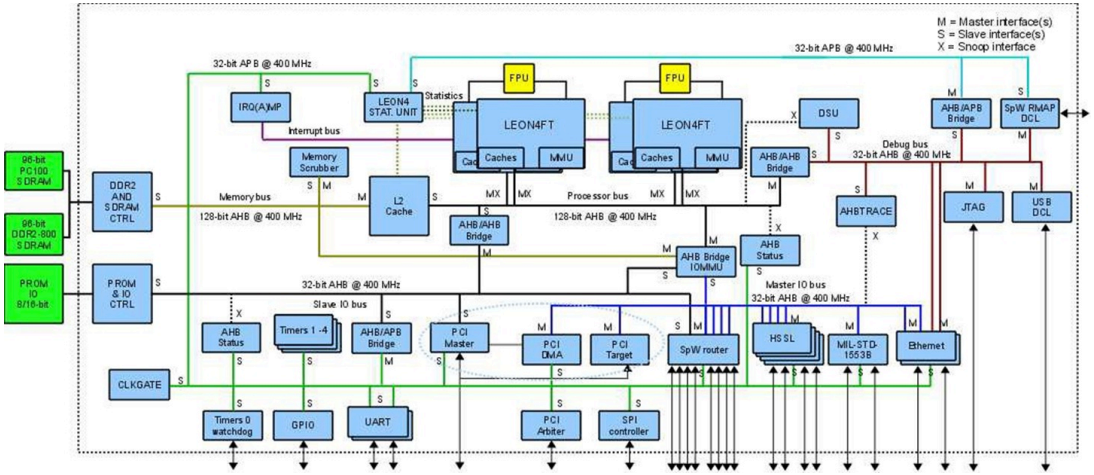
Conservative, secure and reliable design process results in performance gaps compared to non space centric technology

The performance increases

So the complexity will increase too

1. Introduction of commercial off-the-shelf components (**COTS**) by radiation hardness on board level

2. **Incremental optimization** of existing HW/SW Systems



source: Anderson, Hjorth, Habinc, Gaisler: Development of quad-core NGMP space processor, DASIA

(Multi-Processor SoCs – e.g. NGMP)

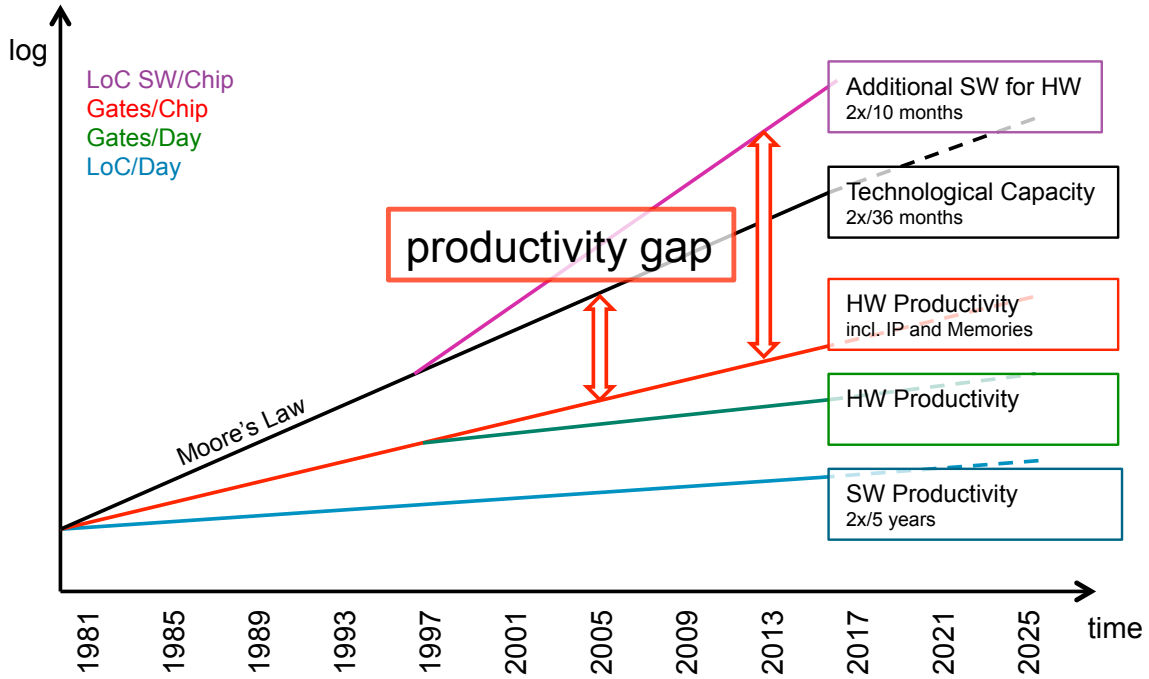
~1GFLOP @ 10 Watts

3. **Research and development of new Architectures** and technologies (e.g. MacSpace 64x Core CPU – 12.8 GFLOP)

Big increase in complexity makes initial design at RT-Level not efficient

The Productivity Gap

And the attempt to close it by Electronic System Level Design



source: ITRS Roadmap 2009

Electronics System Level Design (ESL-D)

"is the use of abstraction for improving the understanding of a system and raising the probability of its successful implementation in a cost-efficient manner."

Martin Bailey
Source: ESL Design and Verification

Outline

Overview

- ✓ Why do we need Virtual Platforms?
- **Where do Virtual Platforms fit in?**
- How do we build Virtual Platforms?

SoCRocket: a framework solution

- The Motivation
- Tools & Techniques
- Modeling of SoCRocket IP
- High-Level DSE demonstration
- Current activities

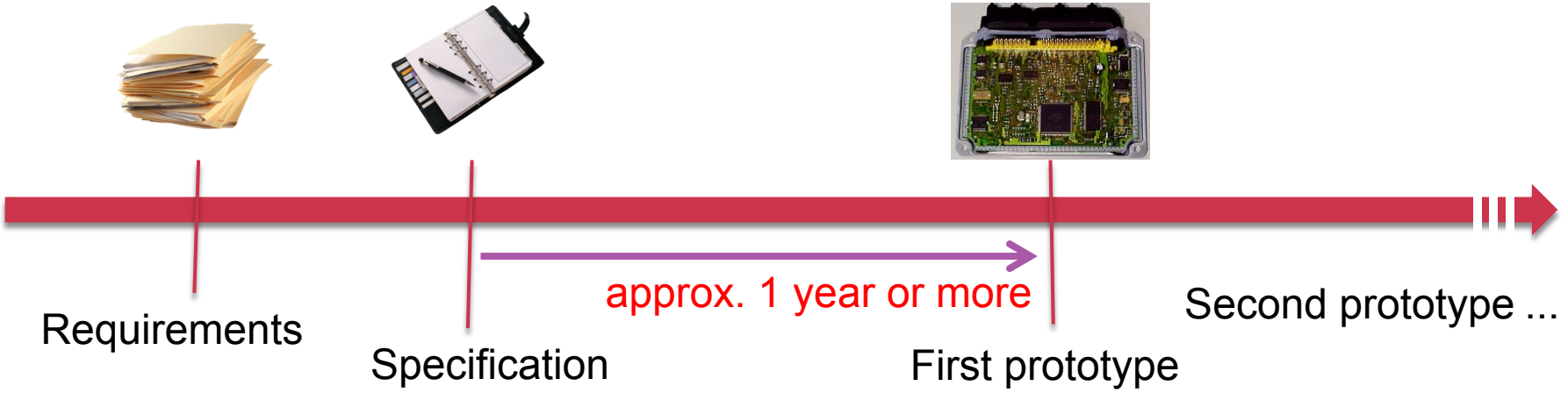
Demo

Abstract Hardware: Virtual Platforms

Vision: Integrated aero space development process

Goals:

- Enable dedicated software development and system validation **before** any HW-Prototype becomes available
- Enable aero space specific system hardening and integration ahead of the prototype stage



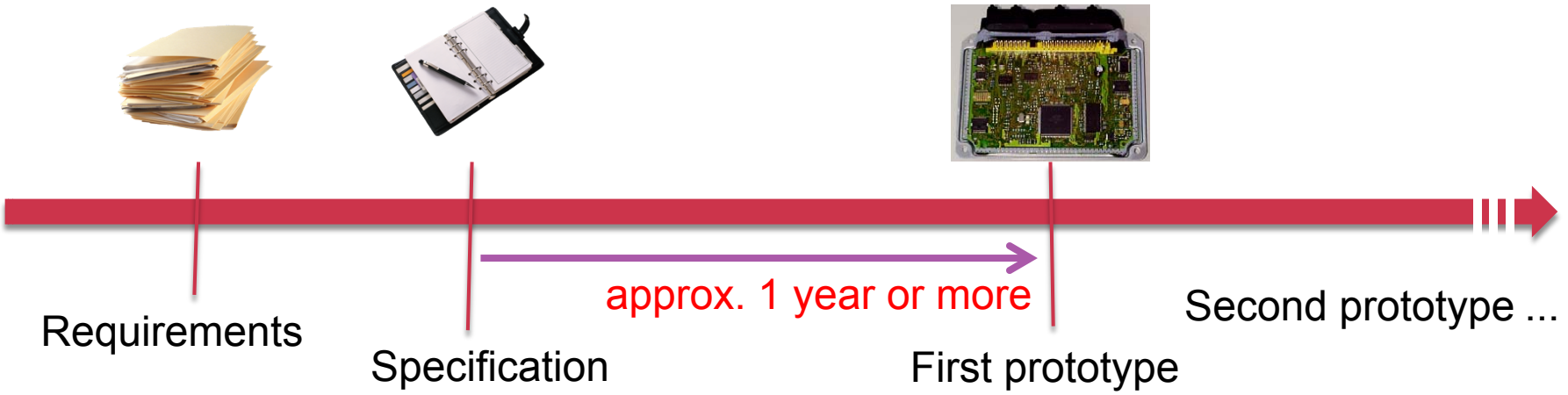
Abstract Hardware: Virtual Platforms

Vision: Integrated aero space development process

Separation of technical and functional integration

- Technical Integration: Securing ability to run
- Functional Integration: Checking functionality of subsystems

Virtual Prototyping: Early availability of an „executable Specification“ which can be distributed between all partners



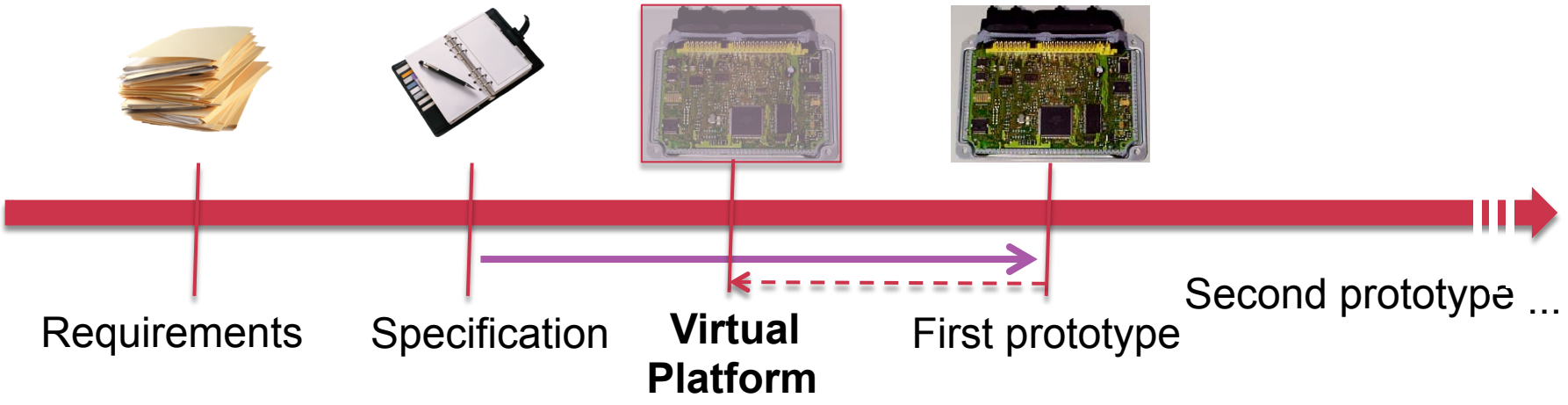
Abstract Hardware: Virtual Platforms

Vision: Integrated aero space development process

Separation of technical and functional integration

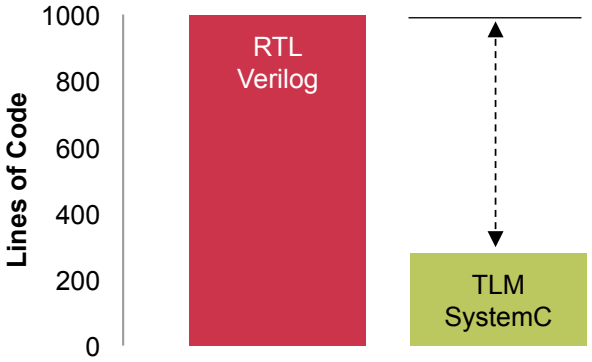
- Technical Integration: Securing ability to run
- Functional Integration: Checking functionality of subsystems

Virtual Platform: Early availability of an „executable Specification“ which can be distributed between all partners

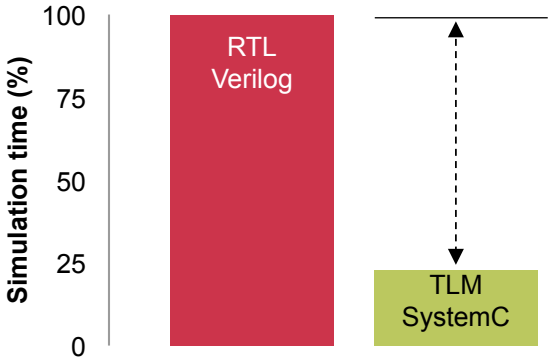


Virtual Platform in numbers

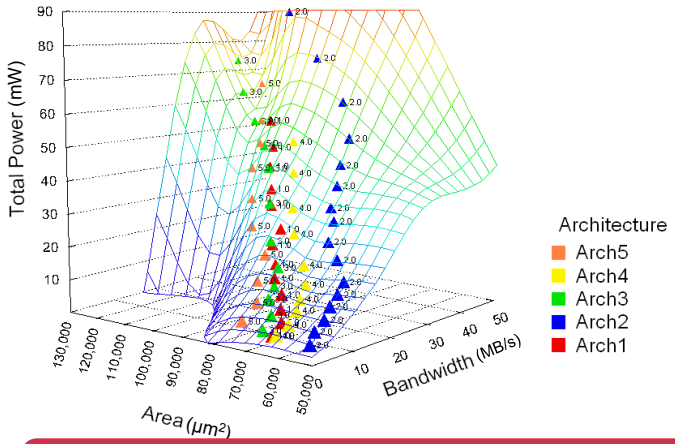
SystemC/TLM based module design accelerates production



~3 x smaller Designs: More efficient

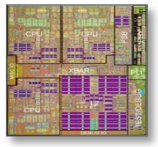


5 – 10 x faster Verification



Faster Design-Space-Exploration:
"Is it possible to run with 200 MHz?"

SystemC → RTL in **~10 days.**
vs. only RTL in **~3 month**



I/F-Controller-Module

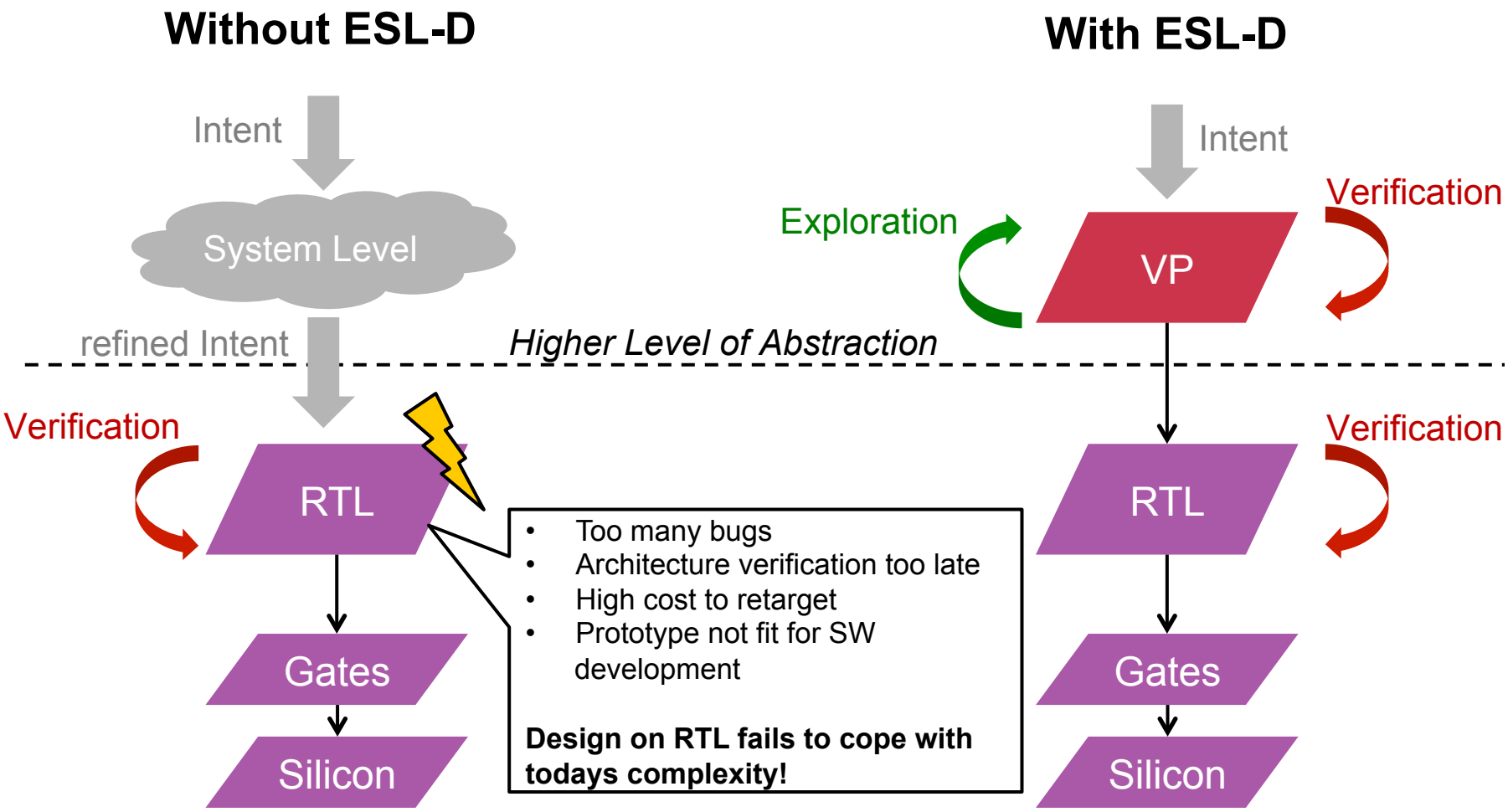
Controller

>10 x productivity increase

source: Cadence

Where does it fit in?

Virtual Platforms in Hardware design



source: Cadence

Outline

Overview

- ✓ Why do we need Virtual Platforms?
- ✓ Where do Virtual Platforms fit in?
- • **How do we build Virtual Platforms?**

SoCRocket: a framework solution

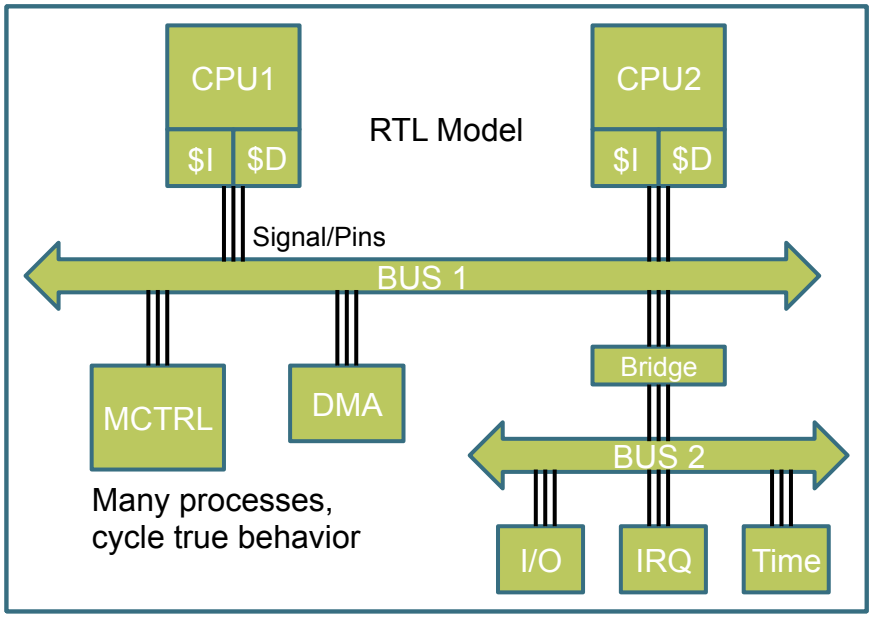
- • The Motivation
- • Tools & Techniques
- • Modeling of SoCRocket IP
- • High-Level DSE demonstration
- • Current activities

Demo

Virtual Platforms

Be fast and accurate if you need to be!

- Implementing Hardware in Software on an adequate abstraction level to fit a specific use case.



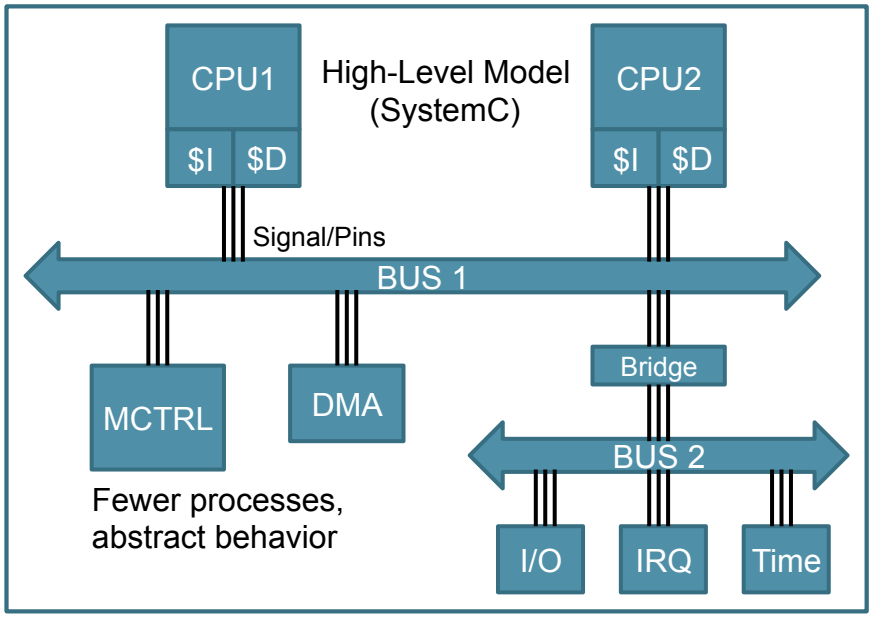
Use case	Needed simulation speed	Needed accuracy
Verification	1 kHz	100%
Architecture exploration, analysis of speed and throughput	~100 kHz	80%-90%
Software development	>10 MHz	Enough to boot an OS

Default tools to implement Virtual Platforms are **SystemC** and **TLM2.0**.

Virtual Platforms

Be fast and accurate if you need to be!

- Implementing Hardware in Software on an adequate abstraction level to fit a specific use case.



Use case	Needed simulation speed	Needed accuracy
Verification	1 kHz	100%
Architecture exploration, analysis of speed and throughput	~100 kHz	80%-90%
Software development	>10 MHz	Enough to boot an OS

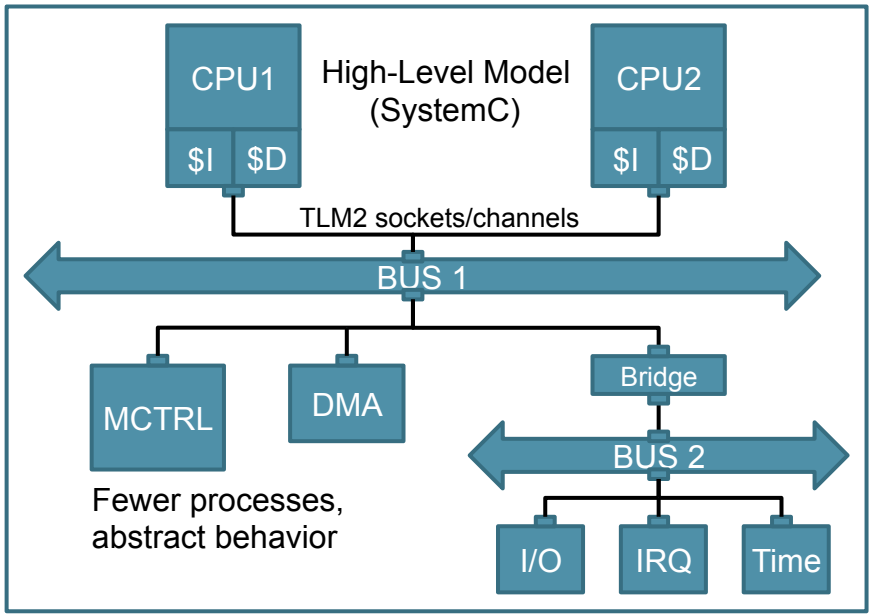
VHDL/Verilog SystemC/TLM

Default tools to implement Virtual Platforms are **SystemC** and **TLM2.0**.

Virtual Platforms

Be fast and accurate if you need to be!

- Implementing Hardware in Software on an adequate abstraction level to fit a specific use case.

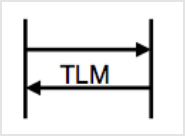


Use case	Needed simulation speed	Needed accuracy
Verification	1 kHz	100%
Architecture exploration, analysis of speed and throughput	~100 kHz	80%-90%
Software development	>10 MHz	Enough to boot an OS

Default tools to implement Virtual Platforms are **SystemC** and **TLM2.0**.

SystemC / TLM 2.0

The basic/standard Virtual Platform infrastructure



Use Cases

- Software-Development
- Software-Performance
- Architecture-Analysis
- Hardware-Verification

Coding Styles

- Loosely-Timed (LT): blocking, function calls, single-phased
- Approximately-Timed (AT): non-blocking, scheduled, multi-phased

Mechanisms

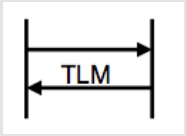
- Blocking Interfaces
- DMI
- Quantum
- Sockets
- Generic Payload
- Phases
- Non-blocking Interfaces

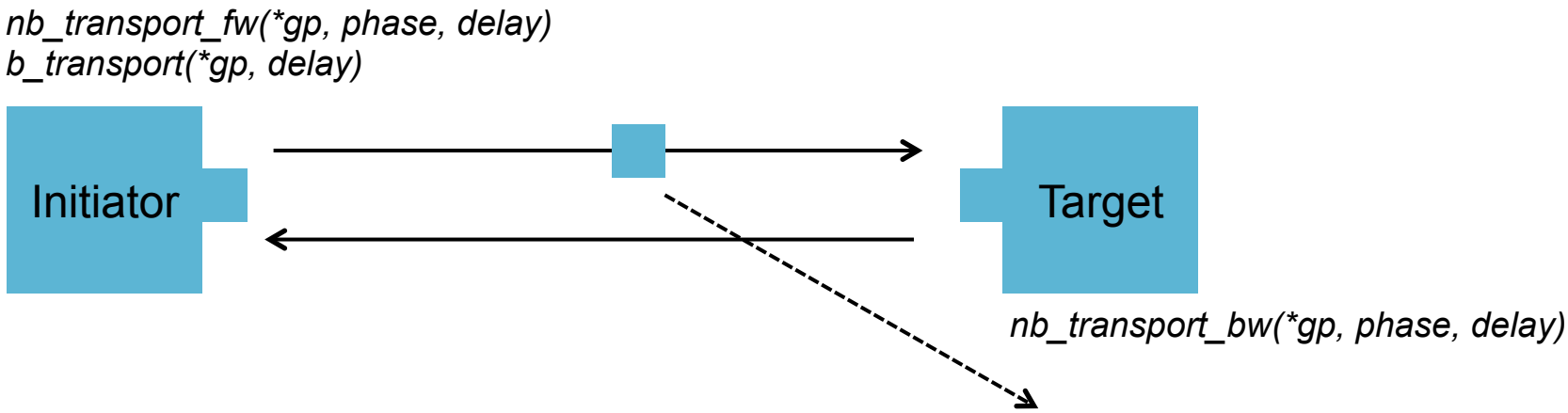
Source: www.osci.org (2011)

TLM: Abstraction of communication using function calls.

source: John Aynsley, Doulos, http://www.doulos.com/knowhow/systemc/tlm2

TLM
Transaction Level Modeling





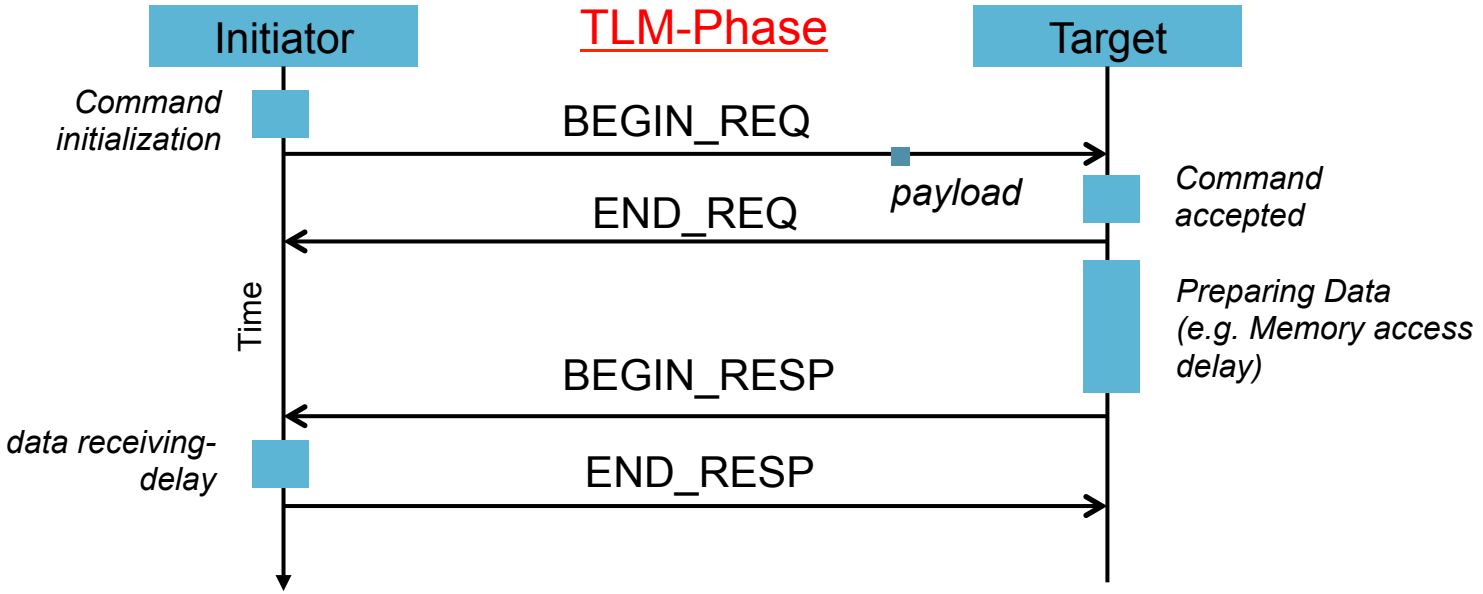
- System consists of initiators and targets
- TLM2.0 includes default *Sockets* and *Channels* to model commination processes
- Data transfer is implemented with pointers referencing *payload* objects
- Depending on the abstraction level the communication can be blocking or non-blocking

Payload-Object

command
address
data
byte enables
streaming
response status
tlm_phase

TLM

TL-Communication / non-blocking (AT)

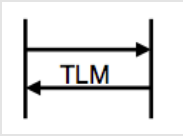


- TLM default protocol can handle up to 4 synchronization points
- Sufficient to model simple memory mapped systems accurately

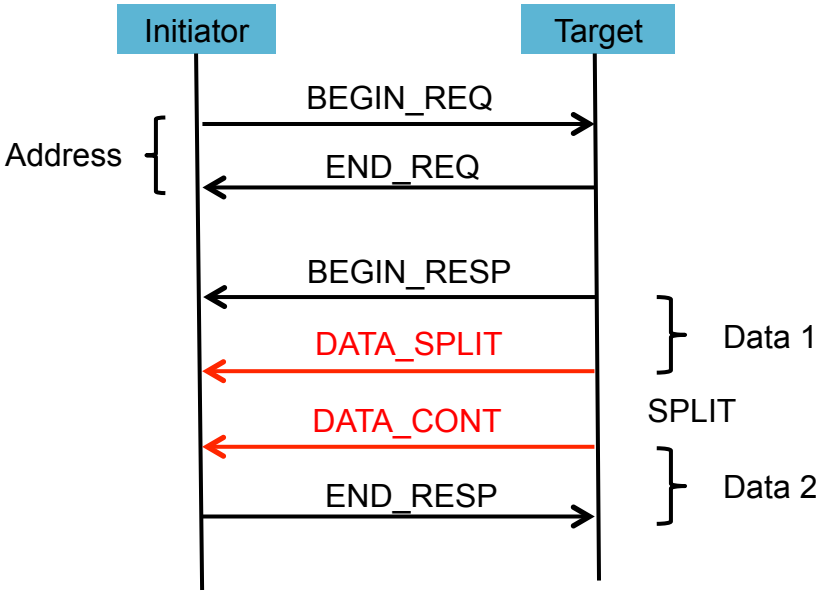
→ **But modeling of higher protocol complexity requires further flexibility**

TLM

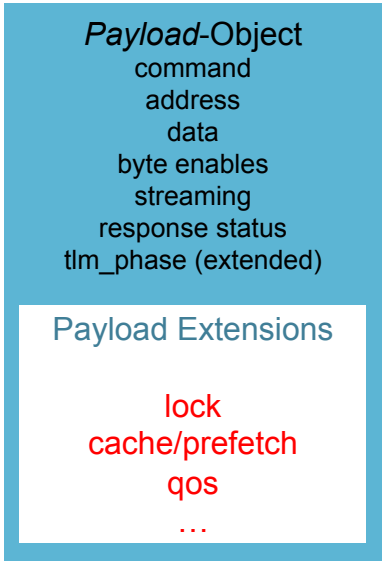
TL-Communication / Increasing simulation accuracy



Extending the protocol:



Extending the payload:



Best practice: Use the default protocol – make *Payload* extensions optional!

Outline

Overview

- ✓ Why do we need Virtual Platforms?
- ✓ Where do Virtual Platforms fit in?
- ✓ How do we build Virtual Platforms?

SoCRocket: a framework solution

- **The Motivation**
- Tools & Techniques
- Modeling of SoCRocket IP
- High-Level DSE demonstration
- Current activities

Demo

Motivation

Industry adoption of ESL-D
(with minor exceptions) is still hesitant:

1. High Costs for migration of workflows and introduction of new tools

Motivation

Industry adoption of ESL-D
(with minor exceptions) is still hesitant:

1. High Costs for migration of workflows and introduction of new tools

Goal of SoCRocket:

- *Development of open tools and models to design embedded systems for the aero space domain*

Cost reduction	Efficiency / Applicability	Availability / Reliability	Acceptance

Motivation

Industry adoption of ESL-D
(with minor exceptions) is still hesitant:

2. Shortage of adequate personnel with expertise of abstract design methods

Cost reduction	Efficiency / Applicability	Availability / Reliability	Acceptance

Motivation

Industry adoption of ESL-D
(with minor exceptions) is still hesitant:

2. Shortage of adequate personnel with expertise of abstract design methods

Goal of SoCRocket:

- *Simplification and acceleration of model and virtual prototype design through encapsulation of complexity and shaping of easy-to-use APIs*

Cost reduction	Efficiency / Applicability	Availability / Reliability	Acceptance

Motivation

Industry adoption of ESL-D
(with minor exceptions) is still hesitant:

3. A lack of reliable simulation models with support for multiple abstraction layers

Cost reduction	Efficiency / Applicability	Availability / Reliability	Acceptance

Motivation

Industry adoption of ESL-D
(with minor exceptions) is still hesitant:

3. A lack of reliable simulation models with support for multiple abstraction layers

Goal of SoCRocket:

- *Composition of a SystemC/TLM design (or model) library with support for approximately accurate simulations (TLM AT); Demonstration of verification methods and tools.*

Cost reduction	Efficiency / Applicability	Availability / Reliability	Acceptance

Motivation

Industry adoption of ESL-D
(with minor exceptions) is still hesitant:

4. Limited support for reusable abstract models due to the absence of standards

Cost reduction	Efficiency / Applicability	Availability / Reliability	Acceptance

Motivation

Industry adoption of ESL-D
(with minor exceptions) is still hesitant:

4. Limited support for reusable abstract models due to the absence of standards

Goal of SoCRocket:

- *Identification and filling of „standardization gaps“ by open solutions*

Cost reduction	Efficiency / Applicability	Availability / Reliability	Acceptance

Outline

Overview

- ✓ Why do we need Virtual Platforms?
- ✓ Where do Virtual Platforms fit in?
- ✓ How do we build Virtual Platforms?

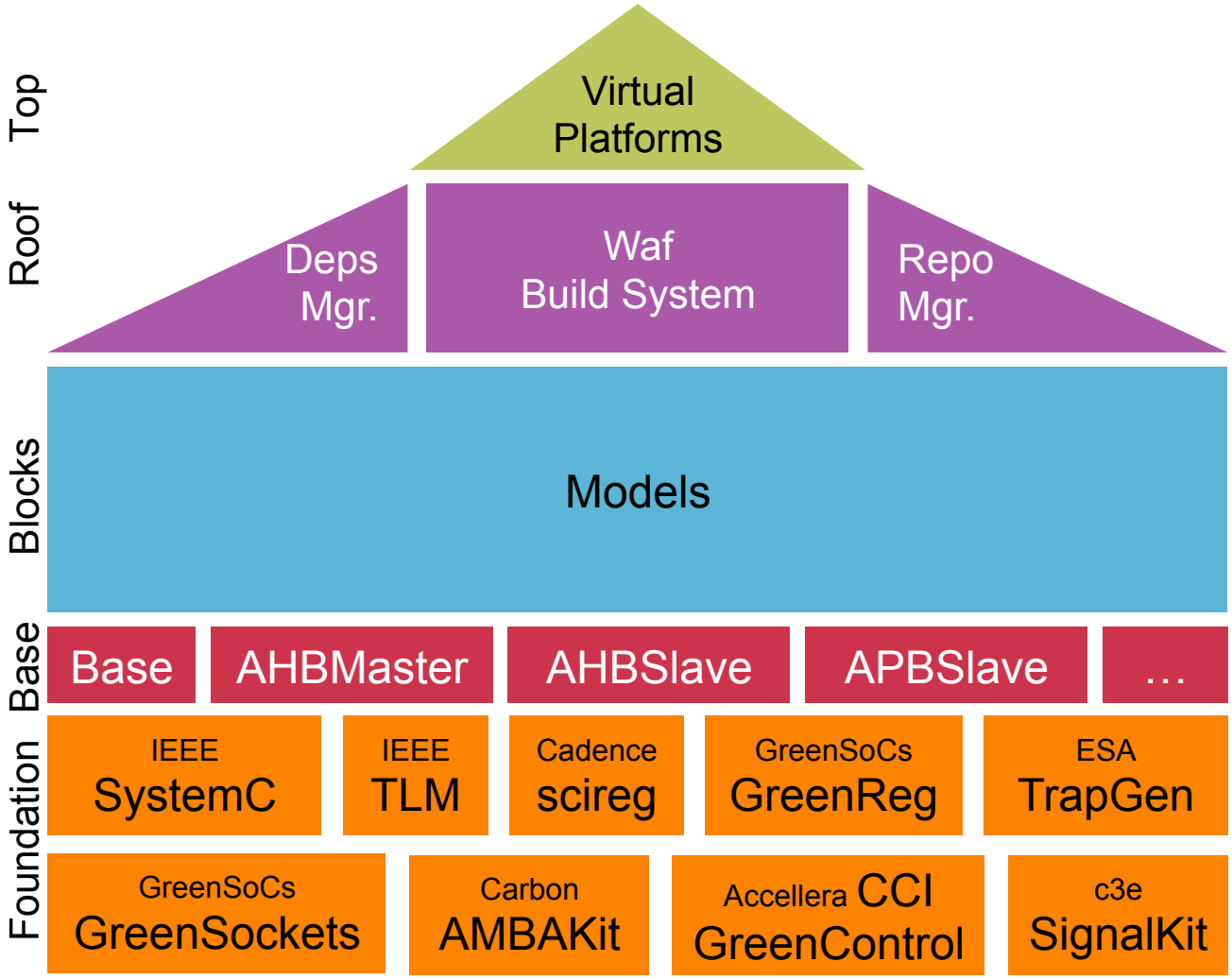
SoCRocket: a framework solution

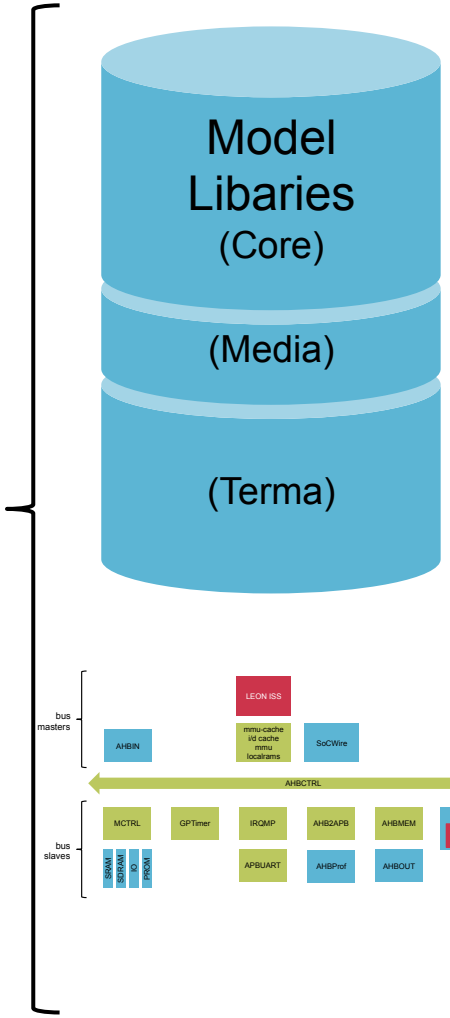
- ✓ The Motivation
- **Tools & Techniques**
- Modeling of SoCRocket IP
- High-Level DSE demonstration
- Current activities

Demo

SoCRocket - The building blocks

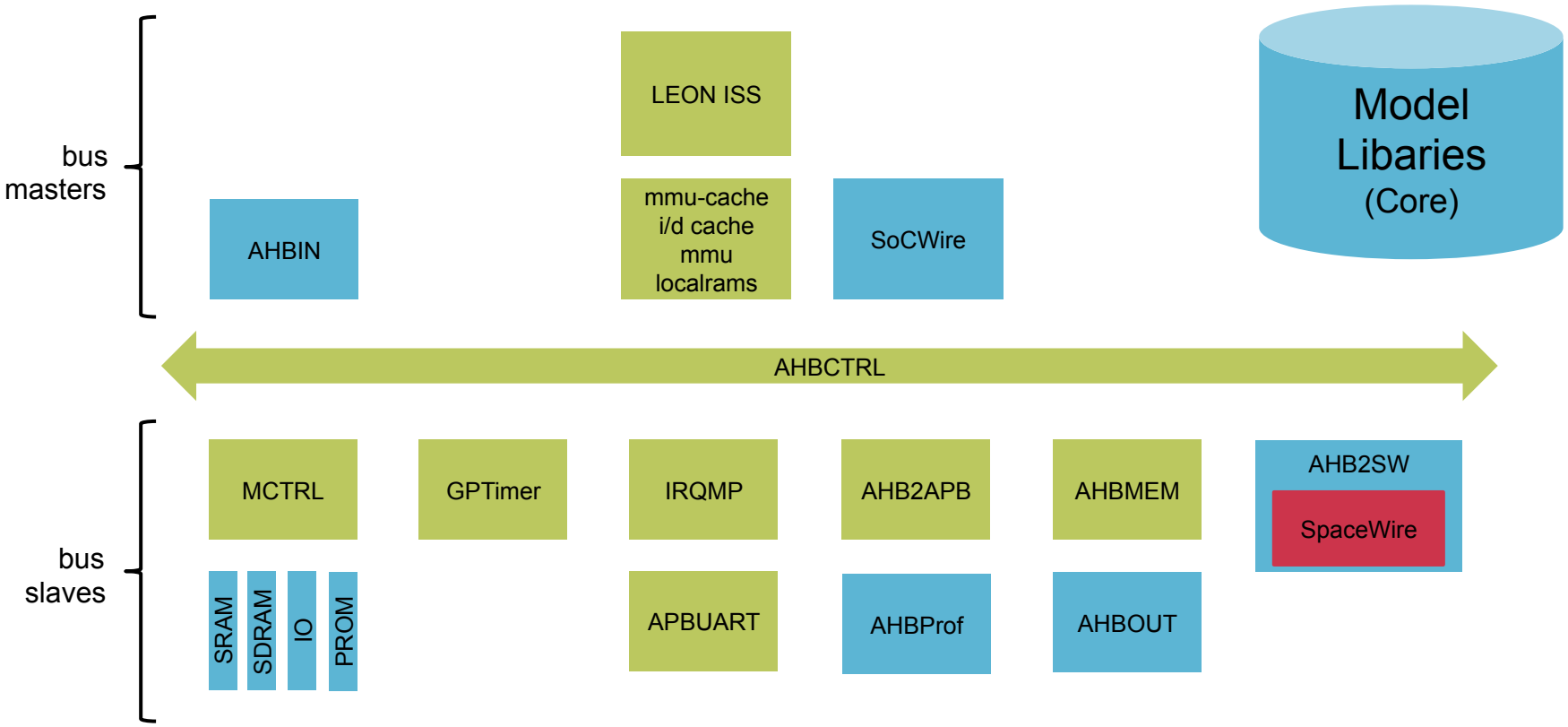
SoCRocket is more than a model library





Included SystemC Models (core)

All models where developed with RTL equivalents as blueprint



All models available in loosely timed (LT), and approximately timed (AT) flavor of TLM2.0.

- Models provided by ESA (integration only)
- Models with GRLIB reference

Waf build System

Make it the Python way

wscript based build system

- Python based
- One “wscript” for each directory in the design hierarchy
- Recursive module inclusion
- One python script for each dependency

Instead of “`./configure; make; make install`”
use: “`./waf configure; ./waf build; ./waf install`”
or `./waf configure build install`

Useful:

```
./waf --help
./waf list
./waf clean
./waf distclean
```

Source: http://lustich.de/bilder/autos/smart-kran/

Outline

Overview

- ✓ Why do we need Virtual Platforms?
- ✓ Where do Virtual Platforms fit in?
- ✓ How do we build Virtual Platforms?

SoCRocket: a framework solution

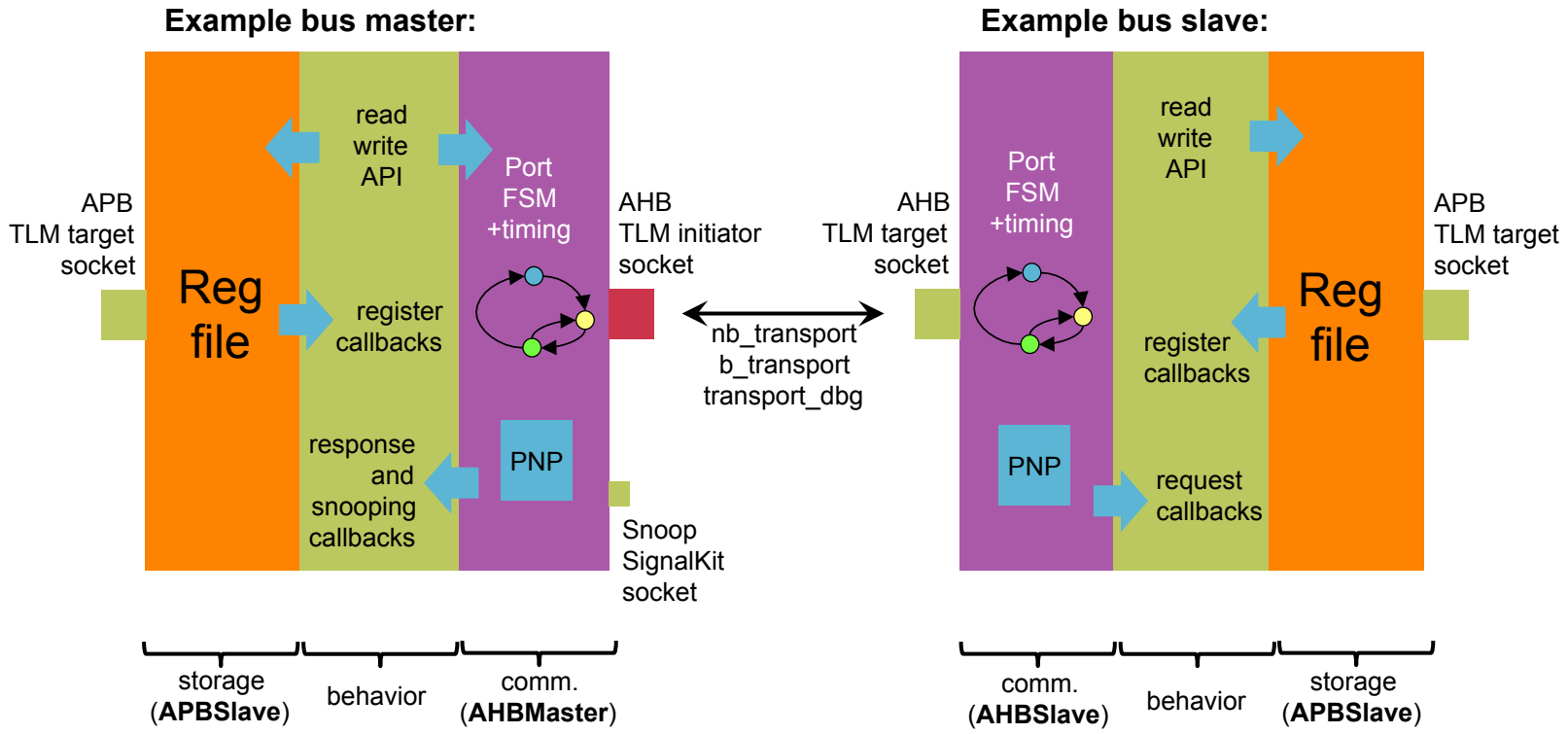
- ✓ The Motivation
- ✓ Tools & Techniques
- • **Modeling of SoCRocket IP**
- • High-Level DSE demonstration
- • Current activities

Demo

Modeling of SoCRocket IPs

Component design with base classes

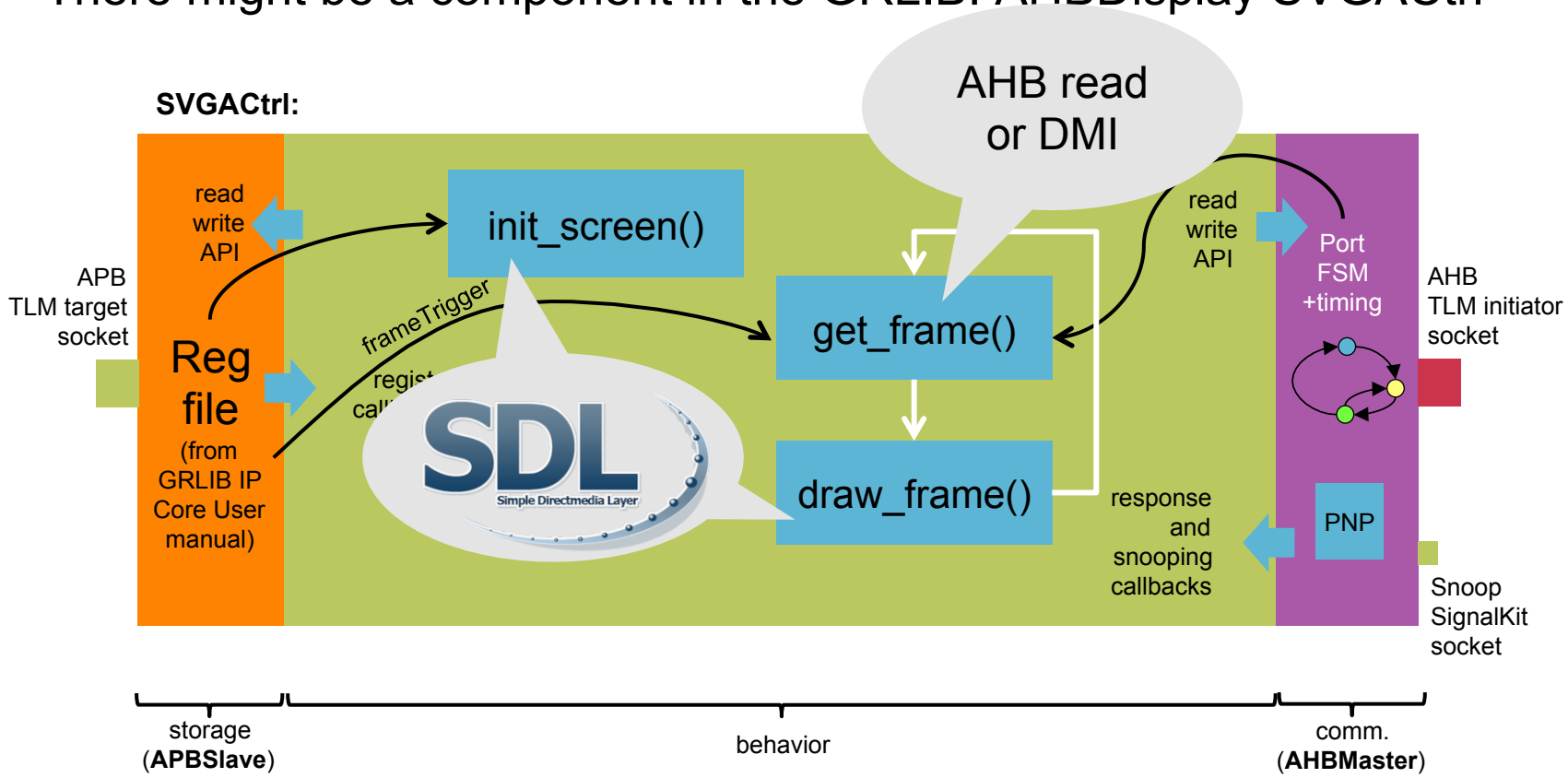
- Library base classes for AHB masters, AHB slaves and APB slaves
- Models inherit library base classes and register file



Modeling of SoCRocket IPs

For example an AHBDisplay

- A framebuffer display as AHBMaster
- There might be a component in the GRLIB: ~~AHBDisplay~~ SVGACtrl



Outline

Overview

- ✓ Why do we need Virtual Platforms?
- ✓ Where do Virtual Platforms fit in?
- ✓ How do we build Virtual Platforms?

SoCRocket: a framework solution

- ✓ The Motivation
- ✓ Tools & Techniques
- ✓ Modeling of SoCRocket IP
- • **High-Level DSE demonstration**
- • Current activities

Demo

Design Space Exploration Flow

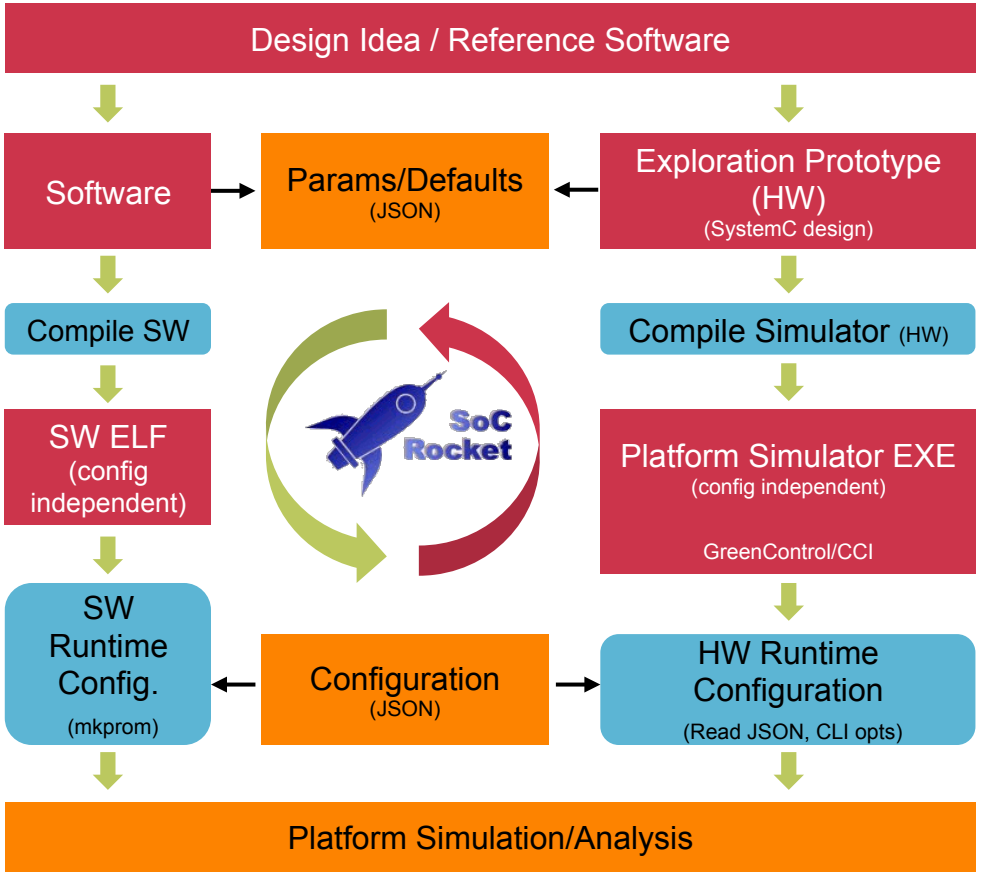
Runtime re-configuration

Path from Design Idea
to VP Prototype

1. HW/SW partitioning
2. Create Exploration Prototype
3. Compile HW & SW (config independent)
4. Extract config parameters
5. Configure HW & SW (configuration wizard)
6. Simulation/Analysis **Design goals met ?**

iterate

done



(From SoCRocket Design Flow Report)

High-Level DSE Demonstration

Exploration Space

A lossless multi spectral and hyper spectral image compression was mapped on the base platform.

(SW executed on top of RTEMS)

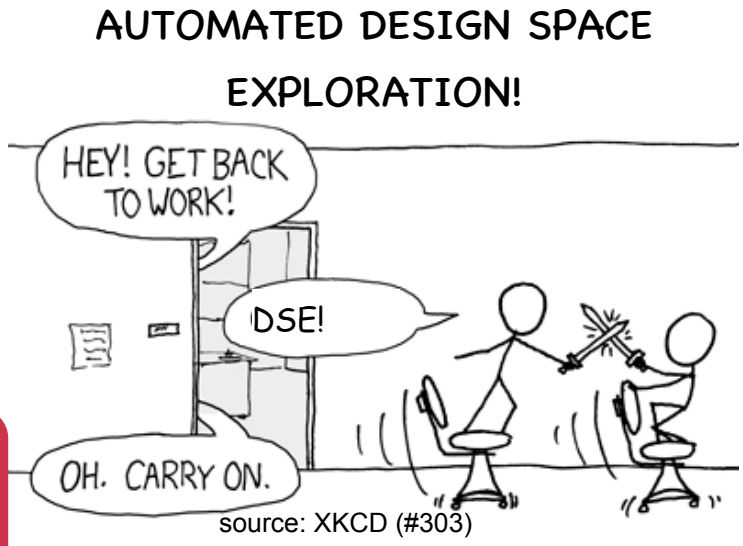
Explored configurations

- Number of instruction cache sets (1 .. 4)
- Capacity per icache set (1kB .. 8kB)
- Number of data cache sets (1 .. 4)
- Capacity per dcache set (1kB .. 8kB)
- Number of CPUs (2, 4, 6)

Optimization parameters

- Runtime (simulated)
- Total power consumption

1280 configurations



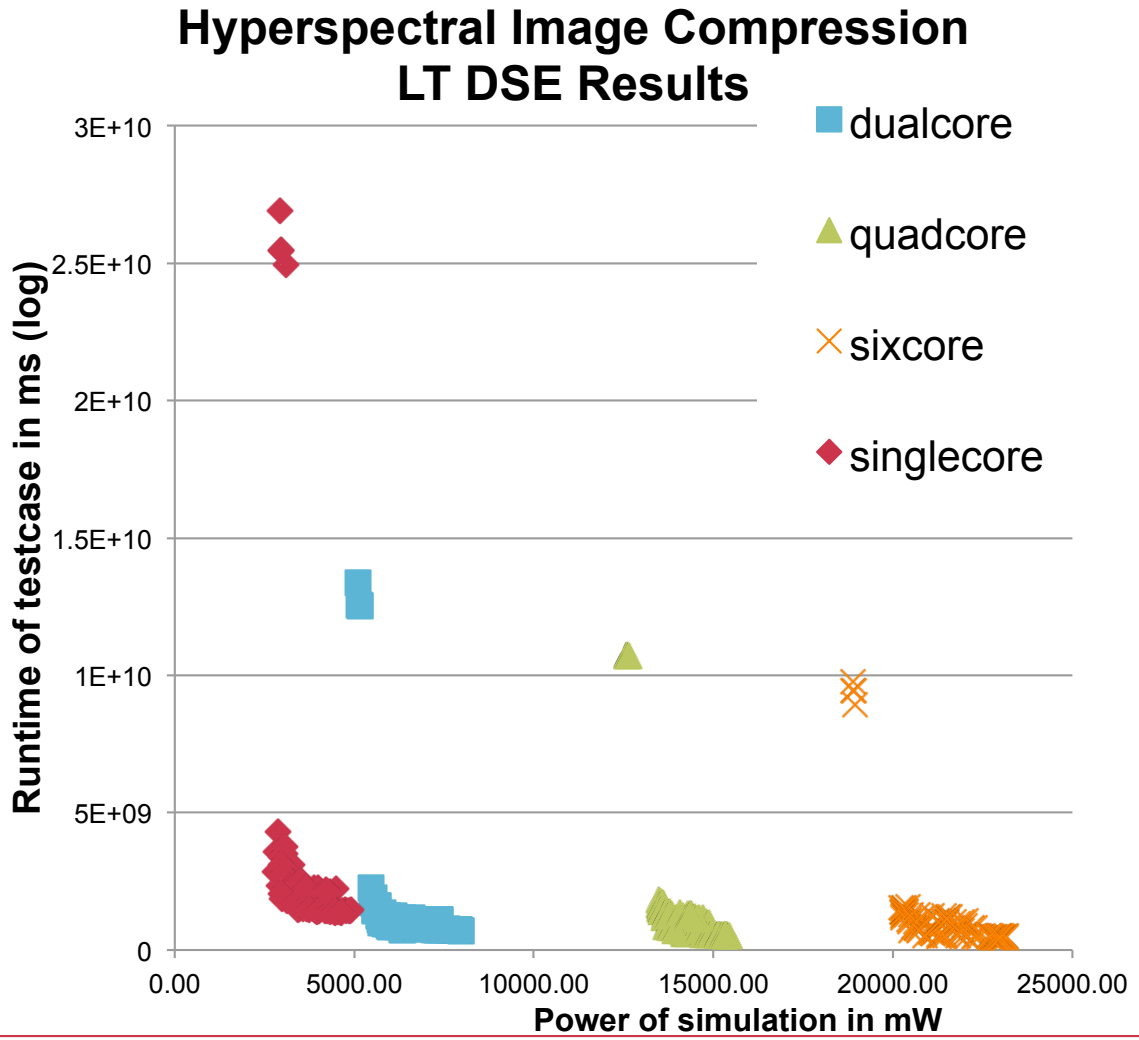
source: XKCD (#303)

High-Level DSE Demonstration

Energy vs. Performance Trade-off

LT High level analysis

- Configurations: 1280
- Total simulation duration: 16h
- Simulations in parallel: 16
- Av. simulation duration: 45s



High-Level DSE Demonstration

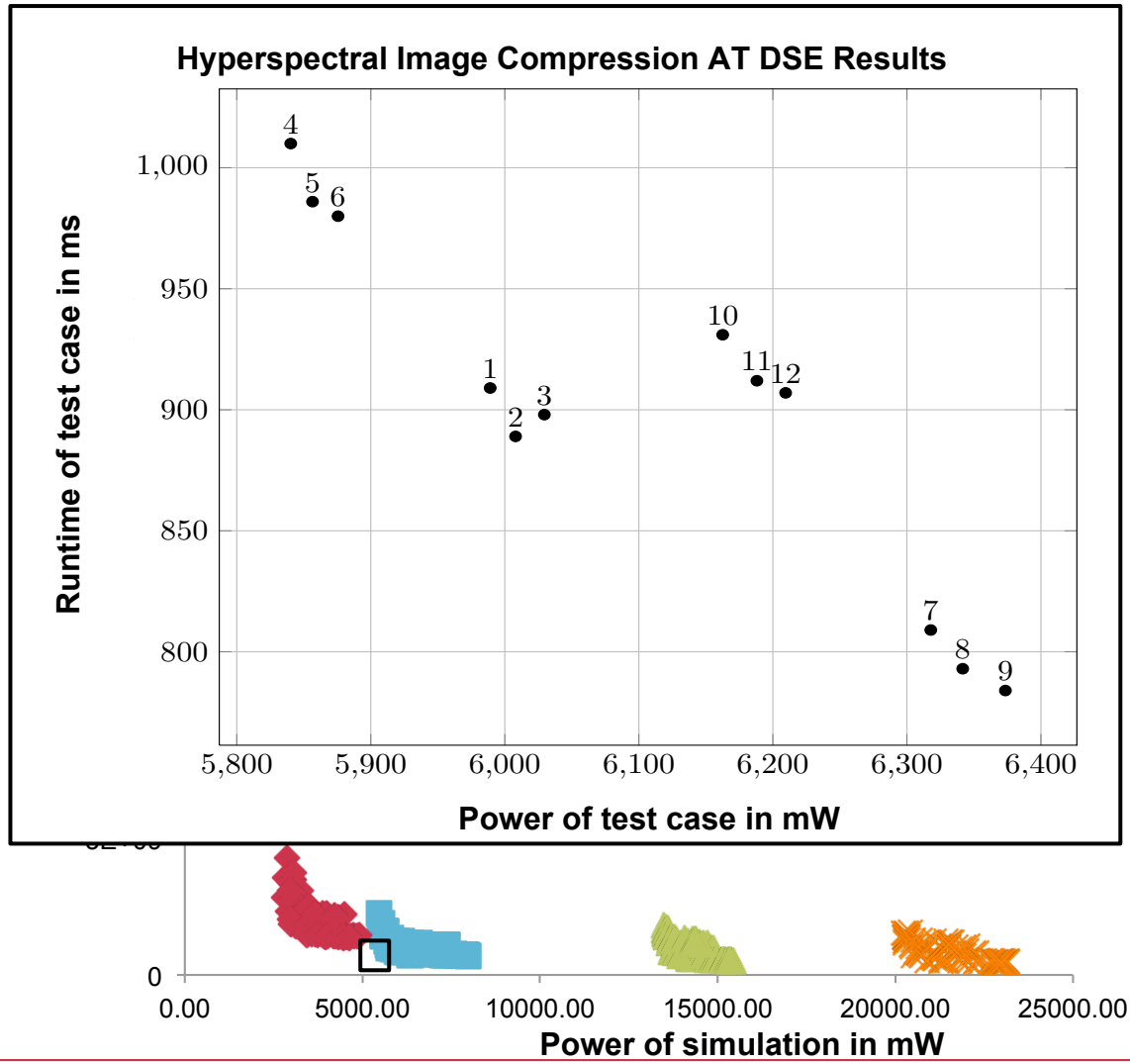
Energy vs. Performance Trade-off

LT High level analysis

- Configurations: 1280
- Total simulation duration: 16h
- Simulations in parallel: 16
- Av. simulation duration: 45s

AT Detail analysis

- Configurations: 12
- Total simulation duration: 4h
- Simulations in parallel: 12
- Av. simulation duration: 20m



Summary

A methodology to

- ... develop fast and low cost HW simulation models on different abstraction levels.
- ... construct, simulate and analyze Virtual Platforms.

Especially

- Development of a Library of base classes to encapsulate the core functionality of HW simulation models.
- Development of SystemC/TL-simulation models for core components of payload processors in the space domain.
- Laying the foundation for a flexible, extendable Virtual Platform to simulate and analyze multi core systems.

> ***SoCRocket***
>
> *"delivers a simulation environment which is as well as accurate as a RTL synthesis by much higher abstraction, more flexibility, deeper introspection and higher simulation performance."*
>
> ***Thomas Schuster***

Outline

Overview

- ✓ Why do we need Virtual Platforms?
- ✓ Where do Virtual Platforms fit in?
- ✓ How do we build Virtual Platforms?

SoCRocket: a framework solution

- ✓ The Motivation
- ✓ Tools & Techniques
- ✓ Modeling of SoCRocket IP
- ✓ High-Level DSE demonstration
- • **Current activities**

Demo

Current activities

Projects and collaborations with SoCRocket

- CCSDS File Delivery Protocol IP (ESA ICT) / C3E
- Next Generation Multi-Processor (ESA ICT) / Terma GmbH
- Embedded Multi-Core Systems for Mixed Criticality Applications in Dynamics and Changeable Environments (EMC2)
- Reconfigurable ROS-based Resilient Reasoning Co-Operating Platform
- Sandwich-PhD with ESA in parallel/NoC simulation
- Cooperation with University Bremen to test new draft standards of TL verification within the SoCRocket-Framework
- Cooperation with University of Dortmund – ARM Integration
- Thales – SMT Linux Verification (planned)

Current activities

Further development within SoCRocket

- Better logging capabilities
  - ViPES 2015
- Introduction of a Universal Scripting Interface for SystemC
  - DVCon Europe 2015
- Simplifying register modeling by replacing GreenReg with scireg
  (Cadence, License: BSD)
- Import for IP-Xact platform designs at runtime
- Export Platform configuration directly into GRLIB designs
- GRMon-Debuging Support

Current activities

Teaching curriculum with SoCRocket

VLSI II Lecture

- SoCRocket is used to explain ESL-D

VLSI II Laboratory

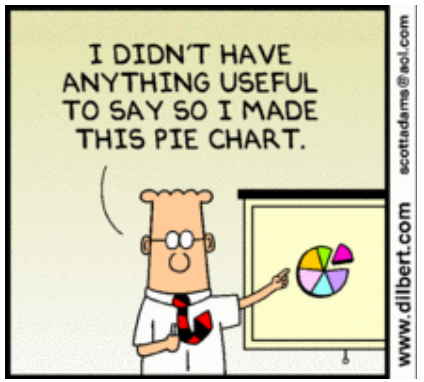
- Developing TL-models to perform a JPEG compression

CuSE II Lecture

- The foundation components are used to explain TLM in depth

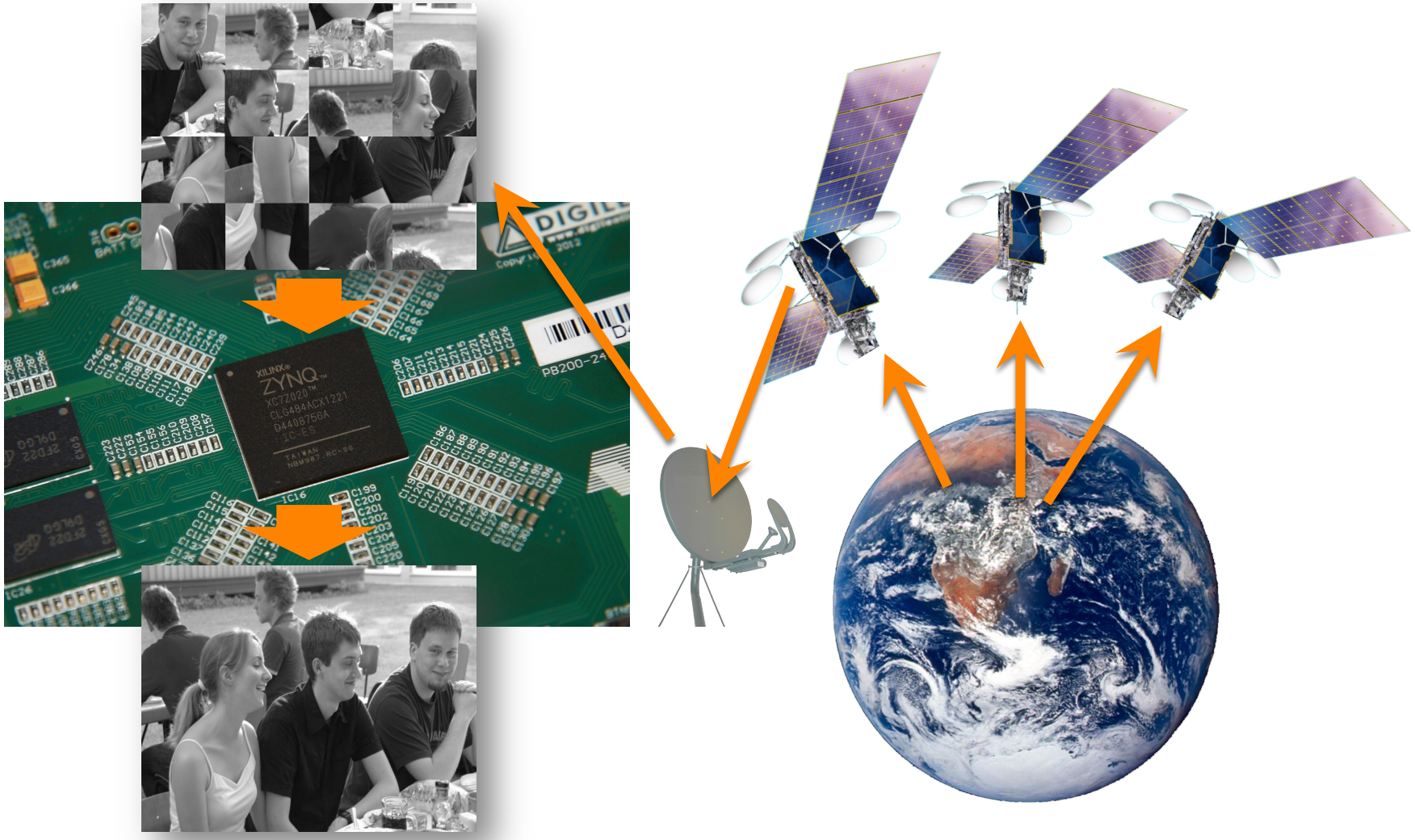
CuSE II Laboratory

- Students need to fix a RTL prototype



CuSE II Laboratory

(Chip and System Development II)

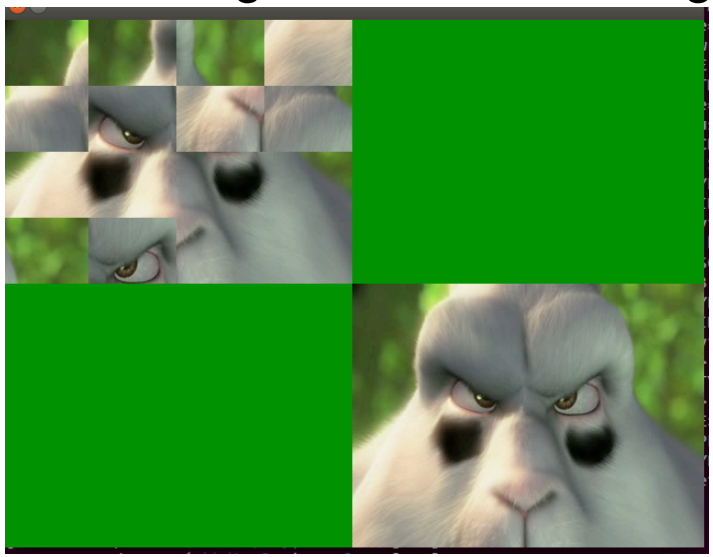


Demonstration VP

SoCRocket demo

What we are doing with our students in the lab:

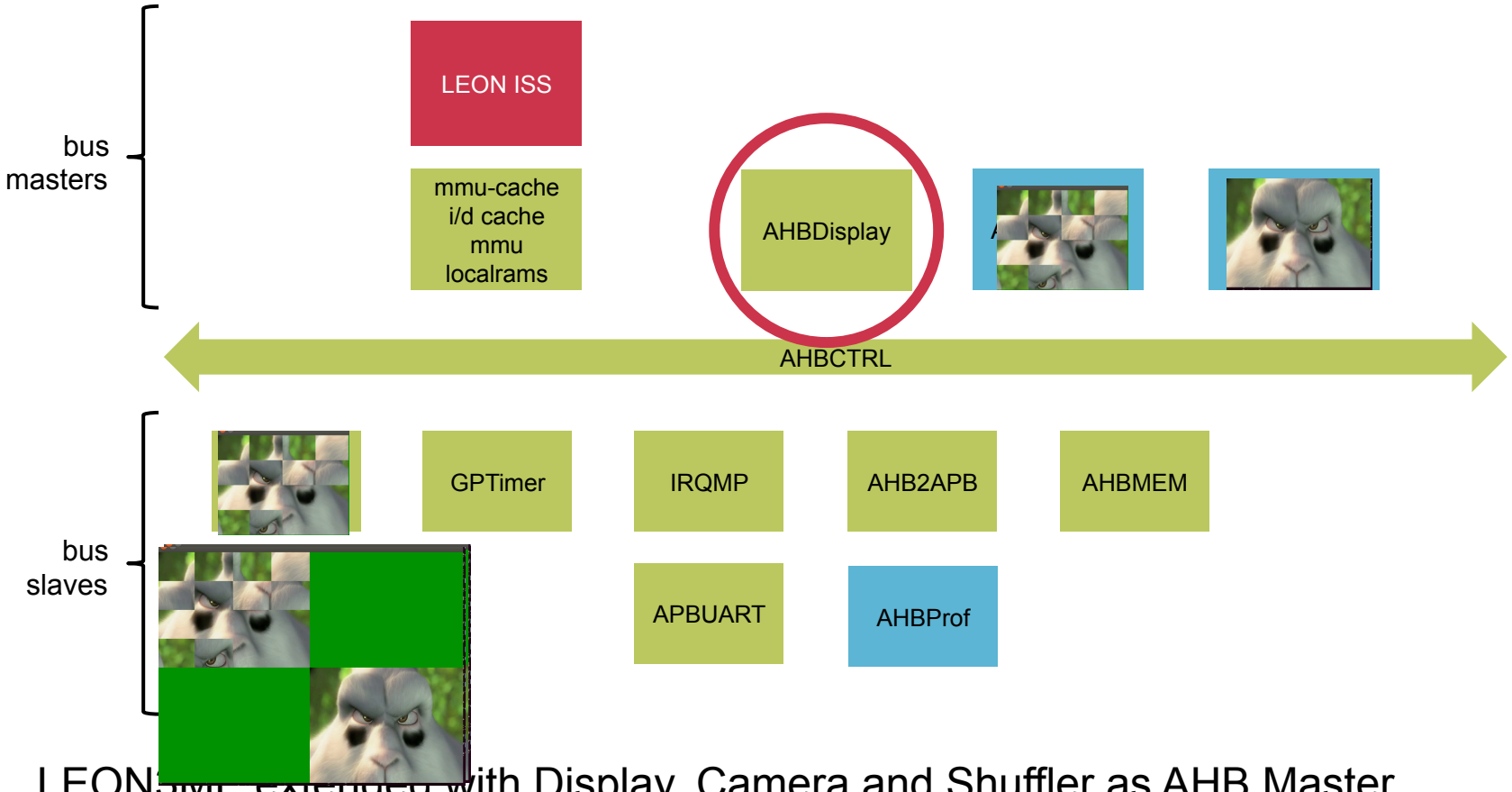
- Reordering of scrambled images received from Satellite



- Solution using software shuffling (on LEON ISS)
- Development of “shuffle” hardware accellerator
- Working with SoCRocket and RTEMS

Demonstration VP

Platform Design



LEON3MP extended with Display, Camera and Shuffler as AHB Master.

- Models provided by ESA (integration only)
- Models with GRLIB reference

Demonstration VP

Delay (LT, DMI enabled)

Current time
47:06

Camera Delay:
15ms

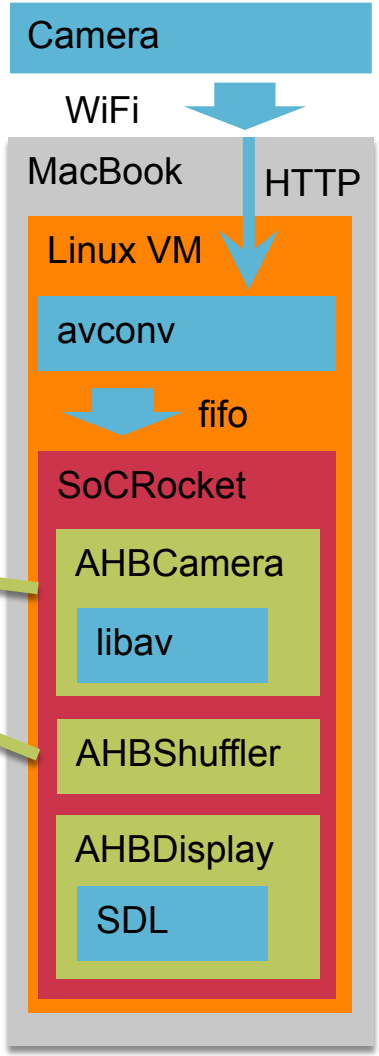
Camera time
46:91

Stream Delay:
14ms

Stream time
46:77

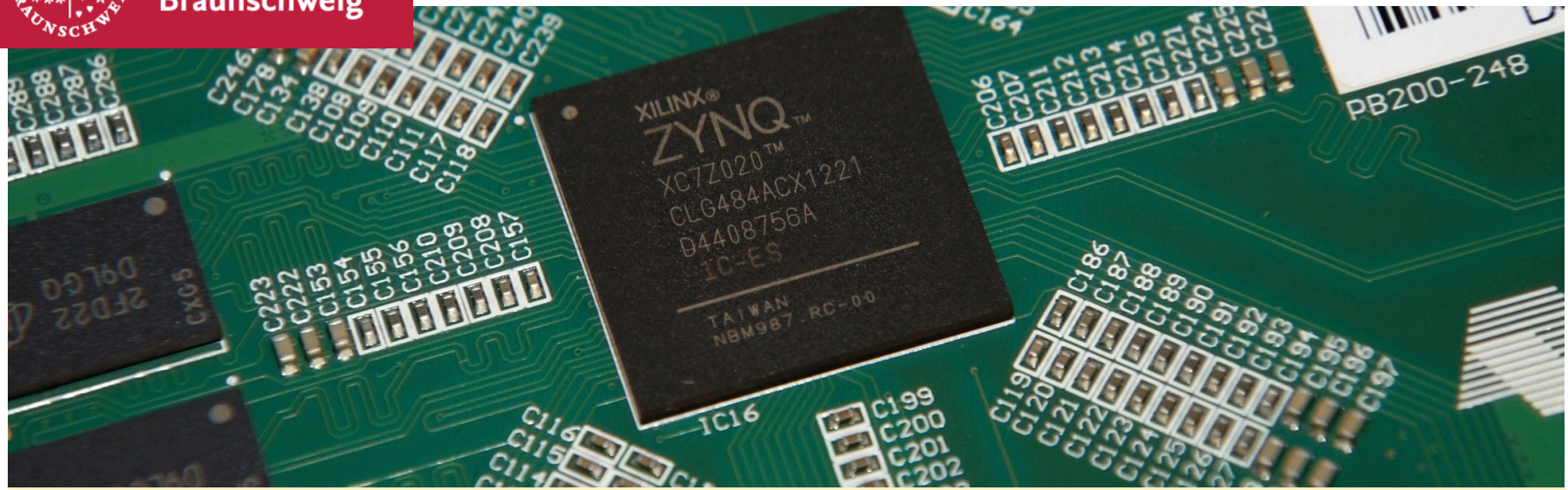
SystemC Delay:
30ms

Platform time
46:47



Technische Universität Braunschweig

Chair for Chip Design for Embedded Computing



SoCRocket is available online:
https://socrocket.github.io/

SoC Rocket

For more information please contact us!

Foundation Tools

GreenSoCs TLM infrastructure

Explored commercial and open-source Virtual Platform solutions.

Selected appropriate infrastructure components:

- **GreenReg**
  - Framework for SystemC register modeling
  - Register bank container which can be bound to TLM sockets
  - Bit-field callback functions etc.

- **AMBA Sockets**
  - TLM convenience sockets with build-in memory management
  - Predefined payload extensions for modeling AMBA bus transfers

- **GreenControl (precursor to Accellera CCI)**
  - Parameter API for model configuration
  - Debugging, Tracing, runtime-reconfiguration

Performance Counting with GreenControl

Analysis Capabilities

Performance Counters

- Monitor performance and behavior of simulation models
- Organized in global namespace:

  e.g. Number of transactions routed by AHBCTRL

  top.ahbctrl.performance_counters.total_transactions
- All counters can be accesses at runtime

Analysis API (based on GreenControl/AV)

- Read/write counters
- List counters
- Trace changes in logfiles
- Write wave form files (VCD)
- Register callback functions

Source: Shutterstock (#158906465)

VPs as tool to develop embedded software

The „Programer's View“ of HW only covers the aspects which are needed to debug target software. Everything else is astracted for speed reasons.

- HW Programmer's View
  - Processor and periphery register „viewing“ und trace,
  - Memory view,
  - Interrupt/reset Trace,
  - “Waveform” type view of interrupts, resets, HW register access,
  - Periphery reachable by a processor.
- Processor/Core Sensitive GUI Views
- Easy launch & debugging of target executable in a VP
- Full debugging support of multi-core und Multi-processor applications
- Non-intrusive target software trace, analysis & profiling
  - Line, Function coverage, Memory Allocation Trace … .

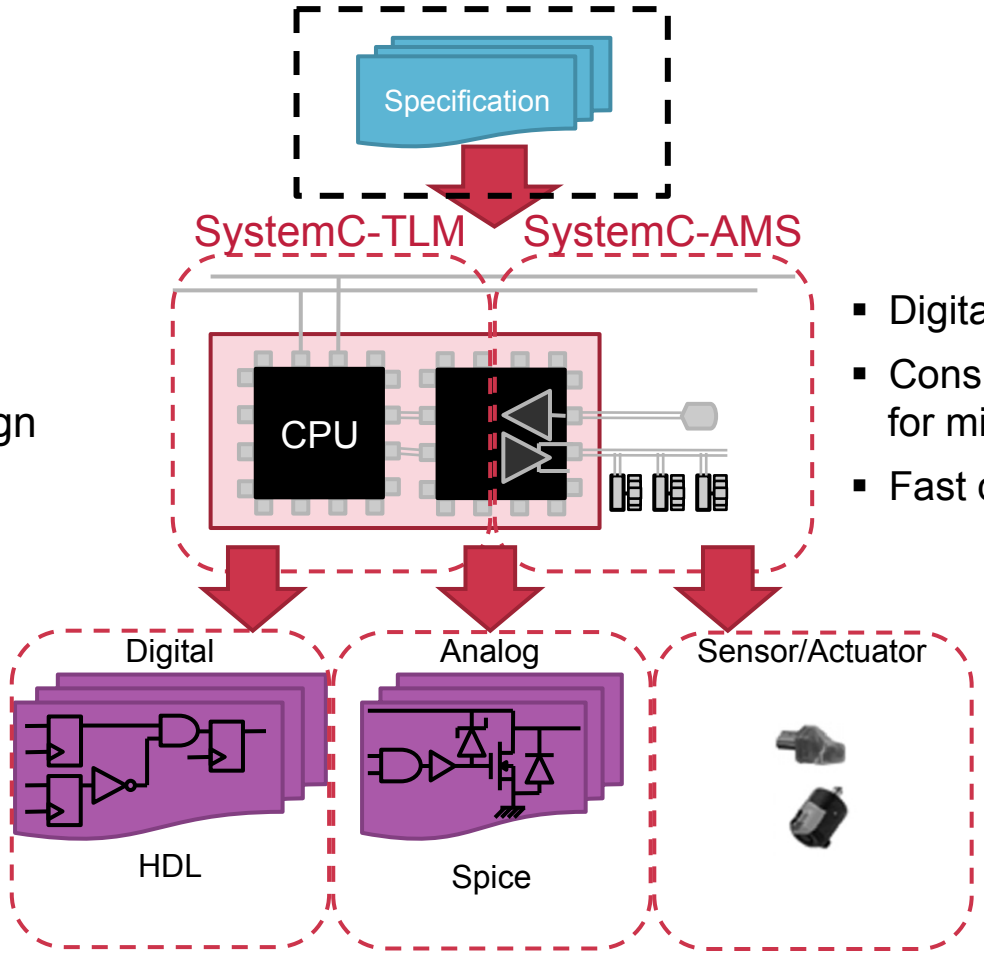
Source: Cadence

Virtual Platform with SystemC/TLM

System-Design
Functional Design

Architecture Design

Implementation



- Digital and analogous simulation,
- Consistent system environment for mixed signal
- Fast digital simulation.

Source: Denso

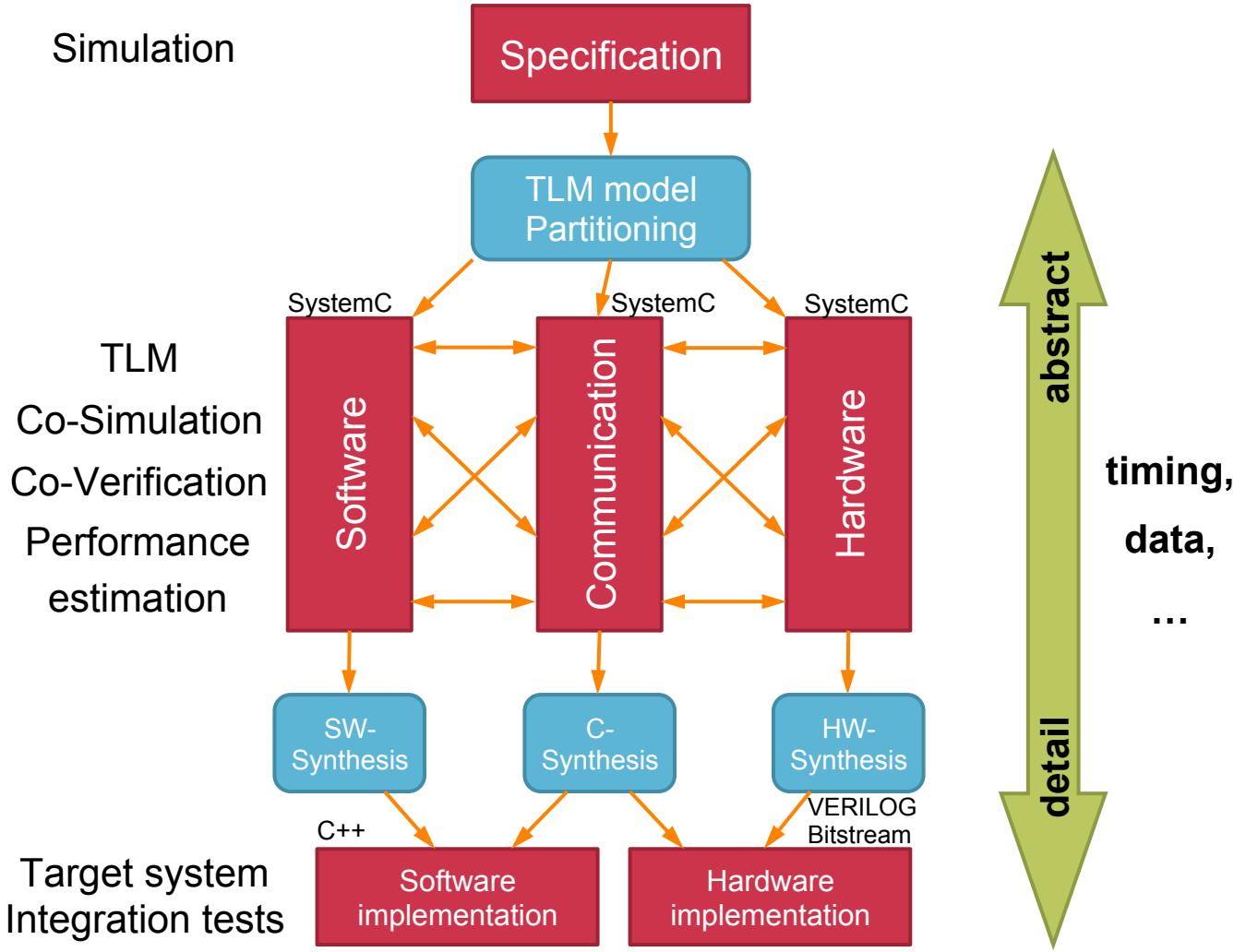
System conception with SystemC and TLM

Specification

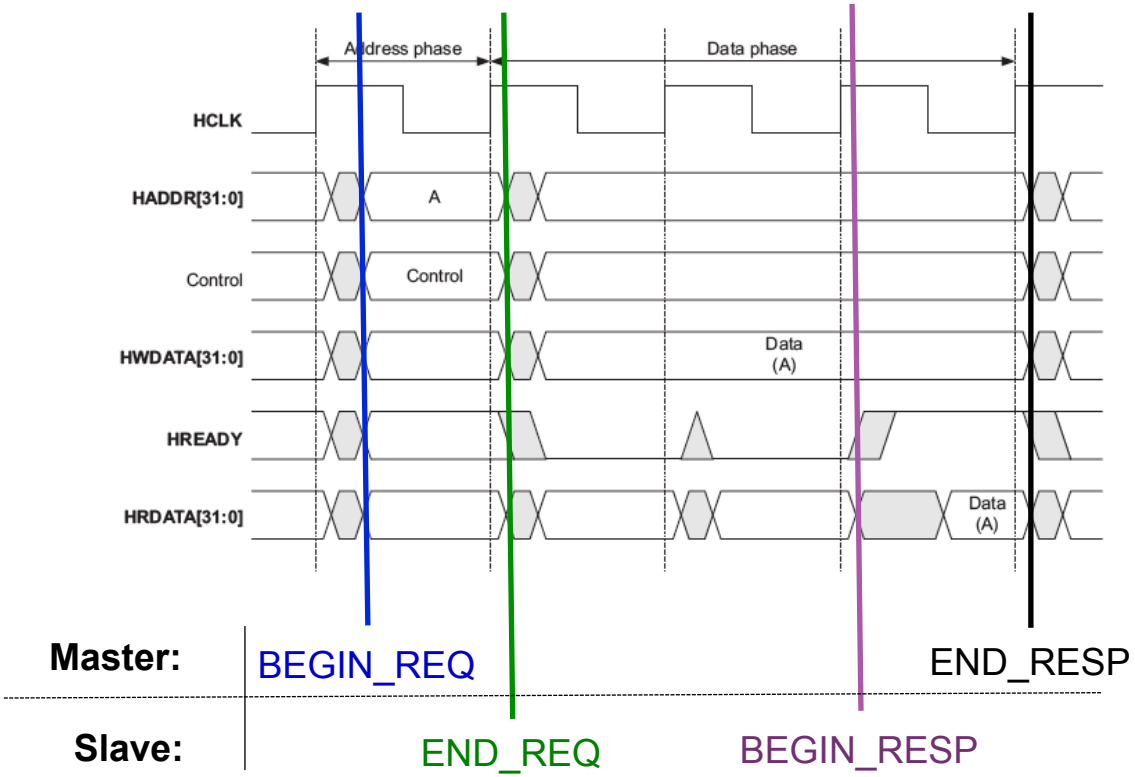
SW modeling

Target code generation

(Virtual) platform generation



High-Level Modeling of SystemC IPs
AHB/APB Bus protocol modeling (AT)



Example read burst:

BEGIN_REQ
Time master sends the first address

END_REQ
Time slave samples the last address

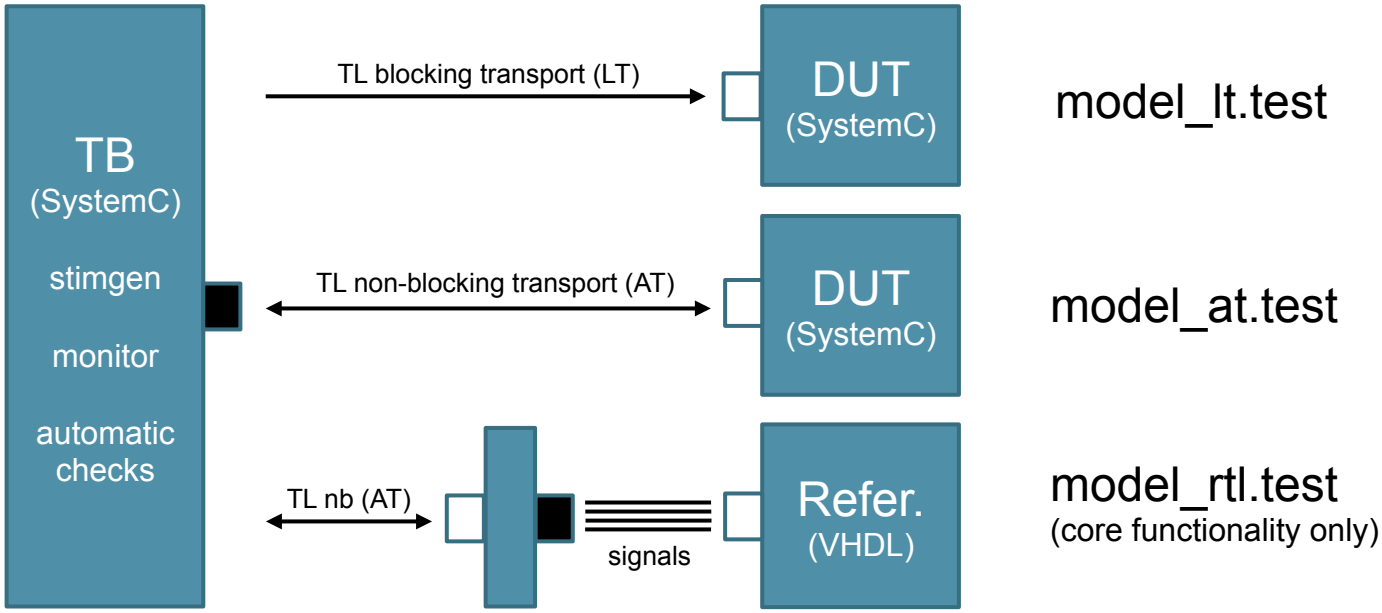
BEGIN_RESP
Time slave sends first data

END_RESP
Time master samples the last data

AHB protocol approximated using 4-phase non-blocking TL communication.

Verification of TL models

- Directed functional testing
- RTL Co-Simulation



Test coverage: ~90% Timing accuracy (AT)*: ~90%

*with respect to available RTL tests

VP Development and Validation

Infrastructure

Developed in the course of this project:

Tailored build system (WAF):

- Host system configuration (env set and checks)
- Centralized control for all build processes and tools
- Automatic unit tests
- Modelsim/QuestaSim co-sim integration

SignalKit:

Modeling kit for fast TL signal communication.

Used for modeling interrupt lines, reset and snooping

Verbosity control:

Debug output configuration and filtering