Virtual Prototyping

45th System Architecture Study Group

Jos Verhaegh

June 5th 2012

Contents

NXP Semiconductors

Virtual platforms

Modeling Methodology & Libraries

Application of Virtual Platforms in NXP

Conclusions

Contents

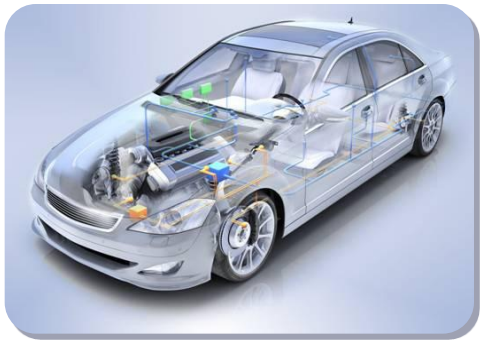
NXP Semiconductors

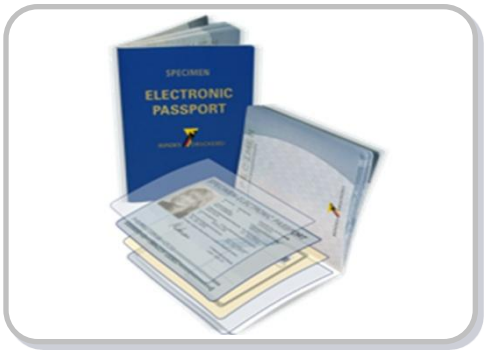
- **President & CEO**: Rick Clemmer
- **Headquarters**: Eindhoven, The Netherlands
- Established in 2006 (formerly a division of Philips)
- 50+ years of experience in semiconductors
- **Focus:** High Performance Mixed Signal products

- **Businesses:**
  - Automotive
  - High Performance Mixed Signal
  - Identification
  - Standard Products

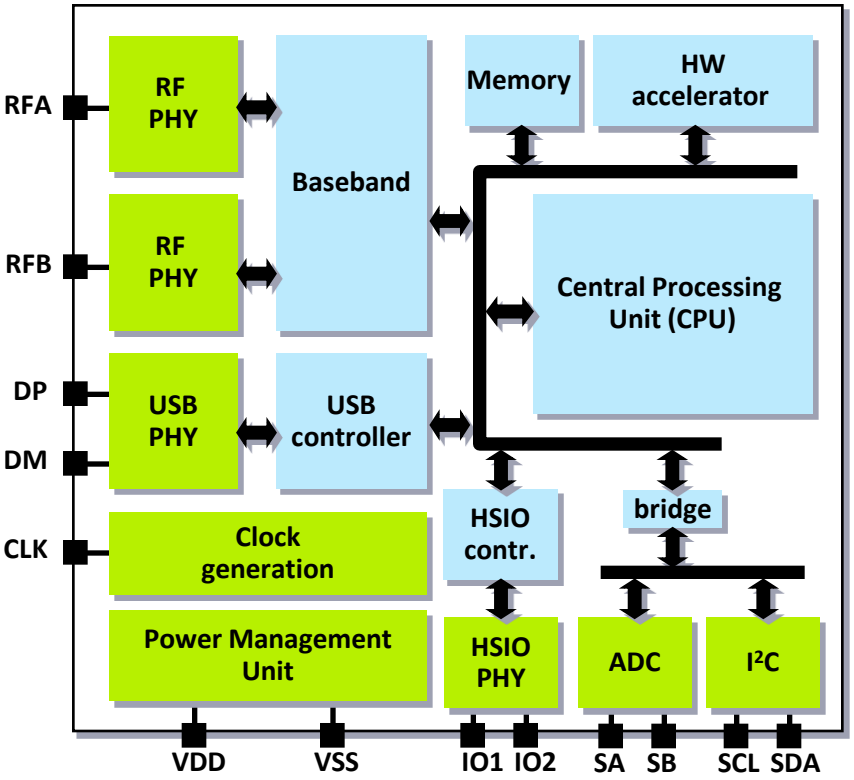
- Owner of NXP Software, a fully independent software solutions company

System Design Challenges – Examples





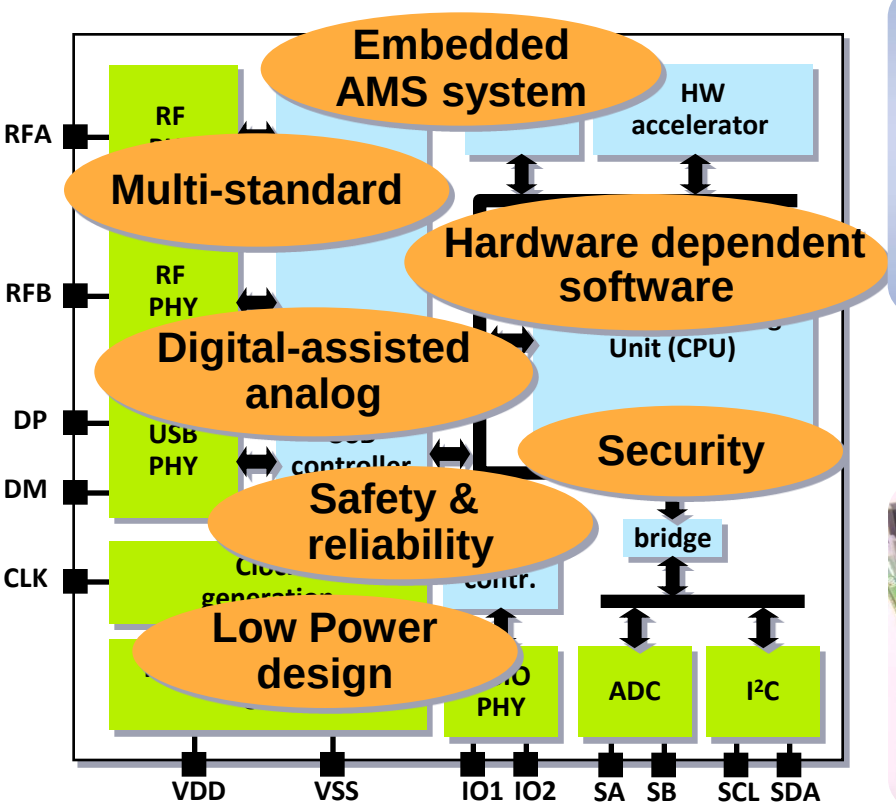
Identification products

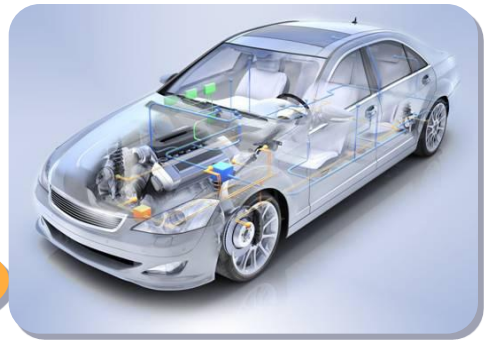


AMS IP Digital IP

Automotive products

System Design Challenges – Examples





Identification products

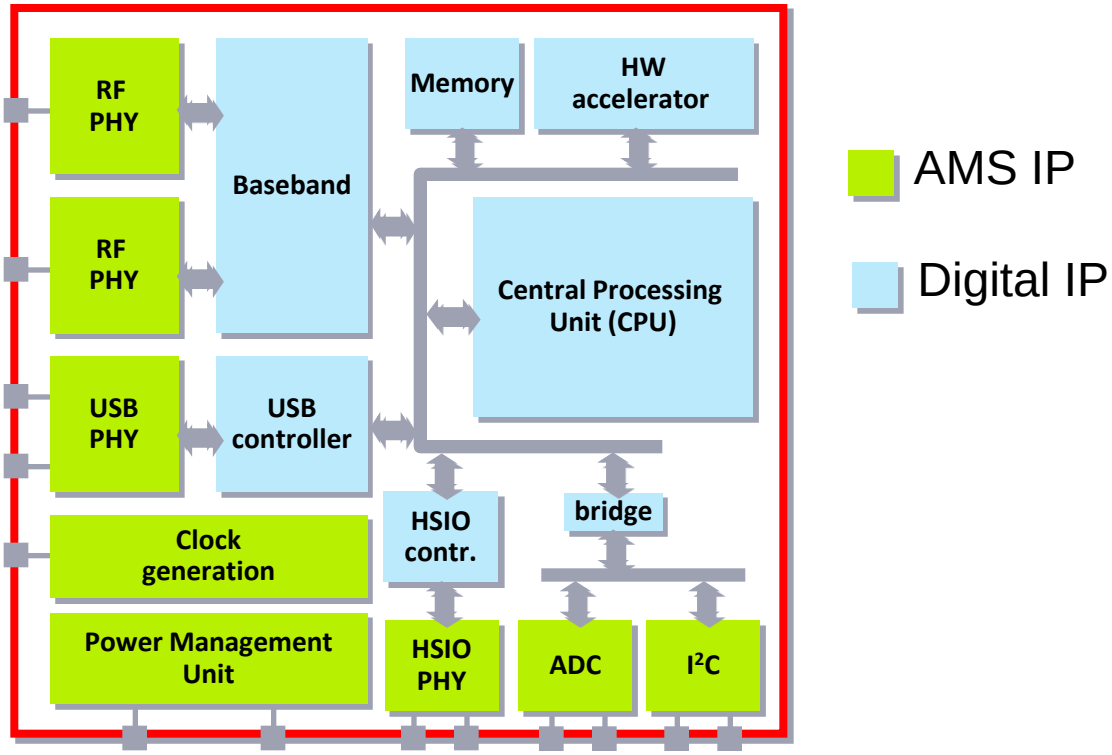
Automotive products

AMS IP Digital IP

Contents

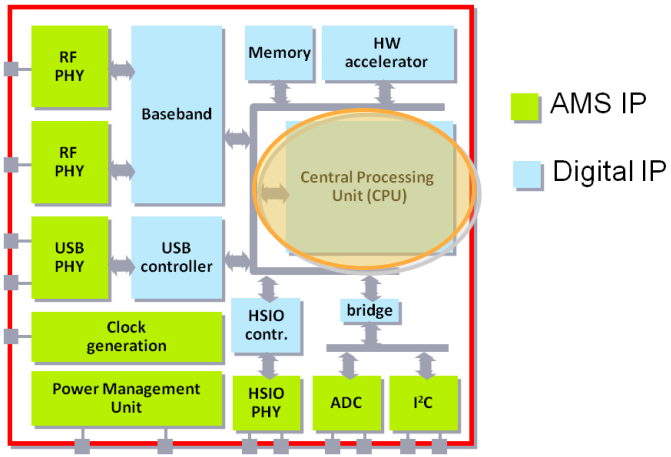
Virtual platforms

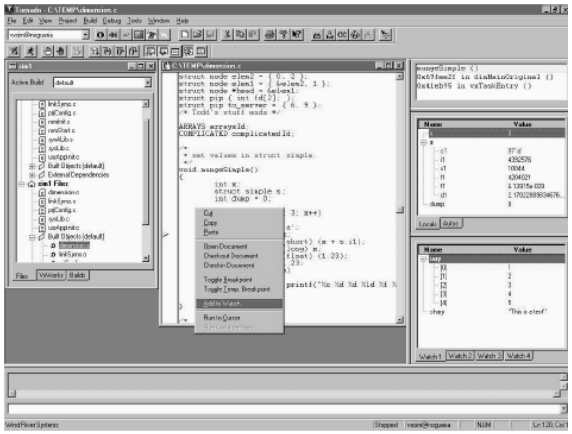
Simulation model of real hardware at high abstraction level



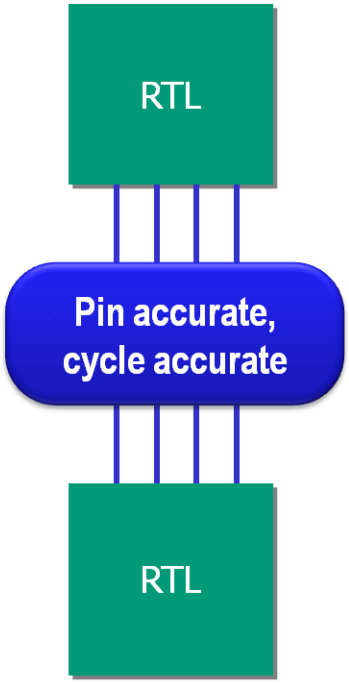
Instruction Set Simulator (ISS)

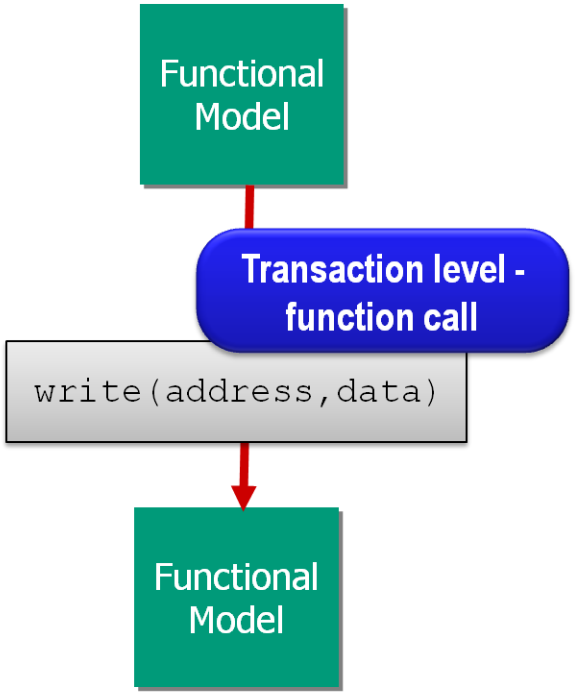
- Simulation model of embedded processor
  - Typically provided by core vendor (ARM, Tensilica,…)
- Executable software image identical to final image running on silicon (Binary compatible)
- Allows to connect to SW debugger (ARM Realview, Keil uVision, Lauterbach,…)

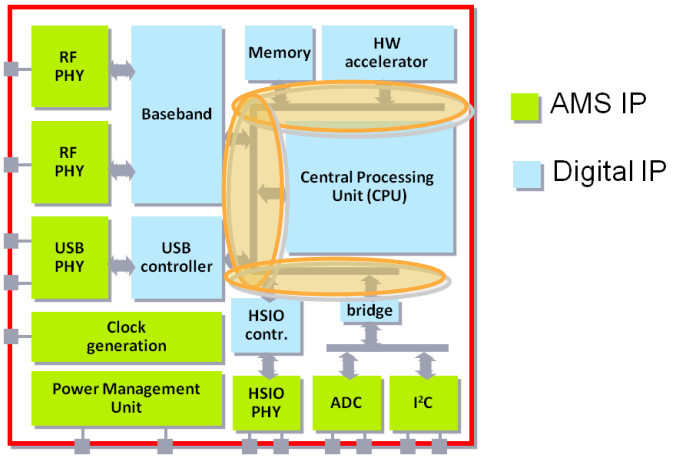




Transaction Level Modelling





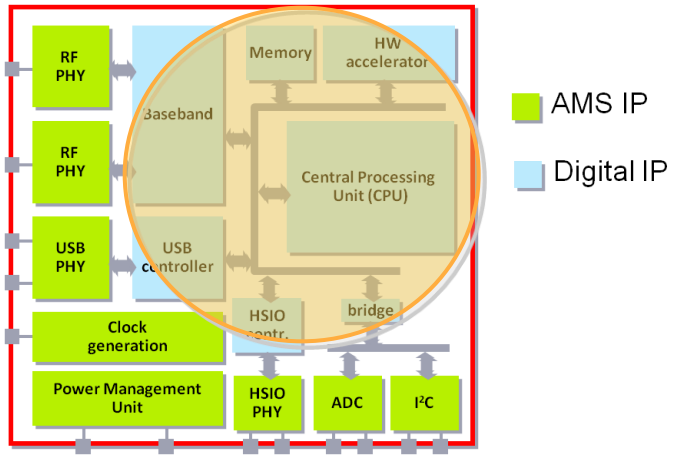


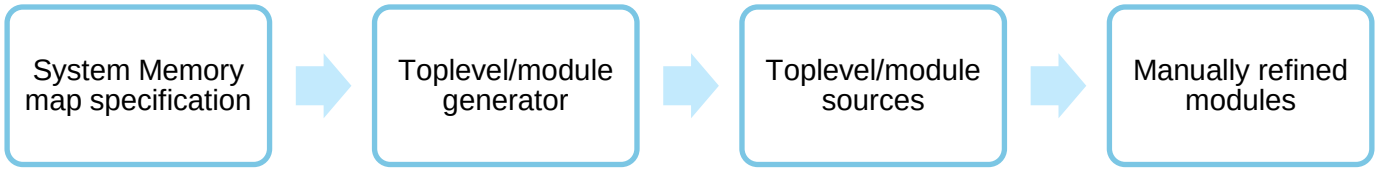
Simulate every event

100-10,000 X faster simulation

Reference platform generation

- Initial SystemC platform automatically generated from system register map (consistent with RTL)
- Generated SystemC reference platform contains ISS
- The Reference platform is ready for basic SW development (register read/write behavior)
- Behavior can be added to SystemC IP model templates

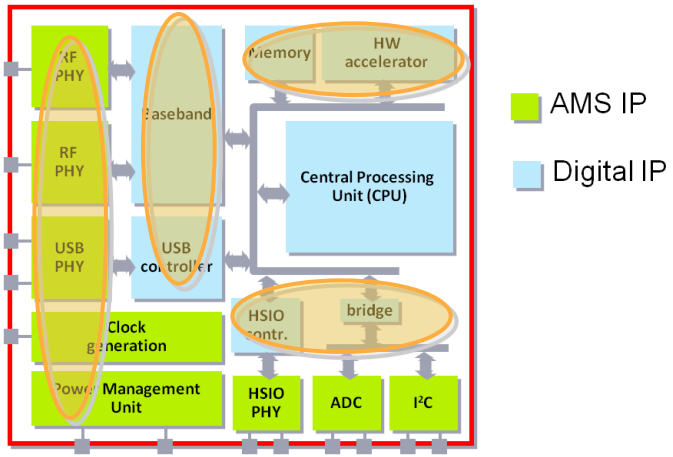




SystemC standards & libraries

IEEE 1666-2005 SystemC

- a set of C++ classes and macros which enables hardware description constructs in C++
- provide an event-driven simulation kernel in C++

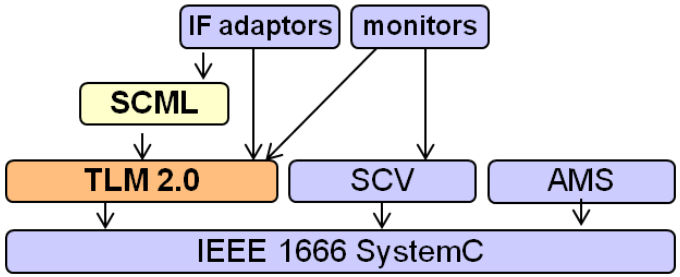


OSCI TLM 2.0

- Transaction level modeling of communication between digital modules
- Function calls abstract away pin level signals

SystemC Modeling Library (SCML)

- Reduce the modeling effort and learning curve
- Enable model reuse and refinement



OSCI SystemC-AMS 1.0

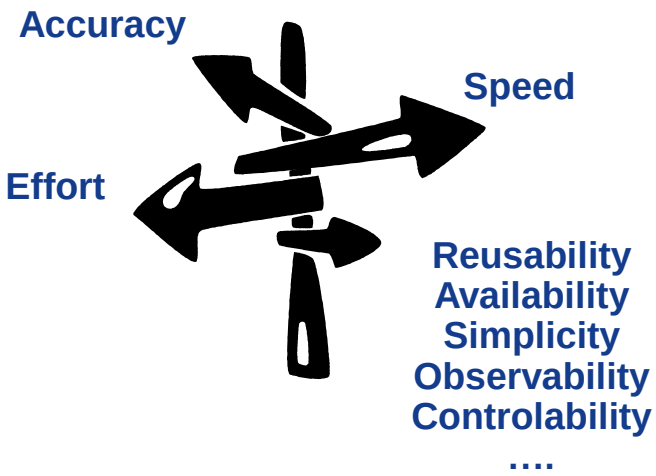
- To abstract analog behavior and communication

Contents

Modeling paradigm

Multidimensional problem

- How to model for maximum speed?
- How to model for full accuracy?
- How to minimize the modeling effort?
- How to make models reusable?
- How to deal with legacy models?
- … and how to achieve all this in a simple way?



Different kind of models with different properties:

- Processor models
- Interconnect models
- Memory subsystem models
- Peripherals

- **Need for clear modeling strategy based on standards**

Minimizing the modeling effort
(…and making models available earlier)

Simplify modeling

- Methodology and guidelines
- Separation of concerns
- Predefined building blocks (Modeling libraries)

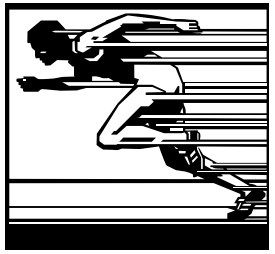
Standards based modeling

- **TLM2.0 compliancy** for Interoperability between models (released std)
- **CCI compliancy** for standardized configuration, control & inspection of models (under dev)
- To enable building of re-usable models (model portfolio)
- To enable easy integration of 3rd party models

Automatic generation

- Translate RTL into CA SystemC (SL) by using Carbon tooling
  - Suitable for verification & performance analysis, but too slow for SW development use case
- Generate modeling template (user only fills specific functionality)
- Generation/wrapping from functional tools/model
  - CoWare SPD, Matlab Simulink, legacy C++

Modeling for speed



Minimize the number of events/context switches

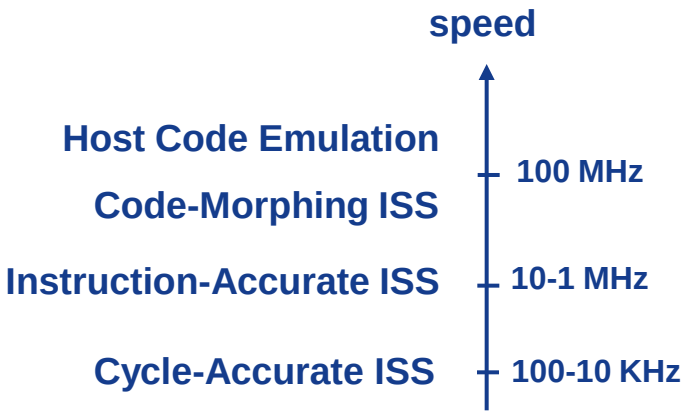
- Coarse grain communication
  - Model cycle-by-cycle communication only when strictly required
- Coarse grain computation
  - Group timed operations, call *wait(…)* only when synchronization is required
- Careful usage (strategy) of *sc_threads*, *sc_methods*, *sc_events*, etc
- Avoid (at all) using *sc_clocks*

Efficient code

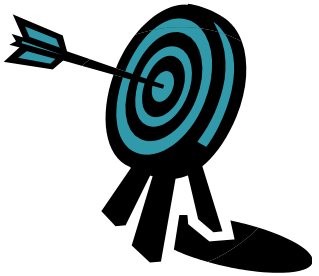
- Profiling tools (gprof, vtune)

Efficient SystemC kernel

- Parallel/distributed implementation



Modeling accuracy



Functional accuracy is not an issue

- Test strategy is mandatory (qualification)

Loosely timing can be modeled easily

- Time can be estimated or extracted from existing models
- For many use cases is enough!

Approximate and Cycle Accurate timing is very hard to achieve

- Manual modeling (it can be as hard as RTL)
- Automated techniques (extraction from RTL to TLM)

Define accuracy windows, window size depends on the use-case

- Verification requires 100% accuracy, i.e. window size is 1 cycle
- Window size for bus exploration should be 1-10 cycles, accuracy >90%
- Window size for software performance measurement can be >100k cycles, accuracy >90%

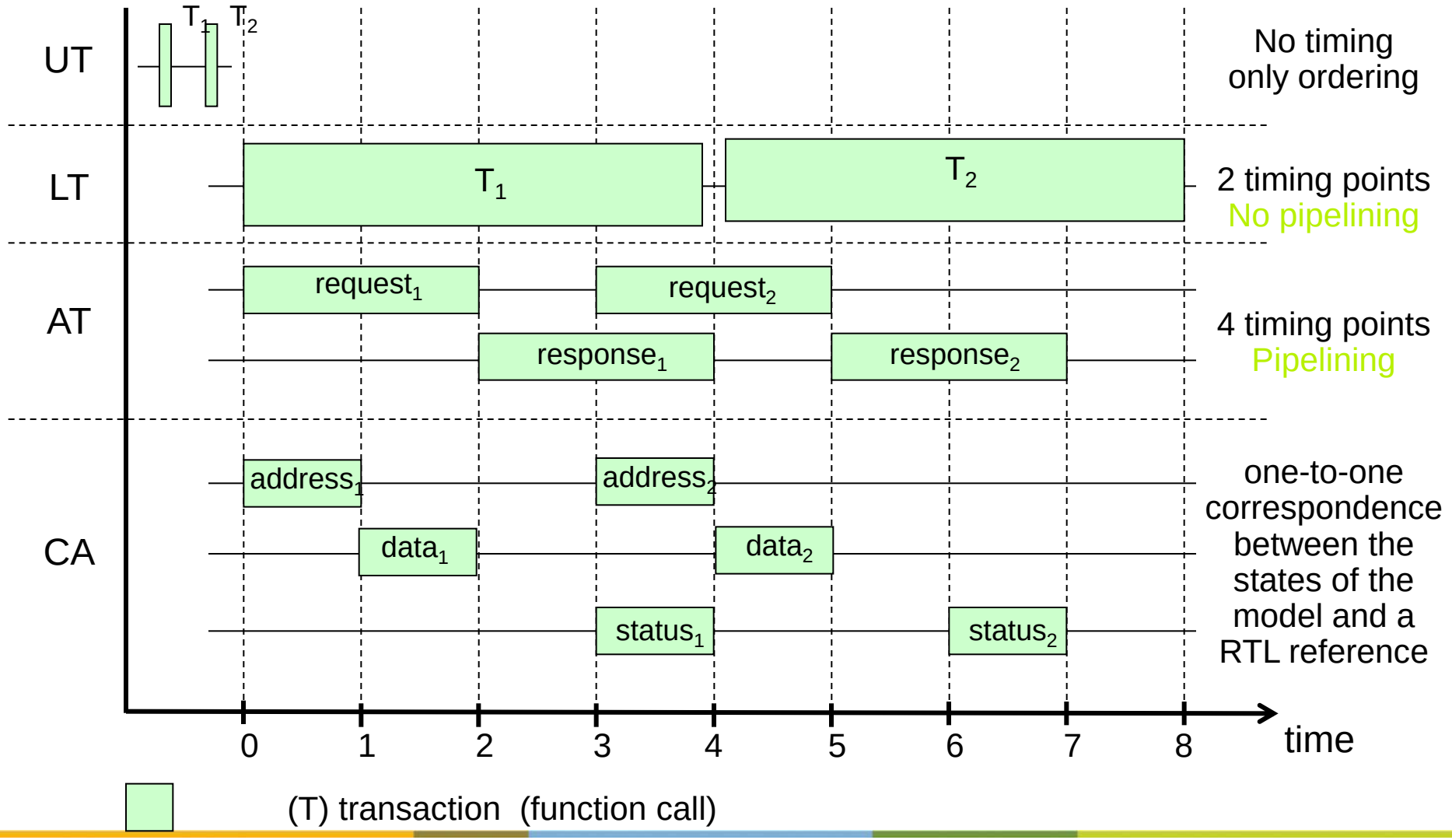
Coding Styles

- Loosely-timed
  - Only sufficient timing detail to boot O/S and run multi-core systems
  - Processes can run ahead of simulation time (temporal decoupling)
  - Each transaction has 2 timing points: *begin* and *end*
  - Uses direct memory interface (DMI)

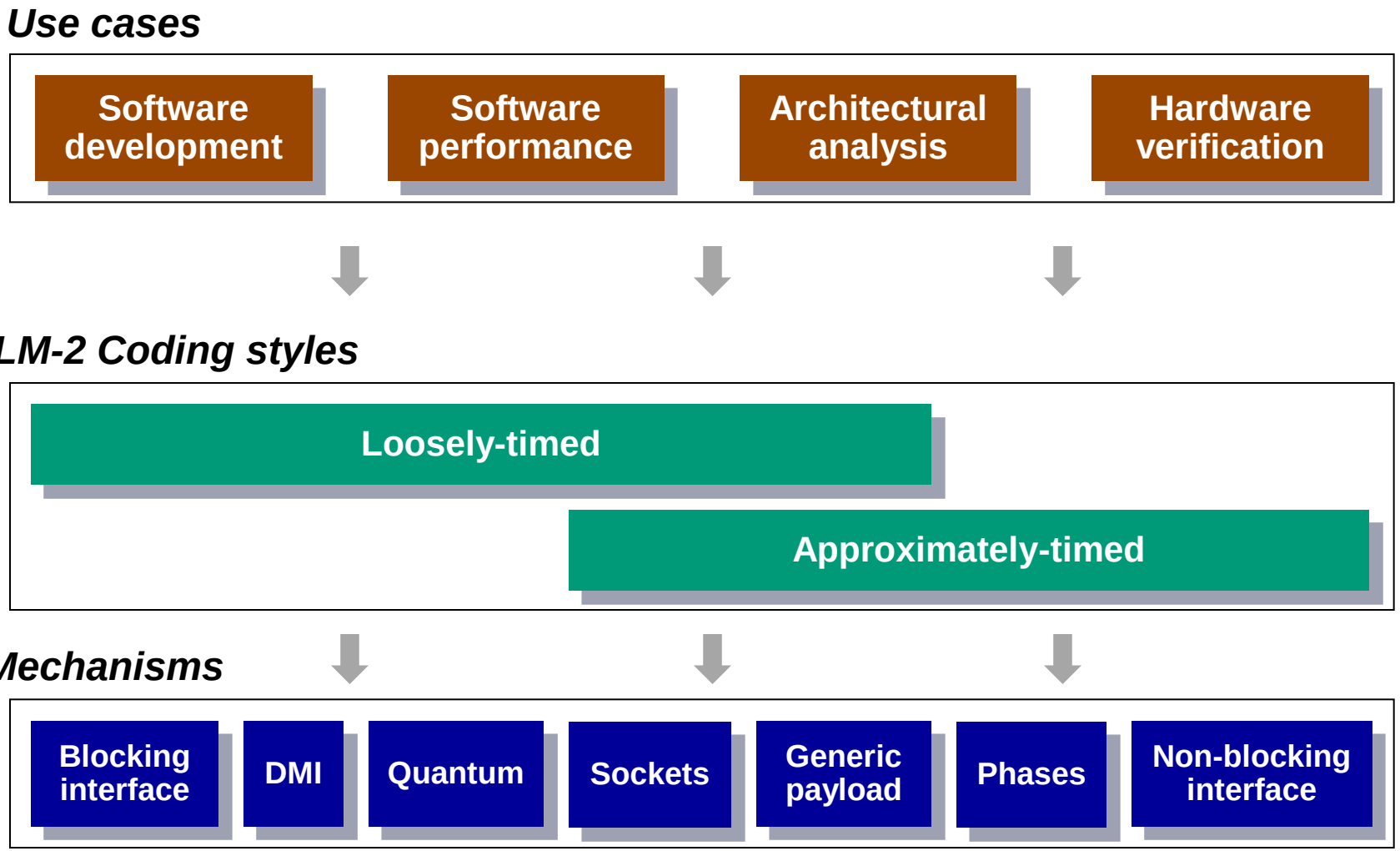
- Approximately-timed
  - *aka* cycle-approximate or cycle-count-accurate
  - Sufficient for architectural exploration
  - Processes run in lock-step with simulation time
  - Each transaction has 4 timing points (extensible)

- Guidelines only – not definitive

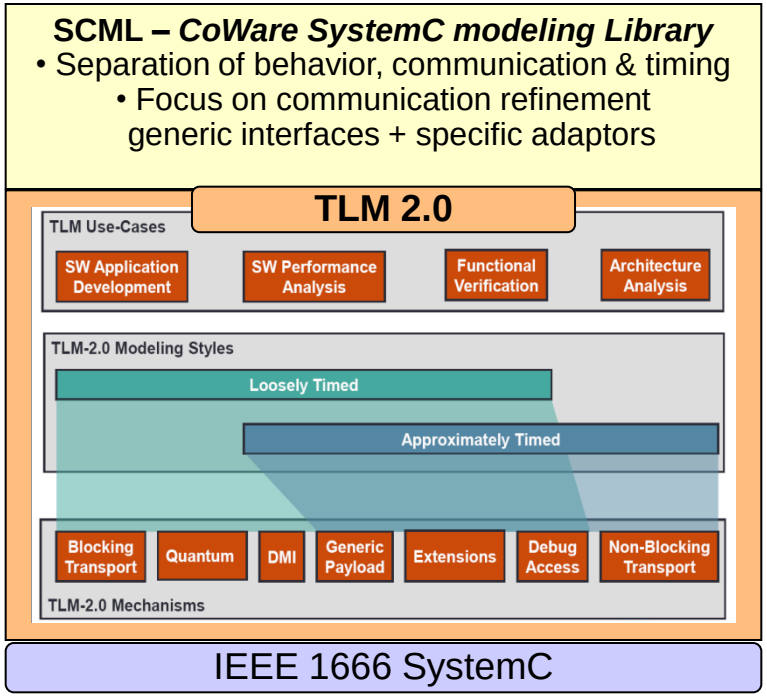
OSCI transaction level Modeling styles (II)

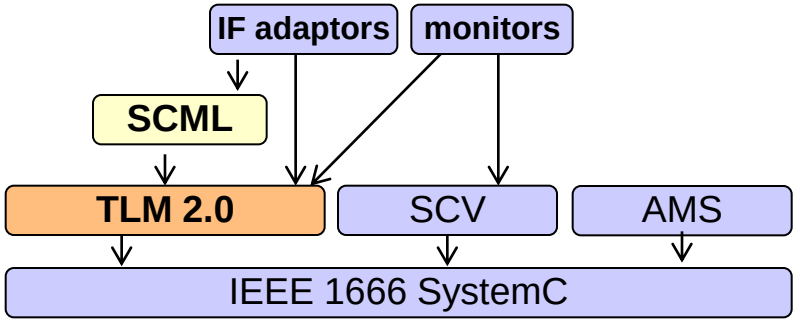


Use Cases, Coding Styles and Mechanisms



Layered modeling approach

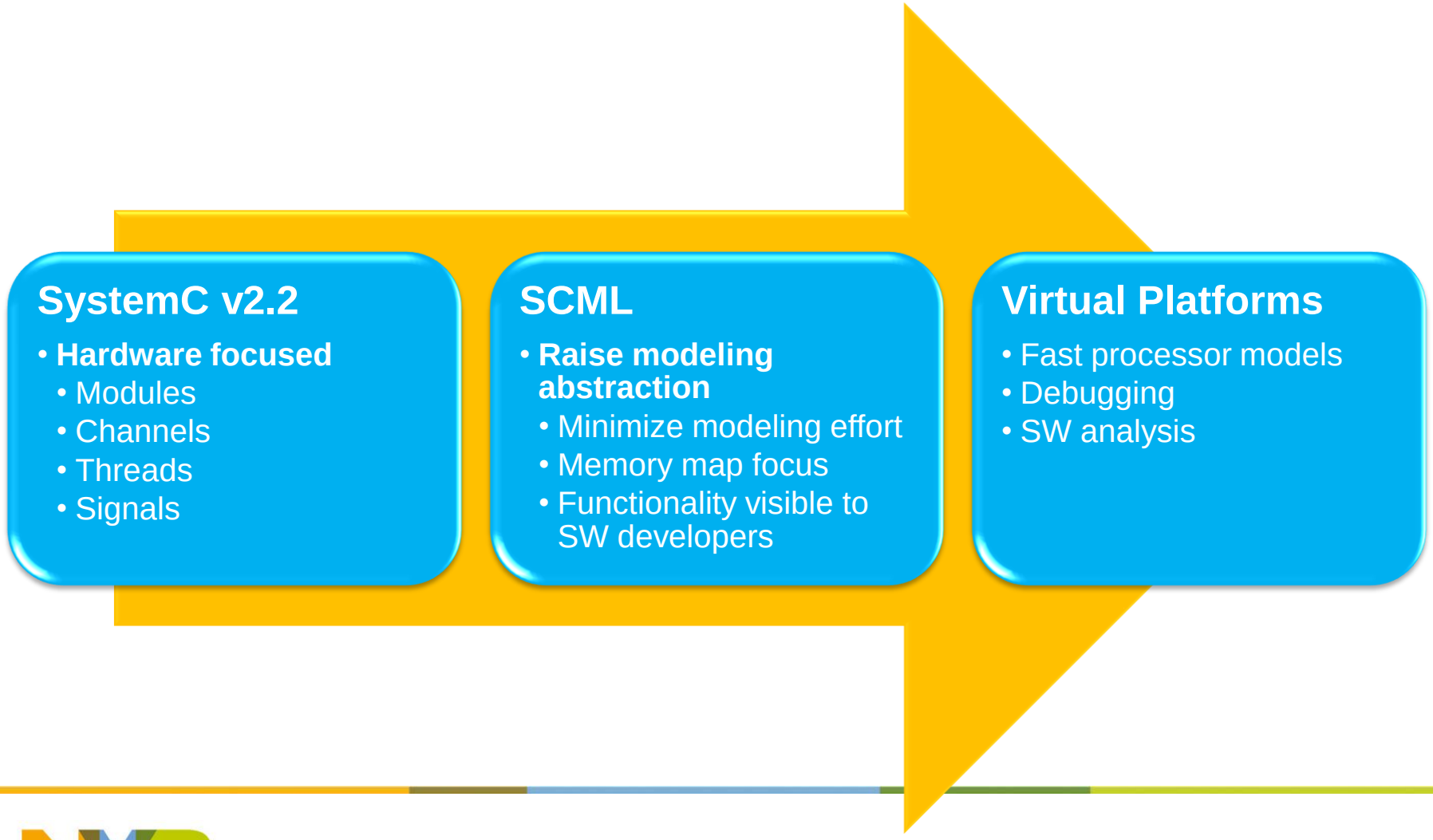




Objectives modeling libraries:

- Reduce the modeling effort and learning curve
- Enable model reuse and refinement
- Improve configurability of models
- Align with industry standards
  - *OSCI TLM 2.0* & CCI
- SystemC AMS extensions library available

SystemC Modeling Library (SCML)

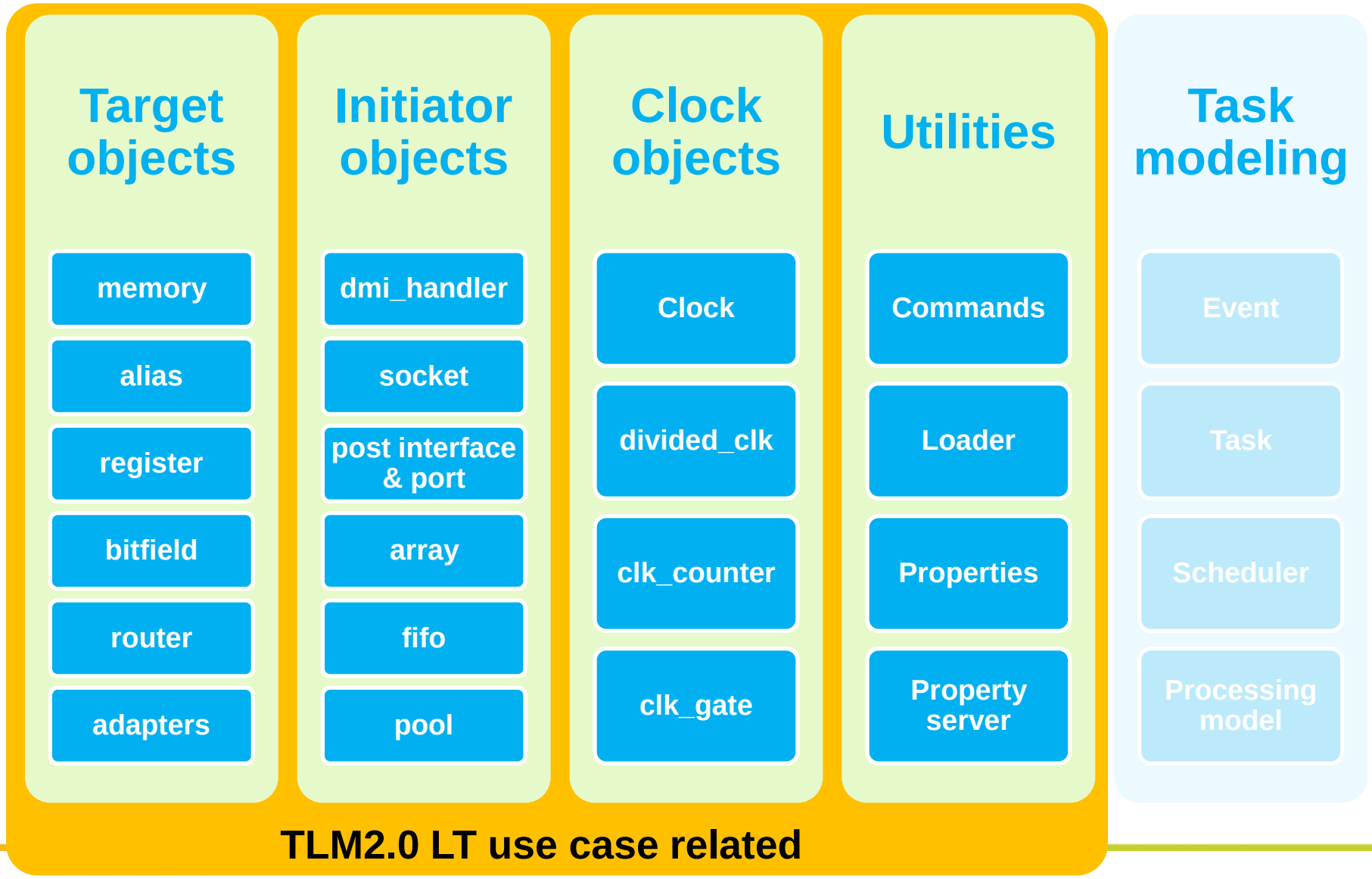


Why SCML?

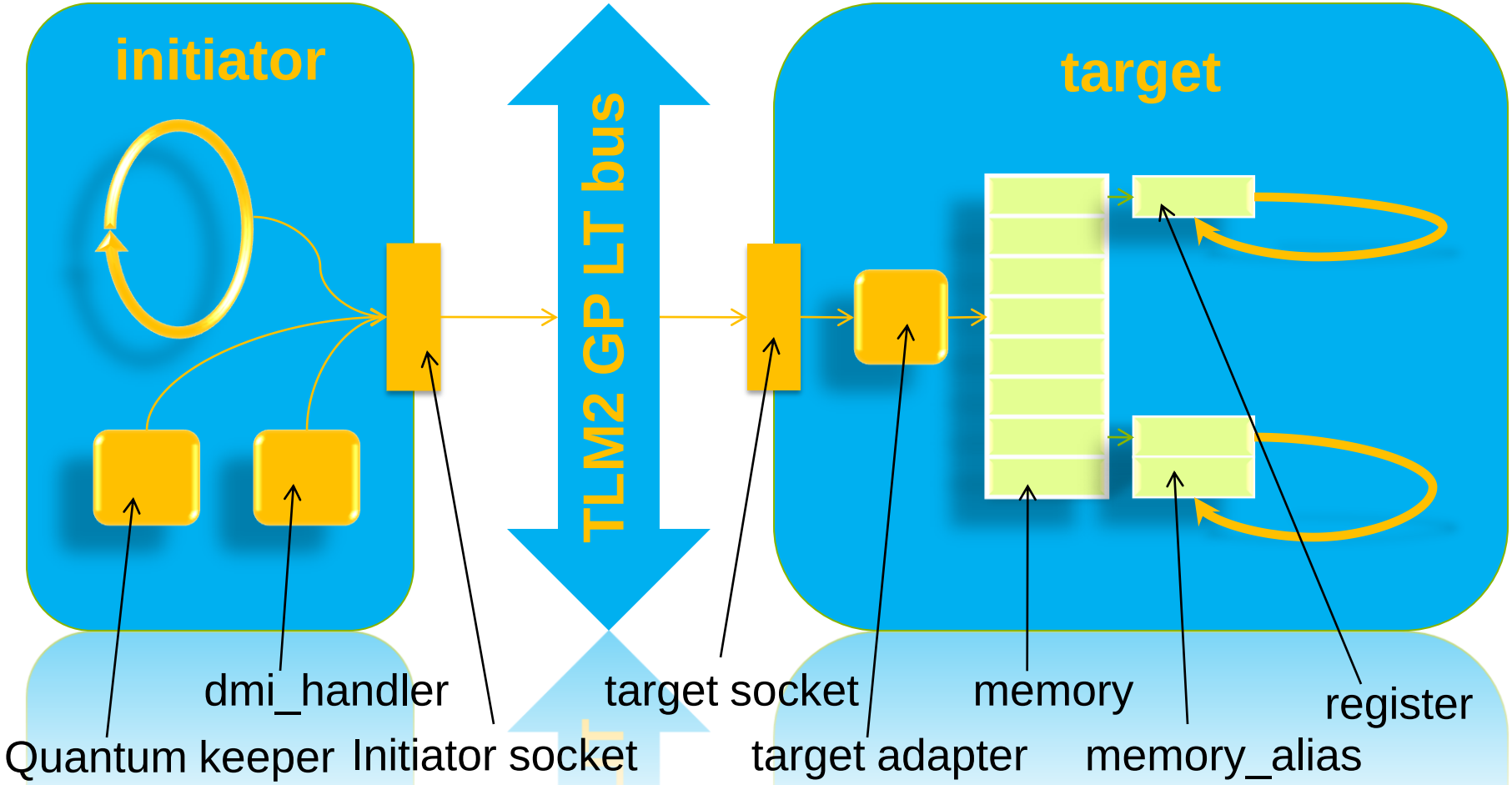
SCML goals

- **Simulation speed**
  - Enable ‘backdoor’ access (supported by TLM2.0 DMI)
- **Ease of modeling**
  - Abstraction through modeling objects
  - Configuration through properties
- **Debugging & analysis**
  - Visualization in VPA
  - Standardized modeling style
- **Interoperability**
  - Source code version of library available
- **Reuse**
  - Support multiple use cases
- **Maintainability**
  - Enforced modeling style to guarantee maintainability

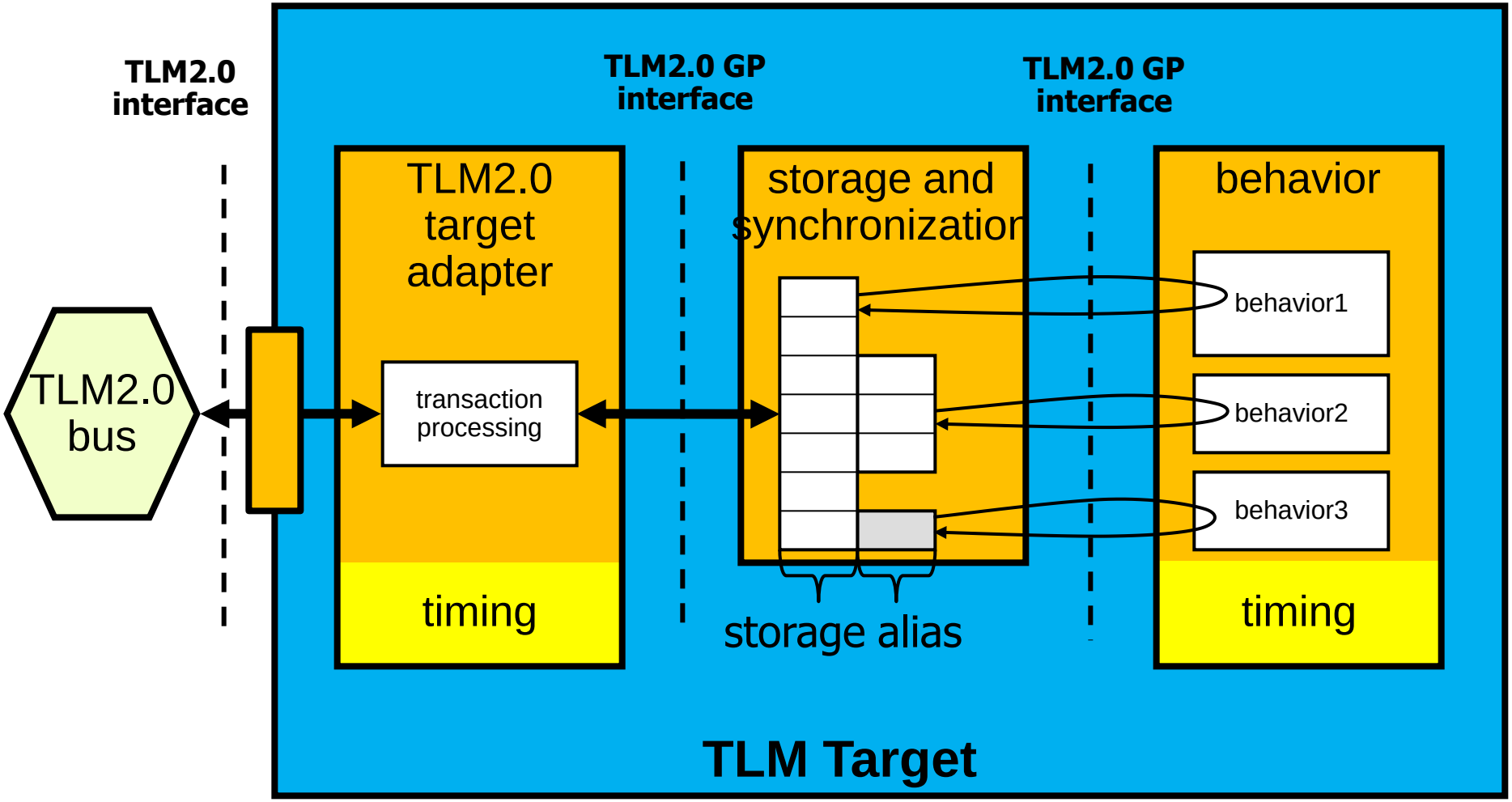
SCML overview



SCML overview



SCML Target Modeling Pattern



Automated System Integration

IP Delivery

ARM

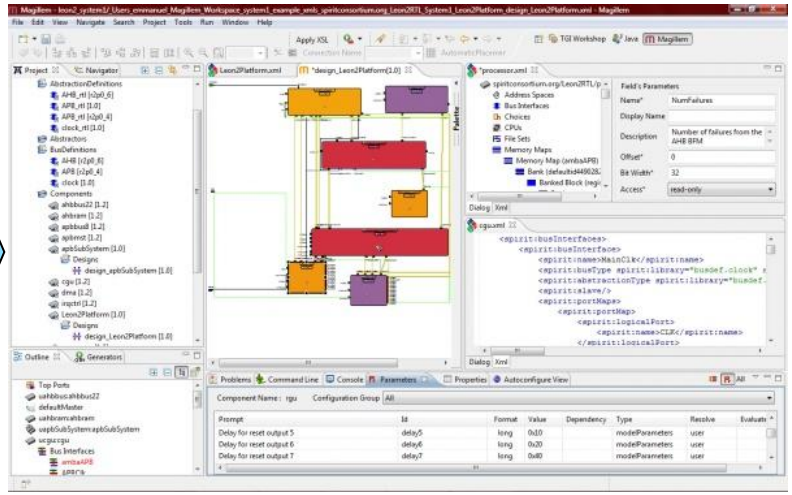
NXP

SYNOPSYS®

VIRAGE LOGIC

evatronix

IP-XACT™ Design Environment



Design Views

- RTL
- SLMODEL
- VERIFICATION
- METADATA
- DOCUMENT

IP-XACT™ System Integration Automation

Mixed Language
VHDL Verilog
SystemC

Mixed Level
RTL & TLM

Mixed Signal
Verilog-AMS

Contents

Early SW development on SystemC platform Introduction

NXP products contain both hardware and software

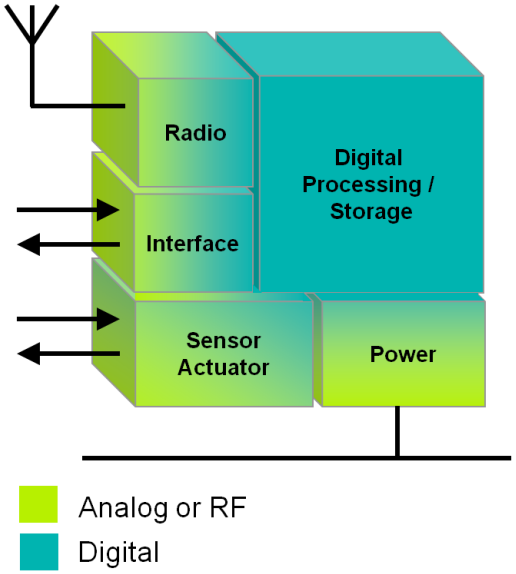
- The software is executed by the hardware

Different types of software development in NXP

- Firmware, driver software, middleware, …

Traditionally software development starts when hardware is available

- RTL, FPGA, silicon,…



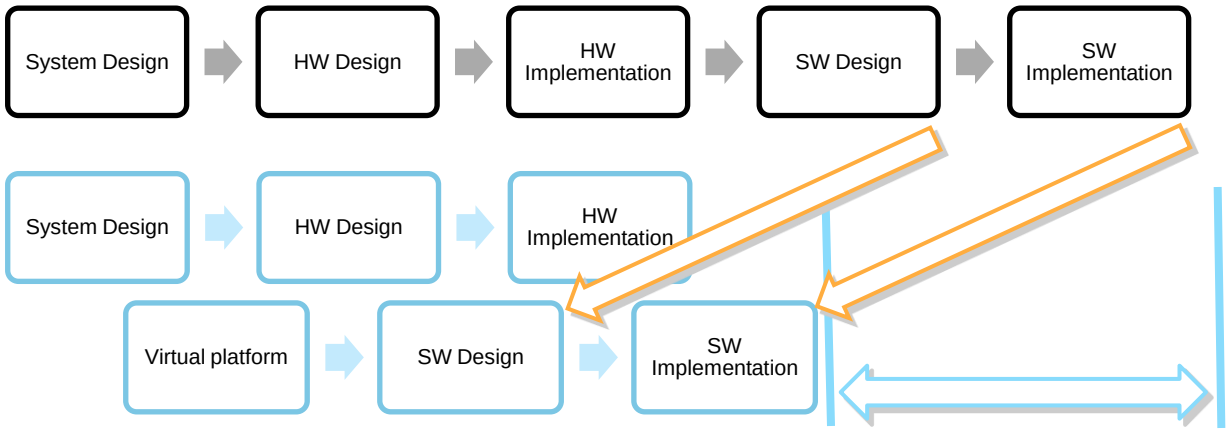
Early SW development on SystemC platform
Our contribution & added value

Our contribution is virtualization of hardware platforms

- Early availability of hardware models for software development
- Existing software development environment can be used with virtual platform

Our added value is to reduce product development times

- Software development can start before hardware RTL model is available



Early SW development on SystemC platform advantages

Shorter TtM

- Early SW development when RTL/Si is not available yet
- Concurrent HW/SW development

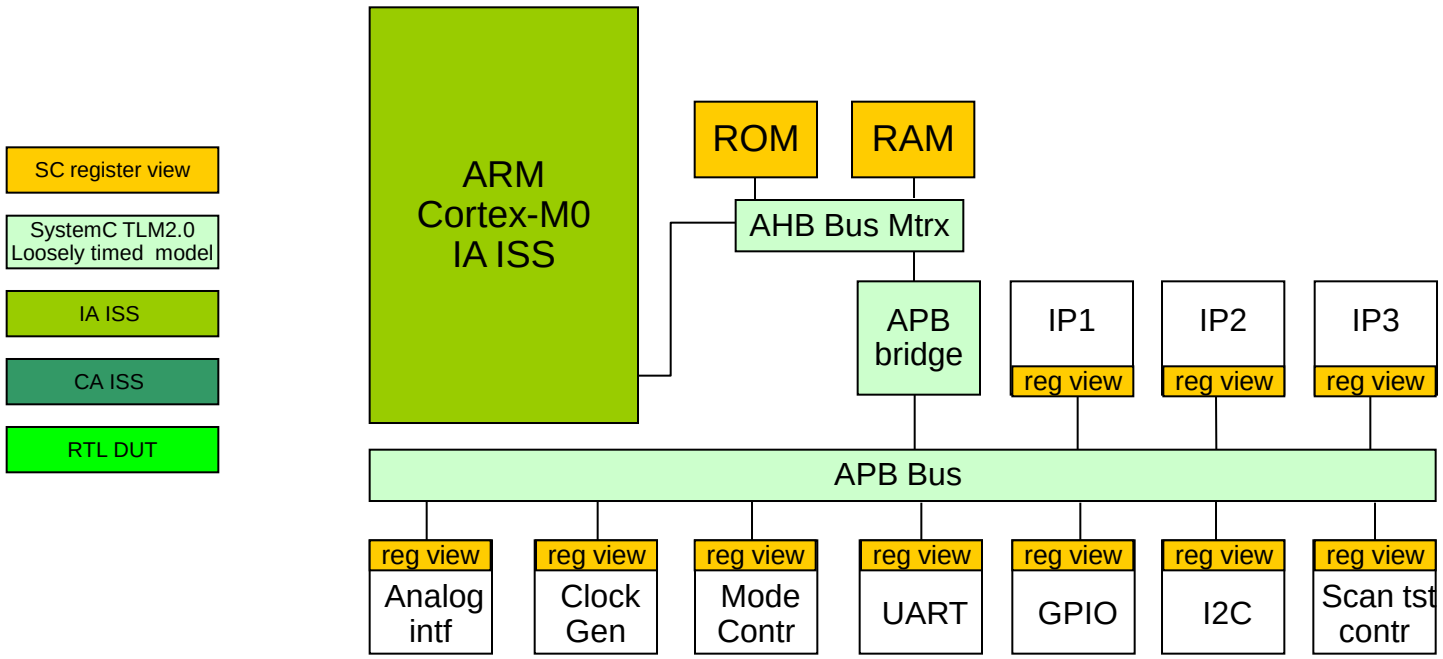
Real High speed SW development environment

- Fast ARM/Keil models
- Binary compatible with final SW on Si
- SW debug capabilities using RVD/Keil uVision

Better product quality

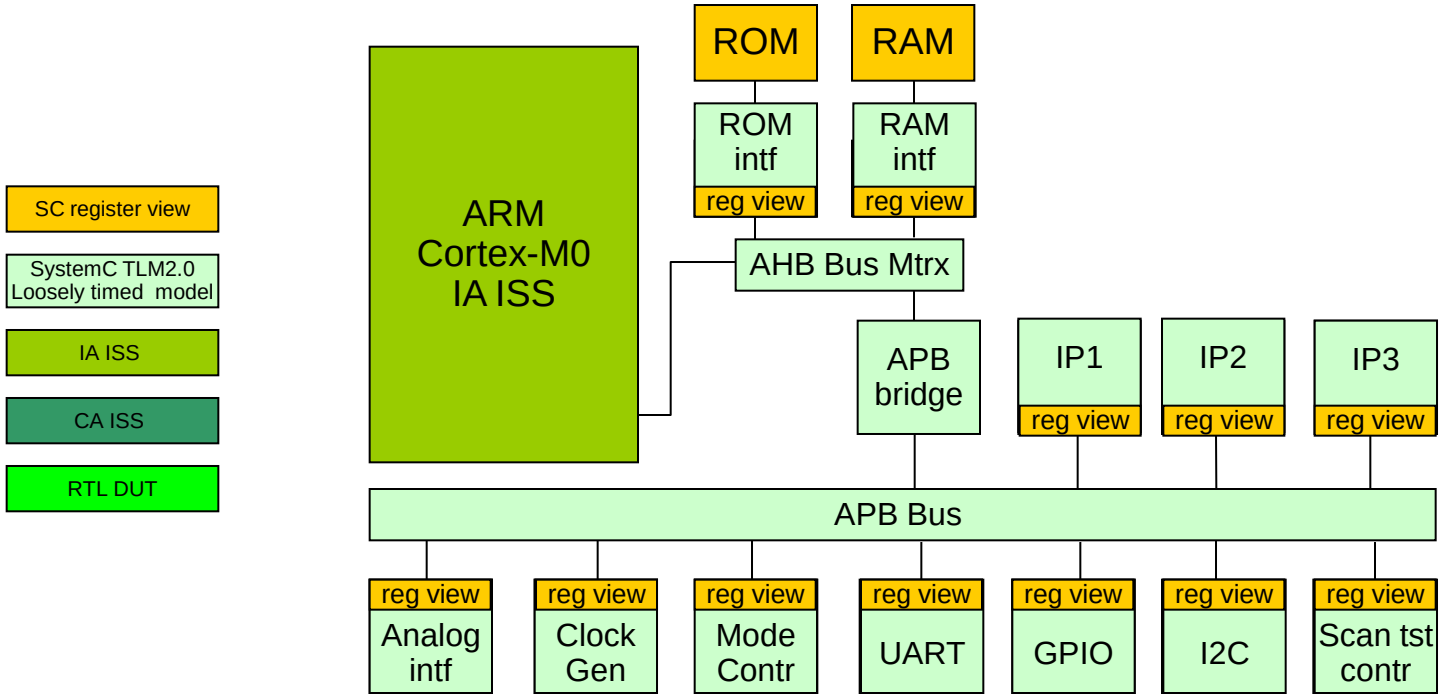
- Synchronized debug environment between Hard- and Software
- Full visibility inside SW on CPU (ARM RVD/Keil uVision), inside RTL (SimVision) and TLM testbench
- Ability to simulate complete system

Early SW development on SystemC platform



- Initial SystemC platform and target peripheral register views are automatically generated from system register map information
- Generated SystemC reference platform contains ISS of ARM Cortex M0
  - Keil ISS connected to uVision (windows only)
  - Fastmodel ISS connected to RVD (both windows and linux)
- An OS can be booted on the SystemC reference platform
- The Reference platform is ready for basic SW development (register read/write behavior)

Early SW development on SystemC platform

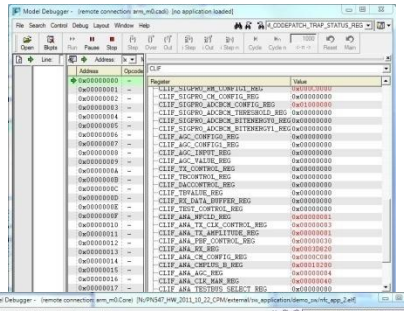


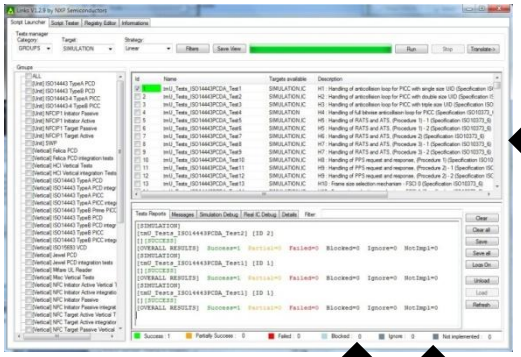
- Behavior can be added to SystemC IP model templates
- The Reference platform is ready for further (driver/application) SW development

Early SW development on SystemC platform HW/SW co-debugging

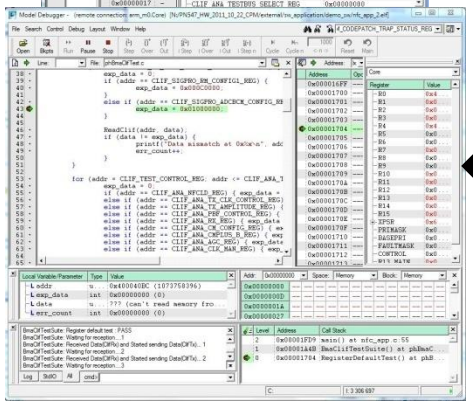
Test environment

HW register values

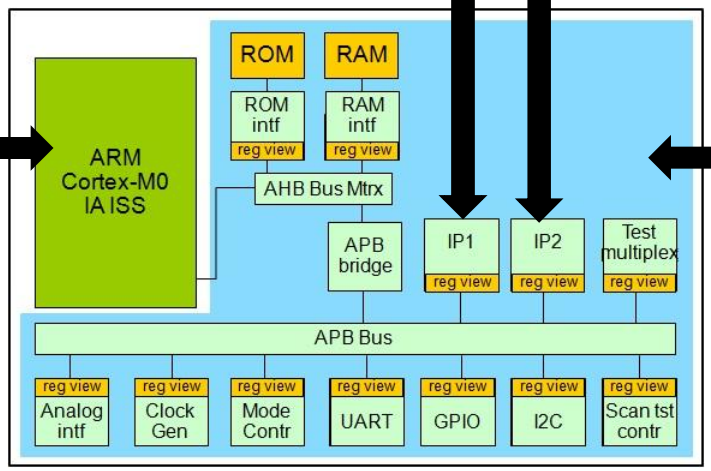


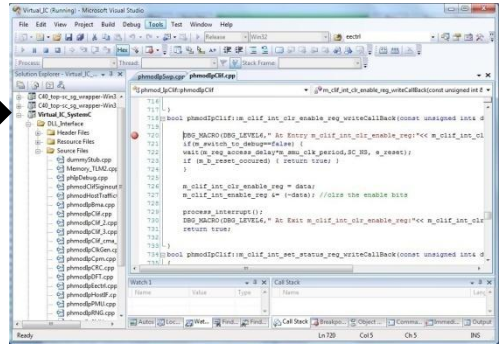


Test database



SW debugger
(ARM RVD/Model debugger)





HW model debugger
(Visual Studio)

Architecture exploration on SystemC platform

Performance metrics based on transactions rather than signals

- Simplifies performance analysis
- Need monitors that translate signal level to TLM

Fast modeling of traffic generators and other IPs

- Generic IP models can be used
- IP models can be highly configurable
- Abstract behavior sufficient

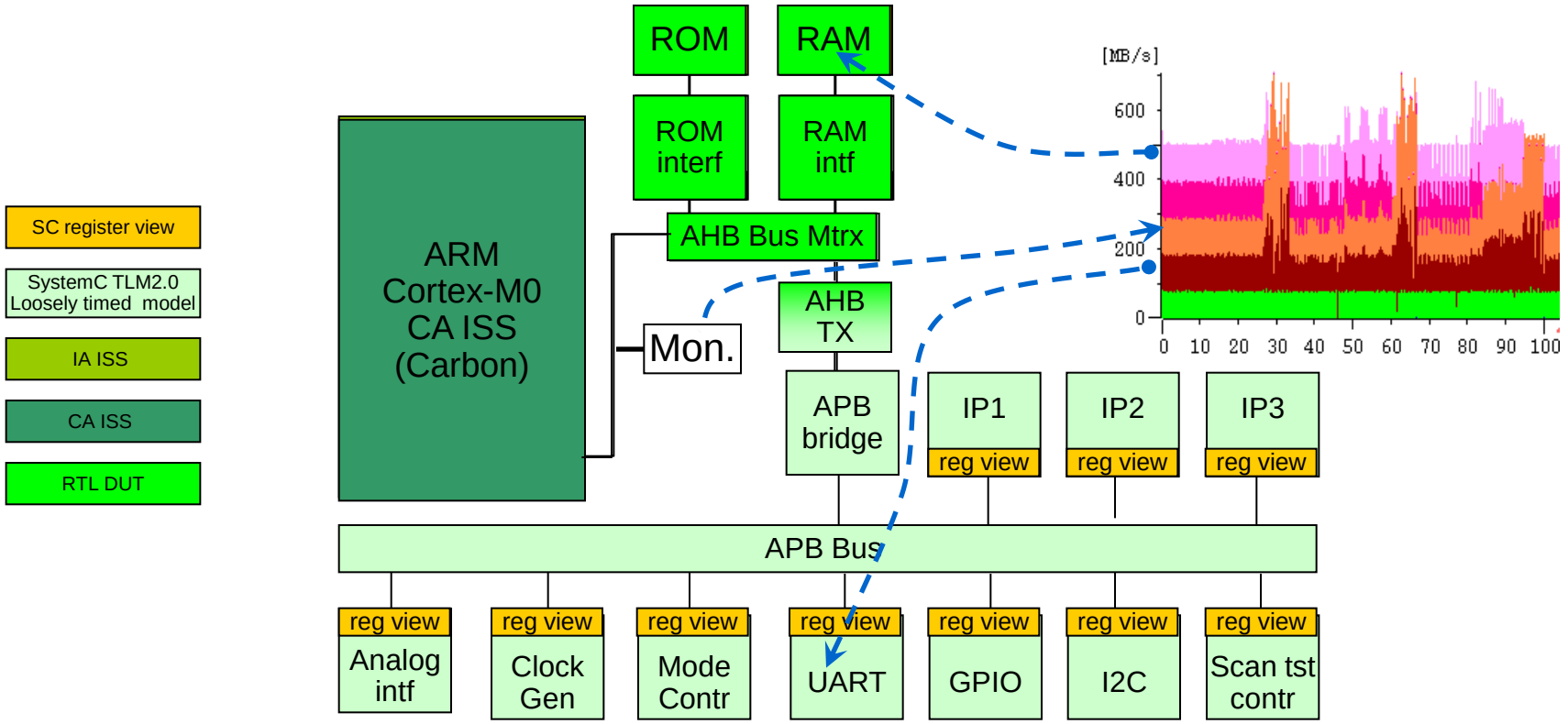
Easy debug of HW/SW

- Full visibility inside SW on CPU (ARM RVD/Keil uVision), inside RTL (SimVision) and TLM testbench
- C++ debugging of IP models

Higher simulation speed than RTL

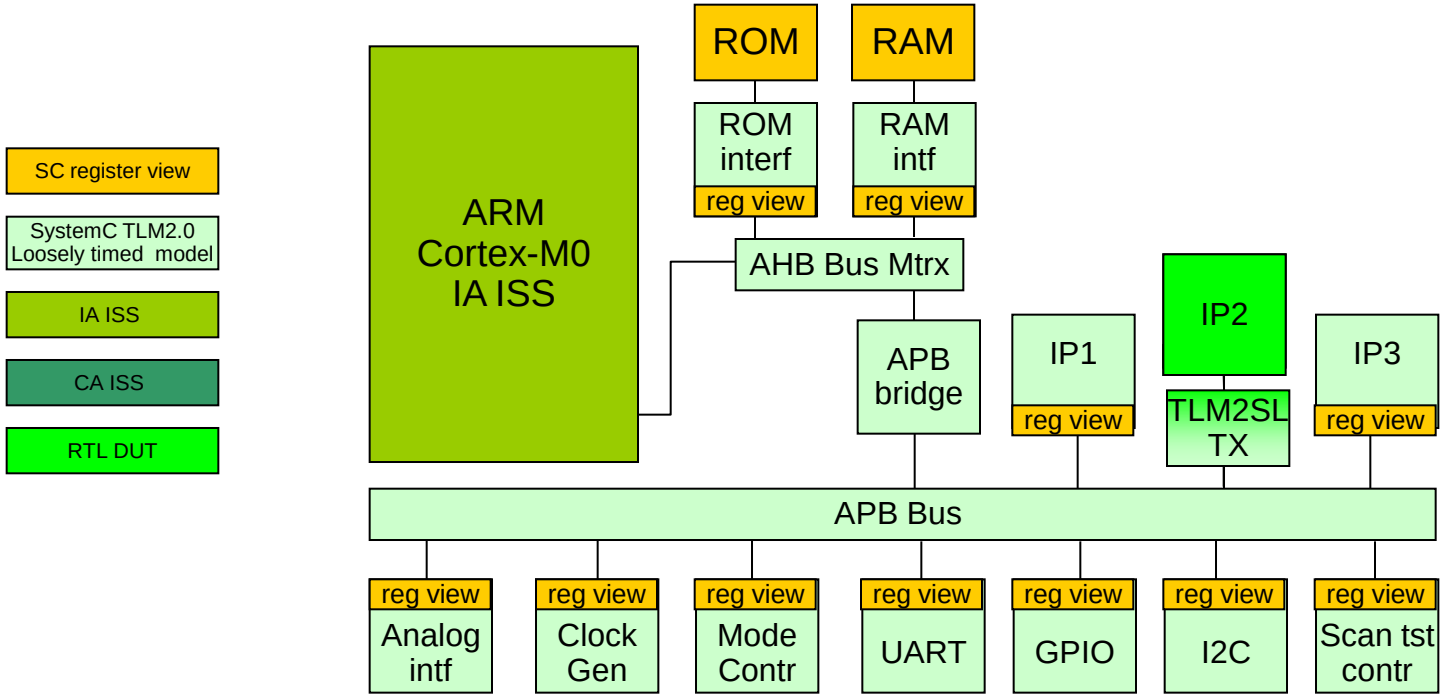
- Abstracted behavior simulates faster

Architecture exploration on SystemC platform



- ARM Cortex M0 Instruction Accurate ISS is replaced by Carbon Cycle Accurate ISS
- Critical components for Performance analysis are replaced by their RTL counterpart
- Performance analysis is done through co-simulation

RTL IP verification using SystemC platform

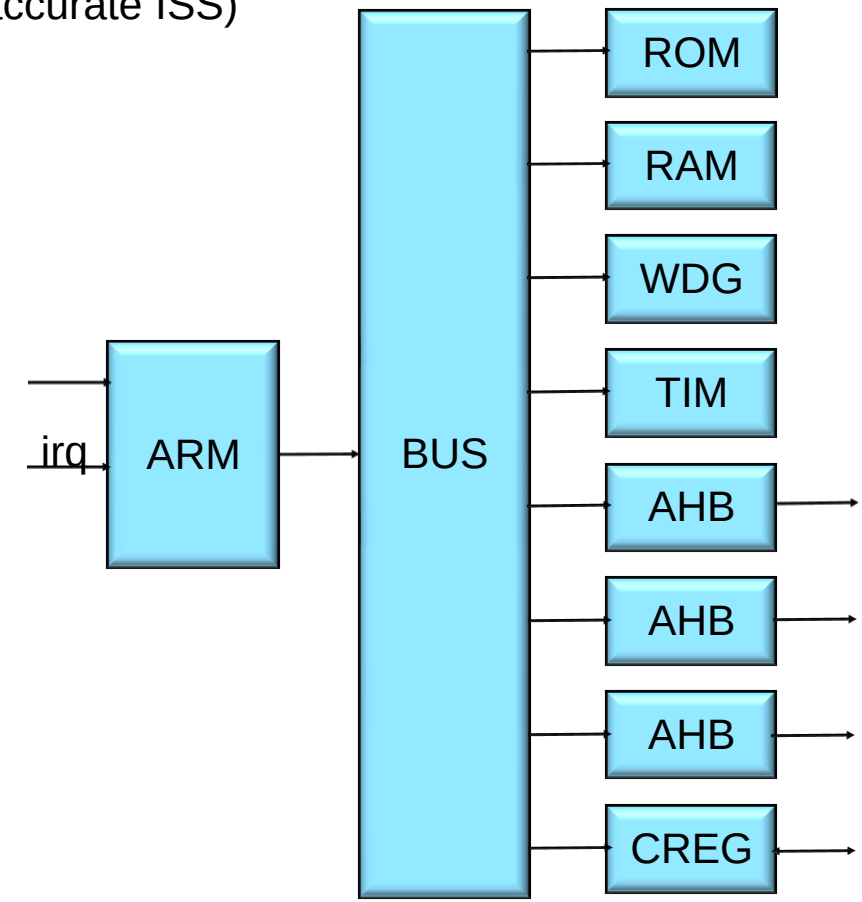


RTL IP functional verification through

- Reuse of SystemC platform for SW development
- Reuse of co-simulation setup for architecture exploration

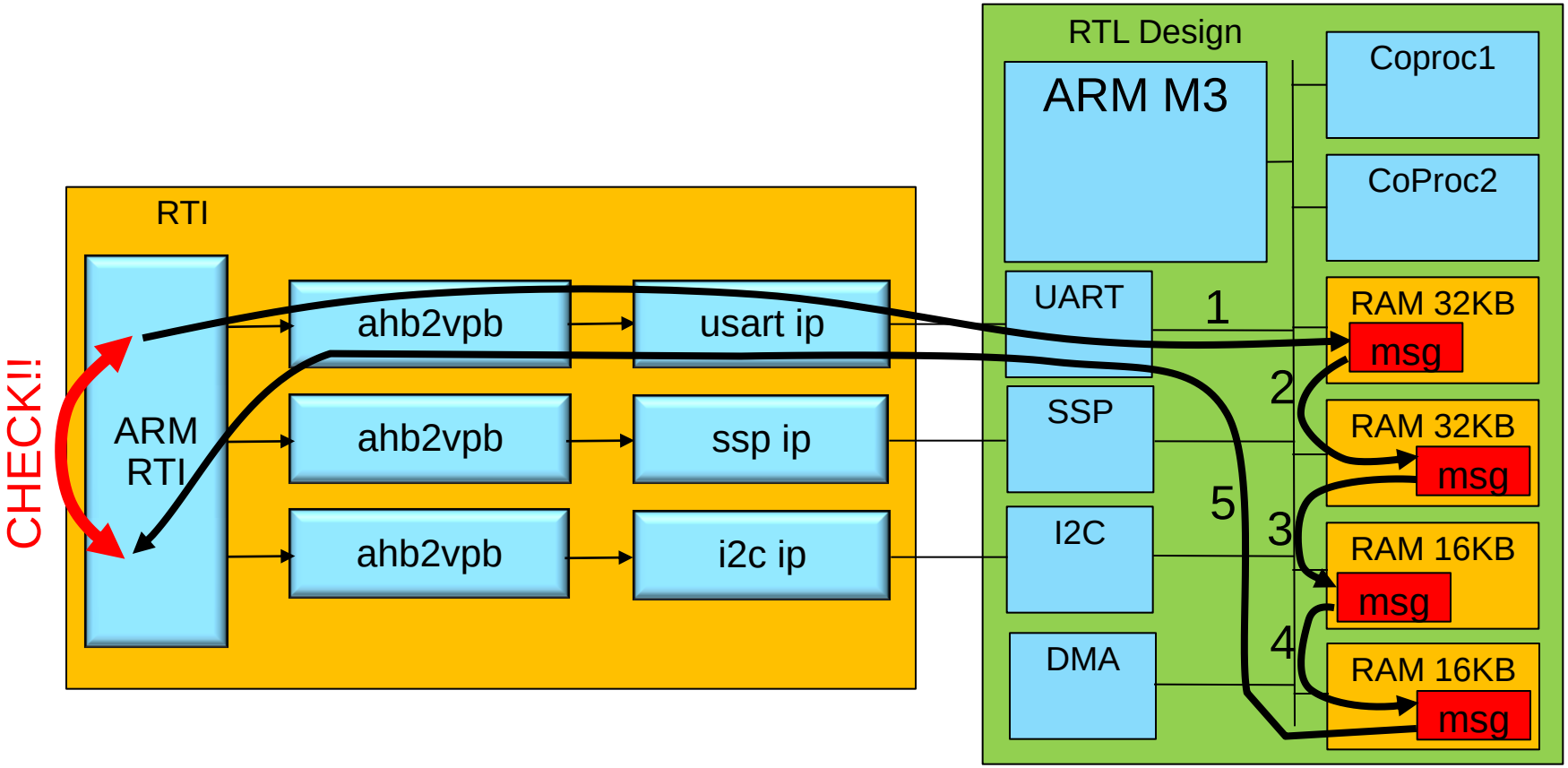
RTL System verification with Reusable Test Infrastructure (RTI)

- ARM FastModel Cortex-M3 (Instruction-accurate ISS)
  - Includes interrupt controller
  - Includes software compilation tool chain
  - Includes software debug solution
- TLM bus
- TLM memory (ROM and RAM)
- TLM timer (WDG and TIM)
- TLM-to-AHBlite signal slave transactor
  - Includes AHB clock and reset
- TLM clock generator
- TLM reset generator
- TLM CREG I/O



*TLM: Transaction Level Model
*IRQ: Interrupt request

RTL System verification with Reusable Test Infrastructure (RTI)



Contents

NXP Semiconductors

Virtual platforms

Modeling Methodology & Libraries

Application of Virtual Platforms in NXP

Conclusions

Conclusions

- BL usage in different projects shows the added value of virtual platforms (TtM, TtV, quality)
- Standards based SystemC modelling methodology enables easy creation and maintainability of abstract system level models
- IP-XACT based Automatic System Integration and Generation is key
- Virtual platforms serve a wide variety of use-cases

THANKS

Questions?