SoCKET Collaborative Project

(SoC toolKit for critical Embedded sysTems)

June 2008 – November 2011

Noordwijk,
V.LEFFTZ & J. LACHAIZE
(Astrium)

Agenda

- SoCKET project presentation
- Proposed Hardware/Software Co-Design Flow
- Focus on key technologies + perspectives
  - SystemC/TLM Modelling
  - Assertion Based Verification
  - High Level Synthesis

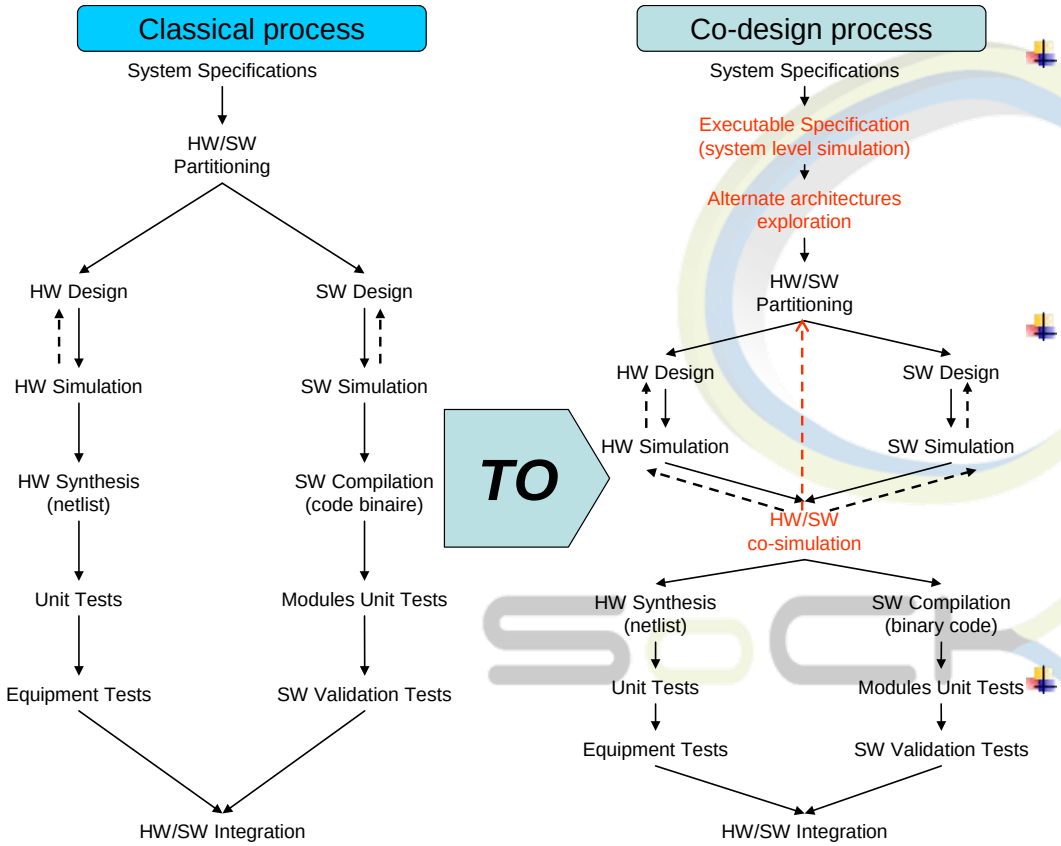
Agenda

- SoCKET project presentation
- Proposed Hardware/Software Co-Design Flow
- Focus on key technologies + perspectives
  - SystemC/TLM Modelling
  - Assertion Based Verification
  - High Level Synthesis

SoCKET: Objectives

- Define a **“seamless” development flow**, integrating the **equipment qualification/certification**, **from the system level**, to the **IC** and **validated SW** on these ICs;
- **Master** the **SoC solutions** for critical embedded systems;
- **Master** the “**system dimension**” (software + hardware) into the SoCs integration problematics;
- **Master** the **complexity**, the **time cycle reduction**, **design optimisation** of SoC-based systems;
- Evaluate the **HW simulation models** (get from the design flow) usage for the integration and the validation of the critical embedded SWs.

“Seamless” design flow



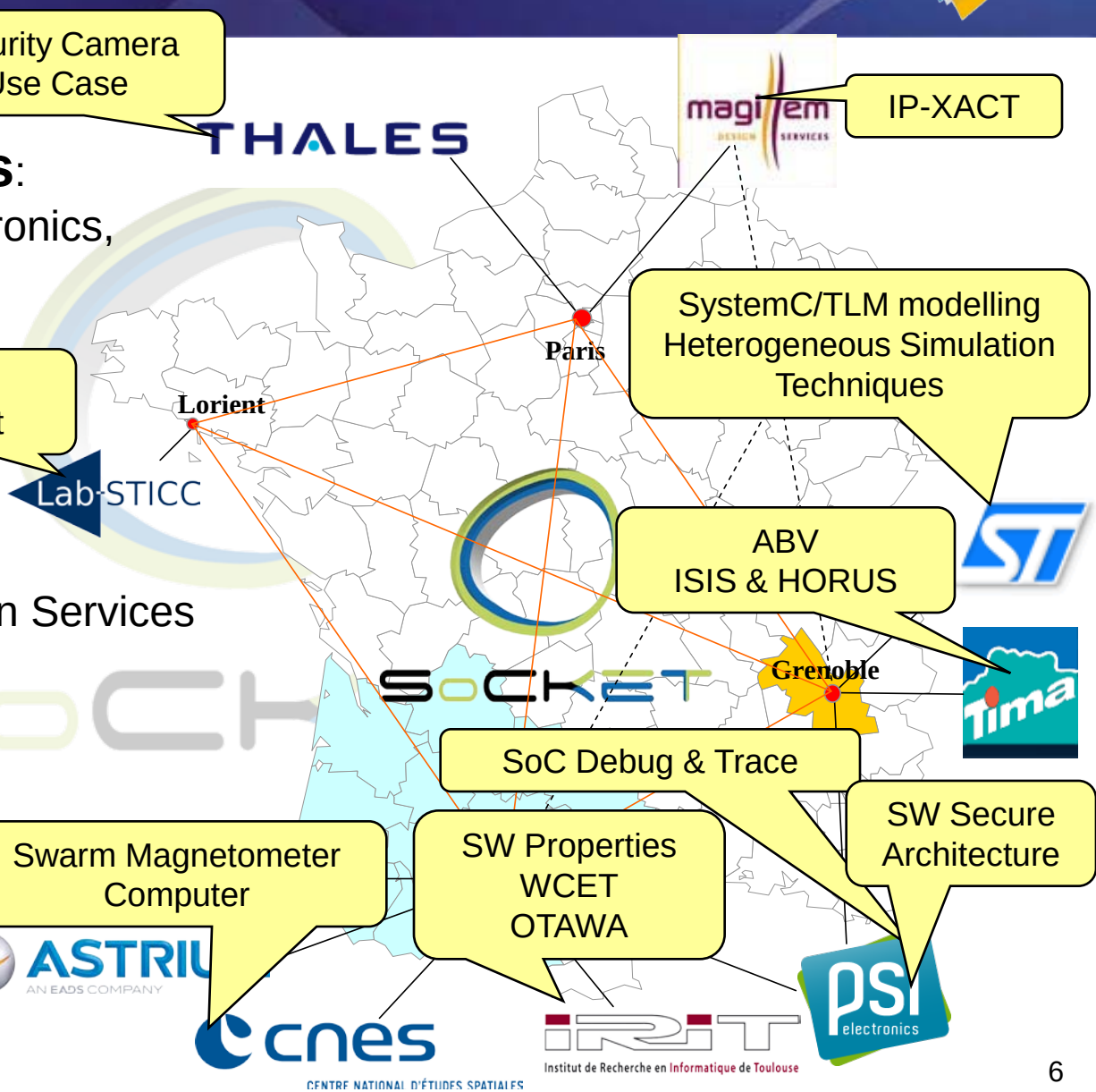
- Formalisms unification
  - Remove any semantic holes into HW/SW interfaces
- Models transformation operators
  - Automation
  - Traceability
  - Overall coherency insurance
- Tools interoperability
  - Keystone of 2 previous points

SoCKET: Consortium

- **International groups**:
Airbus, Astrium, STMicroelectronics, Thalès R&T

- **PMEs**:
PSI-S, PSI-E, Magillem Design Services

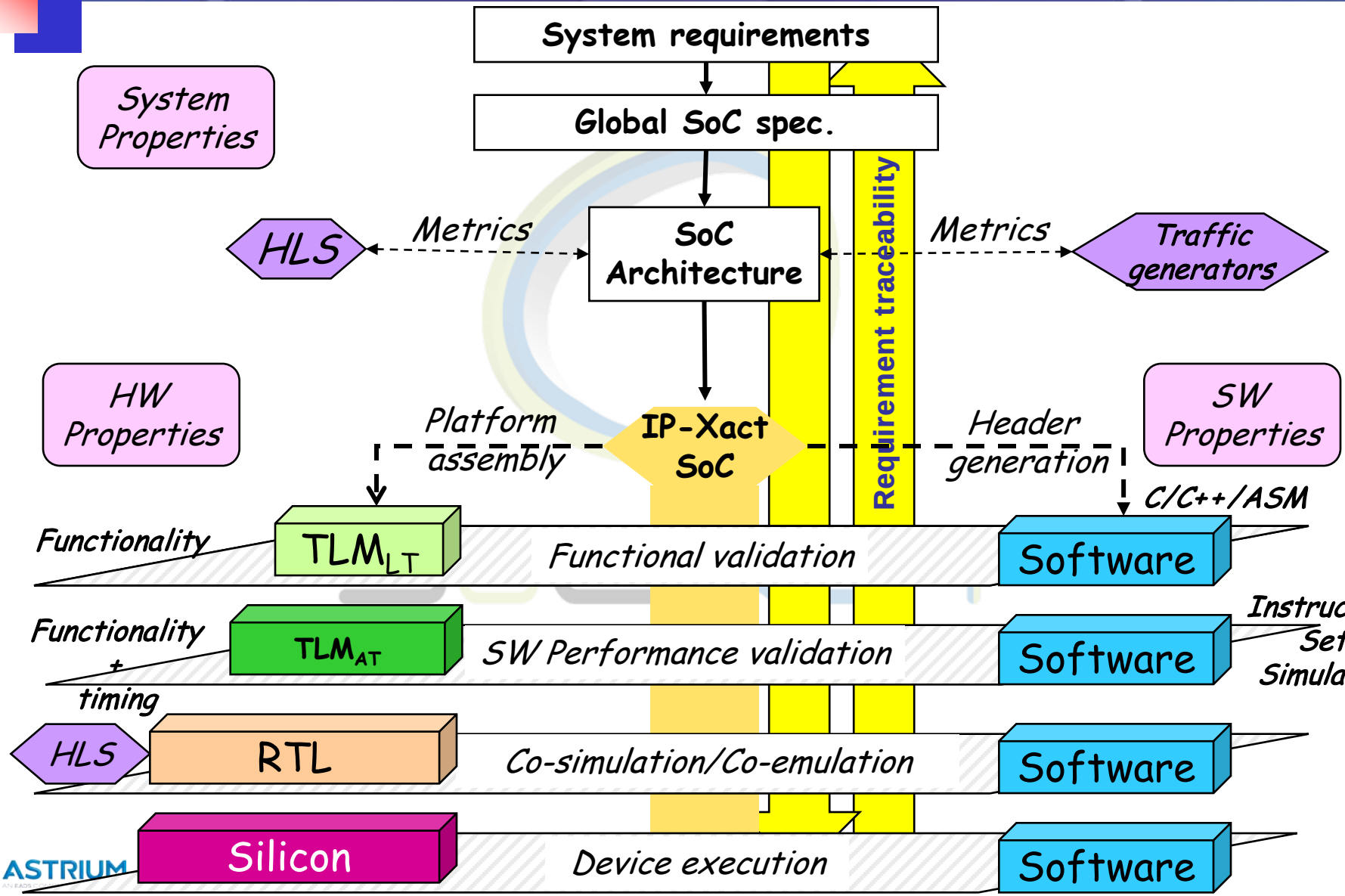
- **Academics and Research Centers**:
CNE
TIM
AST



Agenda

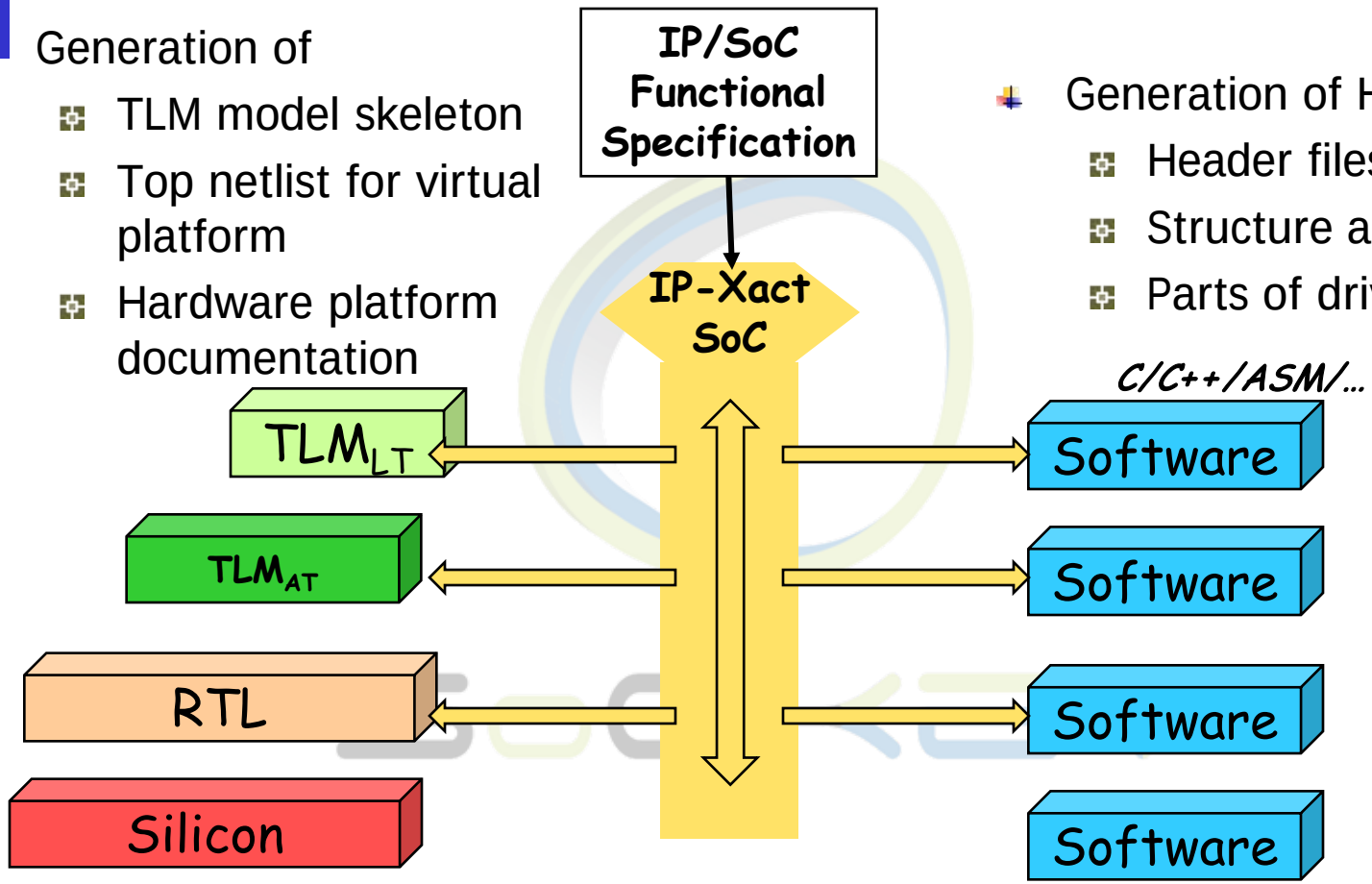
- SoCKET project presentation
- Proposed Hardware/Software Co-Design Flow
- Focus on key technologies + perspectives
  - SystemC/TLM Modelling
  - Assertion Based Verification
  - High Level Synthesis

SoCKET Co-Design flow



Content management with IP-Xact

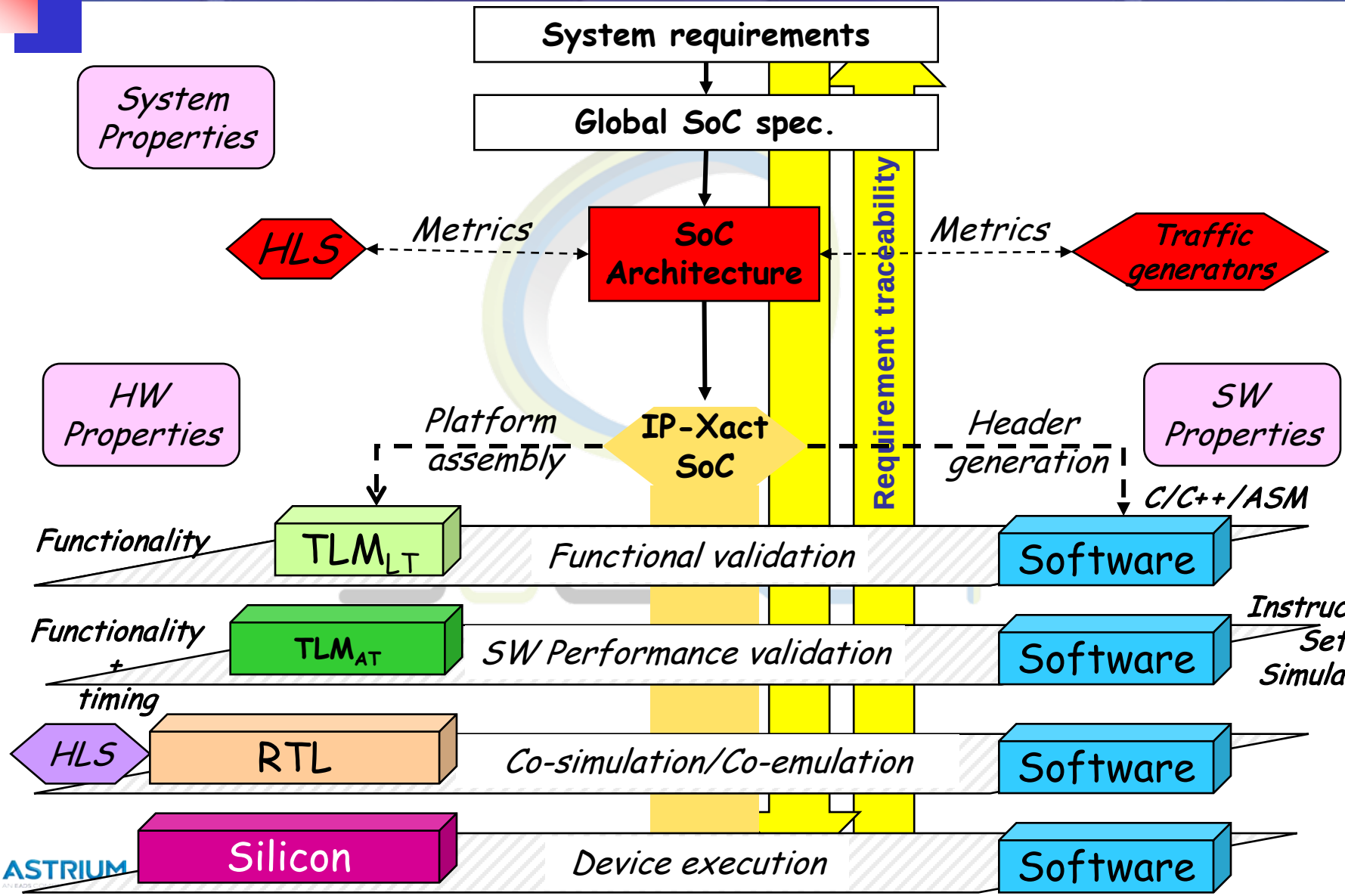
- Generation of
  - TLM model skeleton
  - Top netlist for virtual platform
  - Hardware platform documentation



- Generation of HDS layers
  - Header files
  - Structure and Functions
  - Parts of drivers

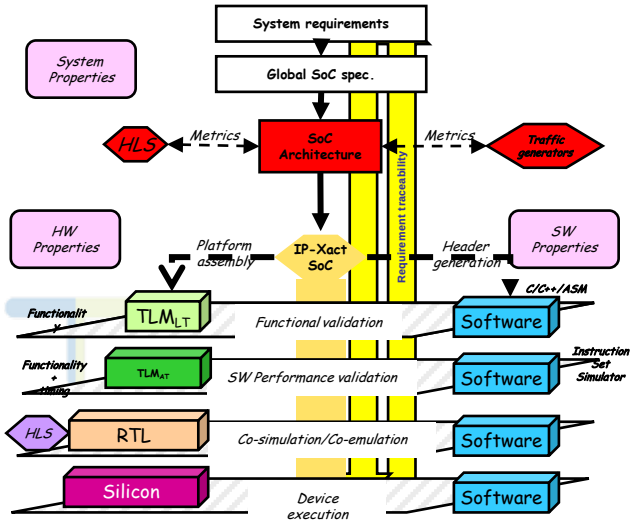
- Consistency vs. heterogeneity
  - Various formalisms (architectures, timing, functionality, power, ...)
  - Various levels of abstraction and languages
  - Implementation in hardware and software

SoC Architecture definition



SoC Architecture definition

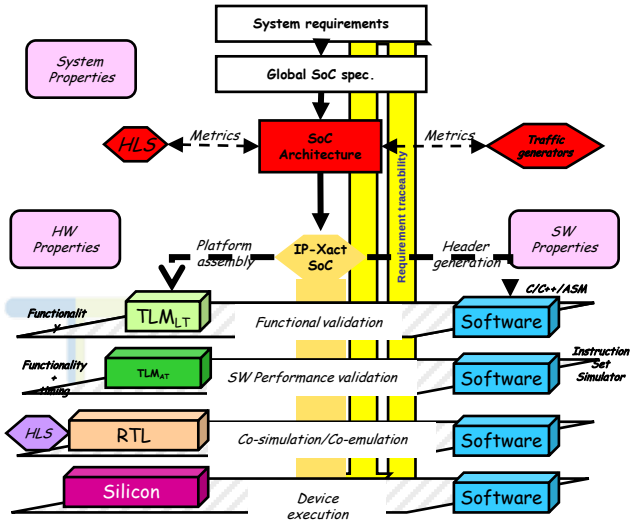
- Define the appropriate partitioning between hardware and software
  - Conform with system requirements
- An architect-driven decision process guided by metrics
  - High Level Synthesis
    - Provides IP internal information
  - IP Traffic Generators
    - Assess bandwidth and latency scenarios



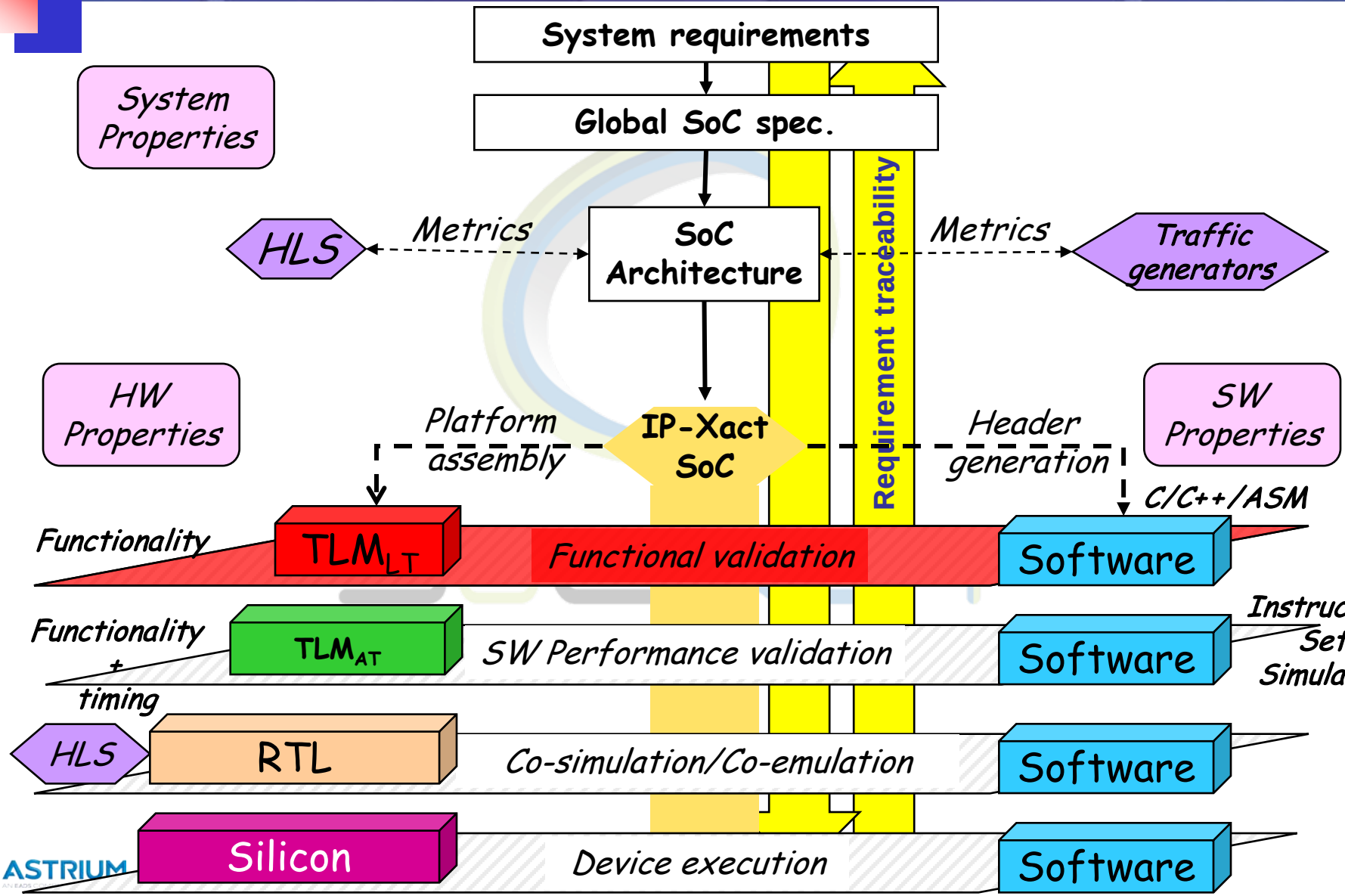
- Resulting architecture is described in the IP-Xact format
  - Refined requirements are associated to the architecture

Traffic Generators

- Objective
  - Analyze interconnect bandwidth and latency
- Means
  - For each IP, characterize traffic profile
  - Assemble a platform with a BCA/RTL interconnect and memory models
  - Generate traffic
  - Exploit results with analysis tools

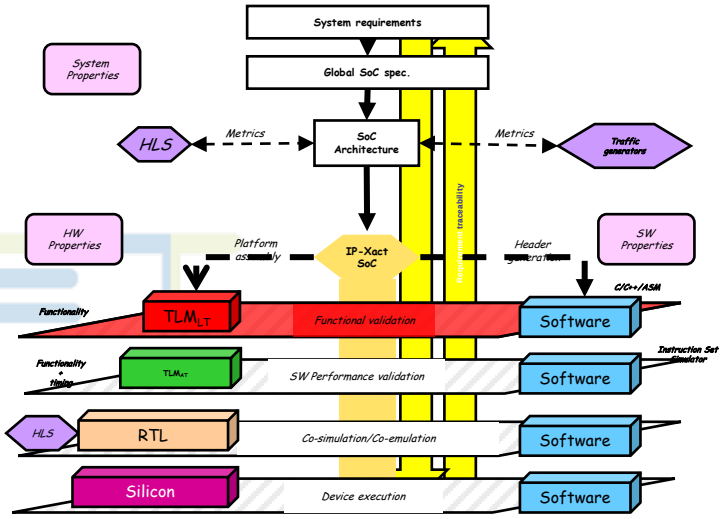


Transaction Level models (LT)



Transaction Level models (LT)

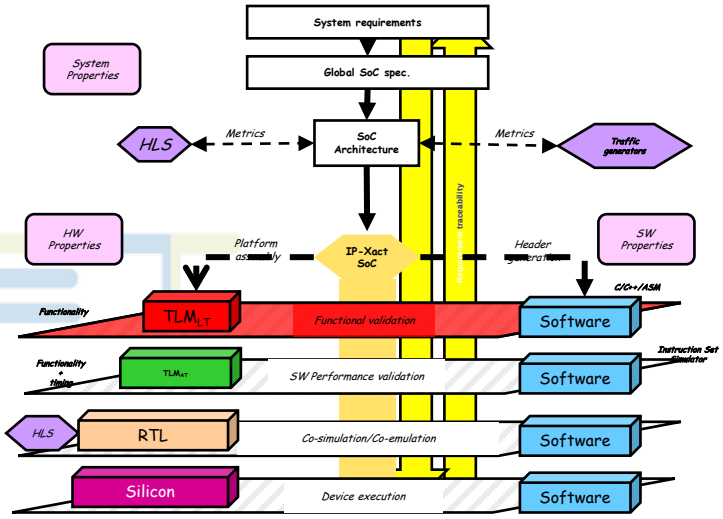
- Model IPs/subsystems at the transaction level
  - Bit true behavior & communication
  - System synchronization points
  - No clock/cycle, but functional timing (e.g. *timer*)
  - Fast to implement and simulate
- TLM LT models often built using
  - C reference model
  - TLM wrapper
    - Model registers
    - serve read/write accesses
- Used for
  - Embedded software development
  - Functional verification activities for RTL IPs



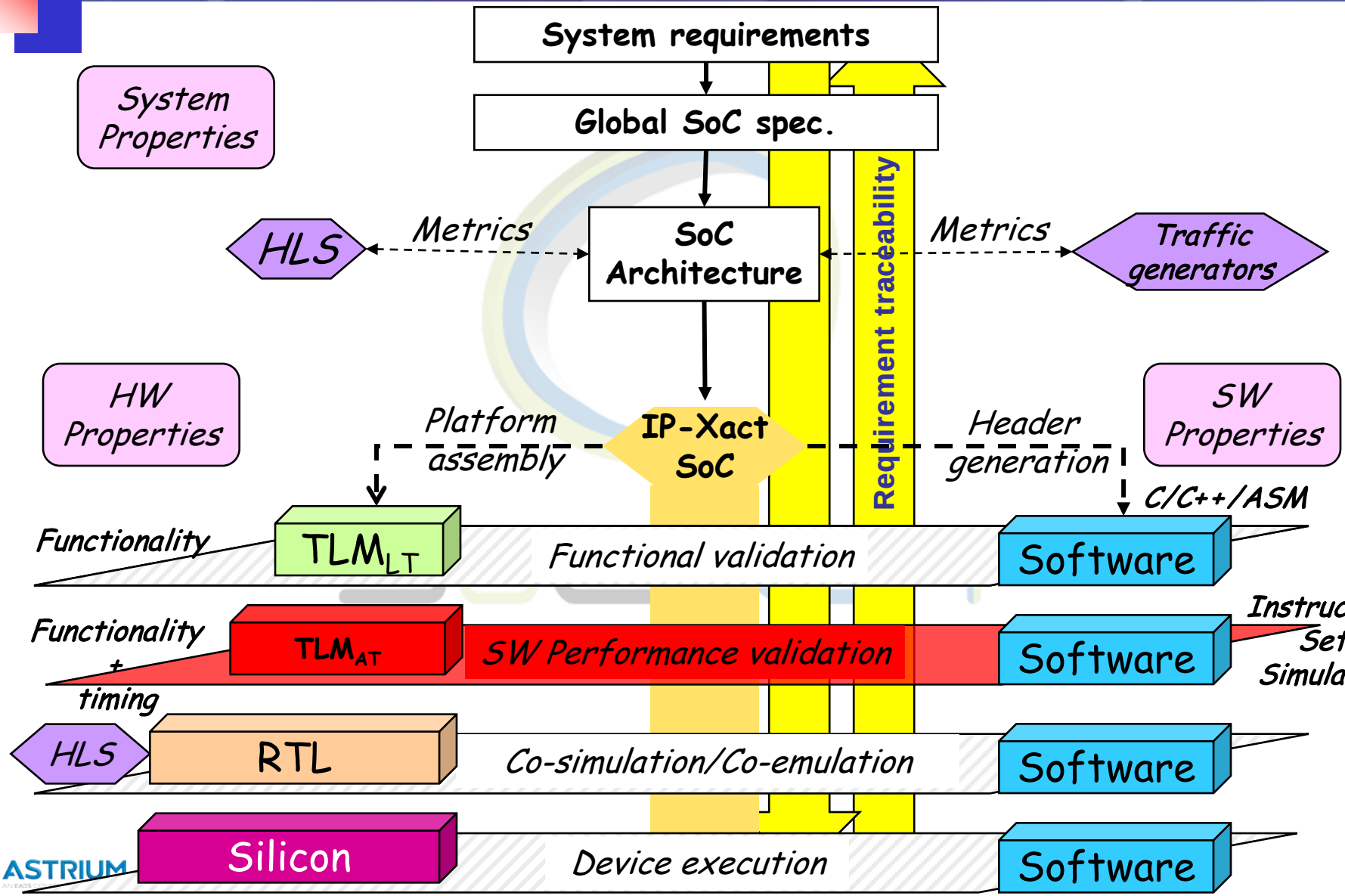
14

Validation of LT models

- The TLM LT model is the golden reference
- Functional test suite is built using the LT model
- TLM LT testbench is used to validate the RTL IP
- High level of confidence for reusing the model

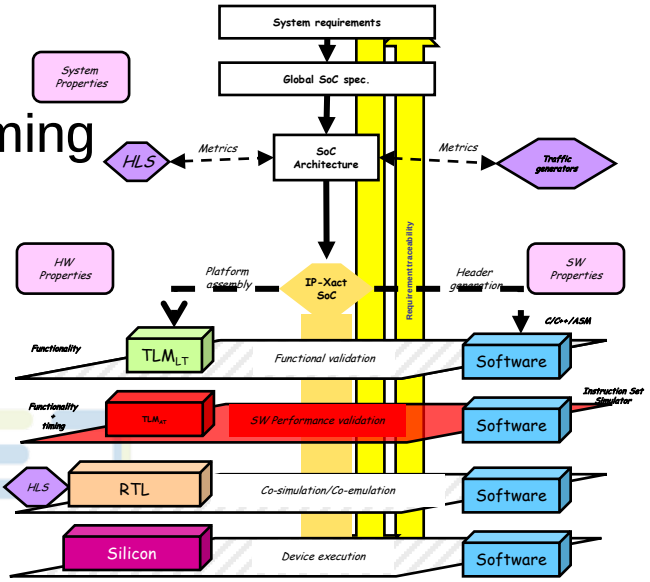


Transaction Level Models (AT)



Transaction Level Models (AT)

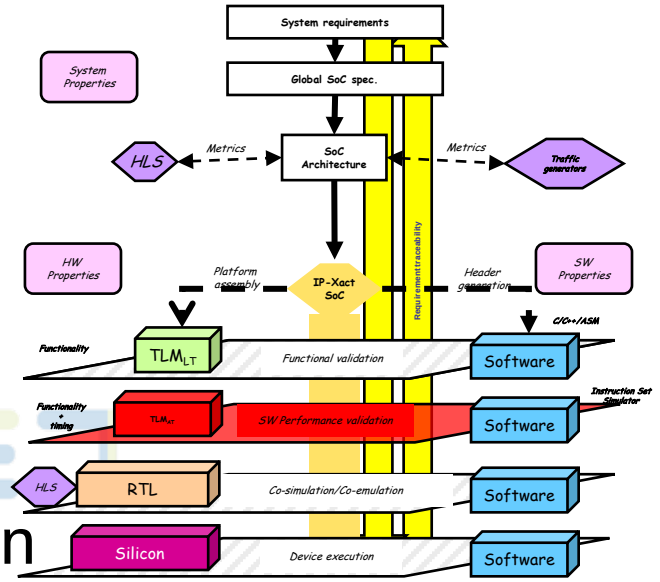
- Targeting performance evaluation
  - Hardware architecture
  - Software
  - Captures micro-architecture information/timing
  - Timing accuracy may be trimmed
- Several technical options
  - LT Model refinement -> rewriting
  - LT Model annotation
  - Composition of LT and T models
  - Generation of a CA model from RTL



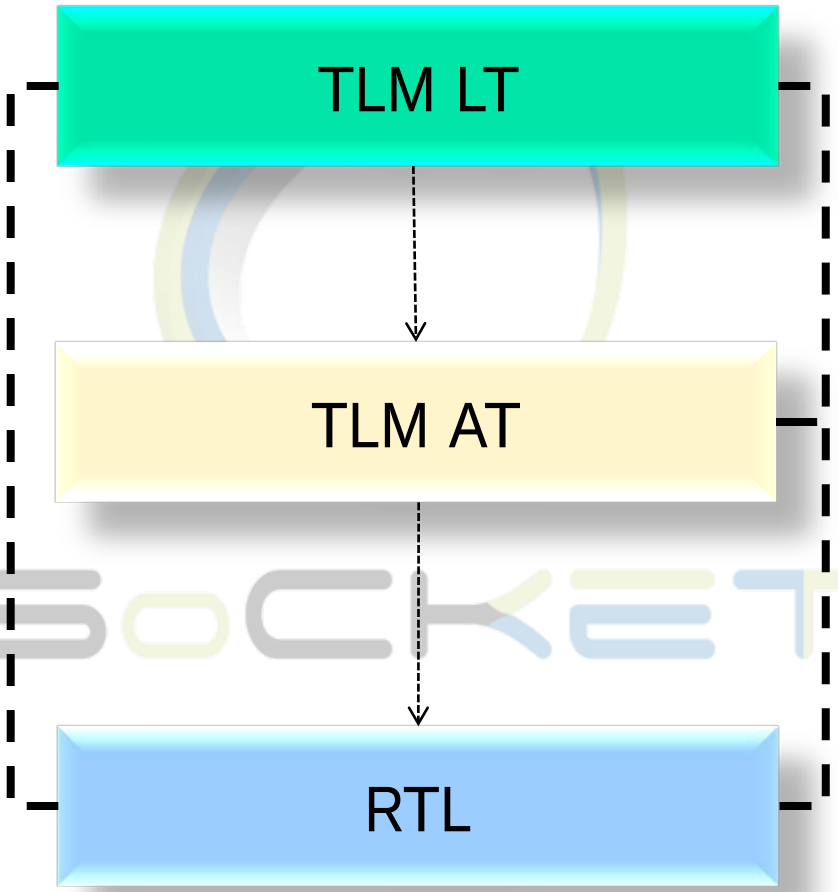
17

Validation of AT models

- Functionality of AT models can be assessed with the LT tests
- Building the temporal model is difficult
  - Extract timing information from the RTL
  - Implement the micro-architecture model
  - Statistical approach
- And impacts significantly the simulation speed
- The hard topic is the temporal validation
  - Reuse of Implementation Verification Patterns when available
  - As difficult as the validation of cycle accurate models



Comparing abstraction levels

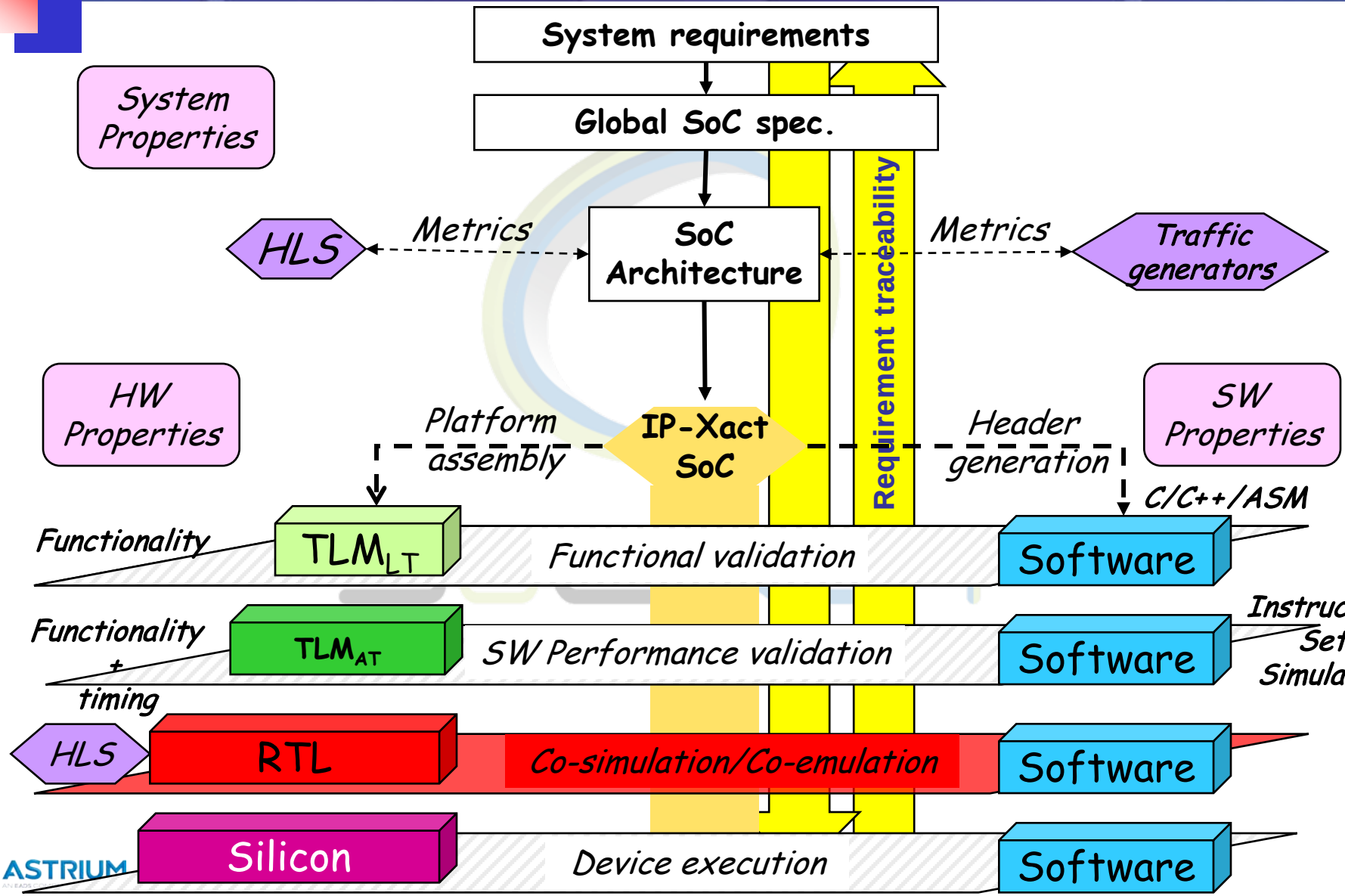


Same functional behavior

Same functional behavior

Same timed behavior

RTL Level


System requirements
Global SoC spec.
System Properties
HLS
Metrics
SoC Architecture
Metrics
Traffic generators
Requirement traceability
HW Properties
Platform assembly
IP-Xact SoC
Header generation
SW Properties
C/C++/ASM
Functionality
TLM_LT
Functional validation
Software
Functionality + timing
TLM_AT
SW Performance validation
Software
Instruction Set Simulator
HLS
RTL
Co-simulation/Co-emulation
Software
Silicon
Device execution
Software

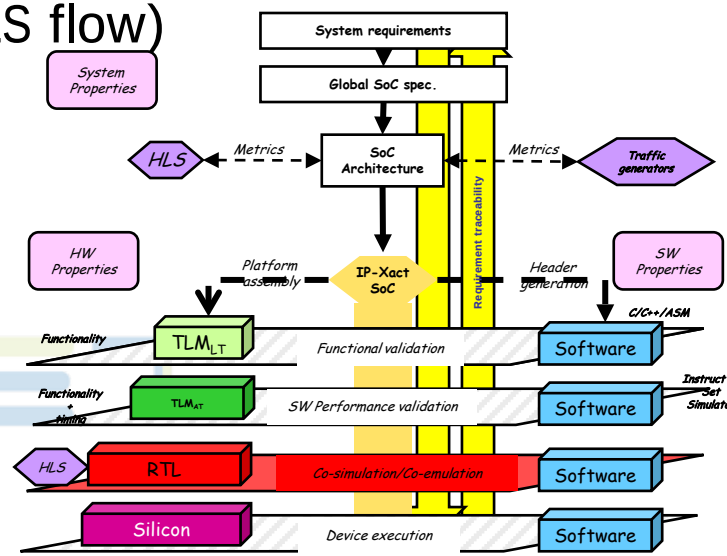
RTL level

- Entry point for logic synthesis flow

- RTL models might be
  - Implemented manually
  - Generated from higher description (HLS flow)

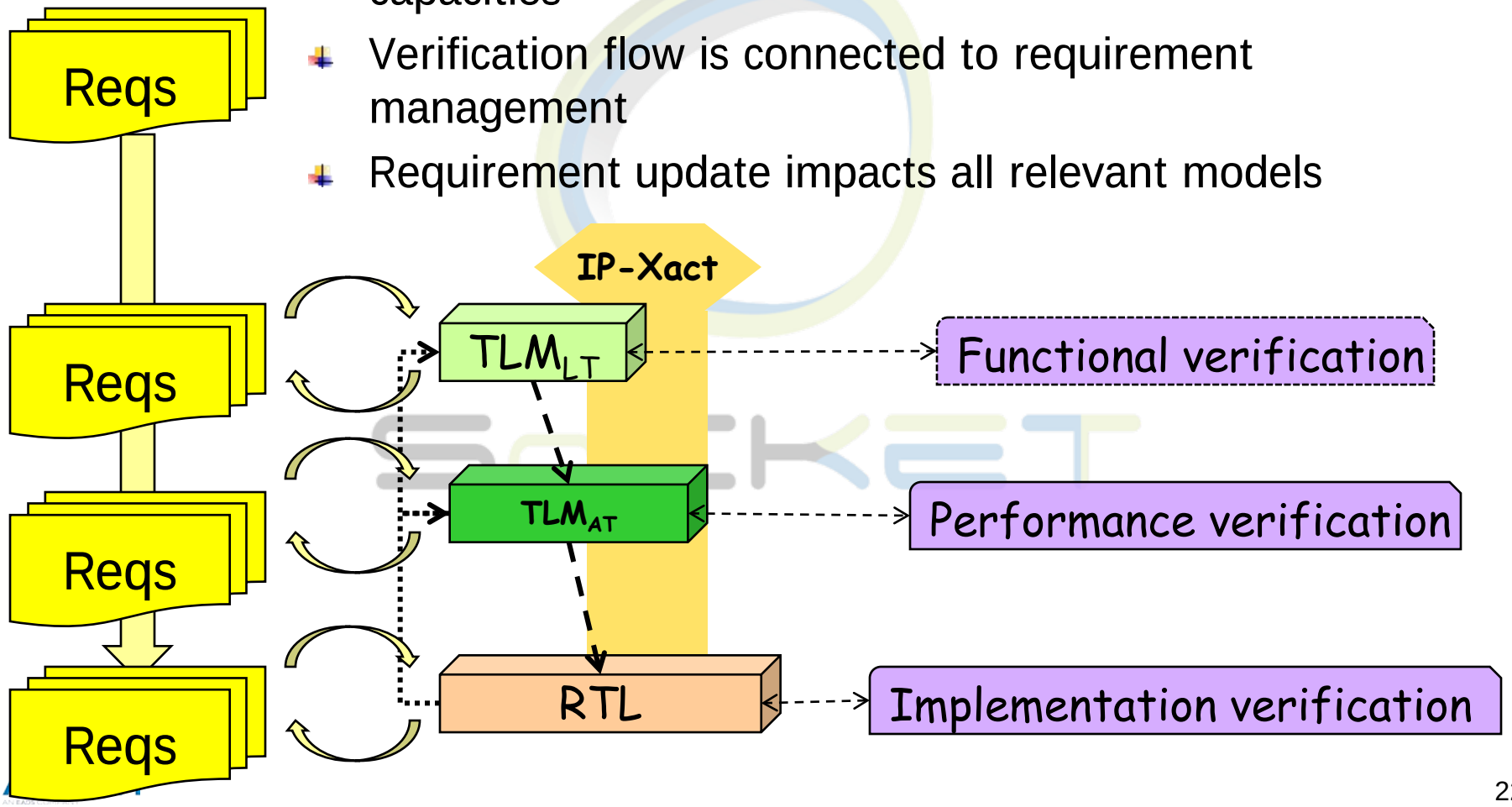
- Co-simulation
  - Joint simulation of SystemC/TLM and VHDL/Verilog models

- Co-emulation
  - Simulation of SystemC with the execution of VHDL/Verilog models mapped on a hardware emulators



Verification concern

- Each model is a golden model for his level of abstraction
- Verification at a lower level extends the validation capacities
- Verification flow is connected to requirement management
- Requirement update impacts all relevant models



Agenda

- SoCKET project presentation
- Proposed Hardware/Software Co-Design Flow
- Focus on key technologies + perspectives
  - SystemC/TLM Modelling (Time modelling)
  - Assertion Based Verification
  - High Level Synthesis

Programmer's View (PV) or functional simulation (LT)

- Time is not represented, only functionality is modelled.
- Functional synchronization is necessary. It is done at System Synchronization Points (SSP): configuration registers access, interrupts and all state alternating accesses.

The need for time

- Performance measurements
- Design Space Exploration
- Feasibility assessment

...how ???

- Precision?
- Modelling granularity?
- Simulation performance?

The obvious solution: mixing time and functionality

- It works !!!

...but...

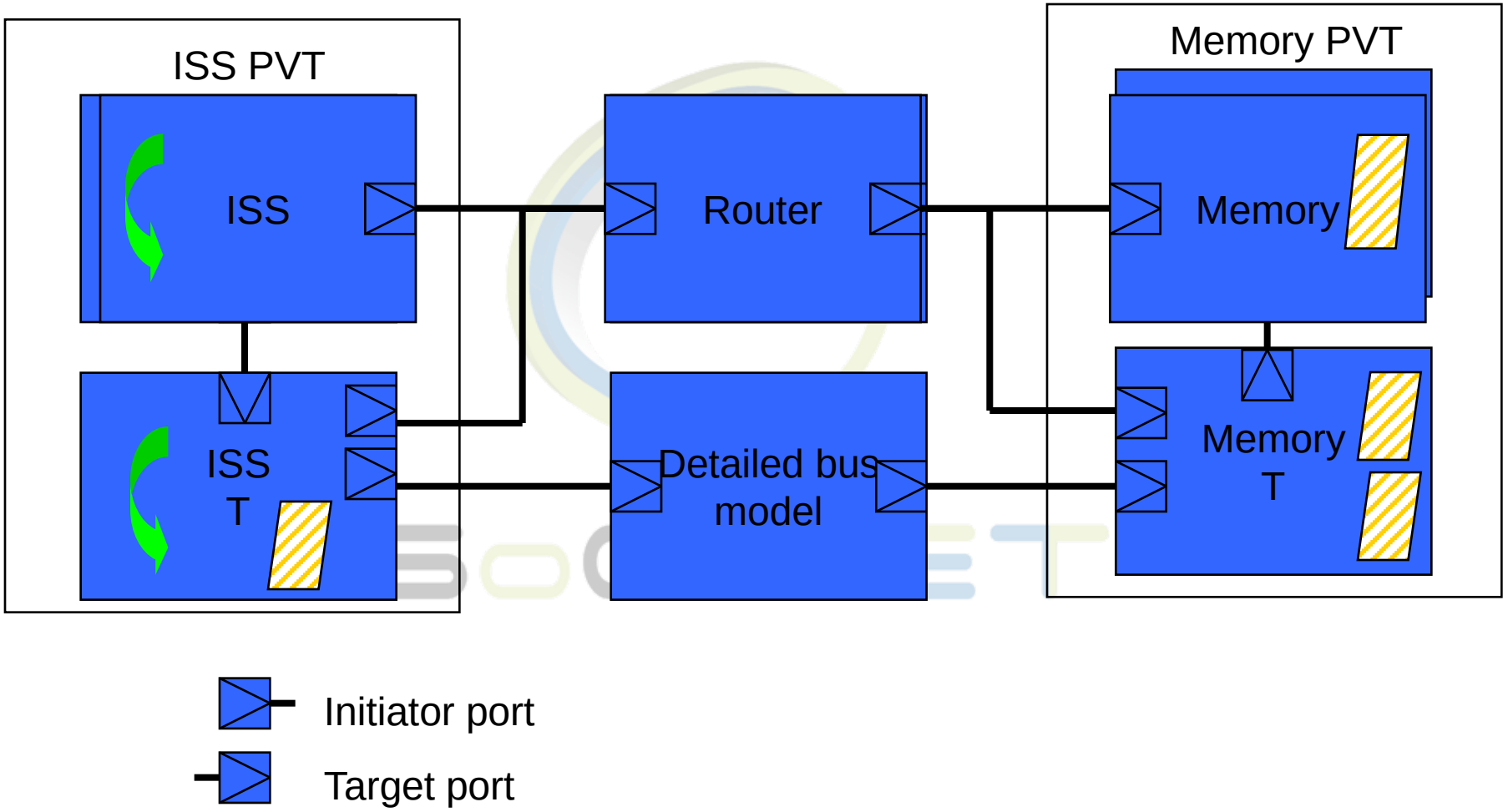
- Functional modifications cannot be verified without having to verify all timed aspects as well
- Modelling granularity is hard to modify once it has been set
- Modules cannot be easily reused for other platforms

Time & functionality separation

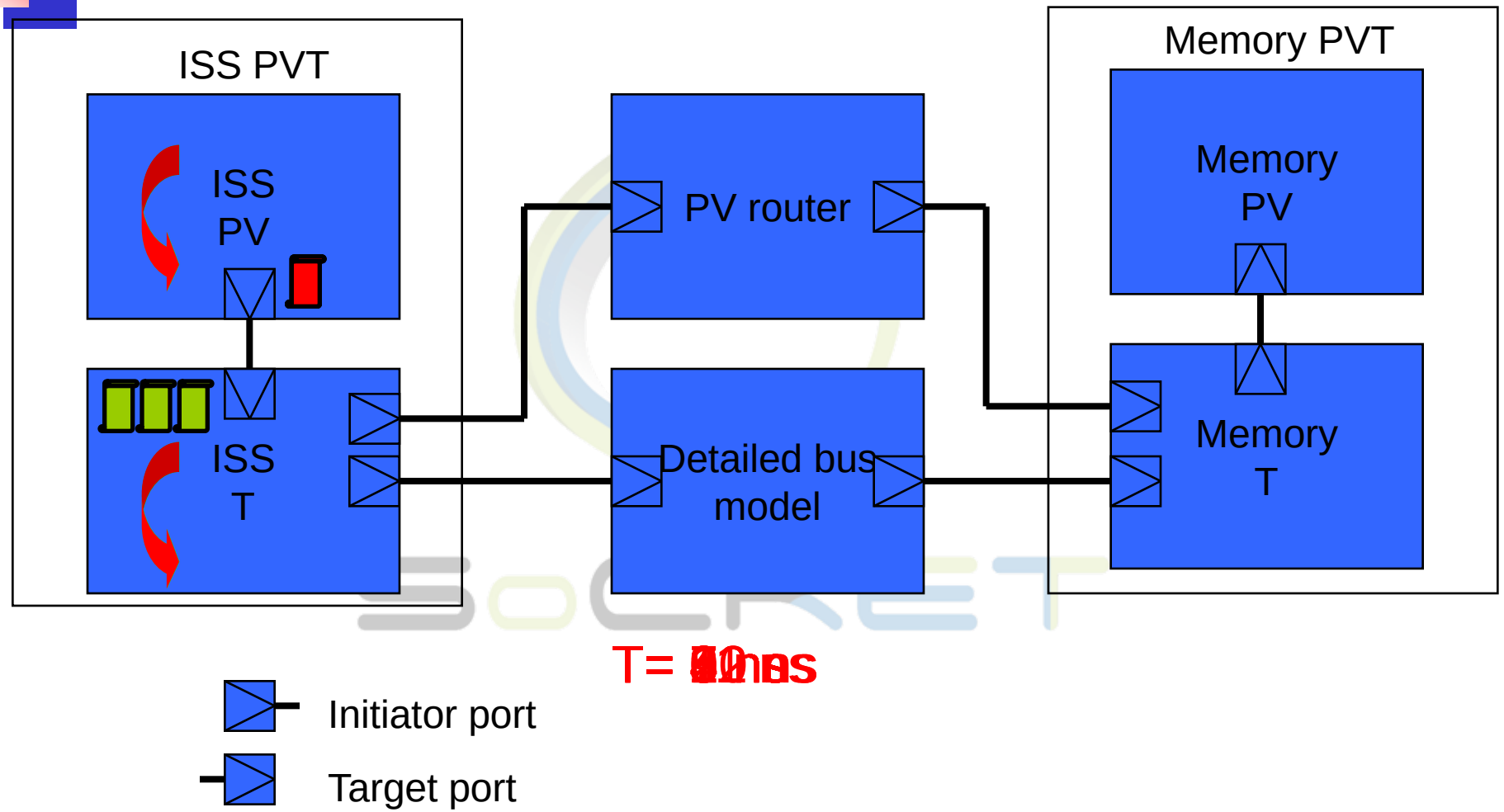
- Each model has a functional (PV) part and a timed part (T)
- Simulation switches between PV and T phases: when all PV models have reached a SSP
- T models record time relevant data (address, size of transfer...) during PV phases
- During T phases, T models “replay” the last PV phase and simulate time accordingly

Jérôme Cornet’s Thesis: “Separation of Functional and Non-Functional Aspects in Transactional Level Models of Systems-on-Chip” (Verimag 2008)

Enriching a functional platform



Functional simulation phase



Advantages and limitations

PV & T mixed

- Modelling is “natural”. Platforms are simple.
- Asynchronous events can be modelled easily

- Granularity is fixed
- Mixed debugging
- no control over simulation performance
- Reuse issue

PV & T separated

- Parallel development and debug of reusable PV and T models
- Granularity can be controlled easily (by changing T model)

- Modelling is more abstract. Platforms are complex
- Asynchronous events are harder to model

Agenda

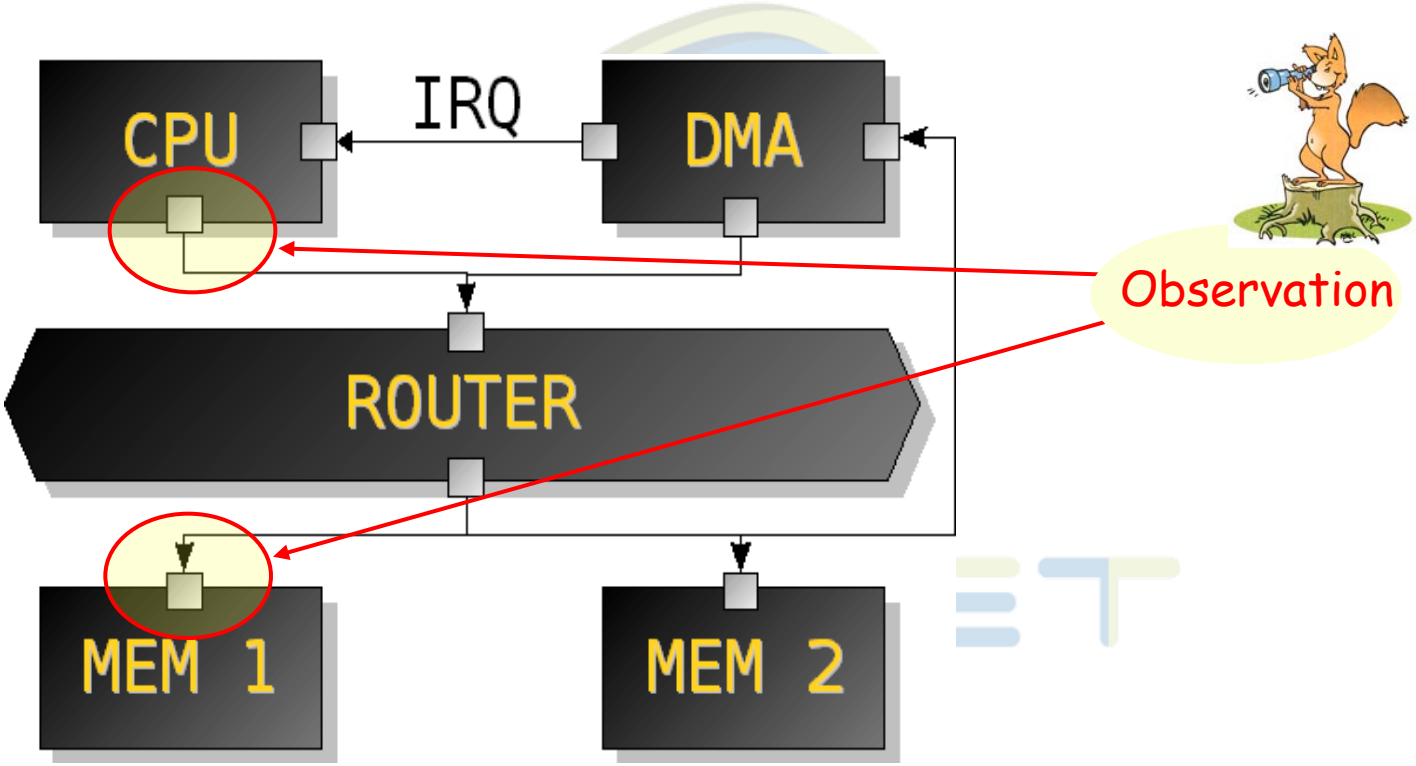
- SoCKET project presentation
- Proposed Hardware/Software Co-Design Flow
- Focus on key technologies + perspectives
  - SystemC/TLM Modelling
  - Assertion Based Verification
  - High Level Synthesis

What is ABV ?

- *Assertion*: statement about the *intended behaviour* or a *requirement* of the design
  - Temporal logics: CTL, LTL,...
  - Specification languages: SVA (IEEE Std 1800), PSL (IEEE Std 1850)
- *Assertion-Based Verification*: does the design obey these temporal assertions?
  - Static analysis (model checking)
  - Dynamic verification (during simulation)

ABV at system level

- Assertions at the system level:



Any time a source address *is transferred* to the DMA, a *read access* eventually occurs and the right address is used

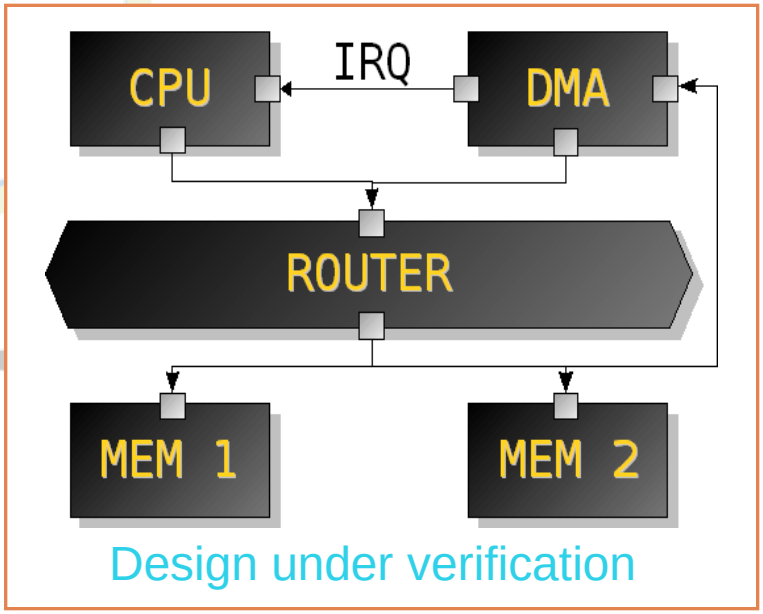
ISIS tool from TIMA Lab

PSL assertion

Automatic transformation

SystemC assertion monitor (checker)

Automatic instrumentation

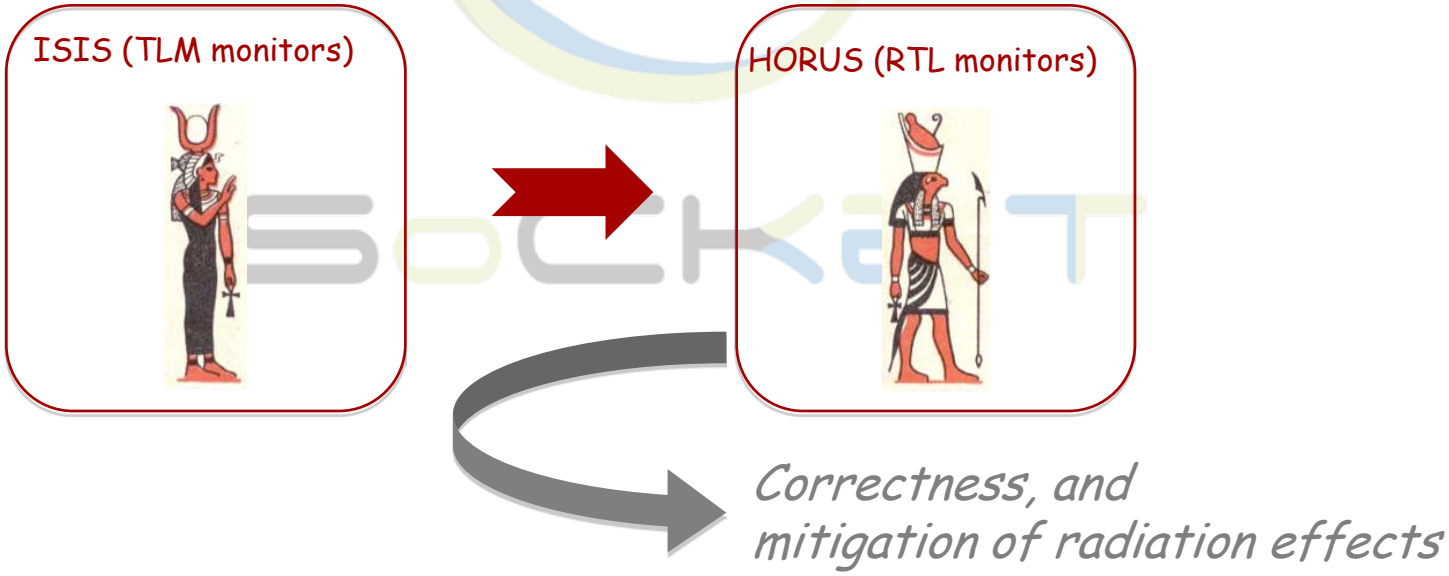


Astrium's REX

- Good expressivity of PSL
- Moderate time overhead induced by monitoring (5-8%)
- The property checkers will provide a valuable help for non-regression testing
- Careful (natural language) expression of the requirements
  - Specify at the TLM interfaces
  - Disambiguate the properties, in particular the meaning of communication actions, and specify your preferred observation points

ABV perspectives

- Among the future works
  - Perform other experiments and refine the definition rules for the expression of properties
  - Relation between PSL properties at the transactional level and at the RT level ?

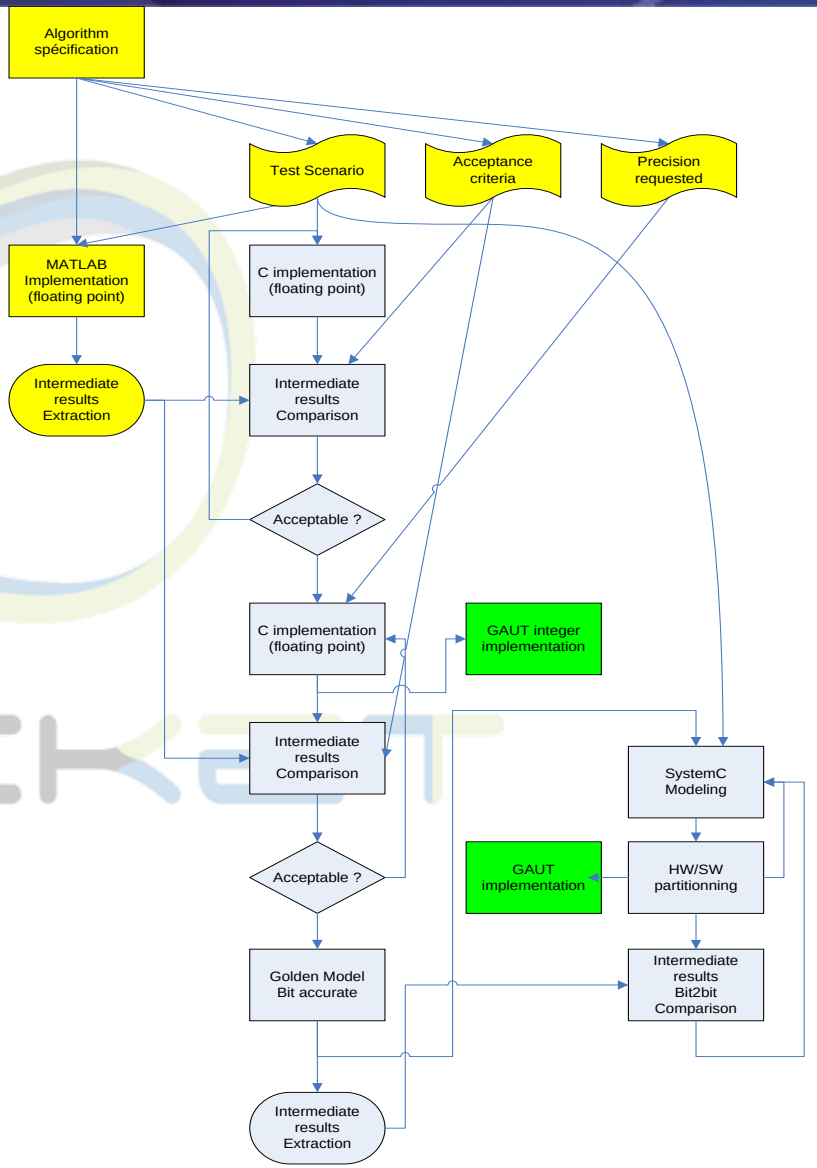


Agenda

- SoCKET project presentation
- Proposed Hardware/Software Co-Design Flow
- Focus on key technologies + perspectives
  - SystemC/TLM Modelling
  - Assertion Based Verification
  - High Level Synthesis

1st step: Algorithm -> HLS

- Validation
  - Intermediate results required
  - Validity criteria (computation precision)



HLS for architecture exploration

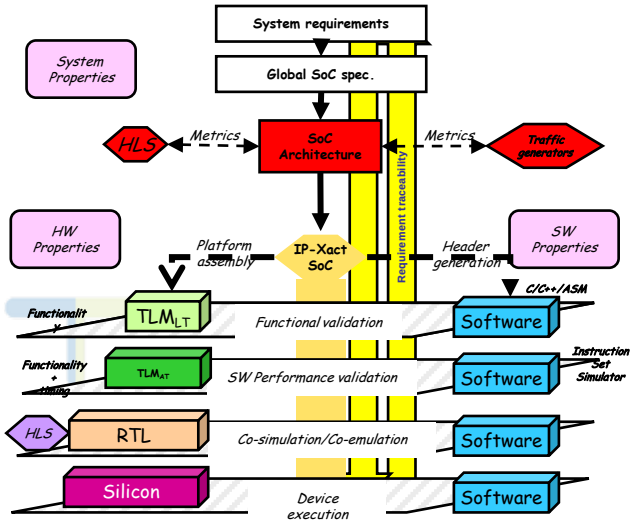
- Metrics (area, performance) – 20% pessimistic - Usable for for tradeoffs
- Help for bit accurate arithmetic migration (fixed point (ac_type/sc_type))
- HLS requires to consider any IO architecture bottlenecks
- HLS incremental refinement try/test loop: heuristic approach
- Allows to measure latency introduced by pipelining
- Separation of the processing process and the IO constraints

Tools evaluation report

- Celoxica HandleC
  - ->2007: GAIA project – metrics for DSE
  - Obsolete since 2009
- Mentor/Calypto CatapultC
  - 2005 evaluation: great tool, but too expensive
  - 2010->…: HLS bluebook (Michael Fingeroff)
- Mathworks HDL Coder
  - 2011: introduction in production process
- Université de Bretagne Sud Lab-STICC GAUT
  - SoCKET

High Level Synthesis: GAUT

- Academic tool
  - Public Domain (CECILL-B License)
  - Open source and free
- Dedicated to DSP applications
  - Data-dominated algorithm
- Inputs :
  - Algorithm written in bit-accurate C/C++
    - Bit-accurate integer and fixed-point from Mentor Graphics
  - Synthesis constraints (data average throughput, clock, I/O constraints...)
- Outputs :
  - RTL Architecture written in VHDL (IEEE 1076)
  - Simulation model in SystemC
- Automated Test-bench generation



GAUT REX

- Powerful academic tool for Data Flow Graph
- Good support of Xilinx targets
- Generated component directly pluggable on a bus (not used).
- ATC18RHA targeting on-going study
- Generated HDL efficiency (# gates and speed)
- Hierarchic synthesis support (not tested)
- Further evolution in the frame of project P
  - CDFG support
  - IO communication pattern instantiation

GAUT expected enhancement

- Pipelining needs to be externally handled
  - A valid signal is not propagated
- Control of loops unrolling
  - Automatic loop unrolling under constraint No manual override
- Output timing constraint propagation
- Add traceability between the C code and generated VHDL code
- Operators set extension (e.g: fixed-point division)
- Smart IO management (optimal data organization, ...)
- Would require some additional work for tool industrialization/qualification (documentation, validation of generated HDL, IO interface configuration/control, ...)

HLS highlights

- Ease IP maintenance/evolution.
- Requires both hardware competence and software skills.
- It's quite natural to transform Matlab to C then to RTL.
- Not optimal for data handling (FIFO, Cache, prefetch)
-  Manager : no speed up development process but the exploration process and avoid some dead-end

Global Conclusion

- ESL techniques very useful:
  - For DSE,
  - For removing ambiguities at HW/SW interfaces level,
  - For improving the fast prototyping phase
- Need of
  - Building a Space ESL ecosystem
    - Common Model Library and Toolbox
    - Common Model Coding Rules for model exchange
  - Can we avoid any dependence with commercial tools

SoCKET: Further information

- Public website: http://socket.imag.fr

- Public “final” workshop:
  - 23 & 24 November 2011 at Toulouse, France
  - 1st day: Tutorial on pillar technologies & Results presentation
  - 2nd day: Industrial Return of Experience

  YOU ARE ALL WELCOMED

Thank you for your attention

Any questions ?

Backup

Technical pillars (1)

- High level synthesis (Lab_STICC)
  - Under time constraint
  - Under resources constraint
- Heterogeneous simulation techniques (STM)
  - SystemC/TLM
  - LT/AT/CABA abstraction levels
- IPs encapsulation and interoperability (MDS, STM)
  - SPIRIT/IP-XACT (structural information)
    => configuration/documentation/deployment
  - OCP-IP

Technical pillars (2)

- Validation techniques
  - Formal and semi-formal methods (TIMA, IRIT)
    - Formal verification by model-checking
    - Semi-formal verification by automatic generation of monitors
  - Mutation analysis techniques (STM)
    - Atomic modification of the design to check if the validation environment detect it
    - Today, at RTL level, SoCKET will try to extend it to system level
  - Test cases automatic generation (STM)
    - E language OK for IPs, performance issues for SoCs
    - Evaluation of techniques based on IP-XACT description and C generators