System-Level Design using SystemC

Miltos Grammatikakis

TEI of Crete

(mdgramma@cs.teicrete.gr)

System-on-Chip

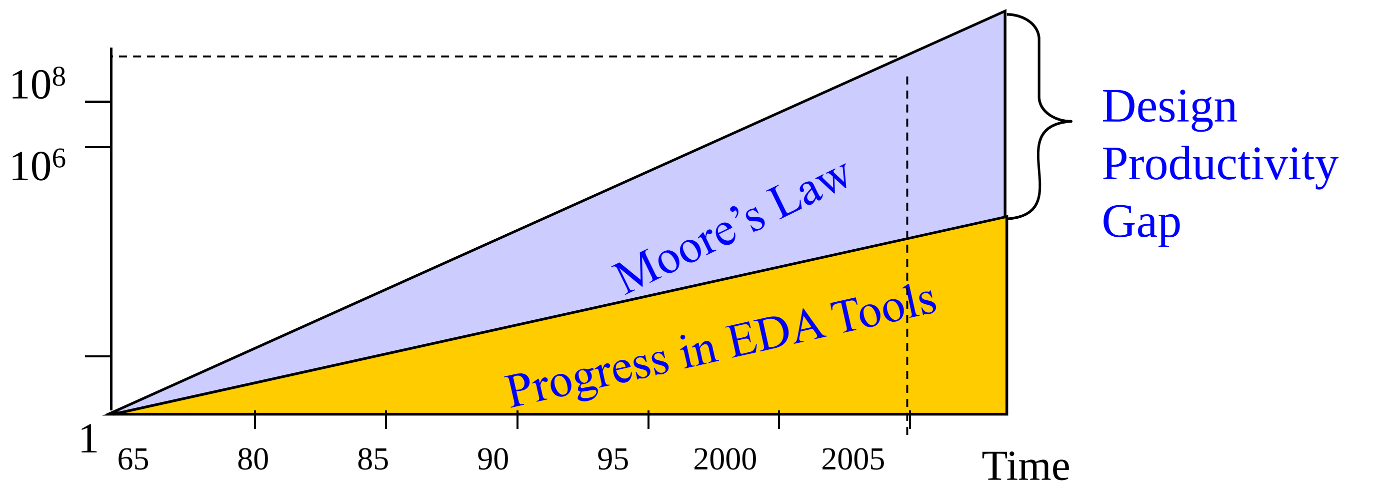
Due to steady downscaling in CMOS device dimensions, complex systems, called *System-on-Chip* (*S*oC) contain multimillions or billions of transistors. SoC will consist of various interconnected components, including

- embedded DRAM, FLASH, FPGA, and application-specific IP, e.g. Ethernet or MPEG,
- programmable components, such as general purpose processor cores, digital signal processor (DSP) cores and VLIW cores, and
- analog front-end, peripheral I/O devices and optical MEMS.

Devices based on SoC, called *embedded systems*, are now common in

- consumer electronics, e.g. set top boxes,
- telecommunications, e.g. VDSL, and
- automotive technology, and robotics/plant control.

Design Productivity Gap



- Moore's Law: Capacity of integrated chips doubles every 18-20mo
- Solutions
  - Top-down (platform-based) vs. bottom-up (component-based) design
  - IP reuse – standardization
  - SoC modeling at higher abstraction level
  - High-level synthesis
  - Model-driven design engineering, e.g. from UML

Electronic System Level Design for SoC

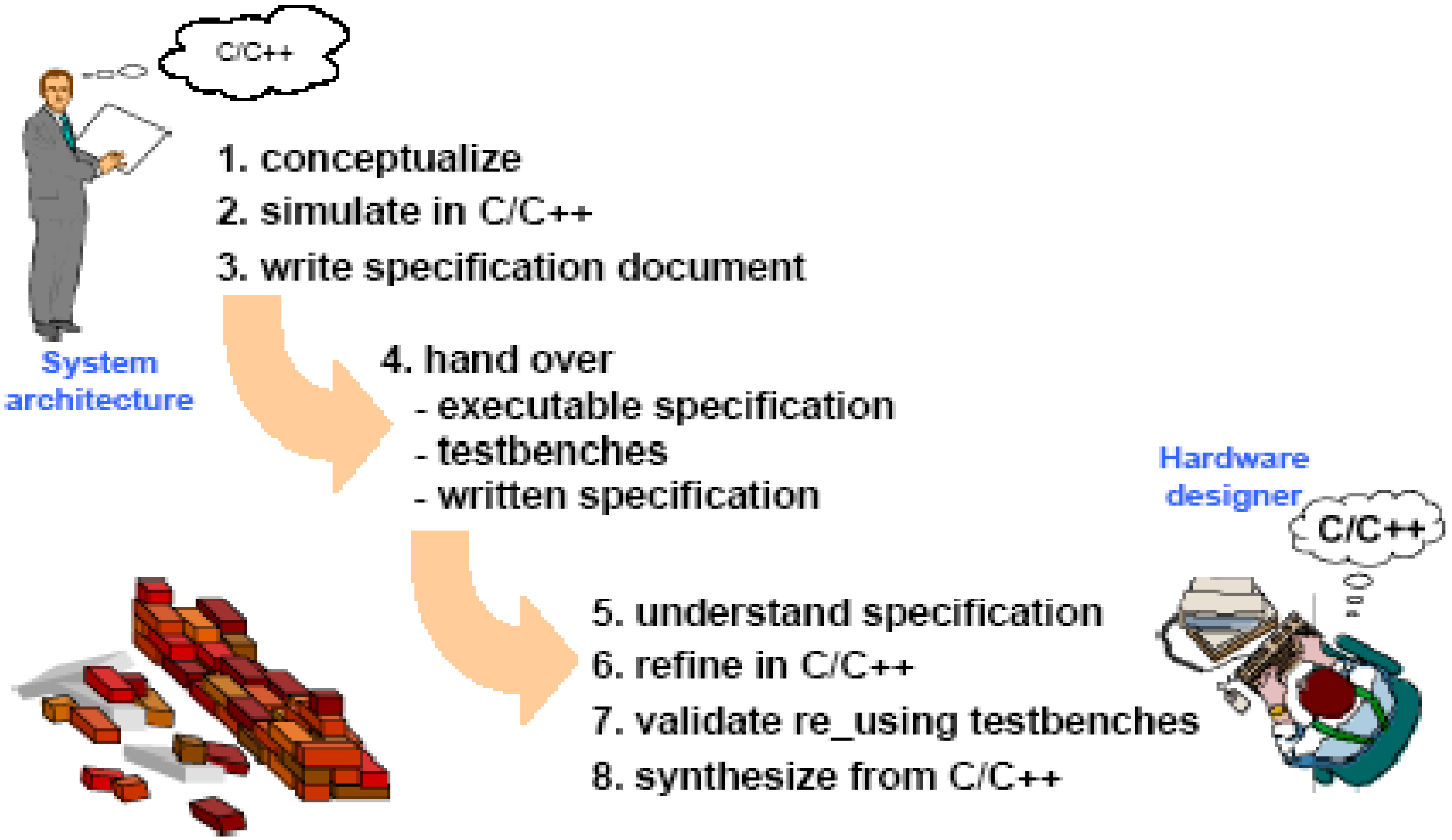
The escalating gate count, desired system heterogeneity and trend towards increased productivity for complex SoC devices challenges traditional RTL-to-gates design methodology based on VHDL or VERILOG simulation.

Thus, Electronic System Level design methodology (ESL) focuses on understanding the functionality and relationships of the primary system components, separating system design from implementation.

Low level implementation issues greatly increase the number of parameters and constraints in the design space, thus, extremely complicating optimal design selection and verification. Similar to near-optimal combinatorial algorithms, e.g. travelling salesman heuristics, ESL models prune away poor design choices and focus on closely examining feasible options.

System-Level Design

- C/C++ based design methodology

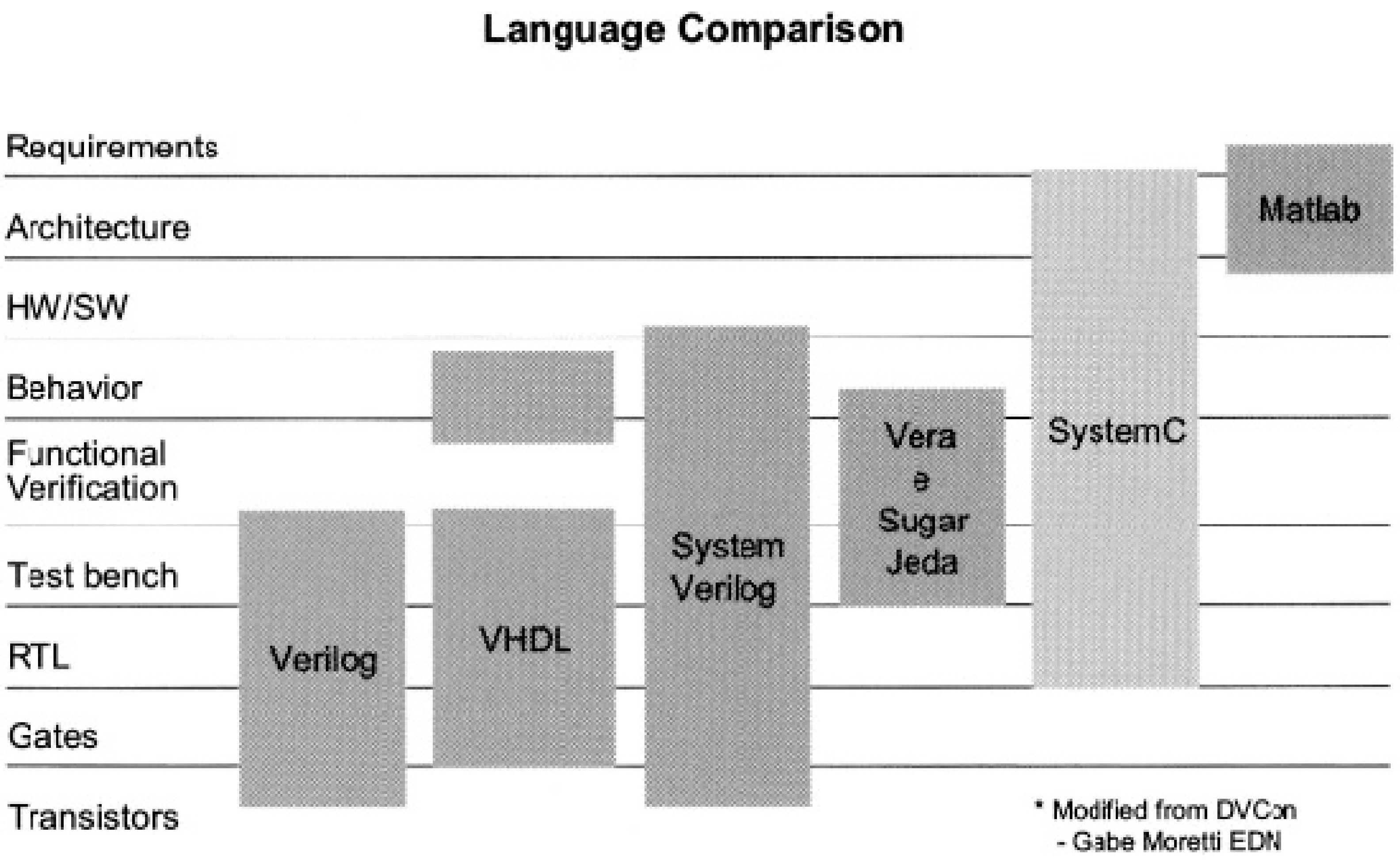


1. conceptualize
2. simulate in C/C++
3. write specification document

4. hand over
   - executable specification
   - testbenches
   - written specification

5. understand specification
6. refine in C/C++
7. validate re_using testbenches
8. synthesize from C/C++

Design Languages: Overview



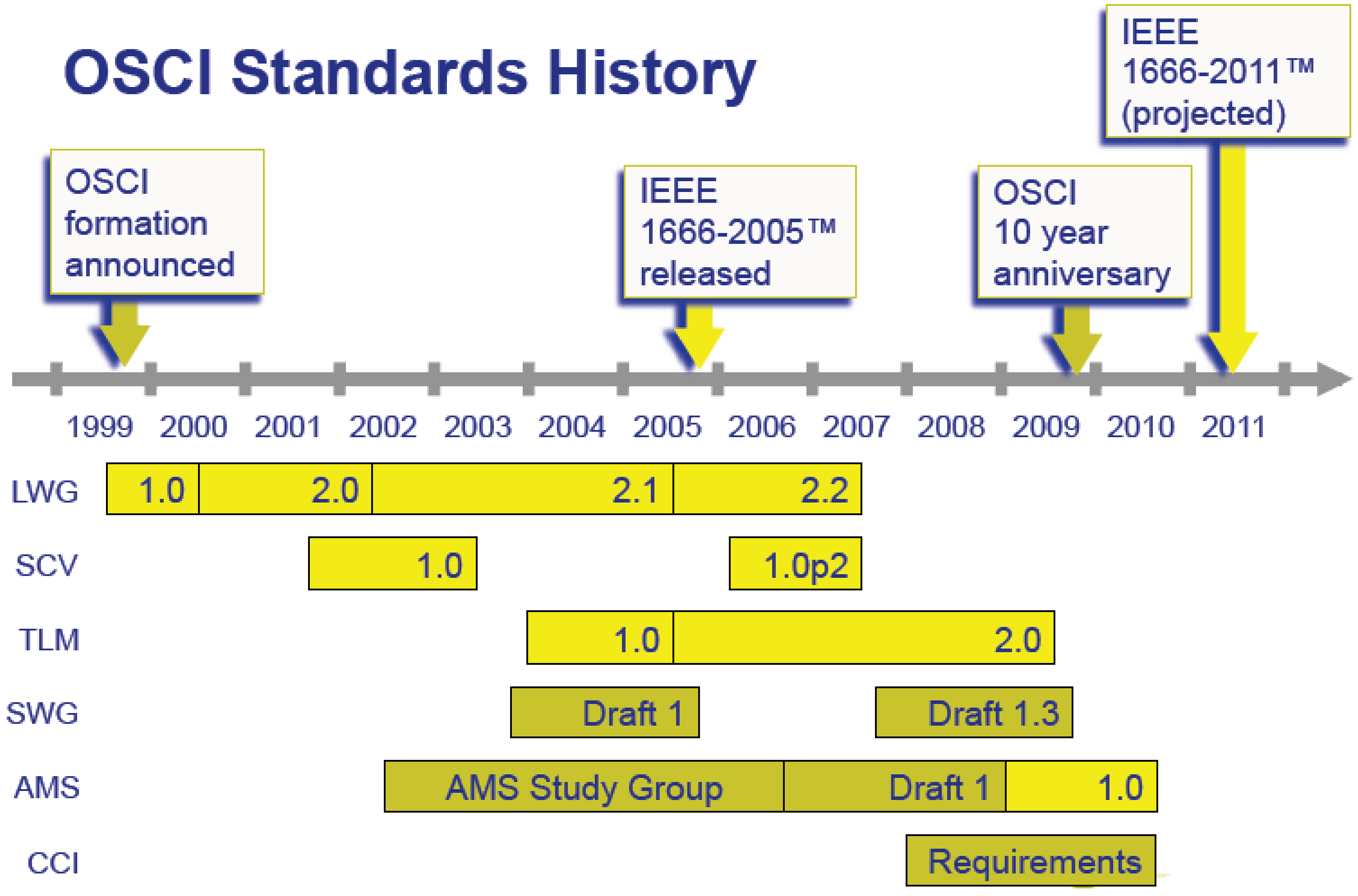
SystemC

•www.systemc.org

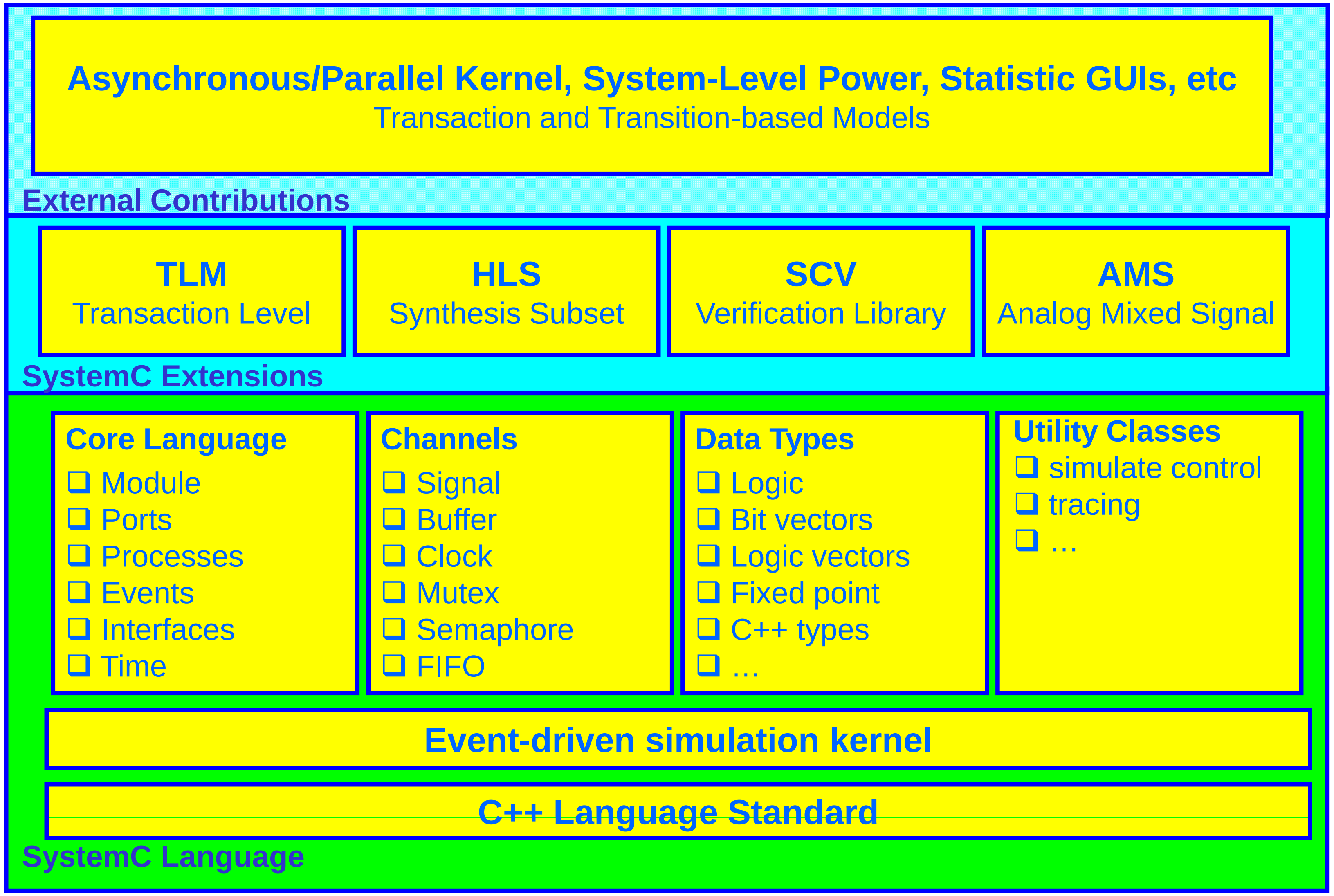
- SystemC is a leading open industry-standard modeling language based on a C++ library targeting architectural, algorithmic, and transaction-level modeling, although it can also be used for RTL.
- SystemC acceptance is driven by increasing design complexity, pushing towards system-level hw/sw modeling and simulation at higher level of abstractions,
  - enabling effective early design space exploration and co-design for software performance studies and dynamic (concept) validation,
  - reducing development costs, and
  - improving design quality.
- Steering Committee: Open SystemC Initiative (OSCI)

  Technical Working Groups
  - Synthesis (SWG)
  - Language (LWG)
  - Transaction-Level Modeling (TLM-WG)
  - Analog/Mixed Signal (AMS-WG)
  - Configuration, Control and Inspection (CCI-WG)
- User Groups in Europe, North America, Latin America, Japan, Taiwan
- IEEE 1666-2005 SystemC Language Reference Manual (LRM)

OSCI Standards



SystemC Layers & Extensions



Refinement and Transaction-Level Modeling (TLM)

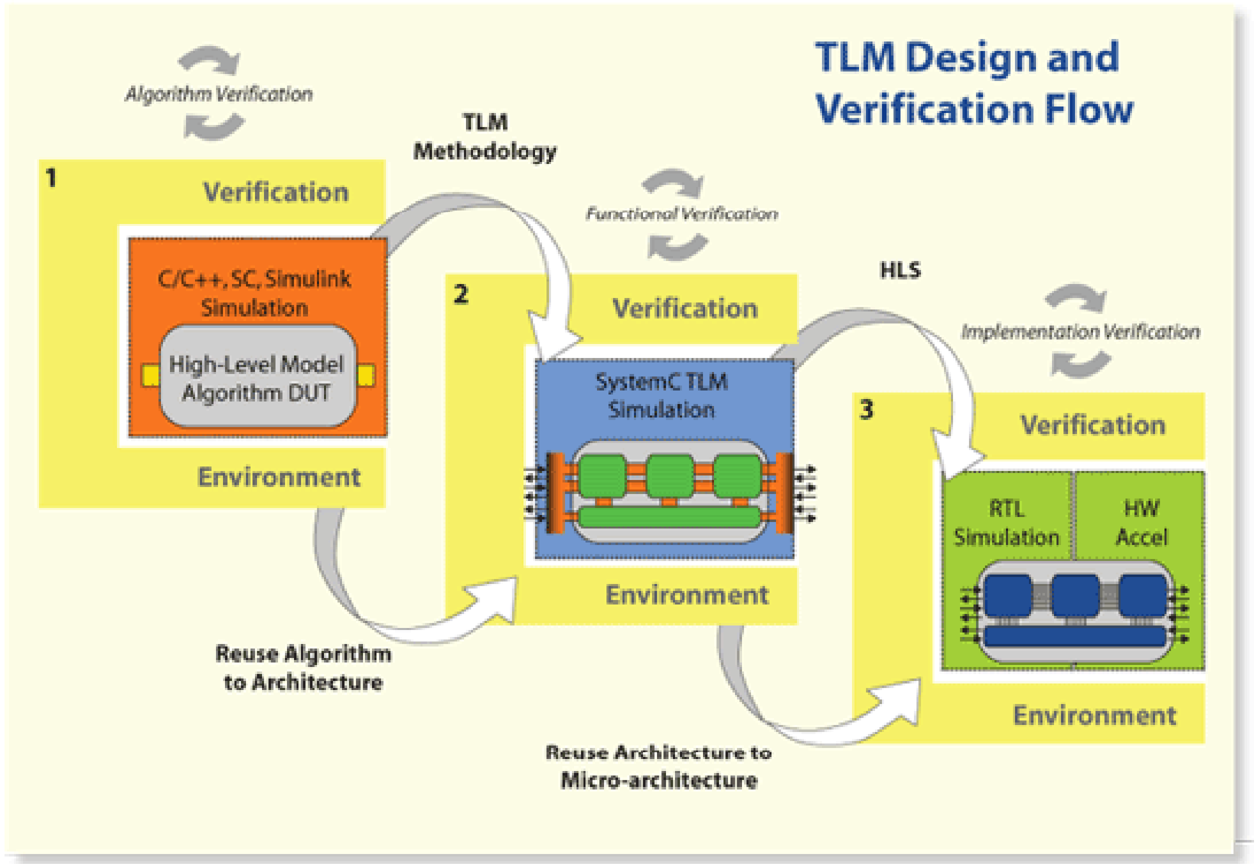
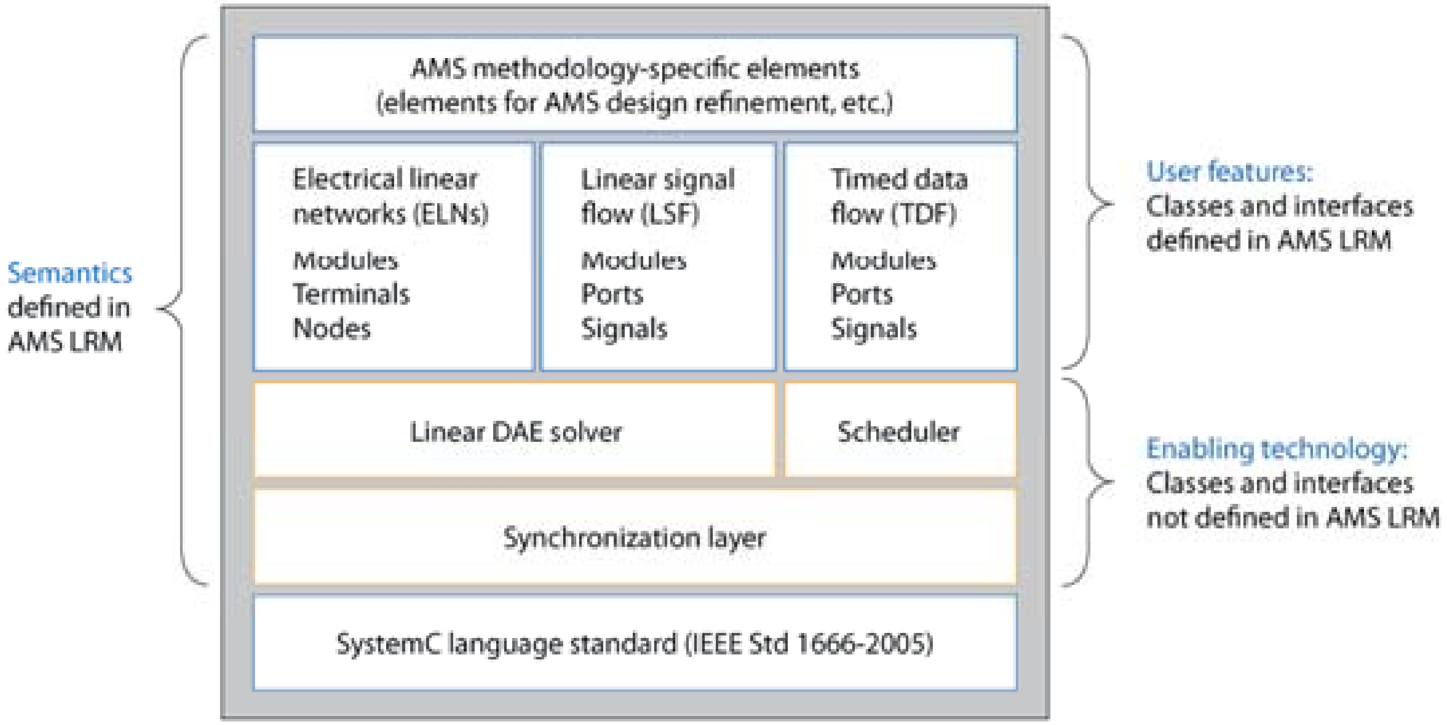

TLM Design and Verification Flow
Algorithm Verification
TLM Methodology
1
Verification
C/C++, SC, Simulink Simulation
High-Level Model Algorithm DUT
Environment
Functional Verification
2
Verification
SystemC TLM Simulation
Environment
HLS
Implementation Verification
3
Verification
RTL Simulation
HW Accel
Environment
Reuse Algorithm to Architecture
Reuse Architecture to Micro-architecture

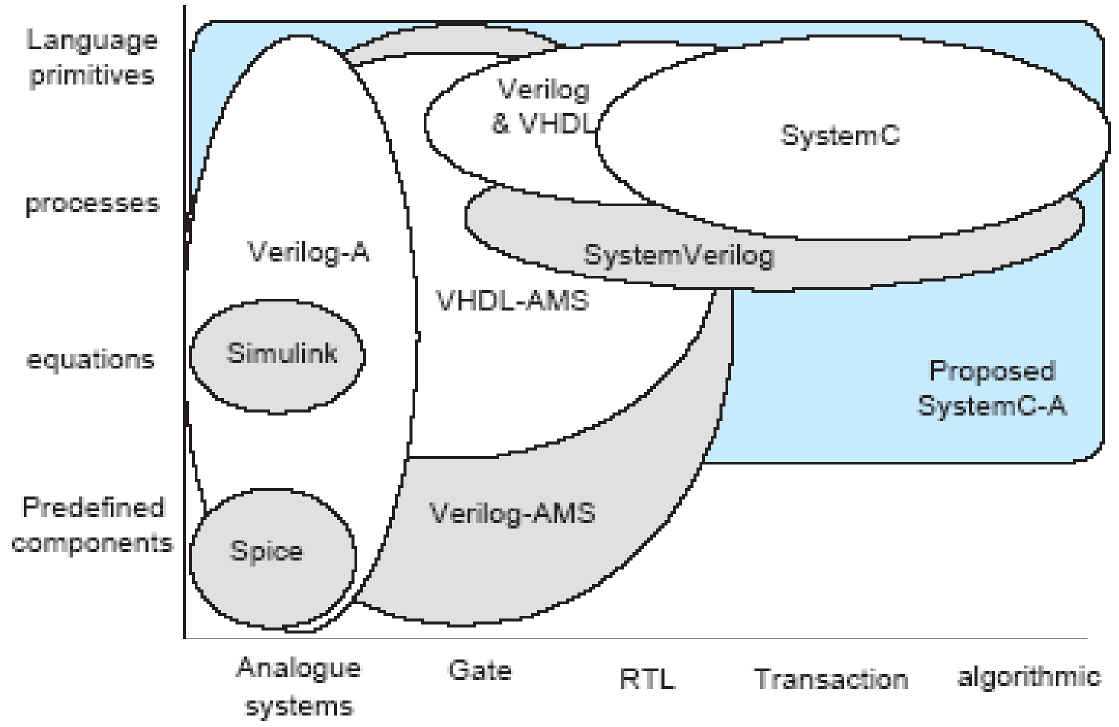
Figure 4. TLM design and verification flow.

SystemC AMS Extension


AMS methodology-specific elements (elements for AMS design refinement, etc.)
Electrical linear networks (ELNs) Modules Terminals Nodes
Linear signal flow (LSF) Modules Ports Signals
Timed data flow (TDF) Modules Ports Signals
Linear DAE solver
Scheduler
Synchronization layer
SystemC language standard (IEEE Std 1666-2005)
Semantics defined in AMS LRM
User features: Classes and interfaces defined in AMS LRM
Enabling technology: Classes and interfaces not defined in AMS LRM

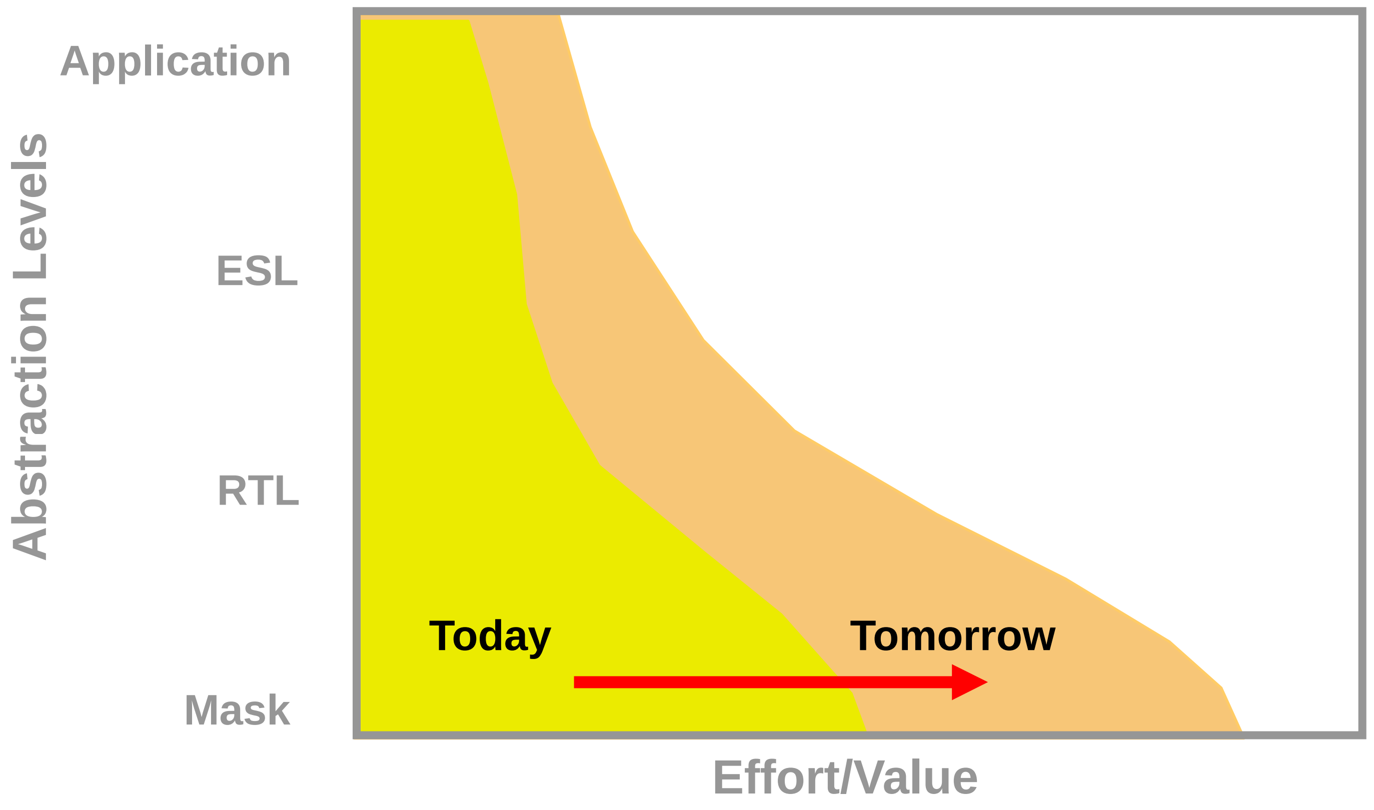
- In 2009, OSCI developed a User Guide and LRM for the SystemC-AMS 1.0 standard.
- It defines the execution semantics and extensions to SystemC constructs (classes, interfaces, analog kernel and modeling of continuous-time analysis) for IP modeling of embedded analog/mixed signal systems at system-level, analog behavioral and netlist level.

Comparative Map of AMS Design Languages



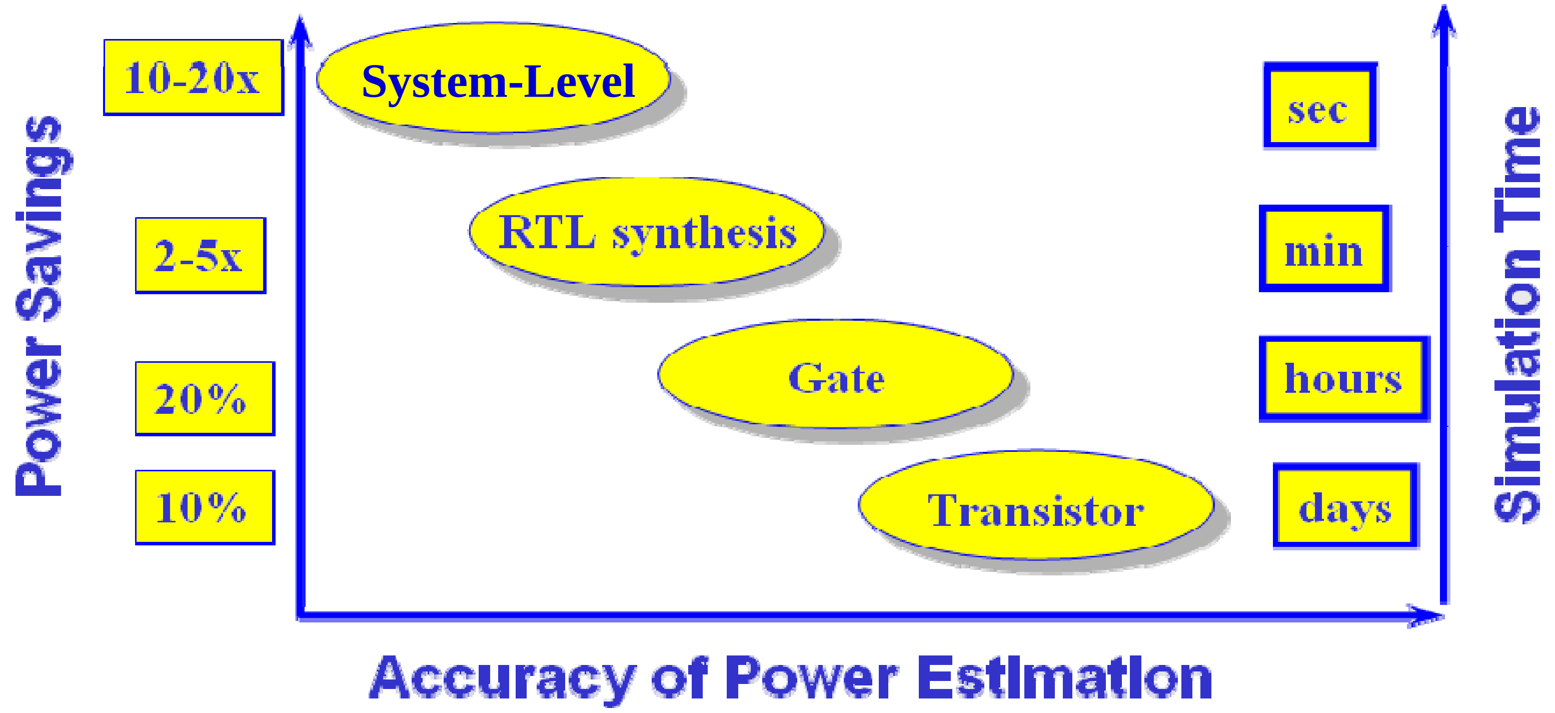
- SystemC-AMS is based on C/C++ language. It complements circuit level simulation for critical analog mixed signal or RF circuitry (e.g. using Spice), as well as behavioral modeling, such as Verilog-AMS or VHDL-AMS.
- Although not currently approved by the SystemC initiative (OSCI), an implementation is currently provided by Fraunhofer Institut for Integrated Circuits.

Ideas for New System-Level Tools



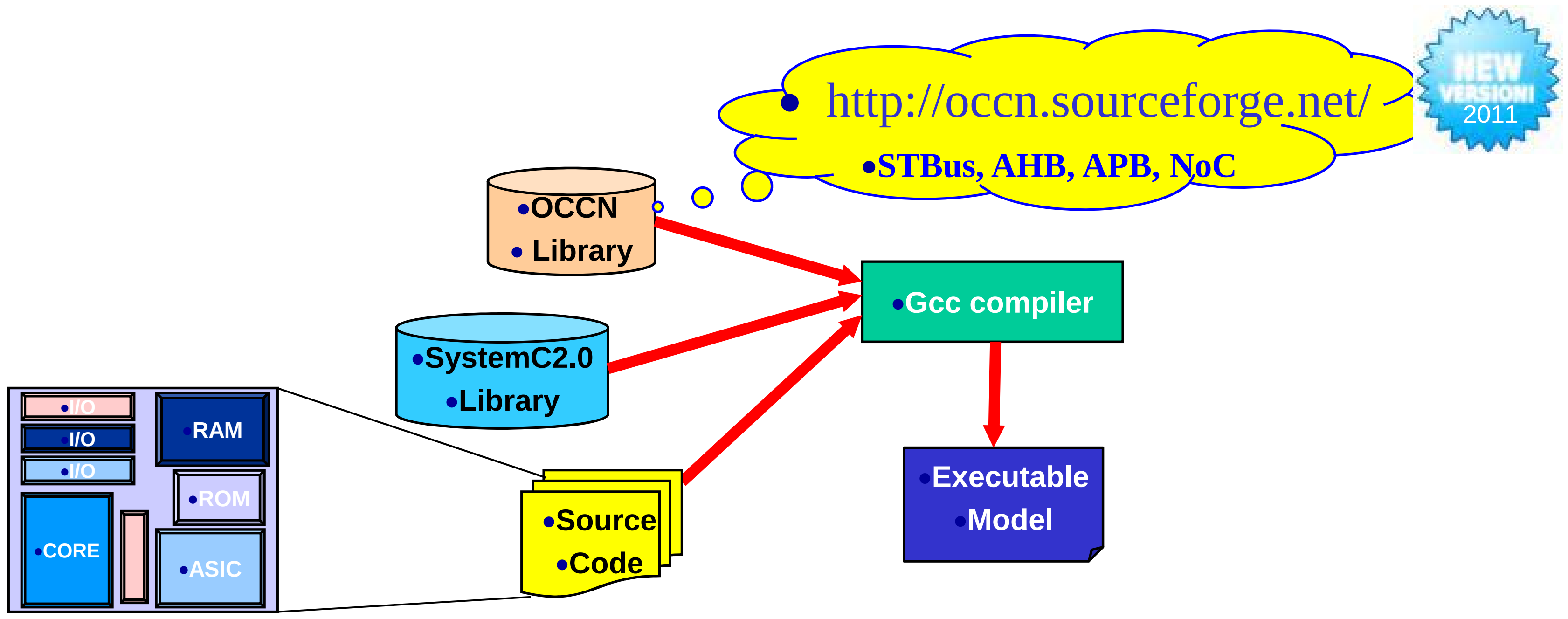
- With the fast pace of DSM technology advances, system-level tools and methodologies are interesting.

System-Level Power Estimation



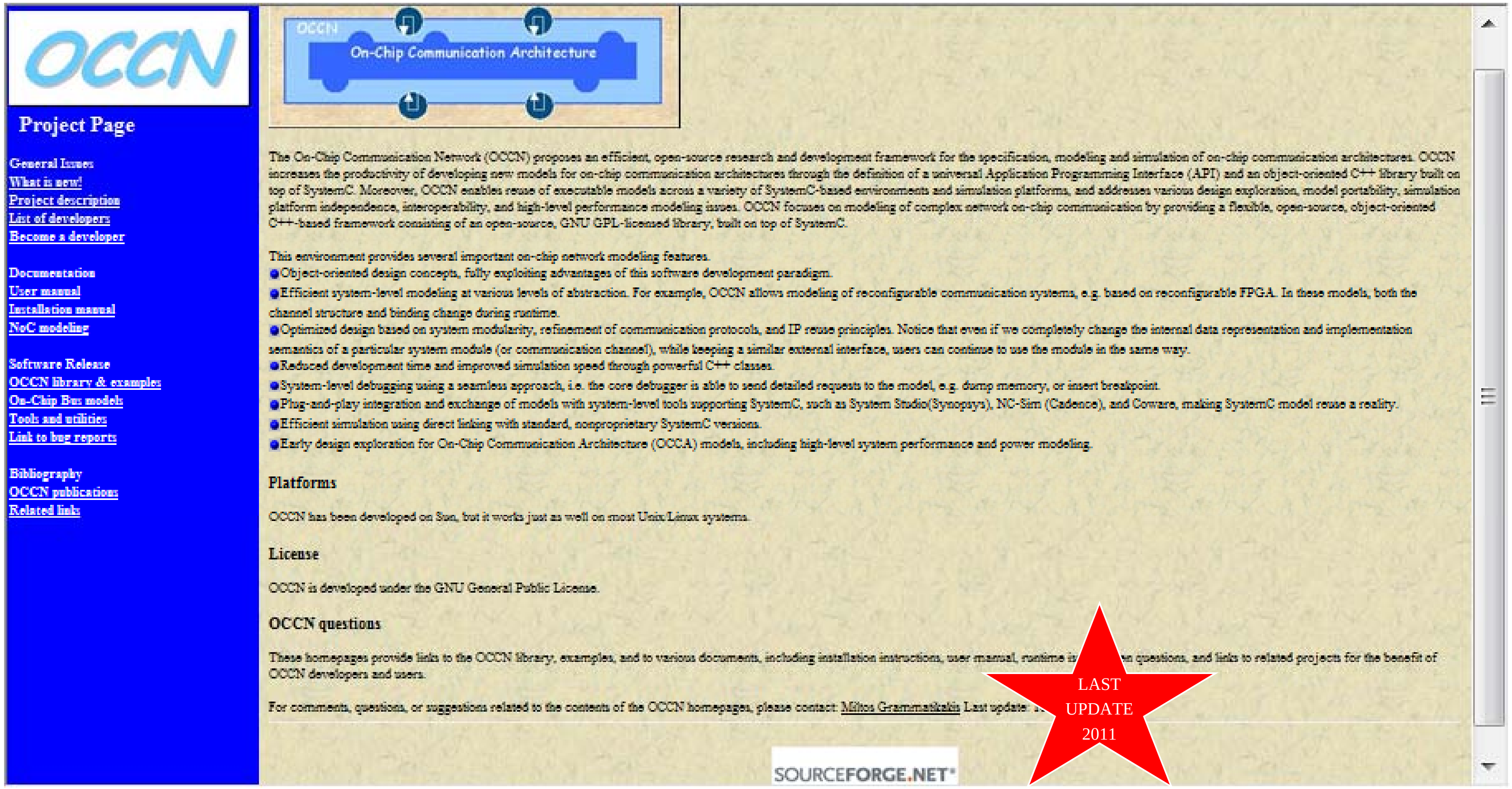
- System-level power estimation provides sufficient accuracy at improved cost, performance and productivity compared to RTL flow.
- Low-level power useful only for late design optimization.

OCCN Methodology for NoC Modeling (2003-)



- OCCN (On-Chip Communication Network) is an open-source communication modeling environment, with a SystemC-based library, a runtime test environment, and communication refinement extending general SoC design methodology.
- OCCN abstraction levels range from functional to transactional clock-cycle, bit-accurate. RTL level is supported by SystemC behavioral synthesis tools, e.g. Synopsys Co-Centric System Studio, or Cadence VCC.

http://occn.sourceforge.net



OCCN

Project Page

General Issues
What is new!
Project description
List of developers
Become a developer

Documentation
User manual
Installation manual
NoC modeling

Software Release
OCCN library & examples
On-Chip Bus models
Tools and utilities
Link to bug reports

Bibliography
OCCN publications
Related links

The On-Chip Communication Network (OCCN) proposes an efficient, open-source research and development framework for the specification, modeling and simulation of on-chip communication architectures. OCCN increases the productivity of developing new models for on-chip communication architectures through the definition of a universal Application Programming Interface (API) and an object-oriented C++ library built on top of SystemC. Moreover, OCCN enables reuse of executable models across a variety of SystemC-based environments and simulation platforms, and addresses various design exploration, model portability, simulation platform independence, interoperability, and high-level performance modeling issues. OCCN focuses on modeling of complex network on-chip communication by providing a flexible, open-source, object-oriented C++-based framework consisting of an open-source, GNU GPL-licensed library, built on top of SystemC.

This environment provides several important on-chip network modeling features.

- Object-oriented design concepts, fully exploiting advantages of this software development paradigm.
- Efficient system-level modeling at various levels of abstraction. For example, OCCN allows modeling of reconfigurable communication systems, e.g. based on reconfigurable FPGA. In these models, both the channel structure and binding change during runtime.
- Optimized design based on system modularity, refinement of communication protocols, and IP reuse principles. Notice that even if we completely change the internal data representation and implementation semantics of a particular system module (or communication channel), while keeping a similar external interface, users can continue to use the module in the same way.
- Reduced development time and improved simulation speed through powerful C++ classes.
- System-level debugging using a seamless approach, i.e. the core debugger is able to send detailed requests to the model, e.g. dump memory, or insert breakpoint.
- Plug-and-play integration and exchange of models with system-level tools supporting SystemC, such as System Studio(Synopsys), NC-Sim (Cadence), and Coware, making SystemC model reuse a reality.
- Efficient simulation using direct linking with standard, nonproprietary SystemC versions.
- Early design exploration for On-Chip Communication Architecture (OCCA) models, including high-level system performance and power modeling.

Platforms

OCCN has been developed on Sun, but it works just as well on most Unix/Linux systems.

License

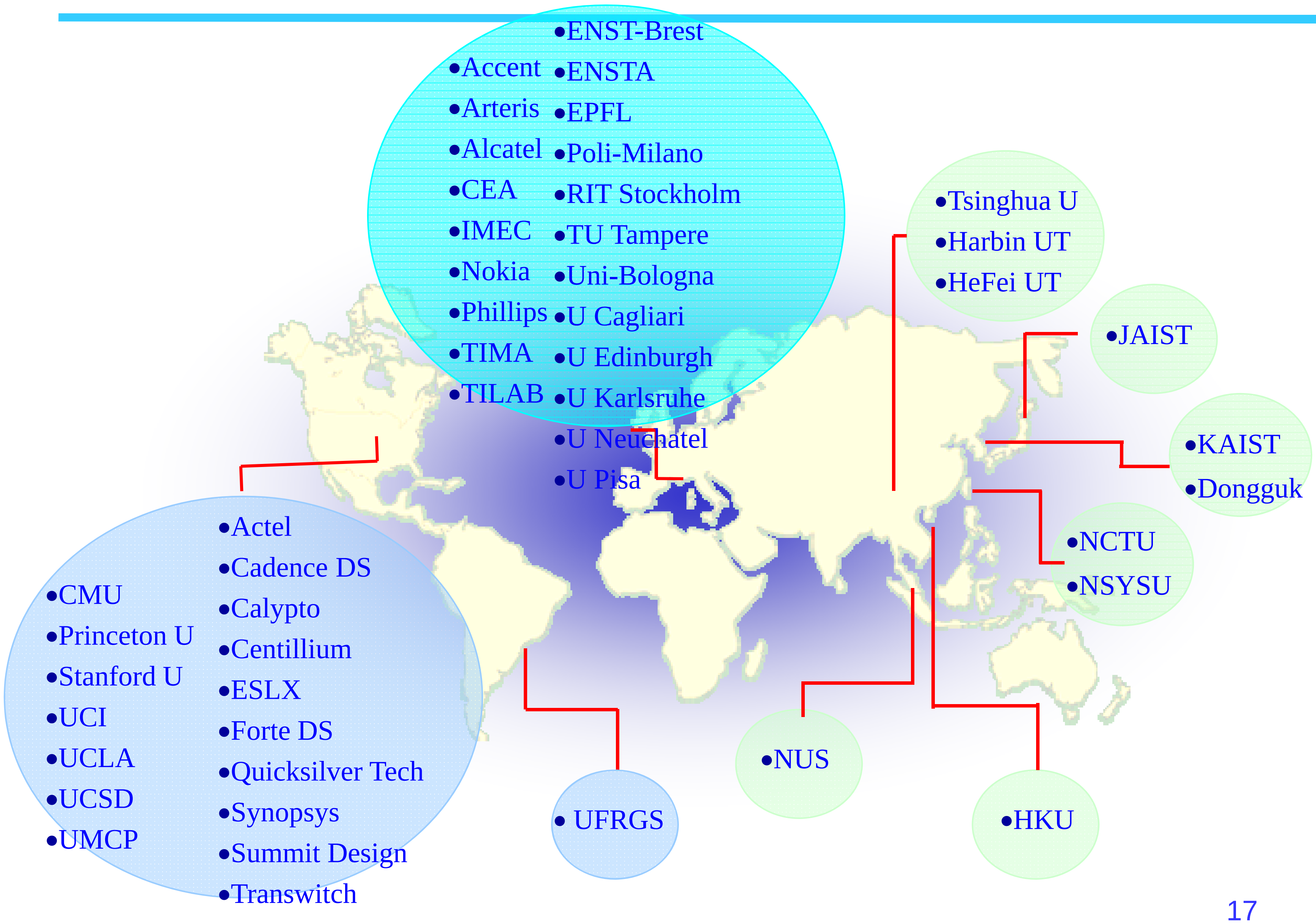
OCCN is developed under the GNU General Public License.

OCCN questions

These homepages provide links to the OCCN library, examples, and to various documents, including installation instructions, user manual, runtime is en questions, and links to related projects for the benefit of OCCN developers and users.

For comments, questions, or suggestions related to the contents of the OCCN homepages, please contact: Miltos Grammatikakis Last update:

OCCN Users (from non-obligatory register list)



Contribution to Open Software

- Open source sites provide free core software, data models, algorithms and tools, enhancing “coopetition” (collaboration among competitors) for increased productivity and quality via increased manpower and broadened expertise.

Conclusions

- In today’s mosaic designs derived by each design language’s unique strengths, weaknesses, advantages and disadvantages (and often strong user preferences), interoperability of design languages and tools is crucial.
- The overlap between design languages makes it easier to mix models written in each language within the same simulation, and also adapt language infrastructures, e.g. verification data types.
- SystemC has promoted an open model creation environment, including interoperability and data flow.
- The business incentives aren't there to motivate an open source EDA environment. Major tools and models – but not the environment –continue to develop on a proprietary basis.

Links to SystemC

- SystemC Community: white papers, user manual, library, examples, regression tests…
  - http://www.systemc.org
- IEEE 1666 Standard: language/simulator semantics, usage details)
  http://standards.ieee.org/getieee/1666/download/1666-2005.pdf
- European/North America/Latin America/Japan/Taiwan SystemC Users Group
  - http://www-ti.informatik.uni-tuebingen.de/~systemc/systemc.html
- Wikipedia.org
  - http://en.wikipedia.org/wiki/Systemc
- SystemC Tools
  - http://www.asic-world.com/systemc/tools.html
- ASIC-World
  - http://www.asic-world.com/systemc

Links to Open Source EDA Tools

- Open Source EDA
  - http://www.opencollector.org/summary.php
- Comparison of EDA Software (Open & Proprietary)
  - http://en.wikipedia.org/wiki/Comparison_of_EDA_Software
- Opencores
  - http://www.opencores.org
- GNU EDA
  - http://www.gpleda.org/index.html

End of Presentation

Thank You