Lecture 2: System-Level Design Methodology

Miltos Grammatikakis

System-on-Chip

Due to steady downscaling in CMOS device dimensions, it is expected that by 2005 complex systems, called *System-on-Chip (*SoC) will contain hundreds of million transistors. SoC will consist of various interconnected components, including

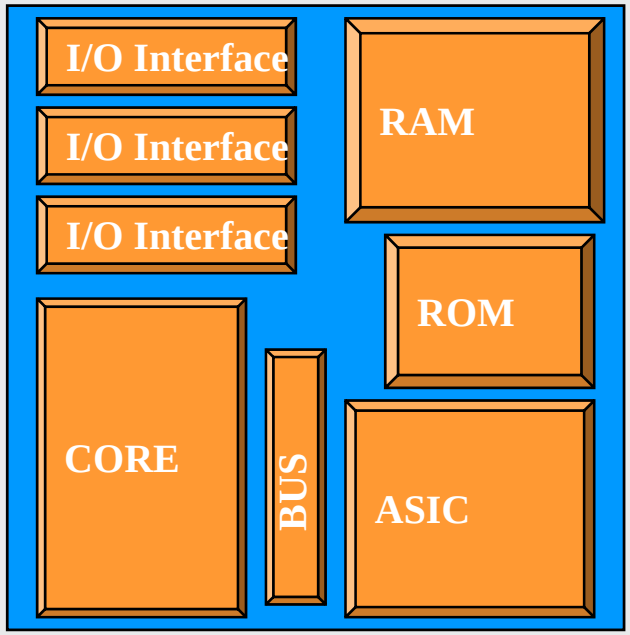
- embedded DRAM, FLASH, FPGA, and application-specific IP, e.g. Ethernet or MPEG,
- programmable components, such as general purpose processor cores, digital signal processor (DSP) cores and VLIW cores, and
- analog front-end, peripheral I/O devices and optical MEMS.

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.

Embedded Systems Architecture





- They consist of
  - ✓ on-chip bus (or network)
  - ✓ general purpose processors
  - ✓ DSP and application-specific accelerators, e.g. mpeg, Ethernet
  - ✓ RAM, ROM
  - ✓ peripheral IPs
- "It is not the strongest (ASIC) of the species that survive, nor the most intelligent (CPU), but the one most responsive to selected change (RECONFIGURABLE)“

Charles Darwin

Embedded Systems: Challenges

- Physical processes: sensors, actuators
- Heterogeneous SoC with AMS and/or digital blocks
- HW/SW partitioning
- Networking & Processing
  - ✓ usually shared memory
  - ✓ distributed processing at the speed of the environment (real-time)
  - ✓ adaptive power-aware system

Semiconductor Industry Market

	1980	1990	2000	2010
1	TI	NEC	Intel	Intel
2	Motorola	Toshiba	Toshiba	Samsung
3	Philips	Motorola	NEC	Toshiba
4	NEC	Hitachi	Samsung	TI
5	National	Intel	TI	Renesas
6	Toshiba	Fujitsu	ST	Hynix
7	Hitachi	TI	Motorola	ST
8	Intel	Mitsubishi	Hitachi	Micron
9	Fairchild	Philips	Infineon	Qualcomm
10	Siemens	Matsushita	Micron	Elpida

Semiconductor Industry Market

1H 2010 Semiconductor Leaders – ARM processors have won the war

1H10 Top 20 Semiconductor Sales Leaders ($M)

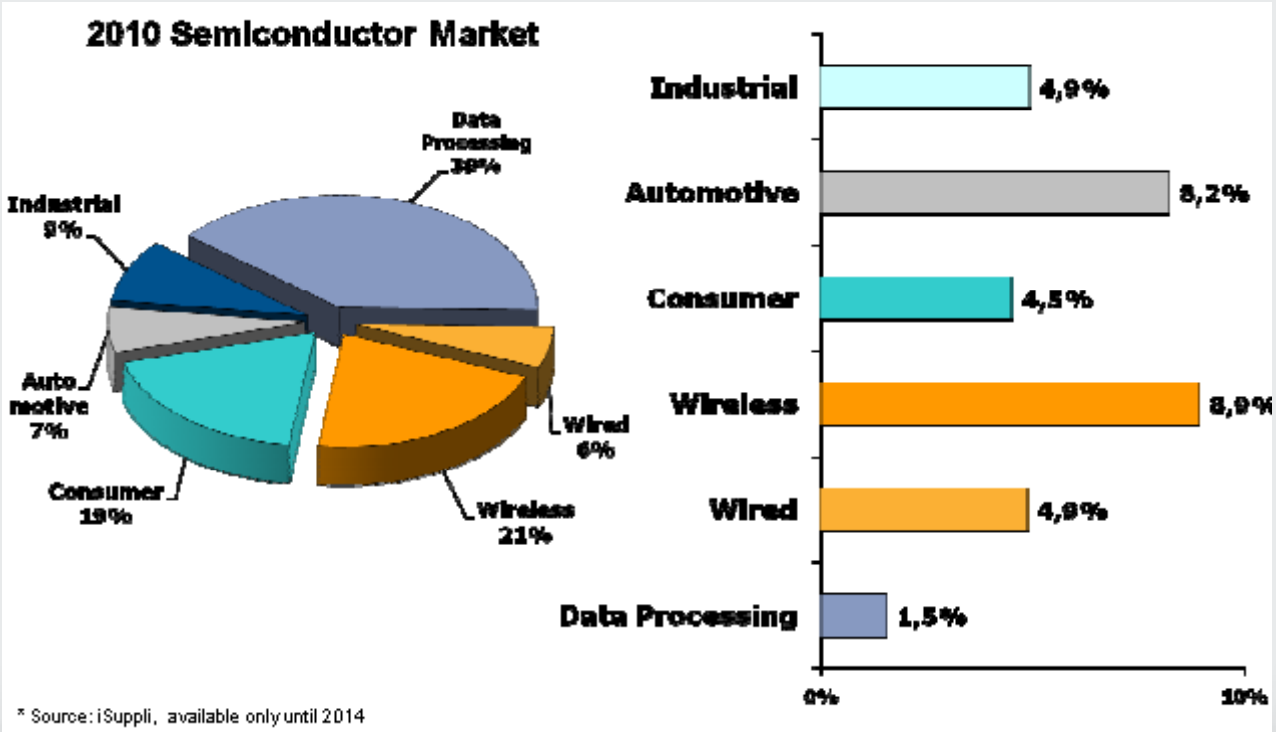
ARM Licensees

1H10 Rank	2009 Rank	Company	Headquarters	2009 Tot Semi	1Q10 Tot Semi	2Q10 Tot Semi	2Q/1Q % Change	1H10 Tot Semi
1	1	Intel	U.S.	32,325	9,485	9,910	4%	19,395
2 ★	2	Samsung	South Korea	21,273	7,137	8,159	14%	15,296
3 ★	3	TI	U.S.	9,697	2,990	3,262	9%	6,252
4 ★	5	Toshiba	Japan	9,537	3,240	3,002	-7%	6,242
5	6	TSMC**	Taiwan	8,989	2,885	3,293	14%	6,178
6	4	Renesas Electronics*	Japan	9,649	2,787	2,821	1%	5,608
7	9	Hynix	South Korea	6,320	2,455	2,807	14%	5,262
8 ★	7	ST	Europe	8,466	2,311	2,507	8%	4,818
9	10	Micron	U.S.	5,450	2,077	2,365	14%	4,442
10	15	Elpida	Japan	3,948	1,625	1,913	18%	3,538
11 ★	8	Qualcomm***	U.S.	6,409	1,537	1,691	10%	3,228
12 ★	11	AMD***	U.S.	5,403	1,574	1,653	5%	3,227
13 ★	13	Infineon	Europe	4,617	1,438	1,540	7%	2,978
14 ★	14	Broadcom***	U.S.	4,271	1,404	1,547	10%	2,951
15 ★	12	Sony	Japan	5,245	1,300	1,248	-4%	2,548
16 ★	16	NXP	Europe	3,547	1,079	1,105	2%	2,184
17	19	Freescale	U.S.	3,302	972	1,056	9%	2,028
18	24	UMC**	Japan	2,815	935	1,043	12%	1,978
19 ★	17	MediaTek***	Taiwan	3,500	1,024	940	-8%	1,964
20	18	Fujitsu	Japan	3,377	955	933	-2%	1,888
—	—	Total	—	158,140	49,210	52,795	7%	102,005

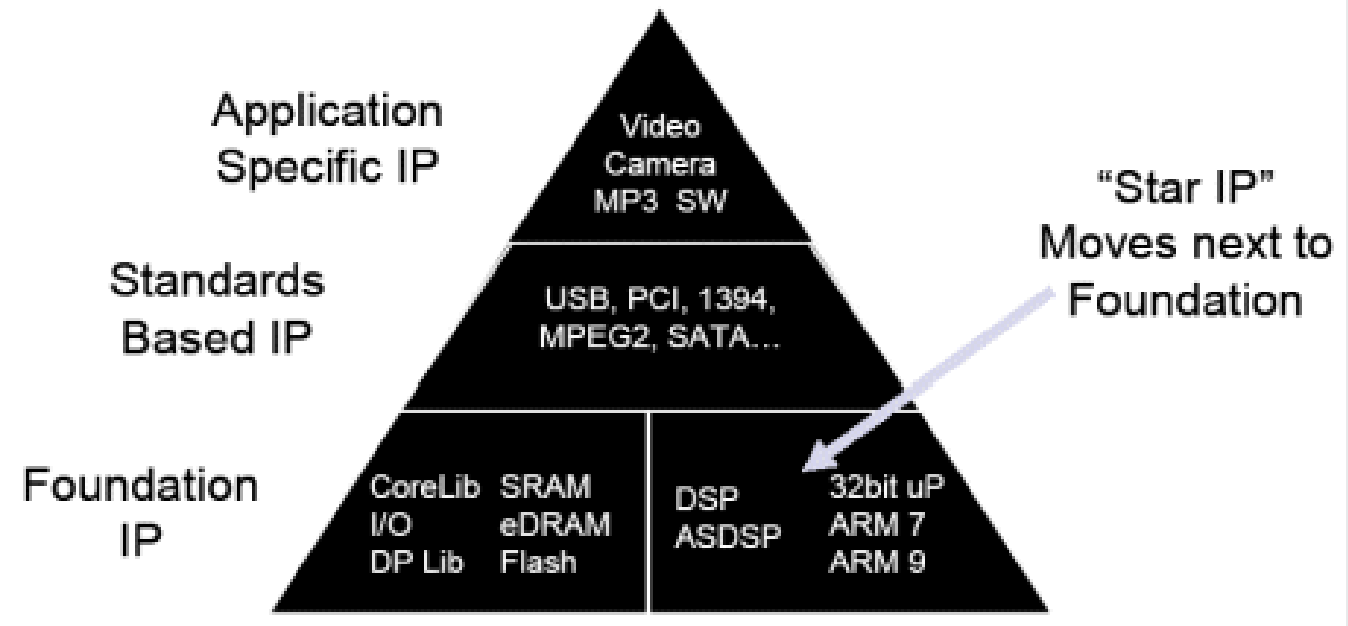
*The merged entity of Renesas and NEC **Foundry ***Fabless

Source: IC Insights, company reports

Semiconductor Industry Market



Foundation IP and Star IP



Magarshack 's (ST) view of IP Pyramid (2004)

- Companies outsource or subcontract for products and services not identified as elements of a core competitive competence (Star IP)

Multicore SoC: ST Nomadik Basic Principle



Quantum

Pocket TV (DVB-H)



JOY

LG ***Smartphone***



SPH-W2100



SCH-B560



SPH-B5600



SPH-B5650

SAMSUNG ***Mobile TV phone (T-DMB)***



Video Phone



TS7000

LENECO



MWN7000D

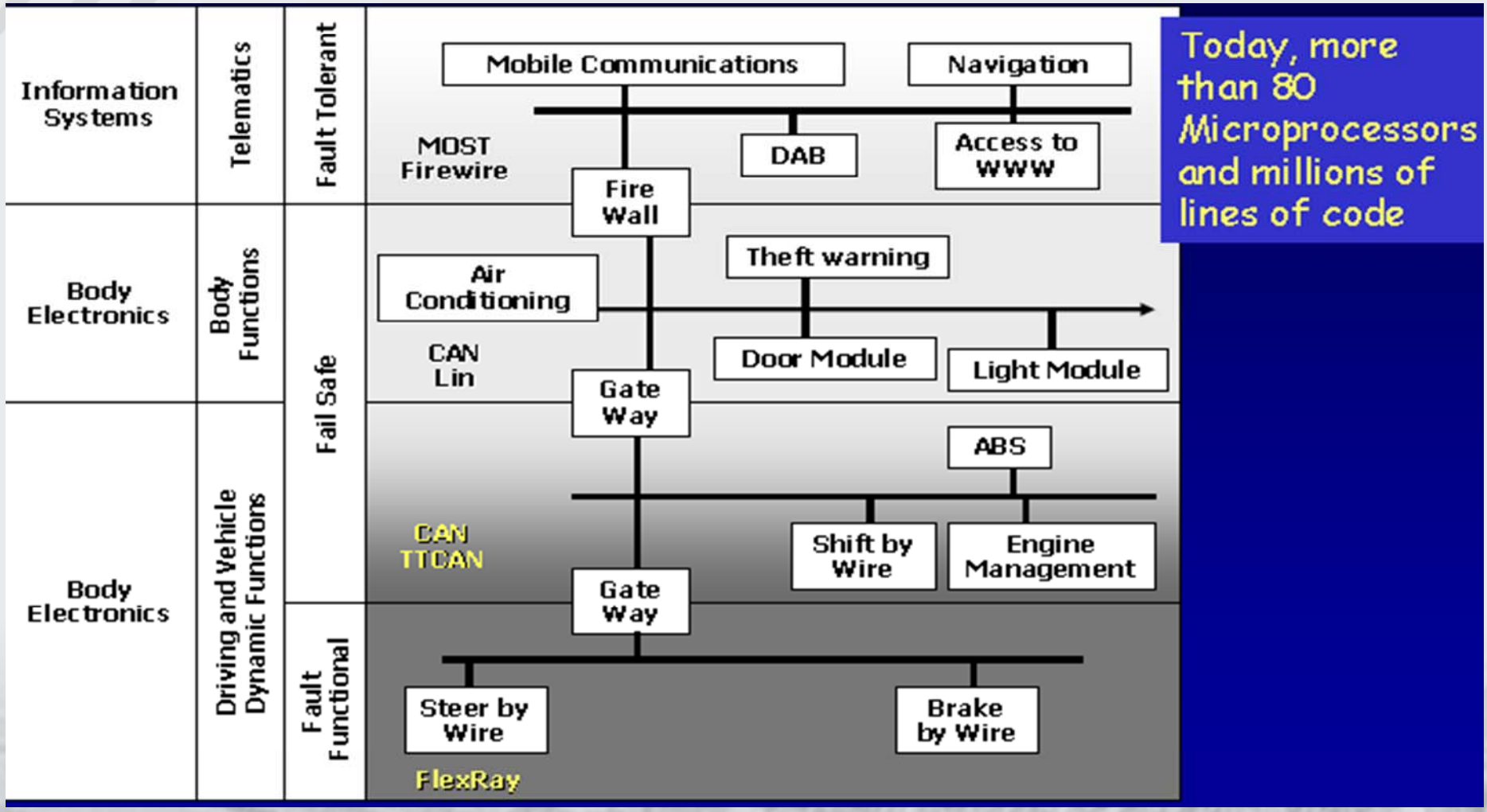
Mando



MMS DECT phone

Mobile TV + Navigation (T-DMB)

Automotive Industry



Trends in Automotive Industry

	2000	2010
Mechanical Parts	76%	55%
Electronics Parts (subsystem, system connection, integration)	13%	24%
Software Parts	2%	13%
Other	9%	8%

- Software cost could increase by 100X over the next 5-6 years. Challenges:
  - ✓ Software system-level architecture
  - ✓ System and subsystem reuse
  - ✓ Standardization of interfaces
  - ✓ HW/SW partitioning for system modularity and reliability
  - ✓ New design technologies and new materials

Electronic System Level Design for SoC

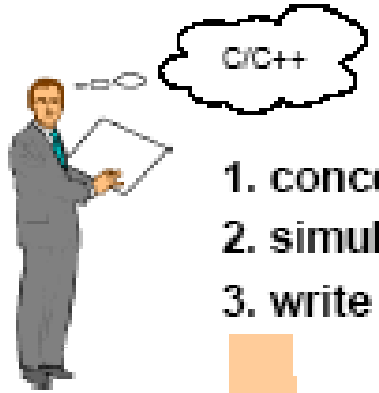
The escalating gate count, desired system heterogeneity and trend towards increased productivity for complex SoC devices challenges traditional RTL-to-gates design methodology based on VHDL or Verilog simulation.

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

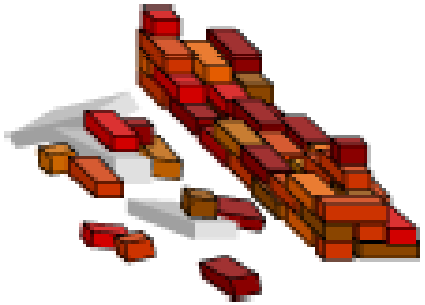
- C/C++ based design methodology



System architecture

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++

Hardware designer



OO Languages for System Design (late 90s)

The use of HDLs, such as Verilog and VHDL, leads to *enormous simulation time*, while forcing the designer to take *unnecessary implementation decisions*, *even* when block detail is not available.

⇒ Thus, industry has addressed SoC design by exploring two options:

- *create new system design tools by bridging multiple languages*, or
- *extend an existing specification/design language*, such as C/C++ or Java with hardware-oriented (memory, control, communication and synchronization) data structures. An object-oriented syntax and semantics enhances system design by defining blocks as an ordered collection of objects, while providing useful overloading and inheritance rules. Common C++-based libraries are either
  - *open source*, e.g. SpecC, SystemC and Cynapps' CynLib, or
  - *proprietary*, e.g. N2C by Coware and Superlog.

Protocol Refinement Methodology

Protocol refinement is a crucial part in the stepwise transformation of a high-level behavioral model into actual architecture implementation,

- allowing modeling at various level of abstractions, thus trading between *model accuracy* and *simulation speed*, and
- providing a *continuous rearrangement of existing IP in ever-new composites*, enabling a key process to new marketing ideas.

We distinguish between two fundamental types of refinement.

- In *top-down refinement*, we focus on capturing desired system requirements and refining the abstraction level down to architecture implementation by filling in details and constraints.
- In *bottom-up IP reuse oriented refinement*, we focus on the evaluation, composition and deployment of prefabricated IP, derived from existing libraries from a variety of sources.

Separation of Concerns: Behavior & Communication

Object-oriented systems consist of virtual components (IP), interacting with each other within a specific system environment. Thus, refinement is naturally based on separating each IP module into two components possibly expressed at different abstraction levels.

- A *behavior* component which describes module functionality. This is an active object in object-oriented programming, since it usually has an associated identity, state and an algorithmic process consuming or producing communication cells, synchronizing or processing data objects.
- A *communication interface* which consists of I/O ports which transfer messages between one or more concurrent behavior components. The interface connects to a communication channel supporting various communication protocols and is considered the only way to interact with the behavior.

Fully-decoupling behavior from inter-module communicationresults in

- ✓ clear environment with distinct functionality and interaction,
- ✓ during refinement we only change the interaction model.

Separation of Concerns: Functionality & Implementation

- Separation of functionality and implementation
  - ✓ functionality refers to the behavior the designers want the system to provide
  - ✓ implementation is a configuration of resources that implement functionality.
- The two design aspects are fairly orthogonal, except that the implementation determines the performance at which the functionality is implemented.
  - ✓ A separate functionality model ⇒ reuse with many possible implementations.
  - ✓ An orthogonal implementation ⇒ reuse in evaluating implementations of alternate functionality.

Separation of Concerns: Capability & Cost

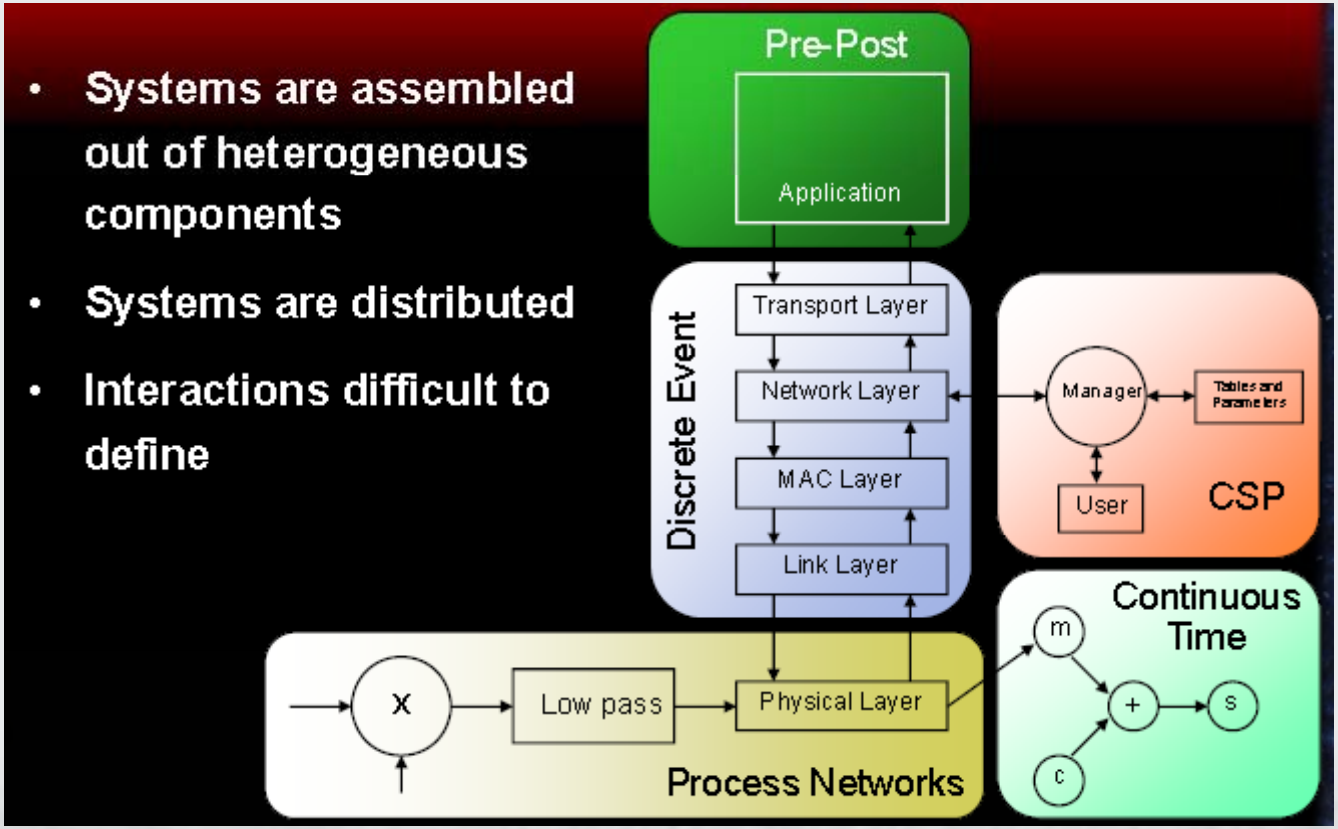
- Capability refers to the entire set of system behaviors, while system cost refers to implementation of a given behavior.
- For example, in CPU modeling:
  - ✓ capability may refer to the type of processor instructions, e.g. addition, data move
  - ✓ cost refers to the latency, throughput or power consumption of an instruction.
- The same set of instructions (capability) realized by another CPU with different cost function $\Rightarrow$ reuse of both models

Refinement Methodology & Partitioning

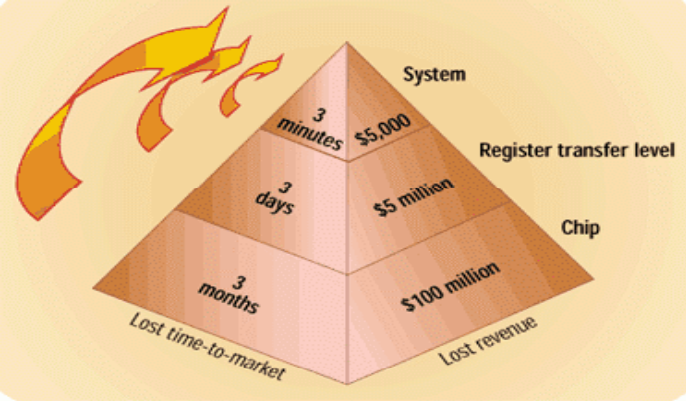
Through the refinement process, behavior and communication interfaces are eventually mapped to hardware/software resources of a particular architecture. This categorization process, called *system partitioning*, is the essential element of co-design.

- Behaviors mapped to hardware are then either synthesized, or selected from existing IP implementations, e.g. processor cores, bus interfaces and protocols.
- Behaviors mapped to software are assigned to a software task or an interrupt-service device driver routine.
- Similarly, communication protocols are mapped to hardware logic, or software processes depending on the particular communication semantics, e.g. shared memory, message passing, Ada-like rendezvous, or other message queuing system, e.g. Mailbox.

Models of Computation

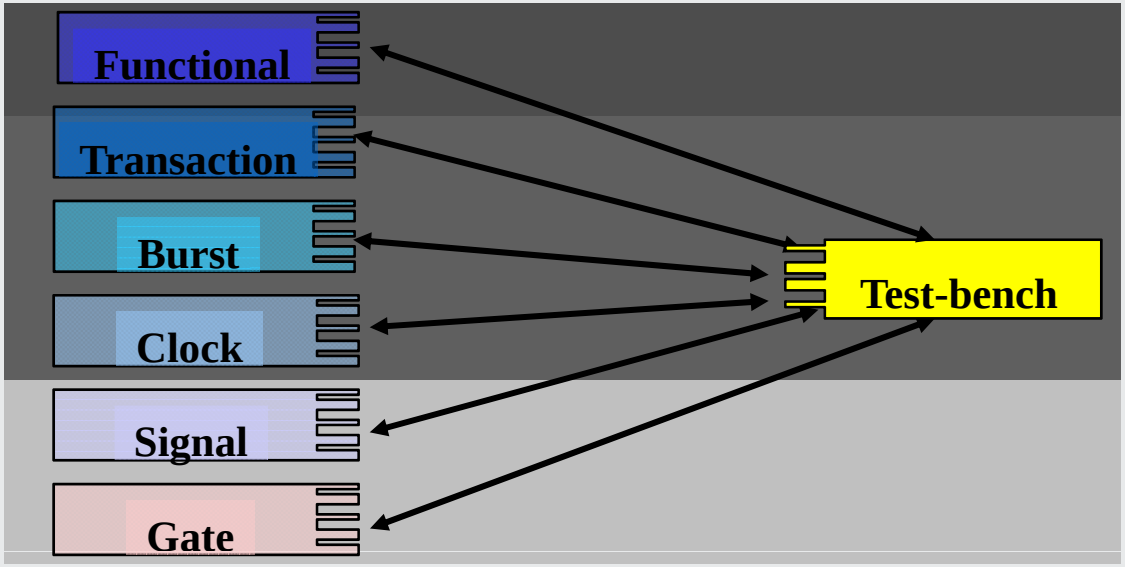


Virtual Platform and Abstraction Levels



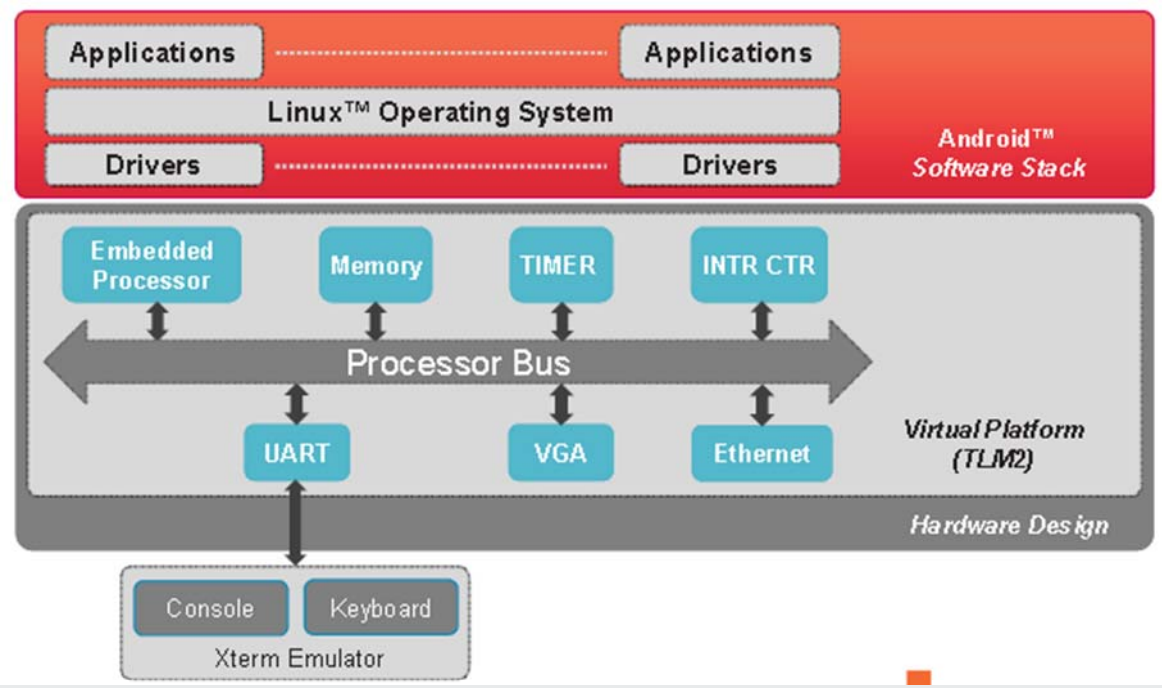
Source: Integrated Communications Design, 2001

Higher abstraction implies shorter iteration cycles and less lost revenue

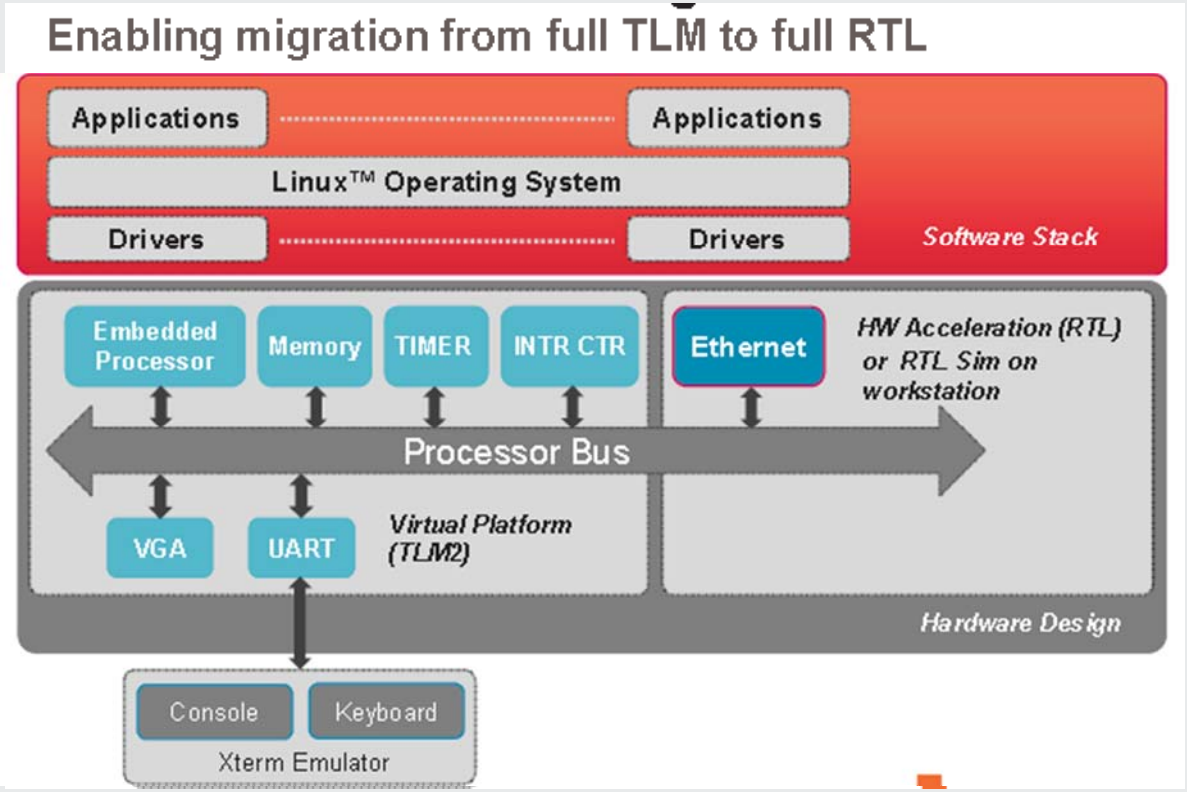


Virtual platform refers to modeling the overall SoC by hiding, modifying or omitting system properties. Abstraction levels span multiple levels of accuracy, with each level introducing new model details.

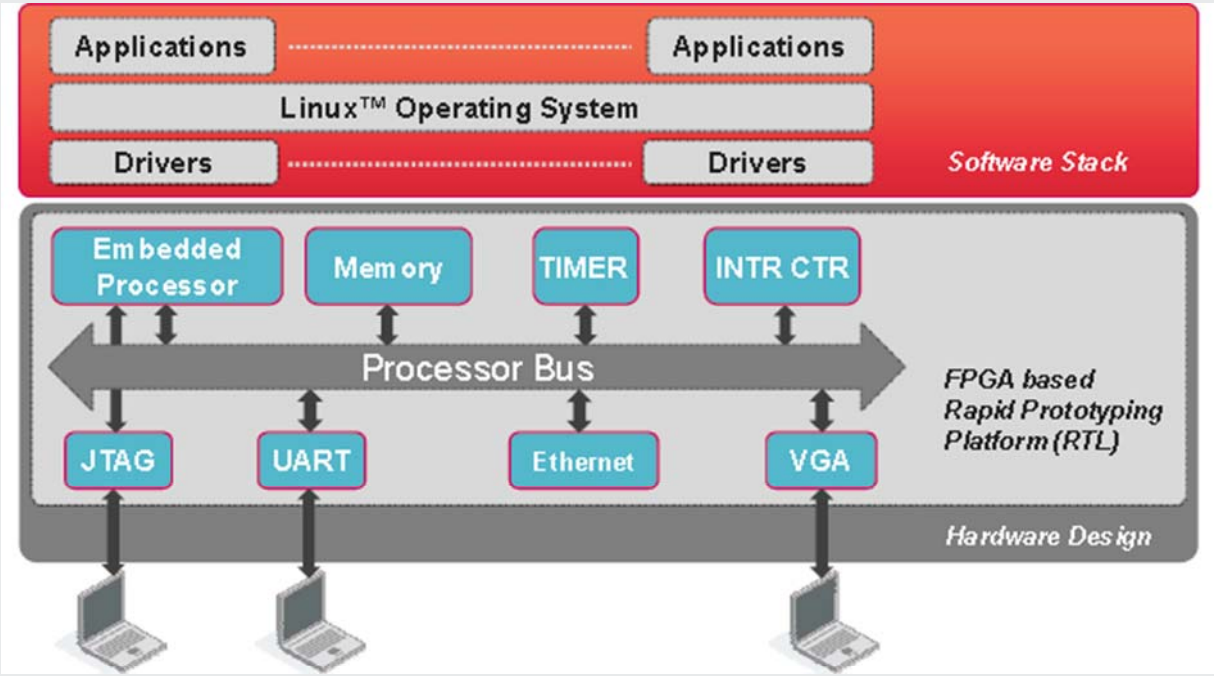
Pre-RTL Software Design&Debug (via VP)



TLM to RTL (HW/SW Partitioning)



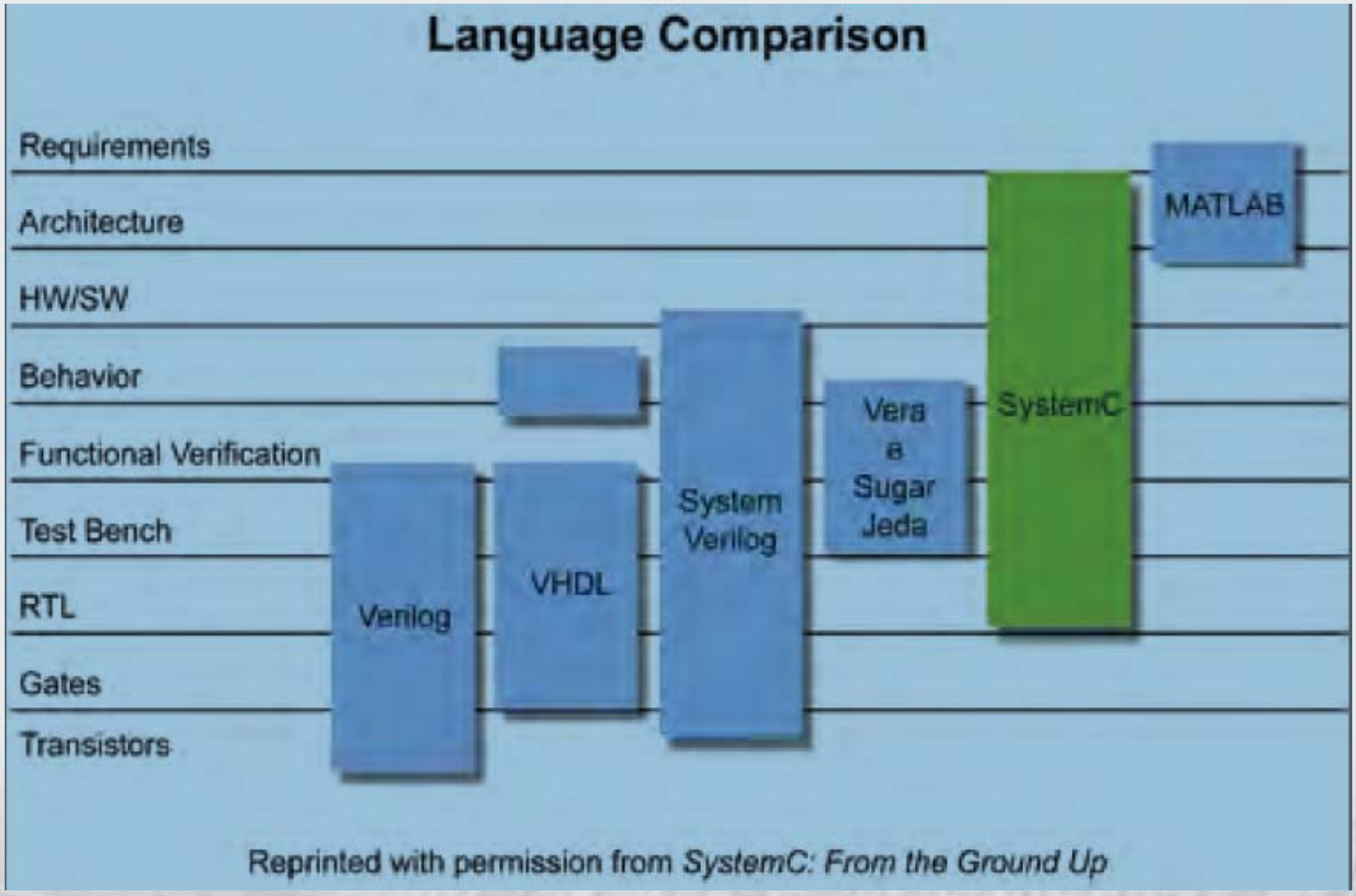
Modeling to FPGA Prototyping



Higher Abstraction for Faster Simulation

- Untimed functional model
  - ✓ Executable specification to check functional correctness
  - ✓ Simulates at the speed of C program execution, but no timing
- Timed architecture model
  - ✓ Used to evaluate HW/SW partitioning
  - ✓ Computation distributed onto system components
- Transaction level model
  - ✓ Used to evaluate system with abstract communication
  - ✓ Transactions vs. bit toggling (data abstraction)
- Bus functional model
  - ✓ Communication modeled at pin-accurate and time-accurate level
  - ✓ Computation modeled at functional level
- Cycle accurate model
  - ✓ HW and SW at cycle accurate level
  - ✓ Communication at cycle accurate level

Design Languages: Overview



Design Languages: Overview

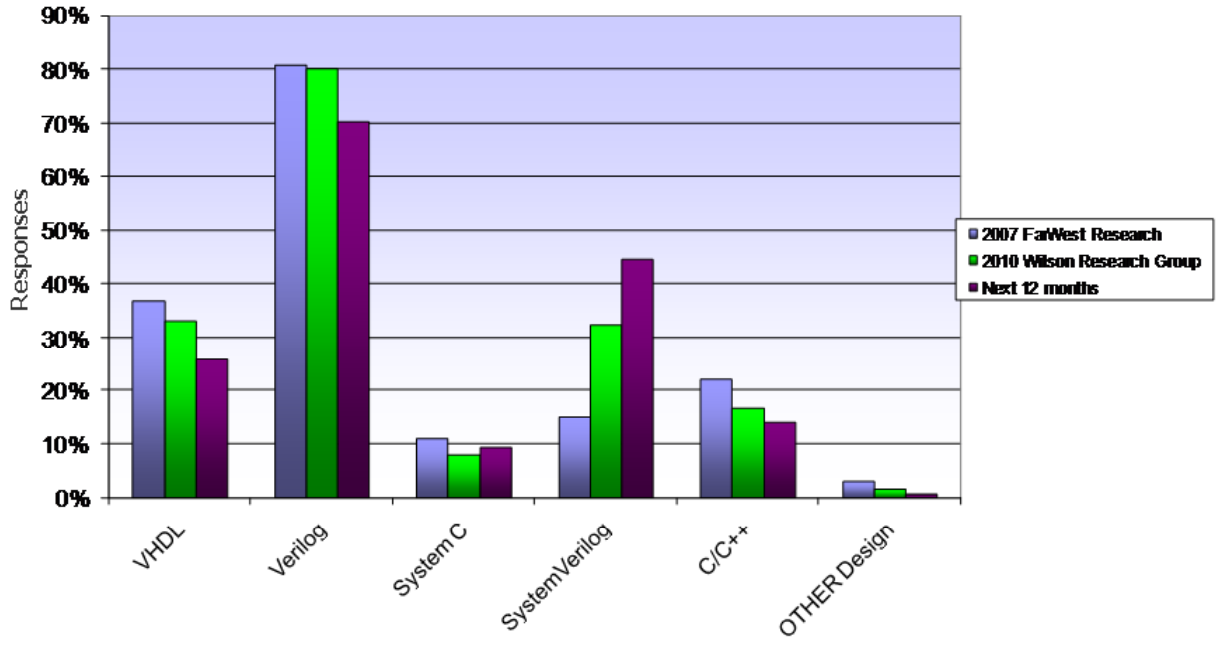


- Matlab and several other languages and tools, such as Synopsys Innovator, can capture system requirements. Other higher-level languages are Esterel and BlueSpec, but the former never proved to be highly popular, while the latter is too new.
- SystemC is an open industry-standard design language targeted at architectural, algorithmic and transaction-level modeling. SystemC is useful for early architecture exploration, performance modeling, hw/sw co-design or virtual prototyping for early software development.
- Vera, e and Testbuilder implement assertion-based, testbench generation, and coverage-driven verification using constrained random techniques for functional verification of complex ASICs.
- SystemVerilog is the industry's first unified hw design and verification language (HDVL) that provides upward compatibility to the Verilog language, merging high-level features from VHDL, C, C++ (object orientation) and commands and constructs from verification languages.
- HDLs (VHDL and Verilog) used for simulating/synthesizing circuits.

Design Languages: Trends

Languages and Libraries

Trends: Languages used for design



Non-FPGA Designs

Wilson Research Group and Mentor Graphics
2010 Functional Verification Study, Used with permission

2 HF - Compilation and Analysis performed in January 2011

© 2011 Mentor Graphics Corp. Company
www.mentor.com

Lecture 3: System-Level Design using SystemC

Miltos Grammatikakis

SystemC for SoC Design

SystemC is a collection of C++ classes describing hardware concepts and a simulation kernel implementing the runtime semantics. SystemC allows the development and exchange of fast system models and seamless tool integration from a variety of vendors. Two commercial tools supporting SystemC have been recently announced.

- In September 2000, Cadence Berkeley Labs' Felix initiative delivered the Cierto Virtual Component Co-design tool suite (VCC 2.0). VCC supports both top-down design based on communication refinement, and bottom-up reuse-oriented design.
- In June 2001, Synopsys formally announced the Cocentric System Studio and Cocentric SystemC compiler. The tool is a design entry and modeling environment that utilizes block and state diagramming techniques using graphical abstractions to represent concurrency, reactive control and dynamic switching concepts.

What is SystemC?

SystemC is a library of C++ classes for hardware modeling together with a lightweight simulation kernel expressing runtime semantics.

The SystemC library includes:

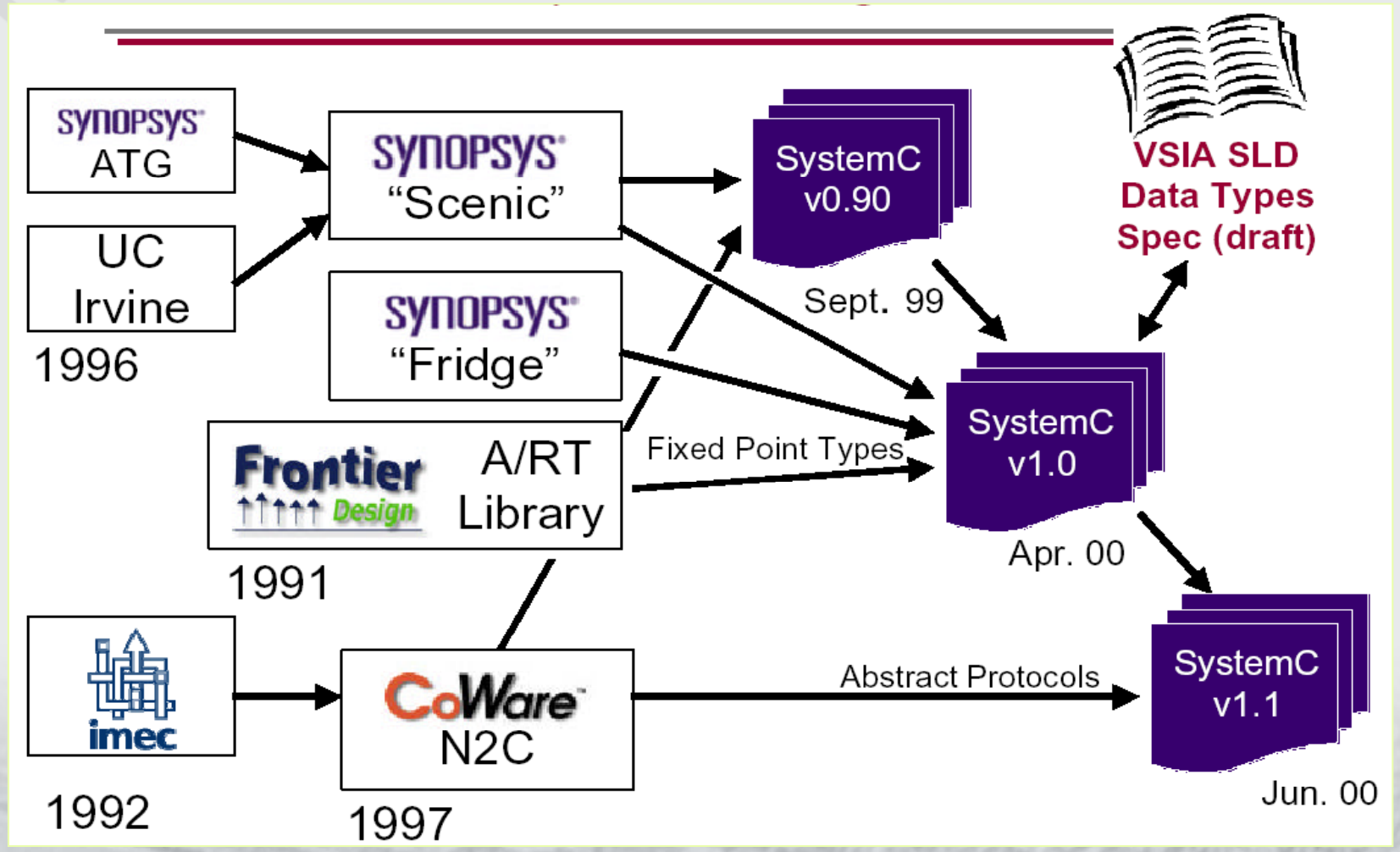
- processes (for concurrency),
- clocks (for timed models),
- hardware data types (bit vectors, 4-valued logic, fixed-point types, arbitrary precision integers),
- waiting and watching (for reactivity),
- modules, ports, signals (for hierarchy), and
- abstract ports and protocols (abstract communications).

Why SystemC ?

- It is the de facto system level modeling standard.
  - √ EDA support from 20+ companies: Cadence, Synopsys etc
  - √ User base: hands down leader in corporations and universities
  - √ Knowledge base: worldwide training, user groups and SystemC forum
- It features the best modeling technology
  - √ Breadth of datatypes, including fixed point, arbitrary width, 4-state
  - √ Abstraction levels, with ability to mix code in C++ and v2.0
  - √ Dynamic processes in v3.0
- It is the most open olution!
  - √ In terms of contributors, processes and supporters.
  - √ Be part of the SystemC community at www.systemc.org

SOURCE: SystemC L

History of SystemC



SystemC History

SystemC History

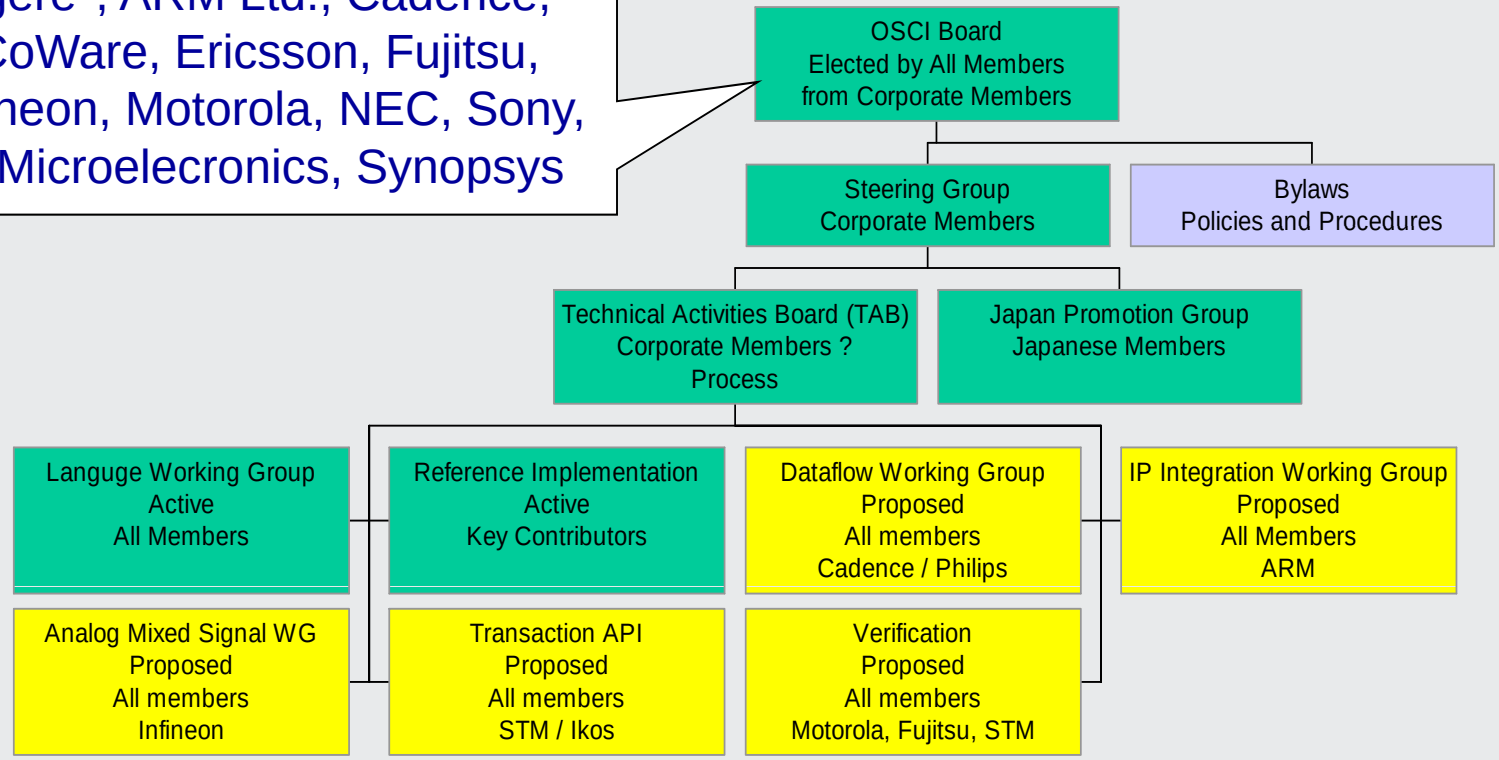
- SystemC 1.0
  - Provide VHDL like capabilities
  - Simulation kernel
  - Fixed point arithmetic data types
  - Signals (communication channels)
  - Modules
    - Break down designs into smaller parts

SystemC History

- SystemC 2.0
  - Complete library rewrite to upgrade into true SLDL
  - Events as primitive behavior triggers
  - Channels, Interfaces and Ports
  - Much more powerful modeling for Transaction Level
- Future SystemC 3.0
  - Modeling of OSs
  - Support of embedded S/W models

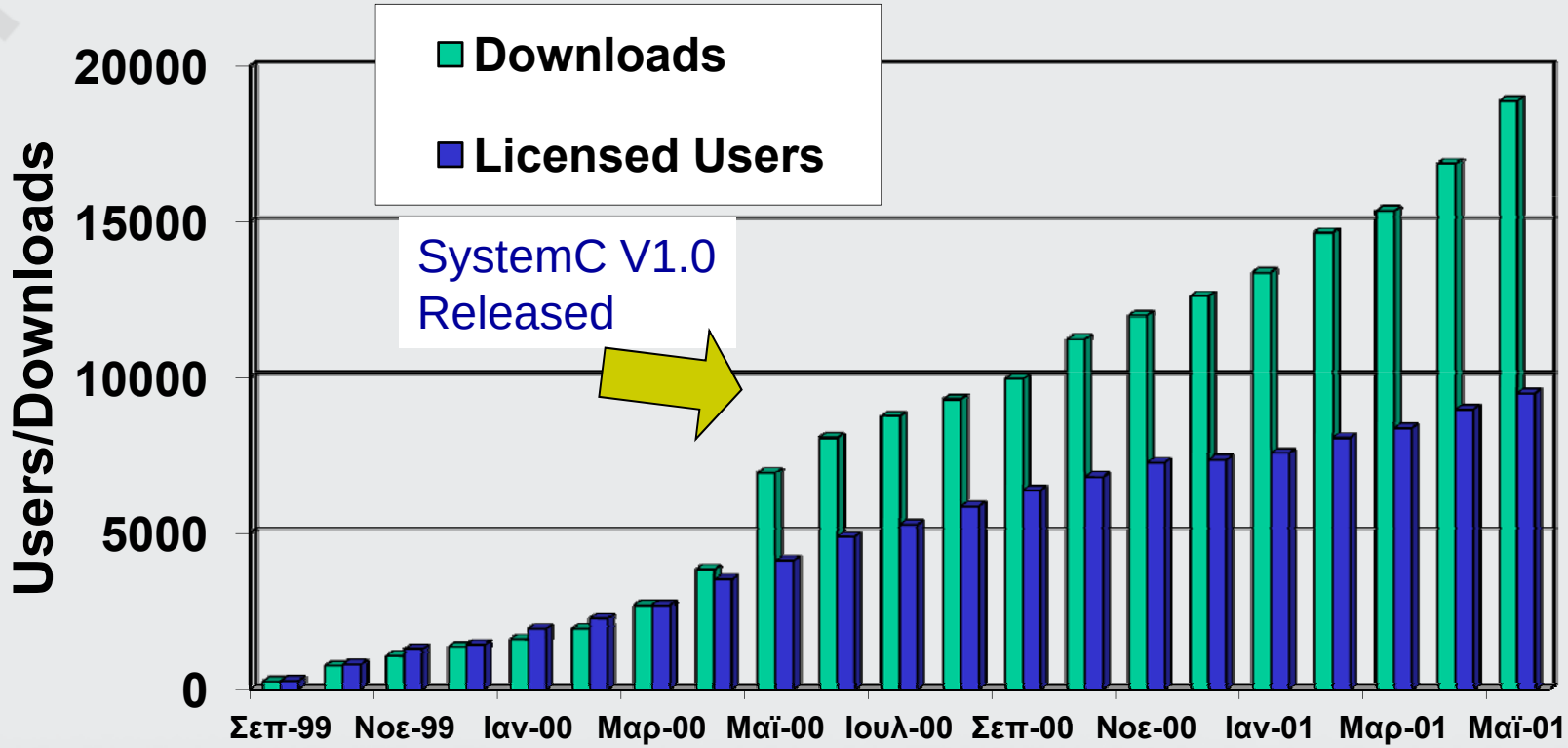
Original OSCI Structure

Agere*, ARM Ltd., Cadence, CoWare, Ericsson, Fujitsu, Infineon, Motorola, NEC, Sony, ST Microelecronics, Synopsys



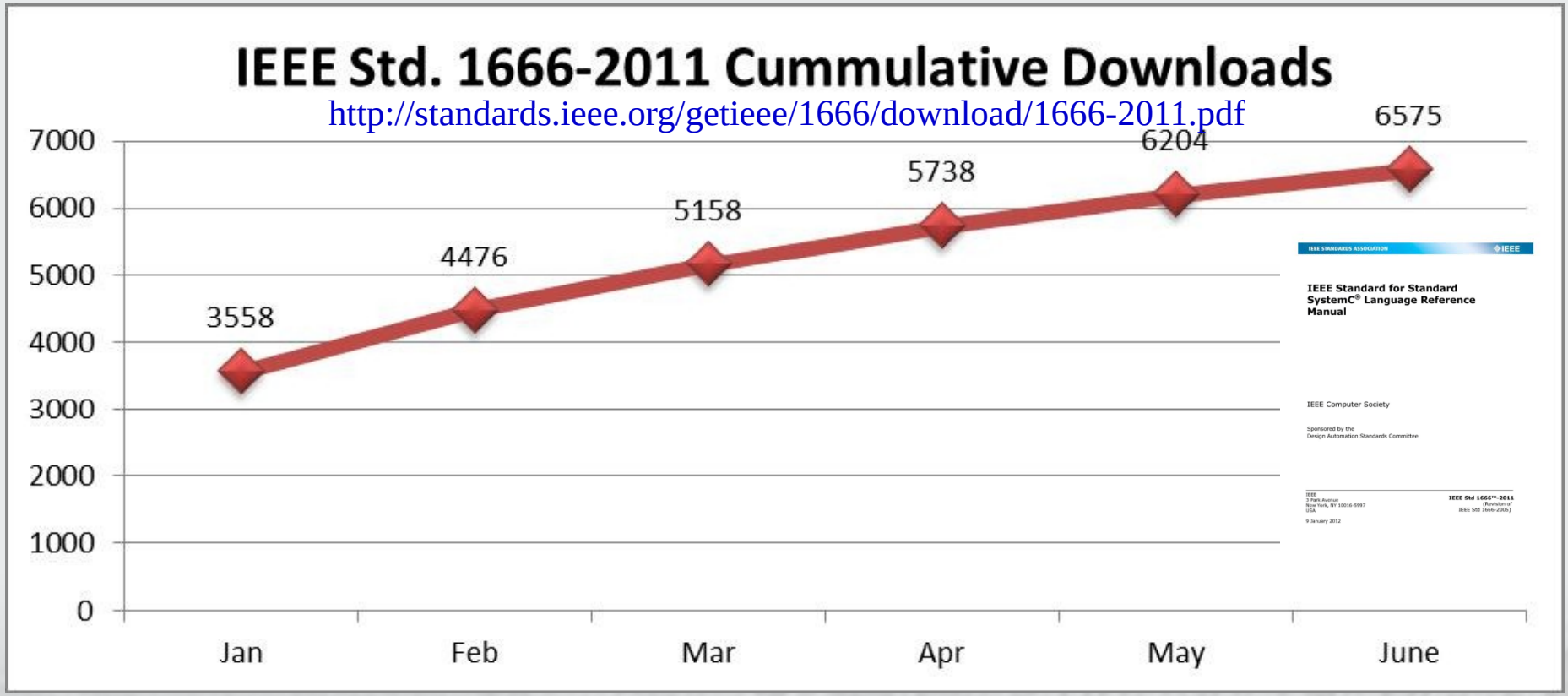
SOURCE: SystemC L

Early SystemC Adoption



SOURCE: SystemC L

SystemC Users Group

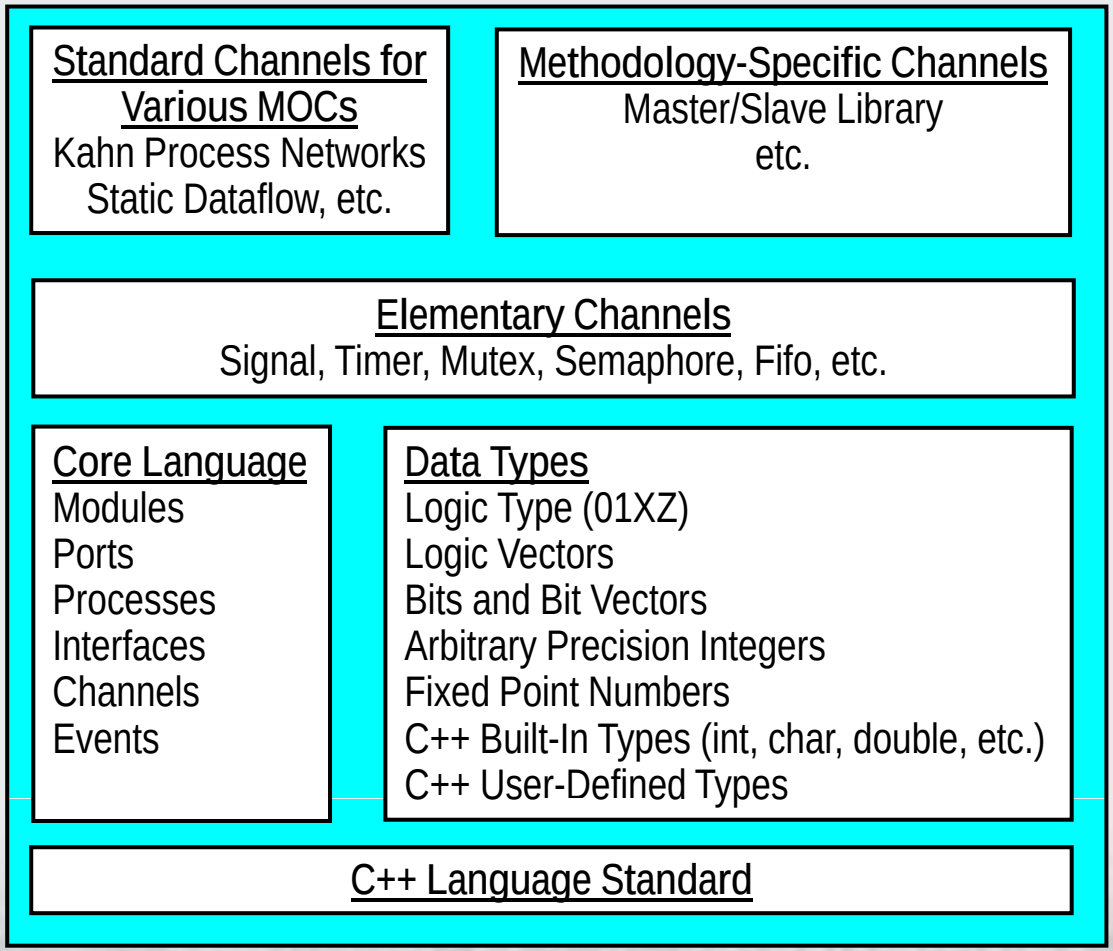


SOURCE: SystemC LWG

SystemC 2.0+ Language Architecture

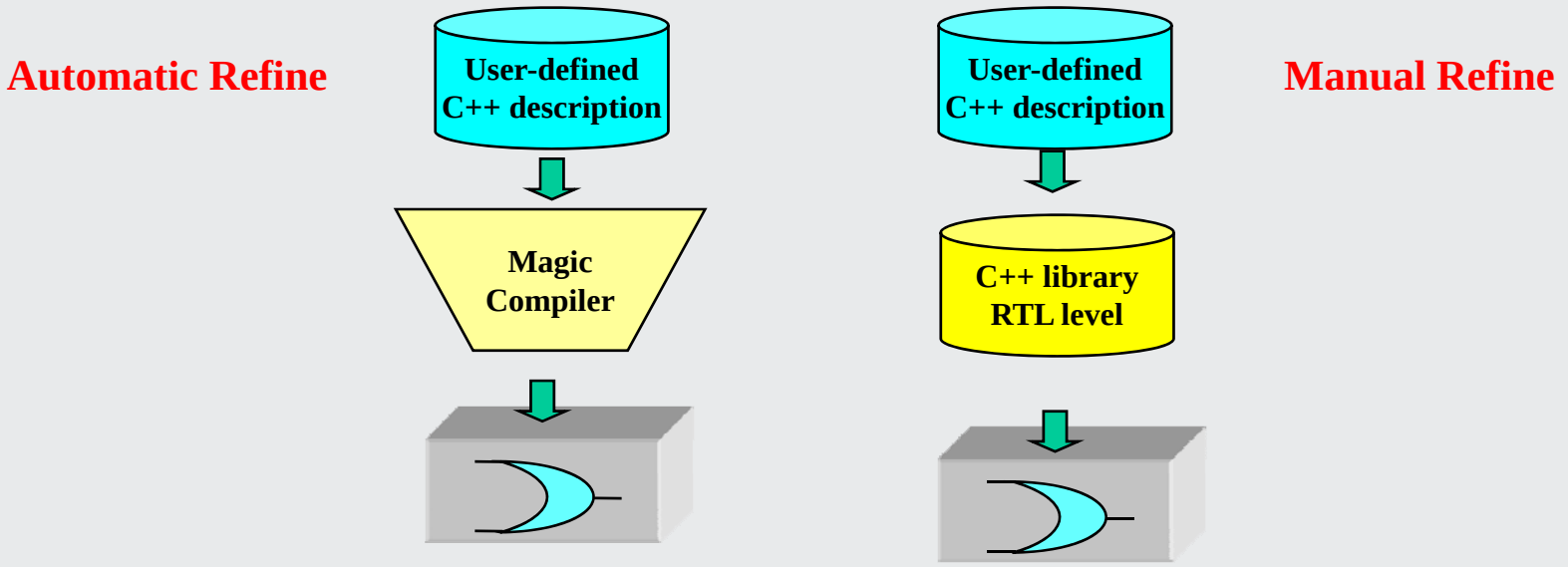
Upper layers are built cleanly on lower layers.

Lower layers can be used without upper layers.



SOURCE: SystemC L

Design Flow for C++ Models



- A specialized C++ library makes C++ appropriate for SoC design.
- Users may either manually decompose their final design to the RTL level of abstraction, or sometimes use SystemC-based behavioral synthesis tools, such as Cadence VCC, or Synopsys Co-Centric Studio.

SystemC Language Architecture

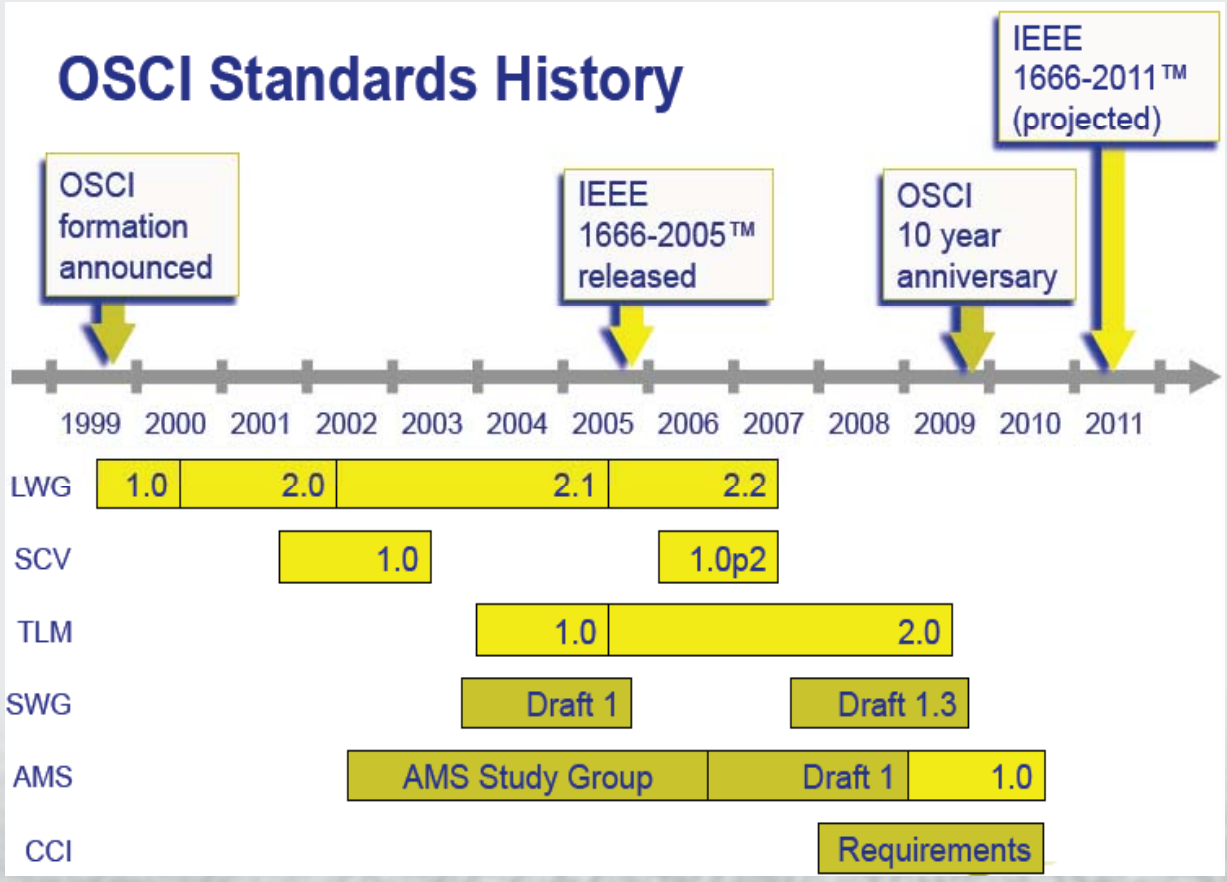
- SystemC is a leading open industry-standard modeling language based on a C++ library targeting architectural, algorithmic, and transaction-level modeling, although it can also be used for RTL.
- SystemC acceptance is driven by design complexity, targeting system-level hw/sw modeling and simulation at higher level of abstractions,
  - ✓ enabling effective early design space exploration and co-design,
  - ✓ 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)

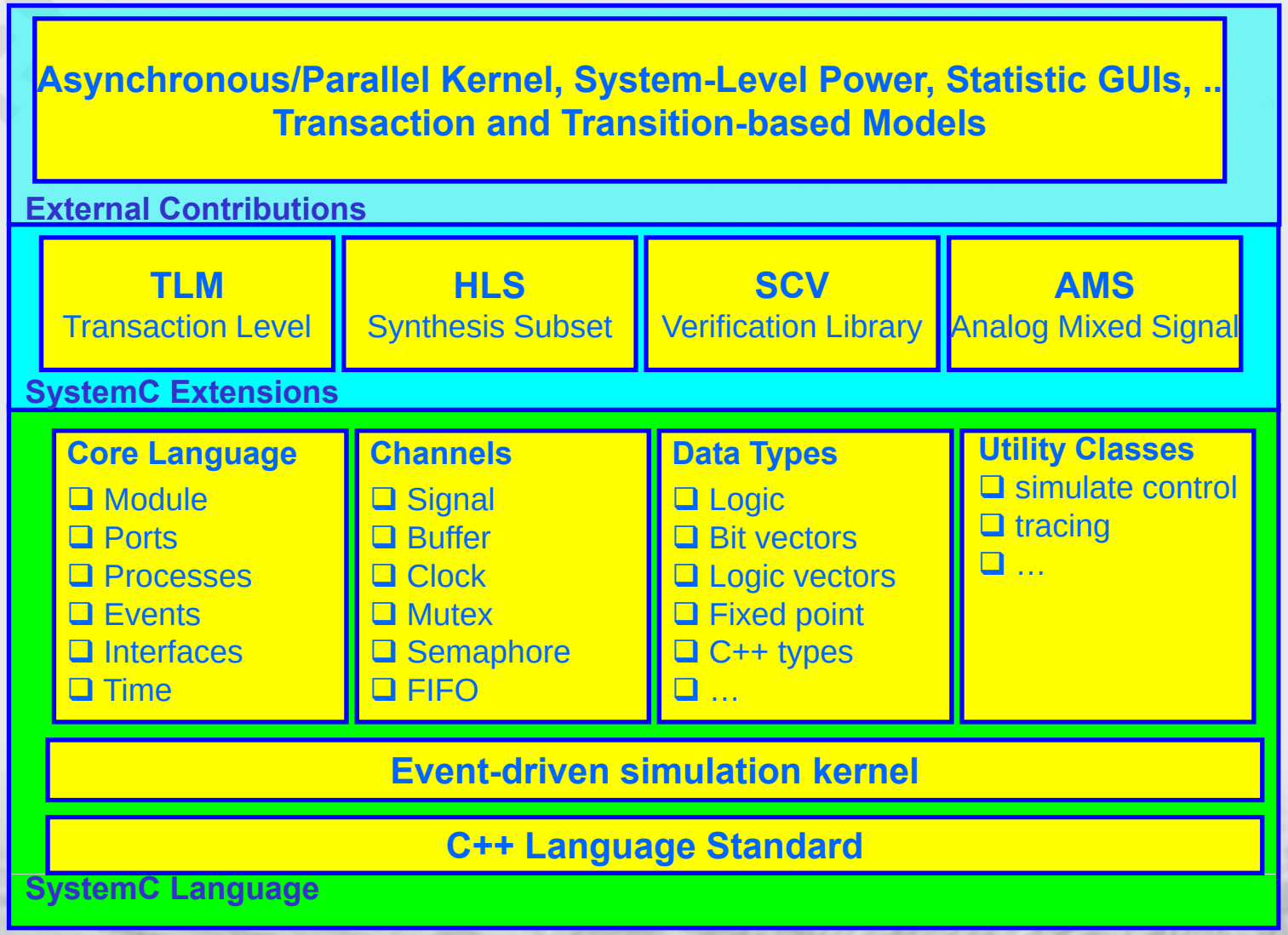


- User Groups in Europe, North America, Latin America, Japan, Taiwan
- IEEE 1666-2005 SystemC Language Reference Manual (LRM)

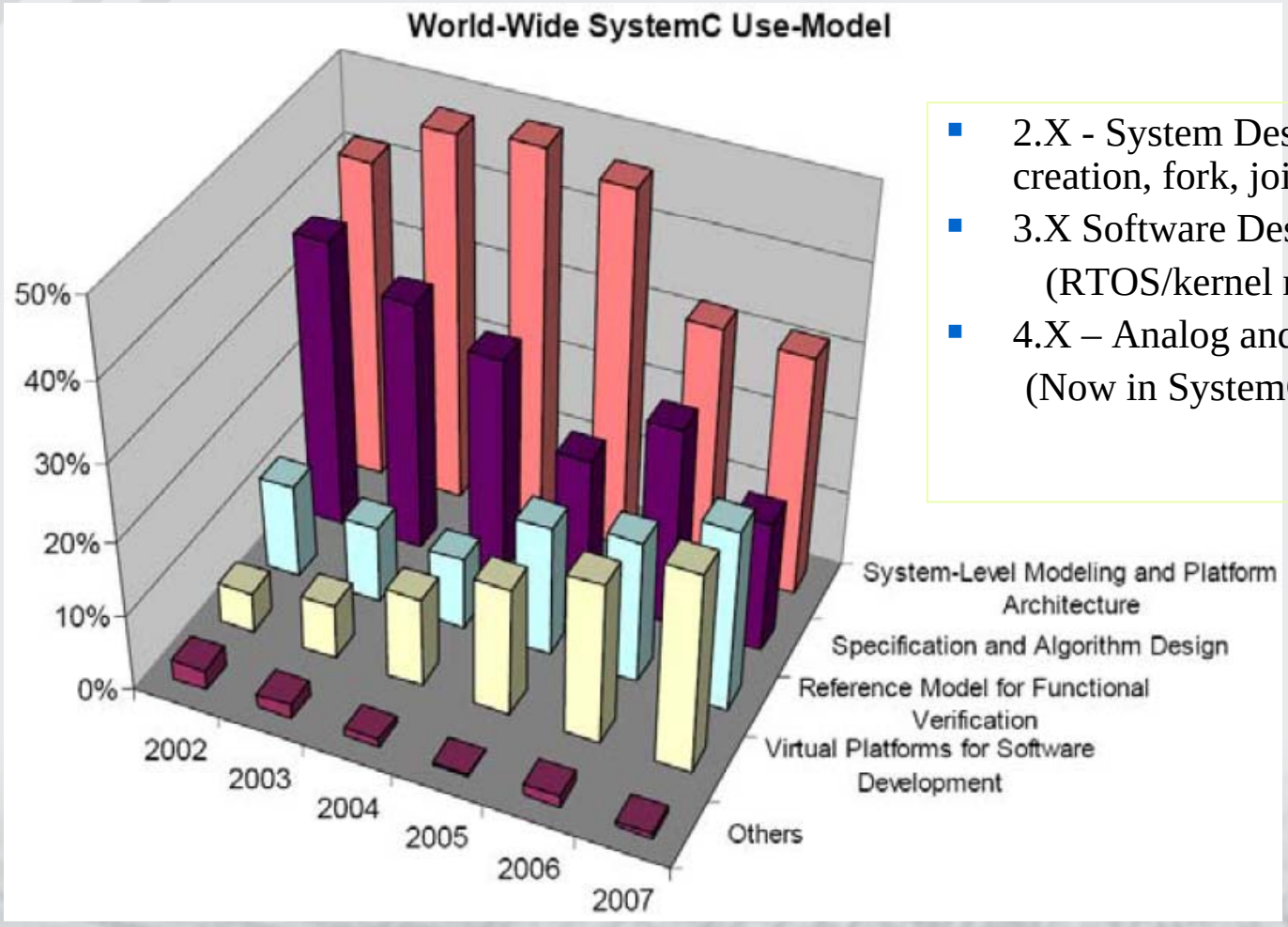
OSCI Standards



SystemC Layers & Extensions

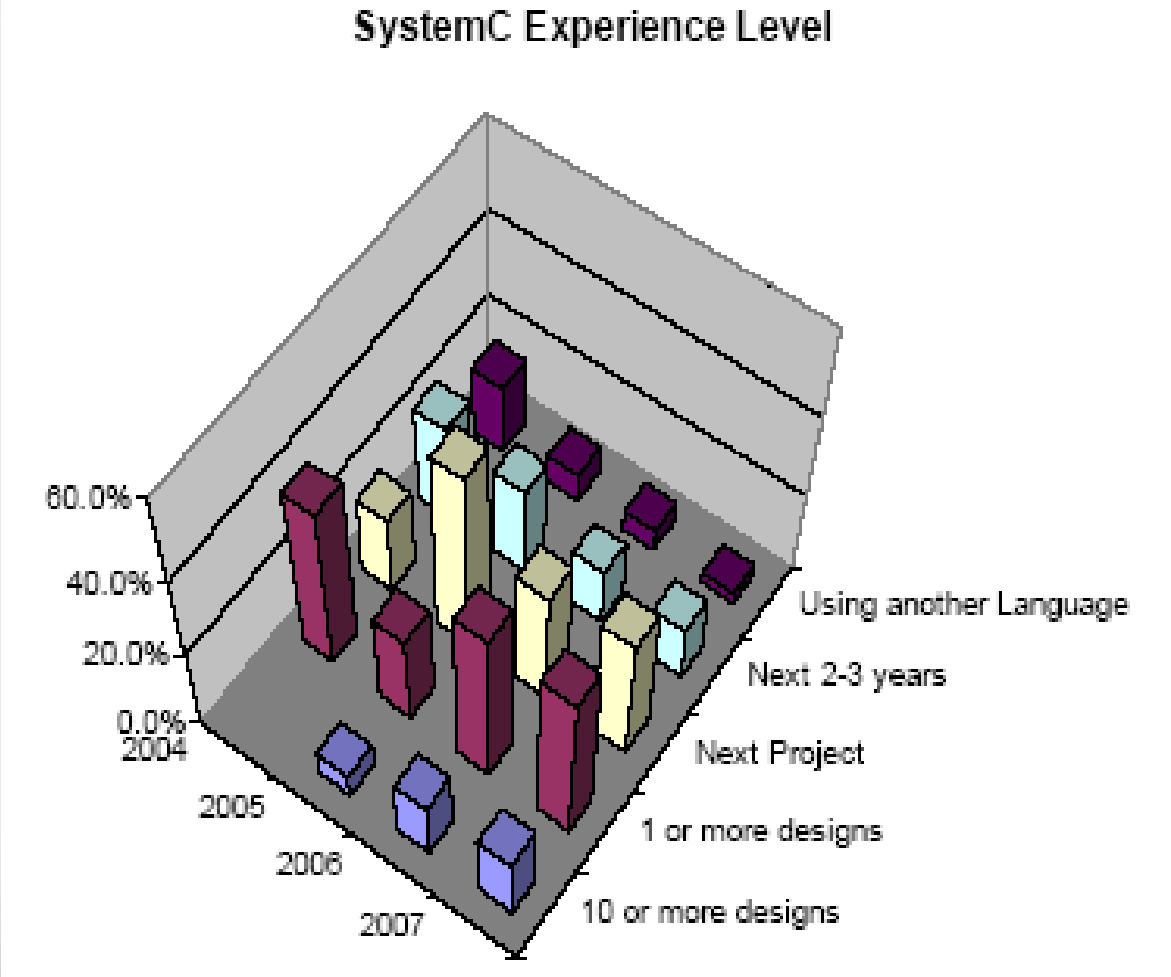


Current SystemC Popularity & Usage

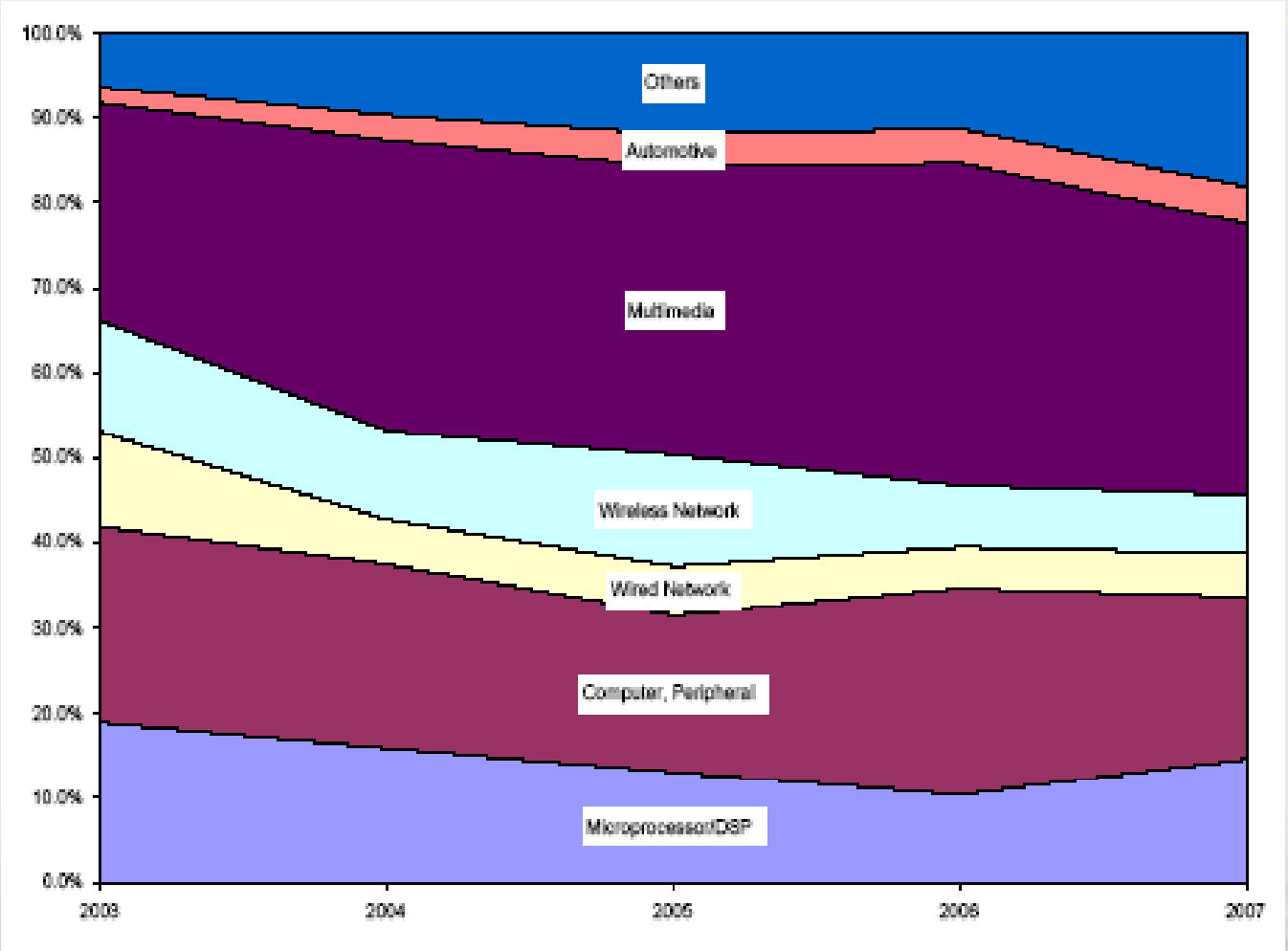


- 2.X - System Design Flow (thread creation, fork, join)
- 3.X Software Design Flow (RTOS/kernel model)
- 4.X – Analog and/or mixed signal (Now in SystemC-AMS v1 .0)

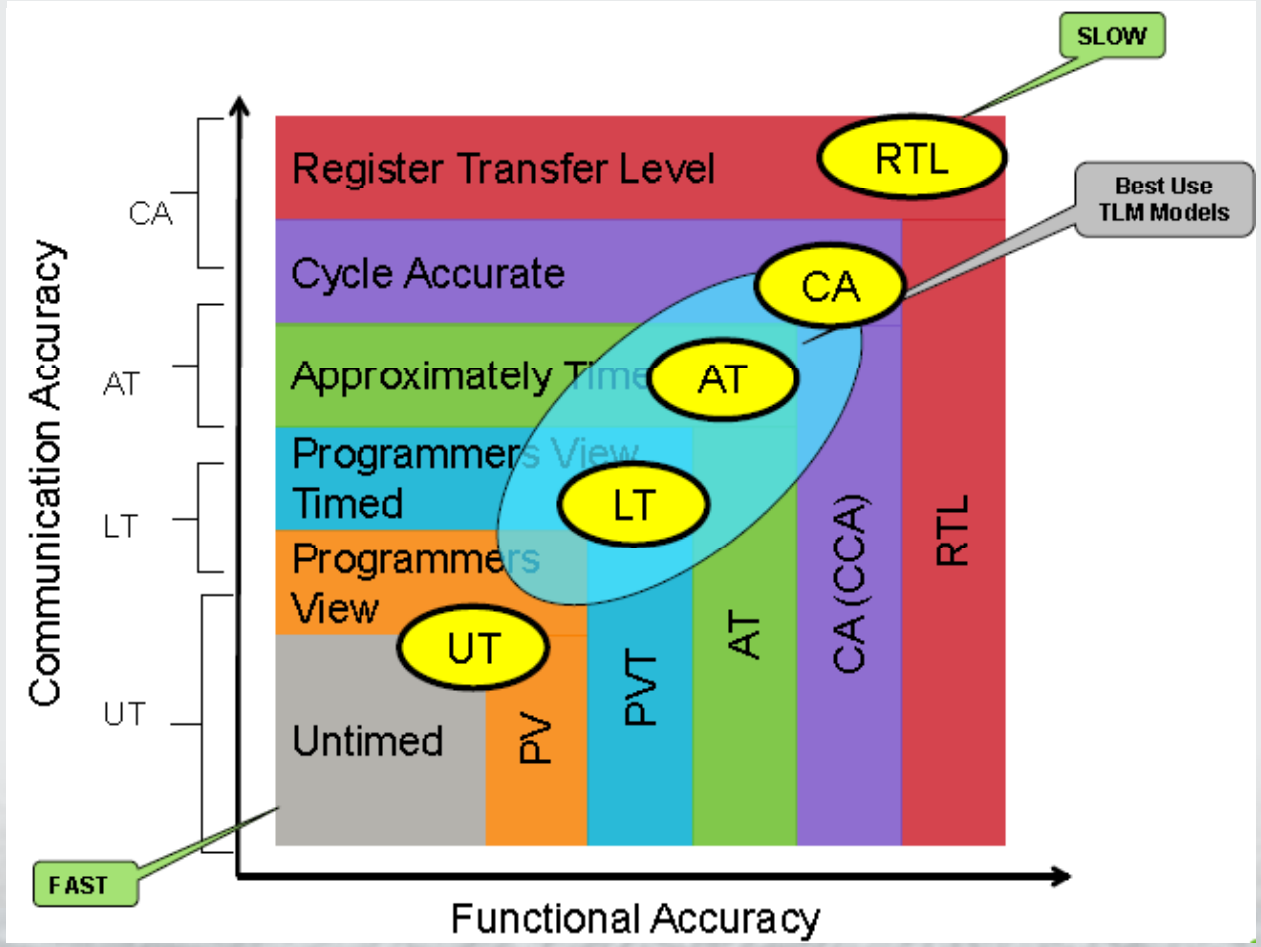
Current SystemC Usage



SystemC Application Areas



Transaction-Level Modeling (TLM)



Refinement & TLM

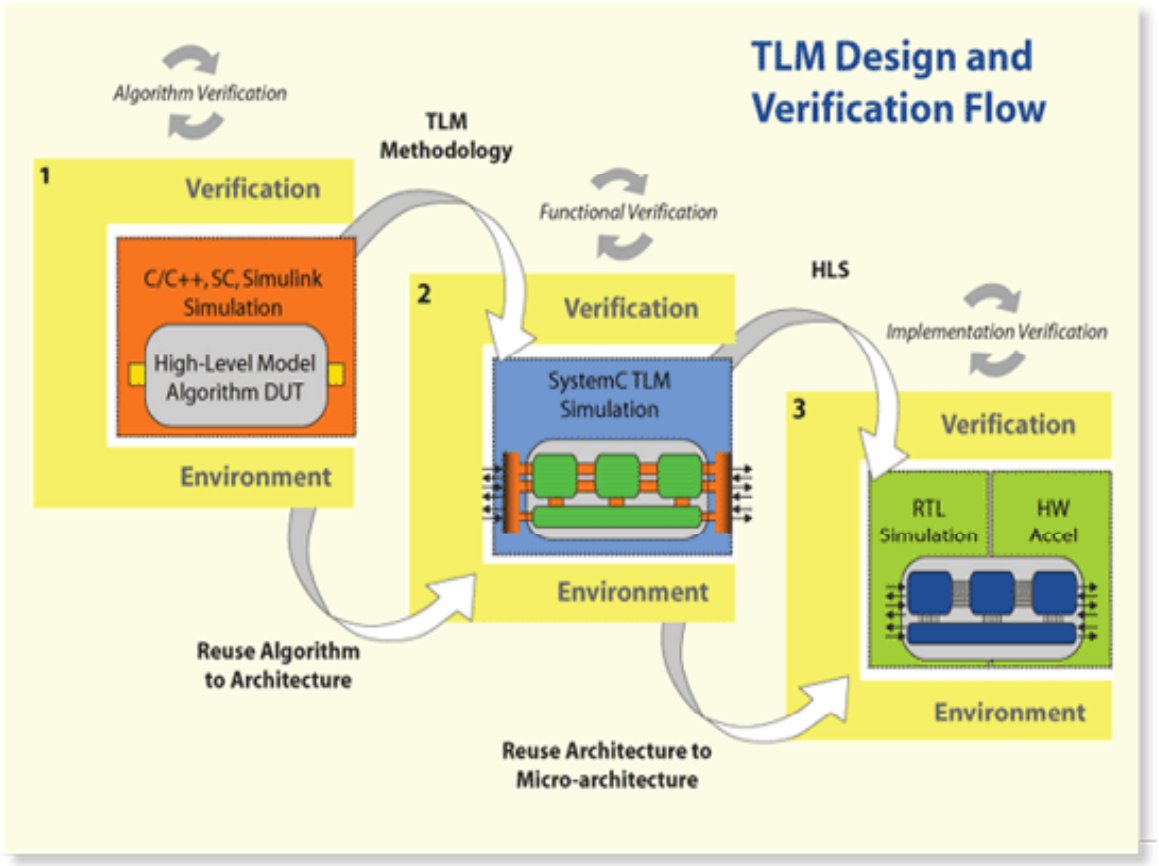


Figure 4. TLM design and verification flow.

TLM Benefits

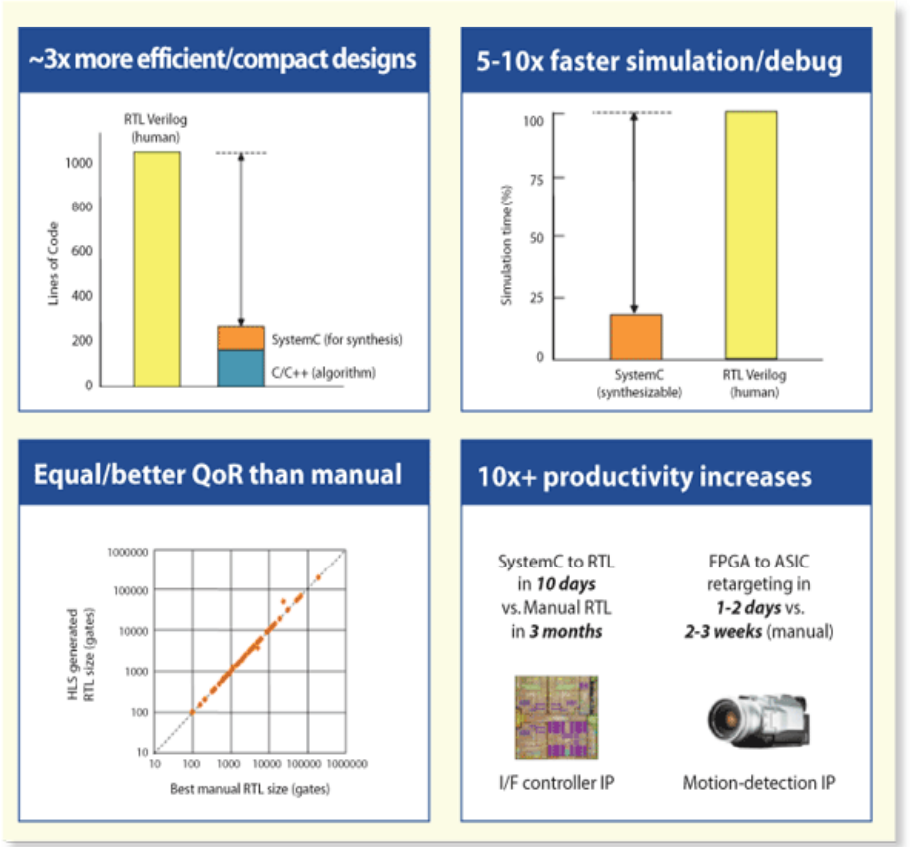
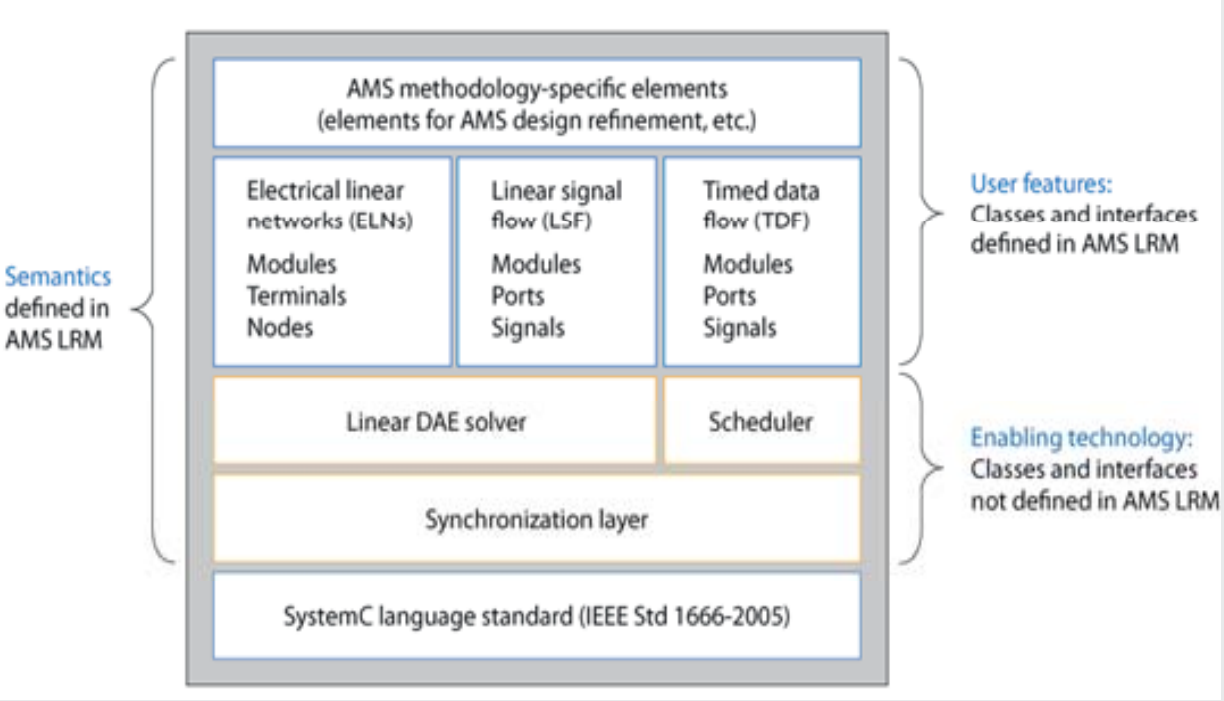


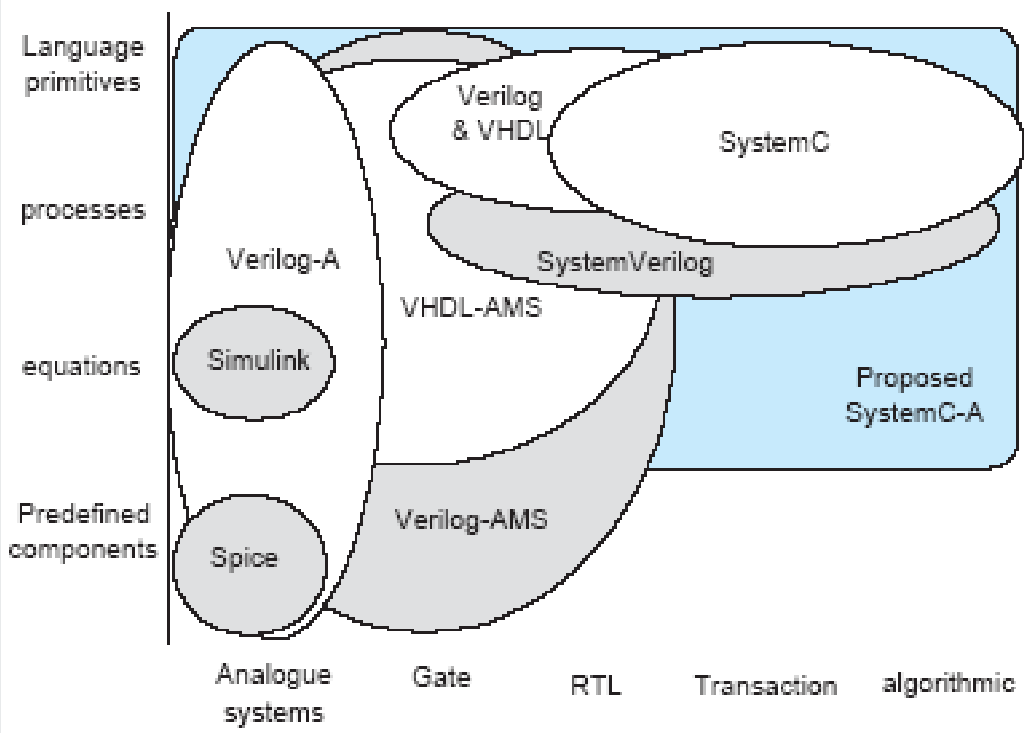
Figure 5. Benefits of new TLM design and verification methodology vs. traditional RTL (summary of actual data obtained in 2007-2008 by early adopters).

SystemC AMS Extension



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

Comparative Map of AMS Design Languages

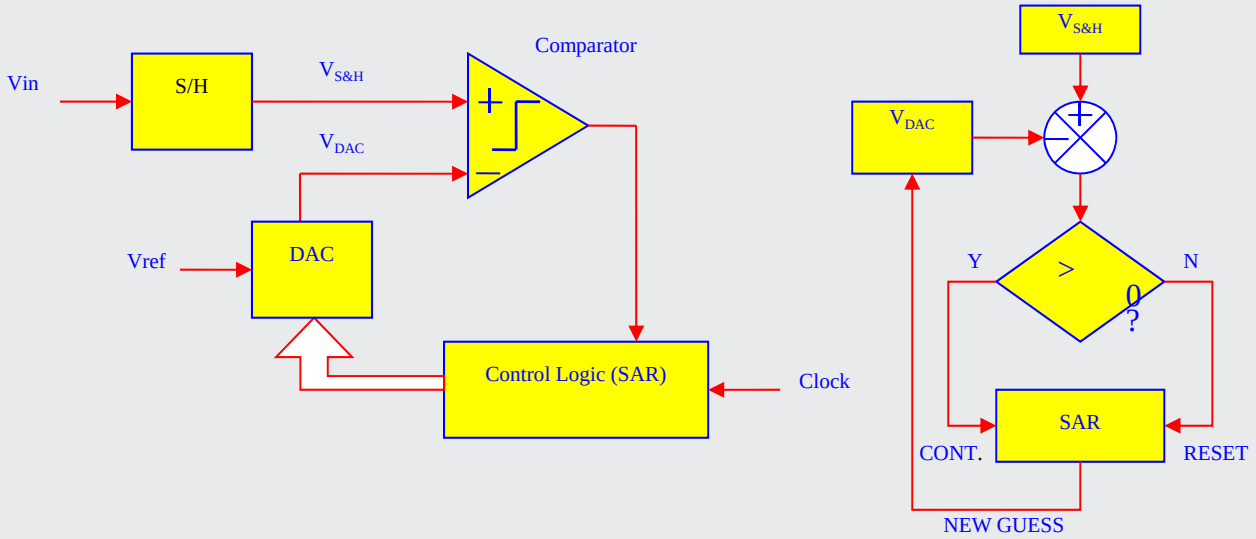


- SystemC-AMS is based on C/C++ language. It complements circuit level simulation for critical analog mixed signal or RF circuitry (e.g. using Spice), as well as behavioral modeling, such as Verilog-AMS or VHDL-AMS.
- Although not yet approved by SystemC initiative (OSCI), an implementation is currently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

Operational Characterization of AMS IPs

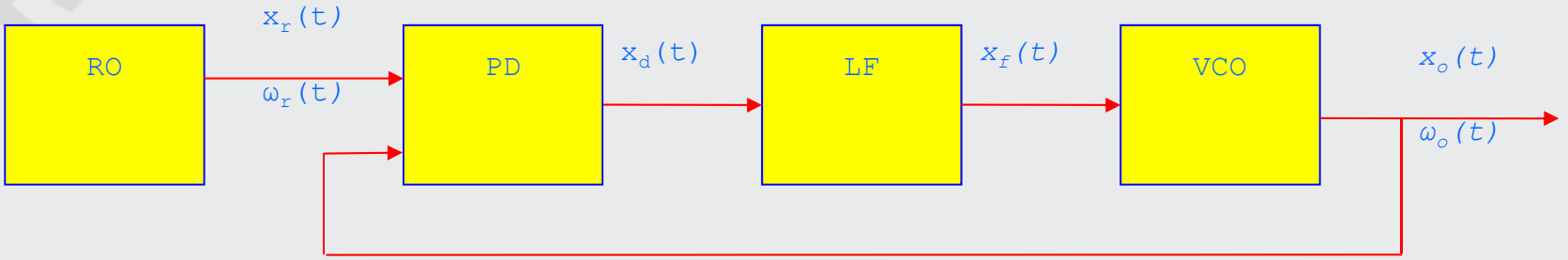
- Design methodology
  - ✓ develop behavioral models of analog mixed signal IPs using SystemC-AMS,
  - ✓ system-level simulation, validation and sensitivity analysis that takes into account both technology variation and component aging,
  - ✓ Example: two case studies:
    - o phase locked loops (PLL), and
    - o analog-to-digital data converter (ADC).
- Provided preliminary macroscopic monitoring metrics (eventually tp be linked to electrical parameters) to characterize the operational range of AMS component.

Example: ADC Architecture



- An ADC provides an output that digitally represents the input voltage or current.
- Applications include wireless networks and audio/video, e.g. MP3, DVD, HDTV.
- Common architectures range in size, complexity accuracy and/or resolution.
- Chose a linear ADC/SAR implementation due to low complexity (binary search).
- Developed system-level models of ADC/SAR with process variation in SystemC-AMS.
- Performed functional validation and sensitivity analysis.
- This will eventually lead to full operational characterization of the ADC.

Example: PLL Architecture



- A PLL provides phase synchronization between an output signal generated by a local oscillator and a reference signal by controlling the VCO frequency
- The PLL is used in various applications, e.g. frequency synthesis, clock signal recovery, signal synchronization, coherent demodulation and noise reduction
- The PLL consists of three basic blocks: a phase detector (PD), a loop filter (LF) and a voltage controlled oscillator (VCO)
- Develop system-level models of a linear PLL with process variation (nonlinearities and thermal or gaussian noise due to aging) in SystemC-AMS
- Perform validation and sensitivity analysis ⇒ full operational characterization

Tradeoffs in Operational Characterization

- Important accuracy and delay tradeoffs in detecting operational failures of AMS IP, due to process variation and noise effects. For example, to detect a noisy sinusoidal wave we can use
  - ✓ multiharmonic analysis of variance,
  - ✓ Scargle's sine method,
  - ✓ regression statistics (e.g. least-squares sine fitting), and
  - ✓ analysis of variance for phase bins.
- Similarly to make phase drift computations insensitive to small frequency drifts, *we can* avoid zero-crossing estimation and resort to
  - ✓ regression statistics (e.g. linear or higher order interpolation),
  - ✓ averaging of values, or
  - ✓ application of identical band pass filters.
- Complex computations with a higher sampling rate (to enhance accuracy and meet Nyquist frequency constraints) increase system latency, possibly exceeding real-time requirements.

End of Presentation.

Thank you.