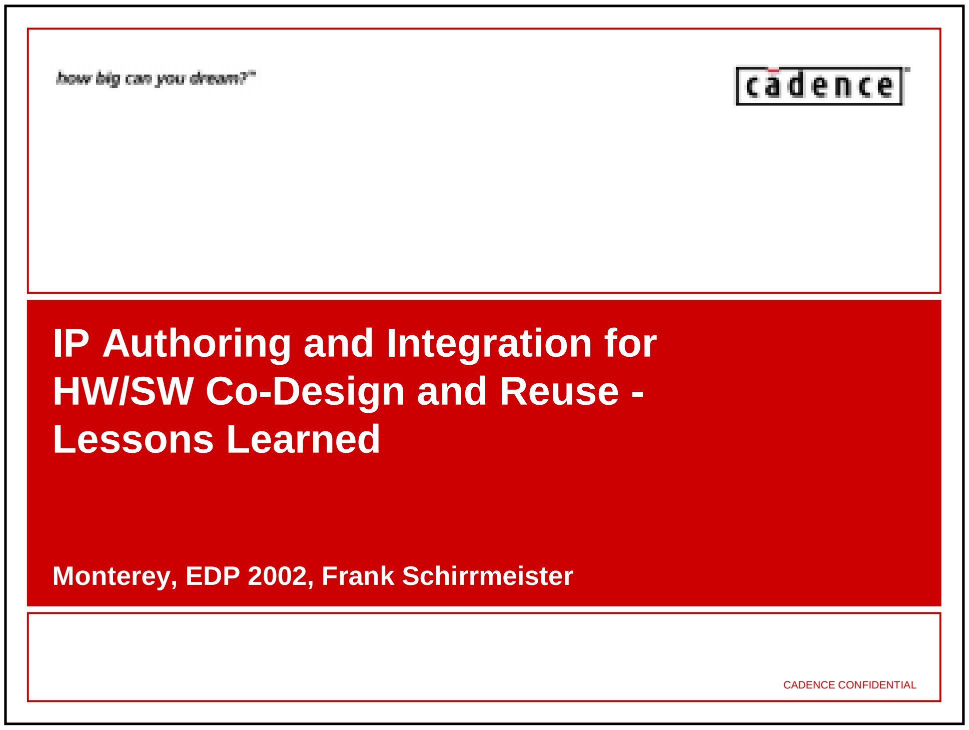
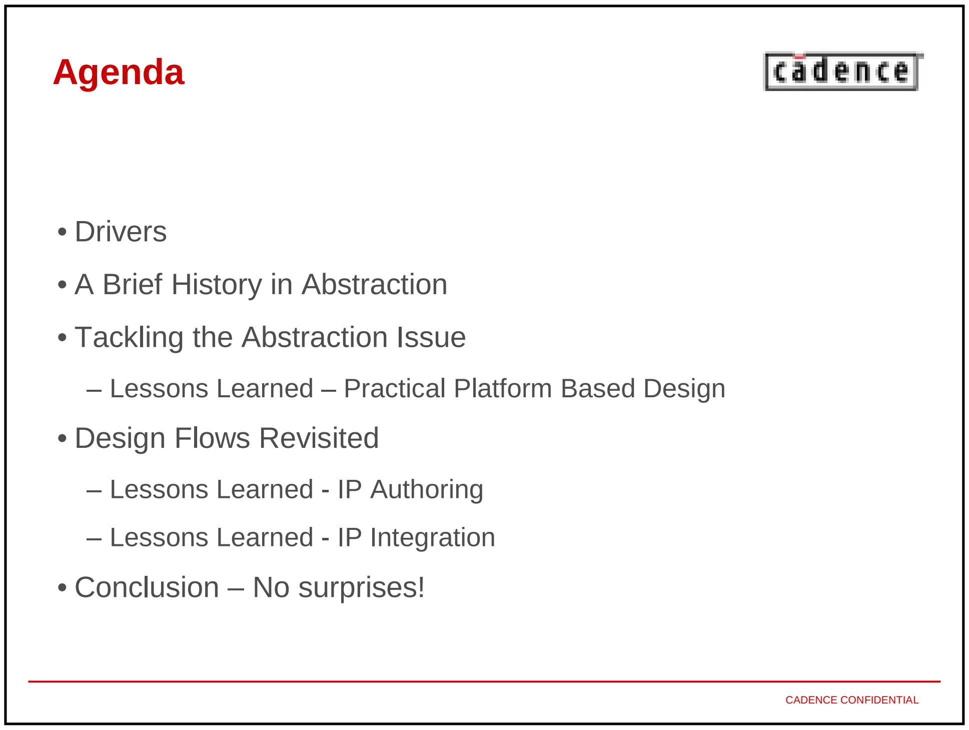


how big can you dream?™

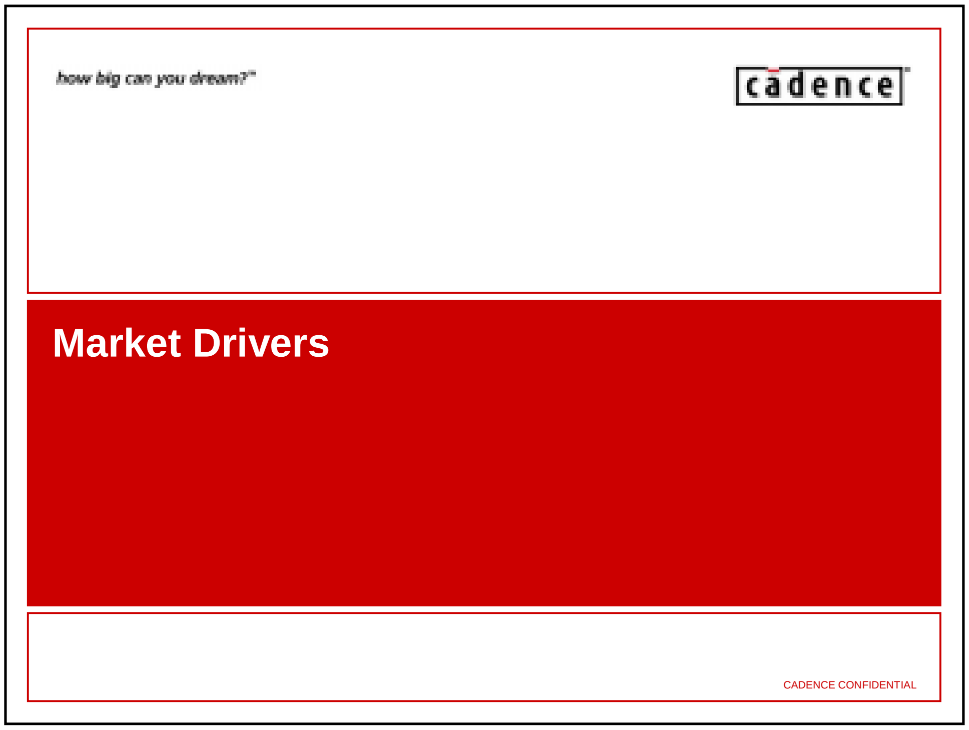
IP Authoring and Integration for HW/SW Co-Design and Reuse - Lessons Learned

Monterey, EDP 2002, Frank Schirrmeister



Agenda

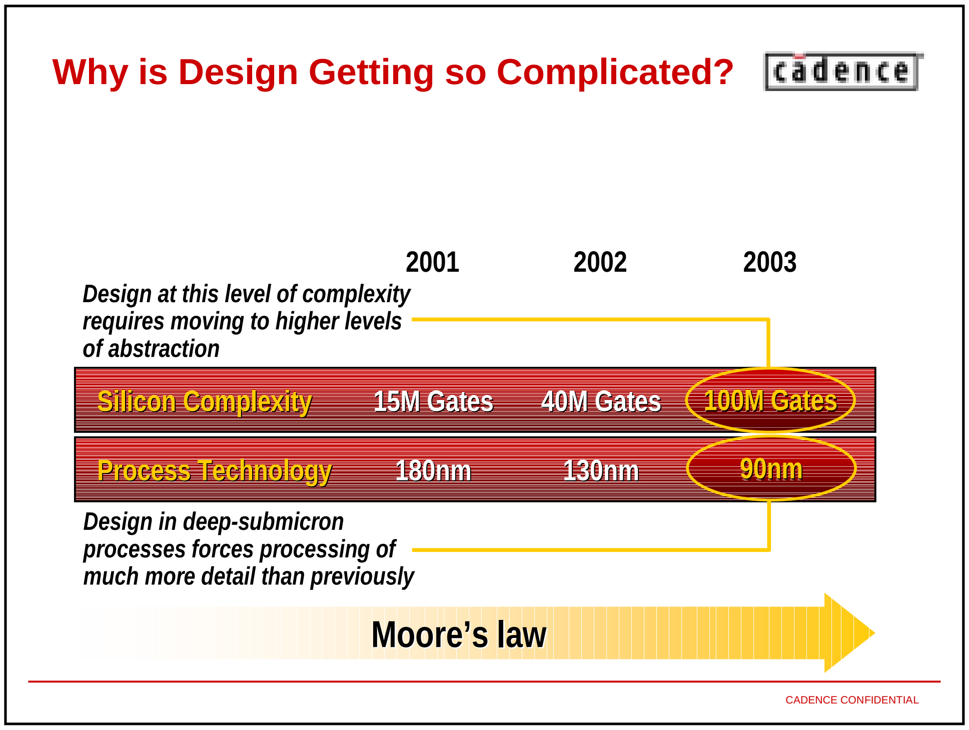
- Drivers
- A Brief History in Abstraction
- Tackling the Abstraction Issue
  - Lessons Learned – Practical Platform Based Design
- Design Flows Revisited
  - Lessons Learned - IP Authoring
  - Lessons Learned - IP Integration
- Conclusion – No surprises!



how big can you dream?℠

Market Drivers

CADENCE CONFIDENTIAL



Why is Design Getting so Complicated?

	2001	2002	2003
Silicon Complexity	15M Gates	40M Gates	100M Gates
Process Technology	180nm	130nm	90nm

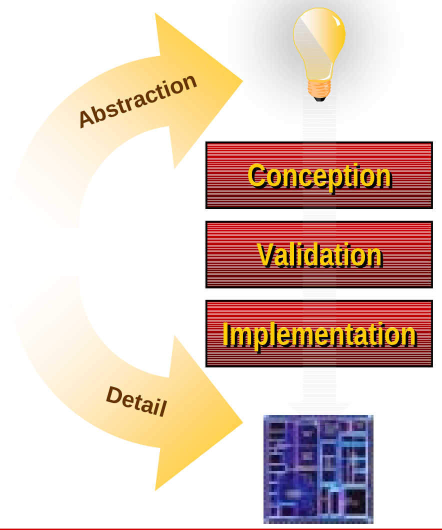
Design at this level of complexity requires moving to higher levels of abstraction

Design in deep-submicron processes forces processing of much more detail than previously

Moore's law

CADENCE CONFIDENTIAL

Why is Design Getting so Complicated?



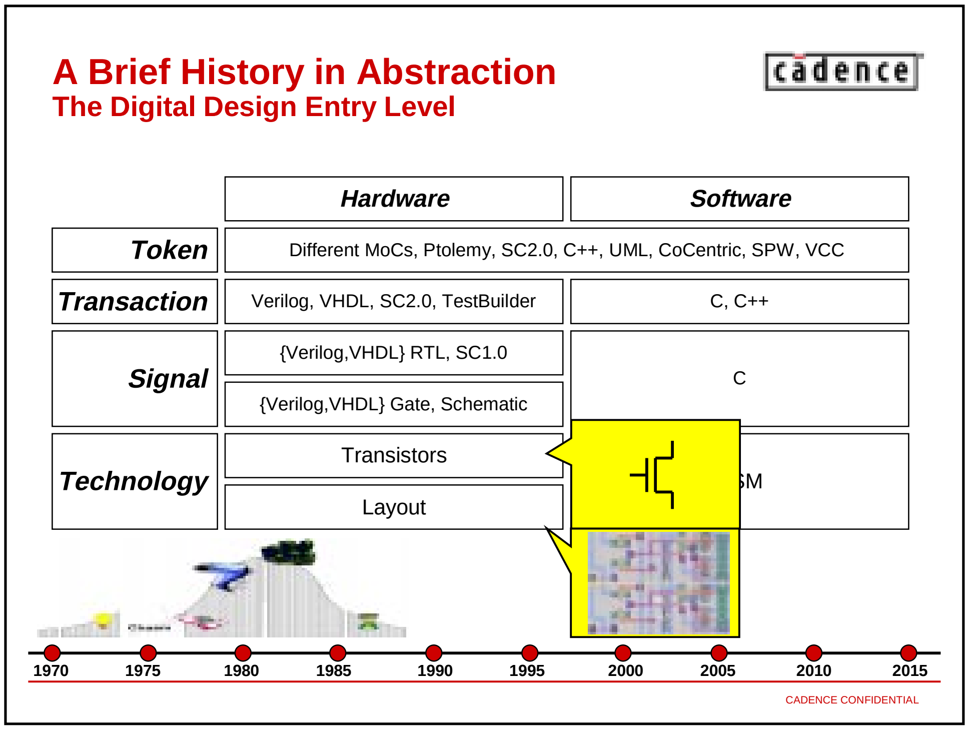
Design at this level of complexity requires moving to higher levels of abstraction

These two effects work against each other

Design in deep-submicron processes forces processing of much more detail than previously

how big can you dream?™

A Brief History in Abstraction

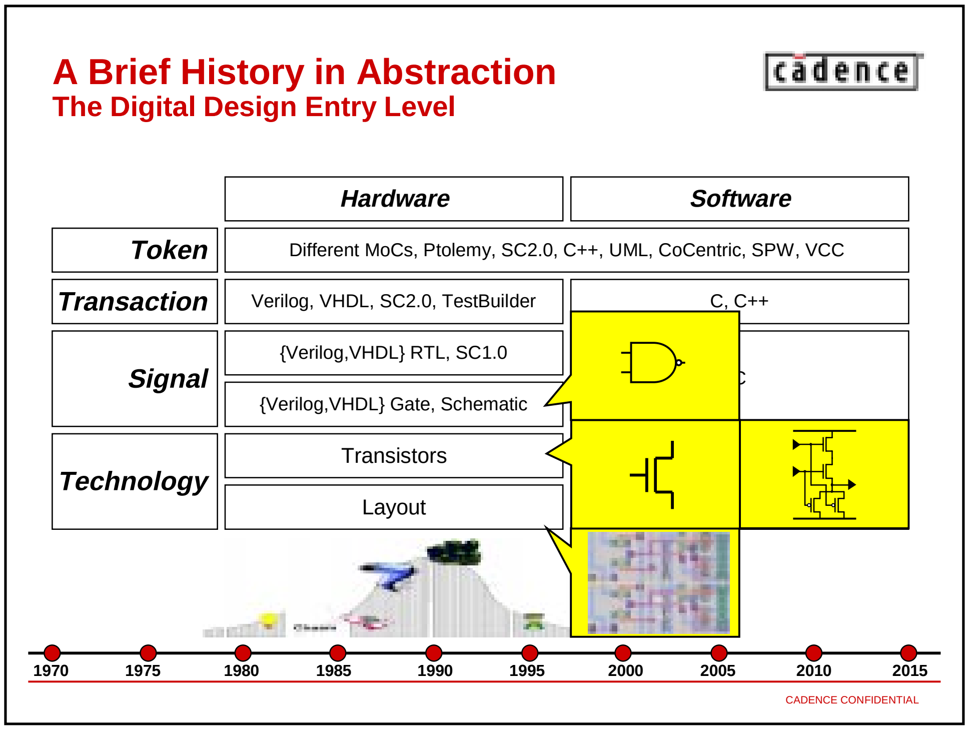


A Brief History in Abstraction
The Digital Design Entry Level

	Hardware	Software
Token	Different MoCs, Ptolemy, SC2.0, C++, UML, CoCentric, SPW, VCC	
Transaction	Verilog, VHDL, SC2.0, TestBuilder	C, C++
Signal	{Verilog,VHDL} RTL, SC1.0	C
	{Verilog,VHDL} Gate, Schematic	
Technology	Transistors	SM
	Layout	

1970 1975 1980 1985 1990 1995 2000 2005 2010 2015

CADENCE CONFIDENTIAL

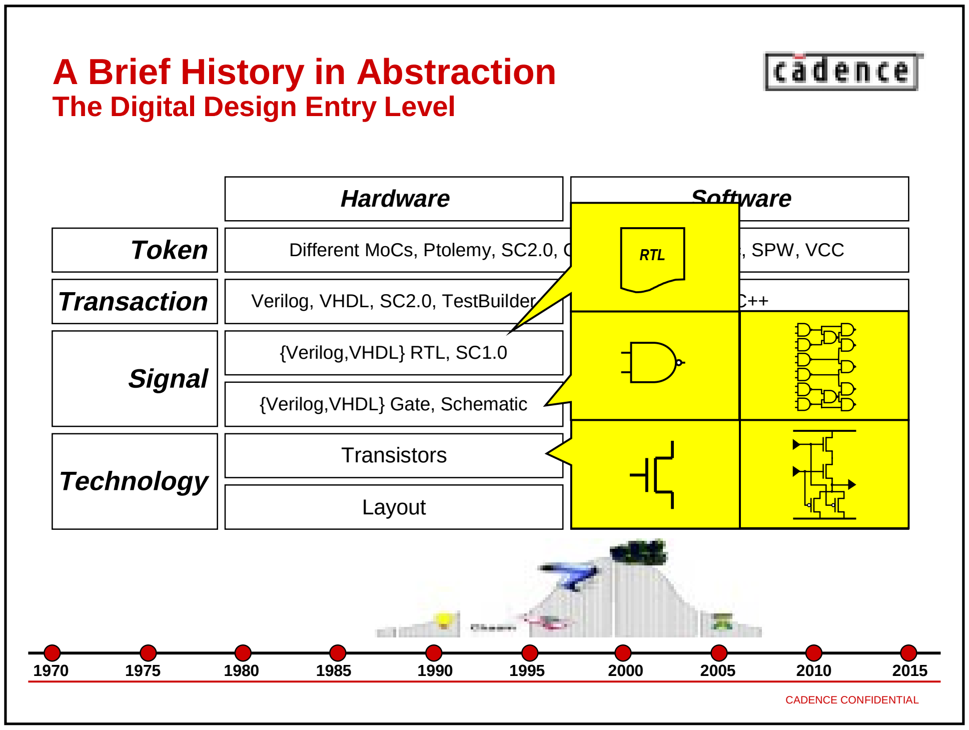


A Brief History in Abstraction
The Digital Design Entry Level

	Hardware	Software
Token	Different MoCs, Ptolemy, SC2.0, C++, UML, CoCentric, SPW, VCC	
Transaction	Verilog, VHDL, SC2.0, TestBuilder	C, C++
Signal	{Verilog,VHDL} RTL, SC1.0	C
	{Verilog,VHDL} Gate, Schematic	
Technology	Transistors	
	Layout	

1970 1975 1980 1985 1990 1995 2000 2005 2010 2015

CADENCE CONFIDENTIAL

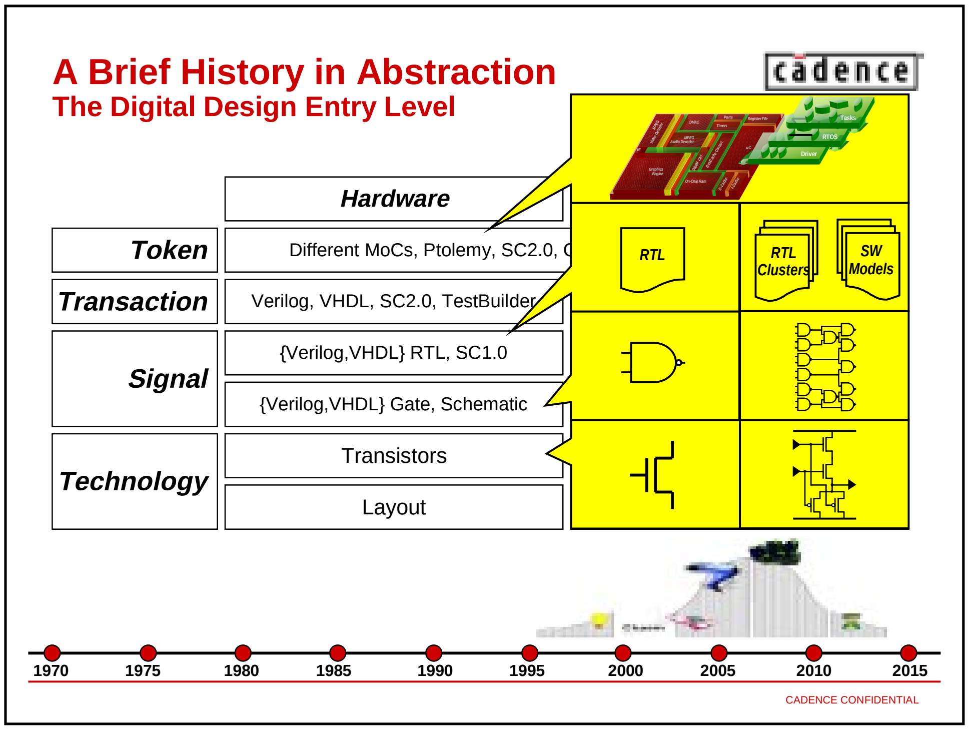


A Brief History in Abstraction

The Digital Design Entry Level

	Hardware	Software
Token	Different MoCs, Ptolemy, SC2.0, C	, SPW, VCC
Transaction	Verilog, VHDL, SC2.0, TestBuilder	C++
Signal	{Verilog,VHDL} RTL, SC1.0	
	{Verilog,VHDL} Gate, Schematic	
Technology	Transistors	
	Layout	

CADENCE CONFIDENTIAL

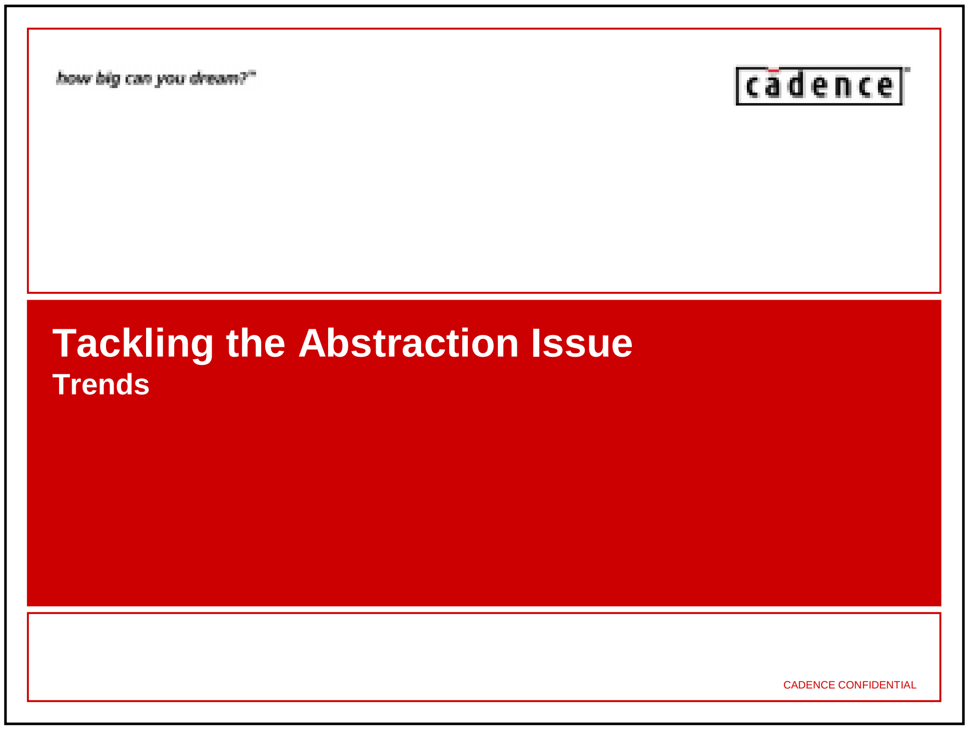


A Brief History in Abstraction

The Digital Design Entry Level

	Hardware
Token	Different MoCs, Ptolemy, SC2.0, C
Transaction	Verilog, VHDL, SC2.0, TestBuilder
Signal	{Verilog,VHDL} RTL, SC1.0
	{Verilog,VHDL} Gate, Schematic
Technology	Transistors
	Layout

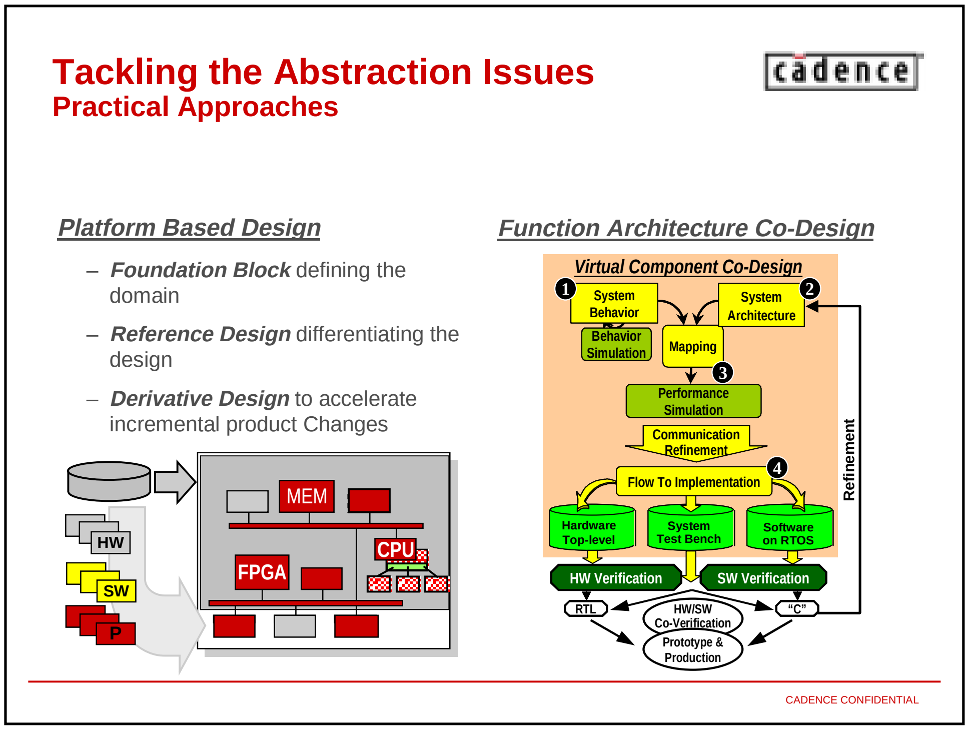
CADENCE CONFIDENTIAL



how big can you dream?™

Tackling the Abstraction Issue

Trends

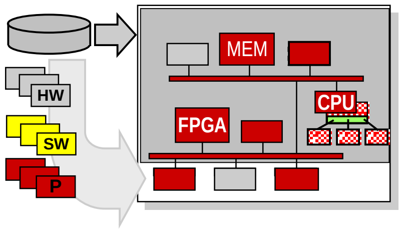


Tackling the Abstraction Issues

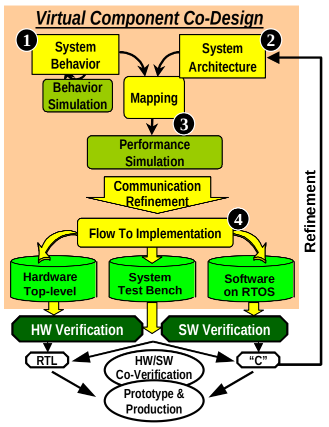
Practical Approaches

Platform Based Design

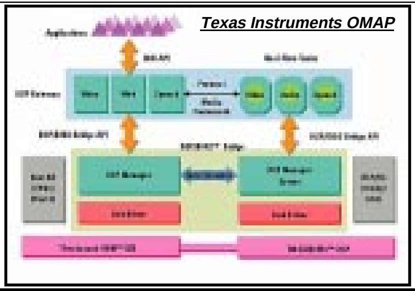
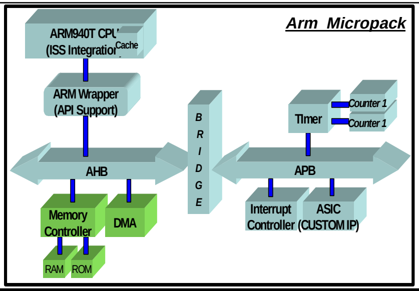
- ***Foundation Block*** defining the domain
- ***Reference Design*** differentiating the design
- ***Derivative Design*** to accelerate incremental product Changes



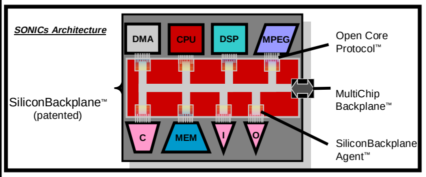
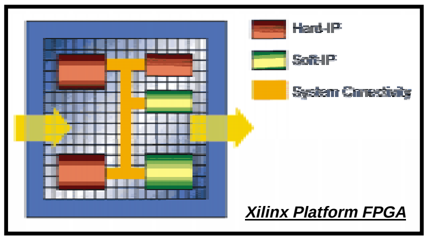
Function Architecture Co-Design



Lessons Learned …
Platform Type Examples

“Full Application HW/SW Platform”	*“Processor Centric Platform”*
Examples: –TI OMAP –Philips nExperia, –Infineon MGold	Examples: – ARM Micropack – ST100 Platform – Motorola Starcore
	 Arm Micropack ARM940T CPU (ISS Integration) Cache ARM Wrapper (API Support) AHB Memory Controller DMA RAM ROM BRIDGE APB Timer Counter 1 Counter 1 Interrupt Controller ASIC (CUSTOM IP)

Lessons Learned …
Platform Type Examples

“Communication Centric Platform”	*“Highly Programmable Platform”*
Examples: – Palmchip – Sonics	Examples –Triscend A7 –Chameleon –Altera Excalibur –Xilinx Platform FPGA
SONICs Architecture  SONICs Architecture DMA CPU DSP MPEG SiliconBackplane™ (patented) Open Core Protocol™ MultiChip Backplane™ SiliconBackplane Agent™ C MEM I O	 Hard-IP Soft-IP System Connectivity Xilinx Platform FPGA

Lessons Learned
Platform User Types / Hand Off Points

"Power User"
- differentiates at all levels – software and hardware
- Develops additional custom hardware and software components

"Platform Differentiator"
- differentiates at the application level
- develops processor Application Software
- Uses existing libraries as hardware accelerators

"Complete Package User"
- expects complete solution (hardware and software)
- limited additional development and differentiations

CADENCE CONFIDENTIAL

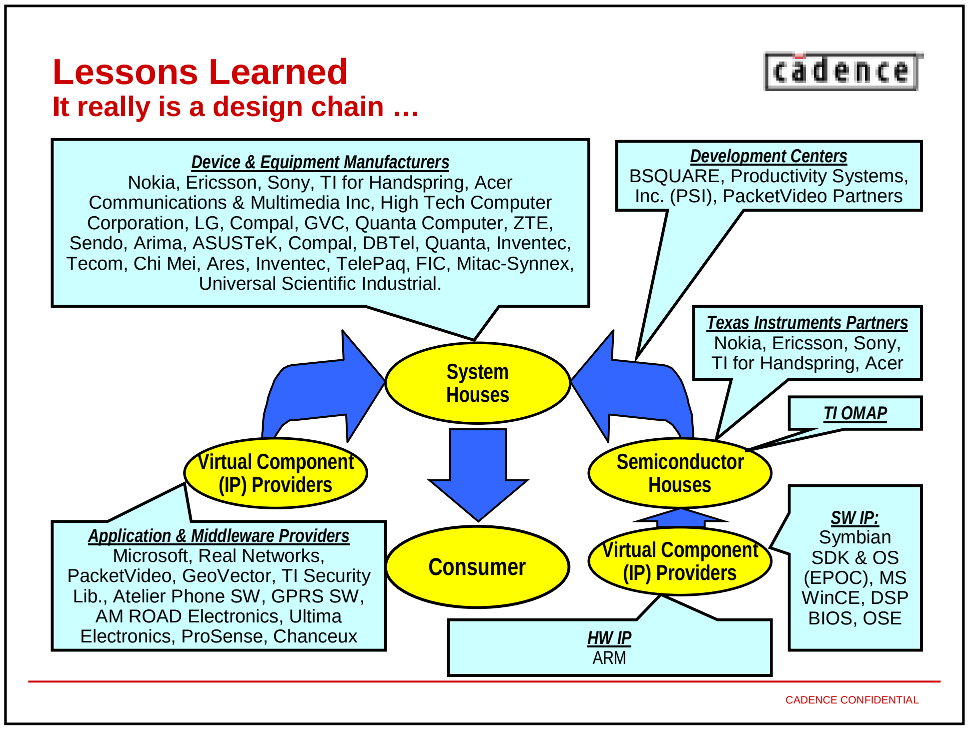
Lessons Learned
Return on Investment Considerations

How to assess ROI of new tools and methodologies?
- How many man month does it save?
- How many new engineers does the organization not have to hire?
- How much faster will the product go out the door? Value per day?
- How much better will the quality of results be?

Semiconductor Platform Example
- Reduce the number of project years required for a fast derivative from the platform

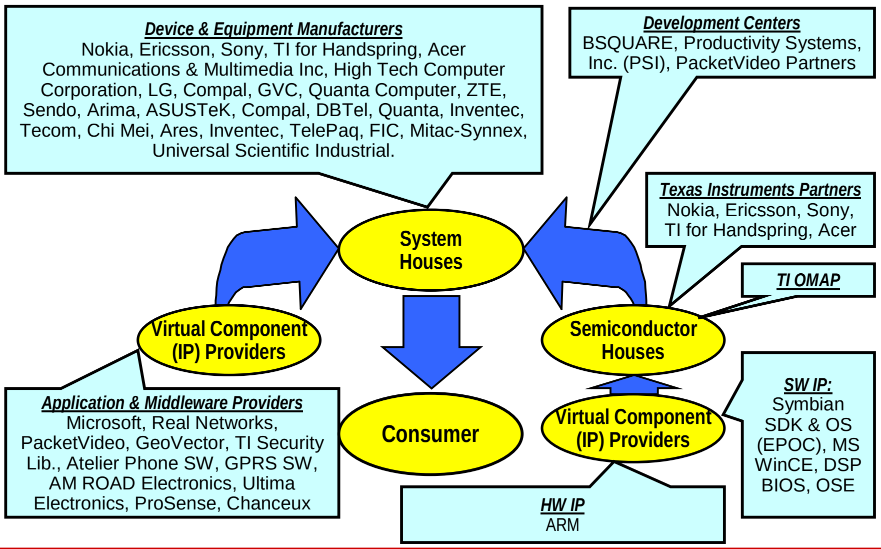
Example Assumption
- Reduction of Effort for derivative design from 30 man years to 10 man years
- 15 derivative designs … result in 15x20=300 man years of cost reduction.

CADENCE CONFIDENTIAL



Lessons Learned

It really is a design chain …



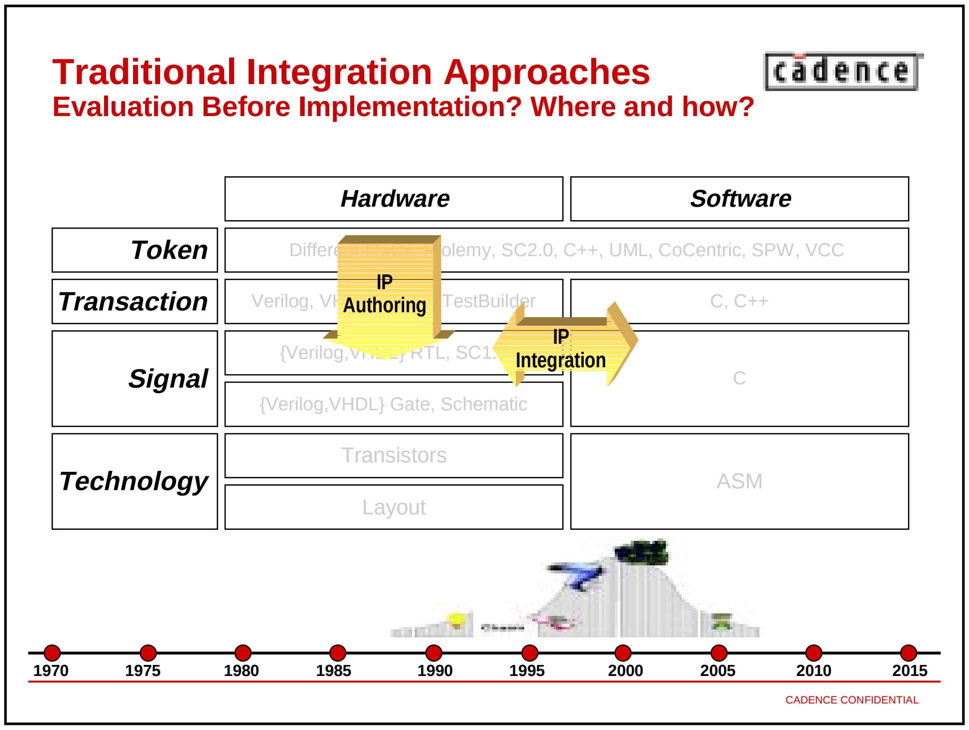
Device & Equipment Manufacturers
Nokia, Ericsson, Sony, TI for Handspring, Acer Communications & Multimedia Inc, High Tech Computer Corporation, LG, Compal, GVC, Quanta Computer, ZTE, Sendo, Arima, ASUSTeK, Compal, DBTel, Quanta, Inventec, Tecom, Chi Mei, Ares, Inventec, TelePaq, FIC, Mitac-Synnex, Universal Scientific Industrial.
Development Centers
BSQUARE, Productivity Systems, Inc. (PSI), PacketVideo Partners
Texas Instruments Partners
Nokia, Ericsson, Sony, TI for Handspring, Acer
TI OMAP
System Houses
Virtual Component (IP) Providers
Semiconductor Houses
Consumer
Virtual Component (IP) Providers
Application & Middleware Providers
Microsoft, Real Networks, PacketVideo, GeoVector, TI Security Lib., Atelier Phone SW, GPRS SW, AM ROAD Electronics, Ultima Electronics, ProSense, Chanceux
SW IP:
Symbian SDK & OS (EPOC), MS WinCE, DSP BIOS, OSE
HW IP
ARM

CADENCE CONFIDENTIAL

how big can you dream?™

Design Flows Revisited

CADENCE CONFIDENTIAL



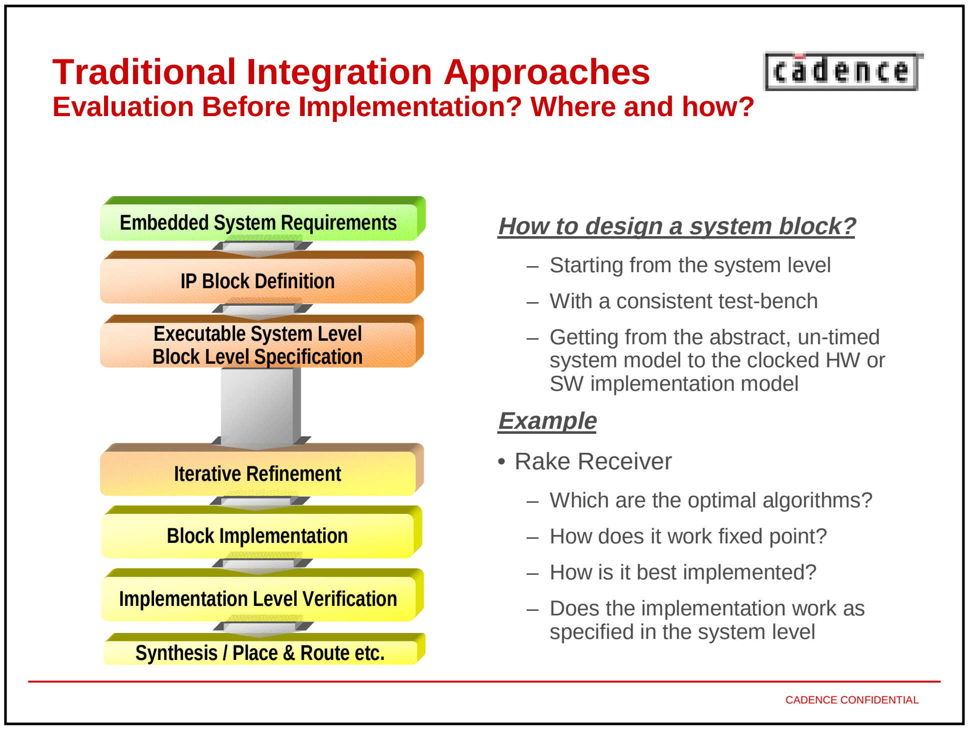
Traditional Integration Approaches

Evaluation Before Implementation? Where and how?

	Hardware	Software
Token	Differe olemy, SC2.0, C++, UML, CoCentric, SPW, VCC	
Transaction	Verilog, VH TestBuilder	C, C++
Signal	{Verilog,VH } RTL, SC1	C
	{Verilog,VHDL} Gate, Schematic	
Technology	Transistors	ASM
	Layout	

IP Authoring
IP Integration
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015

CADENCE CONFIDENTIAL



Traditional Integration Approaches

Evaluation Before Implementation? Where and how?

Embedded System Requirements
IP Block Definition
Executable System Level Block Level Specification
Iterative Refinement
Block Implementation
Implementation Level Verification
Synthesis / Place & Route etc.

How to design a system block?

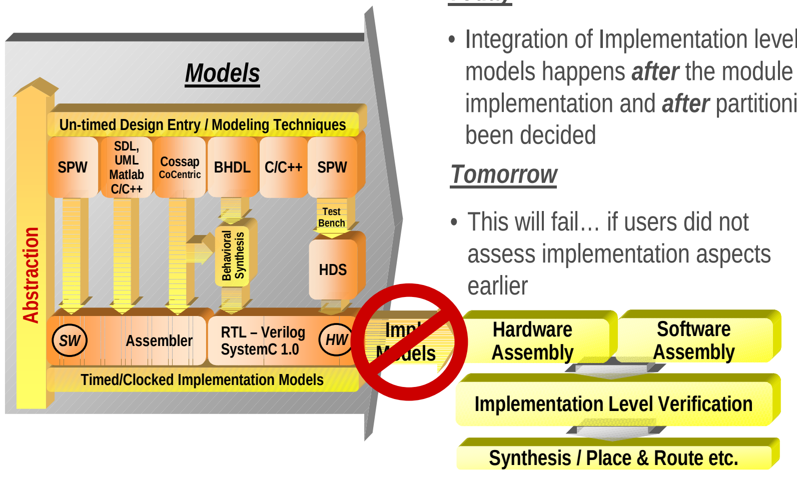
- Starting from the system level
- With a consistent test-bench
- Getting from the abstract, un-timed system model to the clocked HW or SW implementation model

Example

- Rake Receiver
  - Which are the optimal algorithms?
  - How does it work fixed point?
  - How is it best implemented?
  - Does the implementation work as specified in the system level

CADENCE CONFIDENTIAL

Traditional Integration Approaches
Evaluation Before Implementation? Where and how?


Models
Un-timed Design Entry / Modeling Techniques
SPW
SDL, UML Matlab C/C++
Cossap CoCentric
BHDL
C/C++
SPW
Test Bench
Behavioral Synthesis
HDS
Abstraction
SW
Assembler
RTL – Verilog SystemC 1.0
HW
Impl. Models
Timed/Clocked Implementation Models
Hardware Assembly
Software Assembly
Implementation Level Verification
Synthesis / Place & Route etc.

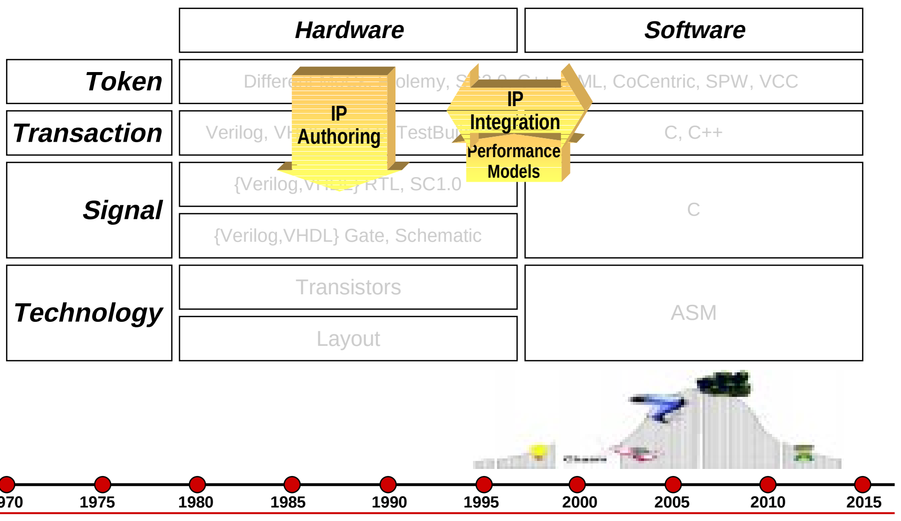
Today

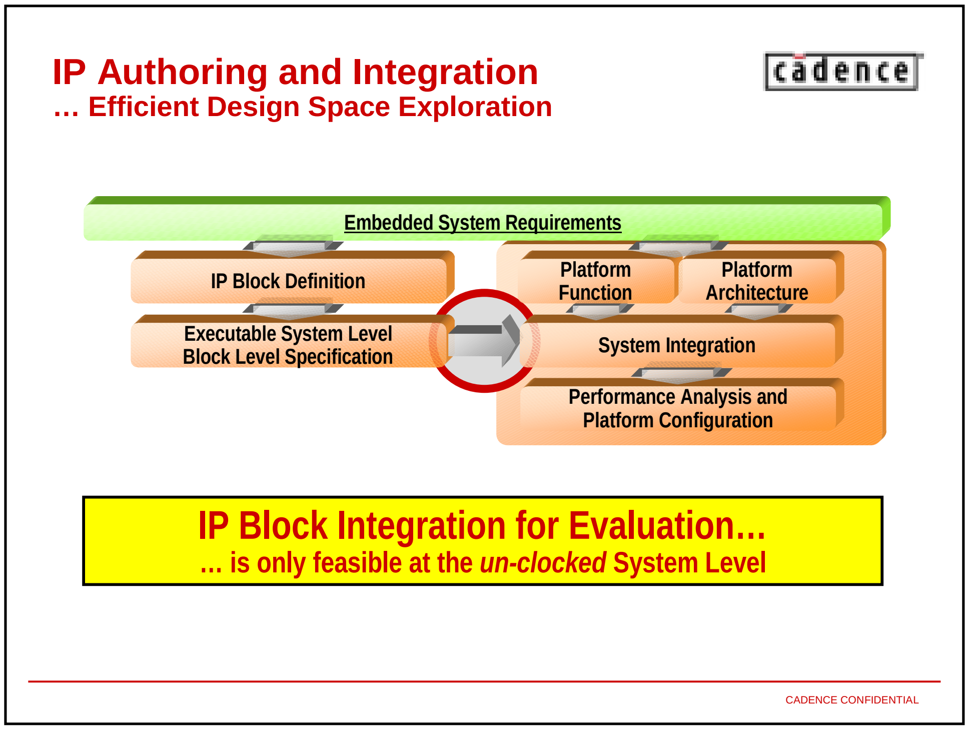
- Integration of Implementation level models happens ***after*** the module implementation and ***after*** partitioning has been decided

Tomorrow

- This will fail… if users did not assess implementation aspects earlier

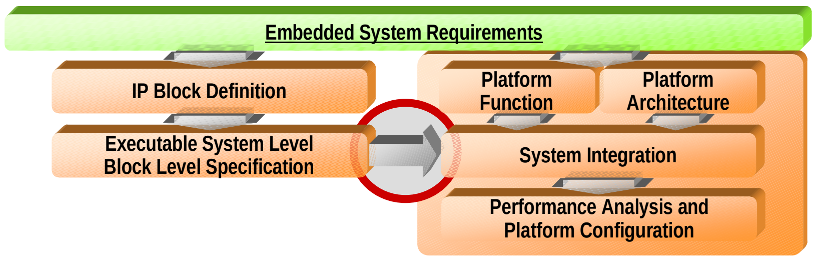
Traditional Integration Approaches
Evaluation Before Implementation? Where and how?





IP Authoring and Integration
… Efficient Design Space Exploration

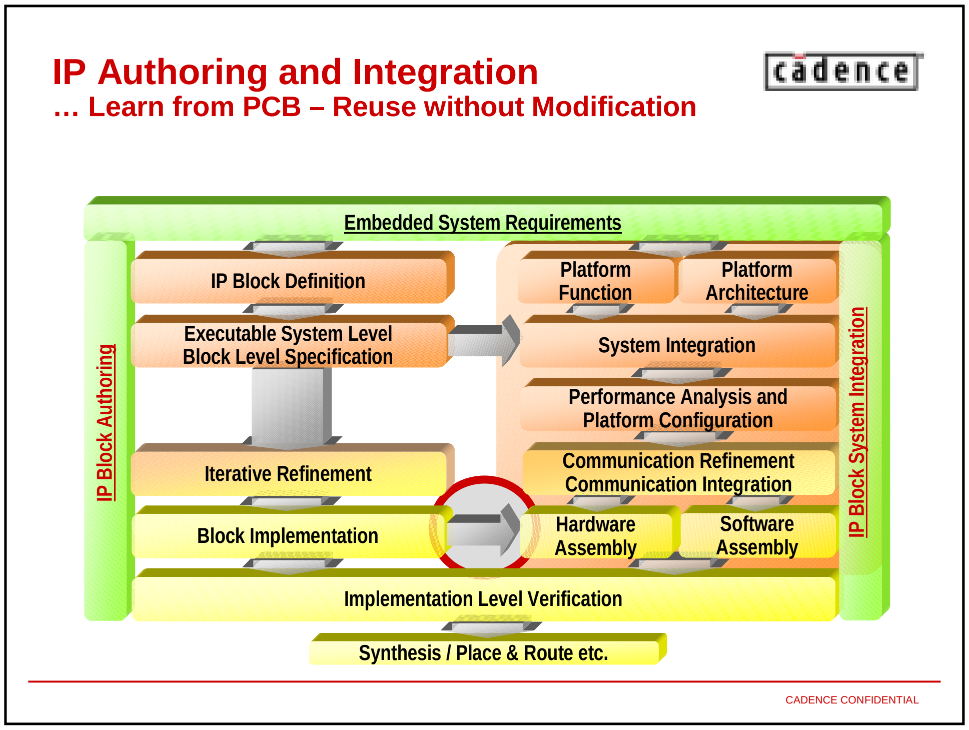
cadence



Embedded System Requirements
IP Block Definition
Executable System Level Block Level Specification
Platform Function
Platform Architecture
System Integration
Performance Analysis and Platform Configuration

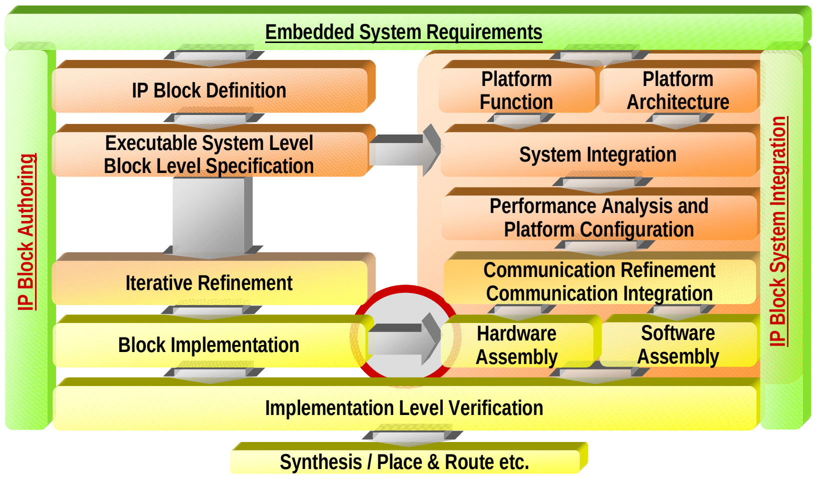
IP Block Integration for Evaluation…
… is only feasible at the *un-clocked* System Level

CADENCE CONFIDENTIAL



IP Authoring and Integration
… Learn from PCB – Reuse without Modification

cadence



Embedded System Requirements
IP Block Authoring
IP Block Definition
Executable System Level Block Level Specification
Iterative Refinement
Block Implementation
Platform Function
Platform Architecture
System Integration
Performance Analysis and Platform Configuration
Communication Refinement Communication Integration
Hardware Assembly
Software Assembly
IP Block System Integration
Implementation Level Verification
Synthesis / Place & Route etc.

CADENCE CONFIDENTIAL

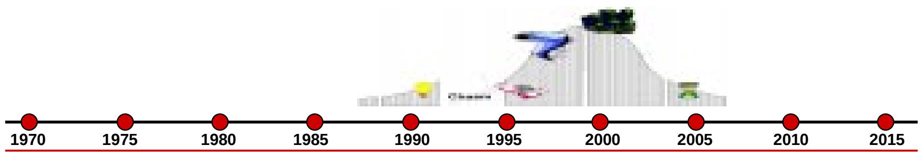
how big can you dream?℠

IP Authoring Use Models

Lessons Learned

IP Authoring Use Models – Research Phase

	Hardware	*Software*
Token	Different MoCs System Modeling [illegible]	[illegible]c, SPW, VCC
Transaction	Verilog, VHDL, SC2.0, TestBuilder	C, C++
Signal	{Verilog,VHDL} RTL, SC1.0	C
	{Verilog,VHDL} Gate, Schematic	
Technology	Transistors	ASM
	Layout	



Lessons Learned
IP Authoring Use Models – Research Phase

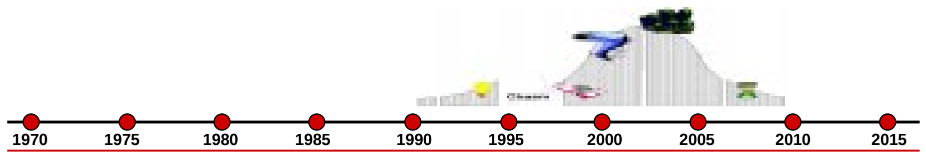
Research Phase

- Scientists play with their ideas
  - on whiteboards, papers and create computer model
  - prove their ideas would work against a certain physical environment
- IP Authoring
  - actually experiment with their preferred algorithms
  - analyze the results, plot and document their findings in papers
  - ease of use, flexible authoring paradigms like to capture of control, dataflow and time continuous domains is of essence
  - experiments are mostly done in the floating domain
    - implementation is not a concern at all yet.

Lessons Learned
IP Authoring Use Models – Top Down

IP Authoring

	Hardware	***Software***
Token	Differe olemy, SC2.0, C++, UML, CoCentric, SPW, VCC	
Transaction	Verilog, VH TestBuilder	C, C++
Signal	{Verilog,VH RTL, SC1.0	C
	{Verilog,VHDL} Gate, Schematic	
Technology	Transistors	ASM
	Layout	



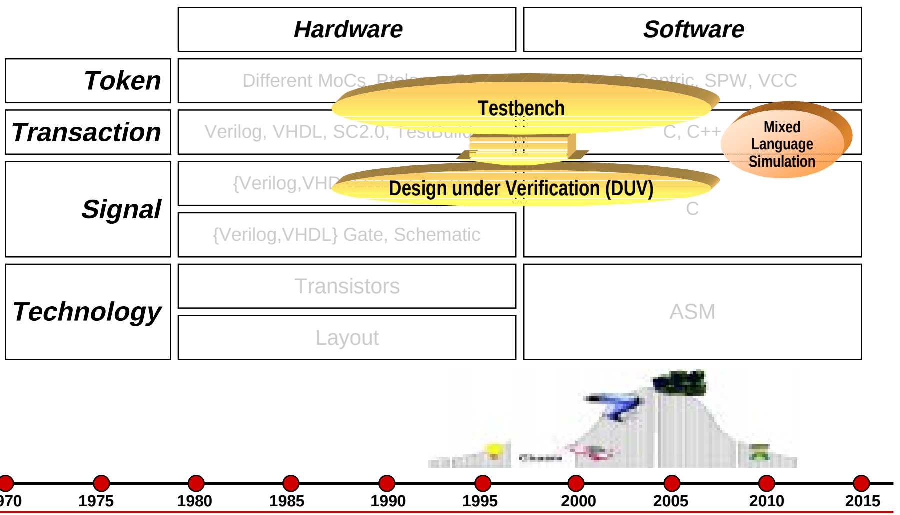
Lessons Learned
IP Authoring Use Models – Top Down

Top Down Product Development

- New applications get visible
  - Development teams start working building reference models, introducing real channel effects and capture the performance data of a system which should resemble the standard
  - The teams are building executable specifications, and bring on their own ideas for algorithms to address specific problems
- IP Authoring
  - Cadence SPW is widely used in the communications and multimedia industry to capture those simulation models, in fact, Cadence and partners like NIST [22] do actually also provide those models
  - Getting a reference library gives a huge productivity boost and makes it easy for design teams to deliver on time [23].

CADENCE CONFIDENTIAL

Lessons Learned
IP Authoring Use Models – Verification/Reuse



CADENCE CONFIDENTIAL

Lessons Learned

IP Authoring Use Models – Verification/Reuse

Product Development Applying Reuse

- Graphical, hierarchical parametrizable models written in C, C++, SystemC, UML or behavioral analog languages allow for the best modeling styles selected for the task in mind
- IP Authoring
  - Simulators are required which can execute various simulation domains together without giving a performance hit to the user
  - support standard interfaces like OMI [24] and should allow a free mix of C/C++/SystemC/VHDL/Verilog/VerilogA as the Cadence SPW/NC-Sim environment [25] does
  - Reuse of previously generated HW blocks would not be a problem, and also the RF design team and the Baseband engineers can easily explore the performance of their end to end system before they integrate the first prototype in the labs

CADENCE CONFIDENTIAL

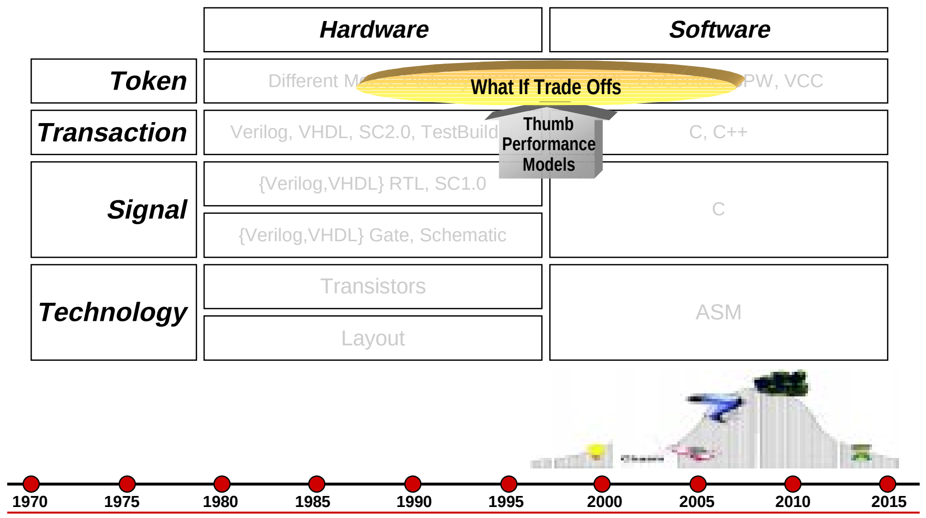
how big can you dream?™

IP Integration Use Models

CADENCE CONFIDENTIAL

Lessons Learned

IP Integration Use Models – Top Down Blank Sheet



CADENCE CONFIDENTIAL

Lessons Learned

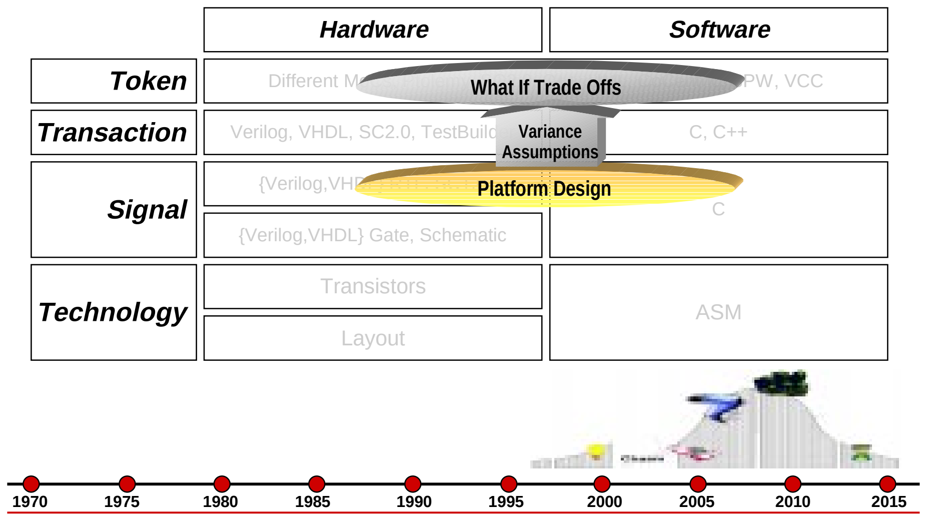
IP Integration Use Models – Top Down Blank Sheet

General Top Down Design Exploration

- value in the exploration how systems connect and behave on different target architecture options
- Ericsson reports in [11] their experience with HW/SW Co-Design and re-emphasizes the importance of separation of function and architecture
  - This enables much easier modification of designs at the system level and easier reuse of behavior and architecture virtual components.
  - The ability to make ‘what-if’ kinds of changes in architectures and mappings is important to build understanding of the system under design.
- BMW reported at various occasions including [21] about their design space exploration efforts using VCC.

CADENCE CONFIDENTIAL

Lessons Learned
IP Integration Use Models – Design the Platform



CADENCE CONFIDENTIAL

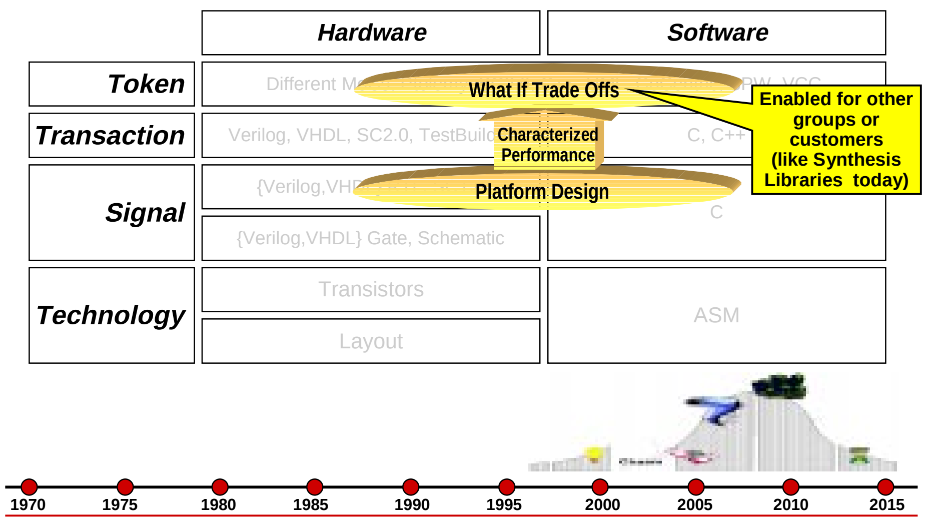
Lessons Learned
IP Integration Use Models – Design the Platform

Top Down Platform Exploration

- trade of decisions during the actual development of the platform
  - A system house will define a platform in which the architectural components are not yet bound to real implementations
  - This way trade offs between bus systems, memory hierarchies etc. can be analyzed and fed back to the development
- define a platform depending on the availability of architecture components (semiconductor house)
  - This use model is applied during the development of an actual platform
  - The challenging part here is to define the range of application variants in a appropriate way
  - This range then directly translates into the amount of scalability within a platform

CADENCE CONFIDENTIAL

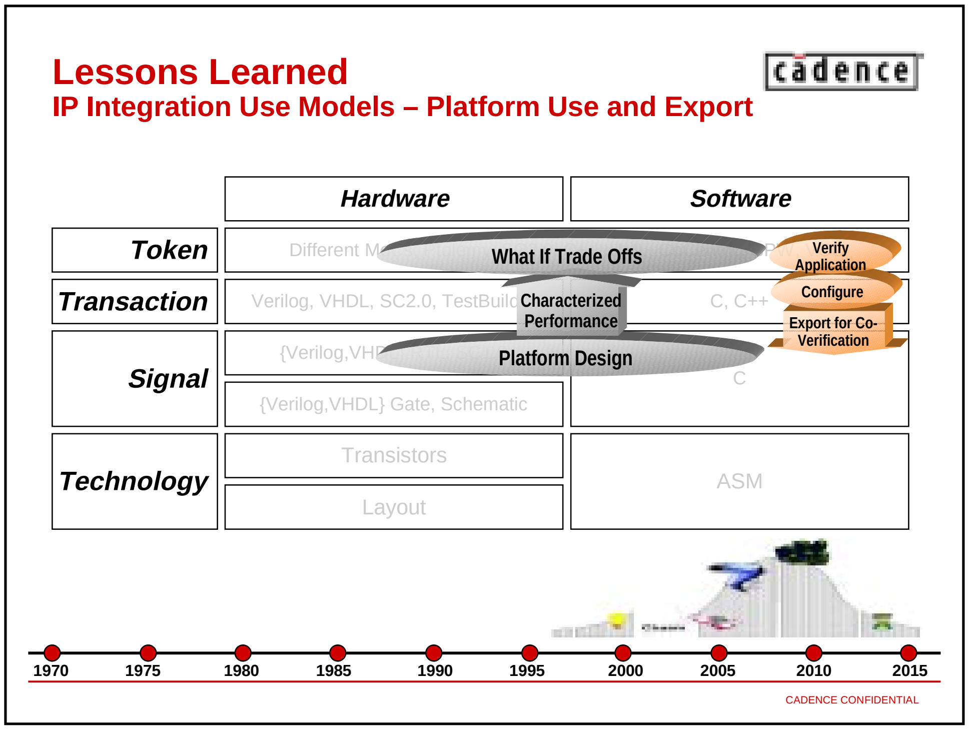
Lessons Learned
IP Integration Use Models – Platform Characterization



Lessons Learned
IP Integration Use Models – Platform Characterization

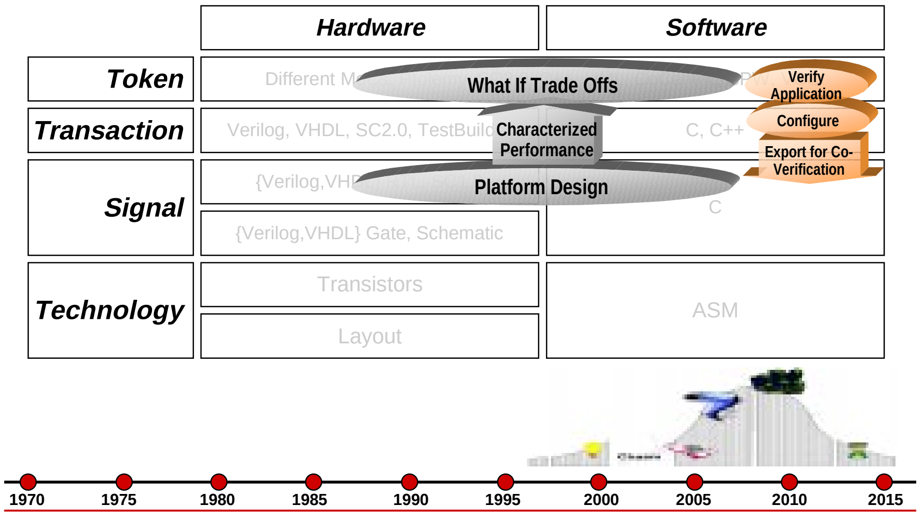
Bottom Up Platform Characterization

- Once a platform definition exists the different components can be characterized to reflect the implementation issues.
- Busses, RTOSs, processors etc. build the characterized components of a platform, which can be used by system houses as a target to map application variants to.
  - ST Microelectronics reported in [18] about efforts to provide and characterize architectural components for consumption by internal and external customers.
  - Philips reported in [12], [13] and [14] about efforts to characterize parts of a multimedia platform with a strong focus on the communication design.

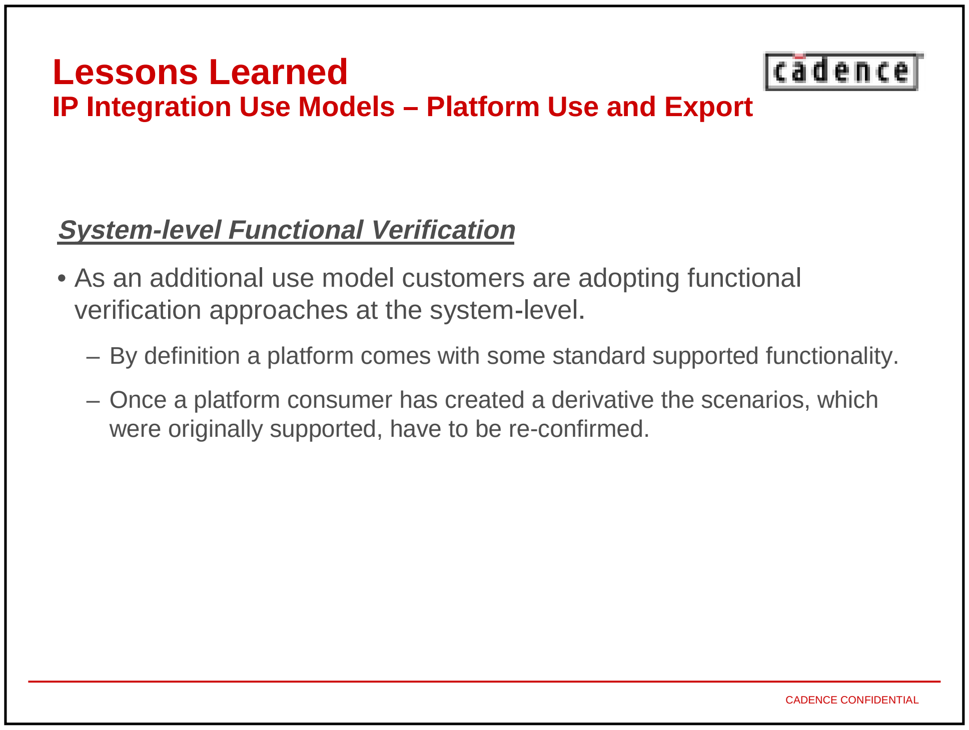


Lessons Learned

IP Integration Use Models – Platform Use and Export



CADENCE CONFIDENTIAL



Lessons Learned

IP Integration Use Models – Platform Use and Export

System-level Functional Verification

- As an additional use model customers are adopting functional verification approaches at the system-level.
  - By definition a platform comes with some standard supported functionality.
  - Once a platform consumer has created a derivative the scenarios, which were originally supported, have to be re-confirmed.

CADENCE CONFIDENTIAL

Lessons Learned
IP Integration Use Models – Platform Use and Export

Configuration of Characterized Platform

- Having received a characterized platform the system house can map application derivatives to the platform and explore different alternatives.
  - Magneti Marelli reports in [15] about their interaction with IP and architectural component providers.
  - Motorola describes in [20] a similar use model. The platform has here not been formerly characterized except through the system-level design team.
  - The BWRC also followed similar flows according to [17] during wireless protocol design.
  - Thomson CSF (now Thales) describes in [16] how characterization of IP components worked and how they verified the characterization itself.
- configuration of the platform is moved up to the system-level, at which simulation times are more appropriate for design space exploration.

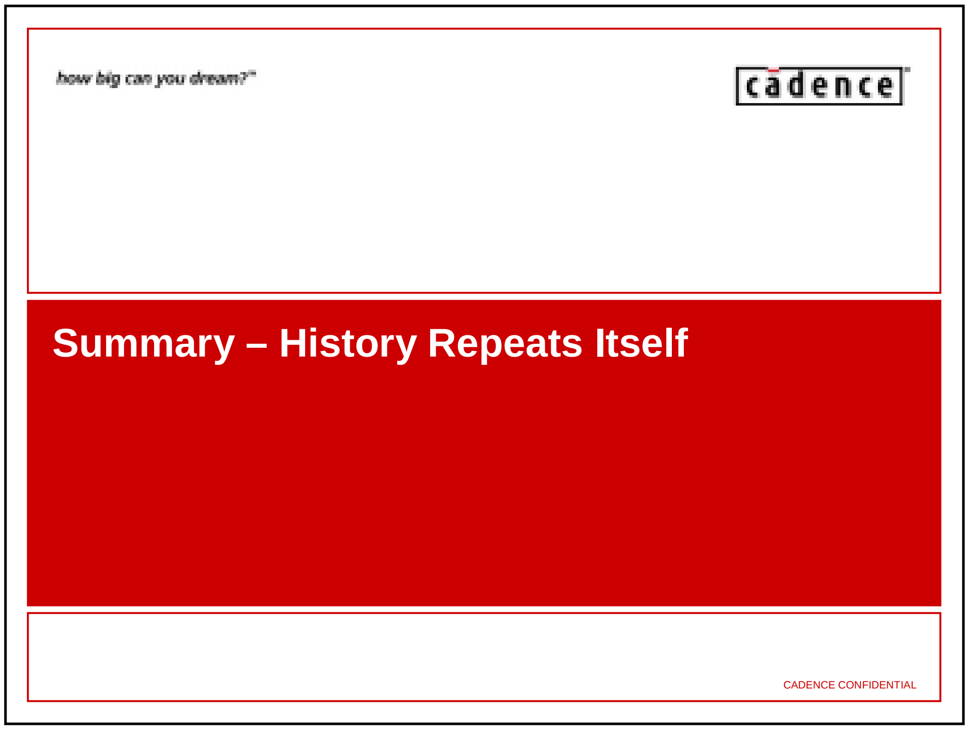
CADENCE CONFIDENTIAL

Lessons Learned
IP Integration Use Models – Platform Use and Export

Platform Assembly for Co-Verification

- Design export feeding Co-Verification environments like Yokogawa VirtualICE and Mentor Seamless.
- The assembly process in itself provides value as a front-end process to Co-Verification, in which the set up of the environment often is a cumbersome task.
  - refining the architectural platform to the state at which top-level netlists and the associated software can be exported

CADENCE CONFIDENTIAL



how big can you dream?™

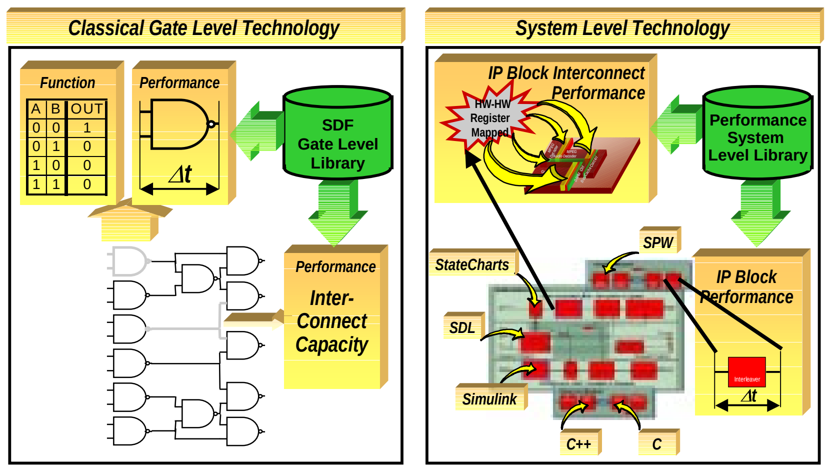
cadence

Summary – History Repeats Itself

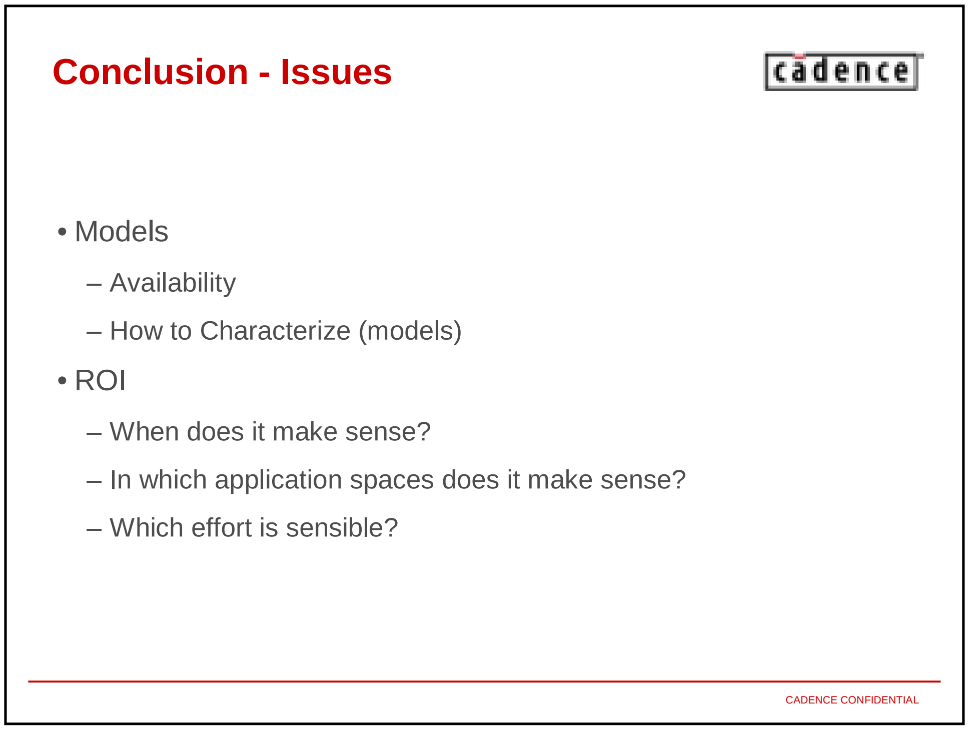
CADENCE CONFIDENTIAL

Performance Modeling …
… the System Level equivalent of SDF !

cadence



CADENCE CONFIDENTIAL



Conclusion - Issues

- Models
  - Availability
  - How to Characterize (models)
- ROI
  - When does it make sense?
  - In which application spaces does it make sense?
  - Which effort is sensible?