Virtually Solving Debug Challenges of the MPSoC Era

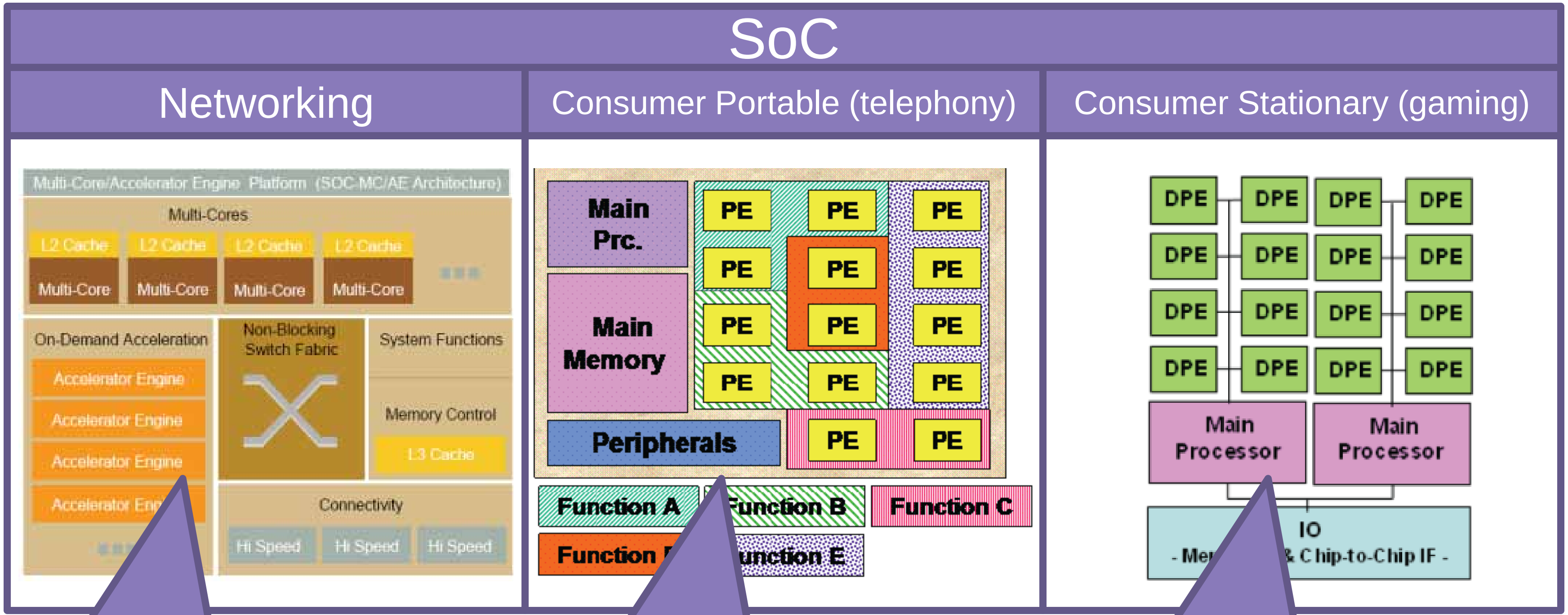
MPSoC 2008, Aachen

Kevin Smart, Senior Manager R&D
Frank Schirrmeister, Product Marketing Director
Filip Thoen, Solution Architect
Synopsys, Inc.

Executive Summary

- MPSoCs will have a daunting number of processors
- Software will be challenging, especially migration and debug
- Some debug examples
- Virtual platforms offer the solution
- SystemC TLM-2.0
- Modeling Style Examples
- Demo
- Wrap & Questions

ITRS Forecast



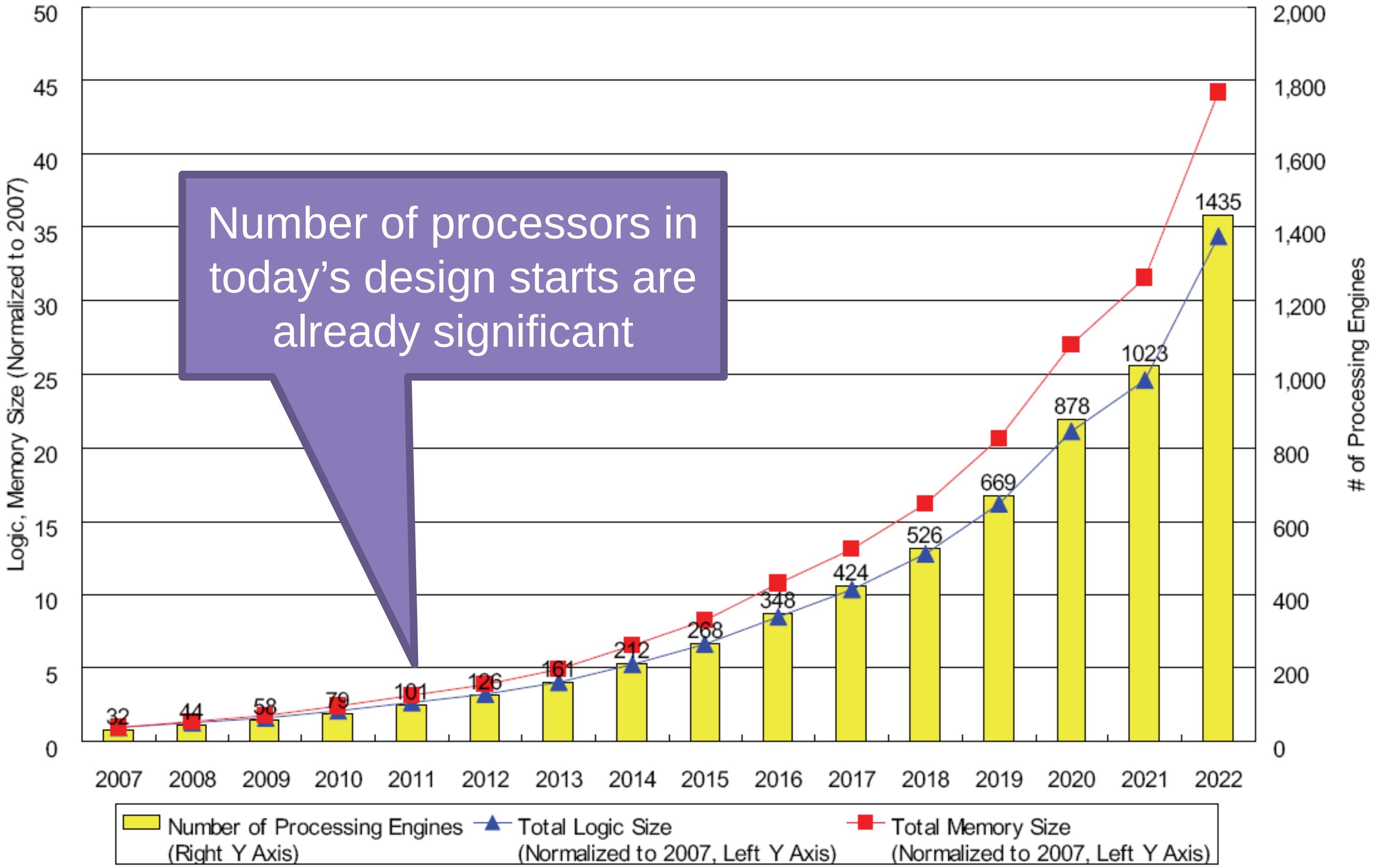
"networking needs no longer met by increasing operating frequencies…"
"geometric scaling…"
"unused programming cycles due to lack of visibility…"
"requires greater investment in simulation…"

Functions distributed across processors
"must minimize power…"
"short time to market…"
"high performance and function…"

"performance most important differentiator…"
"functions implemented mainly by software…"

Source: International Technology Roadmap for Semiconductors 2007

Consumer Portable: Processor Driven

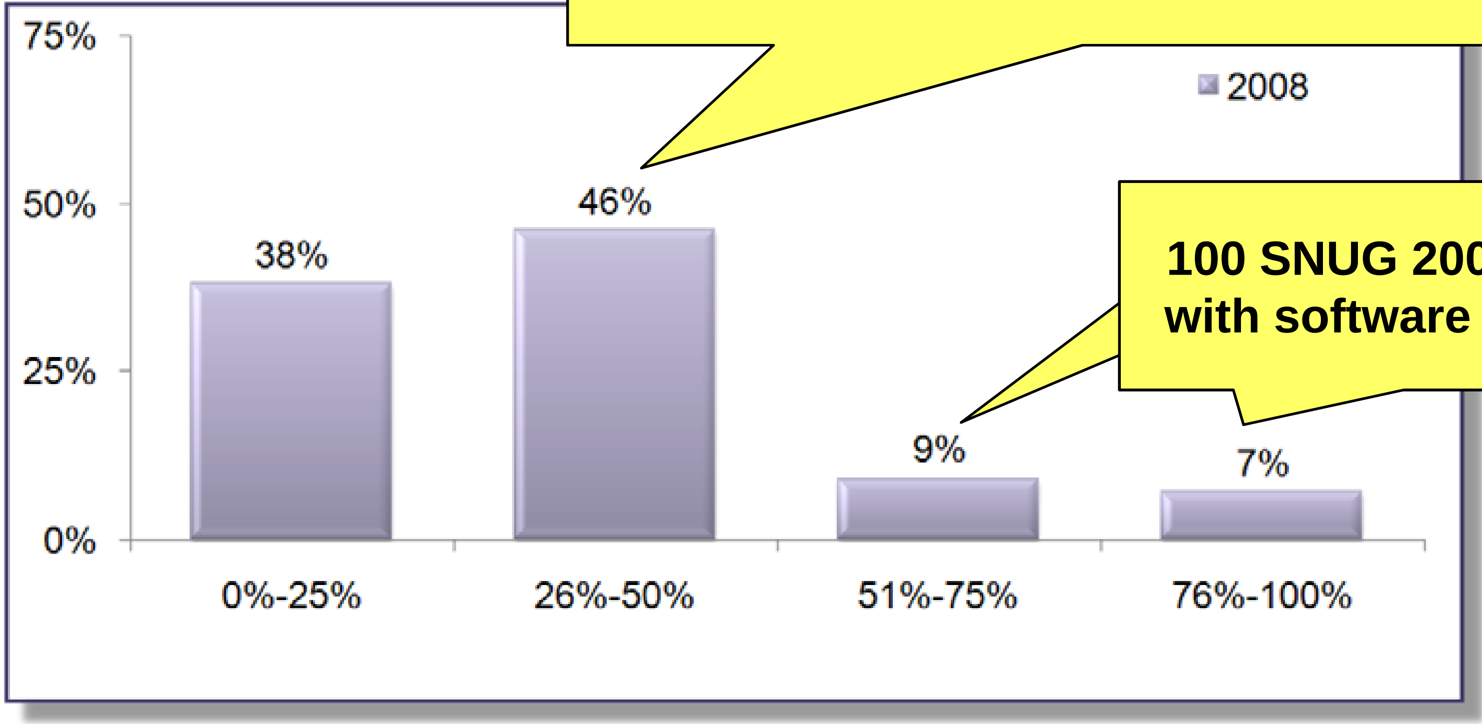


Number of processors in today's design starts are already significant

Source: ITRS 2007

Market Dynamics

Synopsys SNUG Data confirm the software trend!



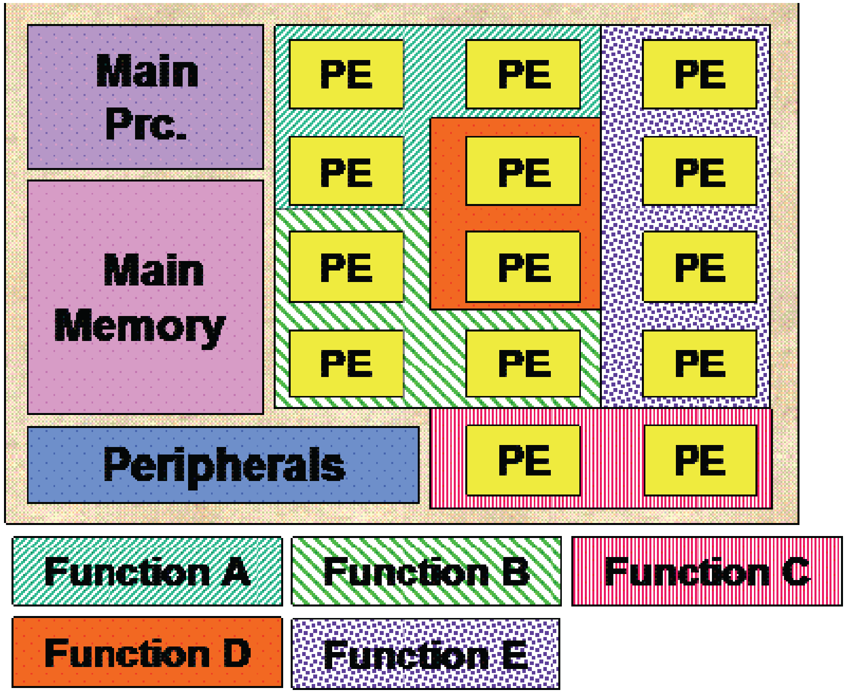
Almost 300 SNUG 2008 attendees already claim software issues !

100 SNUG 2008 attendees with software effort > 50%

What percentage of your total project effort is spent on software development (vs. hardware development) during design?
2008 N = 404; Margin of error = +/- 5%

Source: Synopsys San Jose SNUG Survey

Some Challenges



- How to distribute software across cores?
  - Lots of legacy
  - Need to be able to migrate existing serial code
- Debug
  - Races, deadlocks, memory corruption
- Performance
  - Stalls etc.

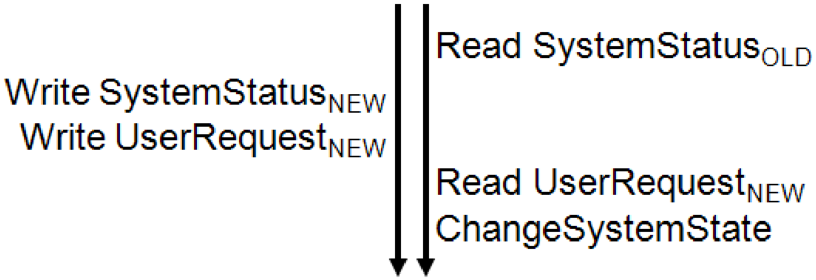
➢ Need lots of insight into the actual platform and the execution of software on it

SOME DEBUG CHALLENGES

Data Races

Leading to unpredictable results

- Two processes run in parallel accessing the same set of data
- While process A is accessing data it is interrupted and process B modifies the data
- Consistency of data read by process A unpredictable



Read SystemStatus$_{OLD}$

Write SystemStatus$_{NEW}$

Write UserRequest$_{NEW}$

Read UserRequest$_{NEW}$

ChangeSystemState

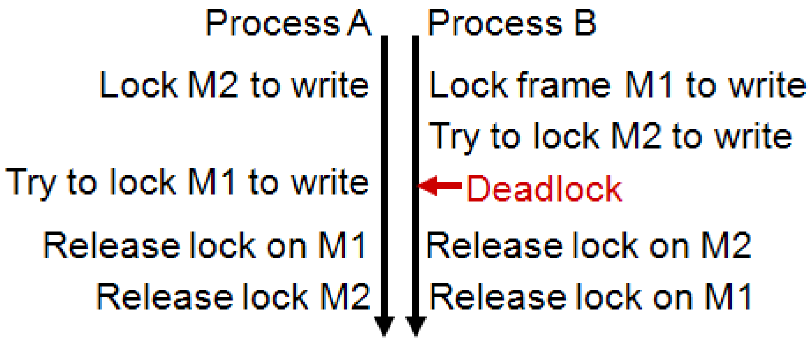
➢ For debugging, users need good insight into the platform and its memory

Deadlocks

Bringing the MP application to a halt …

- Two processes run in parallel, their data accesses are protected by semaphores
- Both processes individually lock the semaphore which the other needs next
- Deadlock occurs when the order is out of sync

➢ For debugging, users need good insight into the platform and its memory

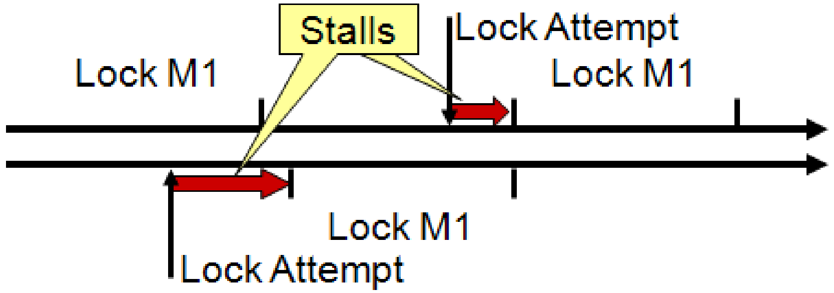


Process A	Process B
Lock M2 to write	Lock frame M1 to write
	Try to lock M2 to write
Try to lock M1 to write	Deadlock
Release lock on M1	Release lock on M2
Release lock M2	Release lock on M1

Stalls

Locks causing performance issues

- Not really an actual bug but has heavy impact on performance
- Happens when several parallel processes access the same locked area and have to wait for their turn a lot

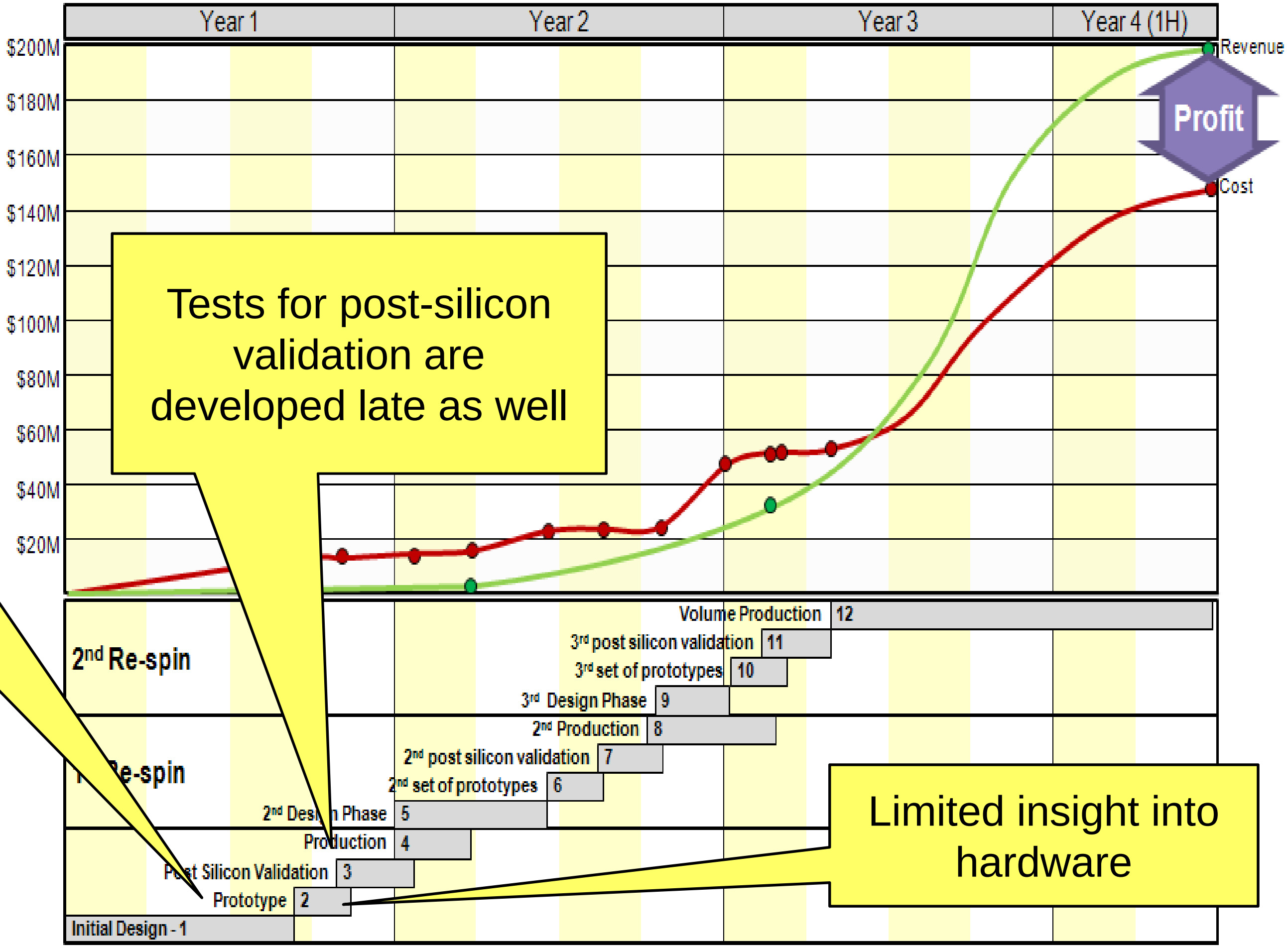


- ➢ For debugging, users need good insight into the platform and its memory
- ➢ Also, timing annotations are useful to quantify performance impact

SOLUTION:
VIRTUAL PLATFORMS

Current approaches will fail!

A Chip Design Project P&L Without Virtual Platforms



Prototype for software development available late in the design cycle

Tests for post-silicon validation are developed late as well

Limited insight into hardware

Source: Derived from IBS Data, 130nm, Wireless Application

Virtual Platforms offer the solution

Like Hardware – Only Much, Much Sooner!

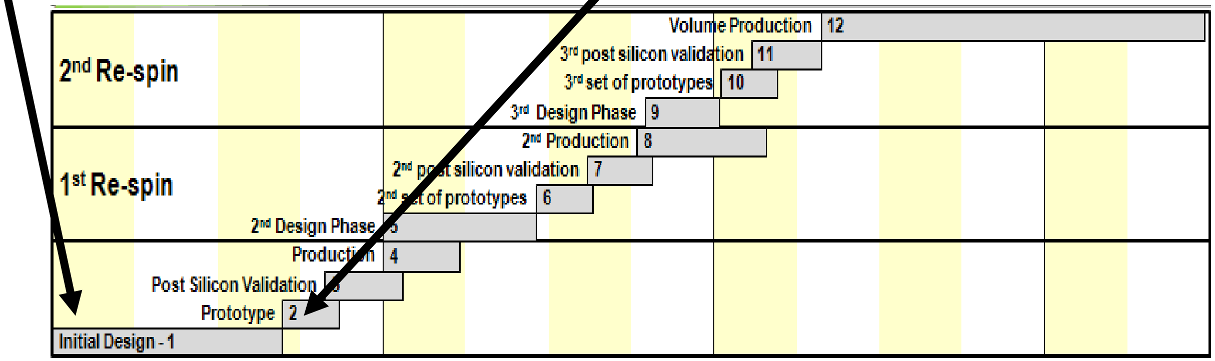
Early Availability
Enhanced
Debugging
Easy Deployment

Fully functional software model of SoC, board, I/O, user interface

Executes **unmodified** production **code**

Runs at **almost real-time**

High system **visibility** and **control** incl. **multi-core** debug

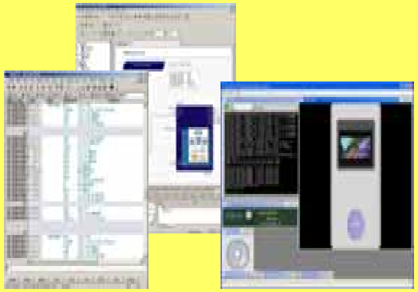


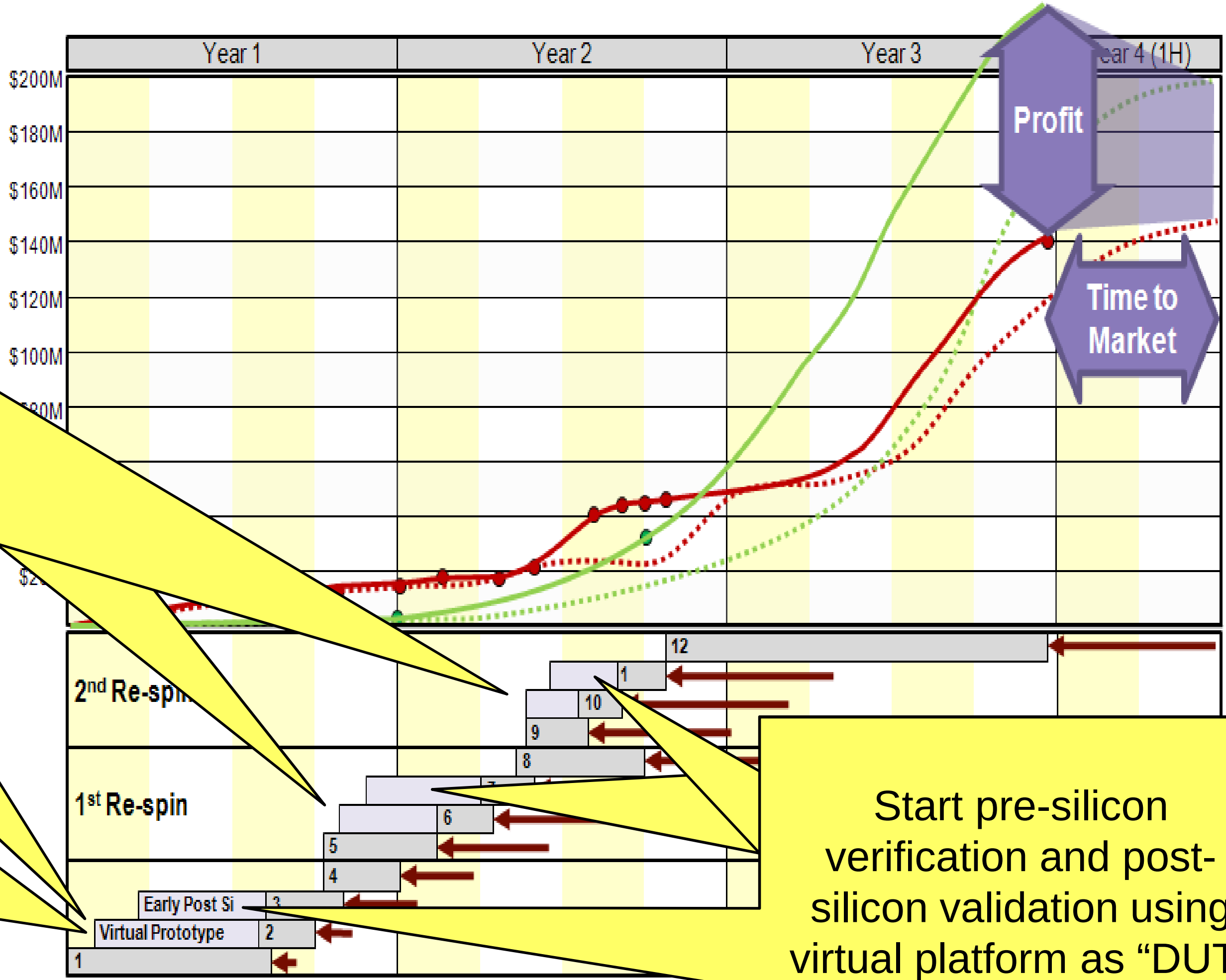
Source: Derived from IBS Data, 130nm, Wireless Application

Virtual Platform Impact

Gain full insight into design, get to market early and increase profit

Start software development pre-silicon, long before hardware is available !





Full insight into the platform for efficient debugging

Start pre-silicon verification and post-silicon validation using virtual platform as “DUT”

Requirements / Technology Vectors
Using TL Models for Early Software Development

- **Completeness – Model of complete SYSTEM**
  - Required as “system software” requires all these aspects
  - Coverage required for: (1) board-level, (2) SOC models, (3) system I/O, (4) system/device user interface
- **Extreme Simulation Performance - > 10-50 MIPS for full platform**
  - Higher performance allows bigger software stacks to be run
  - Required to support fast “edit-compile-debug” development cycle
- **Binary compatibility**
  - Run actual target binaries, requiring no changes between simulator & target
  - Avoids porting effort & reduces risk & enables SW optimizations
- **Visibility & controllability**
  - Leverage simulator unique features to improve visibility into “black-box” SoCs
  - Opportunity for dramatic SW development productivity improvement (2-5x)
- **Virtual I/O – Emulates System I/O**
  - Allows SW developer to test end user scenarios (e.g. contact sync)
  - Acts as a real SW development target – real world connectivity
- **Supports standard SW development interfaces & tools**
  - No changes in SW development cycle (no new tools, methodologies, …)
  - No new debug infrastructure needs to be developed

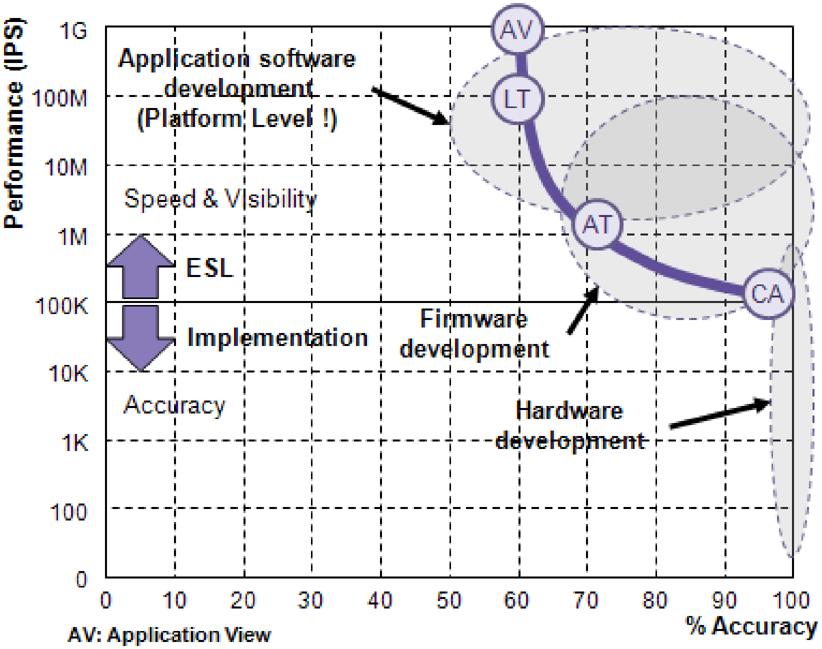
SYSTEMC TLM-2.0

The Impact of SystemC TLM-2.0

Enabling Interoperability and Scalability

- Previously proprietary (backdoor) APIs & new additions have now been standardized:

- (DMI) Direct Memory Interface
  - Direct backdoor access into memory
  - Allows un-inhibited ISS execution
- LT (Loosely Timed) modeling
  - Declare but don't execute timing
  - Allows speed/accuracy trade-offs
- Temporal Decoupling
  - Only synchronize when necessary
  - Allows multicore speedup

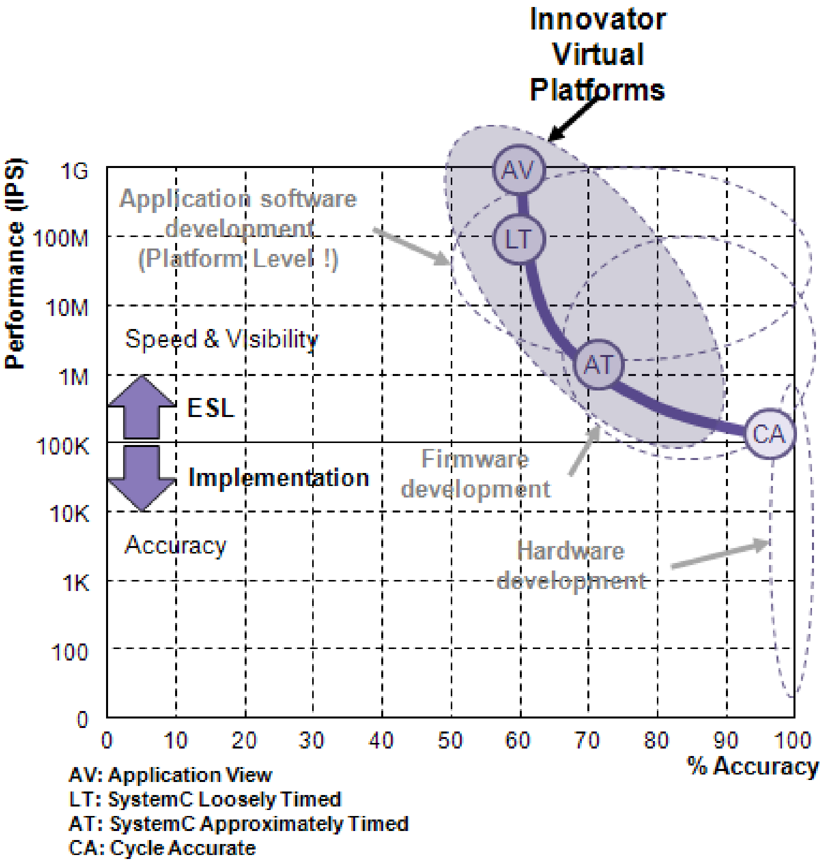


AV: Application View
LT: SystemC Loosely Timed
AT: SystemC Approximately Timed
CA: Cycle Accurate

The Impact of SystemC TLM-2.0

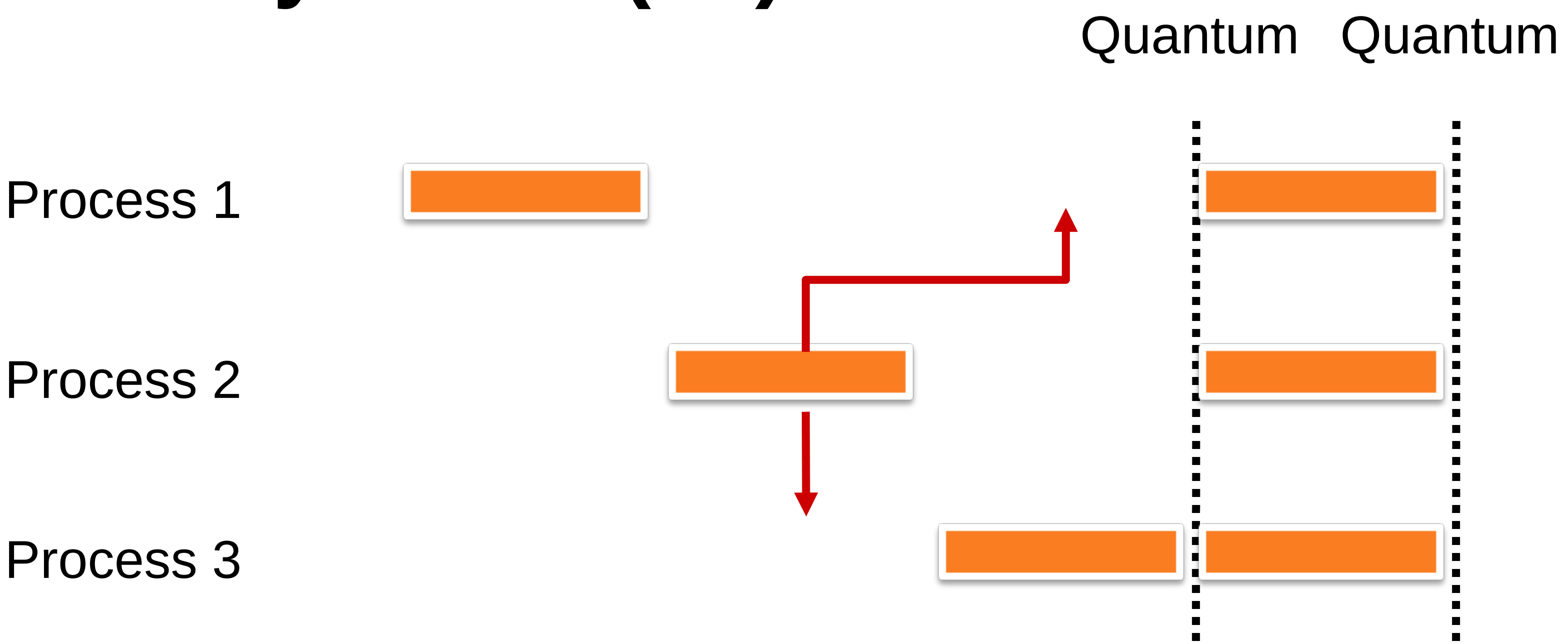
Synopsys has the complete portfolio!

- Innovator
  - PV (LT) modeling
  - PV+T (AT) “timing annotation”
- DesignWare® IP
  - Implementation, Verification and System-Level IP
- Modeling Services
- VCS
  - For hard CA requirements running in software
- Synplicity HAPS



MODELING STYLE EXAMPLES

Loosely-timed (LT)



Quantum Quantum

Process 1

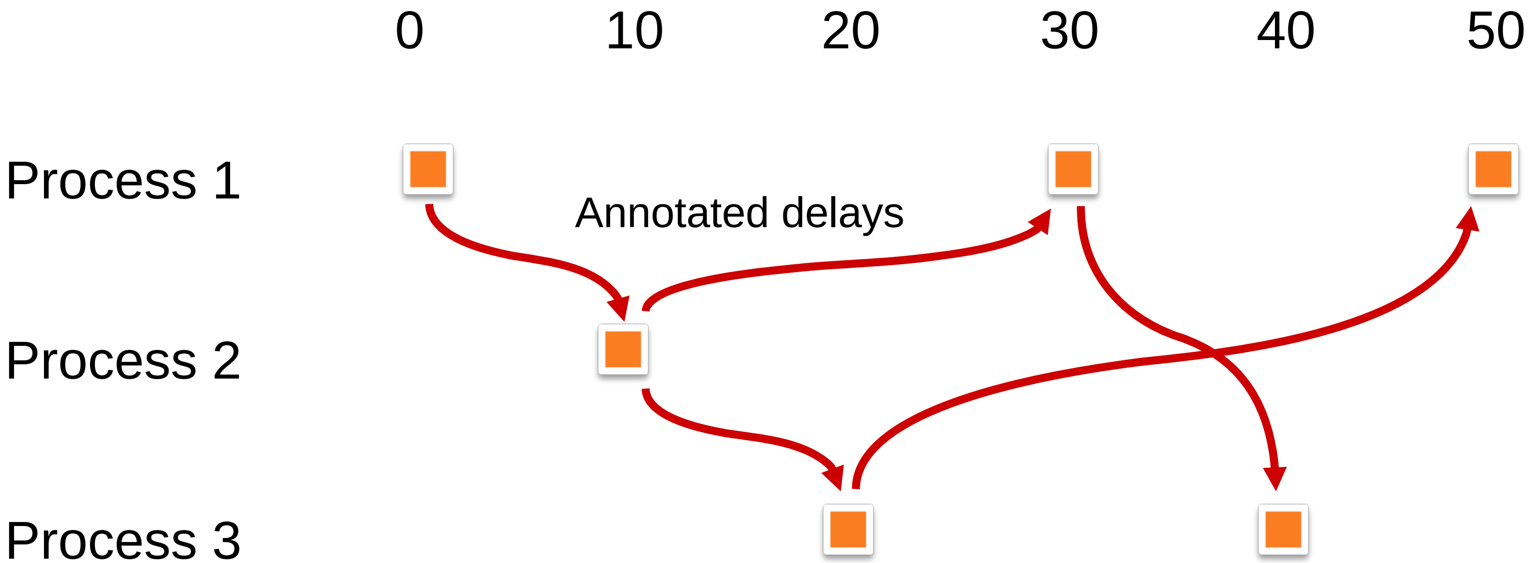
Process 2

Process 3

- Only sufficient timing detail to boot O/S and run multi-core systems
- Processes can run ahead of simulation time (temporal decoupling), for faster performance
- Uses direct memory interface (DMI)- no contention

- Quantum is user configurable. Data races, deadlocks and stalls may not be observed if Quantum is too long

Approximately-timed (AT)



- Sufficient for architectural exploration
- Processes run in lock-step with simulation time

- A model may switch between the LT and AT coding style during simulation. Run rapidly through the reset and boot sequence at the LT level, then switch to AT modeling for more detailed analysis once the simulation has reached an interesting stage

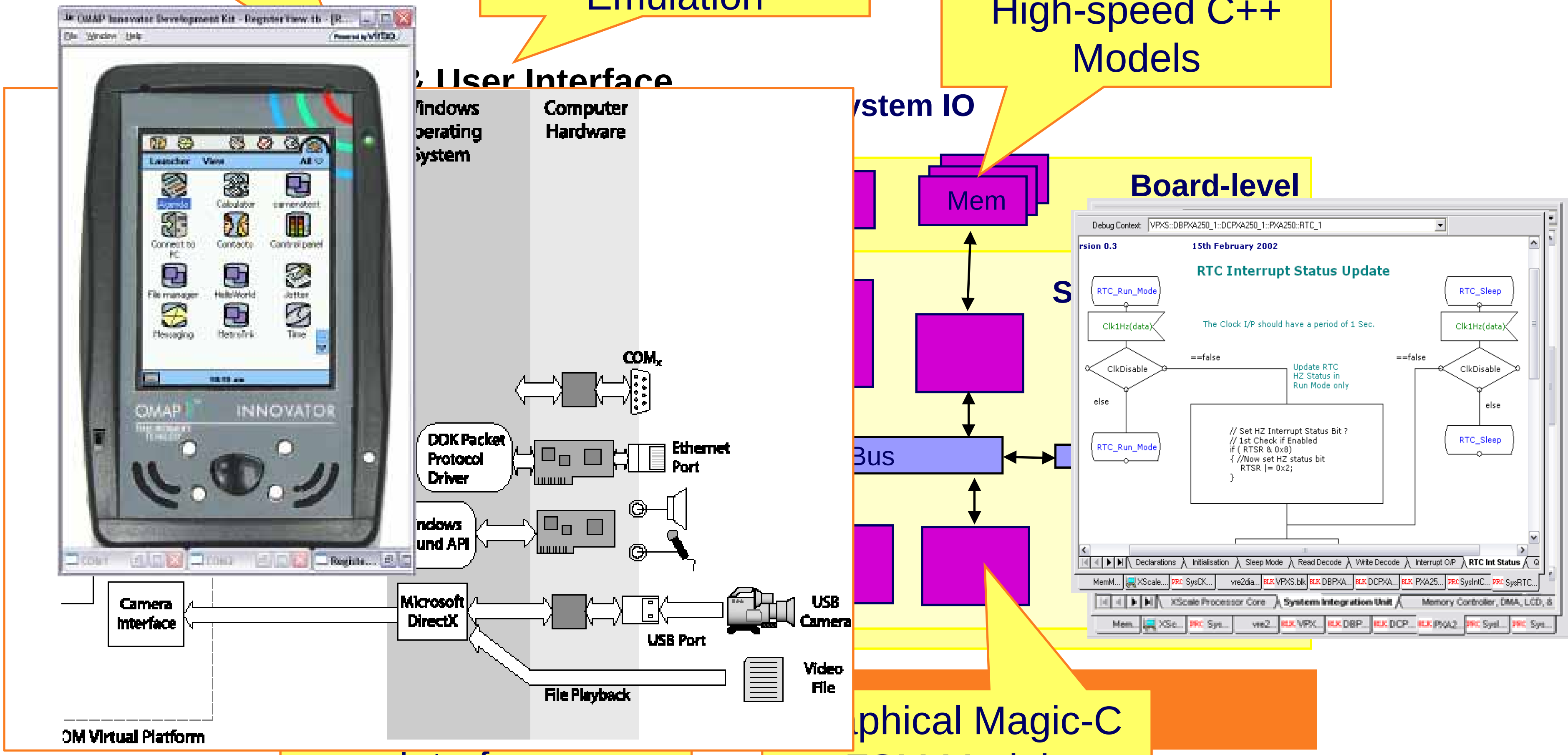
SystemC LT System Model

Fast Transaction-level Models - Technology Details

Virtual I/O

User Interface Emulation

High-speed C++ Models



Debug Context: VPXS::DBPXA250_1::DCPXA250_1::PXA250::RTC_1

rsion 0.3 15th February 2002

RTC Interrupt Status Update

The Clock I/P should have a period of 1 Sec.

Update RTC HZ Status in Run Mode only

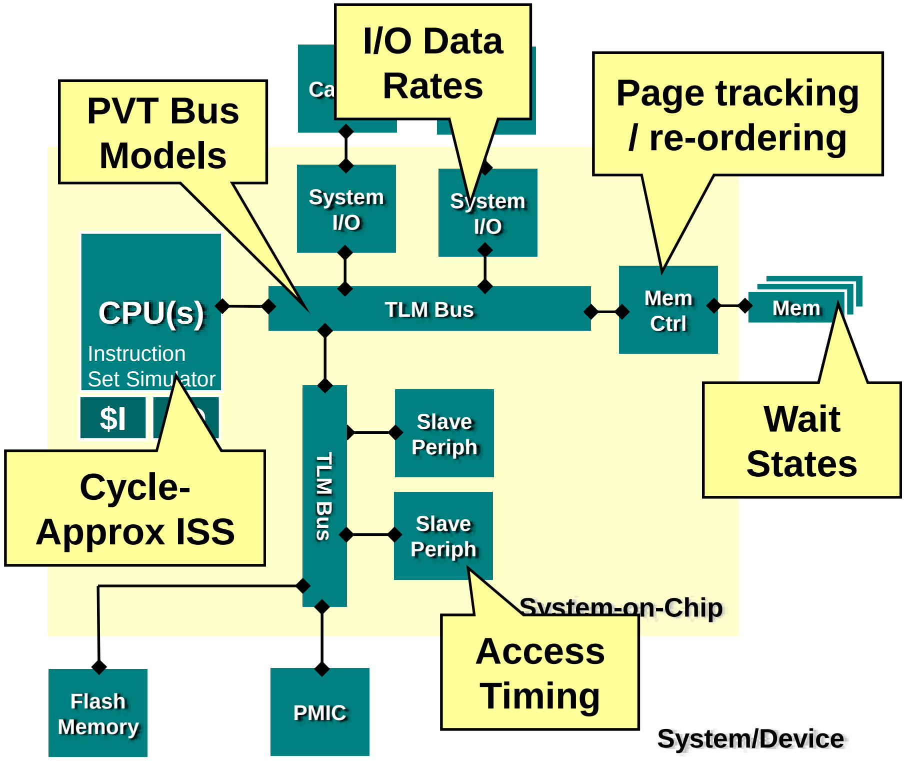
```
// Set HZ Interrupt Status Bit ?
// 1st Check if Enabled
if ( RTSR & 0x8)
{ //Now set HZ status bit
   RTSR |= 0x2;
}
```

Interfaces

Graphical Magic-C FSM Models

SystemC AT System Model

Features & Capabilities



I/O Data Rates

PVT Bus Models

Page tracking / re-ordering

Wait States

Cycle-Approx ISS

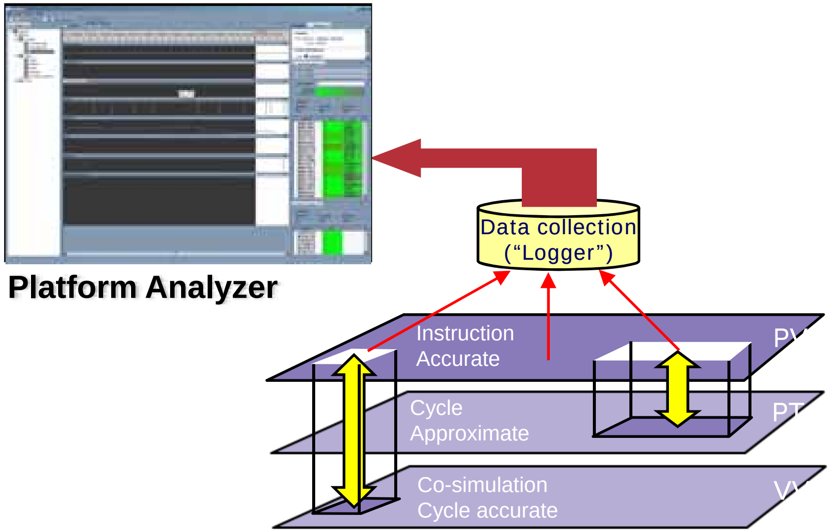
Access Timing

- **Hardware / Timing Features Modeled**
  - **Buses**
    - Contention & Arbitration
    - Pipelining & concurrency
    - Burst-at-once (typical), or individual phases
  - **Slaves**
    - Access timing
    - Processing delay
  - **CPU**
    - Cache(s)
    - Instruction cycle timing: CPI, or
  - **Memory controllers**
    - Static delay (‘wait states’), or
    - Pages / banks
    - Dynamic (transaction re-ordering)

- **Performance Statistics Generated**
  - **CPU: c**ycle counts, cache hit/miss rates, average cache line age, ..
  - **Bus**: effective bandwidth, # transactions, ..
  - **Memory controllers:** #page hits/misses, #re-schedules, queue size
  - **System-specific statistics**

…

Required Logging & Visualization

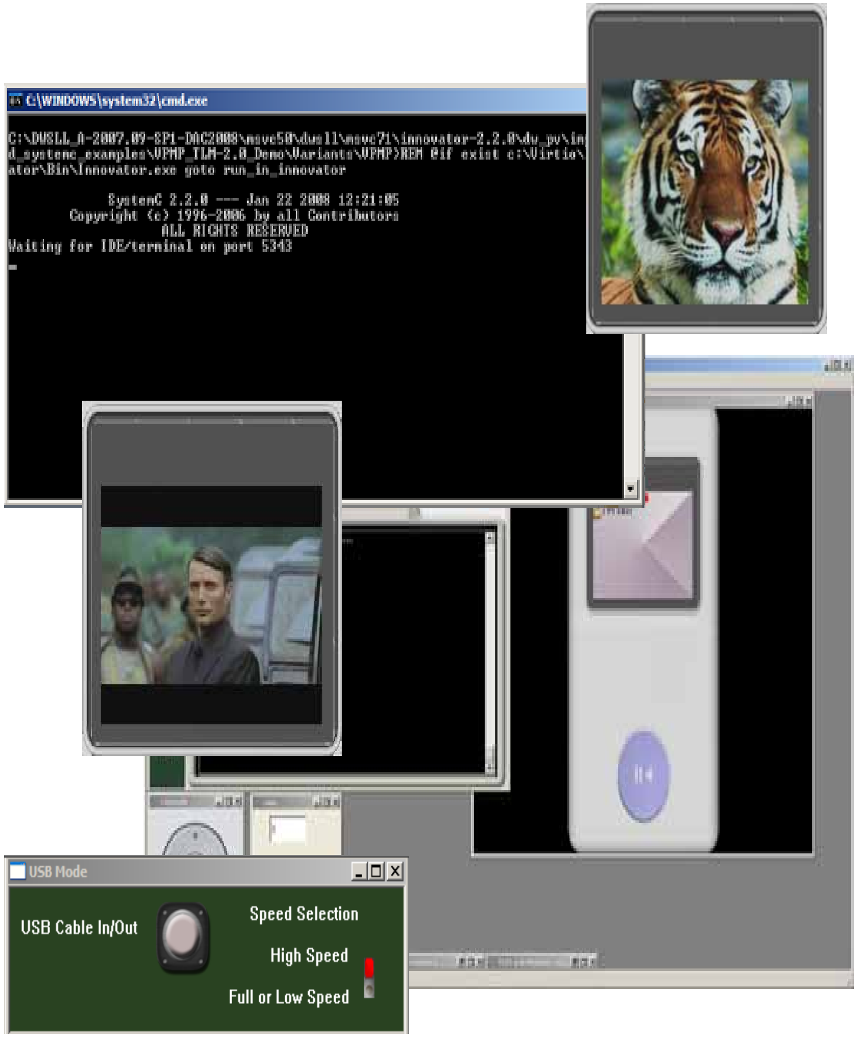


Predefined Views

- Bus transactions / traffic view
- Bus transaction statistics
- Timing display of system (power) events
- Memory
  - Memory utilization
  - (Effective) Memory bandwidth
  - Memory heat map
- MMU: TLB hits/misses Cache & memory heat map
- Cache statistics
  - Cache hit/miss
  - Cache eviction rate
  - Cache heat map
- IRQ latency, together with laxity window
- Bus bandwidth history
- Instruction traces

A TLM-2.0 Virtual Platform

1. Platform in SystemC using TLM-2.0 LT modeling
2. Runs at real-time speed
3. Interact live with virtual platform design
4. Connect to environment via Virtual I/O
5. Run real time video/audio
6. Start/Stop/Break for debug in multicore scenarios



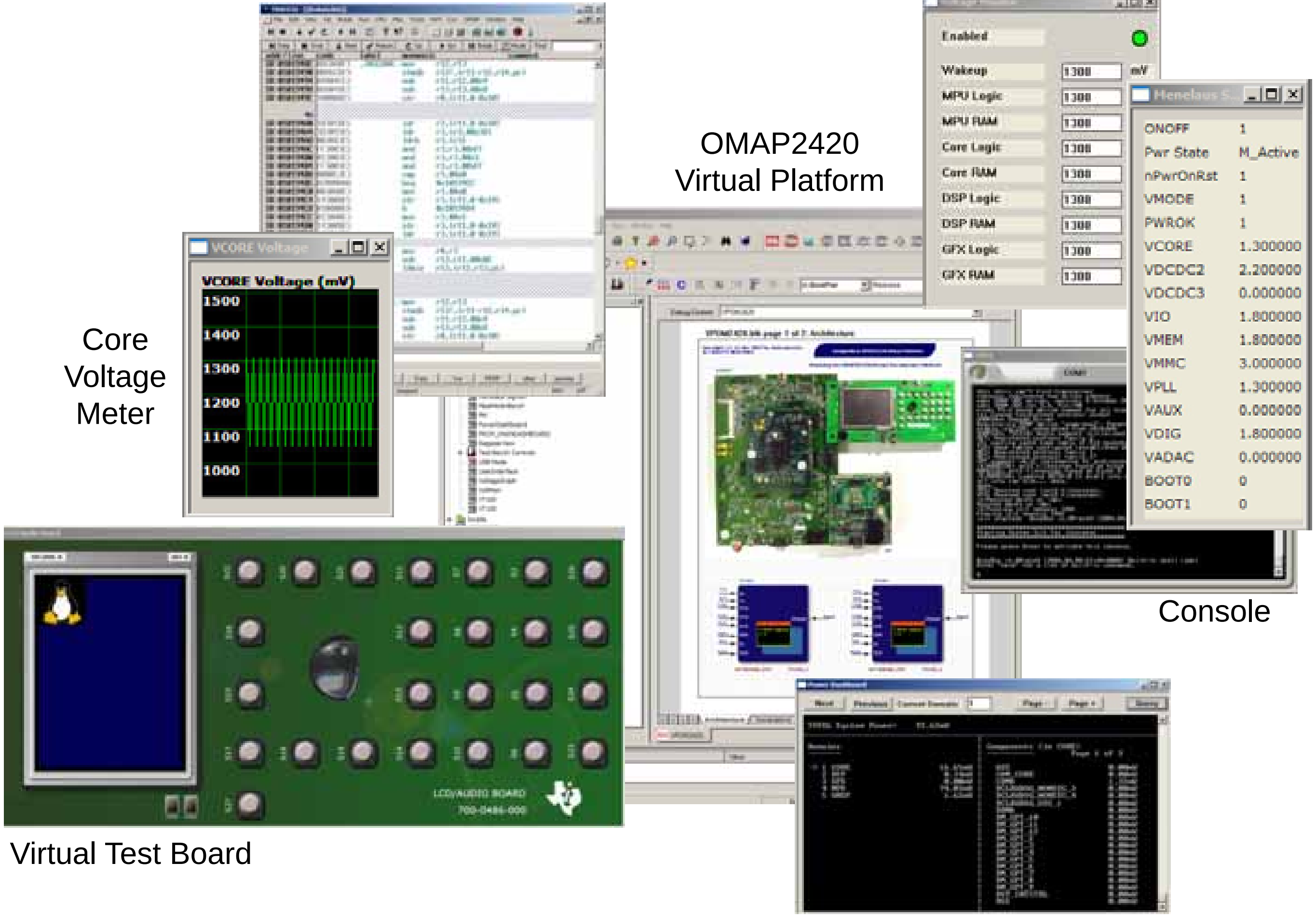
C:\WINDOWS\system32\cmd.exe

```
C:\DWSLL_A-2007.09-SP1-DAC2008\msvc50\dwsll\msvc71\innovator-2.2.0\dw_pv\in
d_systemc_examples\VPMP_TLM-2.0_Demo\Variants\VPMP>REM @if exist c:\Virtio\
ator\Bin\Innovator.exe goto run_in_innovator

            SystemC 2.2.0 --- Jan 22 2008 12:21:05
        Copyright (c) 1996-2006 by all Contributors
                   ALL RIGHTS RESERVED
Waiting for IDE/terminal on port 5343
_
```

Texas Instruments OMAP 2420

Multicore Debug Example



Software Debugger

Voltage Monitors

OMAP2420 Virtual Platform

Core Voltage Meter

Console

Virtual Test Board

Enabled ●

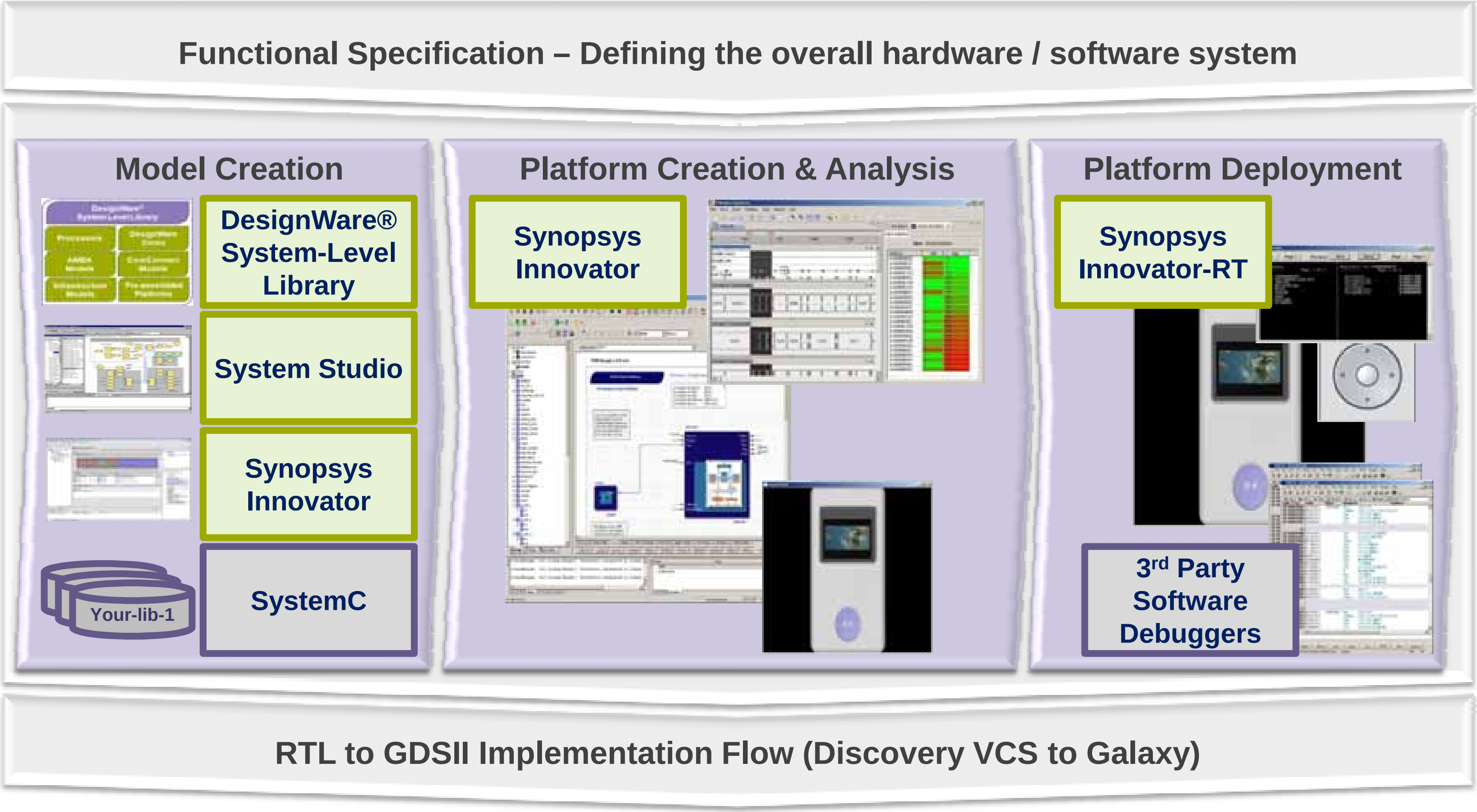
Wakeup	1300	mV
MPU Logic	1300	
MPU RAM	1300	
Core Logic	1300	
Core RAM	1300	
DSP Logic	1300	
DSP RAM	1300	
GFX Logic	1300	
GFX RAM	1300	

ONOFF	1
Pwr State	M_Active
nPwrOnRst	1
VMODE	1
PWROK	1
VCORE	1.300000
VDCDC2	2.200000
VDCDC3	0.000000
VIO	1.800000
VMEM	1.800000
VMMC	3.000000
VPLL	1.300000
VAUX	0.000000
VDIG	1.800000
VADAC	0.000000
BOOT0	0
BOOT1	0

SUMMARY

Virtual Platforms in the Design Flow

Leading from functional specification to RTL Implementation



Functional Specification – Defining the overall hardware / software system

Model Creation

DesignWare® System-Level Library

System Studio

Synopsys Innovator

SystemC

Platform Creation & Analysis

Synopsys Innovator

Platform Deployment

Synopsys Innovator-RT

3rd Party Software Debuggers

RTL to GDSII Implementation Flow (Discovery VCS to Galaxy)

Executive Summary

- MPSoCs will have a daunting number of processors
- Software will be challenging, especially migration and debug
- Some debug examples
- Virtual platforms offer the solution
- SystemC TLM-2.0
- Modeling Style Examples
- Demo
- Wrap & Questions