Scaling FV Up

John O'Leary
Principal Engineer
Strategic CAD Labs
john.w.oleary@intel.com

23 February 2010

Focus questions

- What is the role of verification in designing and building reliable systems?
- How can verification help in decomposing problems and composing solutions?
- What new ideas are needed to scale up the technology and dial down our ambition for verifying large systems?
- How do connect verification in the small and in the large?

Why formal verification?

- In 1994, Intel took a $475M charge against revenues to cover replacement costs and inventory writedown due to the Pentium® FDIV hardware flaw
- In 2010 our products are more thoroughly validated, but also
  - Contain many more components
  - Have much more complex functionality, often orchestrated by firmware
  - Penetrate the market much more quickly
  - Ship in much higher volumes
- Could a flaw result in recall of a main stream product? What might it cost in dollars and reputation?
- We need to think about the unthinkable, lest we repeat it

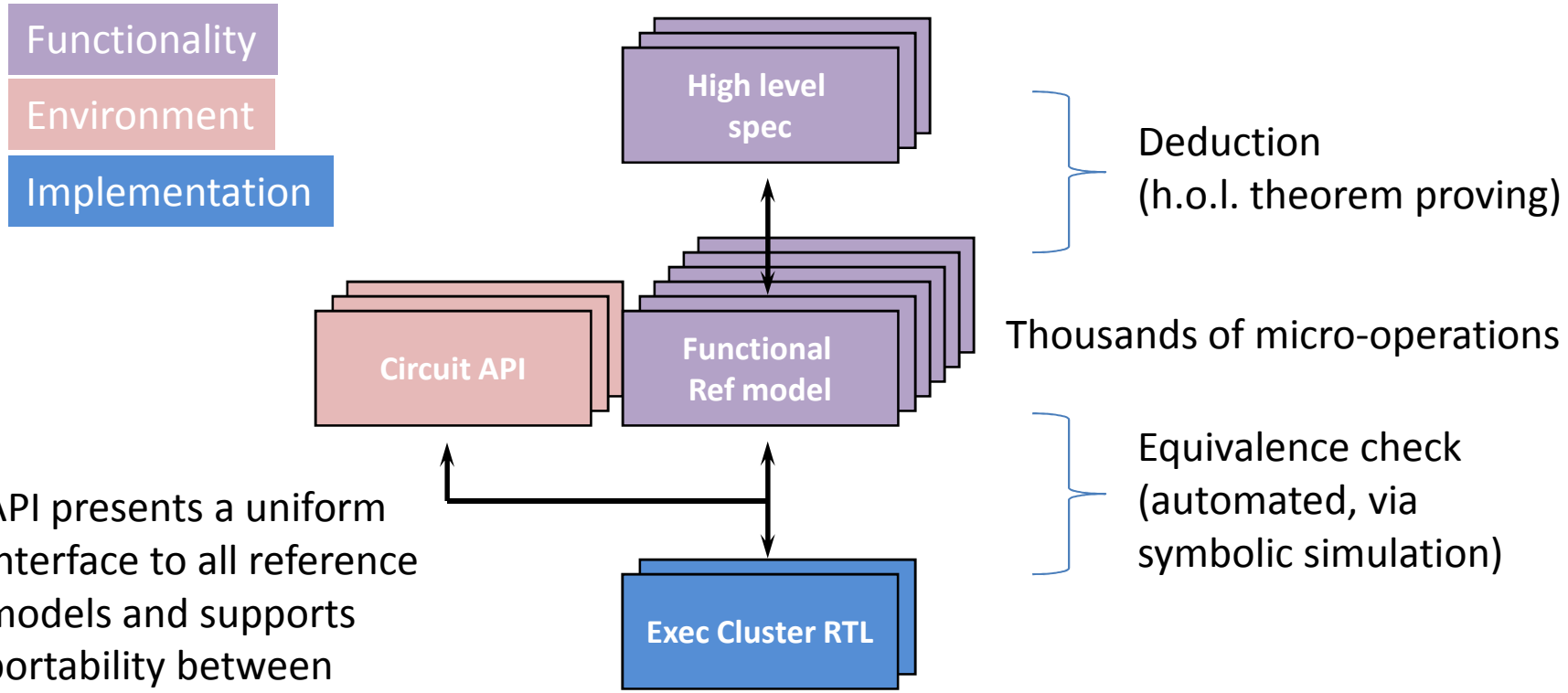
Copyright © 2010 Intel Corporation. All rights reserved.

Formal tools in Intel HW Design

- RTL-to-RTL and RTL-to-schematic equivalence verification
  - Widely deployed across the industry
- Explicit-state model checking
  - TLA+/TLC, Murphi employed for early, high level verification of cache protocols, uarch algorithms
- Model checking (mostly SAT-based BMC)
  - Broad-spectrum bug detection for RTL blocks, interfaces, etc
- Formal equivalence checking of microcode flows
- Symbolic simulation and theorem proving
  - Prove functional correctness of execution cluster datapaths

Copyright © 2010 Intel Corporation. All rights reserved.

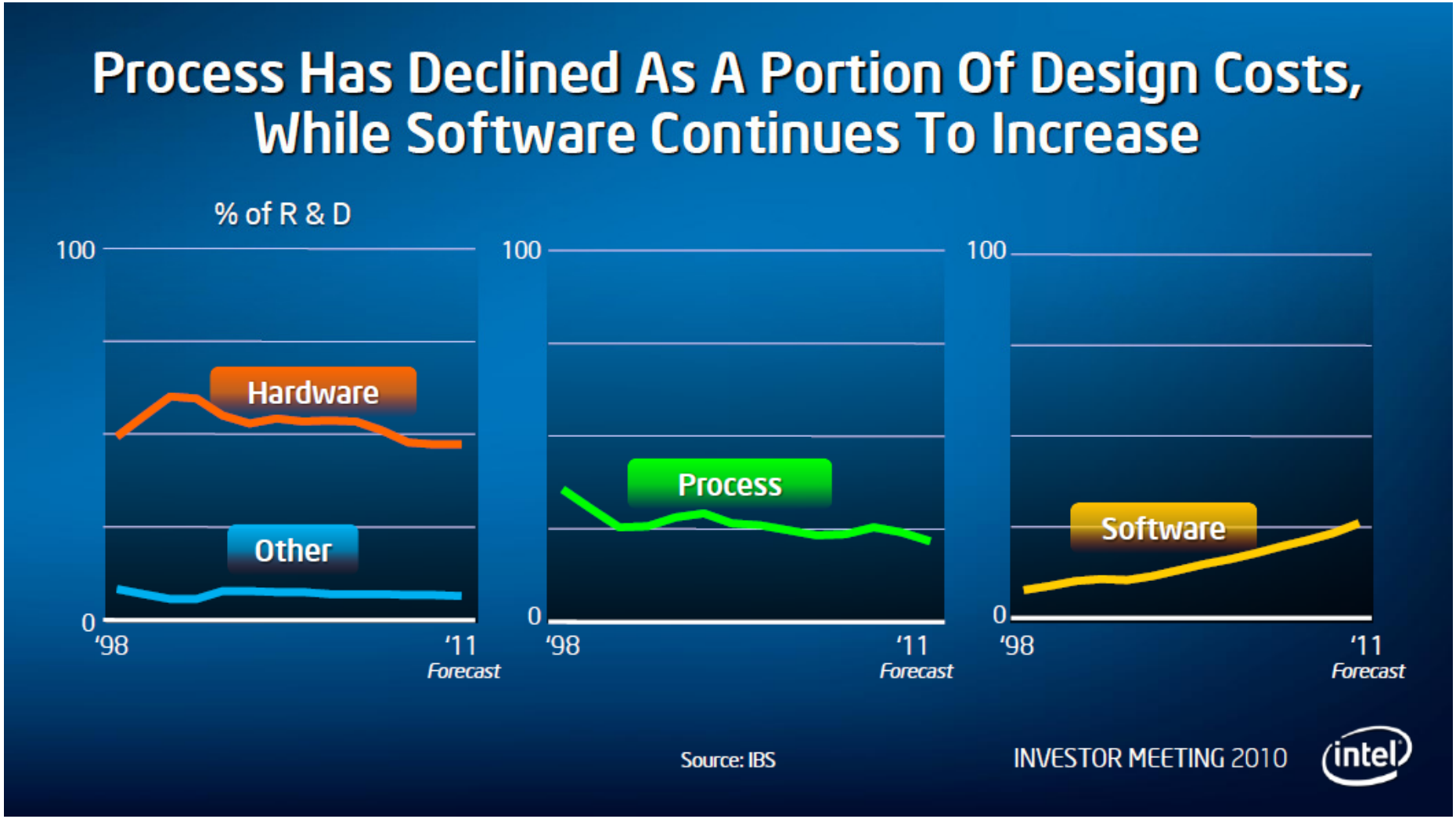
Exec cluster verification methodology



See Roope Kaivola, *et al*. Replacing testing with formal verification in Intel Core™ i7 processor execution engine validation. CAV 2009.

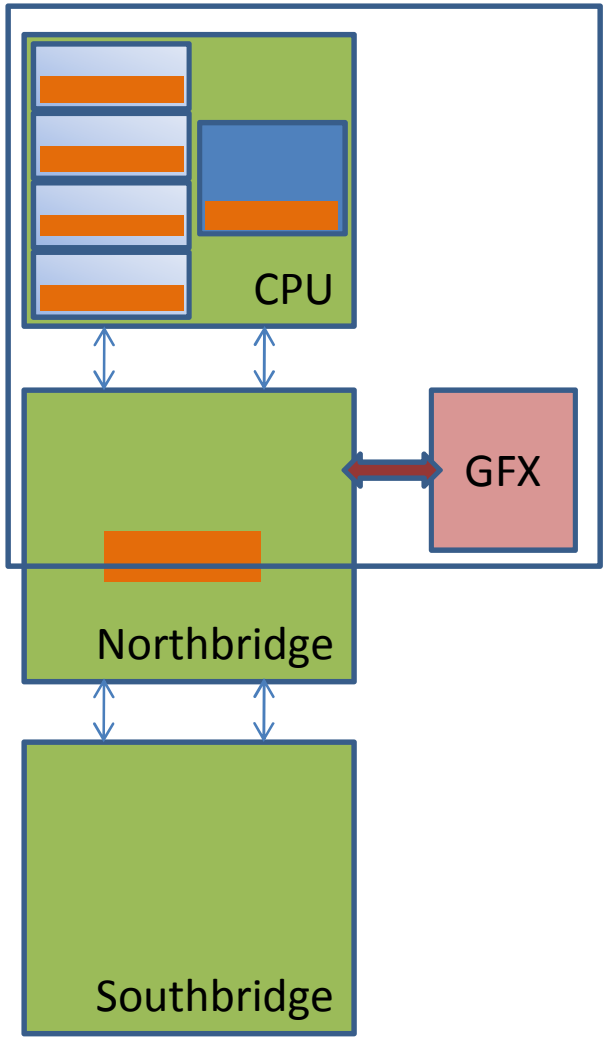
Copyright © 2010 Intel Corporation. All rights reserved.

Are we solving the right problem?



Copyright © 2010 Intel Corporation. All rights reserved.

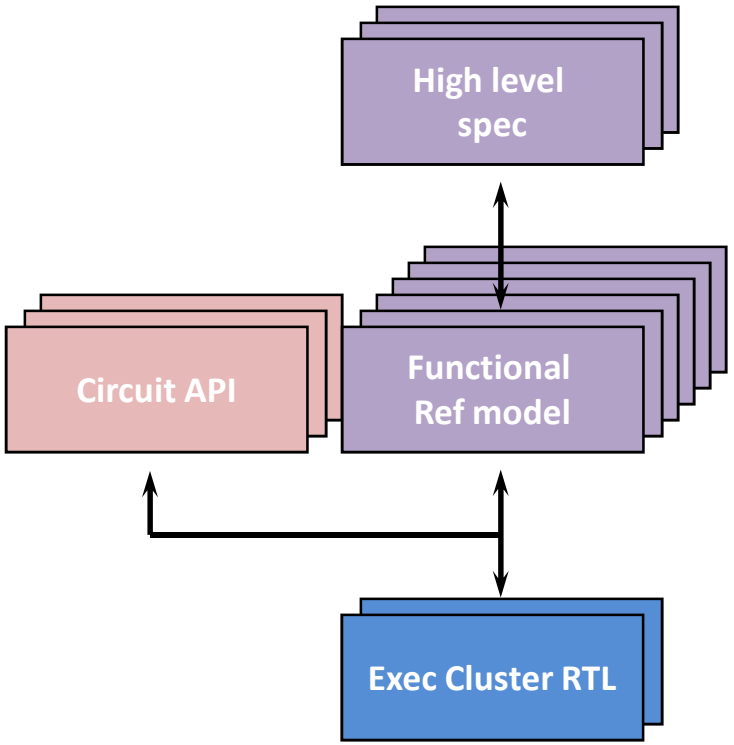
SW, FW and System Integration



- SW/FW growth due to perceived advantages
  - Flexibility & survivability
  - Complex new platform features
    - Intel® vPro™ Technology is a good example
- Increased integration accelerates growth
  - Single core → multi-core
  - Power management
  - Integrated memory controller + PCI Express
  - Graphics in package
- SW, firmware and system integration are on critical path for product development and launch
- *Perceived quality of Intel hardware products is now a direct function of the quality of supporting SW & firmware*

Copyright © 2010 Intel Corporation. All rights reserved.

Scale FV to HW/FW/SW systems



- We can learn some lessons from hardware validation best practices
  - Specify formally
  - Verify equivalence
  - Model environments
  - Re-use specs and proofs
  - Make deductive reasoning practical
  - Fix the programming languages

Copyright © 2010 Intel Corporation. All rights reserved.

Specify formally

- Clear, unambiguous specifications are obviously valuable
  - Even if the specification is prone to change
  - Even if there is no fully-formal link to implementation
- Design intent is expressed by English text and some more-formal artifacts:
  - Tables
  - Diagrams (bubble diagrams, block diagrams, message sequence charts)
  - Pseudo-code
- Research questions:
  - Can we endow readable documentation with a formal semantics?
  - Can we promote specifications that:
    - Offer more abstraction than C and System Verilog programs
    - Are easier to understand and reason about than C and System Verilog programs
    - Deliver value during development/maintenance
  - Can we practically check
    - self-consistency of specifications
    - Firmware/software/hardware implementations against specifications?

Copyright © 2010 Intel Corporation. All rights reserved.

Verify equivalence

- Two common sources of code change:
  - Change code to optimize speed/space/energy while preserving functionality
  - Add new functionality while preserving the old
- Formal Equivalence Verification (FEV) *extremely* useful in hardware design
  - FEV tells you more than regression testing
  - FEV is usually faster than regression testing
  - FEV is accessible to designers with little formal methods expertise
- FEV for software would be valuable

Copyright © 2010 Intel Corporation. All rights reserved.

Model environments

- Modular validation methods require detailed and accurate environment models
  - For hardware blocks, this means modeling neighboring blocks
  - For software routines, this means modeling the caller, subroutines, library functions, etc
  - For firmware the environment might be both software and hardware
- The effort of writing environment models limits the uptake of modular validation methods
- Research needed:
  - Design by contract (MSR's Spec# project illustrates an approach)
  - Automatic abstraction of environment models from interfaces and code
  - Synthesis of environment models from simulation traces
  - Environment modeling at the hardware/firmware interface (esp timing)

Copyright © 2010 Intel Corporation. All rights reserved.

Re-use of specs and proofs

- Products come in related families and generations
  - Families: server, desktop, mobile and ultramobile parts
  - Generations: Intel® Core™2 Duo Processor, Intel® Core™ i5 Processor
- Robust reusable proofs
  - Allow the cost of verification to be amortized
  - Certify common functionality across generations and families
- Many Intel datapath proofs are descended (with modification) from the Intel Pentium® 4 processor generation
  - "The cost of verifying is less important than the cost of re-verifying"
- An analogous scenario in software:
  - Pick a key component of Linux version N
  - Develop a specification and verify the component against it
  - *Do it again* for version N+1
  - *Do it again* for version N+2
  - *Port* to the equivalent BSD component
- We need to better understand how to reuse proofs and verification results (and validation collateral in general)

Copyright © 2010 Intel Corporation. All rights reserved.

Make deductive reasoning practical

- Recent and impressive results:
  - NICTA's seL4: fully and formally verified version of the L4 microkernel (NICTA = National ICT Australia)
  - MS Research: verification of Hyper-V hypervisor (in progress)
- Cost still incredibly high: seL4 is 7500 LoC, est 30 PY to verify
- Research is needed in
  - Programming logics
  - Satisfiability Modulo Theories (capacity, quantifier reasoning)
  - Special purpose solver technology for, e.g., separation logic
  - Combination of techniques
    - Combining guided theorem proving + automatic model checking has proven successful for hardware datapaths
  - Proof and proof-script reuse

Copyright © 2010 Intel Corporation. All rights reserved.

Fix the programming languages

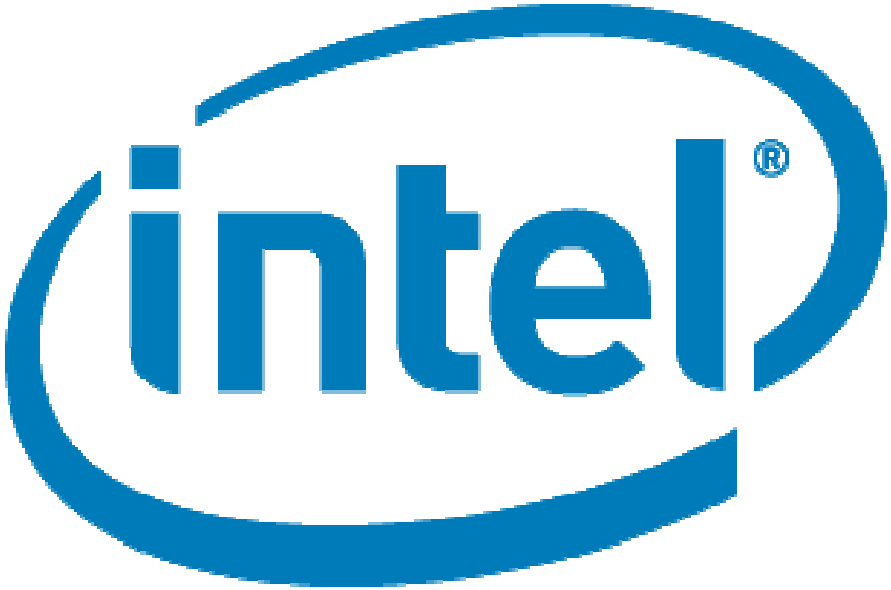
- Firmware at Intel and elsewhere is written in microcode, assembler, C or C++.
  - ✓ Fine grained control over memory allocation and layout
  - ✓ Easy to understand the effect of code changes on execution performance
  - ✗ Weak or nonexistent type systems
  - ✗ Error-prone APIs for memory management and concurrency (e.g. malloc/free/threads in C)
- Dominant language for RTL design is (System) Verilog
  - "ASCII schematics"
- Research topics:
  - Strong type systems suitable for hardware, firmware and low level software
  - Safe (or safer) memory management and concurrency primitives
  - Enabling more effective static analysis, formal reasoning and testing
  - Preserving direct control over memory and execution in higher level languages

Copyright © 2010 Intel Corporation. All rights reserved.

Summary

- We expect continued growth in the volume and variety of software and firmware in all our CPU and SOC products
- Validation of software/firmware and integration with the hardware components is already on or near the critical path for product development and launch
- To scale formal methods to meet the challenge we can learn from what has worked in hardware validation
  - Specify formally
  - Verify equivalence
  - Model environments
  - Re-use
  - Make deductive reasoning practical
  - Fix the programming languages

Copyright © 2010 Intel Corporation. All rights reserved.


intel

Industry trend

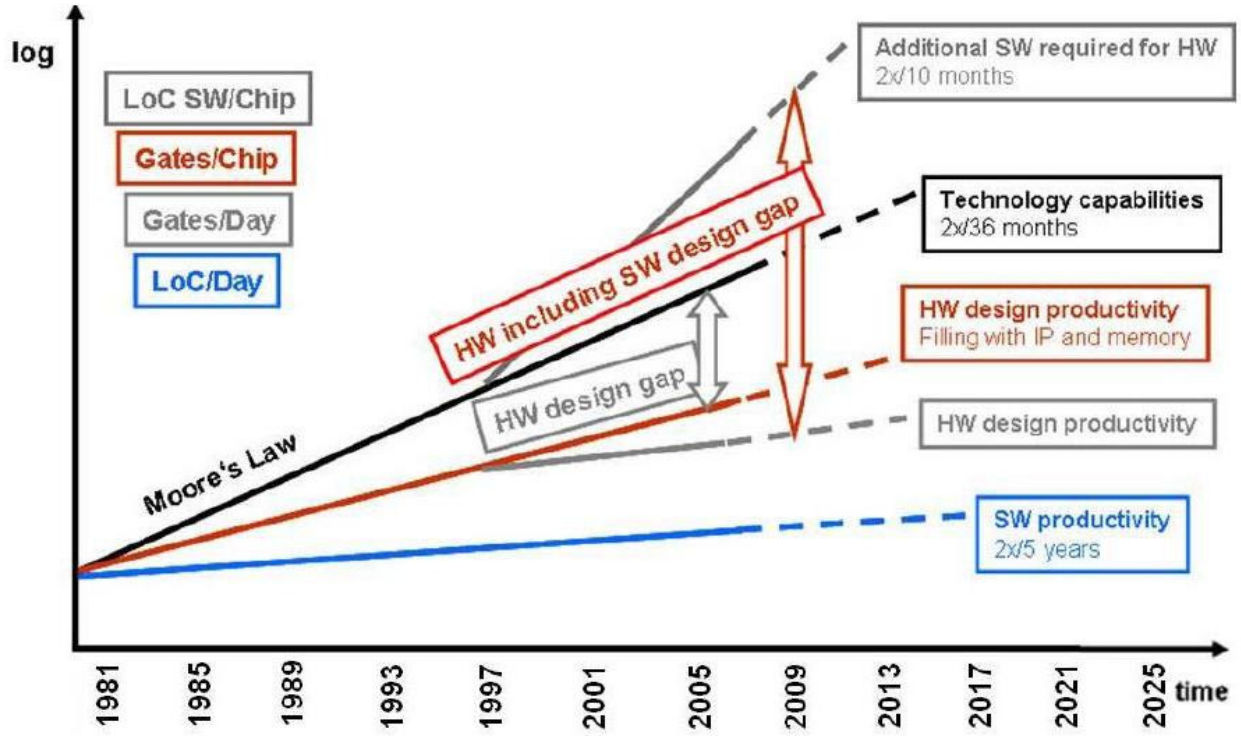


Figure DESN3 Hardware and Software Design Gaps versus Time[5]

Source: ITRS 2009

Copyright © 2010 Intel Corporation. All rights reserved.