# INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS

# 2004 UPDATE

# Design

Jointly Sponsored
by

European Semiconductor Industry Association

Japan Electronics and Information Technology Industries Association

Korea Semiconductor Industry Association

Taiwan Semiconductor Industry Association

Semiconductor Industry Association

THE ITRS IS DEVISED AND INTENDED FOR TECHNOLOGY ASSESSMENT ONLY AND IS WITHOUT REGARD TO ANY COMMERCIAL CONSIDERATIONS PERTAINING TO INDIVIDUAL PRODUCTS OR EQUIPMENT.

# TABLE OF CONTENTS

Design ........ 1
2004 Update Highlights ........ 1
Working Group Section Updates ........ 1
Design For Manufacturability [**New Section Added**] ........ 1
Design Verification **Updated** ........ 7

# LIST OF FIGURES

Figure a Possible variability abstraction levels ADDED ........ 3
Figure b Current Verification Tool Landscape ADDED ........ 9
Figure c General Design Potential Solutions ADDED ........ 17

# LIST OF TABLES

Table A Design For Manufacturability Challenges ADDED ........ 2
Table 13 Near-term Breakthroughs in Design Technology for AMS ........ 4
Table 14 Design Process Challenges ........ 4
Table 15 System-level Design Challenges ........ 5
Table 16 Logical, Circuit, and Physical Design Challenges ........ 6
Table 17a Design Verification Challenges—Near-term UPDATED ........ 8
Table 17b Design Verification Challenges—Long-term UPDATED ........ 12
Table 18 Design Test Challenges ........ 14
Table 19 Additional Design Technology Requirements UPDATED ........ 15
Table 20 Design Technology Improvements and Impact on Designer Productivity ........ 16

# DESIGN

## 2004 UPDATE HIGHLIGHTS

This update includes (a) a revised "General" design technology requirements (revised), with some new rows, and new explanations as footnotes for each row, but no changes in the values of existing rows, (b) a new design technology solutions table after the requirements table, which includes a definition of each row, (c) a new Design For Manufacturability (DFM) section with completely new content, and (d) an update of the verification section.

*Link to the 2003 ITRS Design chapter*

## WORKING GROUP SECTION UPDATES

### DESIGN FOR MANUFACTURABILITY [NEW SECTION ADDED]

Increasing variability, mask cost and data explosion, and lithography hardware limitations, are posing significant design challenges for the manufacturability of integrated circuits:

*1. Architecture challenges*—architectural redundancy will be required due to the difficulty in making circuits yield. It will be hard to do much more at this level of abstraction.

*2. Logic and circuit challenges*—digital and mixed-signal adaptive circuits will be increasingly necessary. Statistical design, including power and timing convergence will be fundamental. But it will be hard to make work without addressing two primary challenges: characterization and modeling inputs to statistical design tools; and the evolution from statistical analysis to optimization with the subsequent increase in computation complexity. Finally, statistical timing tools and statistical optimization methods must model actual variations rather than crude and inaccurate abstractions. The composition of variations during statistical analyses and optimizations should match the methods used for initial decomposition via statistical metrology techniques during process characterization. A mismatch between composition and decomposition will introduce unnecessary error and impart dubious value to the computationally costly results.

*3. Layout and physical design challenges*— First, the complexity of design rules checking is increasing (include graph). Rules have evolved to a two-tier system (required versus suggested rules) and may need to evolve either to a three-tier system or to a no-tier system (where rules are not pass-fail but provide a point in a curve and the designer needs to apply criteria for tape-out). All this will need to be done without increasing designer-perceived complexity. Second, the limitation in lithography hardware resolution will require design flows to more explicitly account for the impact of Resolution Enhancement Techniques (RET). RET tools, such as OPC and CMP fill, must become explicitly aware of circuit metrics such as timing and power. Such awareness aligns the tools to overall product goals and enables yield enhancements, manufacturing cost reductions, and mask data preparation time. As an example, OPC would be applied only to features in critical timing paths. This entails a tighter flow integration to communicate circuit intent downstream, and to avoid independent modifications by several tools leading to incorrect results.

*4. Yield prediction and optimization as design challenge*— The ground rules as specified are not longer 'hard numbers', where designer and EDA tools used to consider as strict 'hard' numbers. To archive reasonable mature yields and steep ramp capability a strategy for meaningful relaxation have to be used. These 'recommended' rules have been derived from the interaction of the design layout and wafer process requirements, such as alignment tolerances, optical proximity corrections, RET enhancements and many other constrains. DFM measures acts different for each design because it impacts power, area and speed as you tune for yield. During design process the mutual interaction off yield, area, power and speed must be analyzed and a commercial useful trade off must be found. This involvement of DFM measures effecting the functional yield and parametric yield must be integrated as a new optimization feature into the design flow and tools, not as post processing that allows only limited results and is time consuming. That means that yield prediction has to be integrated into design tools for Synthesis and Place & Route to consider all design targets to optimize simultaneously yield, performance, power, signal integrity and area (Yield as explicit new design target).

Table A Design For Manufacturability Challenges ADDED

| *Challenges ≥50 nm/Through 2009* | *Summary Of Issues* |
|---|---|
| Mask cost | ALL—CAD tools that explicitly account for mask cost in their algorithms |
| Data explosion | ALL—RET tools that are explicitly aware of circuit metrics (timing, power) |
| Limitations of lithography hardware resolution | ALL—RET-aware CAD tools, radically-restricted design rules, including yield prediction and optimization |
| Voltage supply and threshold variability | ALL—Statistical analysis tools and flows that account for this variability, including yield prediction and optimization |
| BEOL Planarization and dimensional variability | ALL—Statistical design tools and flows that account for this variability; including yield prediction and optimization |
| Heterogeneous components (analog, MEMS, eRAM) | |
| Leakage as a limiter of manufacturability | ALL—Statistical leakage analysis and optimization tools |
| *Additional Challenges <50 nm/Beyond 2009* | |
| Leakage as a limiter of manufacturability | ALL—Statistical leakage optimization tools |
| Uncontrollable CD and doping variability | ALL—Statistical parametric optimization tools |
| Extreme device and circuit variability | ALL—Architectures fundamentally resistant to variability (redundancy, ECC) |
| Critical need to integrate RET into design | ALL—RET-aware CAD tools |
| Package, system, and software variability | ALL—adaptable and redundant circuits that can adapt to these variations |
| BEOL planarization and dimensional variability | ALL—post-tapeout RET that interacts with synthesis, timing, and P&R |

*This table summarizes challenges to the design process advances implied by the above four trends. Each challenge is labeled with a list of the most relevant system drivers (S—system on chip, P—microprocessor, A—analog/mixed-signal, M—memory).*

### VARIABILITY MODELING AND ROADMAP

Since variability is expected to be the source of multiple critical DFM challenges, a systematic method to roadmap required and/or desired variability trends will become a crucial component of the overall Design roadmap. It may also allow for design-manufacturing "co-roadmapping", which may help gather indications of whether our industry should invest in variability reduction or in design productivity improvements. The requirement for such a framework is underlined by the sensitivity of variability-related information for industry participants. Since the design community needs to view variability from their parameter-based perspective, such a variability framework needs to be multi-level, i.e., it needs to cover multiple levels of design abstraction.

A Variability Roadmap Framework (VRF) is being developed by the Design TWG as illustrated in Figure 19. In this Framework the following three levels of abstraction have initially been considered:

- *Circuit/chip level*: this is the abstraction level most relevant to designers. Ideally, timing and power consumption variability for a given circuit would be road-mapped at this level, based on lower-level parametric variations.
- *Device level*: since circuits are composed of devices, at this level device-related parameters are roadmapped, such as the threshold voltage $V_t$, or the off-current $I_{off}$.
- *Physical level*: this is the level closest to the design-manufacturing interface, where parameters such as CD or effective device length ($L_e$), and actual doping level ($N_A$) are road-mapped. These parameters' variability stems from some of challenging issues enumerated above, including lithography hardware resolution limitations, and the inability to control the exact number of dopants in a channel.

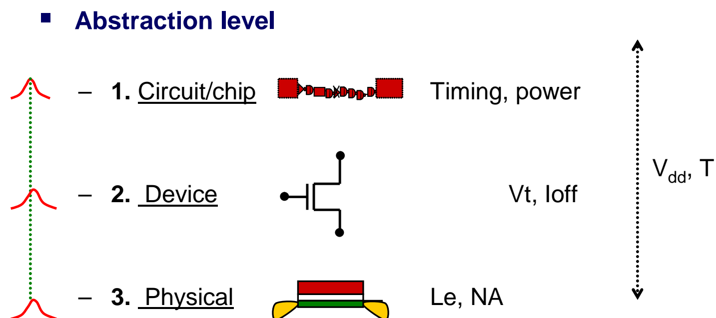



*Figure a Possible variability abstraction levels* ADDED

In this framework, the modeling process can be symbolically described as

$$\Delta\ outputs = model\ (\Delta\ inputs) \qquad [1]$$

Based on published approximated models, this model is based on a simplified gate + wire circuit slice where parameters are decomposed into gate-related and wire-related parameters. Performance is analytically modeled in the current model as total delay, which in turn is decomposed into gate delay and wire delay. Delay variability is modeled as a delta that is statistically computed from a statistical distribution of individual delay values that stems from “simulating” distributions of the input parameters in a Monte Carlo style. Model inputs include gate channel length and width, and oxide thickness; wire width, length, and thickness; ILD thickness, and wire sheet resistance.

Among other applications, this framework will be used for circuit performance variability trending. For example, estimated delay variability will be estimated over time for two possible scenarios: 10% required CD (actual channel feature length) variation, and 20% required CD variation. In this particular illustration, the model could indicate that circuit performance variability does not seem to vary very significantly with the CD tolerance requirement, which might suggest the possibility of a relaxation of the requirement.

*Table 13 Near-term Breakthroughs in Design Technology for AMS*

| *Field Of Breakthrough* | *2003* | *2004–2005* | *2006–2007* |
|---|---|---|---|
| **Specification, validation** | **Mixed-signal description languages** | **Multi-language support** | **Complete specification-driven design flow** |
| **Architectural design** | **Algorithm-oriented design** | **Language-based performance evaluation** | **Synthesizable AMS description** |
| **Mixed A/D and RF physical design** | **Procedural layout generation** | **Design centering, performance estimation** | **Constraint-driven synthesis: behavior to layout** |
| **Parasitic extraction, modeling, simulation** | **EMI simulation** | **2D/3D modeling, order reduction** | **Fault-tolerant circuit architectures, robustness** |

*Table 14 Design Process Challenges*

| *Challenges ≥50 nm/Through 2009* | *Summary Of Issues* |
|---|---|
| Silicon Complexity: devices and interconnects | All—Exponential increase in leakage power<br>S, P, A—Power density and distribution<br>All—Technology and library characterization<br>S, P, A—High-frequency noise analysis<br>S, P, A—Transmission-line interconnects<br>All—eDRAM, eFPGA, SiGe, optical, MEMS |
| System Complexity: number of states, design diversity | S, P—Verifying systems with exploding number of states<br>S, P, A—Concurrent multi-factor analysis and optimization<br>S, P—Scalable algorithms<br>S, A—Design and test of mixed analog and digital designs<br>All—Complex package analysis<br>S, A—Integrating A/D tools |
| Design Productivity | S—Integrating third party components<br>S, P—Design tool interoperability<br>S, P—Early analysis and verification methods |
| Time-to-Market | S, A—Support for platform-based design<br>S, P, A—Exploiting parallel processing |
| *Challenges <50 nm/Beyond 2009* | |
| Manufacturability | All—Cross-chip variability<br>All—Sub-wavelength mask correction (AltPSM, OPC, RET)<br>All—Design for yield and manufacturability<br>All—Standards for sharing design and manufacturing data |
| System-level Design | S—Common hardware/software (HW/SW) representation<br>S—SW synthesis, HW/SW optimization |

*This table summarizes challenges to the design process advances implied by the above four trends. Each challenge is labeled with a list of the most relevant system drivers (S—system on chip, P—microprocessor, A—analog/mixed-signal, M—memory).*

Table 14 summarizes challenges to the design process advances implied by the above four trends. Each challenge is labeled with a list of the most relevant system drivers (S = SOC, P = MPU, A = AMS, M = Memory). The remainder of this section gives explanatory comments.

*Table 15 System-level Design Challenges*

| *Challenges ≥50 nm/Through 2009* | *Summary Of Issues* |
|---|---|
| System complexity | S—Higher-level abstraction and specification<br>S—Dynamism and softness<br>S, A—System-level reuse<br>S, A, P—Design space exploration and system-level estimation<br>S—Efficiency of behavioral synthesis and software compilation<br>S—Automatic interface synthesis |
| System power consumption | P, S—Energy-performance-flexibility trade-offs<br>P, S—Novel data transfer and storage techniques |
| Integration of heterogeneous technologies | S, P, A—Codesign (HW–SW, chip-package-board, fixed-reprogrammable)<br>A, S—Non-scalability of analog circuits; analog behavioral modeling and synthesis<br>P, S, A—Top-down implementation planning with diverse fabrics (digital, AMS, RF, MEMS, EO, SW) |
| Embedded software | S—SW–SW codesign onto highly programmable platforms<br>S—System capture and abstraction<br>S—New automation from high level description to HW–SW implementations, including SW synthesis<br>S—Formal verification for SW<br>S—HW–SW coverification |
| Links to verification, test and culture | S—Integration-oriented verification and test architectures<br>S, P, A—Divergent design practices and cultures |
| *Additional Challenges <50 nm/Beyond 2009* | |
| System complexity | S, P—Communication-centric design and network-based communications on chip<br>All—Design robustness |
| System power consumption | P—Non-scaling of centrally organized architectures<br>S—Building large systems from heterogeneous SOCs |
| Integration of heterogeneous technologies | S—Total system integration including new integrated technologies (MEMS, electro-optical, electro-chemical, electro-biological or organic) |

*This table summarizes challenges to the design process advances implied by the above four trends. Each challenge is labeled with a list of the most relevant system drivers (S—system on chip, P—microprocessor, A—analog/mixed-signal, M—memory).*

*Table 16 Logical, Circuit, and Physical Design Challenges*

| *Challenges ≥50 nm/Through 2009* | *Summary Of Issues* |
|---|---|
| Efficient and predictable implementation | P—Scalable, incremental analyses and optimizations<br>P, S—Unified implementation/interconnect planning and estimation/prediction<br>P, S—Synchronization and global signaling<br>S, A, P—Heterogeneous system composition<br>P, S—Links to verification and test |
| Variability and design-manufacturing interface | P, S, A—Uncertainty of fundamental chip parameters (timing, skew, matching) due to manufacturing and dynamic variability sources<br>All—Process modeling and characterization<br>P, S—Cost-effective circuit, layout and reticle enhancement to manage manufacturing variability |
| Silicon complexity, non-ideal device scaling and power management | P, S—Leakage and power management<br>All—Reliability and fault tolerance, soft error<br>P, S—Analysis complexity and consistent analyses / synthesis objectives |
| Circuit design to fully exploit device technology innovation | P—Support for new circuit families that address power and performance challenges<br>P, S—Implementation tools for SOI<br>A, S—Analog synthesis |
| *Additional Challenges <50 nm/Beyond 2009* | |
| Efficient and predictable implementation | S—Reliable, predictable fabric- and application-specific silicon implementation platforms<br>S—Cost-driven implementation flows |
| Variability and design-manufacturing interface | P, S, A—Increasing atomic-scale variability effects |
| Silicon complexity, non-ideal device scaling and power management | P—Recapture of reliability lost in manufacturing test |
| Circuit design to fully exploit device technology innovation | P, A—Increasing atomic-scale effects<br>P, S, A—Adaptive and self-repairing circuits<br>A, S—Low-power sensing and sensor interface circuits; micro-optical devices |

*This table summarizes challenges to the design process advances implied by the above four trends. Each challenge is labeled with a list of the most relevant system drivers (S—system on chip, P—microprocessor, A—analog/mixed-signal, M—memory).*

## DESIGN VERIFICATION UPDATED

Design verification is the task of establishing that a given design accurately implements the intended behavior. The prevailing view by much of the semiconductor industry is that design verification plays a relatively minor supporting role to the design process. (Possibly, such a view is reflected in the very structure of this document.) Current reality, however, is markedly different. Verification has become the dominant cost in the design process. On current projects, verification engineers outnumber designers, with this ratio reaching two or three to one for the most complex designs. Design conception and implementation are becoming mere preludes to the main activity of verification.

This unfortunate situation is the result of two processes. First, the functional complexity of modern designs is increasing at a breathtaking pace. Design size is growing exponentially with Moore's Law. In the worst case, functional complexity, as measured by the number of distinct states of the system that must be verified, can grow exponentially in the size of the design, producing a doubly exponential blow-up.[1] Second, the historically greater emphasis on other aspects of the design process has produced enormous progress (automated tools for logic synthesis, place-and-route, and test pattern generation, etc.), leaving verification as the bottleneck. Without major breakthroughs, verification will be a non-scalable, show-stopping barrier to further progress in the semiconductor industry.

The overall trend from which these breakthroughs will emerge is the shift from ad-hoc verification methods to more structured, formal processes. The mainstream methodology used in industry today attempts to verify the functionality of a system design by repeatedly building models, simulating them on an ad hoc selection of vectors, and then patching any bugs that happen to be triggered. However, such a trial-and-error verification methodology lacks the metrics that make the verification process predictable and repeatable. More structured approaches relay on planning the verification effort, modeling and measuring coverage as verification tasks are being executed. This deliberate verification strategy makes use of simulation engines at the unit, chip and system level as well as formal engines at the unit level.[2] Formal verification techniques are starting to gain momentum and they are complementing simulation methods in the process of verification. Formal methods span multiple aspects of verification: from a formalized verification process, to a formal notation for specification and verification and formal solvers and proof techniques.

In tackling the complexity of very complex systems, and in particular, systems-on-a-chip, a "hierarchical" verification methodology seems to be needed, along with the development of specific methods and tools associated with each level of the hierarchy. In this context, the design and implementation pyramid can be summarized roughly as follows: 1) Module level (structural, dataflow and behavioral module description and synthesis) where the verification tools must prove the correctness of systems in the range of 10K to 500K logic gates. 2) Sub-System level (for the most part, this is the result of structural composition of modules) producing systems with 100k to 10M logic gates. 3) System/Full chip level (obtained by composing sub-systems) producing designs of 1M to more than 100M logic gates. In a hierarchical verification approach different verification techniques would be used at each design level: At the module level, the goal is to produce high-quality modules through the formal verification of properties and assertions. For sub-systems, verification would entail a mix of property verification for the interface and simulation with constrained random pattern generation. Property verification at the sub-system level could be achieved in the future, for instance, through the definition of aggressive automatic design-abstraction techniques, and of automatic techniques for the refinement of such abstractions, driven by counter-examples, which would allow proving sub-system properties while posing a minimum demand on the verification engineering resources. At the system level, simulation-based assertion checking paired with coverage metrics seem the most appropriate among the techniques available. Technological progress depends on developing rigorous and efficient methods to achieve these verification goals (the bulk of formal and semi-formal verification research today), and eventually codifying reliable, predictable engineering practices that simplify many verification challenges. Tables 17 and 18 summarize the main challenges in design verification. The remainder of this section provides explanatory comments.

---

[1] *There are many variations that arrive at this conclusion. For example, a new design that requires doubling the number of transistors on a chip is likely to also double the number of latches on the chip, which likely means roughly squaring the number of reachable states of the design. This analysis assumes that the correct behavior can be verified by examining the set of reachable states of the system, or a similar computation. If verifying correct behavior requires reasoning over sequences of states, the computational challenge will be even worse.*

[2] *This argument applies to hardware emulation as well as to conventional simulation. Hardware emulation buys several orders of magnitude improvement in "simulation" speed, providing an invaluable aid to verification. It also often allows an earlier start to system integration and software development. An emulation system, however, still runs vectors one-at-a-time, so it cannot possibly provide a scalable, long-term solution to the verification problem.*

*Table 17a Design Verification Challenges—Near-term UPDATED*

| *Challenges ≥50 nm/Through 2009* | *Summary Of Issues* |
|---|---|
| Increased verification capacity | S, P, A—Verification complexity is double-exponential in design size<br>S, P, A—Need high coverage as well as capacity to handle large designs<br>S, P, A—Semi-formal techniques |
| Robust verification tools | S, P, A—Highly unpredictable verification algorithms demand improved heuristics and characterizations of problem difficulty |
| Verification metrics | S, P, A—Code, structural and functional coverage<br>S, P, A—Realistic bug models are needed, along with algorithms to determine bug coverage |
| Software verification | S—Software intrinsically more difficult to verify<br>S—Traditional software verification techniques inapplicable<br>S—Integrated hardware/software systems<br>S—Design for verifiability |
| Verification reuse | S—Must allow reuse of verification of IP blocks<br>S—Specify abstract behavior of IP blocks<br>S—Specify environmental constraints of IP blocks<br>S—Hierarchical verification algorithms |
| Verification methodology | S, P—Different cost-benefit trade-off (higher cost acceptable)<br>S, P—Need exceptionally high capacity<br>S, P—Must be very predictable due to long design cycle and pipelined development teams |
| Design for verifiability | S,P—Will be necessary sooner than for other system drivers; specialized techniques are likely |
| Greater concurrency | S,P—Far more concurrency in new processors greatly increases verification complexity |

*This table summarizes challenges to the design process advances implied by the above four trends. Each challenge is labeled with a list of the most relevant system drivers (S—system on chip, P—microprocessor, A—analog/mixed-signal, M—memory).*

### DESIGN VERIFICATION CHALLENGES—NEAR TERM (>50 nm)

Many of the most important challenges for verification are relevant to most or all system drivers. In the near-term, eight primary issues (1–8 below) center around making formal and semi-formal verification techniques more reliable and controllable. In particular, major advances in the capacity and robustness of formal verification tools are needed, as well as meaningful metrics of the quality of verification. In the longer term, four primary issues (9–12 below) center around raising the level of abstraction and broadening the scope of formal verification. These longer-term issues are actually relevant now, but the near-term challenges are already crises. In general, all of the verification challenges apply to SOC. In addition, the following near term challenges are especially important for SOC. MPUs present a distinct verification challenge, both because of their leading-edge complexity, and because of the unique economics of an incredibly complex design that is produced in incomparably high volumes. As a result, different, domain-specific verification challenges and opportunities exist, both near- and long-term.

*1.* *Capacity*—Figure 19 relates the current landscape of available verification tools and design granularity. The horizontal axis presents increasing design complexity, from module components, to sub-systems, multiple modules connected together providing some high-level functionality, to systems, complete systems-on-a-chip integrating multiple functionalities on a single die, and to the application software running on the system; the vertical axis indicates the verification techniques available. Conventional simulation-based methods[3] span a large range; they are practically used

*[3] "Conventional simulation" here refers to verification techniques based on simulating possible behaviors of the system one-at-a-time. "Formal verification" refers to any techniques, such as symbolic simulation, symbolic trajectory evaluation, model checking, and theorem proving, that provide the effect of an exhaustive analysis of all possible behaviors of the system. An occasionally useful distinction can be drawn between theorem-proving approaches, which tend to provide the greatest expressive and analytical power at the expense of requiring substantial human expertise, versus the other approaches, which tend to trade off theoretical power for greater automation. "Semi-formal" refers to a broad range of techniques that attempt to handle larger designs by sacrificing complete, formal coverage, usually by blending techniques from formal verification and conventional simulation.*

for anything from module to full-system verification. However, the quality of the results varies: For small modules it is possible to run many simulation vectors in a short time and thus achieve a fairly good coverage of the design under verification. At the system level the performance of simulation is very low, since the complexity of the algorithms involved is linear on the design size and the number of simulation vectors. Thus, for very complex systems, only a few simulation vectors can be run within a reasonable amount of time, and the fraction of design coverage they provide is even much smaller considering the complexity of the system that is being simulated. Formal verification can achieve very high coverage, granted that a broad set of properties are in place, targeting the various aspects of the system. The downside of these techniques is their limited scalability that usually cannot go above the module level. Semi-formal verification attempts to blend formal and simulation-based techniques. As the Figure shows, they present generally better capacity, sometimes at the cost of lower design coverage. For very complex systems and when the project involves software development, software tools are insufficient to provide any confidence in the correctness in the system. Solutions used at this level include FPGA-based rapid prototyping and/or a silicon production of the hardware. The challenge is to create new solutions that together provide high coverage at all levels of complexity. The two orthogonal methodologies of formal verification and simulation in use today have both important downsides: while formal tools can only handle small to medium size designs, simulation-based tools can simulate designs of almost arbitrary complexity, but they provide a vanishingly small coverage even with extremely long simulation times. Emulation and rapid hardware prototyping perform several orders of magnitude faster than software logic simulators, thus providing the ability to achiever higher coverage. However, the improvement is only a constant factor and does not scale at the same rate as design complexity.

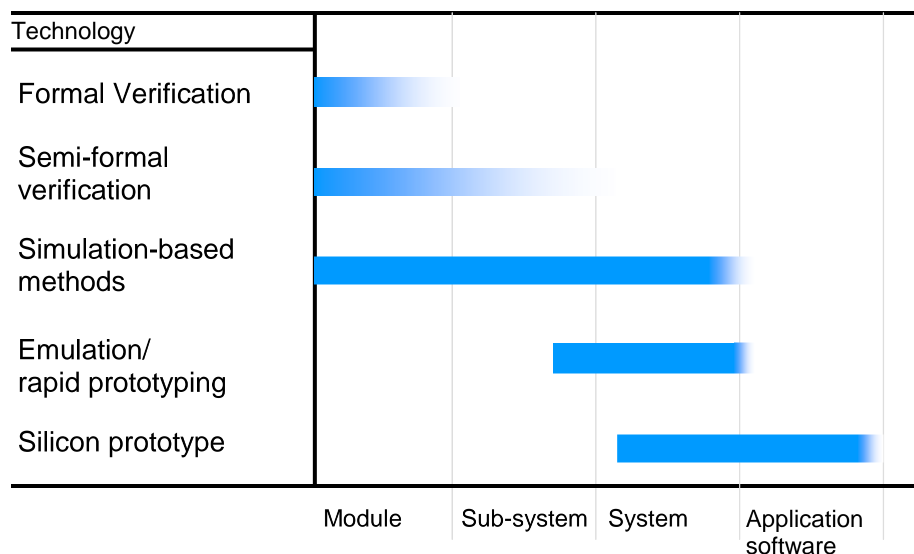



*Figure b Current Verification Tool Landscape* ADDED

2. *Robustness*— A crucial aspect of current verification solutions that is not shown in Figure 19 is their reliability. On one hand, simulation-based methods are fairly predictable, because their execution time per simulation vector scales linearly with design complexity. Their performance, thus, will linearly decrease as we move towards the most complex design levels. Emulation techniques present a similar trend, while silicon prototypes are consistently reliable. On the other hand, formal verification techniques depend on highly temperamental heuristics, in order to cope with the complexity of the verification problem. For any given pair of design and verification algorithm, even an expert can be hard-pressed to determine whether the verification algorithm will complete. Common measures of problem size, such as transistor, gate, or latch counts, correlate only vaguely with formal verification complexity; it is easy to find designs with less than one hundred latches that defy all known verification methods, as well as designs with thousands of latches that can be verified easily. Such unpredictability is not acceptable in a production environment. A crucial verification challenge is to make the verification process more robust. This can take the form of either improved heuristics for the verification algorithms, or improved characterization of the difficulty of verifying a given design, leading to methodologies for easy-to-verify design.

*3. Verification metrics*— An important near-term verification challenge is the need to quantify the quality of the verification effort. In particular, a meaningful notion of coverage is needed. Multiple types of coverage metrics are available, each with their own benefits and limitations:

- Code coverage is concerned with measuring which fraction of the design source code has been stimulated during simulation (for example, lines of code, expression coverage). Variants of line coverage are concerned with covering all components of expressions in the code, or all cases in switch statements, etc…The limitation of line coverage is in its projecting a dynamic simulation execution to a static source code, thus missing many potential problems that could be still present because of all the possible paths of execution available during simulation.

- Structural coverage, such as state or pair arc coverage. State coverage focuses on the finite state machines of the design and measures which states have been covered by simulation. However, it is generally not possible to consider the whole design as a single finite state machine, rather only a few machines are considered by coverage. The downside is that while the coverage provides good results for single machines, combinations of states resulting from products of multiple machines are not considered.

- Functional coverage targets each of the design's functionalities. Since these differ for each distinct design, the specific definition needs to be provided by the verification team based on the design's specification. The quality of the result is thus dependent also on the quality of the coverage definition.

Code and structural coverage metrics quantify aspects of the designs that do not necessarily imply its correctness. Functional coverage potentially could achieve this goal of measuring the fraction of a design's functionality that has been verified. However, the challenge is in the definition of a good functional coverage. In the current landscape, there is a lack and a need for a common foundation metric for functional coverage in a unified environment, so that achieving a certain level of coverage in a design could have a meaning beyond the specific functional coverage used in that particular design. Moreover, there is no general methodology in designing the functional coverage units; an abstract fundamental model to use as a guideline during the development is needed and could support the development methodology. Ultimately, there is a need for a defect model for functional bugs, much like there is a defect model for testing, to support the development of coverage and a unified metric for verification.

*4. Software*— The verification of complex systems, such as systems-on-a-chip (SOC) entails the verification of the hardware components, of the hardware/software interface and of the application software running on the system. The software components of an SOC can be divided into application software, running over an operating system, or, in general, a layer that is hardware-independent, and a lower-level hardware-dependent software, such as drivers, etc. Because in these systems the software layer could provide much of the functionality, a major challenge in SOC verification is how to verify software and the hardware/software interface. Presently, software development is not as rigorous as hardware development in terms of design reviews, analysis tools, and testing. Software is intrinsically harder to verify: it has more complex, dynamic data and an enormous state space. Techniques used today include "on-chip-verification", that is, running the software on a production version of the hardware components to verify the software. While this technique is required by the intrinsic complexity of software, and allows very fast simulation, its downside is that software verification can only happen very late in the design cycle. Classical formal techniques for software verification are still too labor-intensive to be widely applicable for SOC, to attack systems with such large state spaces, and require very aggressive abstraction techniques. The verification of the hardware/ software interface is a challenge on its own, since it requires verifying the two domains together, a task of enormous complexity. To make this task manageable, there is a need for techniques that provide a proper abstraction of the interface activity, verification techniques to check the correctness of the abstracted driver layers, and assertion checking for drivers' invariants at the non-abstract level. The near-term challenge will be to develop techniques that allow verification of even elementary and low-level pieces of software. The longer-term challenge will be to develop robust verification methods for software, and hardware/ software interfaces, as well as an understanding of design-for-verifiability as applied to software.

*5. Reuse*—Pre-designed IP blocks promise to allow assembling SOCs of unprecedented complexity in a very short time. The major challenge is to develop the corresponding verification methodology to allow rapid verification of a system assembled from pre-designed (and pre-verified) blocks. Key issues are how to rigorously and completely describe the abstract behavior of an IP block, how to describe the environmental constraints assumed by the IP block, and how to exploit the hierarchy to simplify verification. Some IP components have started to ship with associated verification IPs, from standard protocols verification IPs, to abstract models for general IP blocks, protocol checkers to check environmental constraints surrounding the block, to transaction generators. However, these are still preliminary attempts,

while there is a need for consistent availability of verification IPs associated with any IP block in order to develop a reuse methodology. Besides verification components associated with IP blocks, there is some availability of independent verification IPs, such as environment generators for specific protocols. Near-term progress will most likely occur for standardized IP interconnects, such as on-chip buses, but the general problem for arbitrary IP block interfaces must be eventually solved.

*6. Specialized verification methodology*— The biggest issue in design verification is that all currently known algorithmic solutions are running out of capacity with respect to the designs being developed today. The only foreseeable way to overcome this issue in the short term is with an adequate verification methodology. Current trends in this direction include coverage-driven verification, both for simulation-based verification and semi-formal verification, more in use today than in the past; specification documents in formal notation, that make easily available the set of formal properties to be verified in the implementation; coverage model templates. Many challenges are still to be solved to obtain a sufficiently robust and complete methodology. For instance, there is a need for ways to obtain consistent abstraction techniques of design components, interfaces, etc. that do not drop key aspects of the design in the abstraction, and, obviously, these aspects depend on the context where the abstraction is used. Formal specifications are starting to be developed; however, a major challenge is the completeness of such specifications, a challenge that becomes even more compelling in a world where different IP components in the same system are developed by completely unrelated design teams. The acceptance of new verification methodologies is progressing slowly, even the basic deployment of formal properties in a design is a challenge since it often requires a global understanding of some specific aspect of a design, which involves additional effort on the development team. Finally, verification methodology is ultimately an evolving target that can only provide a risk reduction to the development, not a guarantee of correctness through a well-defined recipe.

*7. Specialized design-for-verifiability*— To enable more effective verification, a few specialized activities in design for verifiability are already foreseeable. For instance, facilities for software and/or hardware debug can be put in silicon. In this direction there is preliminary work being done on self-checking processors, in which a small watchdog processor verifies the correct execution of the main processor. For mixed-signal designs, the insertion of loop-back modes to bypass the analog portion of the design allows to verify the system as a fully digital design. The use of synchronizers, in particular at the software level forces tasks not to proceed independently past the synchronization points, creating checkpoints for verification and reducing the searchable state space. In MPU designs synchronizers would reduce the complexity of verifying speculative execution systems. The challenges in this area will be the development and the adoption of design-for-verifiability techniques for a few major domains. Efforts are also made in developing design methodologies by incremental refinement to produce systems that are correct-by-construction; however, it is not clear how automatic the refinement process can be made, while, on the other hand, manual intervention is a potential source of design errors.

*8. New kinds of concurrency*—As MPU designs become more complex, new kinds of concurrency become important. Already, many of the bugs that elude verification relate to cache coherence and other concurrency issues. New designs greatly complicate the verification process by increasing the level of concurrency via techniques such as chip-level multiprocessing and on-chip cache coherence protocols, and simultaneous multithreading. In the future, new kinds of concurrency both at the intra-processor level and in multi-processor systems, or in other hardware context will present difficult challenges to verification. There is a need for new models of failure to grasp this additional level of complexity. The solution will probably require a mix of hardware and software techniques to reduce the complexity of the interactions and make the concurrent protocols verifiable.

*Table 17b Design Verification Challenges—Long-term UPDATED*

| *Additional Challenges <50 nm/Beyond 2009* | |
|---|---|
| Design for verifiability | S, P, A—New methodology needed<br>S, P, A—Characterize and minimize performance and area impact |
| Higher levels of abstraction | S, P, A—New algorithms needed<br>S, P, A—Complexity of designs enabled by higher-level design<br>S, P, A—Equivalence checking vs. RTL |
| Specification for verifiability | S, P, A—Specifications of correctness will become unmanageable<br>S, P, A—Need to understand what kinds of specifications are most understandable<br>S, P, A—Need to consider how to make specifications modular and modifiable |
| Verification in the presence of non-digital effects | S, P, M, A—Hybrid systems verification for analog effects<br>S, P, M, A—Hybrid systems verification for analog properties<br>S, P, M, A—Verification of probabilistic systems |
| Heterogeneous systems | A, S—How to model, analyze, and verify MEMS, EO devices, and electro-biological devices |
| Analog-Mixed signal | A—Extremely primitive state-of-the-art forces difficult hybrid-systems issues into near term |
| Soft failures | S, P, M—Reliable systems in face of soft errors |
| Verification of redundancy | S, P, M—Sufficient reliability in face of soft errors |

*This table summarizes challenges to the design process advances implied by the above four trends. Each challenge is labeled with a list of the most relevant system drivers (S—system on chip, P—microprocessor, A—analog/mixed-signal, M—memory).*

### DESIGN VERIFICATION CHALLENGES—LONG TERM (<50 nm)

*1. Design for verifiability*— As the solutions to the near-term challenges produce understandings of what is easy or hard to verify and how design errors occur, the longer-term challenge arises of how to codify that understanding into producing easy-to-verify designs. Without design-for-verifiability, it is unlikely that verification will be tractable for the designs envisioned beyond 2007. Major changes in methodology may be required, and some performance degradation is likely. A useful analogy is to sequential testability, where the computational intractability of sequential ATPG has resulted in near-universal adoption of scan-based testing.

*2. Higher levels of abstraction*— As design moves to a level of abstraction above RTL, verification will have to keep up. The challenges will be to adapt and develop verification methods for the higher-levels of abstraction, to cope with the increased system complexity made possible by higher-level design, and to develop means to check the equivalence between the higher-level and lower-level models. This longer-term challenge will be made much more difficult if decisions about the higher-level of abstraction are made without regard for verification (e.g., languages with ill-defined or needlessly complex semantics, or a methodology relying on simulation-only models that have no formal relationship to the RTL model).

*3. Specification for verifiability*—A continuing challenge for design verification is how to specify the desired behavior of a design. Current available notations for specification are not powerful enough to approach this problem in a generalized way. A deeper understanding of what makes a specification clear or opaque, modifiable or intractable will be needed to guide development of languages that are used to specify ever more complex designs. For instance, there is a need for automatic ways to check the self-consistency of a specification document, so that different specifications don't state conflicting requirements. In addition, specific training is needed for designers to use these specification notations and to be able to develop formal specifications consistently.

*4. Verification in presence of non-digital effects*— To date, design verification has mainly focused on the discrete behavior of digital systems. The dual challenges of silicon complexity and system complexity will force future verification efforts to analyze a broader class of aspects. The complexity of silicon integrated circuit systems is making the clean, digital abstraction of a VLSI system increasingly precarious. Analog electrical effects will impact performance and, eventually, functionality. The existing simulation methodology (SPICE) for analyzing these effects is too slow, and

may become unreliable as smaller devices become increasingly sensitive to process variations. In this direction there is preliminary work being done on architectural simulation of microprocessors, in which the high-level architectural simulator, interacts with a low-level context-specific simulator to acquire metrics such as timing and voltage, which are then fed back for overall system evaluation with minimal performance impact. In the long term, formal techniques will be needed to verify these issues at the boundary of analog and digital, treating them as hybrid systems.[4] Similarly, at the highest-levels of design, system complexity dictates that future verification tasks will likely require specifying and verifying analog and probabilistic behaviors (such as quality of service guarantees in a network processor). Thus, there will be the challenges of hybrid systems and probabilistic verification.

*5. Heterogeneous systems*— The development of new technologies that are placed side-by-side to a digital design in a silicon die presents a whole set of new challenges. Examples are MEMS, electro-optical devices, and electro-biological devices. These new components will require modeling of both the interface between the digital portion and the non-digital components and a proper abstraction of the non-digital system behavior in order to still be able to verify the digital portion of the system.

*6. Analog-Mixed signal*— Today analog systems are mostly verified through classic systems analysis tools for continuous systems, through system modeling and analysis in the frequency domain. The bulk of verification happens in the post-design phase, by verifying test dies with analog lab equipment. Mixed-signal designs undergo separate verification activities for the digital and analog portions. In the future, mixed-signal systems will become a more relevant fraction of all silicon developments, bringing the development of proper verification methodologies in this arena to a critical level. The challenge in this area is in tying the verification of the analog portion of a system with its digital counterpart. One of the requirements in achieving this goal is in bridging the current performance gap between digital and analog simulation.

*7. Soft failures*— In the long term there is a need to develop verification techniques that can provide information on a system's reliability in presence of soft errors.

*8. Verification of redundancy*— The quantifiable evaluation of the reliability of a redundant system, and the estimation of the level of redundancy needed to achieve a certain level of reliability is particularly important for safety-critical application. In the future, the increased complexity of electrical effects in a system performance and correctness will make these evaluations even more important, even for non safety-critical designs.

---

[4] *Hybrid systems have both complex discrete behavior (e.g., finite-state machines) as well as complex continuous behavior (e.g., differential equation models). The discipline borrows techniques from both discrete formal verification as well as classical control theory.*

*Table 18 Design Test Challenges*

| *Challenges ≥50 nm/Through 2009* | *Summary Of Issues* |
|---|---|
| Effective speed test with increasing core frequencies and widespread proliferation of multi-GHz serial I/O protocols | P, S—Continuation (avoidance) of at-speed functional test with increased clock frequencies<br>P, S—At-speed structure test with increased clock frequencies<br>P, S, A—DFT, test and on-chip measurement techniques for multi-gigahertz serial I/Os and non-deterministic interfaces |
| Capacity gap between DFT/test generation/fault grading tools and design complexity | P, S—Better EDA tools for advanced (open, delay, etc.) fault models<br>P, S—DFT to enable low-cost ATE<br>P, S—Non-intrusive logic BIST (including advanced fault models)<br>A—AMS DFT/BIST, especially at beyond-baseband frequencies |
| Quality and yield impact due to test process and diagnostic limitations | P, S—Power and thermal management during test<br>P, S—Fault diagnosis and design for diagnosability<br>S—Yield improvement and failure analysis tools and methods<br>All—Increasing difficulty to fault isolate and root cause yield limiting defects |
| Signal integrity testability and new fault models | P, S—Signal integrity (noise, interference, capacitive/inductive coupling, etc.) testability<br>A—Fault models for analog (parametric) failures |
| SOC and SIP test | S—Integration of SOC test methods into chip-level DFT<br>S—Integration of multiple fabric-specific test methodologies in cost-effective manufacturing flows<br>A—DFT, BIST and test methods compatible with core-based SOC environment and constraints<br>M—Embedded memory (DRAM, SRAM, Flash) built-in self-diagnosis and self-repair<br>All—Test reuse in context of higher integration |
| *Additional Challenges <50 nm/Beyond 2009* | |
| Integrated self-testing for heterogeneous SOCs and SIPs | A—Test of multi-gigahertz RF front ends on chip<br>S—Use of on-chip programmable resources for SOC and SIP self-test<br>S, A—Dependence on self-test solutions for SOC (including RF and analog)<br>A— (Analog) signal integrity test issues caused by interference from digital to analog circuitry<br>S—Test methods for heterogeneous SOC and SIP including MEMS and EO components |
| Diagnosis, reliability screens, and yield improvement | A—Diagnosis and failure analysis for AMS parts<br>P, S—Electrical automated fault isolation techniques below gate level<br>P, S—Design for efficient and effective burn-in to screen out latent defects<br>P, S—Quality and yield impact due to test equipment limits<br>P, S—New timing-related fault models for defects/noise |
| Fault tolerance and on-line testing | P, S—DFT and fault tolerant design for logic soft errors<br>S—Logic self-repair using on-chip reconfigurability<br>S—System-level on-line testing |

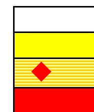
*This table summarizes challenges to the design process advances implied by the above four trends. Each challenge is labeled with a list of the most relevant system drivers (S—system on chip, P—microprocessor, A—analog/mixed-signal, M—memory).*

Table 19 Additional Design Technology Requirements UPDATED

| | Year of Production | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2012 | 2015 | 2018 | Driver |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Technology Node | | hp90 | | | hp65 | | | | | | |
| | DRAM ½ Pitch (nm) | 100 | 90 | 80 | 70 | 65 | 57 | 50 | 35 | 25 | 18 | |
| | MPU/ASIC ½ Pitch (nm) | 107 | 90 | 80 | 70 | 65 | 57 | 50 | 35 | 25 | 18 | |
| | MPU Printed Gate Length (nm) | 65 | 53 | 45 | 40 | 35 | 32 | 28 | 20 | 15 | 10 | |
| | MPU Physical Gate Length (nm) | 45 | 37 | 32 | 28 | 25 | 22 | 20 | 14 | 10 | 7 | |
| Add | Complexity --> (system) % of memory | | 47% | 50% | 53% | 55% | 58% | 61% | 65% | 65% | 65% | |
| WAS | SOC new design cycle (months) | 12 | 12 | 12 | 12 | 12 | 12 | 11 | 11 | 10 | 9 | SOC |
| IS | Productivity --> design cycle (mo) [1] | 12 | 12 | 12 | 12 | 12 | 12 | 11 | 11 | 10 | 9 | SOC |
| WAS | SOC logic Mtx per designer-year (10-person team) | 1.9 | 2.5 | 3.3 | 4.3 | 5.4 | 7.4 | 10.6 | 24.6 | 73.4 | 113 | SOC |
| IS | Productivity --> Logic Mtx per designer-year (10-person team [2] | 1.9 | 2.5 | 3.3 | 4.3 | 5.4 | 7.4 | 10.6 | 24.6 | 73.4 | 113 | SOC |
| Add | Productivity --> % software in design cycle (mo[3]) | --- | 60% | 60% | 60% | 60% | 60% | 60% | 60% | 60% | 60% | SOC |
| WAS | SOC dynamic power reduction beyond scaling (X) | 0 | 0.1 | 0.1 | 0.2 | 0.2 | 0.2 | 0.2 | 6 | 4.7 | 8.1 | SOC |
| IS | SOC dynamic power reduction beyond scaling (X) [4] | 0 | 0.1 | 0.1 | 0.2 | 0.2 | 0.2 | 0.2 | 6 | 4.7 | 8.1 | SOC |
| WAS | SOC standby power reduction beyond scaling (X) | 0.37 | 1.4 | 2.4 | 3.4 | 5.1 | 6.4 | 8.73 | 18.8 | 44.4 | 232 | SOC |
| IS | SOC standby power reduction beyond scaling (X) [5] | 0.37 | 1.4 | 2.4 | 3.4 | 5.1 | 6.4 | 8.73 | 18.8 | 44.4 | 232 | SOC |
| WAS | %Test covered by BIST | 20 | 20 | 25 | 30 | 35 | 40 | 45 | 60 | 75 | 90 | MPU, SOC |
| IS | Manufacturing interface --> %logic Test covered by BIST [6] | 20 | 20 | 25 | 30 | 35 | 40 | 45 | 60 | 75 | 90 | MPU, SOC |
| Add | Design cost --> improvement, normalized[7] | --- | 100% | 86% | 120% | 56% | 78% | 105% | 143% | 149% | 155% | SOC |
| Add | Area density --> increase (X) [8] | --- | 1 | 1.4 | 2 | 2.8 | 4 | 5.6 | 15.6 | 43.8 | 122.8 | SOC |
| Add | Area density --> increase beyond scaling (X) [9] | --- | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SOC |

Mtx—Million transistors

Manufacturable solutions exist, and are being optimized
Manufacturable solutions are known
Interim solutions are known
Manufacturable solutions are NOT known



Added Notes for Table 19:

[1] Number of months from design requirements definition to tape-out.
[2] Number of logic transistors in a chip divided by the number of person-years in a design team, for a 10-person team.
[3] % of software design effort as the number of software development person-years versus the total number of person-years.
[4] % of dynamic power savings on top of the % of active power savings derived from technology scaling..
[5] % of leakage power savings on top of the % of leakage power savings derived from technology scaling.
[6] % of the logic gates' stuck-at faults covered by BIST.
[7] % improvement in design cost per year for low-power PDA SoC driver. It accounts for all design technology improvements.
[8] Increase in absolute area density (number of gates per unit chip area.
[9] Increase in area density on top of density improvement coming from technology scaling (feature reduction).

*Table 20 Design Technology Improvements and Impact on Designer Productivity*

| *DT Improvement* | *Year* | *Productivity Delta* | *Productivity (Gates/Desn-Year)* | *Cost of Component Affected* | *Description of Improvement* |
|---|---|---|---|---|---|
| **None** | **1990** | | **4K** | | |
| **In-house place and route** | **1993** | **+38.9%** | **5.55K** | **PD**<br>**Integration** | **Automated block placement and routing.** |
| **Engineer** | **1995** | **+63.6%** | **9.09K** | **Chip/circuit/PD**<br>**Verification** | **Engineer than can pursue all required tasks to complete a design block, from RTL to GDSII.** |
| **Reuse—small blocks** | **1997** | **+340%** | **40K** | **Circuit/PD**<br>**Verification** | **Blocks from 2,500–74,999 gates.** |
| **Reuse—large blocks** | **1999** | **+38.9%** | **56K** | **Chip/circuit/PD**<br>**Integration**<br>**Verification** | **Blocks from 75,000–1M gates.** |
| **IC implementation suite** | **2001** | **+63.6%** | **91K** | **Chip/circuit/PD**<br>**Integration**<br>**EDA support** | **Tightly integrated tool set that goes from RTL synthesis to GDS II through IC place and route.** |
| **Intelligent testbench** | **2003** | **+37.5%** | **125K** | **SW development**<br>**Verification** | **RTL verification tool (“cockpit”) that takes an ES-level description and partitions it into verifiable blocks, then executes verification tools on the blocks, while tracking and reporting code coverage.** |
| **Electronic system-level (ES-level) methodology** | **2005** | **+60%** | **200K** | **SW development**<br>**Verification** | **Level above RTL, including both HW and SW design. It consists of a behavioral (where the system function has not been partitioned) and an architectural level (where HW and SW are identified and handed off to design teams).** |
| **Very large block reuse** | **2007** | **+200%** | **600K** | **Chip/circuit/PD Verification** | **Blocks >1M gates; intellectual-property cores** |
| **TOTAL** | | **+15,000%** | | | |

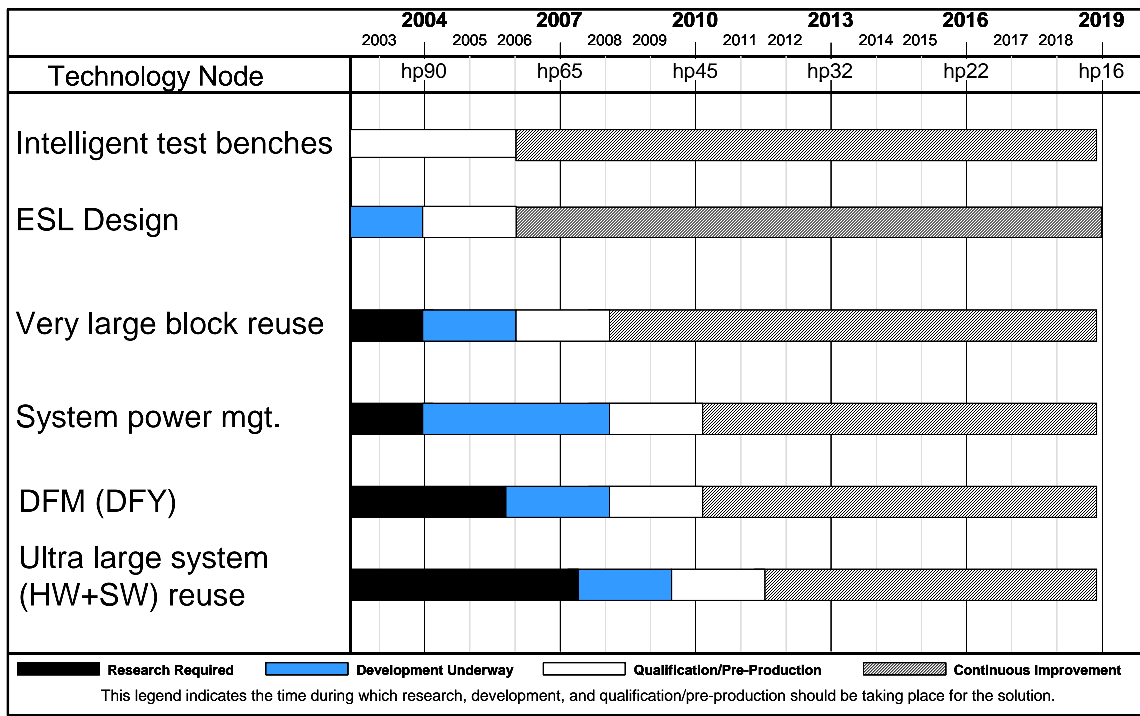



*Figure c General Design Potential Solutions* ADDED**DEFINITIONS**

*Intelligent test benches*—RTL verification tool that takes ES-level description, partitions it, verifies blocks, and tracks code coverage

*ESL design*—Level above RTL, includes HW and SW design. It consists of a behavioral (before HW/SW partitioning) and architectural level (after)

*Very large block reuse*—Blocks >1M gates; intellectual-property cores

*System power management*—Automated design of power management features applied at the system level

*DFM (DFY)*—Design-For-Manufacturability and Design For Yield automation

*Ultra large system (HW+SW) reuse*—Reuse of ultra-large blocks, including massively parallel processing HW and SW units (>10M "gates")