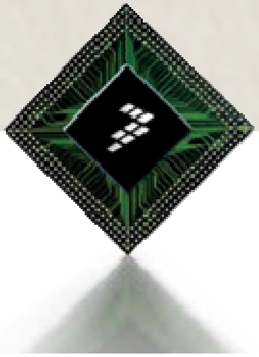
ESL Adoption at Freescale Semiconductor's Wireless Operations

Ryan Bedwell
System Level Design Manager, Modem Products Division

Ver 2

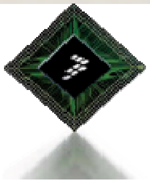


freescale™
semiconductor

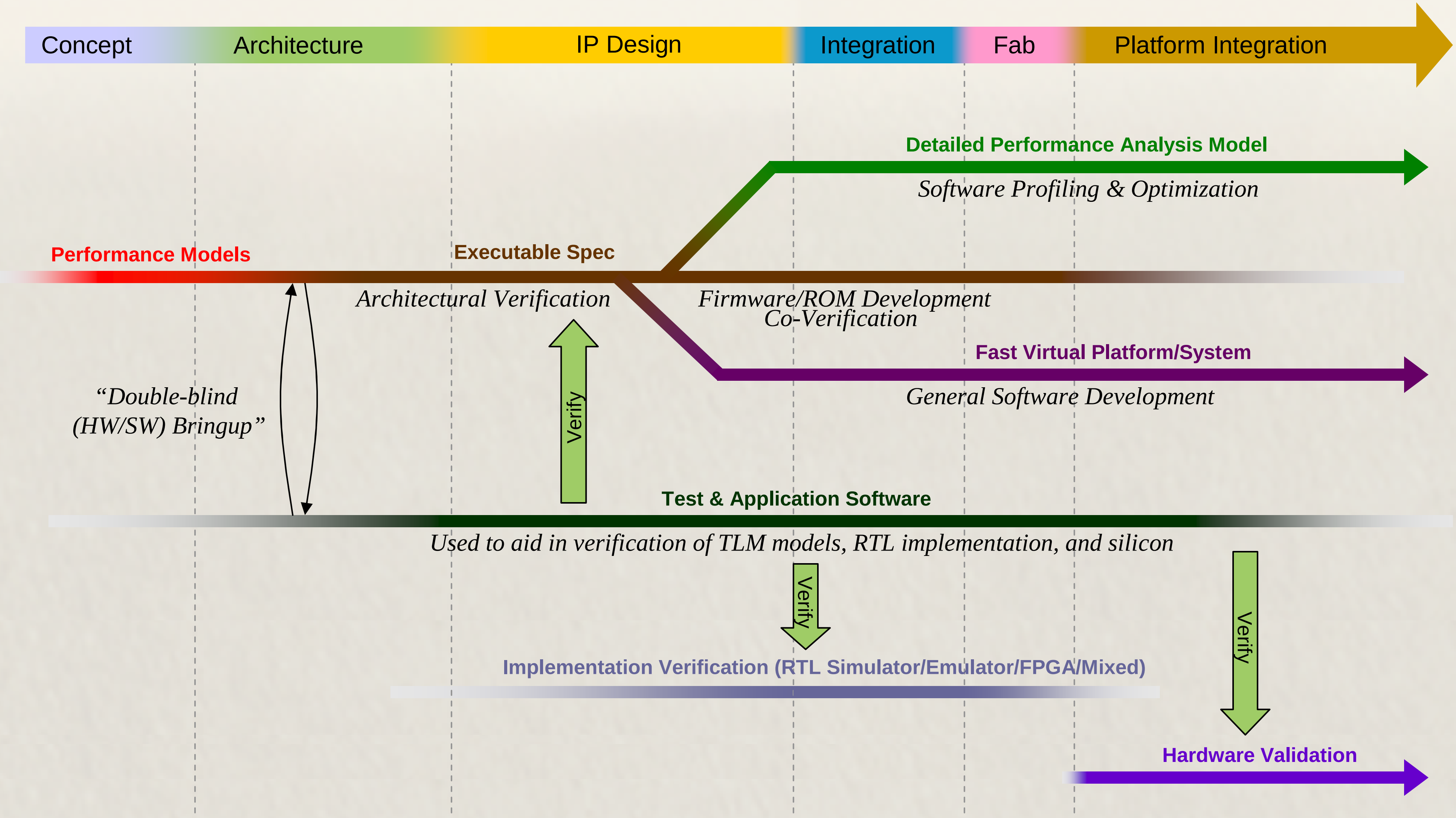
Why ESL?

- Increasing Design Complexity
  - “**Architecture by intuition**” becoming very difficult
  - Must be able to **explore tradeoffs** quickly
  - **Time-to-market** pressures further compress schedule
  - Need **verification re-use** between architecture & design
  - Industry is poised to adopt techniques for more **abstract design**
  - The **cost of mistakes** (time & money) is growing

- Increasing Importance of Software
  - Customers require **platform delivery** rather than “bag of IC’s”
  - **Software effort** is very significant for smart consumer devices
  - Market demand requires:
    - > **Pre-silicon Software** development & testing
    - > **Software/hardware integration** much earlier than previously done
    - > **Software development models** capable of many MIPS operation



ESL-Based Modeling Flow



Concept

Architecture

IP Design

Integration

Fab

Platform Integration

Detailed Performance Analysis Model

Software Profiling & Optimization

Performance Models

Executable Spec

Architectural Verification

Firmware/ROM Development
Co-Verification

Fast Virtual Platform/System

General Software Development

"Double-blind (HW/SW) Bringup"

Verify

Test & Application Software

Used to aid in verification of TLM models, RTL implementation, and silicon

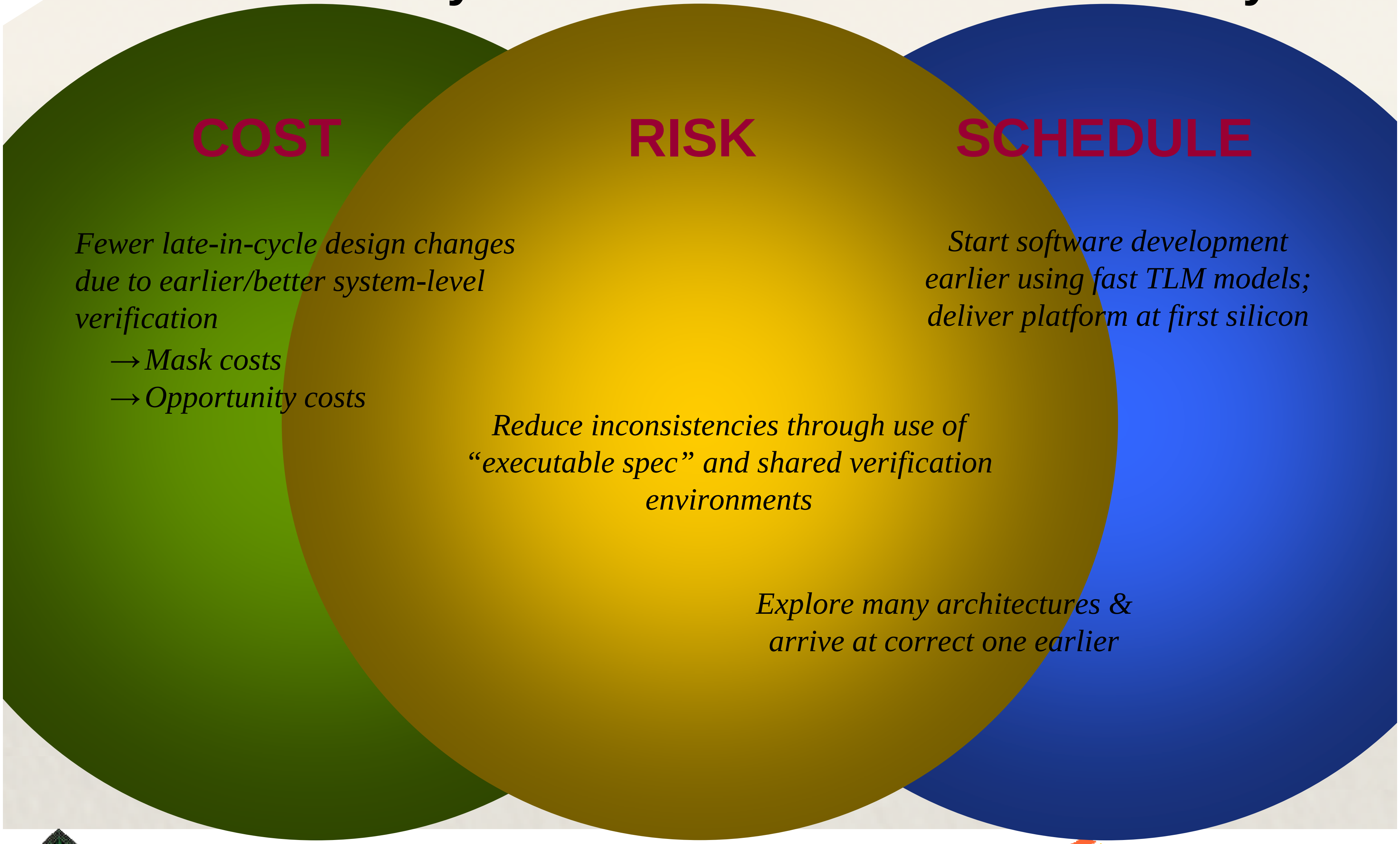
Verify

Verify

Implementation Verification (RTL Simulator/Emulator/FPGA/Mixed)

Hardware Validation

Key “Pressure Points” Addressed by ESL



COST

RISK

SCHEDULE

Fewer late-in-cycle design changes due to earlier/better system-level verification

- → *Mask costs*
- → *Opportunity costs*

Start software development earlier using fast TLM models; deliver platform at first silicon

Reduce inconsistencies through use of “executable spec” and shared verification environments

Explore many architectures & arrive at correct one earlier

“Internal” Implications …

… On Architects:

- Create & use system-level models for architectural decisions
- Work with designers on common executable spec and verification environment
- Require system-level models in external IP deliverables

… On IC Designers:

- Work on common verification environments between RTL & system-level models
- Keep abreast of developments in behavioral synthesis tools
- Prepare for eventual paradigm shifts to higher abstraction in design methodologies

… On Software Developers:

- Require “virtual system” delivery from system teams
- Use available models for pre-silicon software development
- Drive new features in models for enhanced debug & visibility capabilities

… On Management:

- Evaluate & recognize the value of system-level modeling; fully fund and staff the effort
- Require measurable benefits

“External” Implications …

… On IP Vendors:

- Make TLMs a part of standard IP delivery (at least 2 views: untimed & cycle-accurate)
- Support open APIs and standards; remain agnostic to tool vendors
- Support – and participate in – standardization efforts

… On EDA Vendors:

- Support open APIs (bus, debug, profiling, …); we cannot be locked into a specific vendor!
- Find product differentiators that work in the context of open standards, such as:
  - Fast, accurate processor core and standard IP (i.e. bus) models
  - Seamless integration with standard software development tools
  - HDL co-simulation and co-emulation tools
  - Speed enhancements

… On the SystemC™ Community:

- SystemC™ is excellent, but not enough…
- Build on TLM spec; push standards to higher levels:
  - Bus convenience APIs (specific buses and/or configurable buses)
  - Registers
  - Debug (non-disturbing) accesses
  - Profiling and measurement
- Constant attention to speed/accuracy tradeoff

Critical Success Factors

- Expand ESL efforts beyond in-house/proprietary projects
  - Use and drive standards for languages, APIs, abstraction levels, etc.
  - Staff effort appropriately with architects, IC designers & software engineers
- Provide fast virtual platforms of entire systems
  - Beyond SoC… beyond PCB… include "end-to-end" system
  - Untimed model performance ≥ 10 MIPS per host processor GHz
- Users drive & adopt industry standards
  - APIs – buses, registers, etc.
  - Abstraction levels – untimed functional, cycle-approximate, cycle-accurate, …
  - Compatibility across tools vendors!
- Link with IC design flows
  - "Executable specs"
  - Shared verification benches
  - Behavioral synthesis…