FPGA SUMMIT

Tutorial 2A. Practical Exercise: Get Your FPGA Application Up and Running

December 10, 2008

Introductions

Lee Goldberg	EN-Genius Network
Daniel Platzker	Mentor Graphics
Steve Knapp	Prevailing Technology
Dave Orecchio	GateRocket
Sanjay Thatte	Mentor Graphics
Andy Stevens	VMETRO a Curtiss-Wright Company
Dan Isaacs	XILINX

Order of Business

- **Four short presentations** (15 minutes each)
  - Design Tools and Design Portability
  - Test Tools and Equipment
  - Verification
  - From System Specification to FPGA Implementation
- **Breakout sessions on key challenges**
  - Design Methods
  - System Interface
  - Verification
- **Summary and panel session**

Design Tools/Design Portability
Tips and Tricks for Vendor Neutral FPGA Development

Daniel Platzker

FPGA Product Line Director

Mentor Graphics Corp.

Why Vendor Neutrality: Device & Tools

Important to retaining flexibility to switch vendors

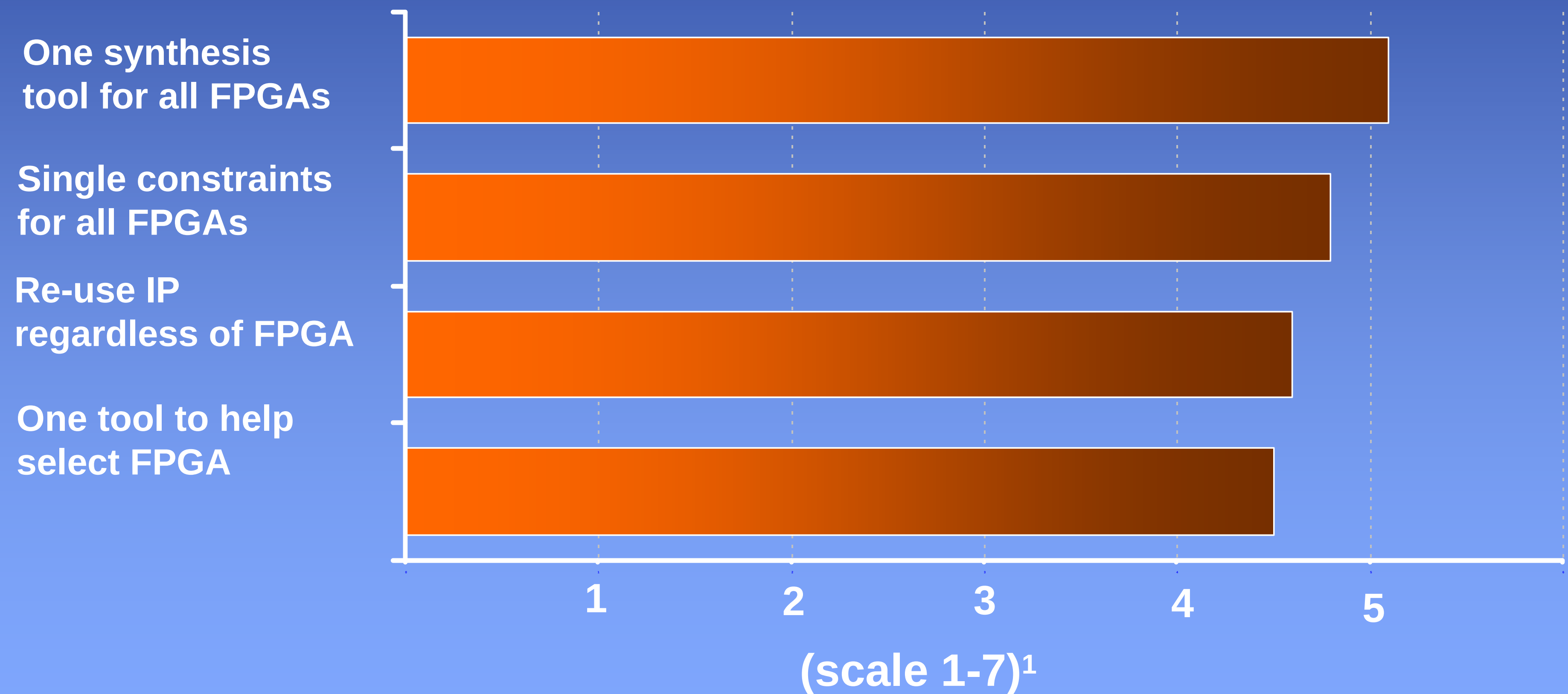
- Existing vendors leapfrog each other on capabilities/cost
- New vendors offer unique capabilities, (Achronix, eAsic)
- Retain price leverage with vendors

Single tool flow less expensive

- Reduce maintenance and training
- Objective device selection

Users Rate Importance of Vendor Independence

All attributes rate above average!



Source: Image & Market Measurement, January 2006

What is Required for Vendor Neutral Development

1. Vendors neutral tools
2. Vendor neutral methodologies
   - Industry standards
   - Vendor neutral code
   - Wrappers
3. Minimal use of vendor dependent IP
4. <u>Discipline</u>

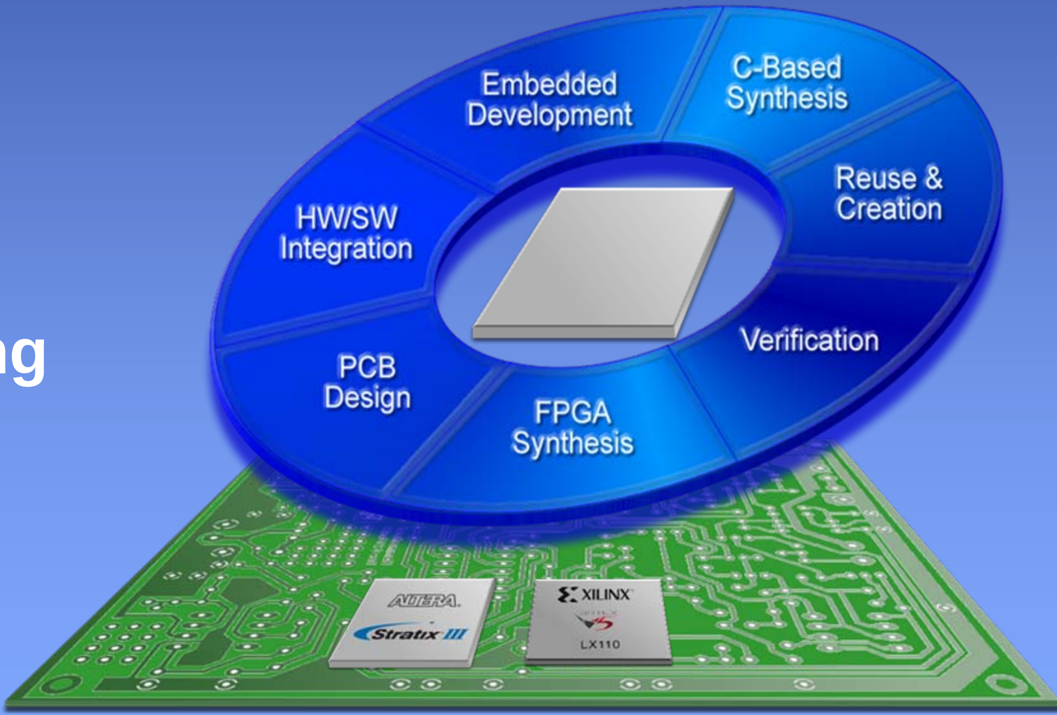
1. Vendor Neutral Tools

All tools in the flow should support multiple vendors:

- ESL
- Design Creation
- Synthesis
- Equivalence checking

The Exception

- Place & Route

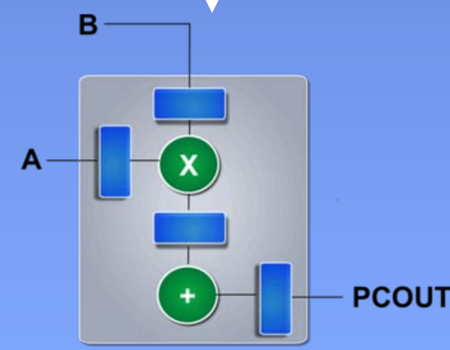


2. Vendor Neutral Methodologies

- Higher levels of abstraction
  - C++, Mathworks, SystemC
- Industry standards
  - SDC, LRM compliance
- HDL coding style
  - Inference instead of instantiation
- Wrappers for vendor dependent IP
  - DCMs, PLLs

Technology Independent Inference

```
always @(posedge clk)
begin
  mult0_result <= A * B;
  PCOUT<= mult0_result
+ PCIN;
end
```



DSP Block

3. Minimize Vendor Dependent IP

- **Best:**
  - vendor independent IP
  - In-house or 3rd party IP vendors
- **Acceptable:**
  - Everything with standard interface that can be swapped
  - Wrappers needed for some IP (e.g., DCM, PLL)
- **Challenge:**
  - Vendor CPUs (Software, peripherals)
  - Consider portable CPUs, e.g., ARM, Gaisler, Tensilica

4. Discipline

Management needs to endorse vendor neutral development tools and methodologies and stick to it

Summary

Vendor neutrality = ability to select the best FPGA for your next project, regardless of silicon vendor

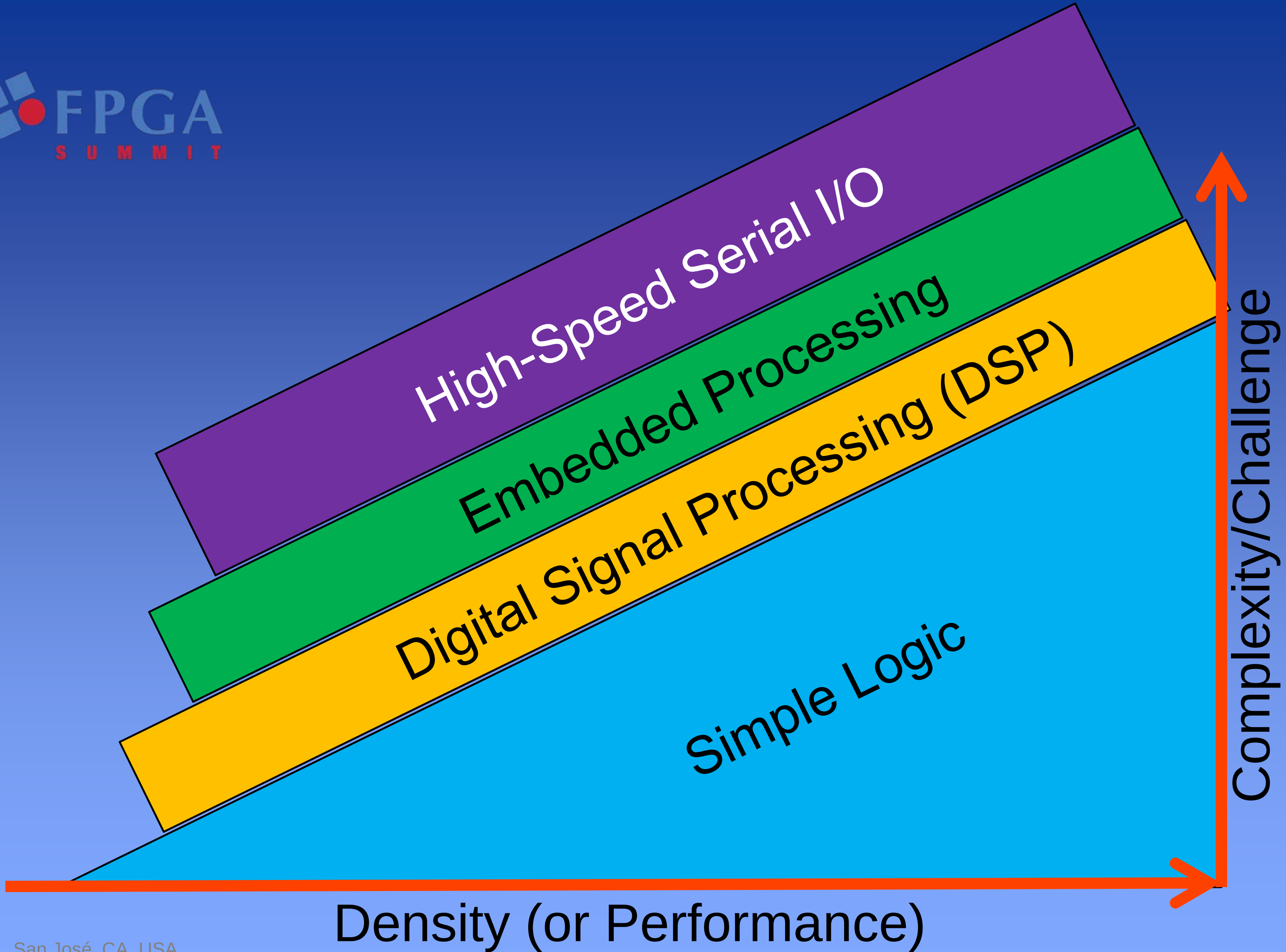
- Why?
  - Existing vendors leapfrog each other on capabilities & cost
  - New vendors offer unique capabilities e.g., Achronix, eAsic
  - Allows for price leverage with vendors
  - Single tool flow less expensive (vs. flow per vendor)
- What is necessary?
  - Vendor neutral tools, methodologies, IP, discipline

Test Tools and Equipment

Steve Knapp
Principal Engineer
steve.knapp@prevailing-technology.com

… and Everything Else

- It's obvious that you need FPGA devices and related design software
- But what else is required to be successful?
  - Development boards
  - In-system programming, download, debug cables
  - Device programmers (for some devices)
  - Logic analyzers, signal generators, oscilloscopes
  - In-system debugging tools
  - Test sockets
- The mundane minutiae of success

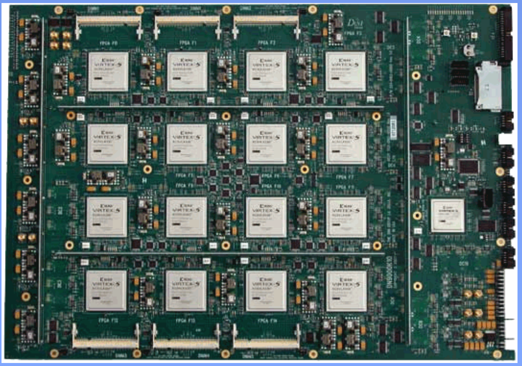

High-Speed Serial I/O
Embedded Processing
Digital Signal Processing (DSP)
Simple Logic
Complexity/Challenge
Density (or Performance)

Lab Time is Expensive!

- Find and fix bugs before you enter the lab!
- Simulate first
  - Create good quality simulation models and testbenches
  - Leverage hardware in the loop if possible
    - Improved accuracy (sometimes)
    - Shorter run times
- The difference between theory and practice
  - Simulation will not find all the bugs
  - But, it is an important and necessary first step

Development Boards

- Build or buy?
- Development boards speed development
- Generic or application specific
- Cost from $40 to $10,000+
  - Inexpensive, **<u>expendable</u>** board, one per engineer
  - More expensive, application-specific board for team



In-System Download/Debug Cables

- Supplied by FPGA vendor, although 3rd party solutions available
  - Sometimes built into development board
- Connects FPGA or associated configuration memory to PC or workstation
  - USB / Ethernet / parallel port

- Provides direct downloading, programming, and debugging capabilities
- A must have!

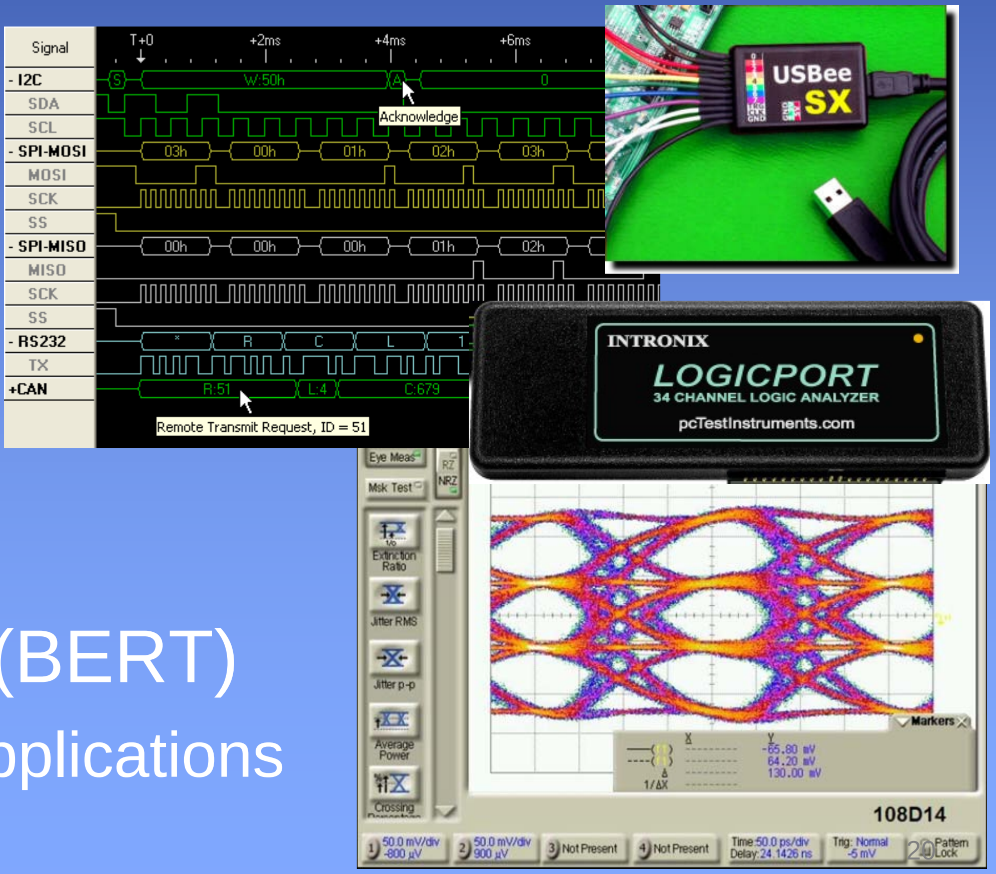
Device Programmers

- Required to program nonvolatile memory
  - Some FPGAs have internal nonvolatile memory
  - Others use a companion memory device
- Prototyping (desktop) vs. production programmer
- Value-added programming services offered by some vendors, distributors
- Don’t forget the required socket adapters

FPGA SUMMIT

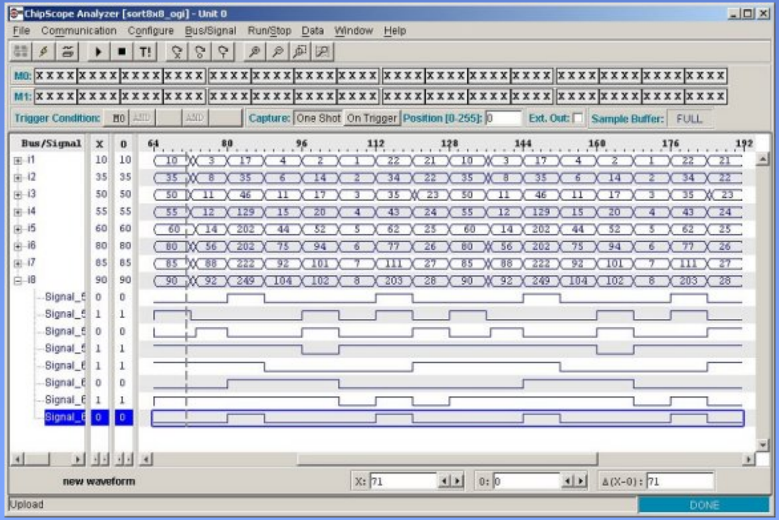
Test Equipment

- What you need depends on the complexity of application?
- Logic analyzers
- Signal generators
- Oscilloscopes
  - Differential probes
  - Jitter measurement
- Bit error rate tester (BERT)
  - High-speed serial applications



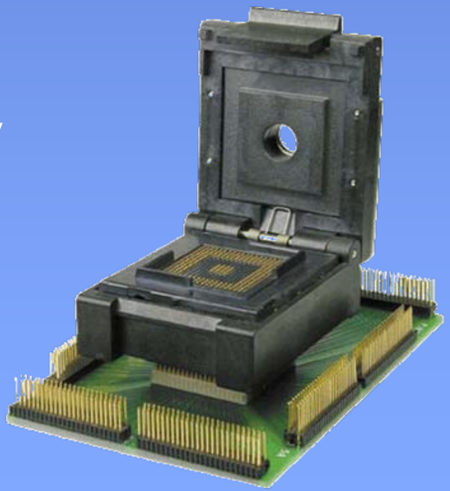
In-System Debugging Tools

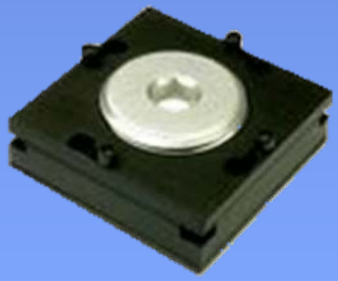
- Visibility is key to successful debugging
- Most horrors are buried deep inside FPGA
- Some vendors offer ability to view internals
  - Xilinx ChipScope
  - Altera SignalTap II
  - Altium, Synopsys,
  - National Instruments
- Debugging tools for processor-based design



Test Sockets

- Many new FPGAs use advanced, high pin-count, surface-mount packages
- Many new FPGAs are high-performance with associated signal integrity concerns
- Sockets can be challenging and difficult to find
  - Embedded Technology
  - Ironwood Electronics
  - Molex





The Key to Success?

- FPGA design is much more than just devices and design software
- These extra items do not guarantee success
- But, will make success come easier
- Skimping on this equipment will cost you far more in the long run
- Advanced capabilities like high-speed serial I/O and embedded processors require specialized tools

FPGA Summit, Tutorial T2A, Wednesday December 10, 2008

FPGA Verification

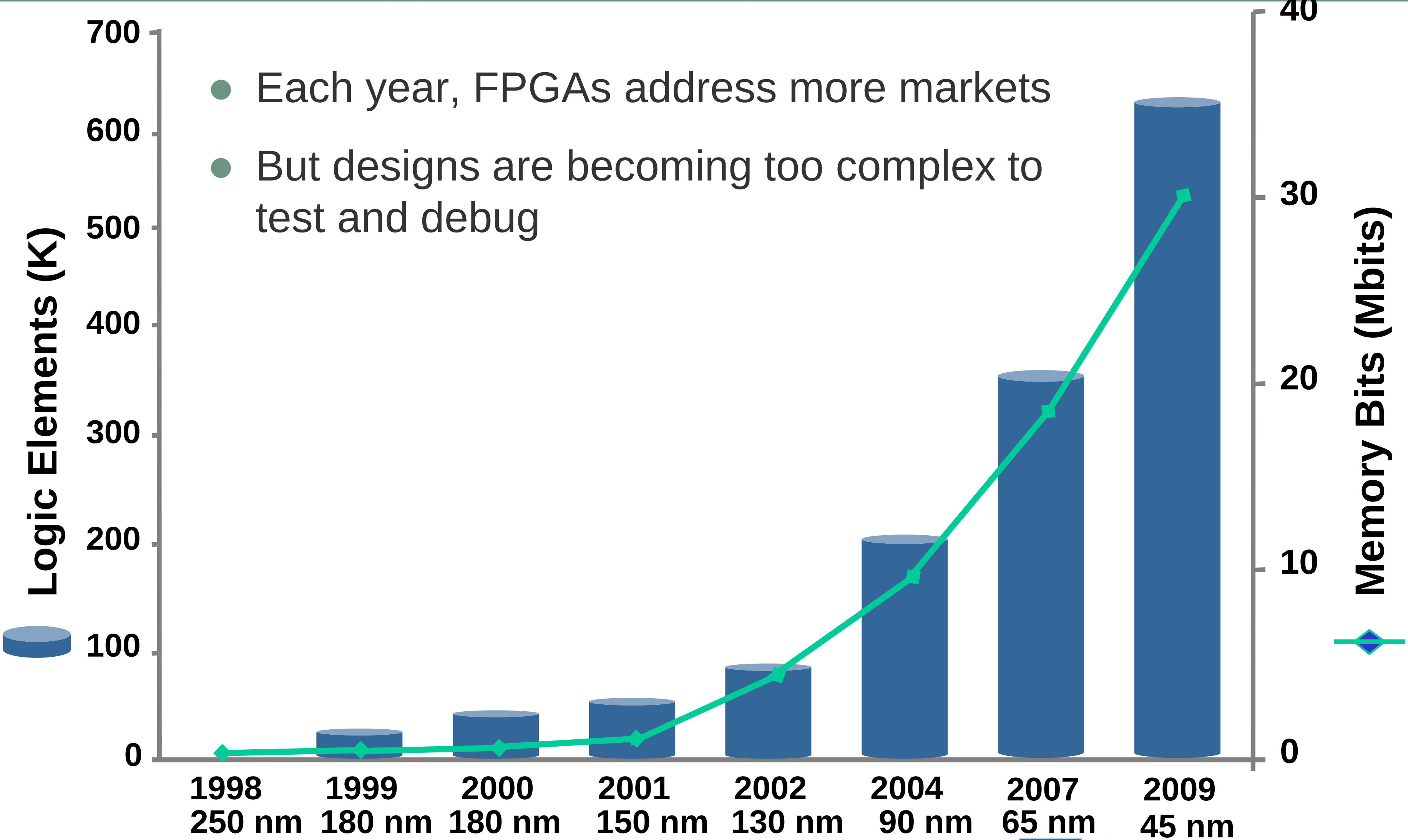
Dave Orecchio, President and CEO

GateRocket, Inc. 19 Crosby Drive, Suite 100 Bedford, MA 01730
Email: RocketDrive@GateRocket.com
Phone: +1 (781) 908-0082

Web: http://www.GateRocket.com

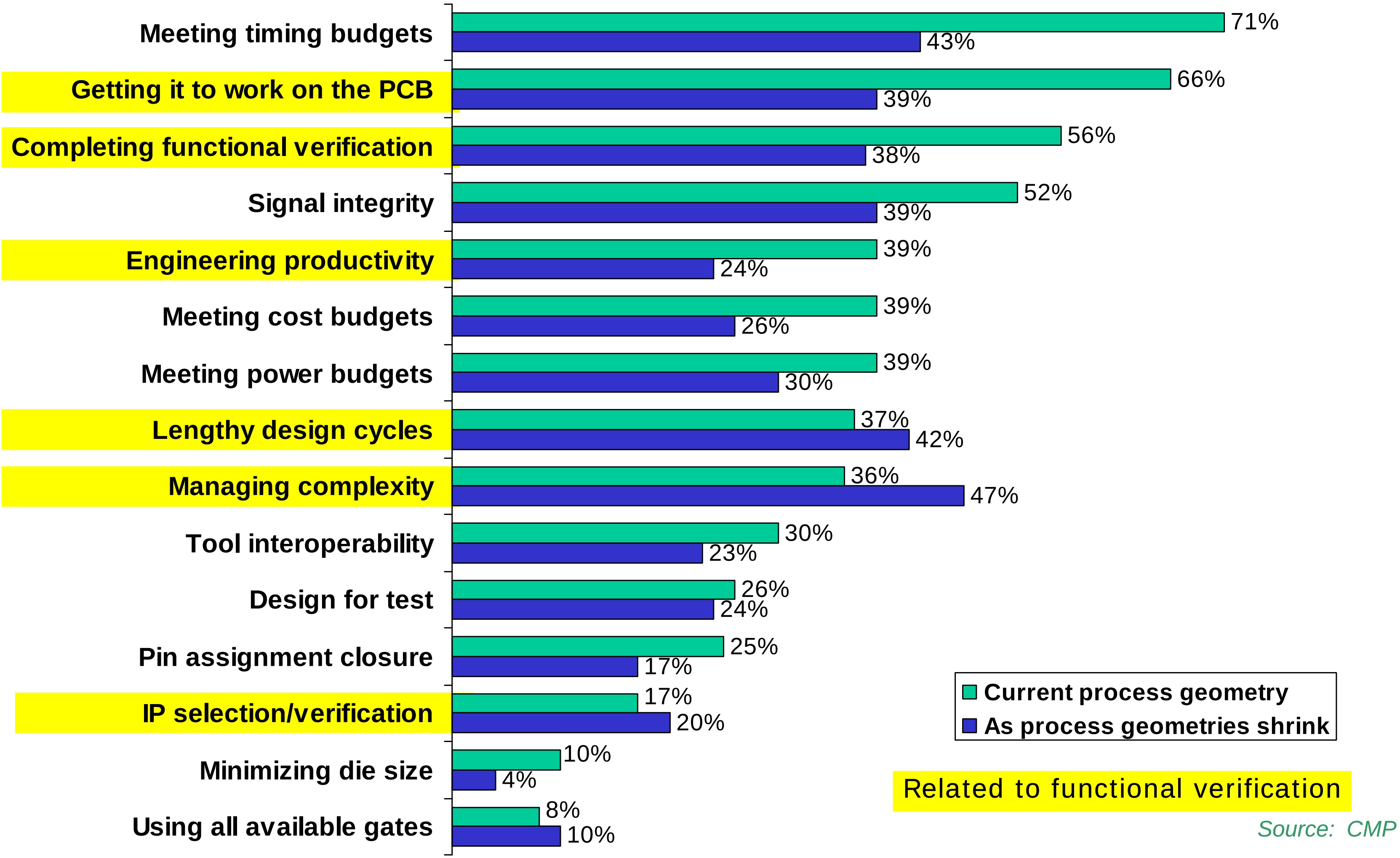
New FPGAs Approach ASIC Complexity

- Each year, FPGAs address more markets
- But designs are becoming too complex to test and debug



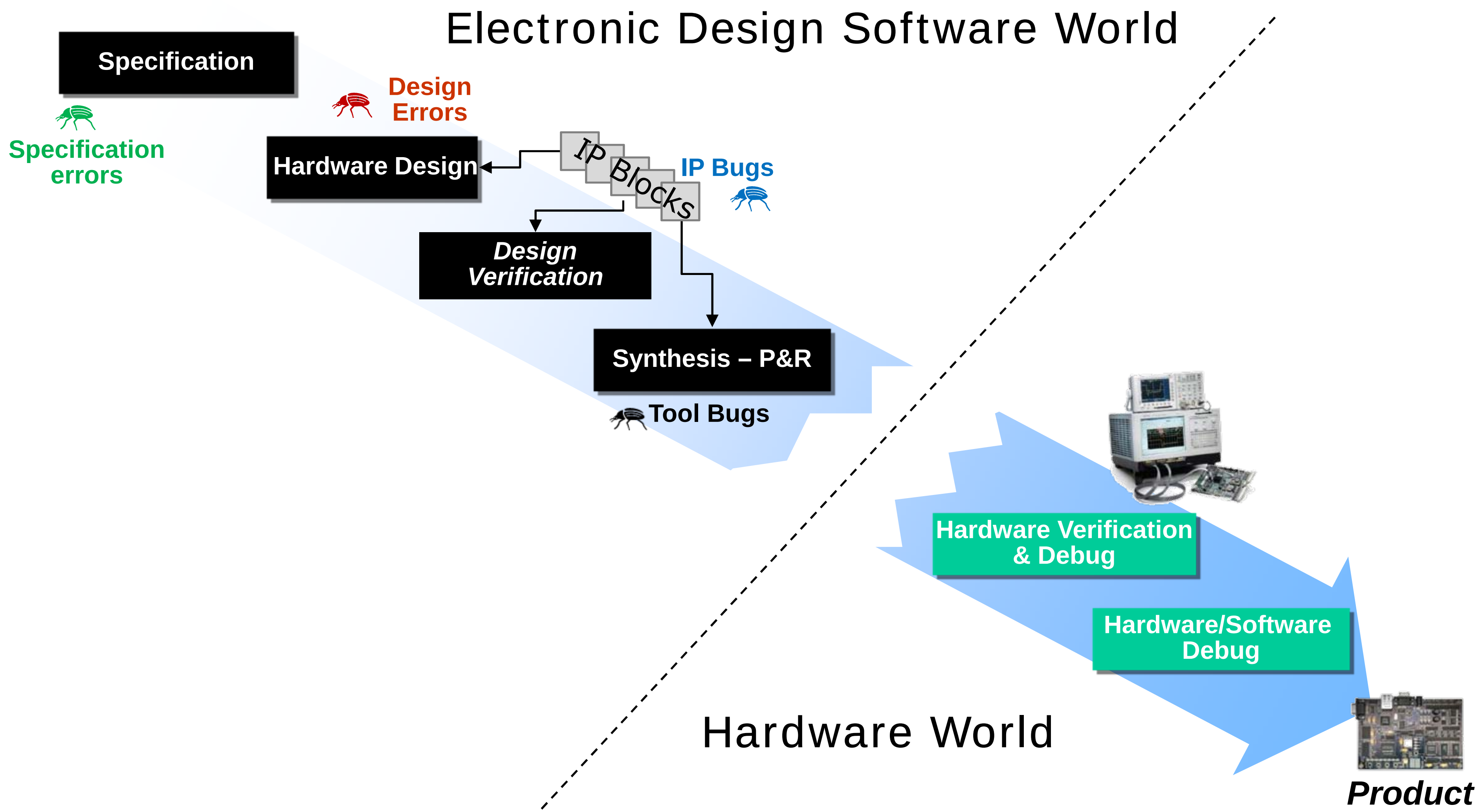
Source: *FPGA Companies*

Issues and Challenges with FPGA projects

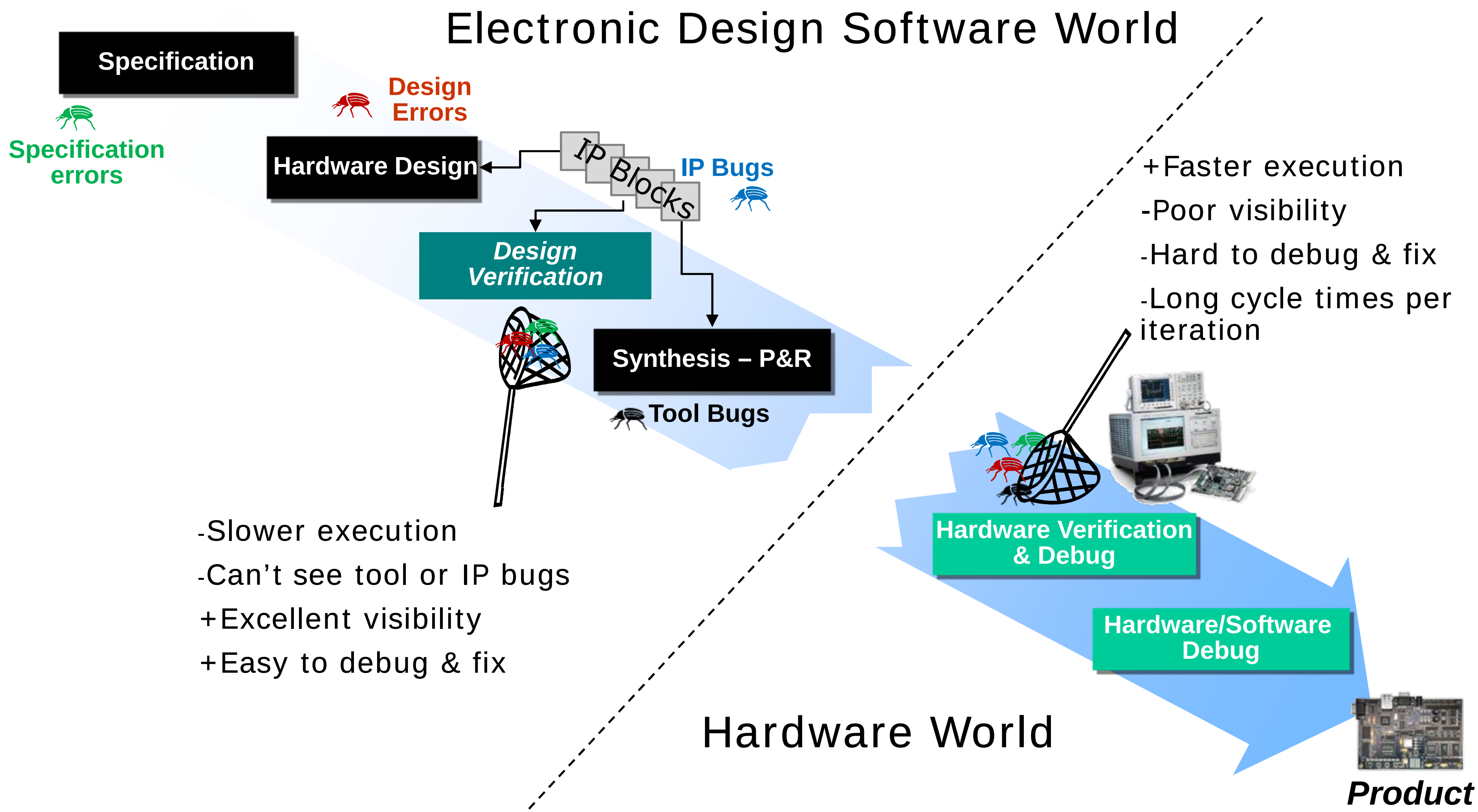


10. How critical are the following issues and challenges with your current FPGA projects? **(Select one best response for each issue / challenge)**
10a. How critical will these issues and challenges become – ***as process geometries continue to shrink***?

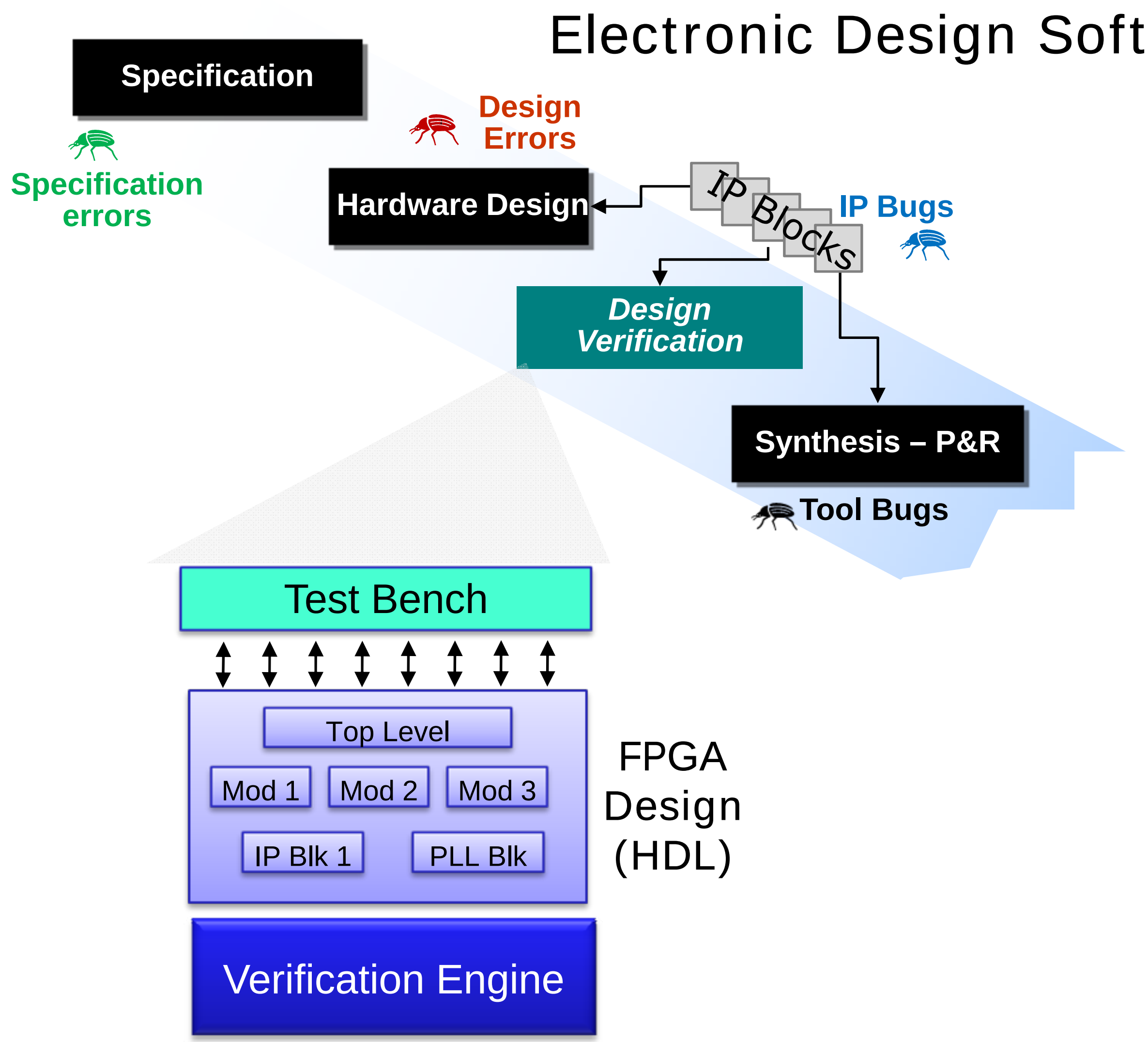
Where do errors originate?



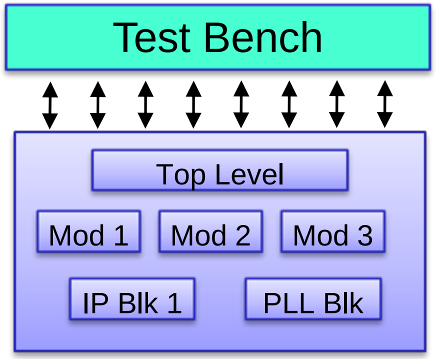
Where are errors detected?



Functional Verification = Testbench + Engine(s)



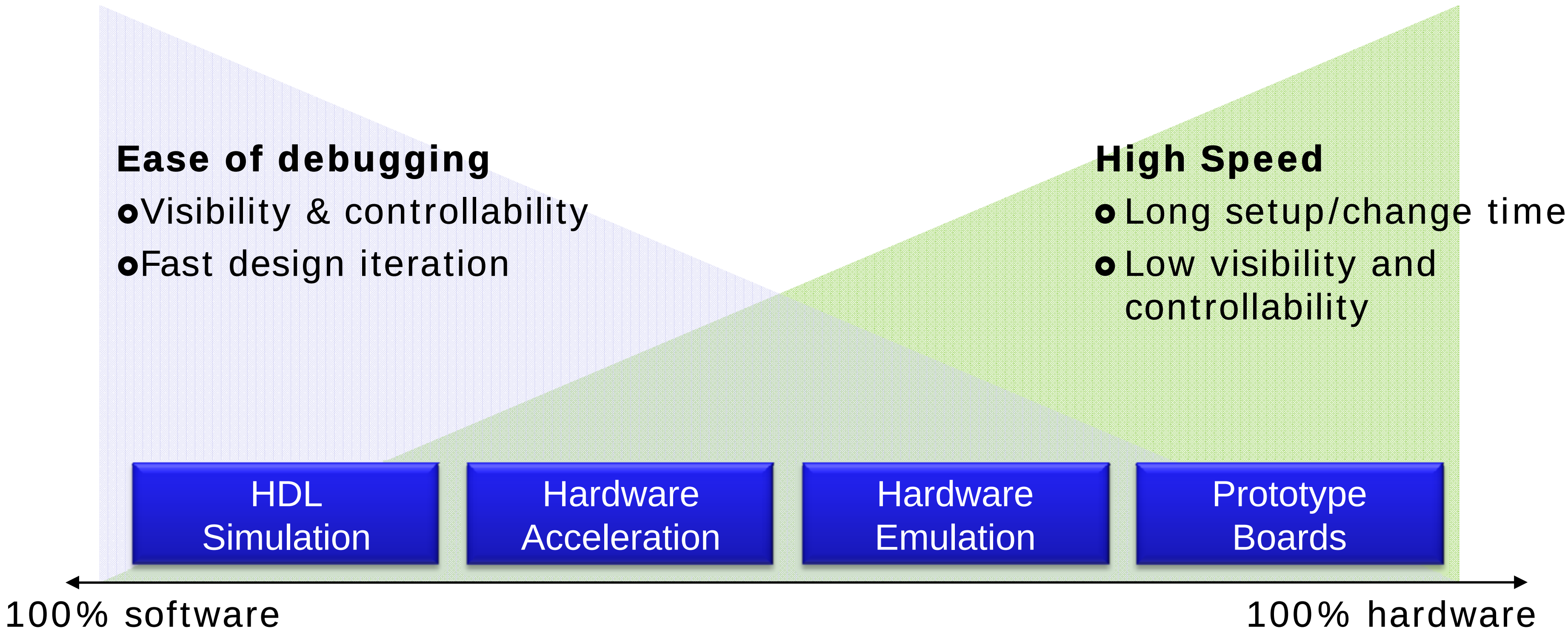
Typical Test Bench Approaches



FPGA Design (HDL)

- HDL-based
  - Directed tests
  - Randomized tests
- HVL-based (SystemVerilog, e, Vera, SystemC)
  - Constrained, randomized
  - Class-based/object oriented
- Embedded software-based

Verification Engine Tradeoffs



HDL Simulation

HDL Simulation | Hardware Acceleration | Hardware Emulation

- Typical application: block-level verification
  - Fast design change turnaround
  - Full visibility/controllability
- Testbench methodology:
  - From: simple directed + randomized tests
  - To: complex object oriented constrained-random (HVL)
- Challenges:
  - Chip-level verification may require compute farm
  - Long “unbreakable” test sequences can take days/weeks
  - Don’t see chip-level or device specific IP issues

Hardware Acceleration

HDL Simulation | **Hardware Acceleration** | Hardware Emulation

- Typical application: block-chip regression test
  - 1-2 orders of magnitude faster than simulation
- Testbench methodology:
  - Typically directed tests with simple randomization
- Challenges:
  - Capital cost
  - Setup time
  - Matching results with simulation
  - Specific hardware or software flow makes it impossible to verify designs with FPGA specific IP blocks

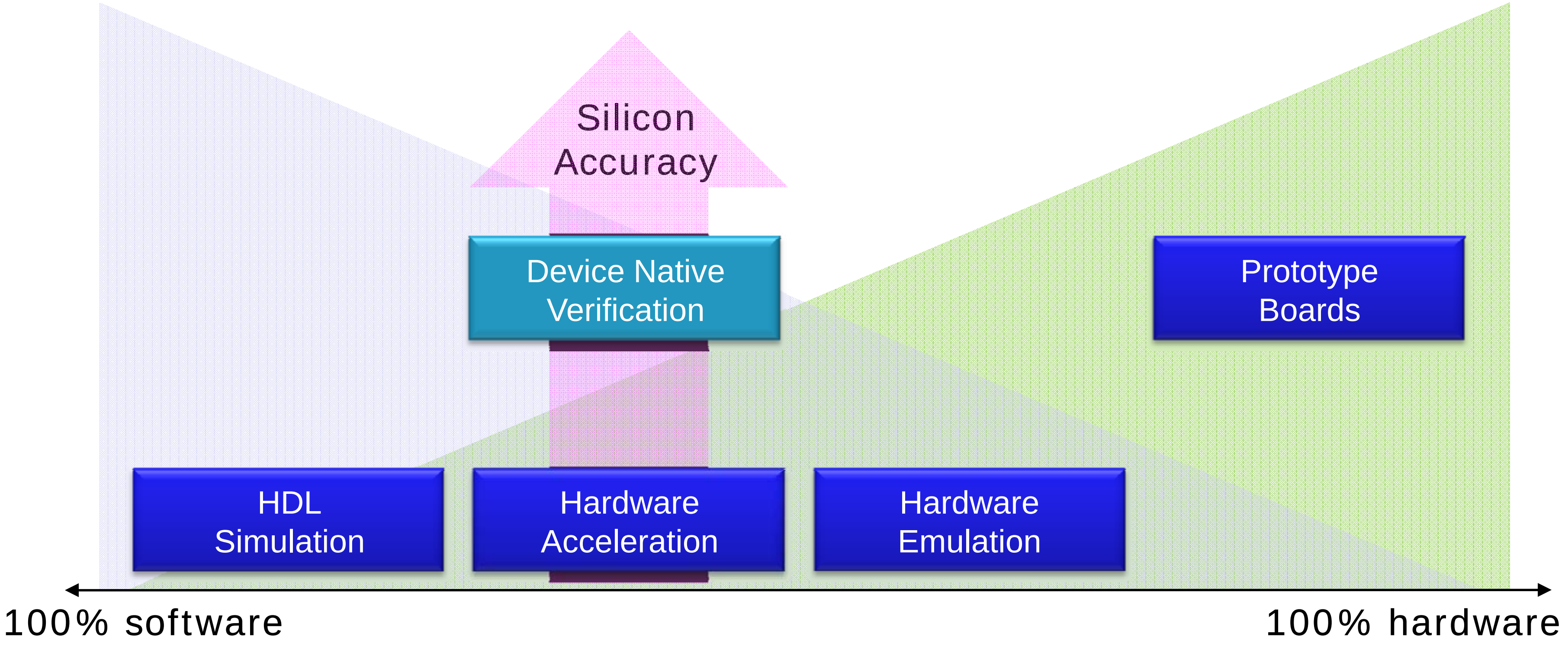
Hardware Emulation

HDL Simulation | Hardware Acceleration | **Hardware Emulation**

- Typical application: chip/system regression test
  - 3-4 orders of magnitude faster than simulation
  - Typically uses different mapping than target FPGA
- Testbench methodology:
  - Embedded code
- Challenges:
  - Capital cost
  - Setup time
  - Matching results with simulation and target FPGA
  - Debugging
  - Flow makes it impossible to verify designs with FPGA specific IP blocks

New: Silicon Accurate, Device Native Verification

Uses the native FPGA device as the accelerator



GateRocket™

Device Native Verification

HDL Simulation | **Device Native Verification** | Hardware Emulation

- Typical application: block-system regression test
  - Order of magnitude speed-up over simulation
  - Full debug visibility
  - Silicon-accurate results
- Testbench methodology:
  - Same as HDL simulation
- Challenges:
  - Inefficient testbench can degrade acceleration

Conclusion

- Thorough functional verification prior to system integration is a must for complex FPGAs
- Simulation coupled with hardware assisted verification offers the best of all worlds
  - Performance/coverage, debugging
- Device-native verification offers a promising, silicon accurate approach

GateRocket™

Learn more - Tutorial T3B - FPGA Verification

Tutorial T3B 2:40 to 5:00 PM Today

- Joe Rodriguez, Prod Mktg Mgr, Mentor Graphics
  - *Presentation: Recalculating the Road to Successful FPGA Verification*
- Chris Schalick, CTO, GateRocket
  - *Presentation: A Case for Hardware-Assisted Verification*
- James Smith, Dir EDA Vendor Relations, Altera
  - *Presentation: FPGAs Increasing Role in Development*
- Judith Smith, Prod Mgr, Agilent Technologies
  - *Presentation: FPGA-Based DDR Memory Controller Validation*
- Carsten Hoffman, Emulation Platforms and Dan Isaacs, Dir Adv Prod Mktg, Xilinx
  - *Presentation: Platform FPGA Automated Regression Test Methodology*

GateRocket™

System Design to FPGA Implementation

FPGA Summit
December 2008

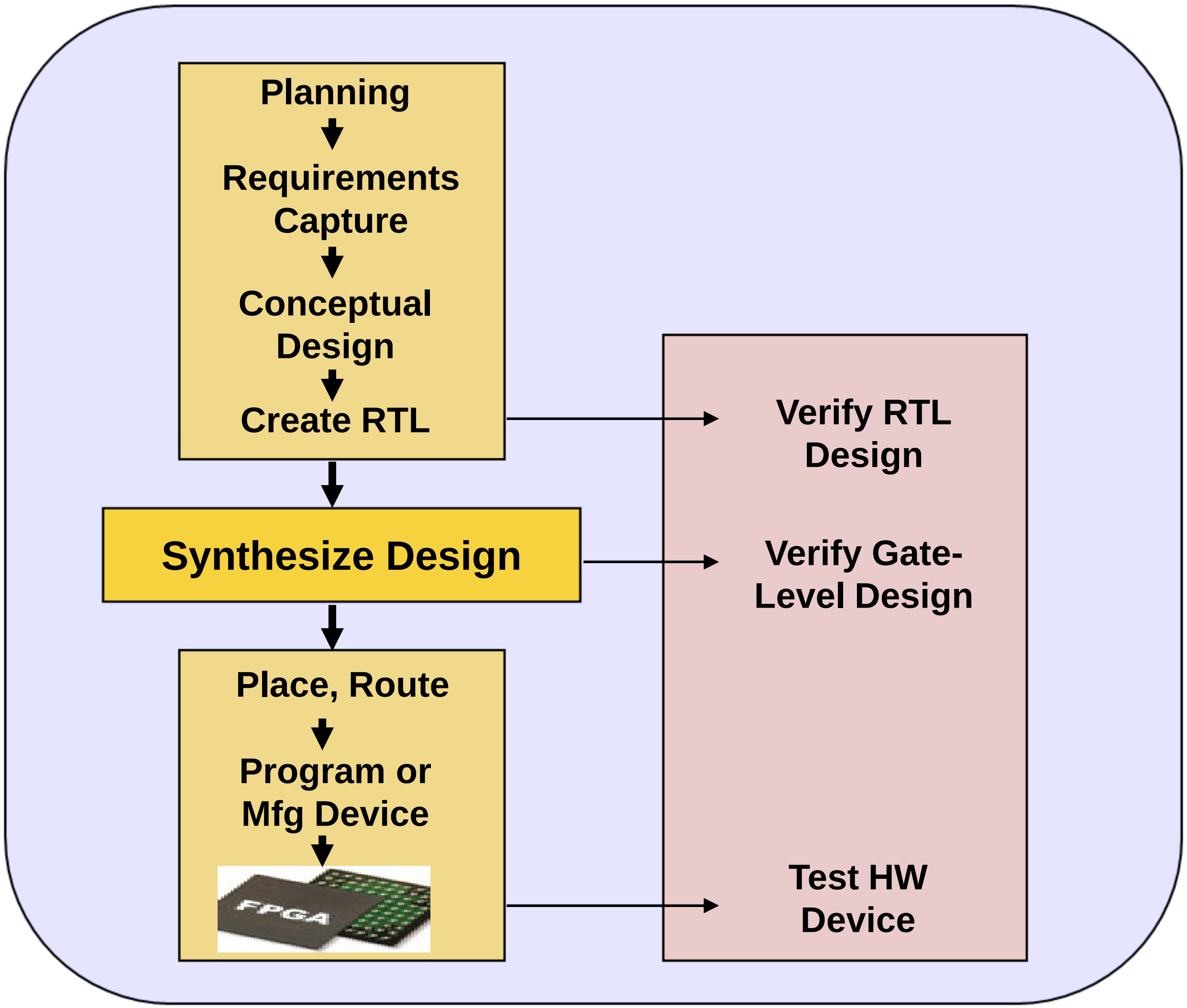
Agenda

- Scope of presentation
  - Different methodologies possible
  - Common theme - Tackling FPGA design complexity
- FPGA design challenges
- Possible design flow solution
  - System Design
  - RTL Design
  - Physical Implementation
  - Verification

FPGA Design Challenges

- Challenges
  - Business driven
    - Reduce cost
    - Meet performance goals
    - Reduce time to market
  - Technology driven
    - Device size and complexity
    - Design size and complexity
    - Certifiable design process
- Solution - ASIC style design methodology
  - Raise level of abstraction
  - Reuse designs
  - Maintain vendor independence
  - Use advanced verification tools

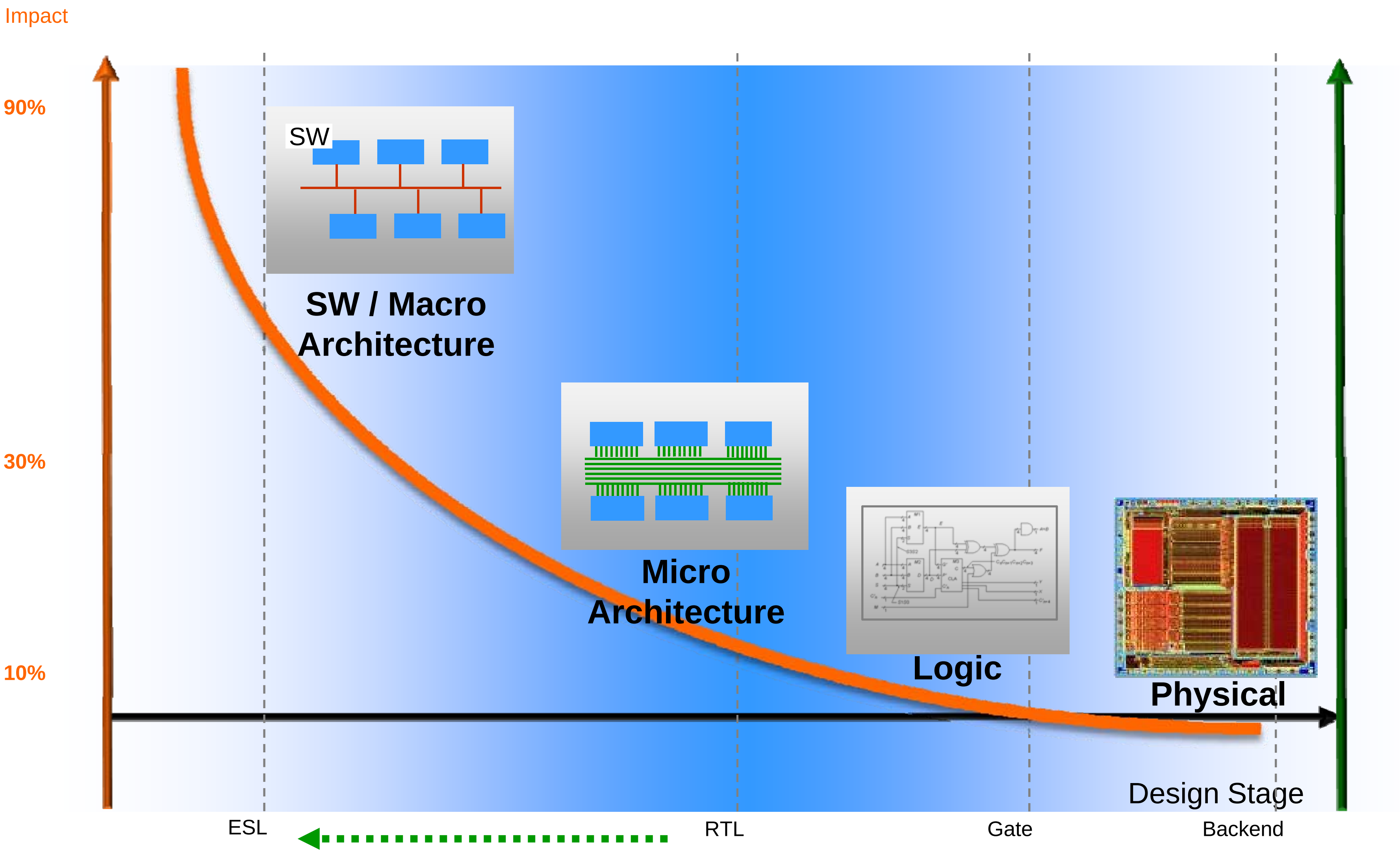
FPGA Design Flow



Trace Requirements Throughout The Flow

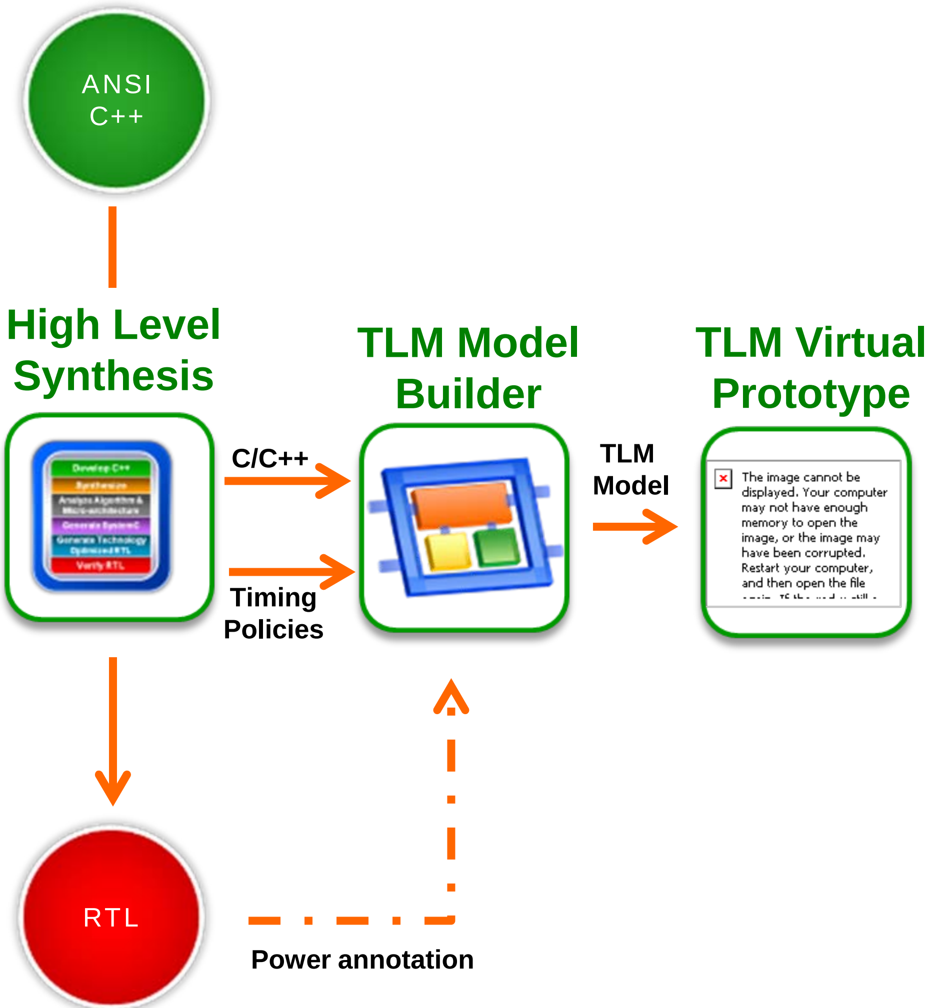
Mentor Graphics

Impact Of Change



Raise Level Of Abstraction

Scalable Transaction Level Modeling



- **Single C/C++ source for synthesis, verification and analysis**
- **Automated timing annotation from HLS scheduler**
- **Fast System simulation at TLM speed [x1,000 and up)**

Raise Level Of Abstraction

Transaction Level Modeling Solutions

Modeling

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

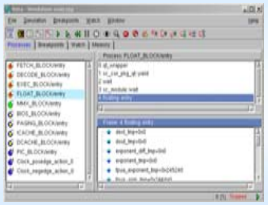
Automate Modeling

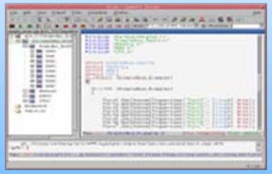
Assembly

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

ESL/RTL Integration

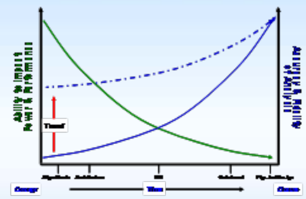
Validation

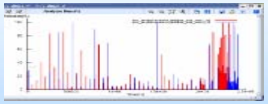




HW/SW Validation

Analysis





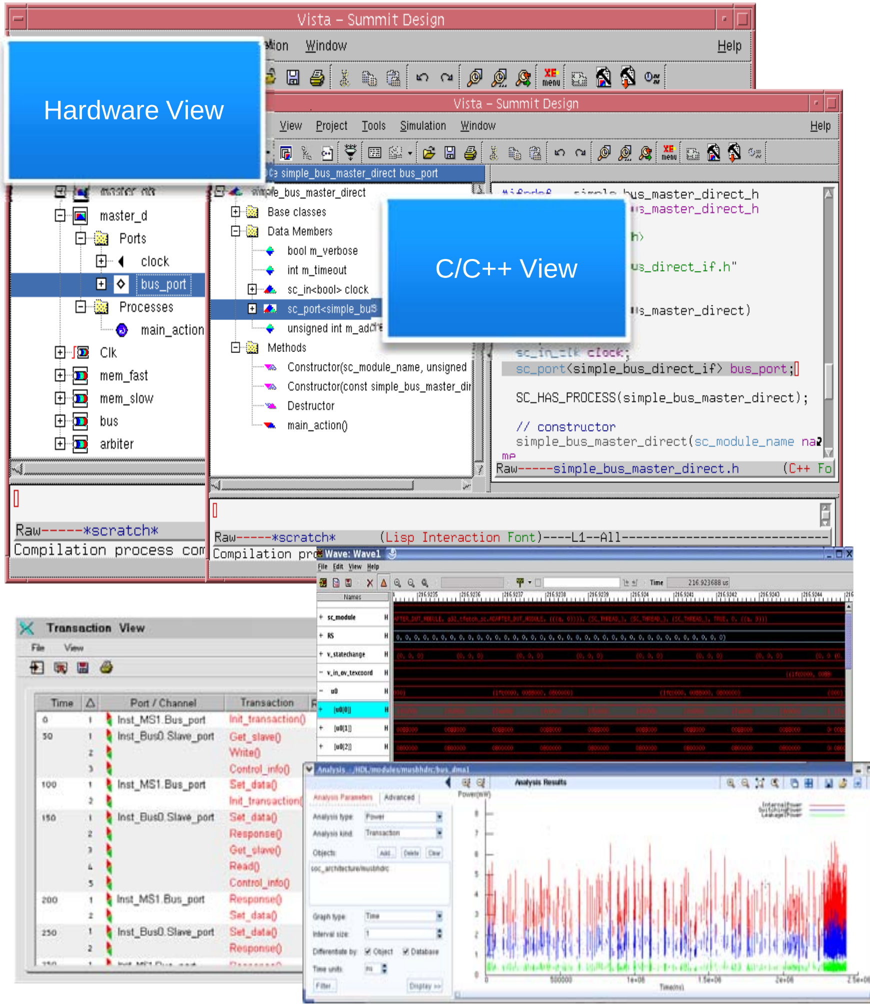
The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

Optimize System Architecture

Raise Level Of Abstraction

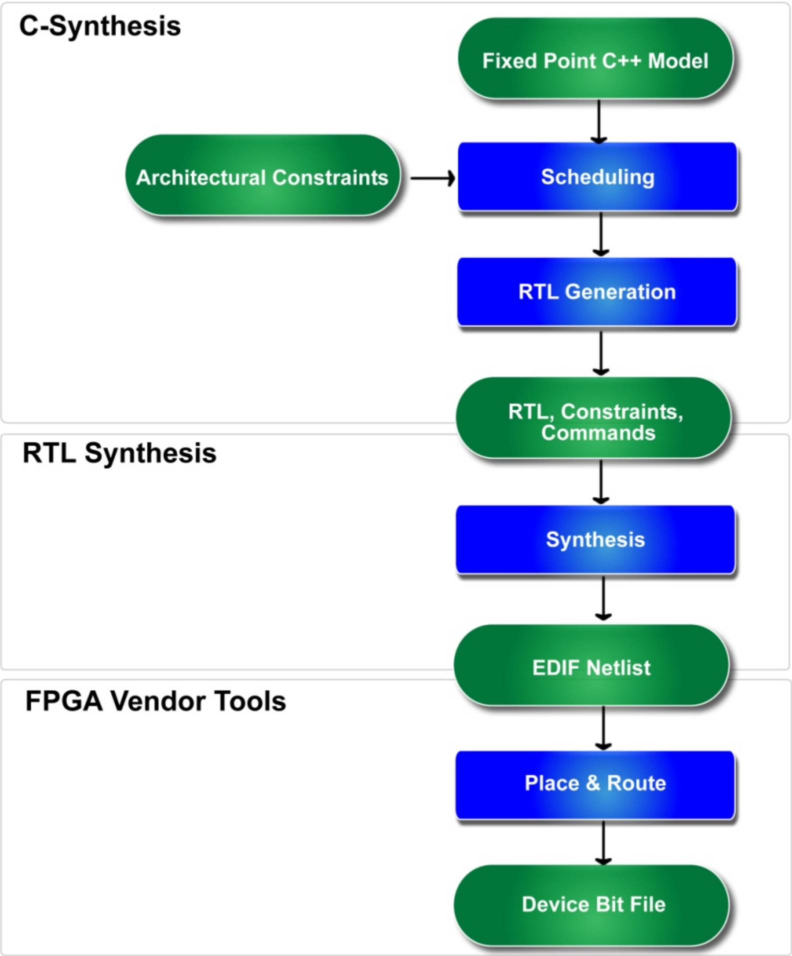
HW/SW Debug & Analysis Platform

- **Key Features**
  - **Powerful HW & C/C++ views**
  - **TLM Tracing**
  - **Powerful Debug – Process Control**
  - **Timed / Untimed simulation**
  - **Timing / Power Analysis**
- **Key Benefits**
  - **Reducing validation cycles**
  - **Bridging SW & HW Design**
  - **System-level Exploration**
  - **Standard Based – OSCI/gdb/gcc**



Raise Level Of Abstraction
ESL Synthesis

- **Use flows that are certified**
- **ESL Synthesis**
  - Synthesizes standard ANSI C++
  - Technology aware scheduling
  - High quality RTL code generation
- **ESL output optimized for FPGA Synthesis**
- **FPGA Synthesis integrated with ESL tool**



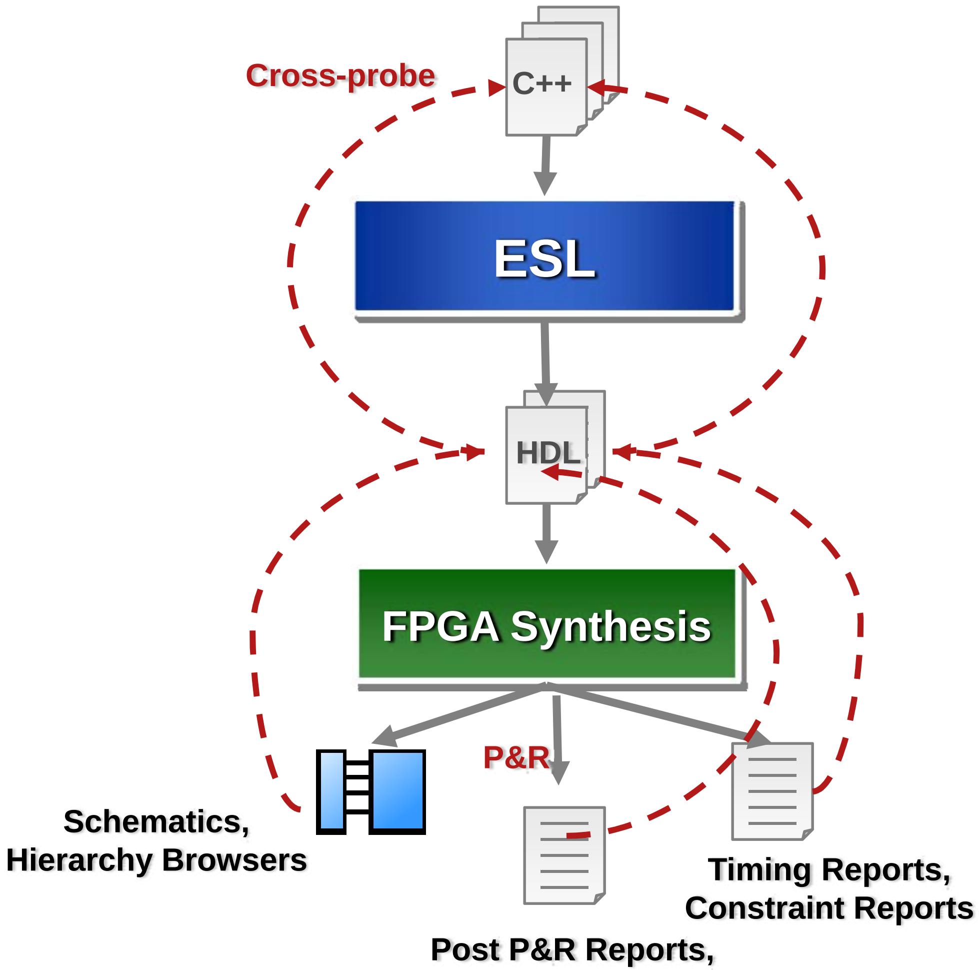
Raise Level Of Abstraction

Better QoR with ESL-FPGA synthesis flow

- **Technology aware C++ synthesis**
  - Allocation, Scheduling based on timing and area of library elements (Library elements: logic gates, arithmetic operators, memories)
  - Timing and area information extracted from FPGA synthesis tool (Library characterization process)
- **ESL tool outputs FPGA synthesis tool ready RTL**
- **FPGA synthesis tool includes ESL tool libraries**
  - Latest FPGA timing data available to ESL tool

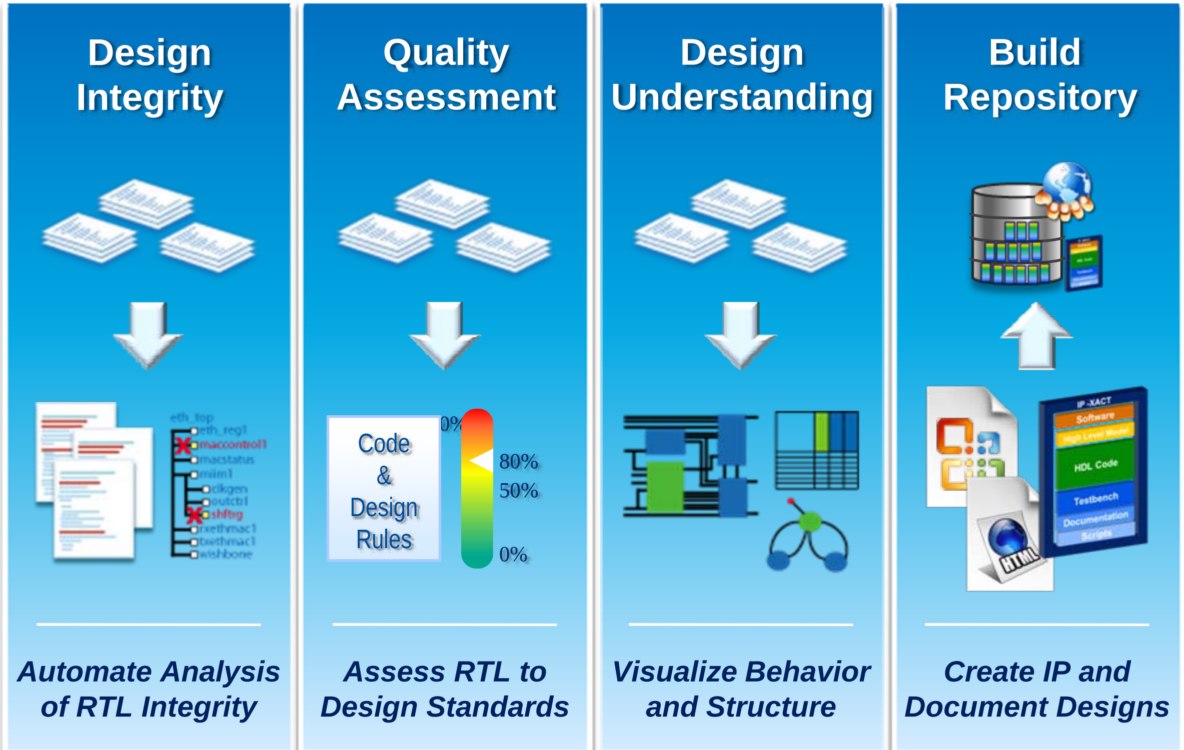
Close integration - ESL and FPGA Synthesis

- **FPGA Synthesis invoked from ESL with project settings being carried over**
- **Extensive one-step cross probing from FPGA Synthesis and Place & Route result files to ESL tool**
- **FPGA Synthesis is tested in ESL tool regression framework**



HDL Design Environment

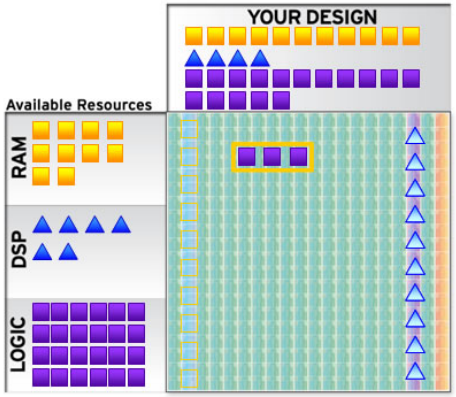
- **Save time by analyzing & correcting design integrity**
- **Ensure the best reuse decision by measuring code quality**
- **Accelerate design understanding of structure & behavior**
- **Extract, package and manage IP**



Maintain Vendor Independence

FPGA Synthesis

- **Efficient Design Creation**
  - SystemVerilog & standards support, ESL integration
- **Physical Synthesis**
  - Broad device support
  - Superior results for all vendors
- **Incremental Synthesis**
  - Support for incremental design flow
- **Graphical Resource Analysis & Control**
  - Allocate resources to optimize for performance or area



Improve QOR

Reduce cost

Speed up time to market

Maintain Vendor Independence

FPGA Synthesis

- **Analysis & Debug**
  - Missing Constraint report
  - Clock Domain Crossing report
  - Comprehensive messaging/warnings
  - Cross-probing between HDL & Schematics
  - Critical-path viewing for timing analysis
  - What if analysis without re-synthesis
  - Graphical resource analysis and control

Improve QOR

Reduce cost

Speed up time to market

- **Technical Support**
  - Support complete flow with proper ownership

Maintain Vendor Independence
FPGA Synthesis

- **Safe Operation**
  - Safe FSM encoding to avoid single-event upset
  - Support for Redundancy
  - Safer synthesis with selective optimization
- **Verifiable & Reproducible**
  - Good integration & support with verification tools
  - Deterministic netlist generation
- **Synthesis Artifacts**
  - Extensive reporting for documentation requirements

Improve Design Safety

Use Advanced verification tools

Advanced Functional Verification

Inputs

- ✓ Design
- ✓ Testbench
- ✓ Assertions
- ✓ Verification Plan
- ✓ Power files

Outputs

- ✓ Waveforms
- ✓ Functional Coverage
- ✓ Code Coverage
- ✓ Toggle Coverage
- ✓ Assertion Coverage

Languages

- ✓ VHDL
- ✓ Verilog
- ✓ PSL
- ✓ SystemC
- ✓ SystemVerilog
- ✓ C/C++

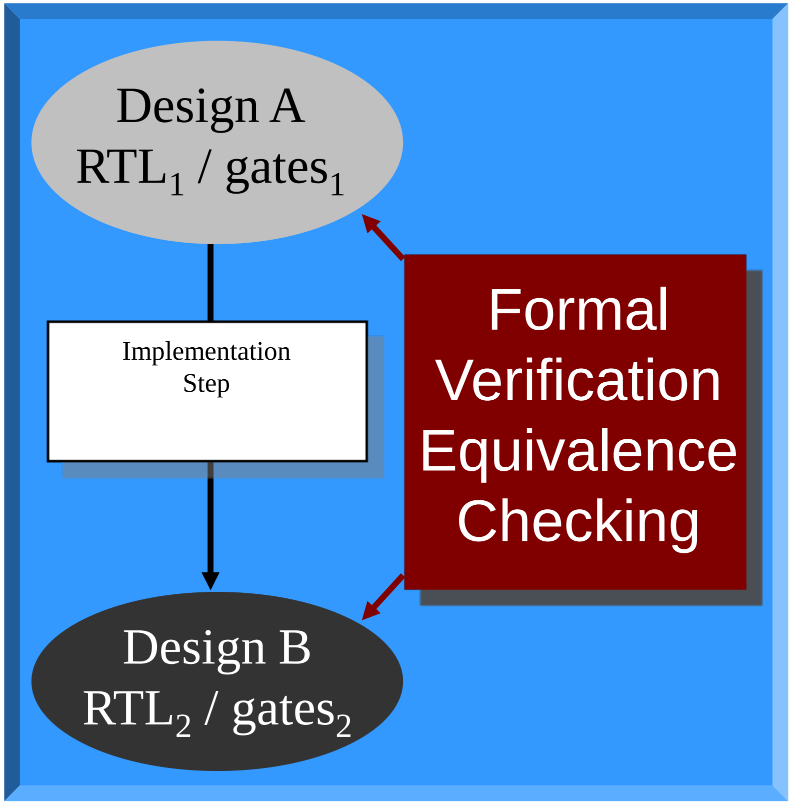
Analysis

- ✓ Hardware debugger
- ✓ UCDB
- ✓ Assertion debugger
- ✓ C debugger
- ✓ Verification management

Use Advanced verification tools

Formal Verification

- **100s times faster then gate level simulation**
- **100% coverage provided**
- **Resolves netlist uncertainty**
- **A complete gate-level test bench is time consuming and large**
- **Place & Route takes a day**
- **Running FPGA live does <u>not</u> test the corners**



Use Advanced verification tools

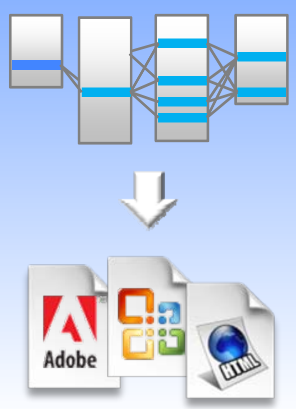
Manage Requirements in Design Flow

- Control & predict project schedules
- Trace requirements through HW design process
- Clearly communicate via visualization & intuitive reports
- Manage impact of requirement changes
- Meet safety critical & DO-254 certification

Requirements Capture & Tracing

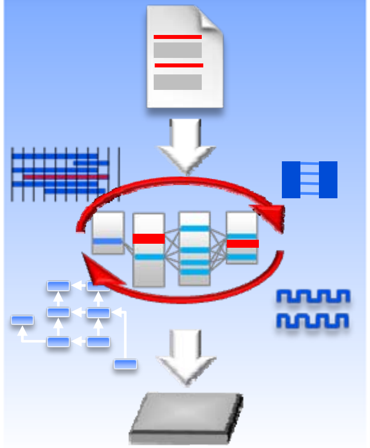
Trace Through Authoring, Coding & Testing

Report & Document



Link Automated Reports, Specs, Design & Results

Change Management



Impact Analysis to Control Schedules

Physical Implementation

- **Input data**
  - Synthesized design netlist
  - Design constraints
- **Improving performance**
  - Different implementation options
  - Design exploration tools
  - Interactive physical design
  - Different synthesis options
  - Modify RTL
- **Debug**
  - Insert debug logic post-synthesis
- **Program the device**
  - Create bit stream

Configure design on selected FPGA device

Summary

- FPGA design challenges are growing
- Design process needs to be certifiable

- Solution is to use ASIC style design methodology
- To enhance effectiveness
  - Raise level of abstraction
  - Reuse designs
  - Maintain vendor independence
  - Use advanced verification tools

Breakout Sessions

- Break into three groups
- Take next 20-30 minutes to discuss topics
  - Design Methods — Daniel Platzker
  - Systems Interface — Andy Stevens
  - Verification — Dave Orecchio
- Report back as part of the panel discussion

Design Tools & Portability: Breakout Questions

- How important is vendor independence (VI) to designers vs. management?
- What are the advantages/disadvantages of using vendor vs. EDA tools?
- When do you write VI code (inferable) vs. instantiating tech cells?
- Do you look at different vendors for each project? Why not?
  - No time?
  - Strong existing vendor relationship?
  - Silicon requirements met by single vendor?
  - IP?
- Using Vendor IP? Why?
  - Because of cost?
  - Happy with the vendor support?
  - Which IP using the most? Processors? SERDES?

Panel Session

- Reports back from breakout sessions
  - Design Methods — Daniel Platzker
  - Systems Interface — Andy Stevens
  - Verification — Dave Orecchio

Thank you for attending!

- Please fill out the session evaluations and return them up front
- Key note address
- Lunch time!