SAND2014-1250P

Exceptional service in the national interest

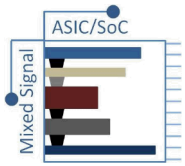
Sandia National Laboratories

Jason Michnovicz (jmichno@sandia.gov)
Microelectronics R&D S&E
Sandia National Laboratories

With contributions from T.J. Mannos, Vivian Kammler, Jae Joon Chang, Ratish Punnoose

Success Story

Using Advanced Verification Techniques to Increase Effectiveness



Mentor Graphics Corporation Sales Kickoff 2014

U.S. DEPARTMENT OF ENERGY | NNSA

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Outline

- **Sandia National Laboratories**
- **Requirements Tracing** – Our Strategy
  - ReqTracer
- **Verification Techniques** – An Overview
- **Our Experiences**
  - SystemVerilog Assertions
  - Formal Verification
  - Universal Verification Methodology (UVM)
  - Unified Coverage Database (UCDB)

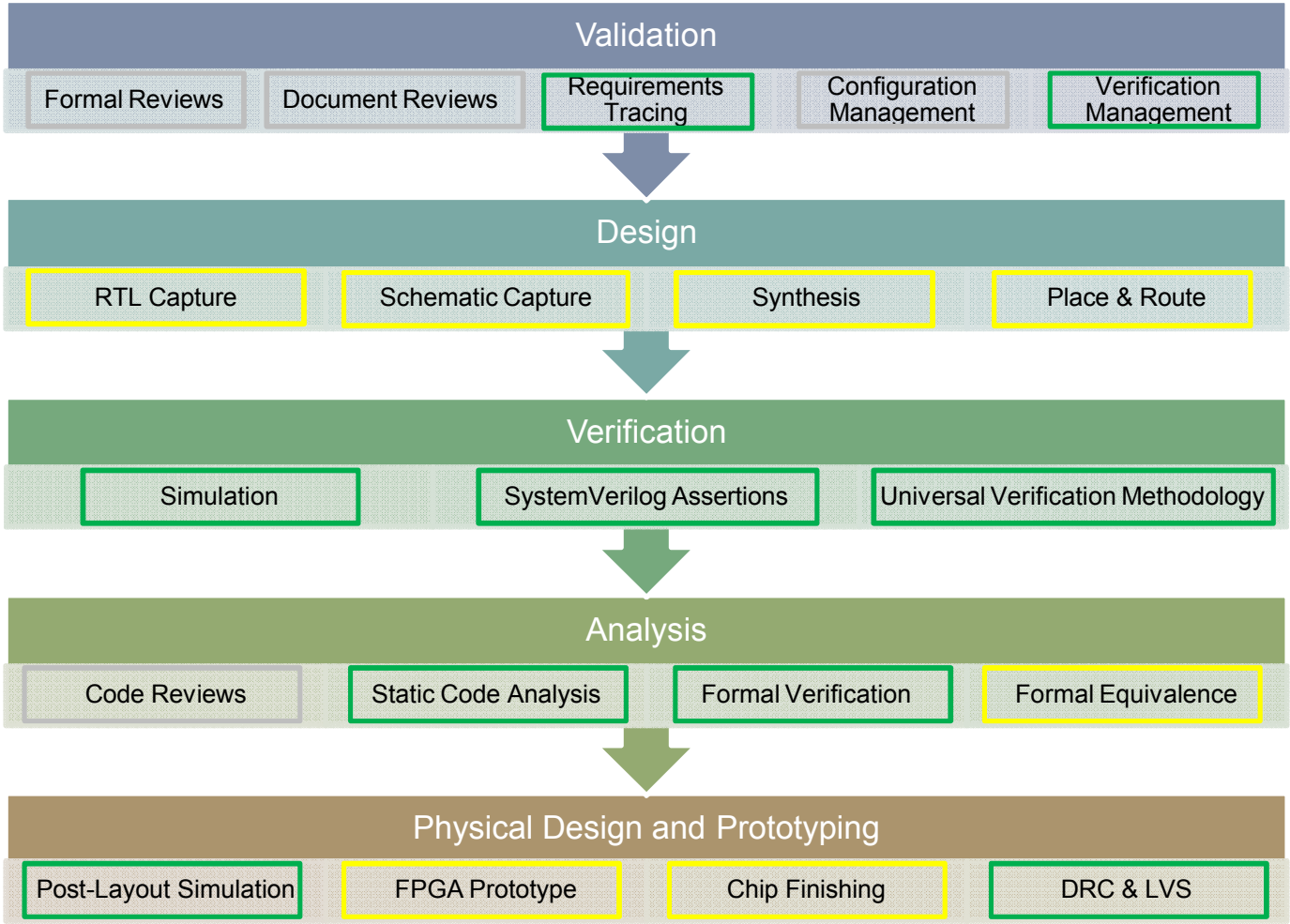
Some background information about

SANDIA NATIONAL LABORATORIES

Sandia National Laboratories

- U.S. Department of Energy R&D Labs
- Wide variety of research activities
  - Nuclear weapons security
  - Energy
  - Climate
  - Cybersecurity
  - Pulsed power
  - Robotics
  - Microsystems
  - Radiation effects

High Rigor ASIC Flow Example



Mentor Product

Competitor Product

No Offering

Mentor Verification Products

Sandia National Laboratories

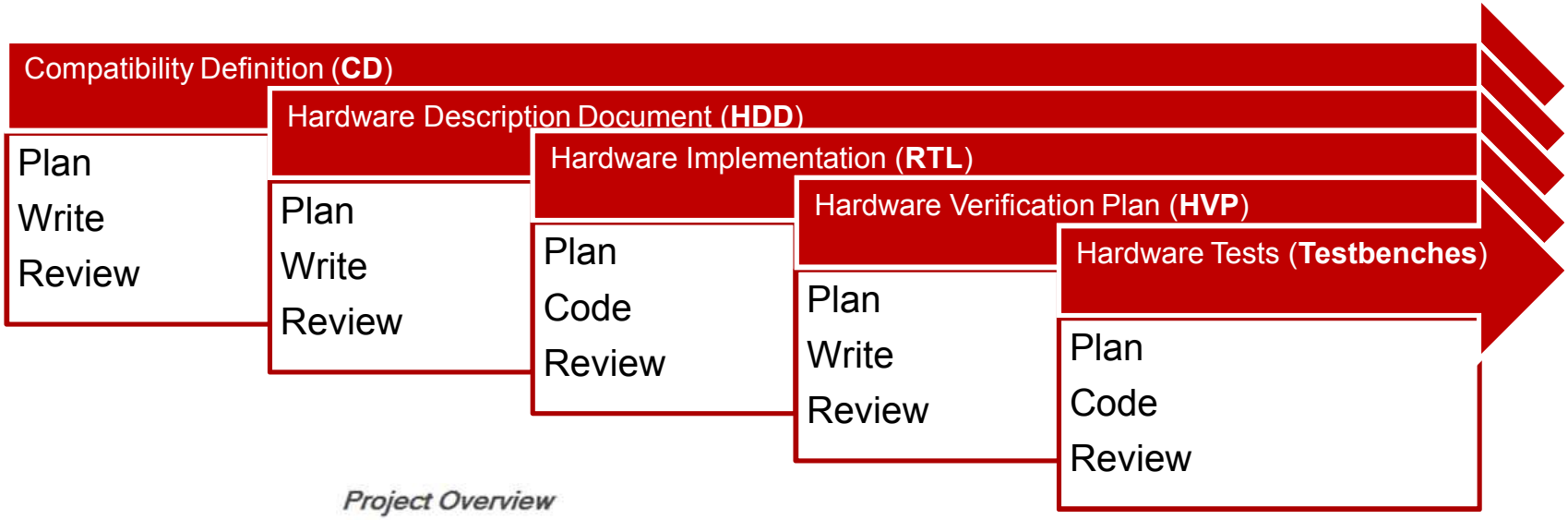
Product	Mentor	Competitor	Do Not Use
Certe Testbench Studio			X
FormalPro		X	
HDL Designer (DesignChecker)	X		
HDL Designer (other features)			X
Questa ADMS		X	
Questa Sim	X		
Questa Formal	X	X	
Questa CDC			X
Questa CodeLink			X
Questa CoverCheck			X
Questa Verification Mgt.	X		
Questa inFact			X
ReqTracer	X		
Veloce Emulator			X

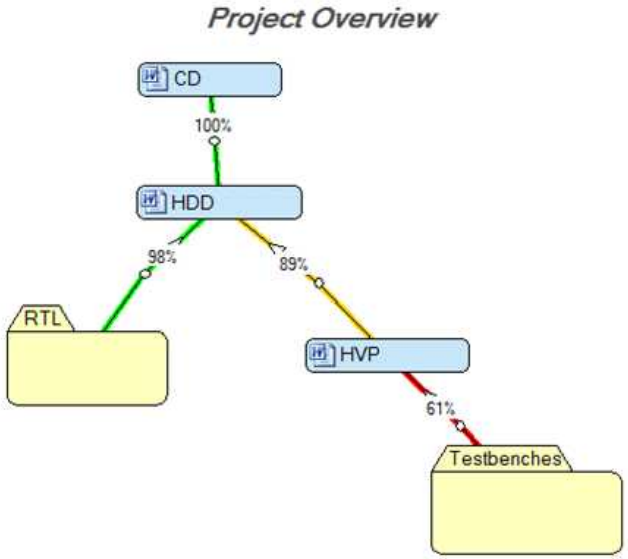
Sandia National Laboratories

How we implemented

REQUIREMENTS TRACING (USING REQTRACER)

Design Lifecycle

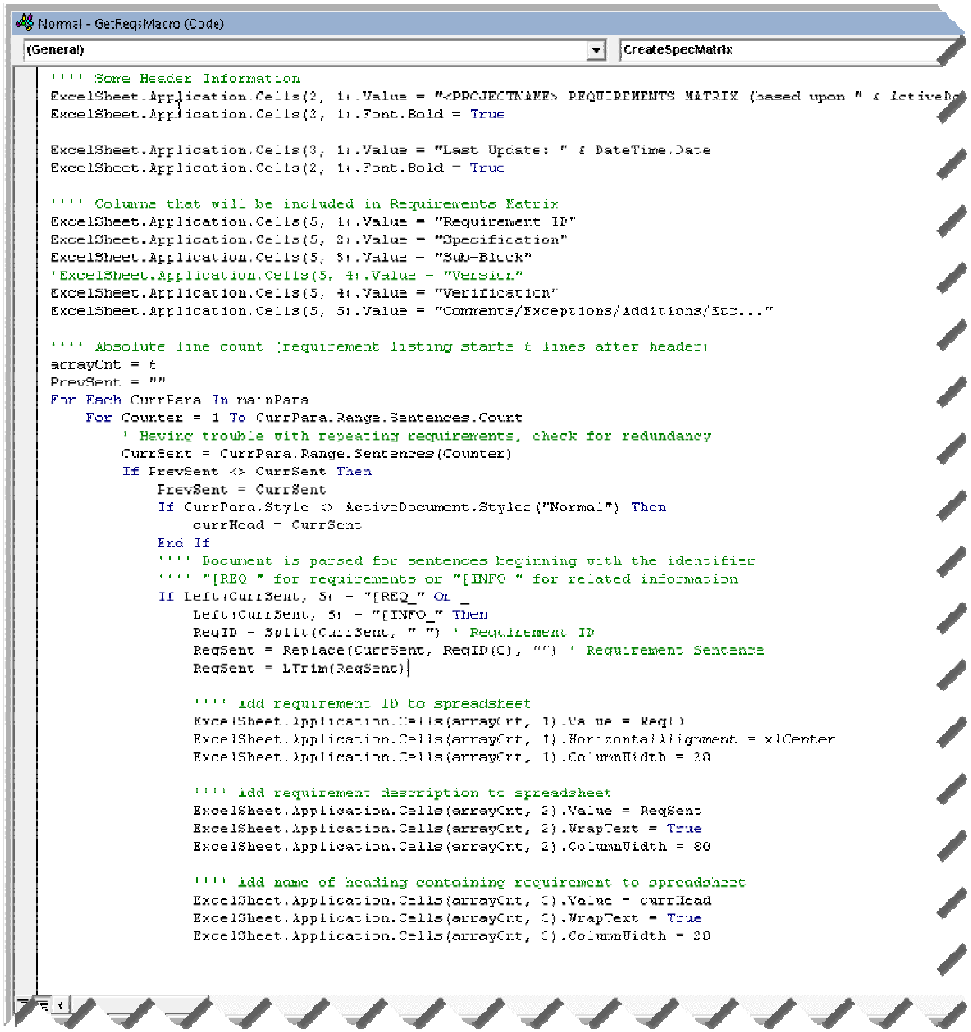




Previous Strategies:

The *Shall* Statement

- Visual Basic parser to look for “shall” statements in Microsoft Word document
- Hard to maintain or modify for other projects
- Format highly constrained (MS Word input, MS Excel output)
- Any ID linking had to manually entered and modified as needed; run the macro once – no dynamic changes



Previous Strategies:

The spreadsheet ("Excel hell")

- Assigned ID's to individual requirements
- Separate matrix for tracking simulations, test validation
- Manually create reports, hyperlinks

	A	B	C	D	E	F	G
1							
2	REQ-ID	METHOD	REQUIREMENT	OWNER	STATUS	REFERENCE	NOTES
3	OLY-2000	Inspect	The Olympics shall be held in Sydney, New South Wales, Australia.	IOC	Complete	Wikipedia Analysis	
4	OLY-2004	Inspect	The Olympics shall be held in Athens, Greece.	IOC	Complete	Wikipedia Analysis	
5	OLY-2008	Inspect	The Olympics shall be held in Beijing, China.	IOC	Complete	Wikipedia Analysis	
6	OLY-2012	Inspect	The Olympics shall be held in London, United Kingdom.	IOC	Complete	Wikipedia Analysis	
7	OLY-2016	Build	Olympic facilities shall be built in Rio de Janeiro, Brazil.	Rousseff	In Progress	Wikipedia Analysis	
8	OLY-2020	Announce	The IOC shall announce Tokyo, Japan as the hosting city for the 2020 Olympics.	Bach	Complete	125th IOC Session	This announcement was made on 7 Sept. 2013
9							

requirements_matrix | test_validation_matrix | functional_case_matrix

Ready

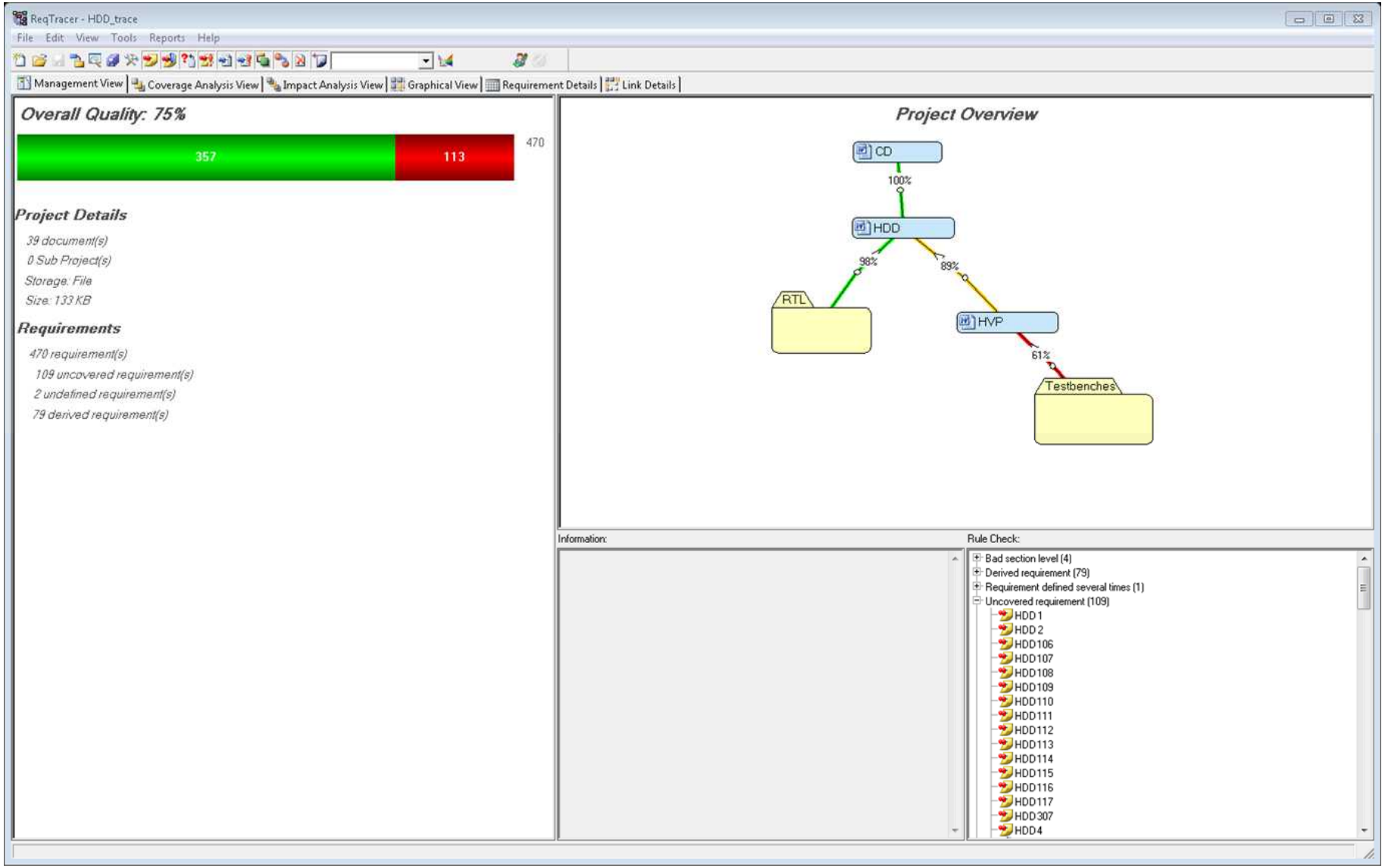
Choosing ReqTracer

- Needed requirements mapping solution compatible with customer databases; hardware design tool integration a plus
- Newer tool (2010), but based on Geensoft Reqtify
  - Reqtify now owned by Dassault Systèmes
  - Core technology provided to Mentor Graphics and LDRA
  - Mentor Graphics ReqTracer and LDRA TBReq developed from there

Mentor Graphics Corporation ® is a world leader in electronic hardware design solutions, providing products, consulting services and award-winning support for the world's most successful electronics and semiconductor companies. Mentor has integrated GEENSOFT requirements tracing technology, Reqtify, into its world-class HDL design and verification tools to provide requirements traceability directly into design and verification data. This highly integrated solution, centered on the new Mentor ReqTracer™ product, meets the stringent requirement traceability objectives of the DO-254 standard for development and design assurance of complex electronic hardware. **For more information on ReqTracer**

LDRA is a specialist technology company. For thirty years LDRA has provided automated analysis and testing tools for software applications on which peoples' lives depend. Blue chip companies in the aerospace, defense, nuclear and automotive sectors use LDRA tools to test their applications to safety-critical standards. LDRA and GEENSOFT partner to provide an effective solution linking requirements, test scenarios and test results, known as **"TBreq"**

ReqTracer Management View



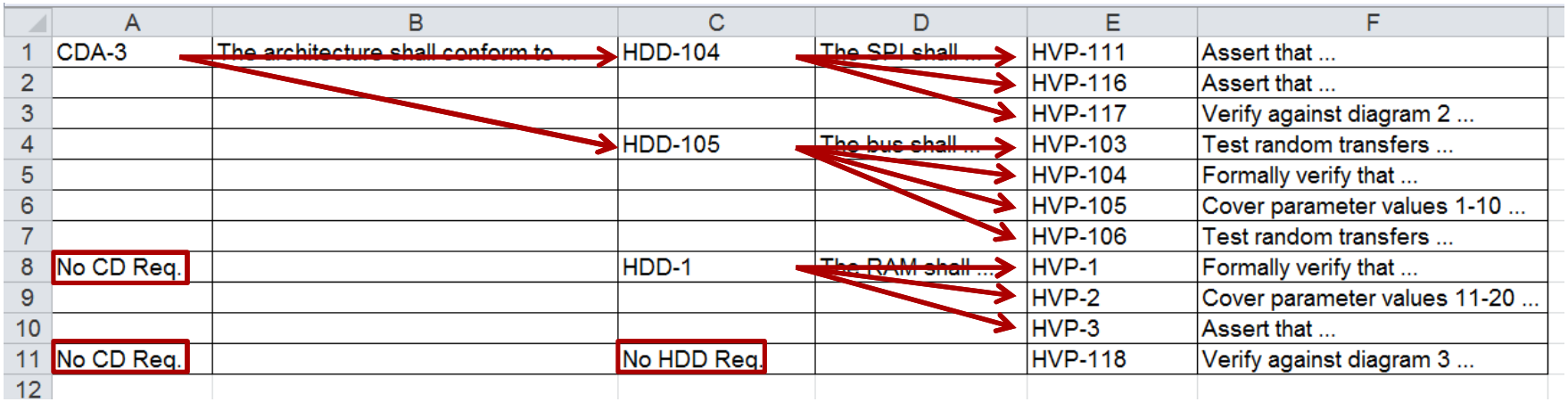
Benefits from ReqTracer

Sandia National Laboratories

- Common Needs Addressed
  - Wide variety of parsing options
  - Wide variety of analyzable file formats
- Fewer Files to Maintain
  - Eliminates manual syncing of documents and spreadsheets
  - Data accessible through generated reports
- Easy-to-Access Traceability Metrics

Results from ReqTracer

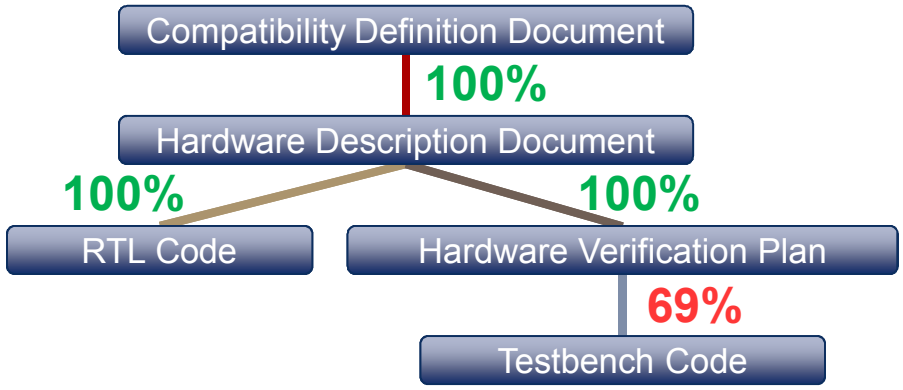
- **Problem: Determine Whether...**
  - More requirements are *necessary*
  - The existing tests are *sufficient*



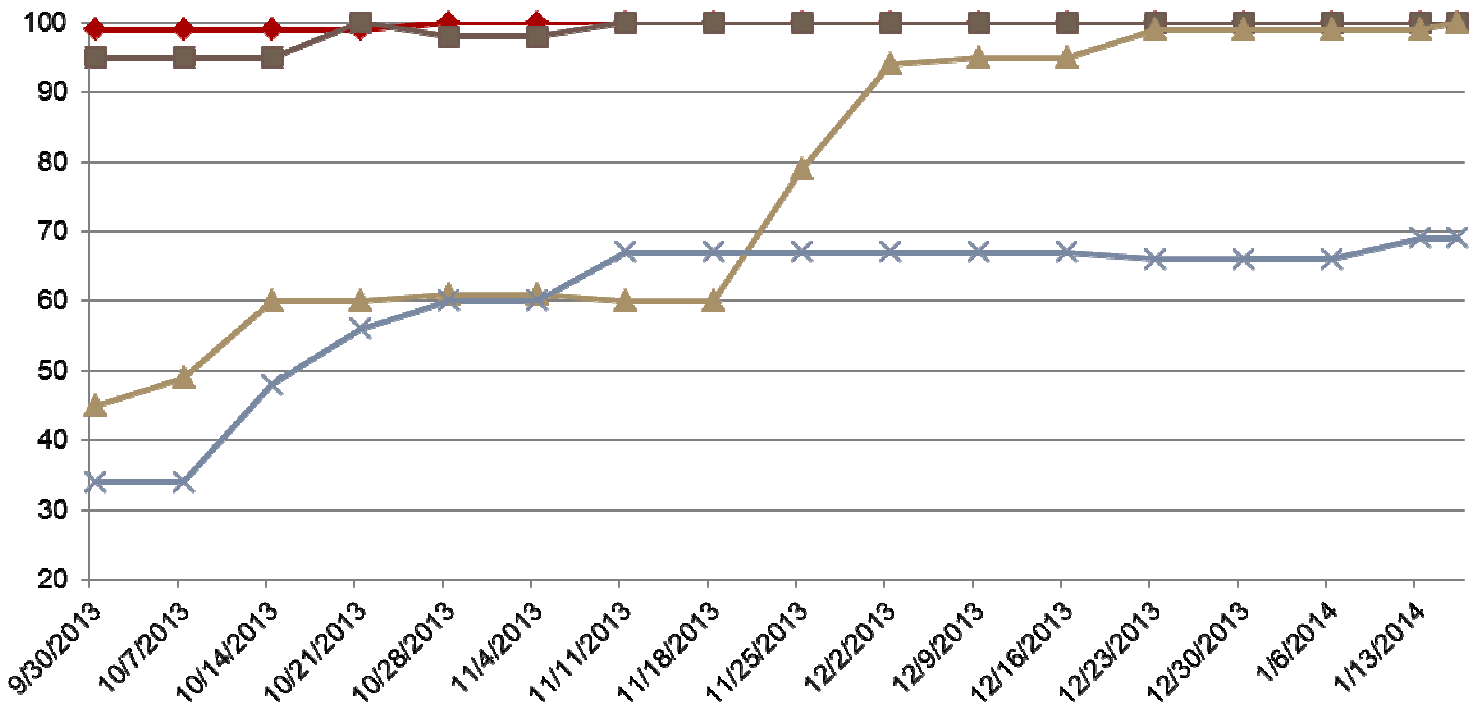
	A	B	C	D	E	F
1	CDA-3	The architecture shall conform to ...	HDD-104	The SPI shall ...	HVP-111	Assert that ...
2					HVP-116	Assert that ...
3					HVP-117	Verify against diagram 2 ...
4			HDD-105	The bus shall ...	HVP-103	Test random transfers ...
5					HVP-104	Formally verify that ...
6					HVP-105	Cover parameter values 1-10 ...
7					HVP-106	Test random transfers ...
8	No CD Req.		HDD-1	The RAM shall ...	HVP-1	Formally verify that ...
9					HVP-2	Cover parameter values 11-20 ...
10					HVP-3	Assert that ...
11	No CD Req.		No HDD Req.		HVP-118	Verify against diagram 3 ...
12						

- **Solution: Report Generation**

Requirement Trend Analysis



Coverage Link	Current	Goal
HDD to CD	100%	100%
HVP to HDD	100%	100%
RTL to HDD	100%	100%
TB to HVP	69%	100%



Pitfalls in ReqTracer

Sandia National Laboratories

- False sense of security
  - Partial coverage reported as full coverage by default
  - Back to square one (Excel hell) if any numbers change
- Report generation GUI is cumbersome
  - Only saves time on large tasks
- Limited documentation
  - Mystery fields in the GUI
  - Lack of introductory guide for scripting (OTScript)

Making the Most of ReqTracer

ReqTracer works best when:

- ReqTracer compatibility is in mind from the start
  - Parser-friendly writing of specifications
  - Parser-friendly writing of code
  - Highly specific requirement defining
  - Static requirement numbering
- There is a large number of requirements
  - Time spent on report-writing interface pays off

Sandia National Laboratories

An overview of

VERIFICATION METHODOLOGIES

Verification Types

Sandia National Laboratories

- Classic: Directed Simulation, Code Coverage
- Advanced: SVA, UVM, Formal, UCDB
  - SystemVerilog Assertions (SVA)
  - Universal Verification Methodology (UVM)
  - Formal Verification
  - The Unified Coverage Database (UCDB)

Advanced Verification - Why

Sandia National Laboratories

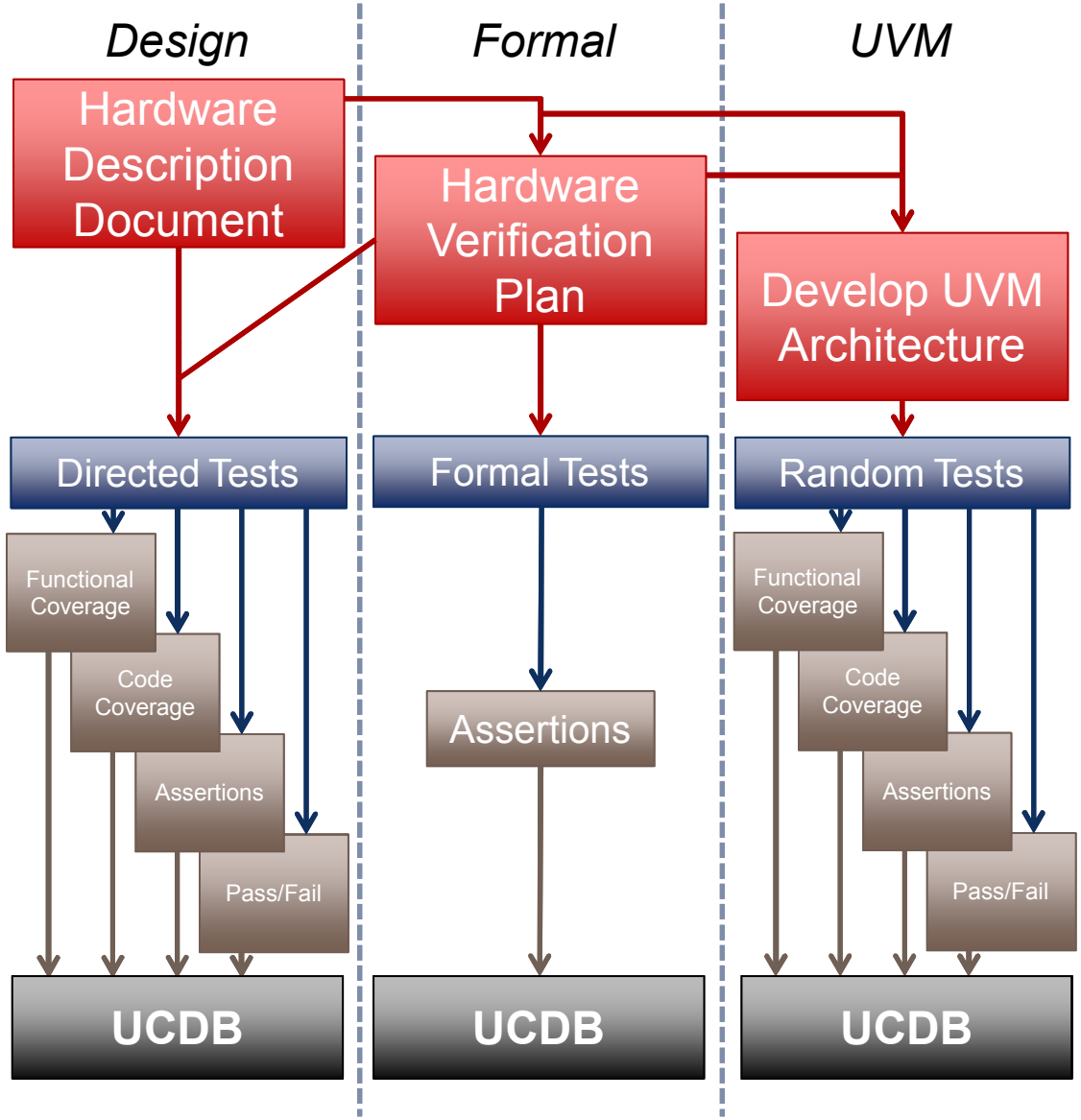
- Thoroughness
  - Tests corner cases (UVM)
  - Describes bug conditions (Formal)
- Reusability
  - Constructs can be used across methodologies (SVA/UVM/Formal)
  - Environment can be used across designs (UVM)
  - Object-oriented techniques can be used in full force (UVM)
- Insight
  - Encourages new ways of thinking about the design
  - Exposes the logical organization of the design

Reusability

Product	Simulation	SVA	UVM	Formal
ModelSim	X			
Questa Core	X	X		
Questa Prime	X	X	X	
Questa Formal		X		X

Assertions developed in SVA can be reused in UVM and Formal as verification needs grow.

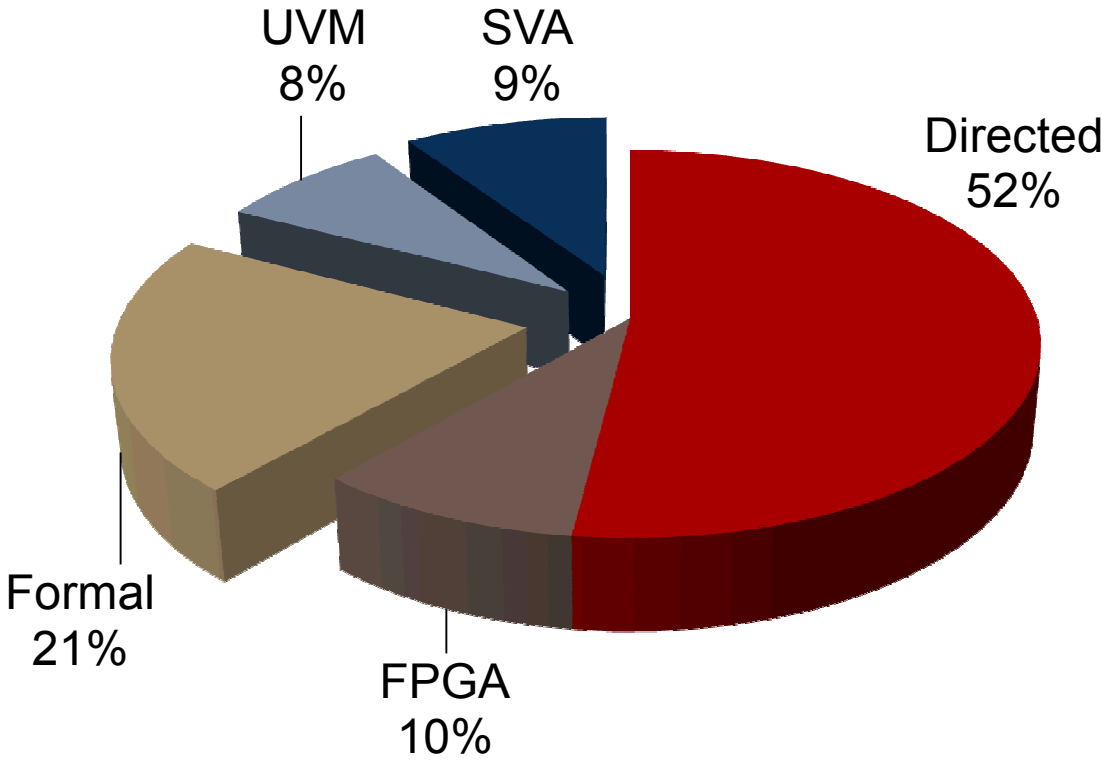
Division of Labor



- Design and verification responsibilities owned by independent teams
- Deficiency reports are filed as bugs are found
  - Tracked on TeamForge
- Directed, Constrained Random, and Formal tests cover HVP
- Unified Coverage Database (UCDB) generated using either Questa Core or Questa Prime

Advanced Verification – Success!

During early adoption of these techniques, advanced types of verification found 39% of recorded deficiencies in a design using them in parallel with classic techniques.



75 Verification-Driven Deficiency Reports (Categorized by First-to-Find Verification Methodology)

Sandia National Laboratories

Our experiences with

SYSTEMVERILOG ASSERTIONS

Basics of SystemVerilog Assertions

Sandia National Laboratories

- Define *Sequences* of Events
- Express *Properties* in Terms of Sequences
- *Assert* Properties
  - Check validity at specific points in time (*procedurally*)
  - Check validity at all points in time (*concurrently*)
- *Cover* Sequences and Properties
  - Count occurrence at specific points in time (*procedurally*)
  - Count occurrence at any point in time (*concurrently*)

Benefits of SVA-Based Verification

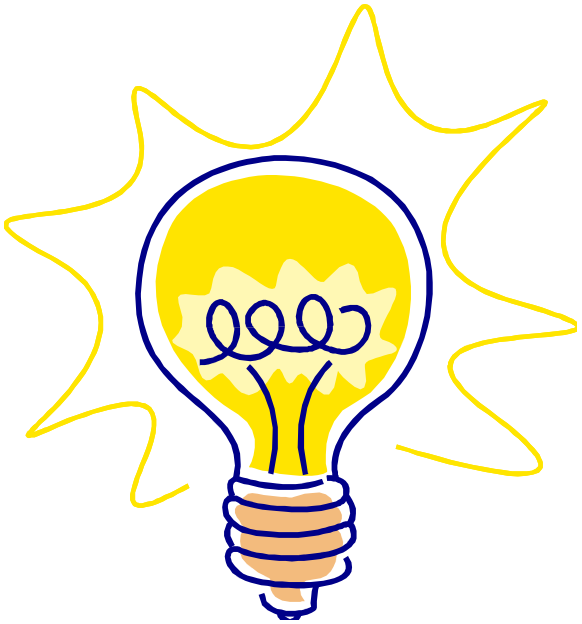
Sandia National Laboratories

- Faster Verification
  - Faster error alerts
  - Reusability across verification methods
- Thoroughness
  - Persistent property checking
  - Support for complex property definitions

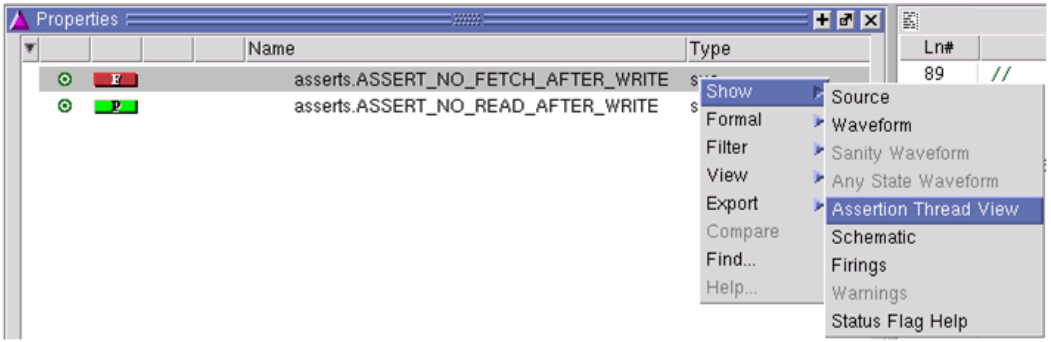
Results from using SVA

Sandia National Laboratories

- Insight
  - Most issues were found when *writing the assertions and realizing they would fail*, not when actually running the assertions
- *"Once we started writing assertions, we started thinking about our design in different ways."*

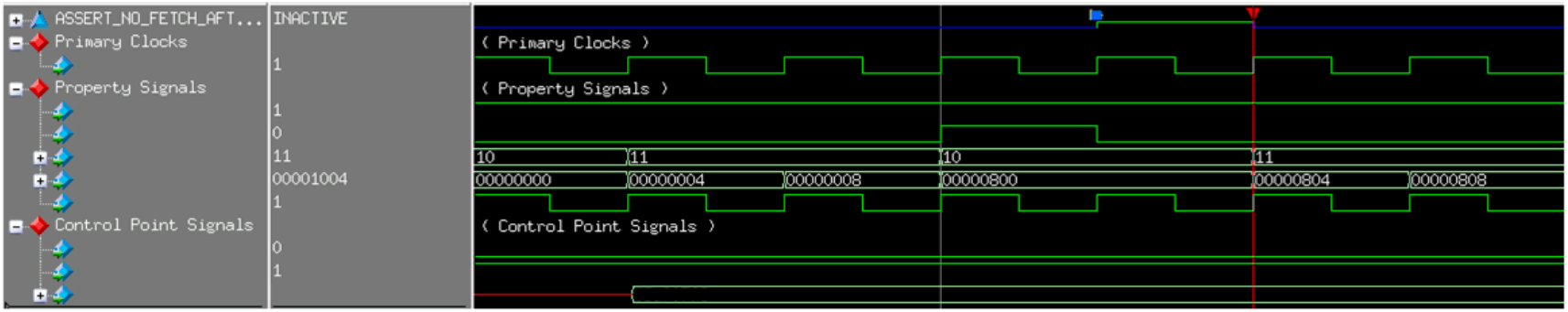


SVA Example - Questa



Formal View

Assertion Thread View



Waveform View

Drawbacks of SVA Language

Sandia National Laboratories

- Support
  - Constructs with limited vendor support
    - Driven by demand, not language reference manual
    - Example: “checker” block
  - Inconsistent between tools
    - Example: “assume” statement in a non-formal context

Sandia National Laboratories

Our experiences with

FORMAL VERIFICATION

Basics of Formal Verification

Sandia National Laboratories

- Prove Properties of a System
  - User provides properties to *assume* and properties to *assert*
    - State space is restricted by the assumed properties
    - Formal tool explores the state space of the DUT
    - Tries to reach the target properties in a mathematically provable way
  - Three possible results returned by a formal analysis attempt
    1. *Property is proven **true***
    2. *Property is proven **false**, and a counterexample is provided*
    3. *Attempt is **inconclusive** given the time*
       - Add more assumptions
       - Partition the design further
       - Refine the requirement

Benefits of Formal Verification

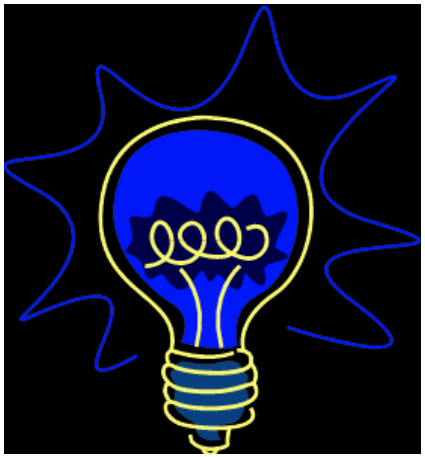
Sandia National Laboratories

- Exhaustive
  - A formal result is a proof, not a simulation
- Provides Counterexamples
  - Failures are illuminated, not just uncovered
  - Formal tool provides a sequence of events that leads to the failure

Results of Using Formal Verification

Sandia National Laboratories

- Gives Insight into Design
  - Shows how the design is logically partitioned
  - Shows which parts of design are exercised by which assertions
  - Shows how blocks of the design can be replaced with abstractions for higher-level verification
- *"It's like shining a black light on your design. You can't see everything, but what you **can** see really shows up."*



Results of Using Formal Verification

Sandia National Laboratories

- Good for Security Applications
  - "Always"/"never" results
- Good for Naturally Partitioned Designs
  - Lowest levels of hierarchy
  - Constructs that translate naturally to "assume" statements
    - Modes of operation
    - Hierarchy with proven abstractions
- *"We weren't even thinking of other techniques."*

Formal: Catching a Bug

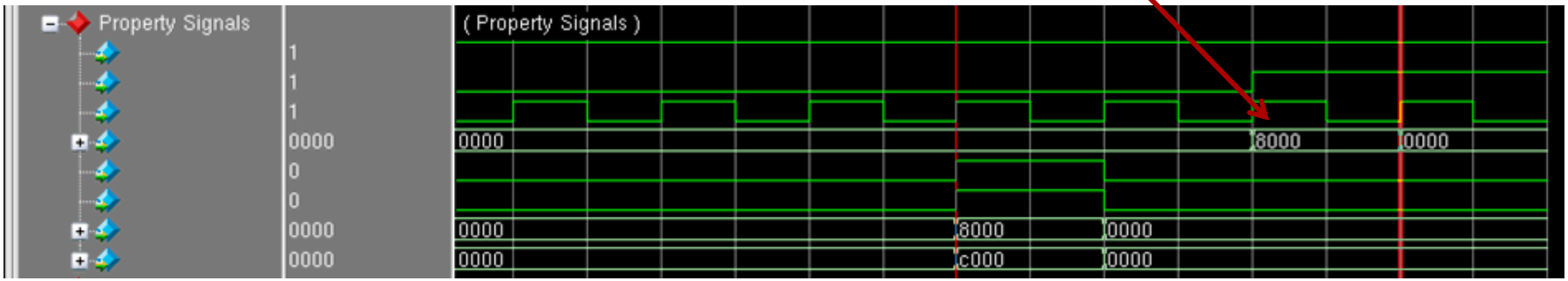
- **Always** (Reliability)
  - The output will **always** transition *following* an authorized event

```
authorized_event  |=>  $rose(output)
```

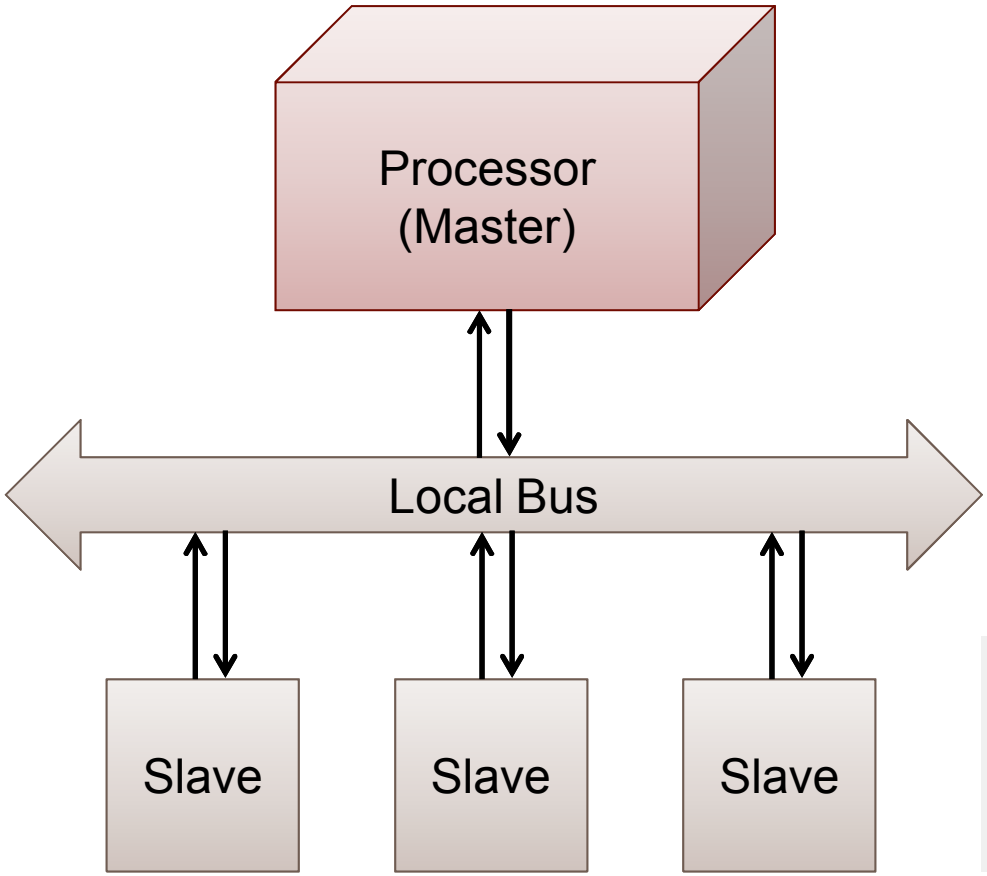
- **Never** (Security)
  - The output will **never** transition unless an authorized event *has occurred*.

```
$rose(output)  |->  $past(authorized_event)
```

Result: the circuit was shown to pass **invalid data** for one cycle following an alarm.

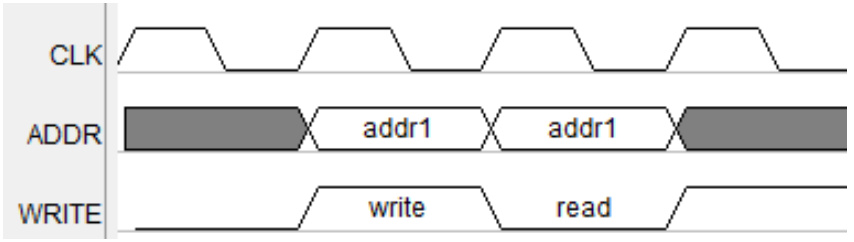


Formal: Questioning an Assumption



Simplifying assumption: the processor, acting master, will **never** read data from the same address it wrote to in the previous address cycle.

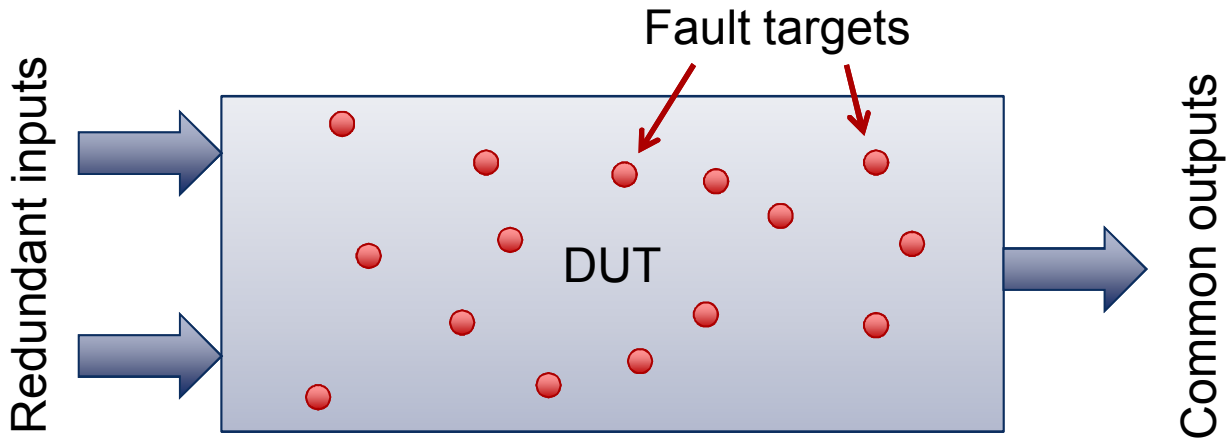
Question: Does this simplifying assumption **always** hold true?



Result: the assumption was proven to always hold true, under all conditions.

Formal: Proving Robustness

- Will the circuit behave properly in the presence of faults?



- Assertions
  1. If the inputs do not match, the output will be inhibited.
  2. If there is a single fault, property #1 will still hold.
- Findings
  - Formal methods combined with fault injection found issues with fault tolerance that would not have been found with other techniques
  - Found problematic implementation of the correct design!

Drawbacks of Formal Verification

Sandia National Laboratories

- Impractical for Some Designs
  - State space is enormous
  - Counters, memories are especially vulnerable
  - Limited support for ways to work around this problem
- Learning Curve
  - Learn how to write formally verifiable properties
  - Learn how to partition a design into formally verifiable blocks
  - Learn how to use assumptions to restrict a problem
  - (Still much easier than UVM!)

Sandia National Laboratories

Our experiences with

THE UNIVERSAL VERIFICATION METHODOLOGY (UVM)

Basics of UVM

Sandia National Laboratories

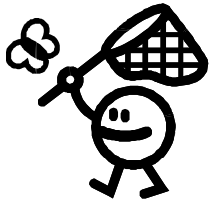
- Constrained Random Verification
- Transaction-Level Modeling
- Complex Framework of Reusable Verification Components
  - UVM Class Library
    - Prepackaged components (agents, monitors, sequencers, drivers, etc.)
    - Founded on object-oriented programming concepts
    - Supports register layer modeling
    - Intended to be highly extensible and reusable
  - UVM Macros
    - Streamline transaction-level operations
    - Streamline reporting

Benefits of UVM

Sandia National Laboratories

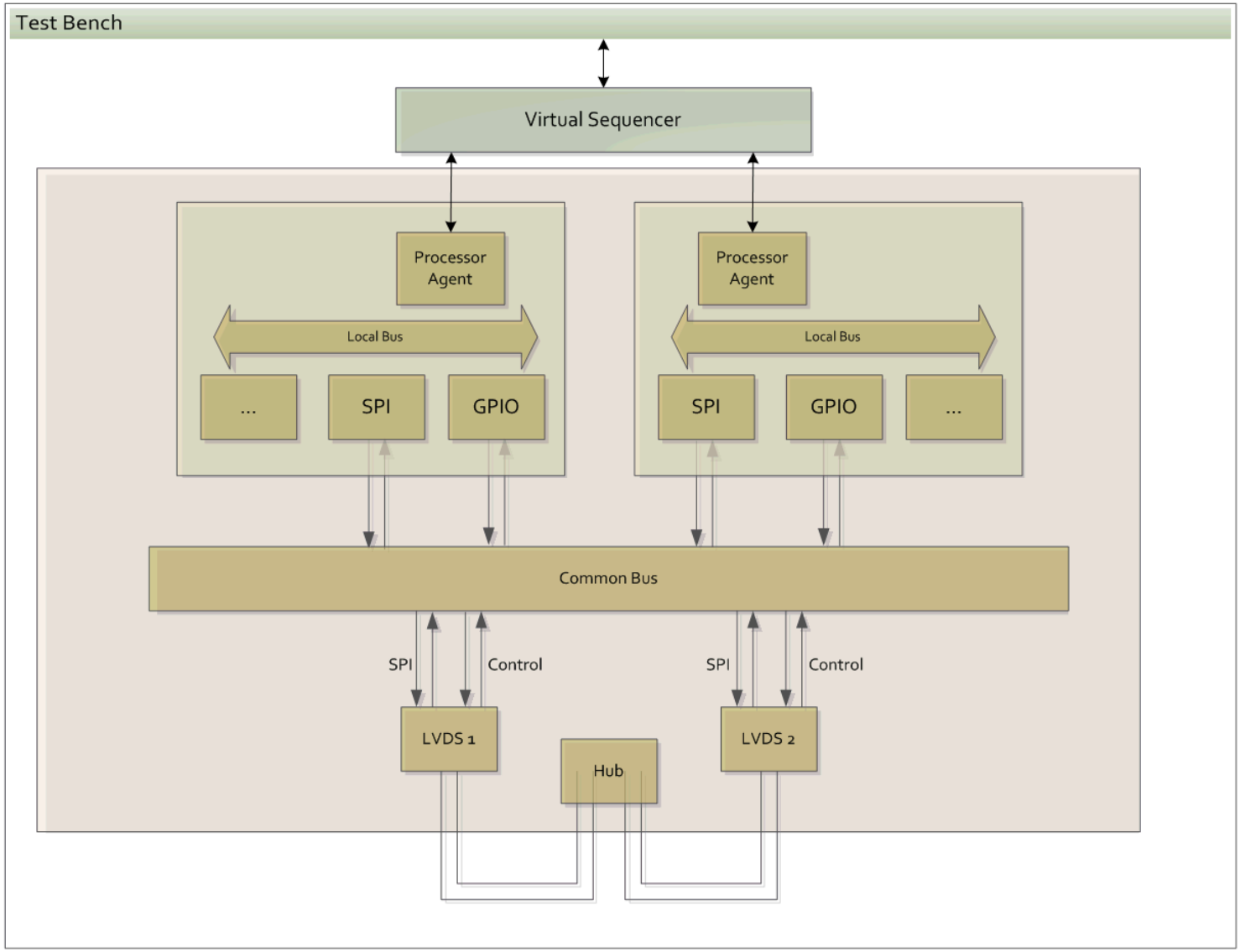
- Extensibility
  - Override classes to support alternate design
  - Override classes to change stimulus
- Reusability
  - Reuse framework in future designs and other tests
  - Test different architectures without rewriting tests
    - Test at an RTL-free level of abstraction
    - Test a similar architecture with a completely different bus protocol
    - Test a second spin of the design with different timing constraints
  - Exercise new demands on the design without copying lots of code
  - *"We adopted UVM because modification of the requirements made it hard to leverage from prototype versions."*

Benefits of UVM



- Casts a Wide Net
  - Well-positioned to deal with increasingly complex systems
  - Finds corner cases that directed simulation does not look for
  - Implement intelligent constraints and then run random transactions
  - *"We adopted UVM because success at first cut became critical for overall cost and development time reduction."*
- Well Documented
  - 852-page UVM Class Reference
- Standardized
  - Accellera UVM 1.1 Standard

UVM Design Architecture



Results from Using UVM

Sandia National Laboratories

- Finding: Unexpected Behavior in SPI Subsystem
  - Only occurred:
    - In certain modes
    - With certain parameter values
    - Transmitting certain patterns of data
  - *An interaction we weren't testing for!*

Drawbacks of UVM

Sandia National Laboratories

- Very Steep Learning Curve
  - Large framework of verification components to put in place
  - Building a thorough checker is like redesigning the DUT
  - Many classes, methods, and macros in the reference manual to learn
  - Lots of "gotchas"
    - e.g. Override the "body" method, but never call it directly
  - Lots of redundant options - not clear which approaches are equivalent
- Hard to Debug
  - An error might be:
    - Faulty verification framework
    - Faulty checker
    - Faulty DUT
  - In the finding in last slide, it took two weeks to determine it was a DUT issue!

Sandia National Laboratories

Our experiences with

THE UNIFIED COVERAGE DATABASE (UCDB)

Basics of UCDB

Sandia National Laboratories

- Mentor Graphics File Format
- Database File Generated by the Verification Tool
- Contains a Wide Variety of Coverage Data
  - Code coverage
  - Covergroups
  - Assertions
  - Formal results
- Can be Customized, Restricted, Merged, and Post-Processed
- Recently (2012) Standardized in Accellera UCIS 1.0 Standard
  - UCISDB – Unified Coverage Interoperability Standard Database

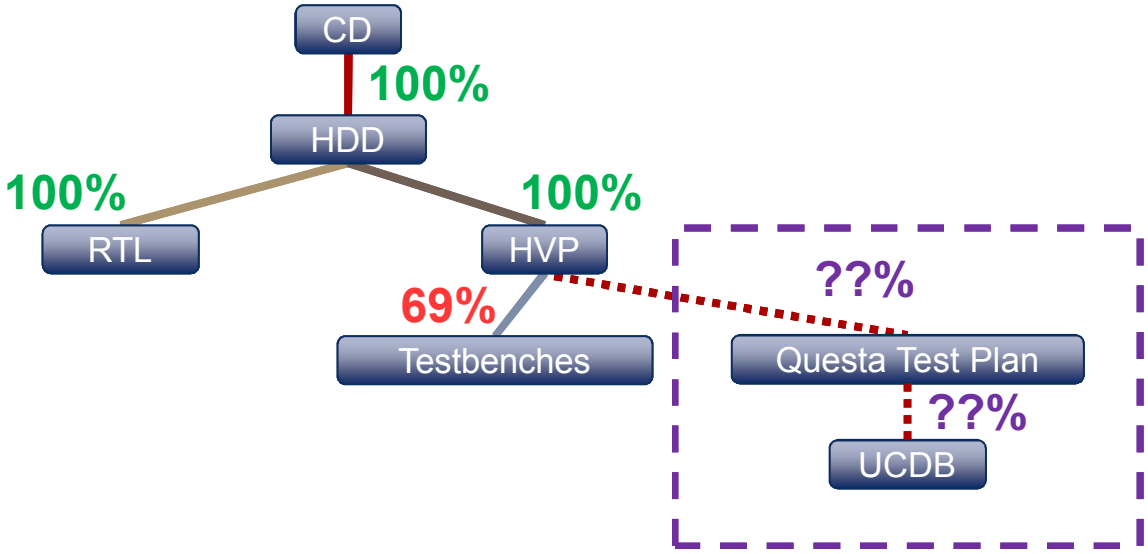
Benefits of UCDB

Sandia National Laboratories

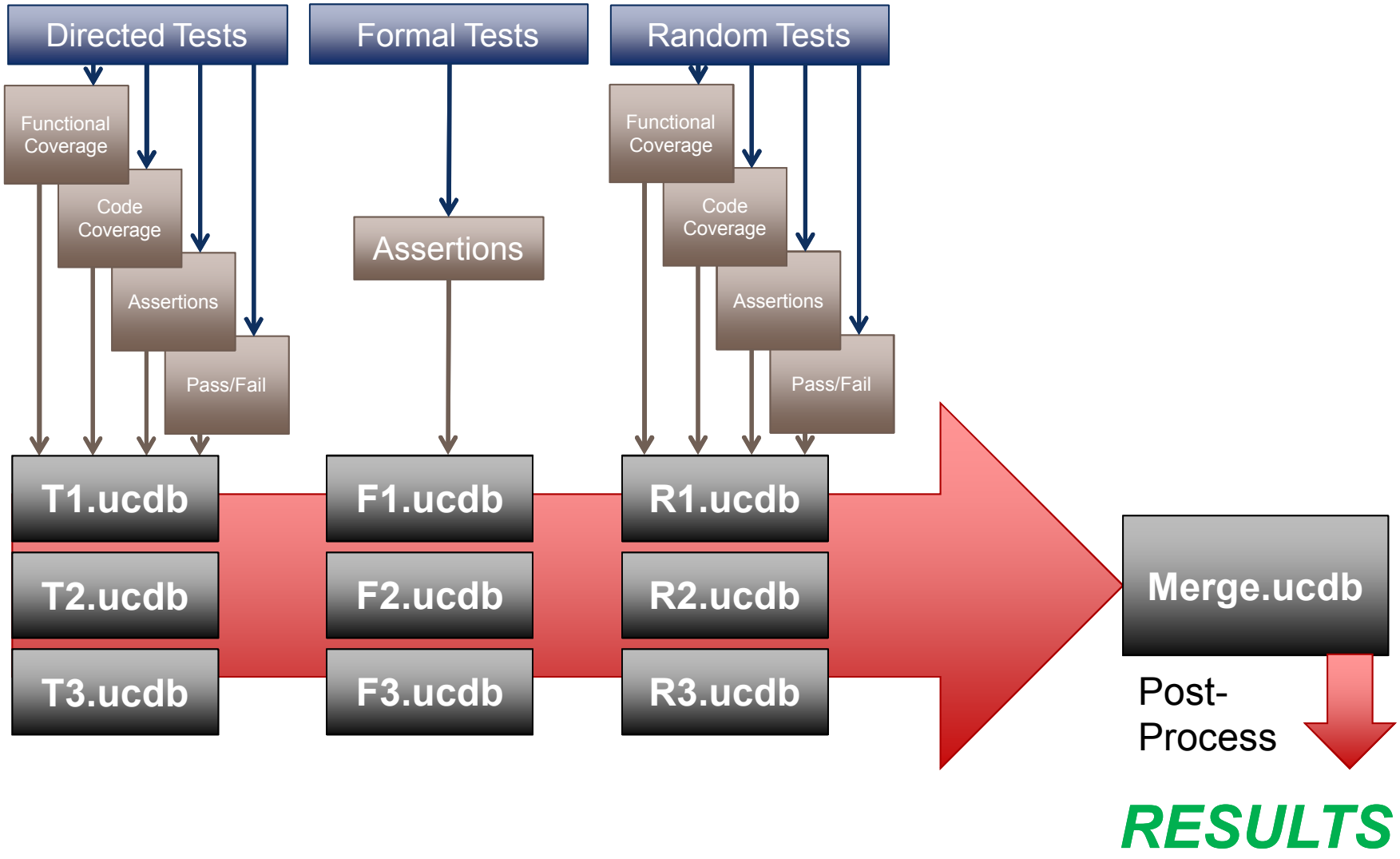
- Fewer Files
  - All types of coverage in the same file
  - Can be merged into a single database
- User-Defined Scope
  - Can be limited to only the design units of interest
  - Can be restricted to limit file size
- Post-Processing Features
  - Customizable merging
  - Test ranking
  - Annotated code

Results of Using UCDB

- Merge and Report Features Increased Efficiency
  - Division of labor
  - Easier missing coverage reporting
  - Easier statistics tracking
- Future Direction: Integrate UCDB Data Into ReqTracer

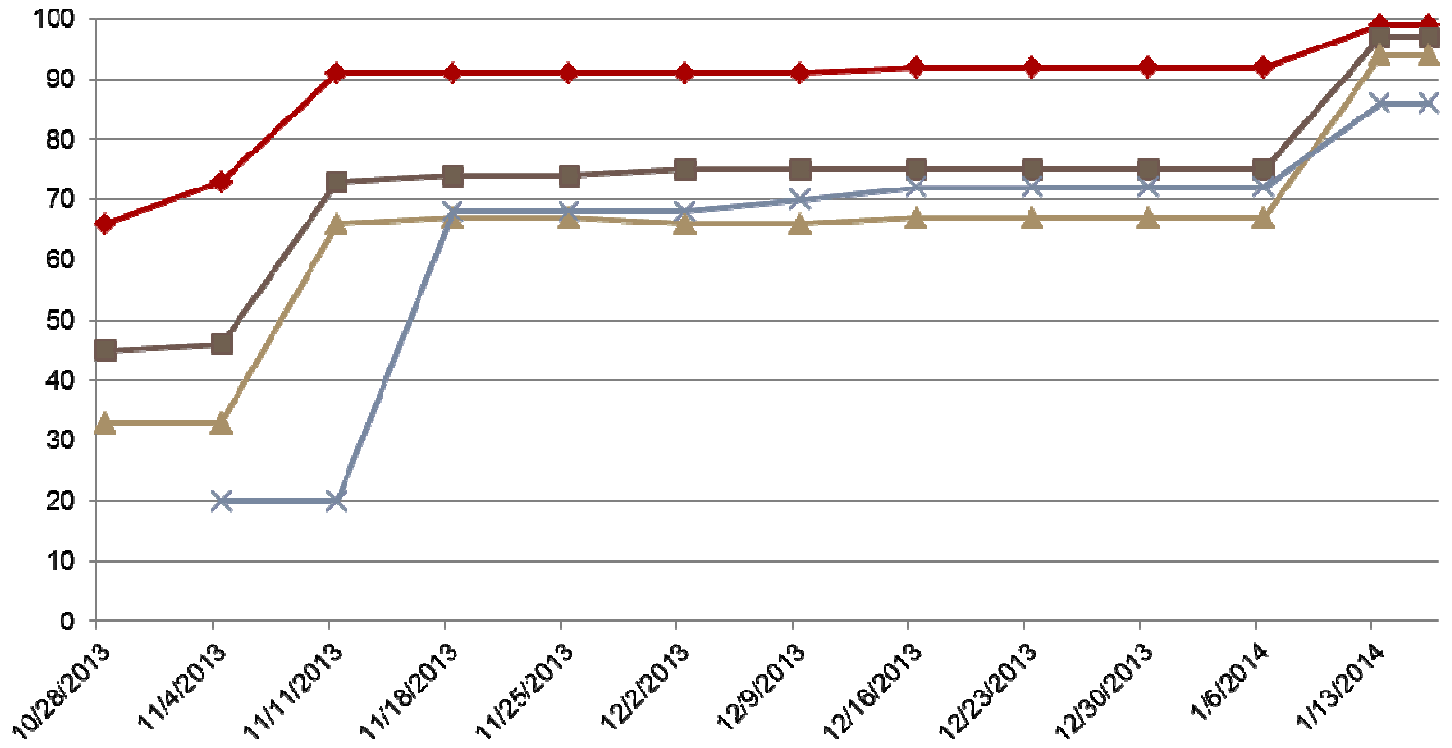


UCDB Flow


Directed Tests
Formal Tests
Random Tests
Functional Coverage
Code Coverage
Assertions
Pass/Fail
Assertions
Functional Coverage
Code Coverage
Assertions
Pass/Fail
T1.ucdb
T2.ucdb
T3.ucdb
F1.ucdb
F2.ucdb
F3.ucdb
R1.ucdb
R2.ucdb
R3.ucdb
Merge.ucdb
Post-Process
RESULTS

Coverage Tracking

Coverage Type	Current	Goal
Statement	99%	100%
Branch	97%	100%
Toggle	94%	100%
Expression	97%	85%
Covergroup	86%	100%



Drawbacks of UCDB

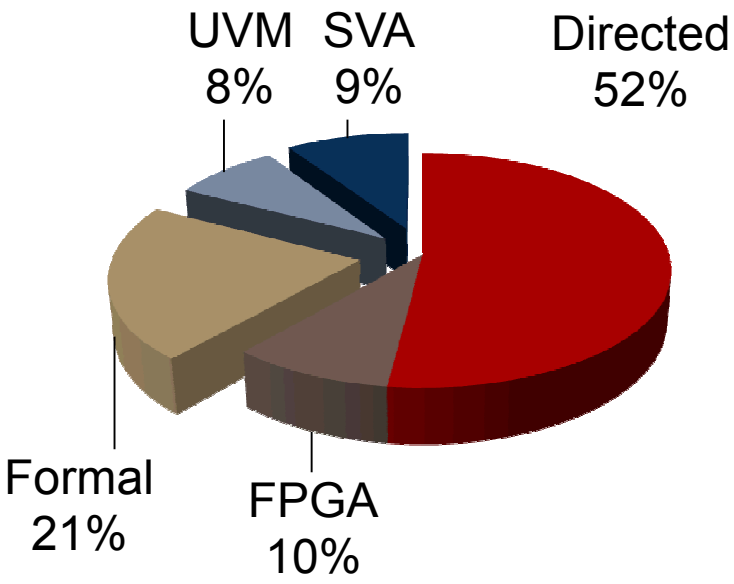
Sandia National Laboratories

- Complicated Post-Processing
  - Data won't make sense without:
    - The right collection options
    - The right exclusions
    - The right merge options
- Limited Support
  - Only recently standardized as UCISDB
  - (Non-Mentor) Vendor support is limited

Conclusion

Advanced Verification Techniques are:

- A Lot of Work



- ... For a Big Payoff!

Questions?

ASIC/SoC

Mixed Signal

U.S. DEPARTMENT OF ENERGY

NNSA

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.