The How To’s of Metric Driven Verification to Maximize Productivity

Author/Prensenter: Matt Graham
Author: John Brennan
Cadence Design Systems, Inc.
The How To’s of Metric Driven Verification to Maximize Productivity

Cadence Design Systems, Inc.
DVCon Europe – Munich, Germany
October 14, 2014
Agenda

Section 1: MDV Methodology IP to SoC Verification

Section 2: MDV Approaches Beyond RTL IP Level

Section 3: Team Based Verification Management

Section 4: MDV In Action
But this is not the end of the story, next comes SOC level verification and associated challenges.

IP/Subsystem UVM e/SV Metric Driven Verification
Main Verification Flow Being Adopted Past 15 years

- Verification Environment
  - Automatic Stimulus Generation
  - BFM Signal Layer
  - Coverage Monitor
  - Data and Assertion Checkers
  - Design IP
  - RTL Simulator

Verify IP exhaustively should work in ANY SoC context
SoC HW/SW Integration & Verification Challenges

Multi-core SW development and HW/SW verification

Integration of Analog-mixed signal

Complex low-power design features need to be verified at SoC level spanning HW/SW

SoC SW and IP integration with 10’s to 100’s of IPs

SoC Coherent and Non-Coherent Interconnect Complexity key to System Performance

Requires many development environments on different platforms

Timing, CDC, clk/reset (x-propagation), gate-level simulation

High speed, wired interface peripherals

Other peripherals

Low speed peripherals

Arm CPU Subsystem

Customer’s Application Specific Components

SoC Interconnect Fabric

Cache Coherent Fabric

A15 A15 A7 A7
L2 cache L2 cache

Cache Coherent Fabric

3D Graphics Core
Modem
Application Accelerators

AES

... ...

A15 A7
L2 cache

Cache Coherent Fabric

DDR3 USB3.0 PCIe Gen 2.3 Ethernet

PHY 3.0 PHY 2.0 PHY

High speed, wired interface peripherals

Other peripherals

Low speed peripherals

HDMI SATA MIPI

WLAN LTE

GPIO UART INTC I2C

Display PMU SPI

JTAG Timer

Complex low-power design features need to be verified at SoC level spanning HW/SW

Timing, CDC, clk/reset (x-propagation), gate-level simulation

Integration of Analog-mixed signal

SoC Coherent and Non-Coherent Interconnect Complexity key to System Performance

Requires many development environments on different platforms

Multi-core SW development and HW/SW verification

Integration of Analog-mixed signal

Complex low-power design features need to be verified at SoC level spanning HW/SW

SoC SW and IP integration with 10’s to 100’s of IPs

SoC Coherent and Non-Coherent Interconnect Complexity key to System Performance

Requires many development environments on different platforms

Timing, CDC, clk/reset (x-propagation), gate-level simulation
## Need for Concurrent HW/SW Development

### Shift Left

### Serial HW->SW Development

<table>
<thead>
<tr>
<th>HW/SW Spec</th>
<th>ROM Content</th>
<th>Diagnostics &amp; Firmware</th>
<th>Drivers / RTOS / Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>Chip</td>
<td>Prototype</td>
<td>Field test</td>
</tr>
</tbody>
</table>

### Concurrent HW->SW Development

- Integrate HW/SW early and often
- HW designed and verified in SW context
- Software exposed early to HW spec changes
- Verify SoC can support required SW applications

**Time to market advantage**
Many Platforms for IP to SoC HW/SW Development
Verification and Software platforms need to interoperate

- **SDK OS Simulation**
  - Highest speed
  - Earliest in the flow
  - Ignore hardware

- **Virtual Platform**
  - Almost at speed
  - Less accurate (or slower)
  - Before RTL
  - Great to debug (but less detail)
  - Easy replication

- **Formal Analysis**
  - Non-scalable
  - Exhaustive
  - Early RTL
  - Great for IP
  - No SW execution

- **HDL Simulation**
  - KHz range
  - Accurate
  - Excellent HW debug
  - Broadly available
  - Mixed-abstractions
  - Limited SW execution

- **Acceleration Emulation**
  - MHz Range
  - RTL accurate
  - After RTL is available
  - Good to debug with full detail
  - Expensive to replicate

- **FPGA Prototype**
  - 10’s of MHz
  - RTL accurate
  - After stable RTL is available
  - OK to debug
  - More expensive than software to replicate

- **Prototyping Board**
  - Real time speed
  - Fully accurate
  - Post Silicon
  - Difficult to debug
  - Sometimes hard to replicate
Challenges with Many Disconnected SoC Development Environments

- Many specialized engineering resources required
- Significant development effort for each environment
- Limited sharing of models/VIP between environments
- Difficult to reuse tests across environments
- A lot of effort to migrate between environments
IP to SoC HW/SW Integration & Verification Flows

- Applications (Basic to Angry Birds)
- Middleware (Graphics, Audio)
- OS & Drivers (Linux, Android)
- Bare Metal SW
- System on Chip
- Sub-System
- IP

Hybrid HW/SW Integration

- SW-Driven Verification
  - Coverage
  - Unreachability Analysis
  - SoC Connectivity Verification
- UVM e/SV Coverage Driven Verification
  - Register Map Validation
  - Formal Assertion Based Verification

SoC IP Integration Verification & Architecture Analysis

- Gate Level Verification
- UVM e/SV Metric Driven Verification & Performance Analysis
- SW-Driven Verification
- Coverage Unreachability Analysis
- SoC Connectivity Verification
- Register Map Validation
- Formal Assertion Based Verification

TLM Design & Verification

Spec

Silicon
IP to SoC Pre-Silicon Verification Platforms
Expanding Requirements for Metric Driven Verification

- Consistent planning and management across different flows
  - CDV, Formal, Low Power, AMS, Use Case SW-Driven

- Need to support large-scale, multi-site SoC projects
  - Scalability of coverage merging and analysis
  - Scalability of aggregating & archiving data from different teams & sites

- Consistent metrics support across verification platforms
  - Simulation, Acceleration, Emulation, Virtual Platform

- Uniform metrics based project tracking from IP to SoC flows
  - Flexibility to “mine” verification database for customized reporting
Section 1: Conclusions and Summary

• Key to optimized IP to SoC verification flow is choosing the best platform for the specific verification task with the right methodology
  – For efficient flow, requires highly integrated SoC development platforms

• Scalable metrics-based verification planning & management across multiple platforms and verification flows

• Early HW/SW Integration critical for fastest time to market
  – Must continually verify HW in SW context

• SW-Driven Verification best suited for SoC integration verification & use case verification
  – Horizontal reuse across virtual, simulation, emulation, & FPGA

• UVM SV/e MDV best suited for IP/Subsystem verification on RTL Simulator or HW Accelerator
  – Use TLM design & verification flow for more efficient development of new IP
  – Formal verification integrated for specific tasks to augment simulation-based verification
Agenda

Section 1: MDV Methodology IP to SoC Verification

Section 2: MDV Approaches Beyond RTL IP Level

Section 3: Team Based Verification Management

Section 4: MDV In Action
MDV: Correlating Metrics with Verification Concerns
Data Driven Decisions and Objective Signoff Criteria

Execute Tests

View Coverage

98% Coverage Grade

68% Overall Grade

Roll Up Coverage Results

Feature A

Feature B

Feature C

Organize By a Plan

MDV Unique Value
Planning is Essential

1. Plan Specifies Metrics Required for DUT Features:
   - Verification Goals based on:
     - Analysis of specifications
     - Experience of the team

2. Plan Provides Feature Based Tracking of Progress
   - Implemented metrics to concretely measure Goals
   - Regression results annotated back to Plan Features

DUT Feature-Based Plan
- Input Interface A
  - Coverage & check requirements
  - 66%
- Core Function B
  - Coverage & check requirements
  - 100%
- Output Interface C
  - Coverage & check requirements
  - 33%
Benefits of an Executable Feature-based Plan

**Without a vPlan**

*(Coverage Driven Verification)*

- Without a vPlan, all coverage appears flat
- Difficult to correlate to verification plan
- Difficult to differentiate between high priority and lower priority coverage

**With a vPlan**

*(Plan based Metric Driven Verification)*

- With a vPlan, sections can be created to organize by feature areas of interest
- Various types of coverage/check metrics can be mapped to each section
- Very easy to measure progress relative to your plan and priorities
IP/Subsystem Verification Flow Concerns
Must be very thorough for efficient SoC verification

• Verification Concerns
  – Interface protocol compliance
  – IP/Subsystem configuration, operations, and data paths
  – Low power modeling
  – Micro-architecture design features
  – Stress testing of complex traffic scenarios

• Create UVM e/SV IP/Subsystem Verification Environment
  – Augment with formal for block level and RTL linting
  – Commercial interface VIP for standard protocols
  – Reuse interface UVCs for proprietary protocols
  – Constrained-random stimulus sequences
  – Reference model, register modeling, and scoreboard for data checking
  – Assertions for protocol checking
  – Functional coverage for measuring features exercised
  – Code coverage for measuring HDL implementation exercised
    – Formal unreachability analysis of code coverage to reach 100%
  – Reuse IP Verification Environments to create Subsystem Testbench
Traditional MDV Methodology
IP and Subsystem Verification

vPlan “Goals”

stimulus sequences

 sequencer

seed new test

 0x223F stimulus
 0xA30E stimulus
 0x94D7 stimulus
 0x95F78 stimulus
 0x3767 stimulus
 0xCX18 stimulus
 0xdA83 stimulus
 0xB2A1F stimulus
 0x95FB stimulus
 0x382E stimulus

scoreboard

monitor

cov check

cov check

monitor

stimulus sequences

DUT

slave

coverage collection

stimulus sequences

stimulus sequences

stimulus sequences

stimulus sequences

stimulus sequences
SoC Interconnect Verification & Performance Concerns

• SoC Interconnect includes hierarchy of connectivity across IPs and memories

• Interconnect Functional Verification
  – Address map and decoding
  – Configuration and address remapping
  – All Initiator to target paths
  – All target from initiator paths
  – Multi-protocol transaction transformations
  – Cache behavior for cache coherent interconnect

• Interconnect (and Memory subsystem) Performance Verification and analysis
  – Latency for critical data paths
  – Bandwidth and throughput for heavy traffic stress scenarios
  – QoS/QVN requirements
  – Cache performance for critical use cases
# SoC Interconnect Verification vPlan

## Design Feature

<table>
<thead>
<tr>
<th>Design Feature</th>
<th>Coverage Metric</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address map and decoding</td>
<td>Functional</td>
<td>Sim</td>
</tr>
<tr>
<td>Configuration and address remapping</td>
<td>Functional</td>
<td>Sim</td>
</tr>
<tr>
<td>All Initiator to all target paths</td>
<td>Functional</td>
<td>Sim</td>
</tr>
<tr>
<td>All target from all initiator paths</td>
<td>Functional</td>
<td>Sim</td>
</tr>
<tr>
<td>Multi-protocol transaction transformations across interconnect</td>
<td>Functional, Assertion</td>
<td>Sim</td>
</tr>
<tr>
<td>Cache behavior for cache coherent interconnect</td>
<td>Functional, Assertion</td>
<td>Sim</td>
</tr>
</tbody>
</table>

- Automatic generation of interconnect TB
- Built on UVM-based VIP
- Same Metrics as IP Verification

![Diagram](attachment:image.png)
SoC IP Integration Verification Concerns

• Signal Connectivity in SoC
  – IP connectivity in SoC
  – Clock, interrupt, & reset connectivity
  – IO Pad connectivity

• IP Configuration, Primary Operations, & Data Path Connectivity in SoC context
  – SoC clocking & reset modes
  – IP access to Memory
  – IP I/O access and data path transaction flow
  – IP programmer’s view and primary operations from SW Driver API
  – IP Interrupt scenarios

• IP Low power integration
  – Hierarchical low power control and power modes – power shut-off and voltage configurations
  – Low power interconnect and interface – isolation behavior
# SoC IP Integration Verification vPlan

<table>
<thead>
<tr>
<th>Design Feature</th>
<th>Coverage Metric</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Connectivity in SoC</td>
<td>Formal Assertion, Toggle</td>
<td>Formal Sim</td>
</tr>
<tr>
<td>Clock, interrupt, &amp; reset connectivity</td>
<td>Formal Assertion</td>
<td>Formal</td>
</tr>
<tr>
<td>IO Pad connectivity</td>
<td>Formal Assertion</td>
<td>Formal</td>
</tr>
<tr>
<td>IP access to Memory</td>
<td>Functional, Toggle</td>
<td>Sim</td>
</tr>
<tr>
<td>IP I/O access and data path transaction flow</td>
<td>Functional, Toggle</td>
<td>Sim</td>
</tr>
<tr>
<td>IP programmer’s view and primary operations from SW Driver API</td>
<td>Functional</td>
<td>Sim</td>
</tr>
<tr>
<td>IP Interrupt scenarios</td>
<td>Functional, Assertion</td>
<td>Sim</td>
</tr>
<tr>
<td>SoC boot/initialization scenarios</td>
<td>Functional, Assertion</td>
<td>Sim/Accel</td>
</tr>
<tr>
<td>Hierarchical low power control and power modes – power shut-off &amp; voltage</td>
<td>Functional, Assertion</td>
<td>Sim/Accel</td>
</tr>
<tr>
<td>Low power interconnect &amp; interface – isolation behavior</td>
<td>Functional, Assertion</td>
<td>Sim/Accel</td>
</tr>
</tbody>
</table>
SoC Use Case Verification Concerns

- **SoC level features**
  - SoC FW boot up and initialization
  - Primary IO Pad configurations
  - Scan chain connectivity and test mode operations

- **End application use case scenarios**
  - Verified on firmware or lower layers of SW stack
  - Adherence to power and performance requirements
  - Cache and IO Coherency
  - End to end data path scenarios
    - E.g., CPU programs camera \(\rightarrow\) camera sends image data \(\rightarrow\) CPU processes image \(\rightarrow\) image sent to display
  - Stress tests on resource contention and multi-master scenarios
  - Cross use case scenarios with low power configurations, modes & sequencing
## SoC Use Case Verification vPlan

<table>
<thead>
<tr>
<th>Design Feature</th>
<th>Coverage Metric</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoC FW boot up and initialization</td>
<td>Functional, Assertion</td>
<td>Sim/Accel</td>
</tr>
<tr>
<td>Primary IO Pad configurations</td>
<td>Functional, Toggle</td>
<td>Sim/Accel</td>
</tr>
<tr>
<td>Scan chain connectivity and test mode operations</td>
<td>Functional, Assertion</td>
<td>Sim/Accel</td>
</tr>
<tr>
<td>Cache and IO Coherency</td>
<td>Functional, Assertion</td>
<td>Sim/Accel</td>
</tr>
<tr>
<td>End to end data path scenarios – functional, power, &amp; performance</td>
<td>Functional, Assertion</td>
<td>Sim/Accel</td>
</tr>
<tr>
<td>Stress tests on resource contention and multi-master scenarios</td>
<td>Functional, Assertion</td>
<td>Emulation</td>
</tr>
<tr>
<td>Cross use case scenarios with low power configurations, modes &amp; sequencing</td>
<td>Functional, Assertion</td>
<td>Emulation</td>
</tr>
</tbody>
</table>
SoC Gate Level Verification Concerns

• Gate Level Focuses on a critical sub-set of concerns
  – Tests to be run in zero delay mode
    – Reset verification, Initialization, & verification of clocking
    – Basic heart beat test to detect functional issues or issues related to X mismatches
    – Verify unexpected synthesis transformations
    – Validate functional effects after DFT and Low Power insertion
  – Tests to be run with timing
    – Tests to cover/verify STA timing constraints like multi-cycle paths, false paths
    – Test to cover asynchronous paths
    – Verify DFT with timing
    – CDC verification because automatic CDC failing too much at SoC level
    – Validation of physical netlist low power implementation
    – Safety standards on reliability testing via Fault insertion

• Uses same environment as for SoC Use Case Verification
  – Except for scan chain verification and other physical netlist artifacts
  – Same metrics and engines used as well
    – Metrics: Black box Functional, Assertion, Toggle
    – Engines: Sim/Accel
SoC HW/SW Integration Verification Concerns

• Key concerns
  – Integration & bring-up of OS & higher SW layers on RTL SoC
    – Debug integration issues on pre-silicon emulated HW platform
    – Validate OS boot up
    – Validate middleware and real applications on SoC platform
    – Validate performance requirements
  – Validate dynamic power usage for critical applications
    – Based on real running real SW application snippets
  – Graphics GPU OpenGL SW API compliance

• Effective Approaches
  – Use-cases, scenarios, and functional metrics
  – Using SW-Driven testbench approaches
  – Leverage Emulation & FPGA Prototypes
MDV Metric Options

Measuring the right metrics for the task at hand

MDV Metrics and Approaches

Test Coverage
Code Coverage
Functional Testing
Constrained Random
Coverage Driven
Plan Based

Test Driven Verification

Advanced Verification

Automation Effort

Productivity Benefits

Days

Weeks

Months

Test Coverage

Constrained Random

Coverage Driven

Plan Based

More effort, but more effective if resources permit

Leverages advanced planning technology

Greater ROI

Functional

Test

Code

DUT

Quickly establish test area

Give credit / track designers work

Effective test driven verification environment

Plan

Construct

Measure / Analyze

Execute

Plan

Construct

Measure / Analyze

Execute

© 2014 Cadence Design Systems, Inc. All rights reserved.
SoC MDV Enabler – Multi Engine, Multi Metric Plan

Executable verification plan that can link to all necessary engines and metrics

![Diagram of a verification plan editor]

- **Testcase Metrics**: 1. SoC
  - 1.1. HW/SW Integration [VIRTUAL PROTOTYPE]
  - 1.2. Use Case [SW DRIVEN]
- **Functional Metrics**: 2. Subsystem
  - 2.1. ENet [SIM-ACCEL]
  - 2.2. IO Subsystem
    - 2.2.1. Interconnect Verification [SIMULATION]
    - 2.2.2. IP Connectivity [FORMAL]
    - 2.2.3. Power Intent [SIMULATION]
- **Assertion Metrics**: 3. IP
  - 3.1. UART [SIMULATION]
  - 3.2. SMC [FORMAL]
  - 3.3. PLL [MIXED SIGNAL SIM]
  - 3.4. GPIO [SIMULATION]
  - 3.5. SPI [SIMULATION]
SoC MDV Enabler - Manage All Metrics in One Spot
Multi Engine, Multi Metric results collection in unified environment
Agenda

Section 1: MDV Methodology IP to SoC Verification

Section 2: MDV Approaches Beyond RTL IP Level

Section 3: Team Based Verification Management

Section 4: MDV In Action
Team MDV: *It Still Starts with a Plan!*

Legacy tests

- `uart_tests`
  - `uart_tests/apb_ioc_uart_1stopbit_test`
  - `uart_tests/u2a_a2u_full_random_test`
  - `uart_tests/apb_uart_rx_ts_data Libertarian`
  - `uart_tests/cdn_uart_scoreboard_traffic`
  - `uart_tests/uart_bad_parity_test`
  - `uart_tests/uart_incr_payload_test`
  - `uart_tests/uart_bad_driver_factory`
  - `uart_tests/uart_data/automation_lab1`
  - `uart_tests/uart_bad_parity_frame_test`

Requirements Management System

- `Requirements Management System`

Heterogeneous Verification Tools (ie Formal, Simulation)

- `Heterogeneous Verification Tools` (ie Formal, Simulation)

Code coverage and other metrics

- `Code coverage and other metrics`

Distributed / Hierarchical Plans

- `Distributed / Hierarchical Plans`

VIP Compliance vPlan and module level vPlans

- `VIP Compliance vPlan and module level vPlans`

The verification plan becomes the anchor to connect teams and technologies together

Brainstorming

- `Brainstorming`

Outline from a Functional Spec

- `Outline from a Functional Spec`
Plan Composure and Creation: Scalability!

- Long paths mapping metrics to plan
- Issue compounded across engines
- Further worsens at great levels of integration

- Connection to data during plan composure enables efficiency
- Export/Import to/from popular formats (XML, CSV, HTML) enables scripting, publishing, etc
- Resultant plan is mapped “Correct by construction.”
MDV for the SoC Team

- Disparate islands of information
- Inconsistent and incompatible verification approaches
  - Verification methodologies
  - Different levels of integration
  - Design technologies
- Everyone contributes, but no single coordinated view of who is doing what and how
- Goal: provide an independent yet integrated [multi-user] metric management and Plan to Closure methodology
Team MDV – Multi-user, Multi-engine, Multi-analysis

- IPB1 vPlan
- IP Block 1 Metrics (SIM/UVM)
- IPB2 vPlan
- IP Block 2 Metrics (SIM/UVM)
- Cnct vPlan
- Connectivity Rules (Formal)
- Integ vPlan
- SoC Integration Tests (SIM/TBA/ICE)

- Metrics stored in Verification DB
- Verification DB App
- SQL DB
- SoC vPlan

- Hierarchically Instantiated Reusable Plans
- MDV Team Solution
Enabling the SoC Verification Team with MDV
Next generation MDV Architecture

- File based data mgmt does not scale
- Data does not inherently stay synchronous
- Single User Environment – Difficult to Share
- Static data – reporting is manual / intensive
- Batch coverage merge not suited to 24/7 runs

- DB gives orders of magnitude greater scaling
- Data synchronicity throughout life of a project
- Multi User Environment – Easy to Share
- Dynamic – fresh data, built-in real time reports
- Continuous operations mode / “always on”
Database Driven Architecture

Complete project data

Direct access to regression data for deeper analysis.
Requirement - Unified Analysis Environment

- Analysis, exclusion and reporting
- Top level verification plan down to low level bin/line/toggle level analysis
  - Historically split between multiple tools (spreadsheet, scripts, single run coverage analysis tools)
- Single environment for ALL metric analysis
  - The right data at the right time
  - Low latency access (seconds, single click)
Requirement - Unified Analysis Environment
Includes Failure Triage

- Failure analysis complements metric roll up in MDV Cockpit
- Integration and automation with debug is a natural fit
- Push button automated rerun with dumping of debug data
- Tight integration with advanced debug platforms
  – e.g. Cadence Incisive Debug Analyzer
Unified Analysis Environment
Failure Triage Included
React Real-time to Trends

- Utilize “One Touch” real time access to up-to-date results
- Track critical verification indicators over time for visibility and predictability
- Project Definition
  - Set of data
  - Metrics to track
  - Criteria for sample
- Project Tracking and Analysis
  - Graphical and textual presentation of the metrics results over time
  - Persistent storage of trend data in the DB enables team access
Practical guidelines

• Consider the intangibles upfront
  – Human factors and verification methodology

• Plan
  – Leverage plans built at all levels of integration, with metrics from all available engines
  – Expedite plan composure with access to metric definition information
  – Instantiate IP level plans for SoC plan creation efficiency

• Collect
  – Take credit for work already done ⇒ aggregate results across users, engines, time
  – Metrics must be easily accessible (view, report, query) ⇒ utilize common database architecture

• Analyze
  – Snapshot results at regular intervals
  – Find trends, filter blips (charts, reports)

• React
  – Objective Data ⇒ Exploit connection of metrics to plan, spec
  – Instant appreciation of project wide effect of decisions based on real time data
Agenda

Section 1: MDV Methodology IP to SoC Verification

Section 2: MDV Approaches Beyond RTL IP Level

Section 3: Team Based Verification Management

Section 4: MDV In Action
Apply coverage at several stages of development cycle

**System/SoC**
- Expand coverage analysis with live interfaces and real software/firmware execution
- Use coverage techniques to optimize designs and software tests
- Continue to use assertions to ensure correctness and localize problems

**Sub-System**
- Create realistic scenarios and transactions to exercise interfaces
- Continue to use assertions to ensure correctness and localize problems
- Use code and functional coverage to monitor interfaces and testbench effectiveness

**Block/IP**
- Create synthetic scenarios to hit paths
- Use assertions to ensure correctness
- Use code and functional coverage to monitor interfaces and testbench effectiveness
## Coverage

### Use Cases

<table>
<thead>
<tr>
<th>Use Cases</th>
<th>User Explorations (examples)</th>
<th>Applicable Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SoC Integration Verification</strong></td>
<td>• What is the activity between sub-blocks?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>• What is the top level activity—perhaps 1 or 2 levels?</td>
<td></td>
</tr>
<tr>
<td><strong>Localized and Full-design focus</strong></td>
<td>• How can I run detailed coverage analysis into specific area of interest?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>• How do I achieve 100% coverage?</td>
<td></td>
</tr>
<tr>
<td><strong>Verify Modes of Operation</strong></td>
<td>• Are two processing units simultaneously active? Were interfaces active simultaneously?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>• Was interrupt issued when CPU transfers data to GPU?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• How do I correlate coverage to design features that I’m testing and measure progress against my overall verification plan?</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Design Optimization</strong></td>
<td>• How is this buffer being used? Undersized? Oversized?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>• What is the latency on this operation? Average? Max?</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Improving Hardware Coverage of Software Tests</strong></td>
<td>• How much of hardware is being exercised by software tests?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>• Should I improve my software tests to achieve higher coverage?</td>
<td>✓</td>
</tr>
</tbody>
</table>
Code coverage problem statement

- Traditional code coverage use model is difficult
  - Add an option and get overwhelmed with data
  - System verification engineers aren’t going to understand coverage data at low levels of the design
  - Even if they did, very difficult to influence low level logic from system level tests

- Solution?
  - Focus on actionable data
Integration verification

• Cover connectivity between top-level modules
  – That’s what’s new and untested
  – Lower level blocks have been verified at the block level
  – Understandable and actionable by system verification engineers
  – Typically would use toggle coverage on ports of top-level blocks
  – Block coverage not as interesting at higher levels  limited RTL
    – Might have small pieces of new system-level controller logic

<table>
<thead>
<tr>
<th># hier levels</th>
<th># toggle signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3,000</td>
</tr>
<tr>
<td>2</td>
<td>101,000</td>
</tr>
<tr>
<td>3</td>
<td>717,000</td>
</tr>
<tr>
<td>5</td>
<td>14,400,000</td>
</tr>
<tr>
<td>all</td>
<td>27,400,000</td>
</tr>
</tbody>
</table>
Localized focus—go deep

- Focus on a particular region of the design
  - Manage “amount of coverage data”
  - New or lesser tested area
  - Specific concerns with coverage in an area
  - Access to designers
  - Can merge multiple regional coverage databases into a complete view
System-level functional coverage example

Block-level verification focus

Sub-system, system-level verification focus
Verify modes of operation

- Can maintain some monitors for coverage from the subsystem level

- Fundamentally, asking different questions at the system level
  - Concerned with interactions between subsystems
  - Implies a system level test plan tied to design spec

- Verifying modes of operation
  - Were these two processing units active simultaneously?
  - Were these interfaces active simultaneously?
  - Have I received an interrupt when the CPU is transferring data to the GPU?
Verify modes of operation
Design optimization

- Investigate performance in real-world scenarios
  - What is the average utilization of the FIFO?
  - If low, can we reduce the FIFO size?
  - If high, can we expand the FIFO or can we optimize the application software?
- You may have seen cases where designers put in special counters and instrumentation
  - Covergroups and cover properties are a very easy way to instrument, plus there are standard tools for merging, reporting and analyzing results

```vhdl
covergroup cg @(posedge clk);
  coverpoint count iff (wr_en);
endgroup
```

![FIFO Utilization Graphs](image)
Improving hardware coverage of software tests

- Software-validation process often independent of hardware-verification process
- How well is the software exercising the hardware?
- Get a sense of “coverage” of the software through enabling hardware coverage during the running of software tests
Improving hardware coverage of software tests

- Software-validation process often independent of hardware-verification process
- How well is the software exercising the hardware?
- Get a sense of “coverage” of the software through enabling hardware coverage during the running of software tests
MDV Tutorial Summary
SoC MDV – Multi User, Multi Engine, Multi Metric
Environment pulling together contributions from all users, engines, and metrics

<table>
<thead>
<tr>
<th>Session Status</th>
<th>Name</th>
<th>Total Runs</th>
<th>#Passed</th>
<th>#Failed</th>
<th>Start Time</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>completed</td>
<td>Chip_ENET.14_02_26_11_35_38...</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>2/28/14</td>
<td>melancon</td>
</tr>
<tr>
<td>completed</td>
<td>ams_smoke</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1/29/14</td>
<td>magraham</td>
</tr>
<tr>
<td>completed</td>
<td>hw_sw_nightly</td>
<td>17</td>
<td>17</td>
<td>0</td>
<td>6/9/10</td>
<td>magraham</td>
</tr>
<tr>
<td>completed</td>
<td>IO_SS_Connectivity</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3/6/14</td>
<td>joseb</td>
</tr>
<tr>
<td>completed</td>
<td>SMC_Block_formal.14_02_27_13...</td>
<td>11</td>
<td>10</td>
<td>1</td>
<td>2/27/14</td>
<td>joseb</td>
</tr>
<tr>
<td>completed</td>
<td>SMC_Block_sim.14_02_27_13_37...</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2/27/14</td>
<td>joseb</td>
</tr>
<tr>
<td>completed</td>
<td>UART_Block_UNR</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2/26/14</td>
<td>joseb</td>
</tr>
<tr>
<td>completed</td>
<td>IO_Subsystem.14_02_28_16_07...</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>2/28/14</td>
<td>johnn</td>
</tr>
<tr>
<td>completed</td>
<td>UART_Block.14_02_26_11_57_41...</td>
<td>12</td>
<td>5</td>
<td>7</td>
<td>2/26/14</td>
<td>johnn</td>
</tr>
<tr>
<td>completed</td>
<td>LowPower_Verification</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1/24/14</td>
<td>johnn</td>
</tr>
</tbody>
</table>
SoC MDV – Multi Engine, Multi Metric Plan

Executable verification plan that can link to all necessary engines and metrics
SoC MDV – Multi Engine, Multi Metric Tracking

Tracking progress of contributions from all users, engines, and metrics
The How To’s of Metric Driven Verification to Maximize Productivity

• MDV has been proved to improve predictability and productivity at IP to Sub-System Levels

• Today you have learned how MDV can be expanded using vManager to operate across specialized verification engines

• Additionally you have learned how MDV can be used thru to SOC level verification.

• MDV at SOC is new and emerging, and Cadence is committed to codify and optimize this for the industry, just like we did with UVM from eRM at IP levels

• Thank you for your participation today. You can learn more about the vManager Solution and MDV on the Cadence website – www.cadence.com
Questions ?