
# Advanced Functional Verification and Debug Methods of PCIe® Designs

Chris Browy

VP Sales/Marketing

Avery Design Systems

# Outline

- Role of commercial Verification IP (VIP)
- Core-level verification challenges
- Compliance checklist and testing foundations
- Beyond compliance checklist-based testing
- Chip-level verification challenges
- Lessons learned

# Role of Commercial VIP

- Support core through chip-level verification
  - ✓ PCIe® cores (EP,RC) and switch designs
  - ✓ Full-chip
- BFMs support normal and error behaviors
- Checklist assertions detect non-compliance
- Core-level testsuites
  - ✓ verify nominal and optional operational sequences
  - ✓ grade compliance coverage

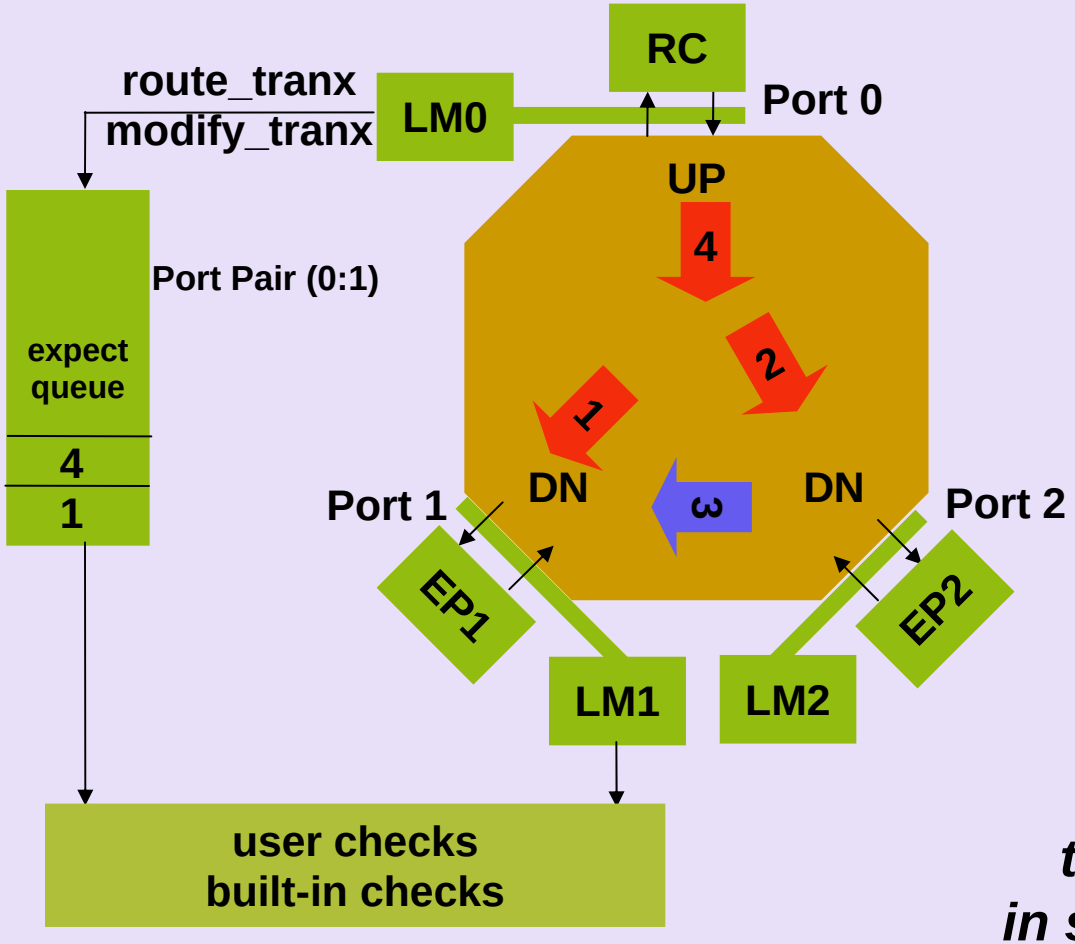
# Core-level Verification Challenges

- Multiple targets
  - ✓ Many component types (EP, RC, SW, PIPE PHY)
  - ✓ Specification revisions (1.0a, 1.1, 2.0, SR-IOV)
  - ✓ ECNs (ARI, ATS)
- Robustness of BFMs
  - ✓ Implementation variations (ex. LW Upconfigure)
- Out of spec design issues
  - ✓ Not all designs require 100% compliance
  - ✓ “Quick” simulation design modes

# Core-level Verification Challenges

- End 2 end verification
  - ✓ Port driver application interfaces are non-standard and impose limitations effecting how core can be used
  - ✓ End 2 end checkers provide background verification (ex. Transaction ordering rules across switch)
- Testsuite design
  - ✓ Context-based verification
  - ✓ Reuse and portability
  - ✓ Easy to add new tests
- Debug and observability

# End 2 End Checking



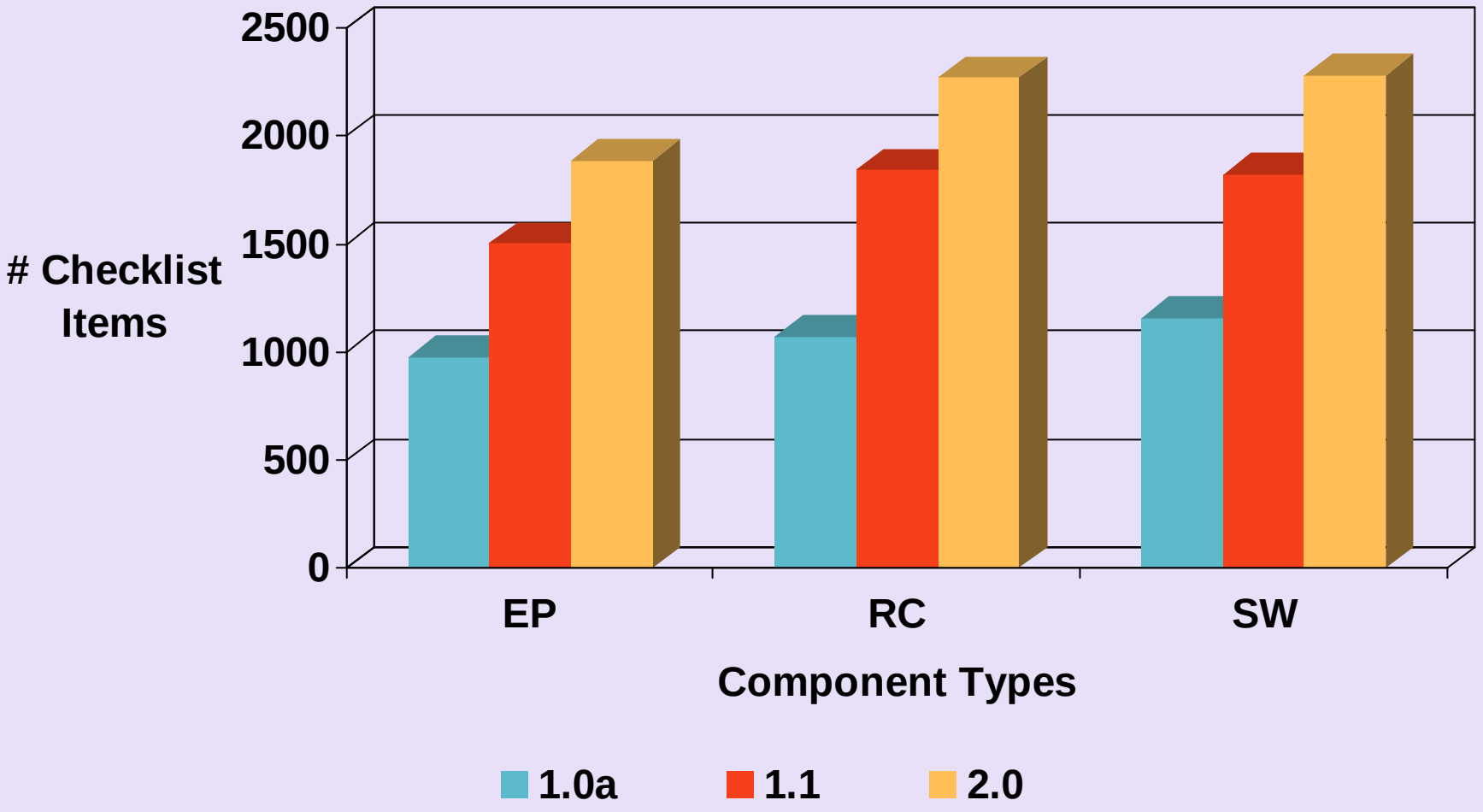

1. Receive transaction
2. Determine switch route
3. Modify packet (opt)
4. Add to expect queue
5. On output port, receive transaction
6. Execute checks
7. Update profiles

***Verifies transaction ordering rules, port arbitration, and transaction completions in switch and bridge designs***

# Compliance Checklist and Testing Foundations

- Scope of compliance checklists
  - ✓ PCIe Endpoint Checklist 1.1, 2.0
  - ✓ PCIe Root Complex Checklist 1.1, 2.0
  - ✓ PCIe Switch Checklist 1.1, 2.0
- Typical design bugs encountered
- Testing framework
- Compliance coverage report

# Scope of Compliance Checklists – Overall Summary



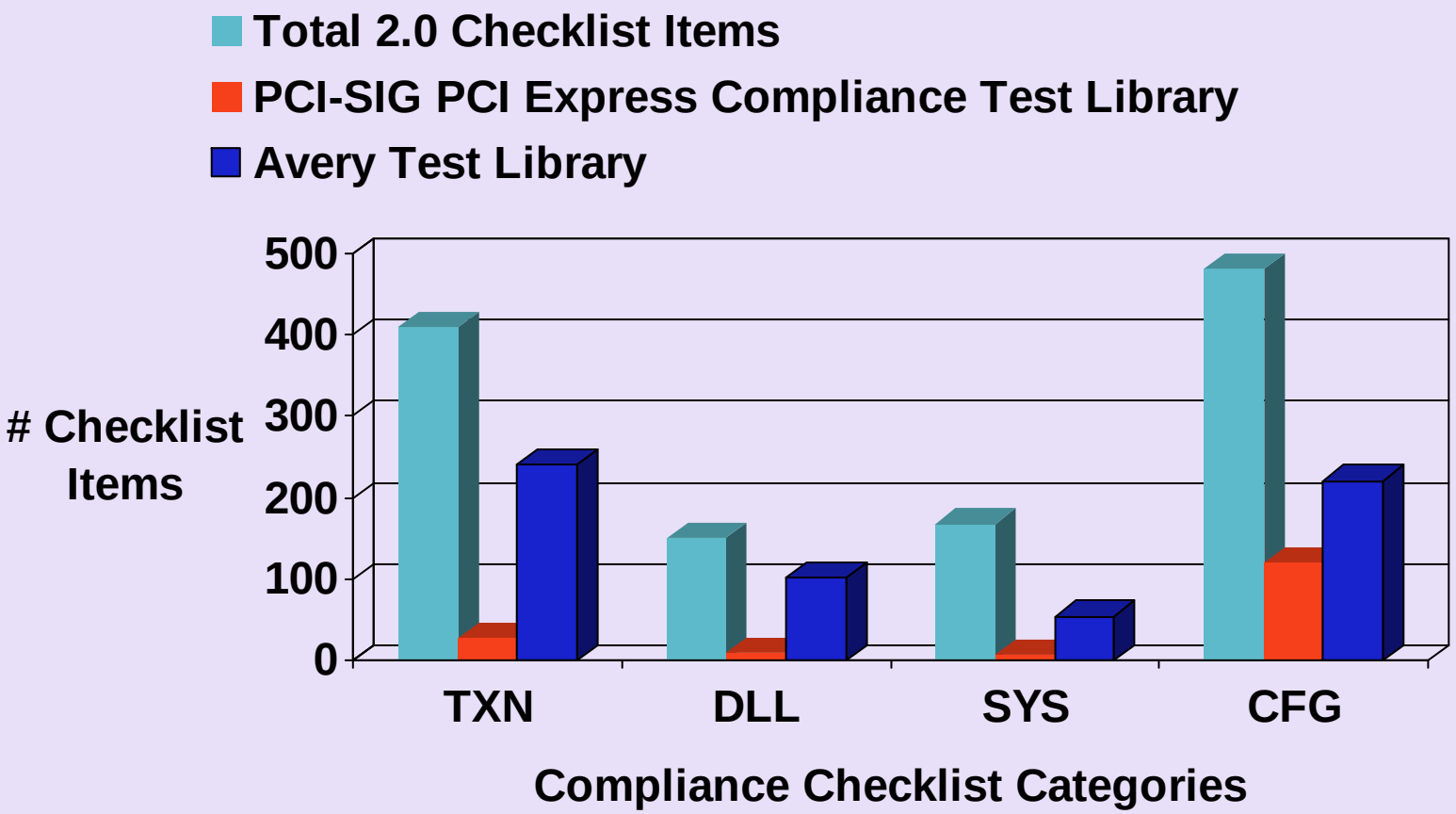
# Scope of Compliance Checklists

- Categories
  - ✓ TPL - Topology
  - ✓ TXN - Transaction
  - ✓ DLL – Data Link Layer
  - ✓ PHY – Physical Layer
  - ✓ PMG – Power Management
  - ✓ SYS – System/BIOS
  - ✓ CFG - Configuration

# Scope of Compliance Checklists

- TXN Sub-categories
  - ✓ ….total of 32
  - ✓ Packet Definitions
  - ✓ Common Packet Header
  - ✓ TLPs with Data Payload
  - ✓ Address Based Routing
  - ✓ ID Based Routing
  - ✓ First/Last Byte Enables
  - ✓ Transaction Descriptor
  - ✓ Attributes
  - ✓ ……..

# Scope of Compliance Checklists – Compliance Program



# Typical Design Bugs Encountered

- Common bugs found in multiple designs
- PCI-SIG® compliance test library does not necessarily cover these cases
- Additional compliance testsuite needed to verify design compliance to most compliance checklist assertions
- Directed test cases preferred method to ensure high compliance checklist trigger coverage

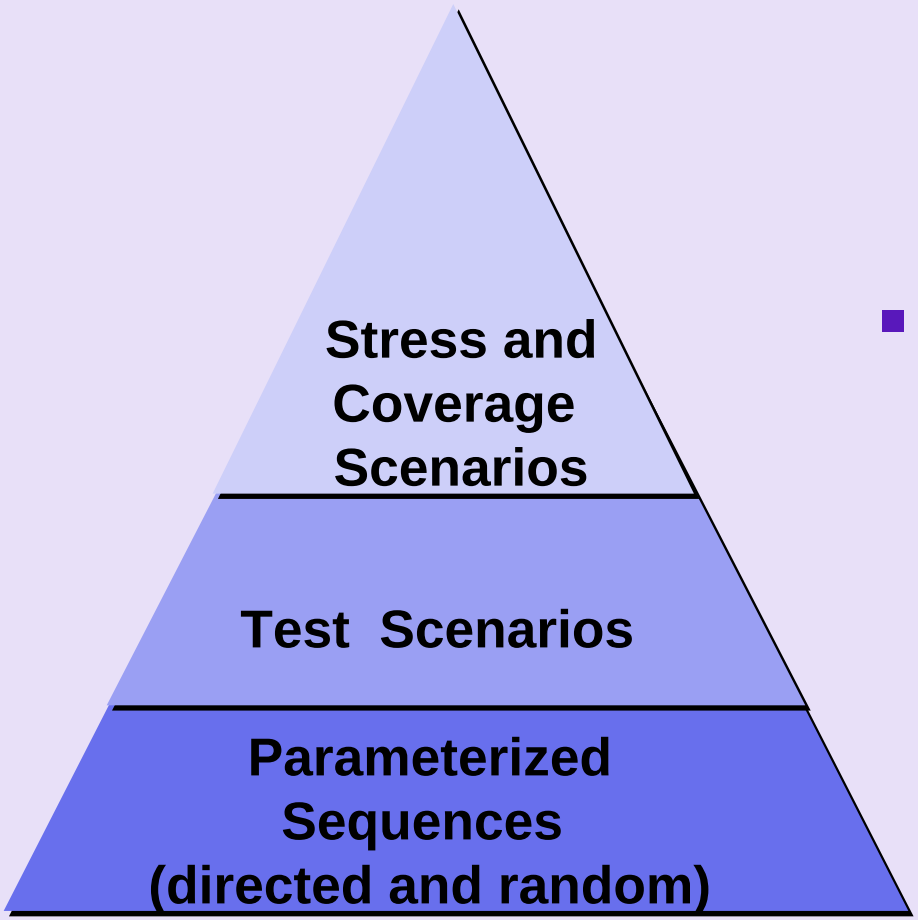
# Typical Design Bugs Encountered

- 1.1
  - ✓ Linkwidth down configuration
    - Premature idling of lanes prior to Configuration.Complete (PHY.02.19#27)
  - ✓ L1 and L2/L3 link state transitions protocol
    - Upstream component fails to detect EI (PMG.03.09#11)
  - ✓ Byte enable and byte count correctness
    - Unsuccessful completions (TXN.03.02#32, #34,#35)
  - ✓ Transaction ordering across switch
    - Read cannot pass write (TXN.04.00#06)
  - ✓ VC Negotiation
    - VC Negotiation Pending not cleared (CFG.11.08#04)

# Typical Design Bugs Encountered

- 2.0
  - ✓ Linkwidth up/down configure
    - Initiator enters Configuration.Linkwidth.Start prior to sending EIEOS on lanes it plans to activate (PHY.02.06#16)
    - Downstream lanes not initiating upconfiguration activates more lanes than initiator (PHY.02.19#58)
  - ✓ Loopback exit
    - Loopback slave does not detect EI (PHY.02.26#06, PHY.02.26#13)
  - ✓ Extended sync in Recovery.RcvrLock
    - Fails to transmit 1024 TS1 (PHY.02.20#19)
  - ✓ PCIe 2.0 configuration space default values

# Testing Framework



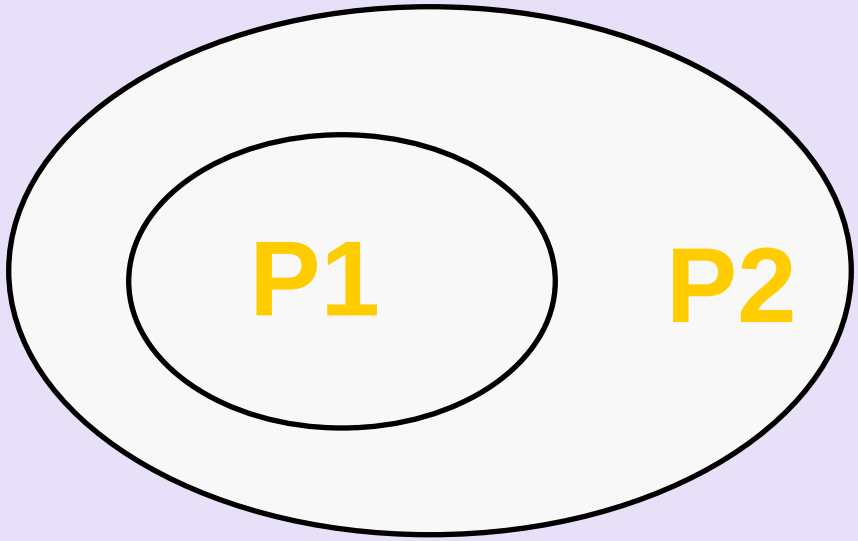

- Supports functional verification and coverage
  - ✓ Unit-level directed testing
  - ✓ Random stress testing
- Regression test and checklist coverage reports

# Testing Framework

- Focus on extensibility and reuse
  - ✓ Design for 2.0 today, SR-IOV tomorrow
  - ✓ RC, EP, SW DUT configurations
- Tests adapt to fully exercise DUT
  - ✓ # of lanes and ports
  - ✓ Support directed and random methods
  - ✓ Easily vary testing parameters
- Self checking including expected and unexpected compliance assertion violations
- Cross-mode testing (ex. 1.1 RC, 2.0 EP)

# Testing Framework

- Random parameter selection enables sequences to be rerun with many variations (Ex. P1 * P2 variations)
- Parameters include
  - ✓ Times, timeouts
  - ✓ Attributes
  - ✓ Error types
  - ✓ etc



# Compliance Coverage Report

**Assertion checker implemented in BFM**

**Assertion checker triggered during regression**

| Applies | Name | Description | BFM | Test | Dynamic Trigger | Scenario_Coverage |
|---|---|---|---|---|---|---|
| RC, EP, SW, BR | CFG.09.00#01 | Extended capabilities in a device configuration space must begin at offset 100h with a PCI Express Enhanced Capability Header. | B | | Y | ./config_1_7.vh ./config_1_8.vh |
| RC, EP, SW, BR | CFG.09.01#01 | Absence of any extended capabilities must be indicated by an Enhanced Capability Header with a Capability ID of 0000h, Capability Version of 0h, and Next Capability Offset of 0h. | B | | Y | ./avery_fcpe_sw.vh ./sys_2_2_sw.vh |
| RC, EP, SW, BR | CFG.09.03#01 | Capability ID field of a PCI Express Enhanced Capability Header must be a valid PCI-SIG defined ID number (or 0xFFFF h or 0x0000g for a terminating header). | B | | Y | ./config_1_7.vh.run ./config_1_8.vh.run |
| RC, EP, SW, BR | CFG.09.03#02 | The Capability Version field of a PCI Express Enhanced Capability Header must correctly indicate the version of the capability structure present. | B | | Y | ./avery_fcpe_sw.vh.run ./sys_2_2_sw.vh.run |

Provides key checklist coverage data to drive compliance checklist closure

**Top ranked test cases to exercise checker**

# Compliance Test Library

**Summary of the scenarios.ep scenarios**

**Scenarios in this directory can be used to verify endpoints and switch DUTs.**

1.1: PCI SIG Transaction Scenarios
1.2: Avery Transaction Scenarios - Random TLPs
1.3: Avery Transaction Scenarios - DMA Read/Write

.....

8.1: Avery Isochronous Scenarios - Contract Parameters
8.2: Avery Isochronous Scenarios - Latency
8.3: Avery Isochronous Scenarios - Transaction Ordering
8.4: Avery Isochronous Scenarios - Flow Control
8.5: Avery Isochronous Scenarios - Bandwidth Allocation
8.6: Avery Isochronous Scenarios - Endpoint Requester

9.1: PCI SIG Platform Scenarios
9.2: Avery Bios Scenarios

10.1: Avery GEN2 Scenarios - LTSSM Transitions
10.2: Avery GEN2 Scenarios - Power Management ASPM
10.3: Avery GEN2 Scenarios - Power Management L-States
10.4: Avery GEN2 Scenarios - Configuration Register Attributes

# Compliance Test Specification

```
* Test Category: 10.2, Avery GEN2 Scenarios - Power Management ASPM, (scenarios.ep)
*
* Test Identifier: 2, Speed change at ASPM L1, Specification 2.0
*
* Description: Initiates speed change when DUT is in L1 state.
*              Test Step:
*              1. Wait for enumeration done.
*              2. BFM program ASPM control field to 2'b10 in DUT.
*              3. DUT is expected to enter ASPM L1.
*              4. BFM initiate speed change.
*              5. L1 => recovery => L0(gen2).
*              6. The link must establish at gen2 speed

*
* Test Origin:  PCI SIG Compliance Test
*
* Source:  gen2_aspml1_speed_change.vh
*
* Topologies Supported:
*     RC, EP, SW
*
* Related: NA
*
* Test Segments:
*     gen2_aspml1.vh (parameters: inarg0: L1 stable timeout)
*
* Released: 1.5.0807
```

# Beyond Compliance Checklist-based Testing

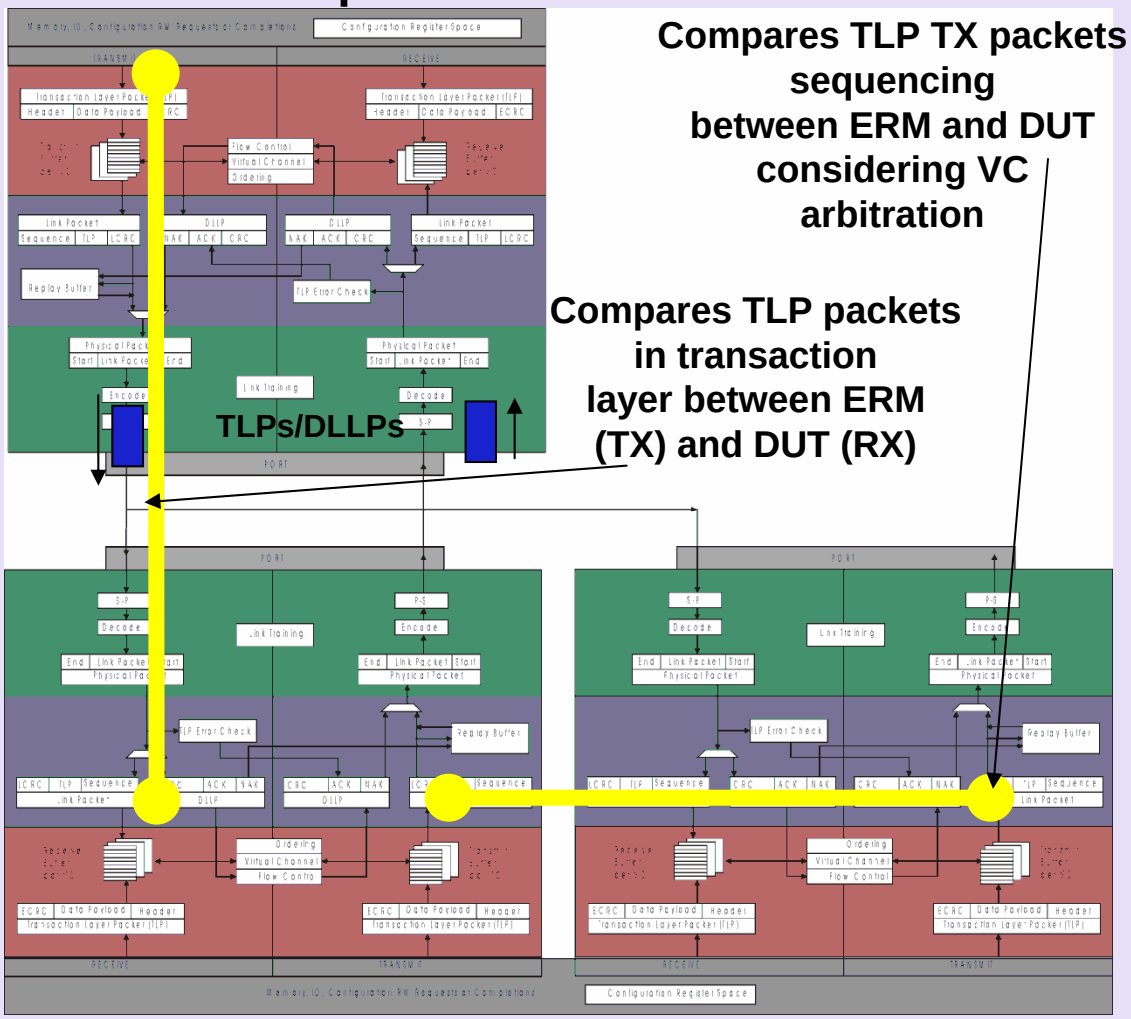
- Extend verification framework for better DUT observability and analysis (“matchpoints”)
  - ✓ Focuses verification on proving key micro-architecture features are consistent between 2 implementations
    - BFM to DUT
    - Reference design to DUT
  - ✓ Enhances both simulation-based verification and formal sequential equivalence checking (SEC)
  - ✓ BFM or Reference design can be used being superset or subset of DUT functionality
  - ✓ Supports automatic result prediction for random testing methods

# Matchpoint Checks

- Sequential consistency focuses on control
  - ✓ FSM states sequence in a similar manner
  - ✓ Protocol checkers trigger/pass/fail consistently
- Data consistency
  - ✓ Similar transaction request and completions
  - ✓ Payload and protocol-specific data fields
  - ✓ Configuration Space state

# Matchpoints

- Matchpoints
  - ✓ between DUT and BFM
  - ✓ within DUT
- Design matchpoint mismatch may indicate design bug



# Matchpoint Examples

- Transition (control)
  - ✓ Lane negotiation FSM (LTSSM)
  - ✓ Retry buffer using saved records and retry trigger
  - ✓ VC's flow credits, rxfc_allocated_m, rxfc_infinite_m
  - ✓ Data link layer control/management FSM
  - ✓ Configuration data, including various capabilities
  - ✓ Power management FSM
- Transport (data)
  - ✓ TLPs at various points, including rx and tx side
  - ✓ Error messages
  - ✓ UR packet to be returned

# Matchpoint Example Implementation

- Match the two LTSSM machines
- Match transmit TLPs
- Supported match items
  - ✓ Variable match: $tb_var_match
  - ✓ Structure match: $tb_record_match

```
initial begin
    $tb_var_match("dut.ltssm_sreg vs gold.ltssm_sreg",
                `EP_DUT.tl.dll0.lk0.ltssm_sreg,
                `EP_GOLD.tl.dll0.lk0.ltssm_sreg, 1, 2);

    $tb_rec_match("correct tx TLP compare at lane",
                 `AVY_DUT.tl.correct_tlp_reg2, `AVY_DUT.tl.correct_tlp_reg_trig,
                 `AVY_DUP_DUT.tl.correct_tlp_reg2, `AVY_DUP_DUT.tl.correct_tlp_reg_trig,1, 1);
    end
```

# Chip-level Verification Challenges

- Multiple PCIe domains
- Chip-level verification relies heavily on
  - ✓ random methods for “realistic” system behaviors
  - ✓ Emulating higher-level operations such as DMA, PCI BIOS, and monitoring interrupt and platform events
- Random BFM responses include
  - ✓ Program random completion timing, splits, ordering based on request attributes
  - ✓ Error injection at various layers and packet types
  - ✓ D-State and L-State power sequences
- Support for chip-level verification reuse methods
  - ✓ VMM, AVM, OVM

# Lessons Learned

- Company/Perspective
  - ✓ Customer and vendor alignment is very important
  - ✓ IP partners are essential to ensure robust solution based on multi-sourced testing and interoperability
  - ✓ Must support for early drafts of specifications to support early adopters
- Role of commercial VIP
  - ✓ Improves productivity and provides objective confirmation of design intent
  - ✓ Customer needs to become expert user of BFMs to get the most out of the solution. There are no push button solutions.

# Lessons Learned

- Core-level verification challenges
  - ✓ BFMs and tests must be flexible to support implementation specific design decisions
    - Spec allows user-defined algorithm
    - Spec has “room for interpretation”
- Compliance checklist and testing foundations
  - ✓ Documentation of testsuites/sequences is essential to understanding 3rd party tests (source code if possible)
  - ✓ Indexing tests to checklist items and coverage presents a more complete compliance picture
  - ✓ Debug and observability improved via verbose messaging modes and source code debug

# Lessons Learned

- Beyond compliance checklist-based testing
  - ✓ Alternative methods offer limited improvements
    - Formal model checking not suited for PCIe
    - Matchpoint verification (simulation-based or SEC) provides sequential consistency checking however requires detailed knowledge of core
- Chip-level verification challenges
  - ✓ BFMs need to support multiple domains for embedded system designs
  - ✓ Some embedded designs are not always 100% compliant and require BFM to support optional out-of-spec features

Thank you for attending the PCI-SIG Developers Conference Europe 2007.

For more information please go to www.pcisig.com