SoC Integration and Compliance Verification

Neill Mullinger
Product Manager
Synopsys

Copyright © 2013, PCI-SIG, All Rights Reserved

Disclaimer

Presentation Disclaimer: All opinions, judgments, recommendations, etc. that are presented herein are the opinions of the presenter of the material and do not necessarily reflect the opinions of the PCI-SIG®.

Copyright © 2013, PCI-SIG, All Rights Reserved

Agenda

- Verification Strategy
- RTL Verification
- Debug and coverage closure
- Q&A

Copyright © 2013, PCI-SIG, All Rights Reserved

Compliance or Integration?

- Rigorously test the entire specification
- 1000s of scenarios

Scoreboard

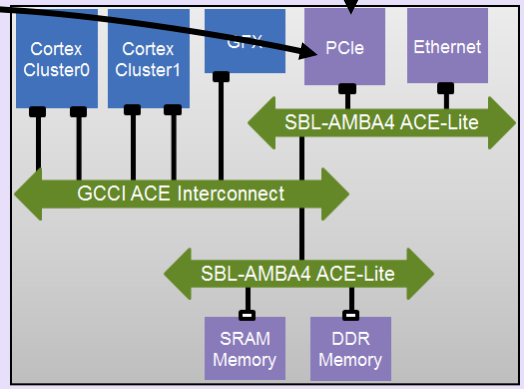
PCIe® VIP

DW PCIe Core

API VIP

- Test integration of a proven core
- Sufficient tests to verify successful integration

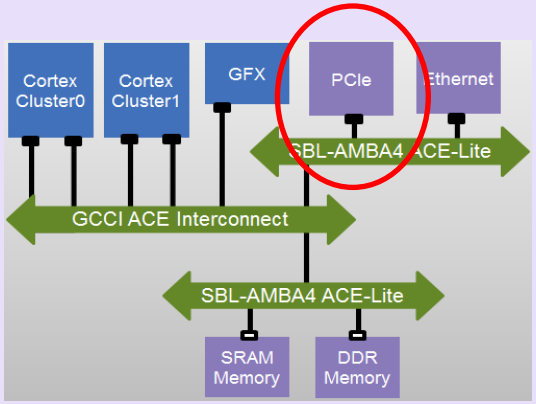
PCIe VIP *(traffic)*



Copyright © 2013, PCI-SIG, All Rights Reserved

RTL Verification Strategy

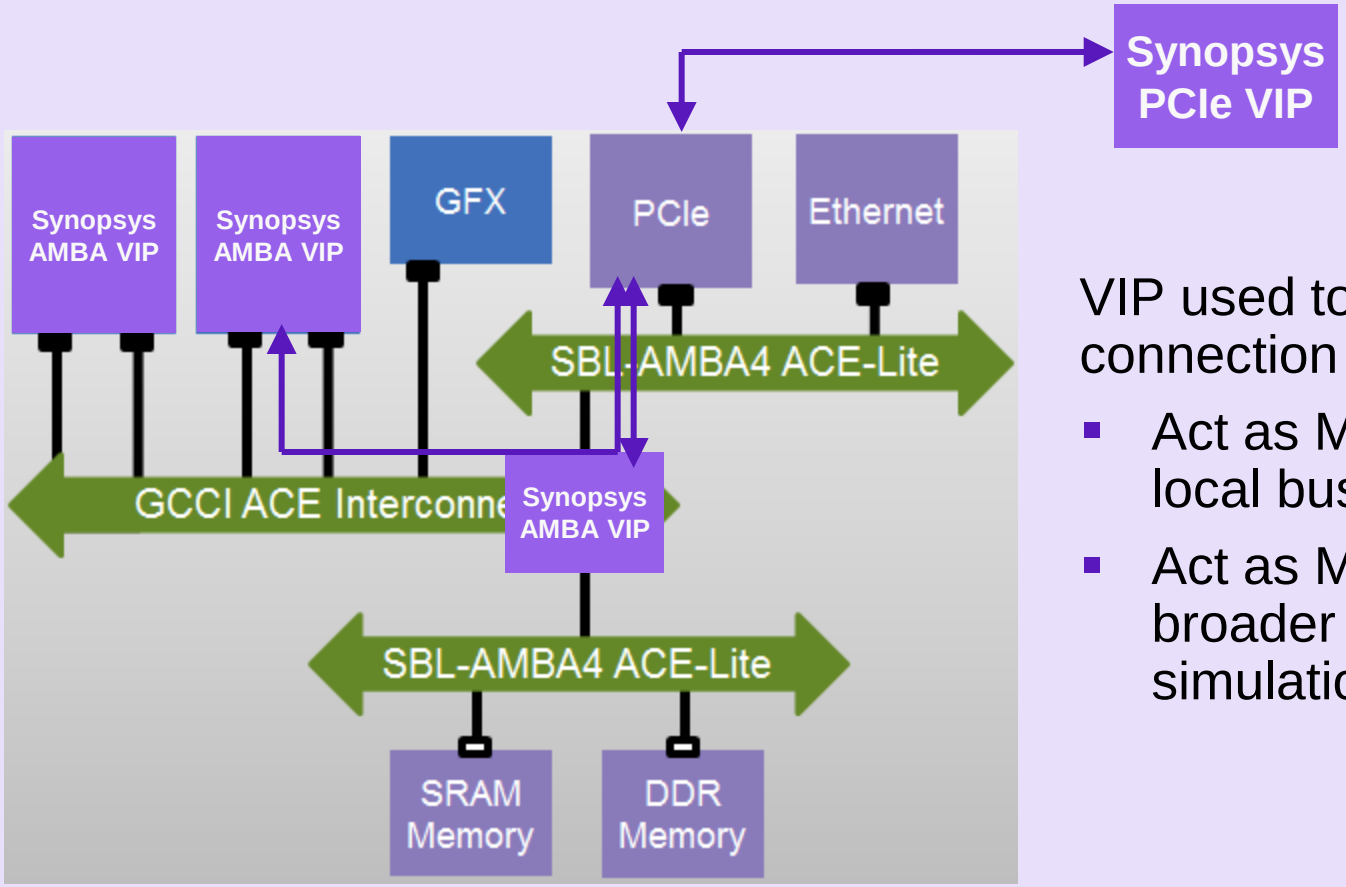
- SoC Team's Job is Integration
  - ✓ IIP from reputable vendor
  - ✓ VIP from reputable vendor
  - ✓ Compliance should be assumed
    - Vendor's regression
    - Time on market
    - Successful chips brought to market



- How to verify integration?
  - ✓ Need to be able to configure the DUT
  - ✓ Need to be able to move data through the system
  - ✓ Errors in the protocol shouldn't break the system
  - ✓ Verify low power states
  - ✓ Monitor Performance

Copyright © 2013, PCI-SIG, All Rights Reserved

Integration Testbench



VIP used to verify core connection into system

- Act as Master/Slaves on local bus
- Act as Masters/Slaves for broader system simulation

Copyright © 2013, PCI-SIG, All Rights Reserved

LTSSM / Enumeration

- Requirements
  - ✓ Confirm the system is ‘wired’ properly
  - ✓ Confirm the link can be trained
  - ✓ Configure IP BARs and match to the VIP

Strategy

- Verify Link
  - ✓ Setup # lanes, speed
  - ✓ Verify L0 is reached
  - ✓ Perform Enumeration
    - Determine memory size of the BAR using config request
    - Write BAR
    - Verify access with memory reads to complete BAR

```
virtual task uvm_test::run_phase(uvm_phase phase);
   …
   reset();
  set_link_width.link_width          = LINK_WIDTH_X4;
  set_supported_speeds.speeds  =  SPEED_2_5G |
                                  SPEED_5_0G |
                                  SPEED_8_0G;
  …
  root0_set_phy_enable_item.enable = 1;
  …
  wait(root0_phy_link_up_subscriber.link_up == 1’b1 );
  …
endtask
```


Copyright © 2013, PCI-SIG, All Rights Reserved

LTSSM / Enumeration

- Verify Support Speeds
  - ✓ Train to L0
  - ✓ Renegotiate the link for all speeds 2.5G, 5G, 8G

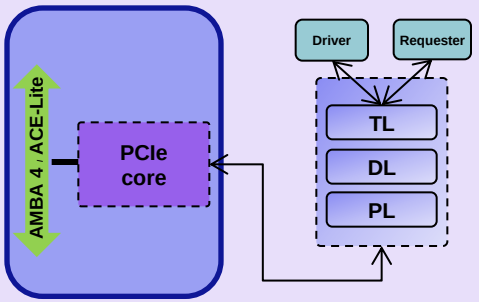
```
108:    DEBUG5:  endpoint0.port0.pl0.LTSSM: State changed from UNKNOWN to DETECT_QUIET
12112: DEBUG5:  endpoint0.port0.pl0.LTSSM: State changed from DETECT_QUIET to DETECT_ACTIVE
…
81232: DEBUG5:  endpoint0.port0.pl0.LTSSM: State changed from CONFIGURATION_COMPLETE to CONFIGURATION_IDLE
81296: DEBUG5:  endpoint0.port0.pl0.LTSSM: State changed from CONFIGURATION_IDLE to L0
81296: INFO    :   endpoint0.port0.pl0.LTSSM: Link training completed.  The link is READY! Speed is 2_5Gb/s
```

- Verify Lane Reversal
  - ✓ Rewire lanes on DUT
  - ✓ Train to L0
  - ✓ Verify lanes are properly reversed

Copyright © 2013, PCI-SIG, All Rights Reserved

General Traffic Testing

- Requirements
  - ✓ Verify sending TLPs end-to-end
  - ✓ Testing the ability to move data from point A to point B



Strategy

- Core as Requester / VIP as Completer
  - ✓ Respond to valid address ranges for the EP/RC
  - ✓ Set min/max completion size and max payload size to fully exercise the DUT
  - ✓ Have the DUT do a series of reads/writes. The goal is verify handling of multiple completions

Copyright © 2013, PCI-SIG, All Rights Reserved

General Traffic Testing

- VIP as a Requester
  - ✓ Series of config writes and read backs to random registers
    - Scoreboard / shadow memory used to verify response
  - ✓ Series of mem/io writes and reads to the random addresses in the DUT.
    - Scoreboard / shadow memory used to verify response
  - ✓ Repeat above with randomized min/max payload sizes
    - Want to verify DUT can handle all valid completion combinations
  - ✓ Use Memory in requester application to do a series of writes and read backs while other traffic is applied with the driver application.
    - Realistic traffic
    - Foreground / background
    - VIP / Scoreboard flags violations

Copyright © 2013, PCI-SIG, All Rights Reserved

Interface Testing

- Requirements
  - ✓ Verify supported interfaces in the PCIe sub-system

Strategy

- ✓ Test supported interfaces
  - SERIAL, PARALLEL, PIPE
- ✓ Lane Error Handling
  - Inject errors on lanes to force negotiation to lower number of lanes
- ✓ Additional Error Injection / Handling
  - Disparity error injection, Bit Flip, etc.
- ✓ PCIe 3.0 Equalization
  - Verify coefficients can be requested / rejected
- ✓ Power Management
  - Entry/exit, hot-reset/hot-plug, clock gating

Copyright © 2013, PCI-SIG, All Rights Reserved

Performance Testing

- Requirements
  - ✓ Verifying PCIe core meets performance objectives as specified by SoC specification.
- Strategy
  - ✓ Stress Testing
    - Create background traffic across the system not targeting the PCIe core.
    - In parallel direct input in / out of PCIe core
    - Verify desired latencies and throughput is met
  - ✓ Application Specific Testing

Copyright © 2013, PCI-SIG, All Rights Reserved

Agenda

- Verification Strategy
- RTL Verification
- Debug and coverage closure
- Q&A

Copyright © 2013, PCI-SIG, All Rights Reserved

Debug and Coverage Closure

- Integration and Debug
  - ✓ Issues will be encountered that require debug
  - ✓ Ability to see inside the model
  - ✓ Ability to track down protocol issues
  - ✓ Ability to follow traffic flow
- Integration and Coverage Closure
  - ✓ Coverage suite must be suited to integrator's requirements
  - ✓ A plan is needed to make sure goals are being met
  - ✓ Coverage is tracked to measure success of plan

Copyright © 2013, PCI-SIG, All Rights Reserved

Debug: Centralized Messaging

- Multiple levels of errors: ERROR|WARNING|NOTICE
- Controllable verbosity and settable exit threshold
- Fine grain control of error suppression for Error Injections
- uvm_log support

```
2837: DEBUG5:  root0.port0.dl0.TransmitPhy: Sent INITFC2-CPL DLLP, Start Del = 0x00, End Del = 0x00:
                 00000: 0xe0190402 <-- VC = 0, Rsvd2 = 0x0, HdrFc = 100, Rsvd1 = 0x0, DataFc = 1026
                                       00001:     0xf842 <-- CRC-16bit
2843: DEBUG4:  root0.port0.tl0.TransmitTLP: Sent packet TLP XID = 0xf0d, len = 4, ptr = 0x10100000
         0000: 0x04008001   Hdr[0]: TLP type=CfgRd0. TC=0, TH=0, TD=1, EP=0, Attr=3'b000, AT=00, Length=1 (0
                                    dwords in this TLP)
           0001: 0x000f0d0f   Hdr[1]: Req ID=0x000f, Tag=0x0d, Byte Enables:  First=4'b1111, Last=4'b0000
                  0002: 0x01000000   Hdr[2]: Bus/Device/Function=0x01/00/0, (Ext)Register=0x(0)00
                                     0003: 0x1e3888fc   Digest/ECRC
2843: DEBUG5:  root0.port0.dl0.FC_INIT_StateMachine: State change from STATE_FC_INIT2 to STATE_FC_COMPLETE for VC0.
        2843: DEBUG5:  root0.port0.dl0.FC_INIT_StateMachine: Flow Control Initialization complete for VC0.
         2843: INFO:    root0.port0.dl0.DisplayInitCredits: InitFC1 VC0 credits sent to the link partner:
                              Posted:       Header Credits = 102,   Data Credits = 1024
                              Non-Posted:   Header Credits = 101,   Data Credits = 1025
                             Completion:    Header Credits = 100,   Data Credits = 1026
```


Copyright © 2013, PCI-SIG, All Rights Reserved

Debug:Transaction Log Output

```
  Time      Name/Dir    TLP Type  R_ID/Tag  Seq_Num  TC  Adr/Reg#/MsgRt/Cpl BE  DataLen  Data_0   Data_1   Data_2   Data_3  EP    ECRC       LCRC    TX/RX
                          DLLP Type                   VC    HdrFC   DataFC  MCode   DW                                                          Error
--------- ----------- ------------------- -------  --  ------------------ - -   ----  -------- -------- -------- -------- -- ---------- ---------- -----
      81333:      root0/T          INITFC1-CPL     --   0      100      002                                                          0x823d
      81353:  endpoint0/R          INITFC1-CPL     --   0      100      002                                                          0x823d
      81413:  endpoint0/T          INITFC1-CPL     --   0      100      002                                                          0x823d
      81433:      root0/R          INITFC1-CPL     --   0      100      002                                                          0x823d
      81509:      root0/T           INITFC1-NP     --   0      101      001                                                          0xb0d7
      81529:  endpoint0/R           INITFC1-NP     --   0      101      001                                                          0xb0d7

                                                                                . . .

      82005:      root0/T      MWr32 0x0001/05   0000   0          0x00001040 f f   0002  00000000 815cfa7a     --       --    0       --    0x7af104f0
      82025:  endpoint0/R      MWr32 0x0001/05   0000   0          0x00001040 f f   0002  00000000 815cfa7a     --       --    0       --    0x7af104f0
      82101:      root0/T      MRd32 0x0001/03   0001   0          0x00001040 f f   0002     --       --       --       --    --      --    0x7be4ecd3
      82121:  endpoint0/R      MRd32 0x0001/03   0001   0          0x00001040 f f   0002     --       --       --       --    --      --    0x7be4ecd3
      82277:  endpoint0/T       CplD 0x0001/03   0000   0 ID=0x0100 Stat=SC BC=0008 0002  00000000 815cfa7a     --       --    0       --    0xb0cfdb5e
      82297:      root0/R       CplD 0x0001/03   0000   0 ID=0x0100 Stat=SC BC=0008 0002  00000000 815cfa7a     --       --    0       --    0xb0cfdb5e
      82357:  endpoint0/T                ACK     0001                                                                                    0x1279

                                                                                . . .
```

Completion Received

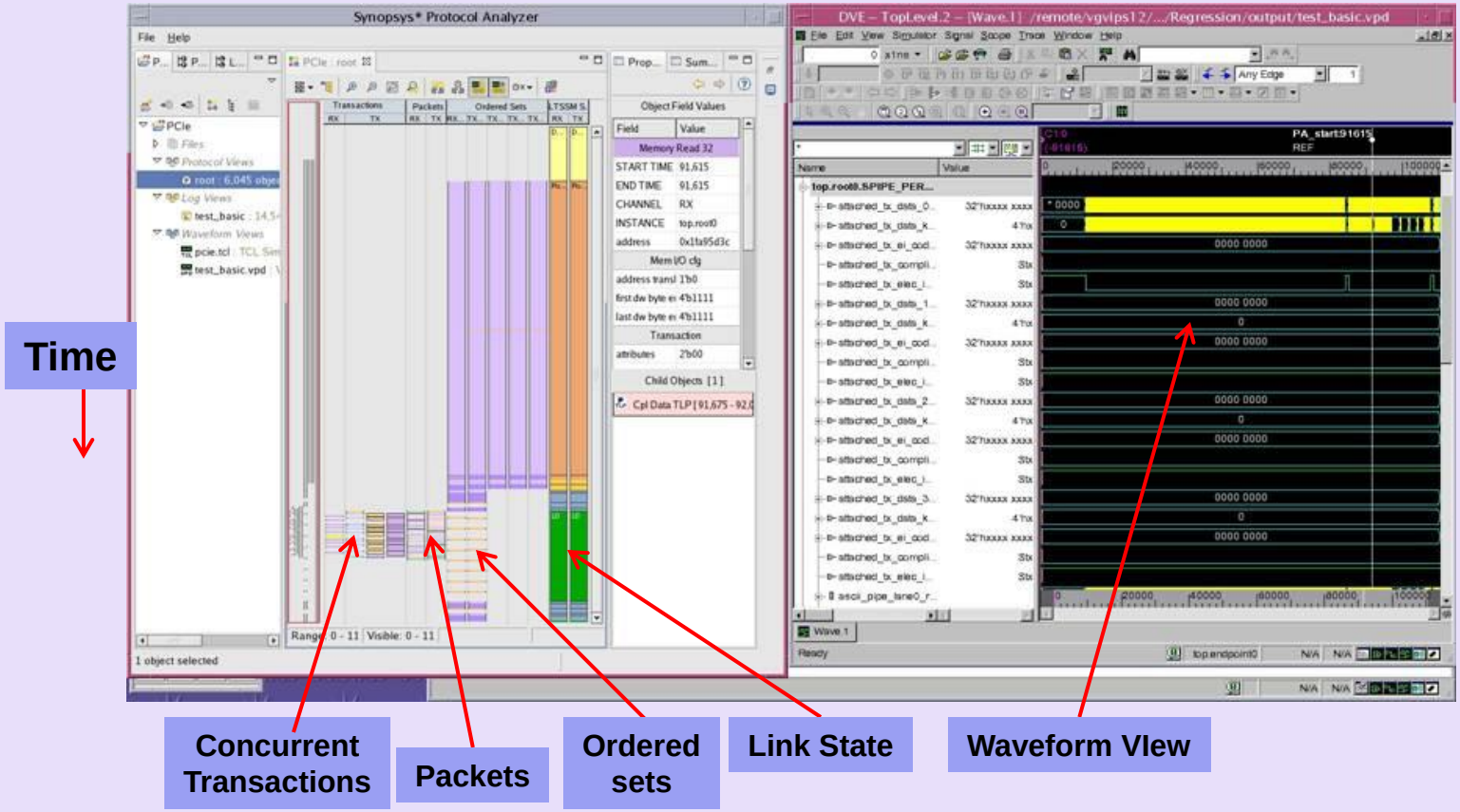
RC initiates MRd32

- Note: Log shown is for with VIP as both RC and as EP. Customer log will typically show one or the other.
- T = Transmitter, R = Receiver

Copyright © 2013, PCI-SIG, All Rights Reserved

Debug: Protocol Aware Debug

- Simplified viewing of protocol activity
  - ✓ View all layers - parent, child, and sibling
- Immediate error identification
  - ✓ Spotlights errors on protocol-centric view
- Unified debug
  - ✓ Synchronized to log files with back annotation of errors into protocol view
  - ✓ Links to Waveform View



Copyright © 2013, PCI-SIG, All Rights Reserved

Coverage Models

- Coverage Goals
  - ✓ Coverage must be specified as per the PCIe subsystem.
  - ✓ Default VIP coverage models may need to be altered or selectivly enabled to match DUT feature set
  - ✓ Must look at TX and RX independently
- Coverage Tracked at all Layers
  - ✓ Transaction Layer
  - ✓ Data Link Layer
  - ✓ Physical Layer
  - ✓ PIPE

Copyright © 2013, PCI-SIG, All Rights Reserved

Transaction Layer Coverage

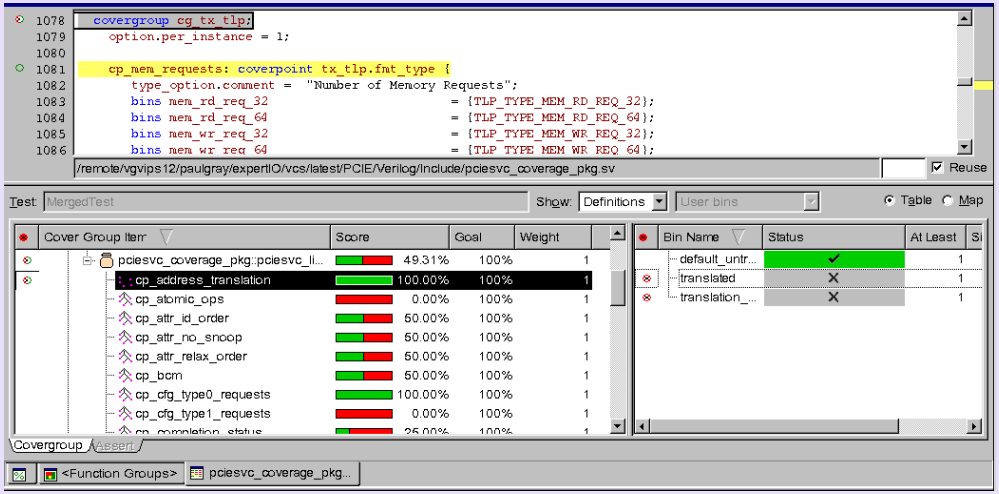
- Goal
  - ✓ Report on combinations of traffic class, TC, and virtual channels, VC
  - ✓ Enable as per DUT specification
- Example
  - ✓ Track all TCs
  - ✓ Track all VCs
  - ✓ Cross the combinations of above

Copyright © 2013, PCI-SIG, All Rights Reserved

Transaction Layer Coverage

Example for TLPs

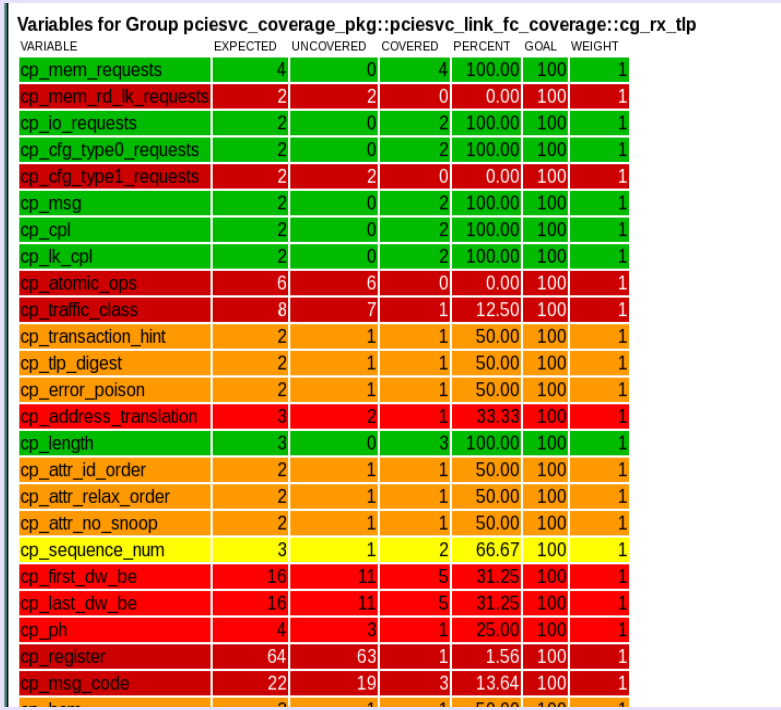
- Track all transaction types: Read, Write, Msg, etc
- Header values
  - ✓ First/last dw byte enable
  - ✓ TC
  - ✓ TLP Digest
  - ✓ EP
  - ✓ Length
  - ✓ Etc
- Error Injections
- Completion Status



Copyright © 2013, PCI-SIG, All Rights Reserved

Link Layer Coverage

- Goal
  - ✓ Track DLLPs
    - Track all DLLP types and look at valid combinations of function codes, FC, and TC
  - ✓ Track TLPs
- DLLP Example
  - ACK/NAKs
  - Sequence #
  - Power Management
  - Flow Control VC0-VC7
  - Cross of FC and DLLP type VC0-VC7
  - Injected errors



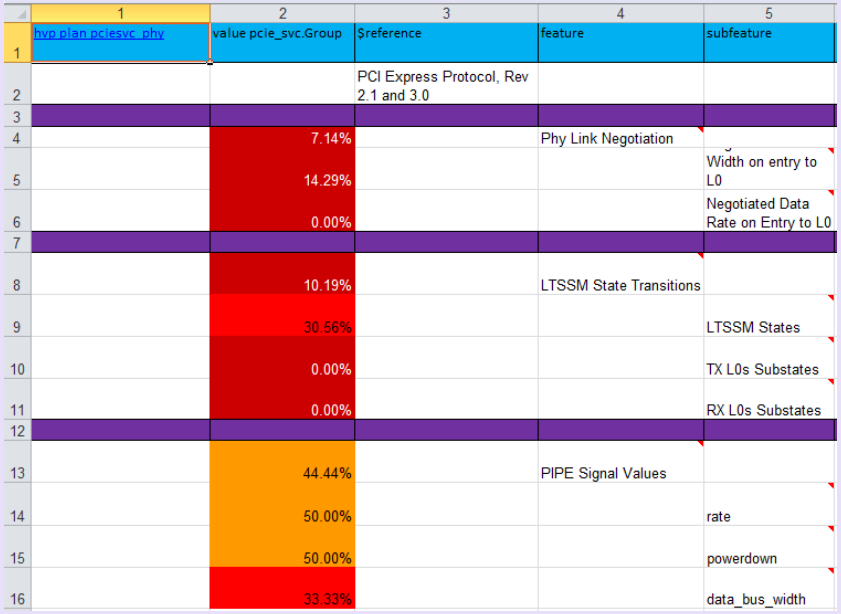
Variables for Group pciesvc_coverage_pkg::pciesvc_link_fc_coverage::cg_rx_tlp

VARIABLE	EXPECTED	UNCOVERED	COVERED	PERCENT	GOAL	WEIGHT
cp_mem_requests	4	0	4	100.00	100	1
cp_mem_rd_lk_requests	2	2	0	0.00	100	1
cp_io_requests	2	0	2	100.00	100	1
cp_cfg_type0_requests	2	0	2	100.00	100	1
cp_cfg_type1_requests	2	2	0	0.00	100	1
cp_msg	2	0	2	100.00	100	1
cp_cpl	2	0	2	100.00	100	1
cp_lk_cpl	2	0	2	100.00	100	1
cp_atomic_ops	6	6	0	0.00	100	1
cp_traffic_class	8	7	1	12.50	100	1
cp_transaction_hint	2	1	1	50.00	100	1
cp_tlp_digest	2	1	1	50.00	100	1
cp_error_poison	2	1	1	50.00	100	1
cp_address_translation	3	2	1	33.33	100	1
cp_length	3	0	3	100.00	100	1
cp_attr_id_order	2	1	1	50.00	100	1
cp_attr_relax_order	2	1	1	50.00	100	1
cp_attr_no_snoop	2	1	1	50.00	100	1
cp_sequence_num	3	1	2	66.67	100	1
cp_first_dw_be	16	11	5	31.25	100	1
cp_last_dw_be	16	11	5	31.25	100	1
cp_ph	4	3	1	25.00	100	1
cp_register	64	63	1	1.56	100	1
cp_msg_code	22	19	3	13.64	100	1

Copyright © 2013, PCI-SIG, All Rights Reserved

Phy Layer Coverage

- Goal
  - ✓ Verify LTSSM successfully transitions to all states
  - ✓ Verify LTSSM successfully negotiates to supported configurations
  - ✓ Verify re-negotiations
- Example for Phy
  - ✓ Negotiated data rate
  - ✓ Negotiated link width
  - ✓ Valid LTSSM transitions
  - ✓ L0 substate transitions



	1	2	3	4	5
1	hvp.plan.pciesvc_phy	value pcie_svc.Group	$reference	feature	subfeature
2			PCI Express Protocol, Rev 2.1 and 3.0		
3					
4		7.14%		Phy Link Negotiation	
5		14.29%			Width on entry to L0
6		0.00%			Negotiated Data Rate on Entry to L0
7					
8		10.19%		LTSSM State Transitions	
9		30.56%			LTSSM States
10		0.00%			TX L0s Substates
11		0.00%			RX L0s Substates
12					
13		44.44%		PIPE Signal Values	
14		50.00%			rate
15		50.00%			powerdown
16		33.33%			data_bus_width

Copyright © 2013, PCI-SIG, All Rights Reserved

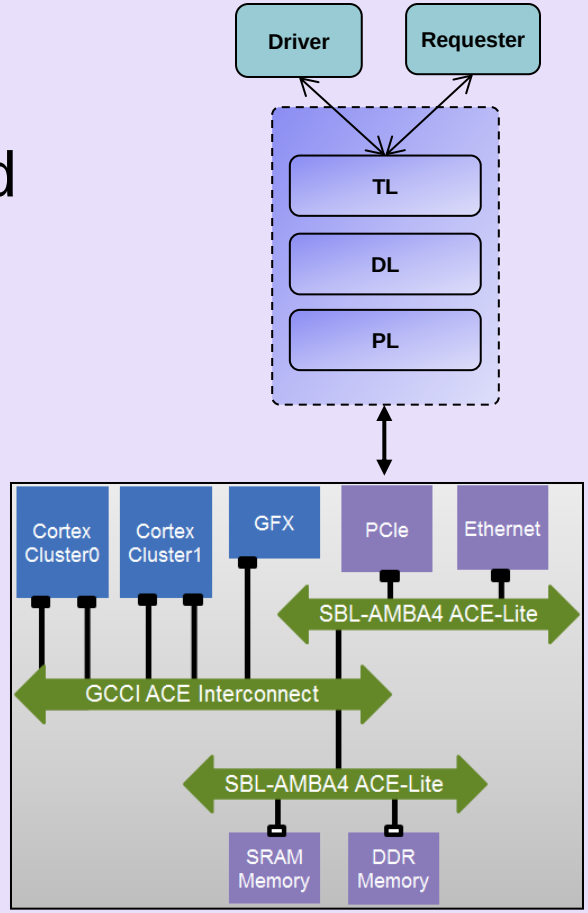
PIPE Interface Coverage

- Goal
  - ✓ Verify supported rates and data bus widths
  - ✓ Verify supported power down states
- Example for PIPE
  - ✓ Rate
    - 2.5G, 5G, 8G
  - ✓ Powerdown P0, P0s, P1, P2
  - ✓ Data bus width
    - Fixed width
    - Fixed clock
      - x8, x16, x32

Copyright © 2013, PCI-SIG, All Rights Reserved

How can VIP help?

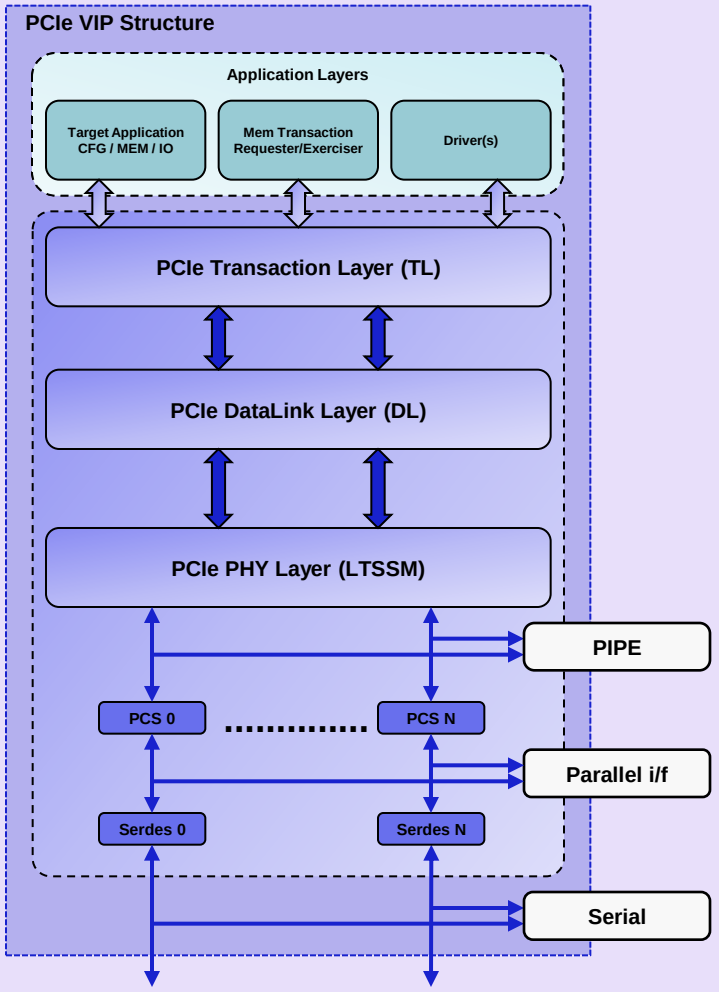
- VIP should have architecture and features for rapid integration test
  - ✓ High performance
  - ✓ Methodology support
  - ✓ Productivity features



Copyright © 2013, PCI-SIG, All Rights Reserved

PCIe VIP Integration Features

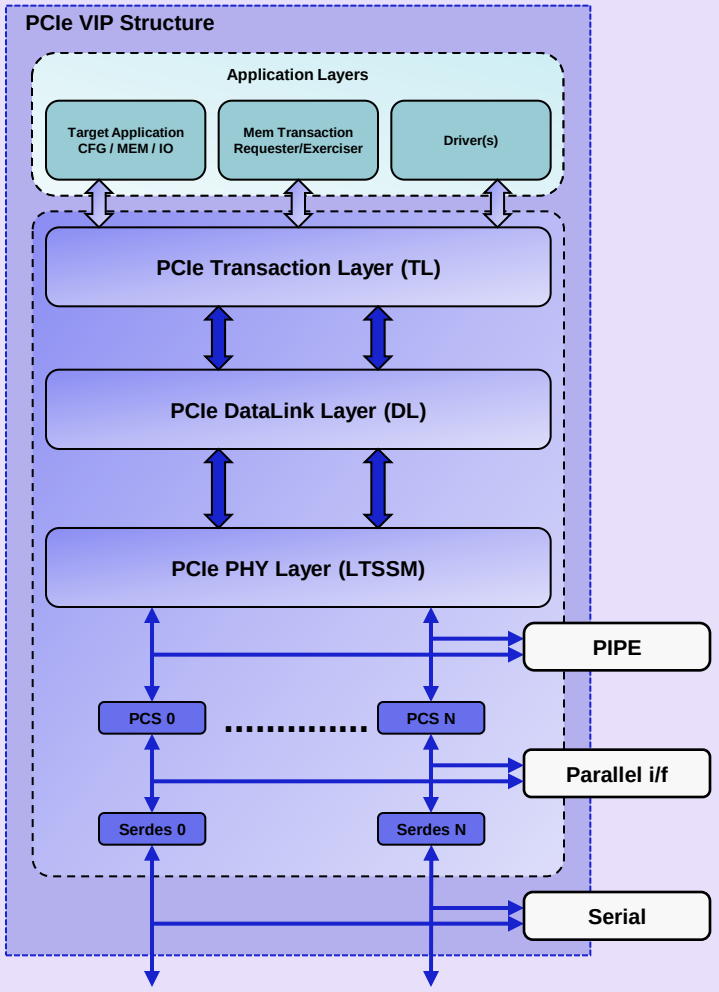
- Easy Configuration
  - ✓ Pre-configured instances
  - ✓ Simple to initiate transactions



Copyright © 2013, PCI-SIG, All Rights Reserved

PCIe VIP Integration Features

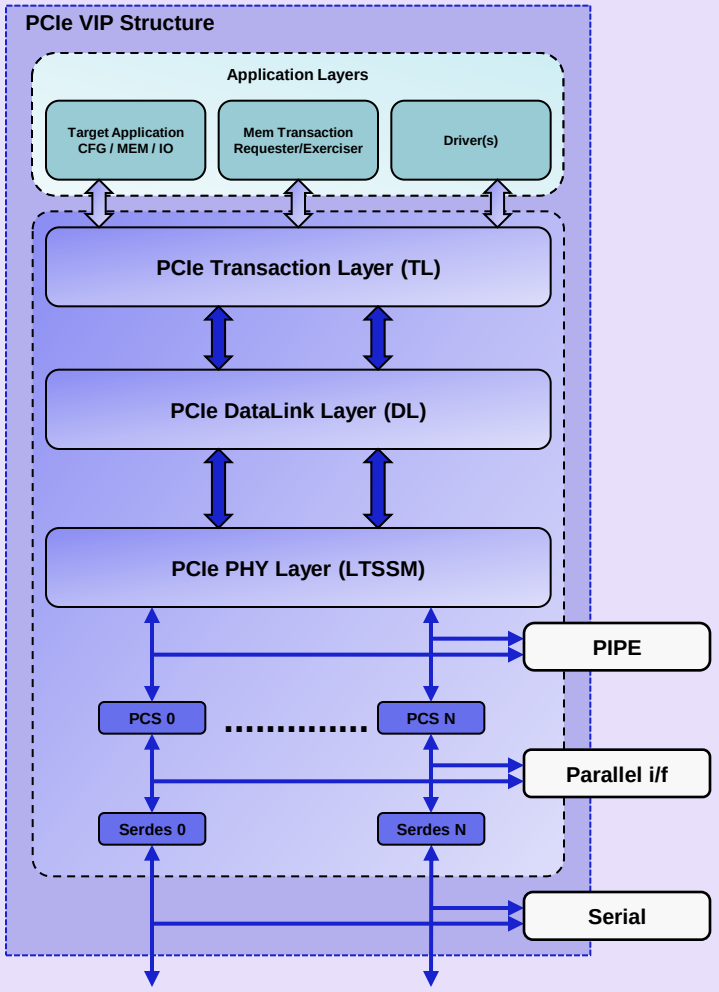
- Ease of Configuration
- Software Applications
  - ✓ Higher level of abstraction
  - ✓ Automated checking and score boarding
  - ✓ Driver
    - Standard PCIe Bus Transactions
  - ✓ Requester
    - Background traffic to various memory ranges
  - ✓ Completer
    - Automatically handles completions to mem/cfg/io writes & read



Copyright © 2013, PCI-SIG, All Rights Reserved

PCIe VIP Integration Features

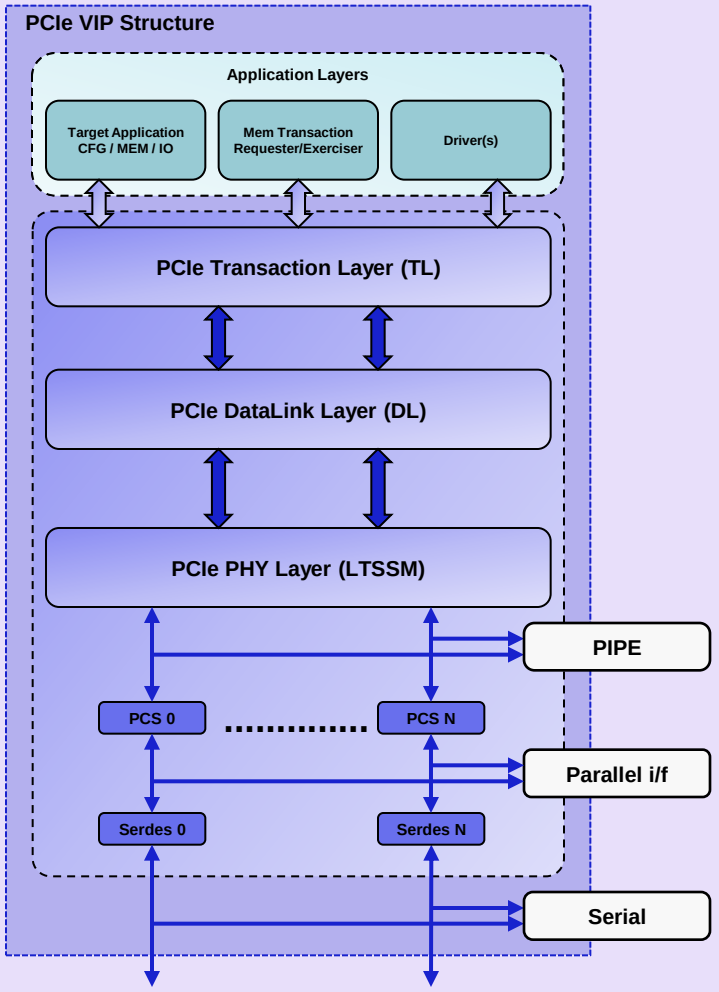
- Ease of Configuration
- Software Applications
- Built in error-injection
  - ✓ Automated injection, checking, and recovery
  - ✓ Multiple methods
    - Per transaction / sequence_item
    - Global constrained randomization of error injections
    - Above may be combined



Copyright © 2013, PCI-SIG, All Rights Reserved

PCIe VIP Integration Features

- Ease of Configuration
- Software Applications
- Built in error-injection
- Debug
  - ✓ Grey box
    - View internal signals
    - Values of internal states
  - ✓ Multiple log options
  - ✓ Protocol Analyzer



Copyright © 2013, PCI-SIG, All Rights Reserved

IP Integration Verification Strategy

SoC Team's Job is Integration

- ✓ Configure the core
- ✓ Verify connectivity
- ✓ Move relevant data through the system
- ✓ Verify error handling and correction
- ✓ Verify low power states
- ✓ Monitor Performance

Copyright © 2013, PCI-SIG, All Rights Reserved

Agenda

- Verification Strategy
- RTL Verification
- Debug and coverage closure
- Q&A

Copyright © 2013, PCI-SIG, All Rights Reserved

Q&A

Copyright © 2013, PCI-SIG, All Rights Reserved

Thank you for attending the PCI-SIG Developers Conference

Israel 2013

For more information please go to www.pcisig.com

Copyright © 2013, PCI-SIG, All Rights Reserved