



# **Looking Beyond the Compliance Checklist: Functionally Verifying PCI Express® Compliance and SOC Functionality**

**Erez Kovshi  
Cadence Design Systems**



# Agenda

- Compliance Checklist Review
- Checklist Implementation
  - ✓ Basic vs. Advanced approach
- Verification needs beyond the checklist
- Summary

# PCI Express Compliance Checklist

- What the Checklist IS
  - ✓ Full and detailed issue list (~1,300 items)
  - ✓ Standardized set of checks
  - ✓ Based on the specification
- What the Checklist is NOT
  - ✓ Not a verification implementation
  - ✓ Does not provide completeness criteria for each check
  - ✓ Does not provide a reporting mechanism

***You'll need help!***



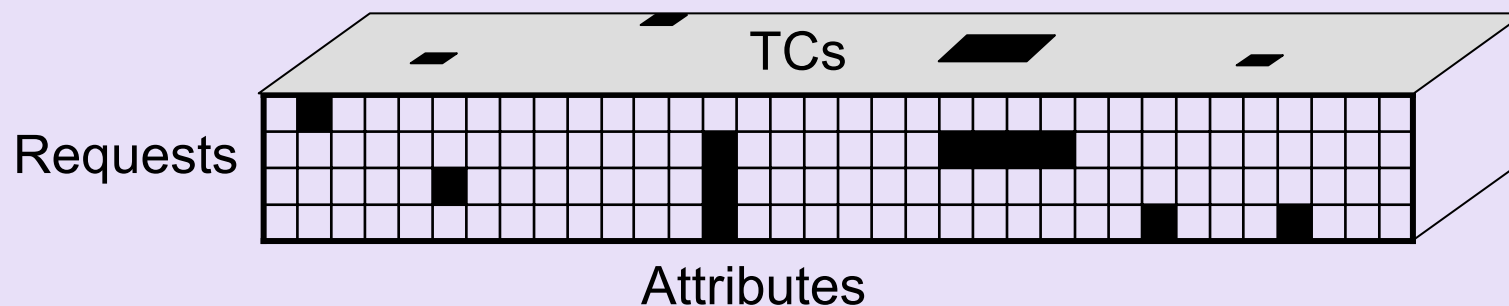
# Checklist Shortcomings Example

**TXN.2.21#19 - “Completions headers must supply the same values for the Requester ID, Tag, Attribute and Traffic Class as were supplied in the header of the corresponding request”**

- The check does NOT:
  - ✓ Implement completion handling logic that the check depends on
    - You’ll need to implement a TXN.2.21#19 check
      - “Check all completions fields with the corresponding requests”
  - ✓ Define which completions need to be check compliant
    - You’ll need to define the TXN.2.21#19 completeness criteria
      - “Check should be done for ALL the requests kind”
  - ✓ Report which completions were and which were not checked
    - You’ll need to implement status measurement and reporting

# Completeness Example

- TXN.2.21#19 - “Completions headers must supply the same values for the Requester ID, Tag, Attribute and Traffic Class as were supplied in the header of the corresponding request”
- 52 valid combinations out of 256 possibilities



# Implementing Checks Using Basic Approach

- Based on directed testing
  - ✓ Write a test for each check
  - ✓ Run all tests with no failures

## Example:

**TXN.2.21#19 - “Completions headers must supply the same values for the Requester ID, Tag, Attribute and Traffic Class as were supplied in the header of the corresponding request”**

- ✓ Test\_2.21#19:
  - “Inject CFG READ request”
- ✓ Check:
  - “Completion fields are the same as the CFG READ request”
- ✓ Run Test\_2.21#19
  - If it passes, the device is “2.21#19 compliant”. *Is it really??*

# Basic Approach Summary

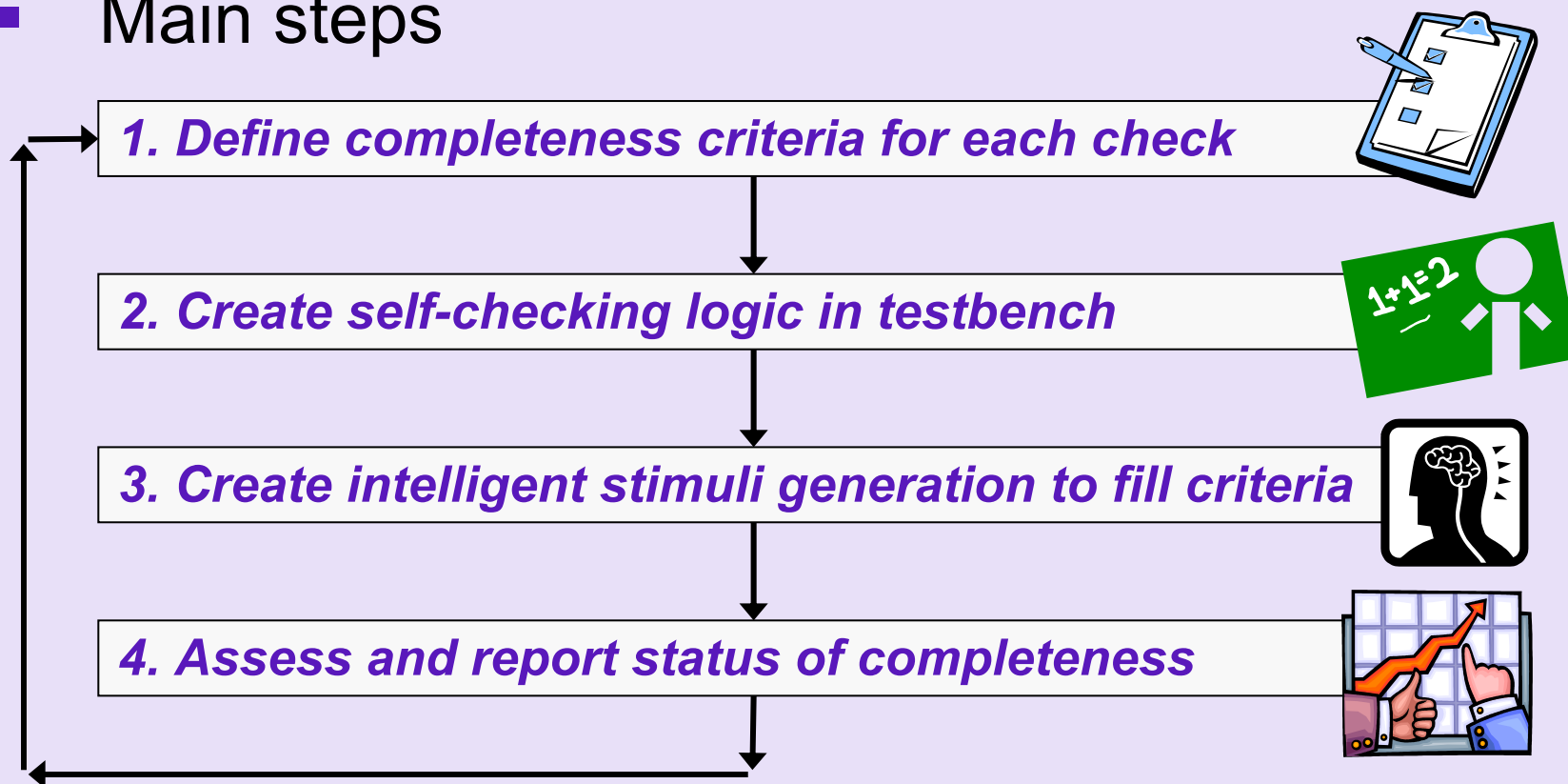
- Pros
  - ✓ “Easy” to write
  - ✓ “Easy” to get running results
- Cons
  - ✓ Maintenance is a major challenge
    - Many (hundreds!) of tests to be written and maintained
    - Many IP configurations—many verification environments
  - ✓ Each test covers only a single scenario of a check
    - Test\_2.21#19 test only completion for a CFG READ request
    - What about CFG WRITE request?
  - ✓ Tests do not find new corner cases
  - ✓ Status reporting is not objective enough
    - “233 of 949 tests completed” – what does that really mean??
    - No way to determine which functionality contains greatest risk



***This approach is error prone and  
manual labor intensive***

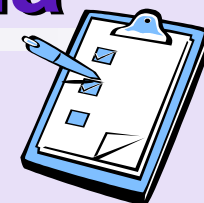
# Checks Implementation The Advanced Approach

- Based on advanced verification approach
  - ✓ Typically applied to complex devices/systems
- Main steps





# 1. Define Completeness Criteria

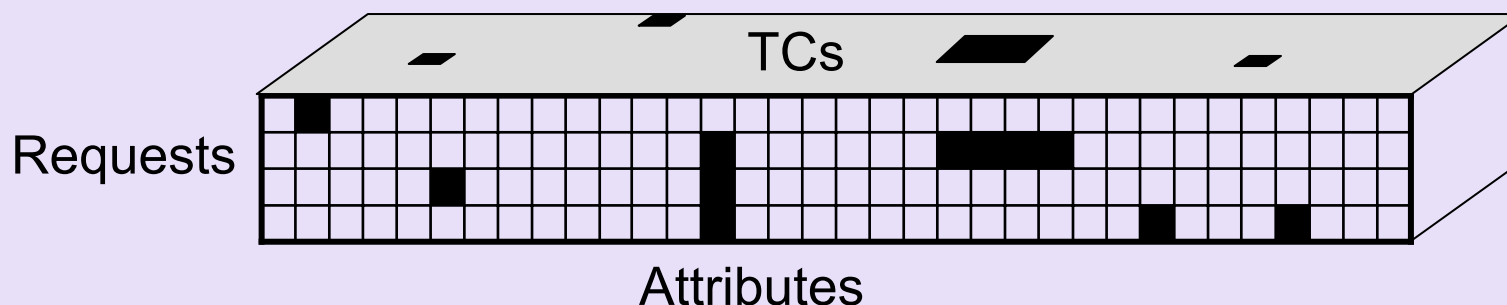


- For each compliance check X
  - ✓ Read the check definition
  - ✓ Define the possible scenarios
  - ✓ Define when your device is “X compliant”

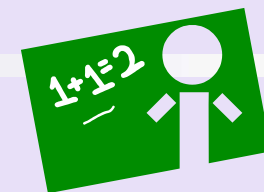
Example:

**TXN.2.21#19** - “Completions headers must supply the same values for the Requester ID, Tag, Attribute and Traffic Class as were supplied in the header of the corresponding request”

- ✓ Identify ALL possible and legal scenarios
  - Completions for the different types of requests
  - Completions for the attributes
  - Completions for TCs



## 2. Create Self-Checking Logic in Testbench



- For each compliance check
  - ✓ Implement an automatic check for all possible scenarios
  - ✓ Check should be as generic as possible
  - ✓ Check should be tested well to eliminate false alarms

### Example:

**TXN.2.21#19 - “Completions headers must supply the same values for the Requester ID, Tag, Attribute and Traffic Class as were supplied in the header of the corresponding request”**

For each completion:

```
check that completion.id == request.id  
    else dut_error();
```

# 3. Create Intelligent Stimuli Generation To Fill Criteria



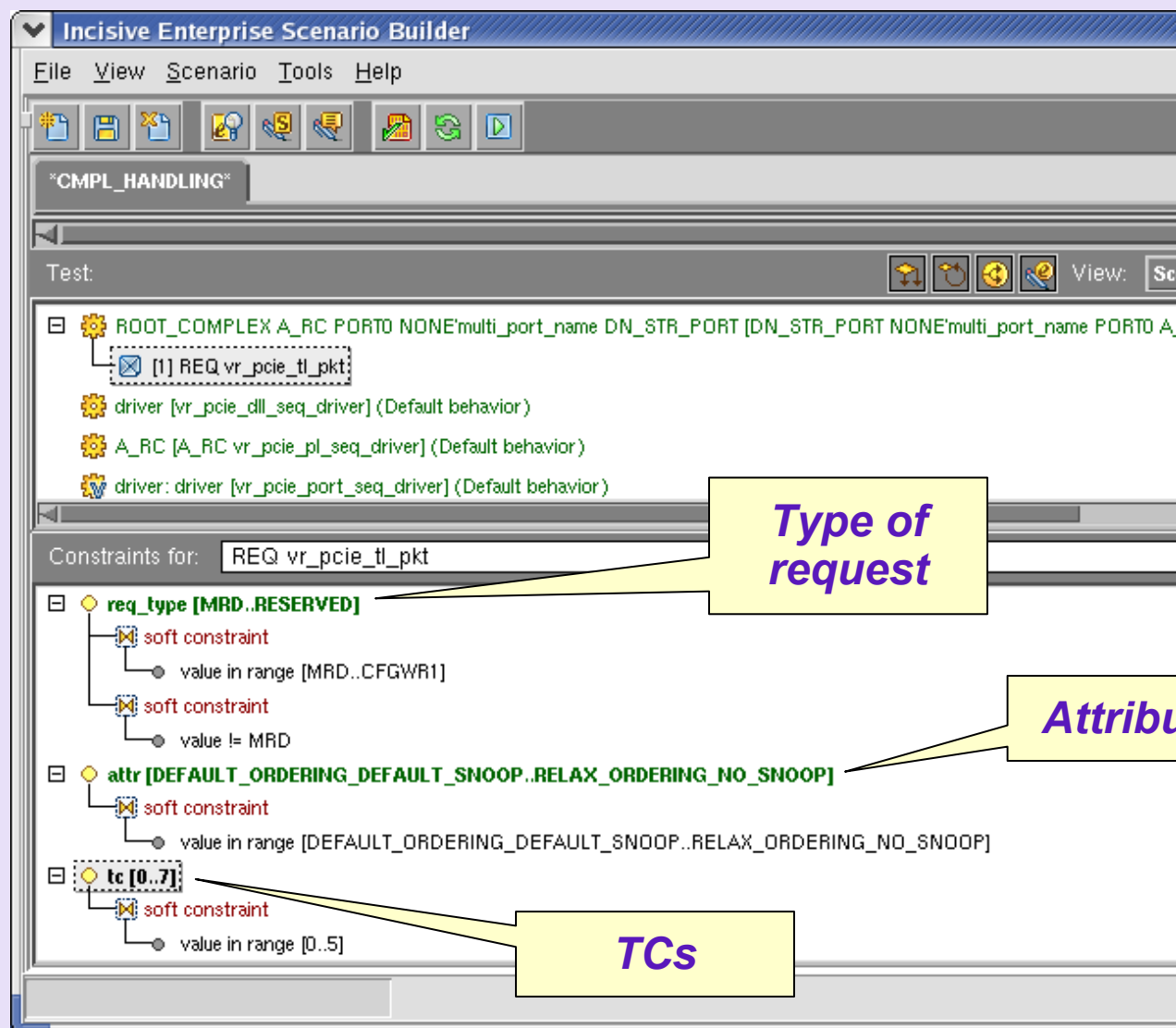
- For any given functionality and its corresponding checks:
  - ✓ Environment must create massive stimuli (randomly)
  - ✓ Environment must also generate directed stimuli (precisely)

## Example:

**TXN.2.21#19 - “Completions headers must supply the same values for the Requester ID, Tag, Attribute and Traffic Class as were supplied in the header of the corresponding request”**

- ✓ Generate random requests to reach high percentage of coverage
  - Random types of requests
  - Random Attributes and TCs
- ✓ Control generation to address non-generated scenarios

# Constrained Random Generation



Screen Capture From  
  
 PCI Express VIP

# 4. Assess and Report on Completeness



- Run random tests
- Collect the errors
- Assess for each check
  - ✓ Were all the possible scenarios exercised?
  - ✓ What are the remaining scenarios?

## Example:

**TXN.2.21#19 - “Completions headers must supply the same values for the Requester ID, Tag, Attribute and Traffic Class as were supplied in the header of the corresponding request”**

- ✓ Which requests were NOT sent to your device?
- ✓ Where are the failures, and how can you reproduce the failed run?

# Reporting Completeness Status Example

**Verification Plan Tree (pcie\_def)**

File Edit View Analysis Options

Export Info Views Runs Report

Read Reload Perspective Rank Correlate Source

**vPlan** Goal Relative Grade vPlan Persp

- 1.2.2.1.14 TXN.2.2.1#18 (NA)
- 56% 1.2.2.1.15 TXN.2.2.1#19
- 56% CMPL fields combination
- 1.2.2.1.16 TXN.2.2.1#20 (NA)
- 40% 1.2.2.1.17 TXN.2.2.1#21
- 100% 1.2.2.1.18 TXN.2.2.1#22
- 59% 1.2.2.2 Handling of Received TLPs

**Bucket Analysis [2]**

File Edit View Analysis Options

Prev Next Export Buckets Holes Project

Buckets of Item: vr\_pcie\_tl\_monitor.tx\_tl\_cmpl\_pkt\_ended(per\_type).cross\_req\_

At least : 1

Buckets Table

req_type	attr	tc	Count	Tests
CFGRD0	DEFAULT_ORDERING_DEFAULT_SNOOP	0	145126	1
MRDLK	DEFAULT_ORDERING_DEFAULT_SNOOP	0	66262	1
MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	5	62400	1
MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	3	62402	1
MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	6	61759	1
MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	7	61611	1
MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	4	61335	1

**Functionality Checking IS Complete**

0.20 cross\_req\_type\_attr\_tc

516 Hits from 4 tests

Grade	req_type	attr	tc	Tests	Hits	Goal	Hits / Goal
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	0	3	16	1	50
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	1	3	44	1	100
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	2	3	28	1	
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	3	2	38	1	
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	4	3	47	1	
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	5	4	46	1	
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	6	3	43	1	
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	7	3	25	1	
0	MRD	DEFAULT_ORDERING_NO_SNOOP	-	0	0	1	
0	MRD	RELAX_ORDERING_DEFAULT_SNOOP	-	0	0	1	
0	MRD	RELAX_ORDERING_NO_SNOOP	-	0	0	1	
0	MRDLK	-	-	0	0	1	
1.00	MRD	DEFAULT_ORDERING_DEFAULT_SNOOP	0	4	64	1	
1.00	IOWR	DEFAULT_ORDERING_DEFAULT_SNOOP	0	3	50	1	
1.00	CFGRD0	DEFAULT_ORDERING_DEFAULT_SNOOP	0	3	25	1	
1.00	CFGRD1	DEFAULT_ORDERING_DEFAULT_SNOOP	0	4	28	1	

**Functionality Checking is NOT Complete**

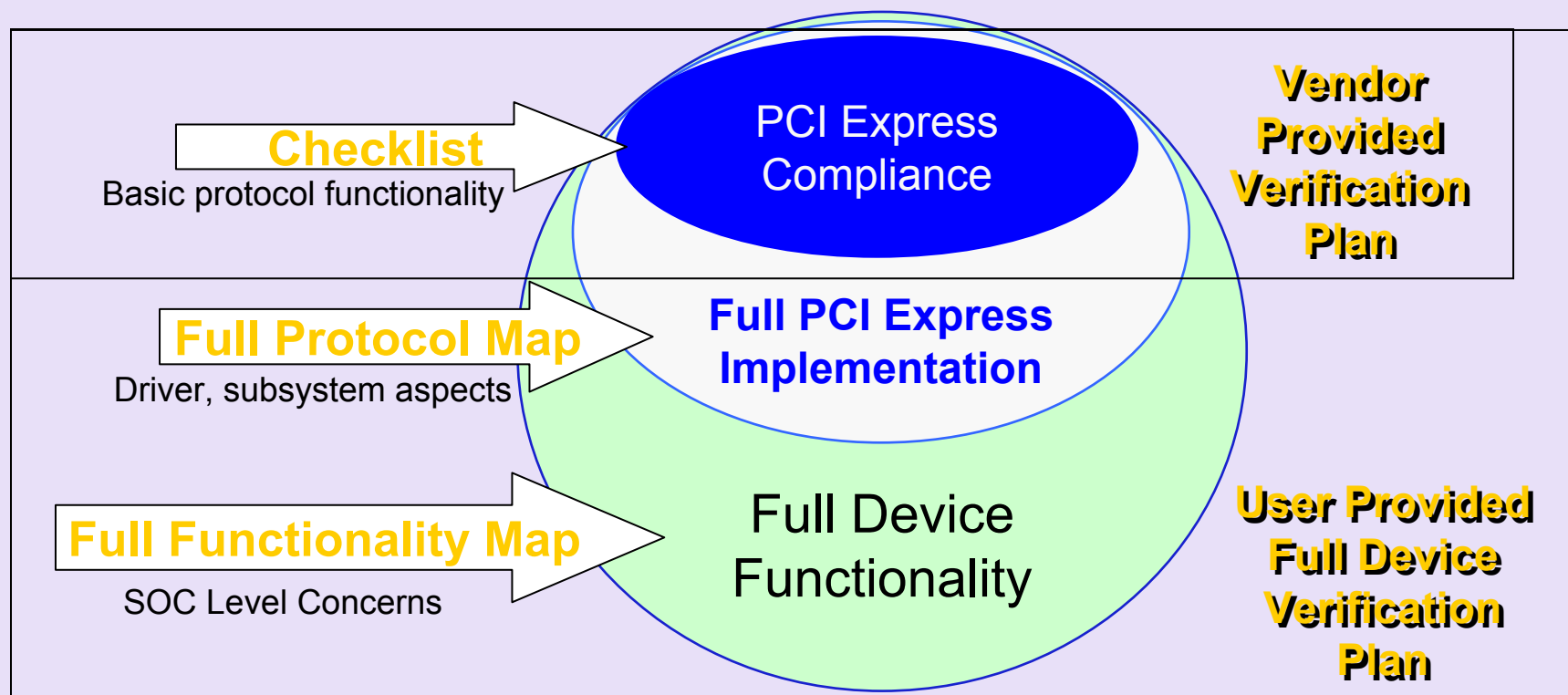
Screen Captures From  
**cadence**  
PCI Express VIP

# Advanced Solution Summary

- The challenge
  - ✓ Implement advanced verification approach to exercise checks
  - ✓ Need to develop and maintain many (hundreds!) of checks
  - ✓ Many complicated scenarios to verify
- Pros
  - ✓ Provides clear goals for each check
  - ✓ Needs only a few random tests each covering many scenarios
  - ✓ Provides automatic means to measure and report progress
- Cons
  - ✓ Requires up front verification planning
  - ✓ Requires advanced verification expertise

***Consider Using Commercial PCI Express Solution***

# Verification Beyond the Checklist



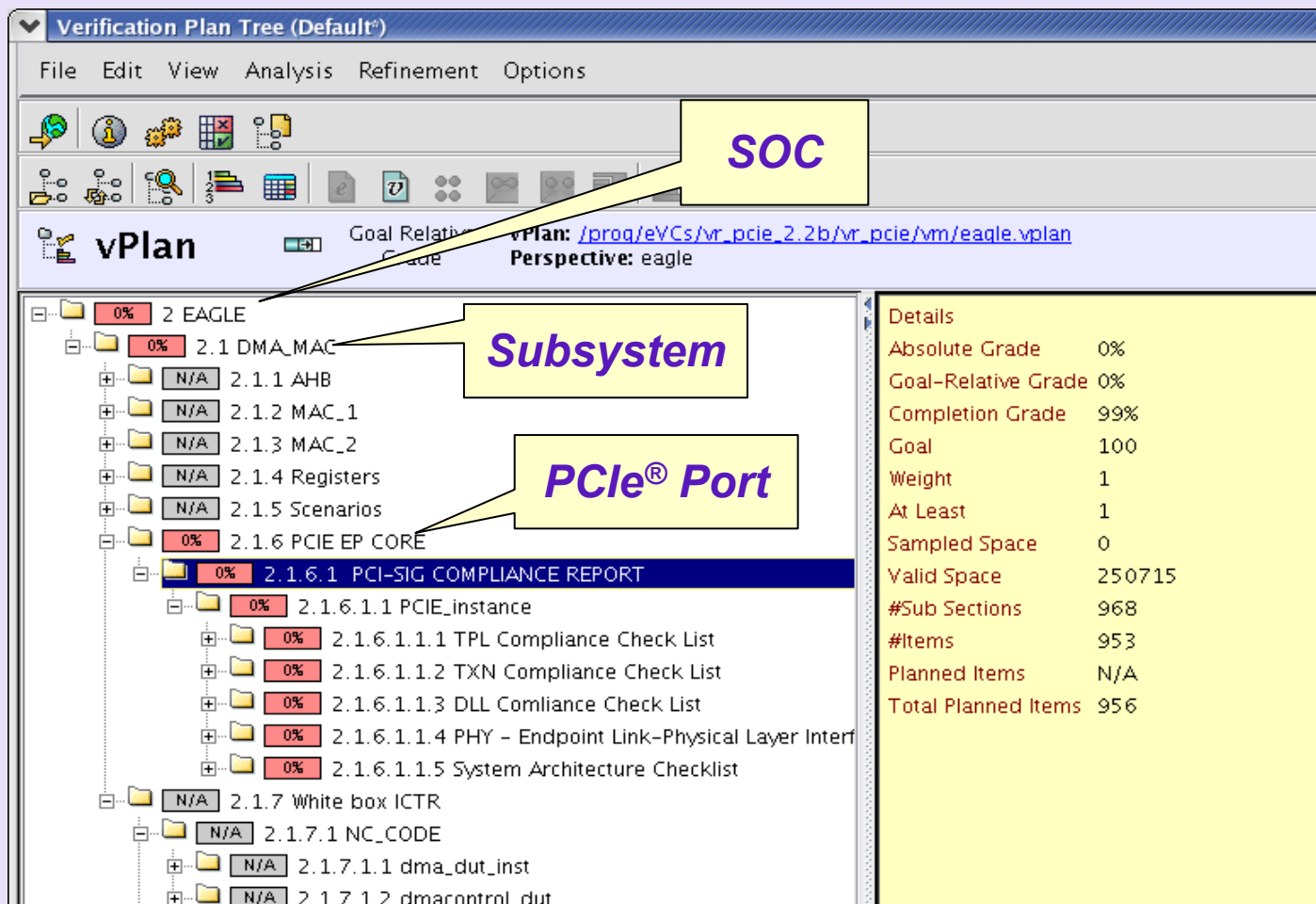
***Compliance Is Necessary But Not Sufficient!***



# Full Device Verification

- With compliance solution in place you can focus on full device verification
- Need an easy way to gather and view compliance data together with (or separate from) full device coverage
- Need assurance for error-free integration
- SOC level aspects must be considered and verified
  - ✓ SW Initialization
  - ✓ System Deadlock conditions
  - ✓ End-to-end verification

# Full Device Verification Plan



**Verification Plan Tree (Default)**

File Edit View Analysis Refinement Options

**SOC**

**Subsystem**

**PCIe® Port**

**vPlan** Goal Relative Grade vPlan: /prog/eVCs/vr\_pcie\_2.2b/vr\_pcie/vm/eagle.vplan Perspective: eagle

- 0% 2 EAGLE
  - 0% 2.1 DMA\_MAC
    - N/A 2.1.1 AHB
    - N/A 2.1.2 MAC\_1
    - N/A 2.1.3 MAC\_2
    - N/A 2.1.4 Registers
    - N/A 2.1.5 Scenarios
    - 0% 2.1.6 PCIE EP CORE
      - 0% 2.1.6.1 PCI-SIG COMPLIANCE REPORT
        - 0% 2.1.6.1.1 PCIE\_instance
          - 0% 2.1.6.1.1.1 TPL Compliance Check List
          - 0% 2.1.6.1.1.2 TXN Compliance Check List
          - 0% 2.1.6.1.1.3 DLL Compliance Check List
          - 0% 2.1.6.1.1.4 PHY - Endpoint Link-Physical Layer Interf
          - 0% 2.1.6.1.1.5 System Architecture Checklist
        - N/A 2.1.7 White box ICTR
          - N/A 2.1.7.1 NC\_CODE
            - N/A 2.1.7.1.1 dma\_dut\_inst
            - N/A 2.1.7.1.2 dmacontrol\_dut

**Details**

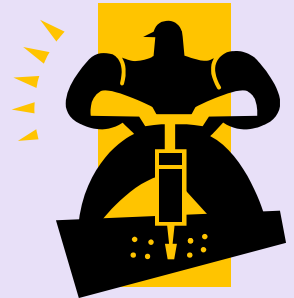
Absolute Grade	0%
Goal-Relative Grade	0%
Completion Grade	99%
Goal	100
Weight	1
At Least	1
Sampled Space	0
Valid Space	250715
#Sub Sections	968
#Items	953
Planned Items	N/A
Total Planned Items	956

Screen Capture From

**cadence**

# Summary

- Checklist is necessary and valuable, but not sufficient
- Furthermore, checklist is only for compliance, not full device verification
  - ✓ Delivering quality products requires full device verification plan
- Using the compliance checks correctly provides an essential tool
- Not using the compliance checklist correctly gives a false sense of security



Thank you for attending the  
PCI-SIG Developers Conference 2006.

For more information please go to  
[www.pcisig.com](http://www.pcisig.com)



**Looking Beyond the  
Compliance Checklist:  
Functionally Verifying PCI Express Compliance  
and SOC Functionality**

**PCI-SIG Developer Conference**

**Erez Kovshi  
Cadence Design Systems**

