Managing Functional Coverage

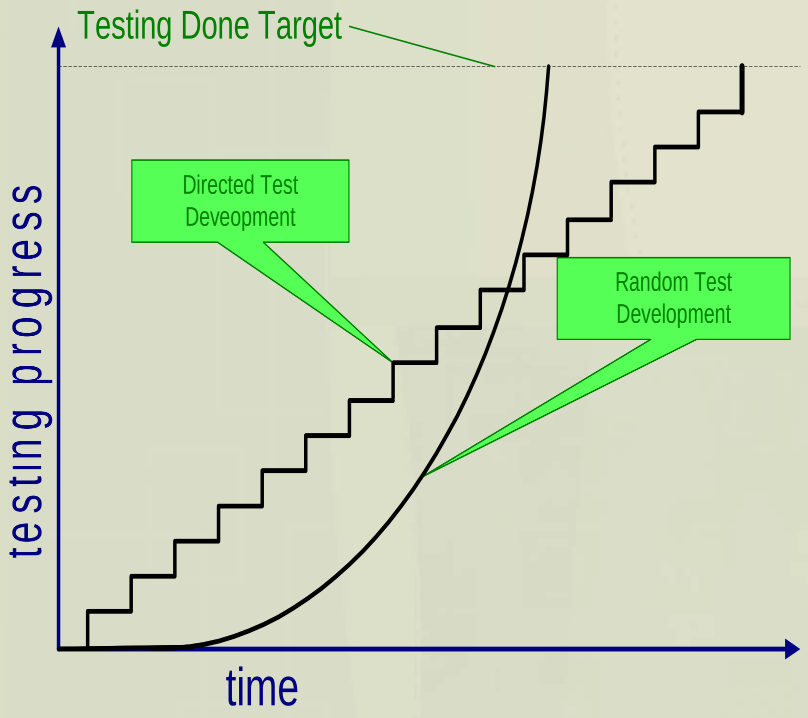
By Stephen D'Onofrio
Principal Consulting Engineer

Introduction

- Functional Coverage Overview
- Functional Coverage Planning
- Functional Coverage Instrumentation & Random Testing
- Tracking/Closing Functional Coverage
  - Closing the loop with the Plan and Closure

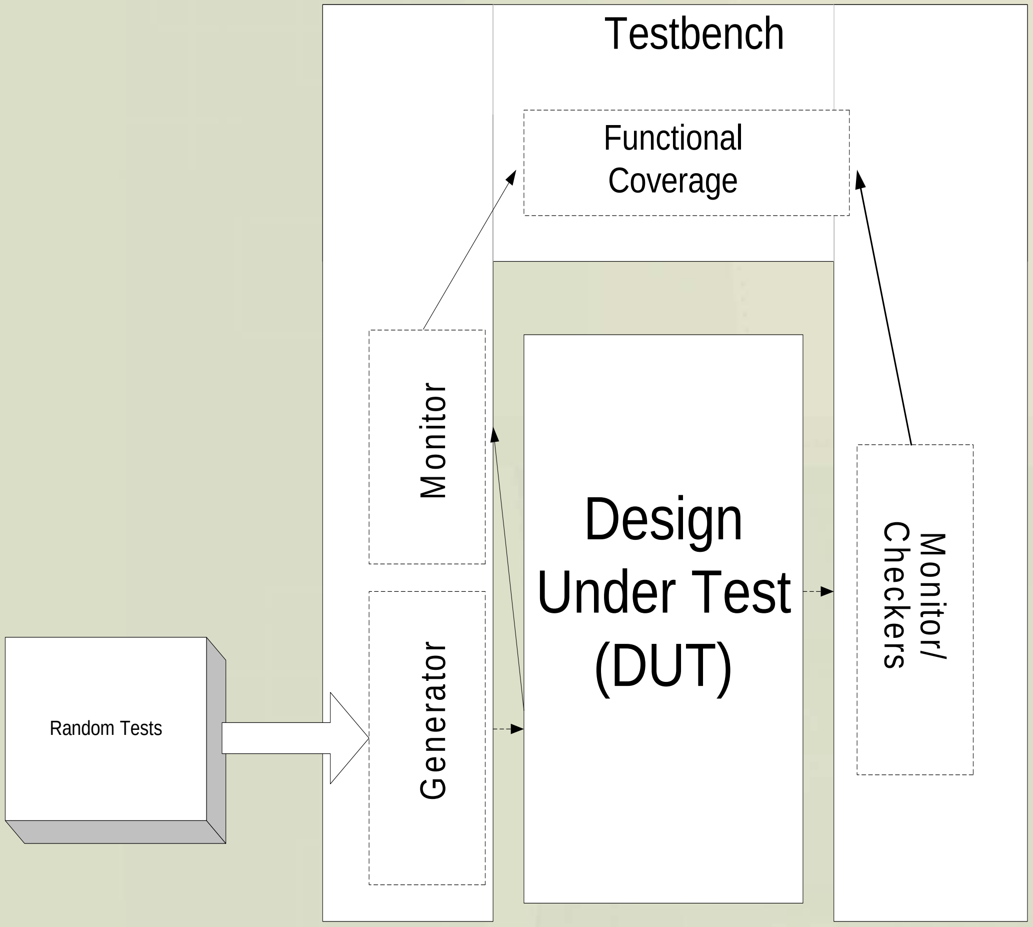
Functional Verification

- Design's features are more complex than ever resulting in huge verification space!
- Constraint random verification with automated checkers



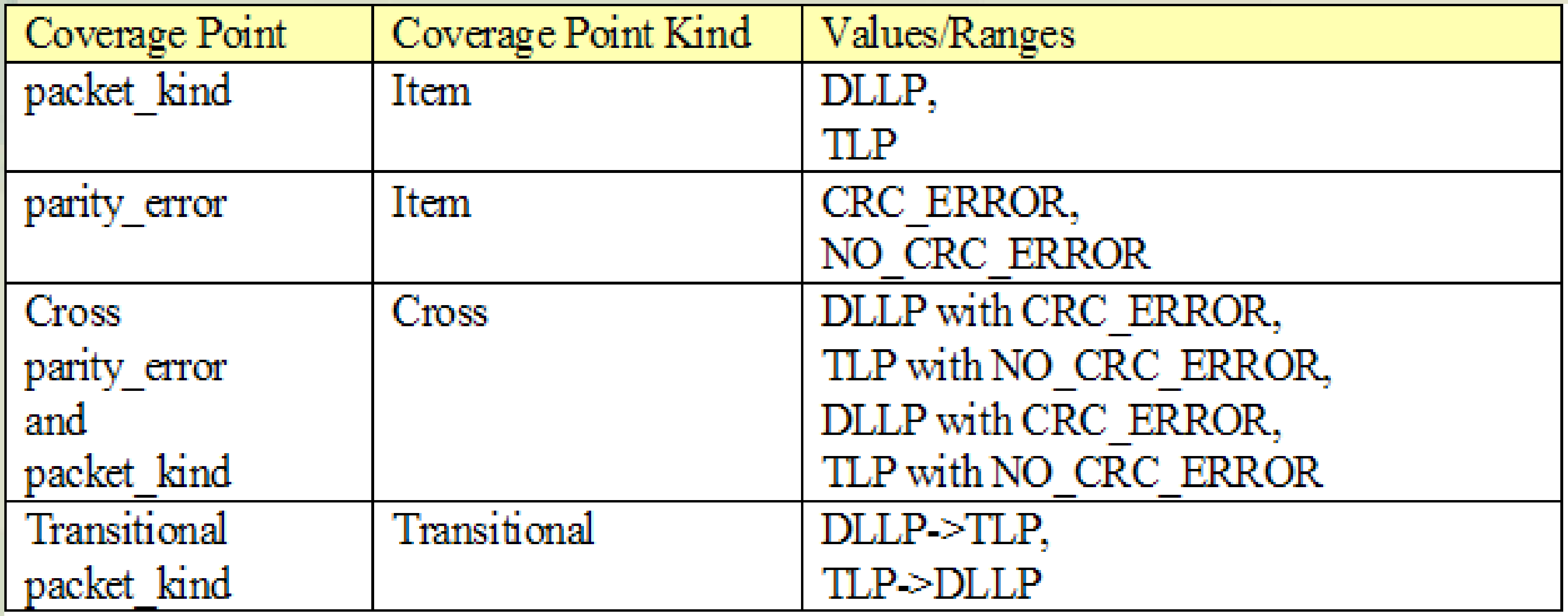
Random Testbench

- Involves three key components
  - Generators
  - Monitors
  - Functional Coverage



Functional Coverage 101

- What is a functional coverage
- What kinds of functional coverage points
  - Item
  - Cross
  - Transitional



Coverage Point	Coverage Point Kind	Values/Ranges
packet_kind	Item	DLLP, TLP
parity_error	Item	CRC_ERROR, NO_CRC_ERROR
Cross parity_error and packet_kind	Cross	DLLP with CRC_ERROR, TLP with NO_CRC_ERROR, DLLP with CRC_ERROR, TLP with NO_CRC_ERROR
Transitional packet_kind	Transitional	DLLP->TLP, TLP->DLLP

PARADIGM® WORKS

Functional Coverage Barriers

- Barriers to using functional coverage
  - Novelty or misinterpretation of functional coverage
  - Very manual process
  - Why not use only Code Coverage?
    - Limited to the design code it covers
    - Input stimulus limits
    - Abstraction level
    - Not completely automatic i.e. pragmas
  - Education plus involvement
    - Functional coverage tells us what randomness was exercised in a regression(s)

Verification Planning

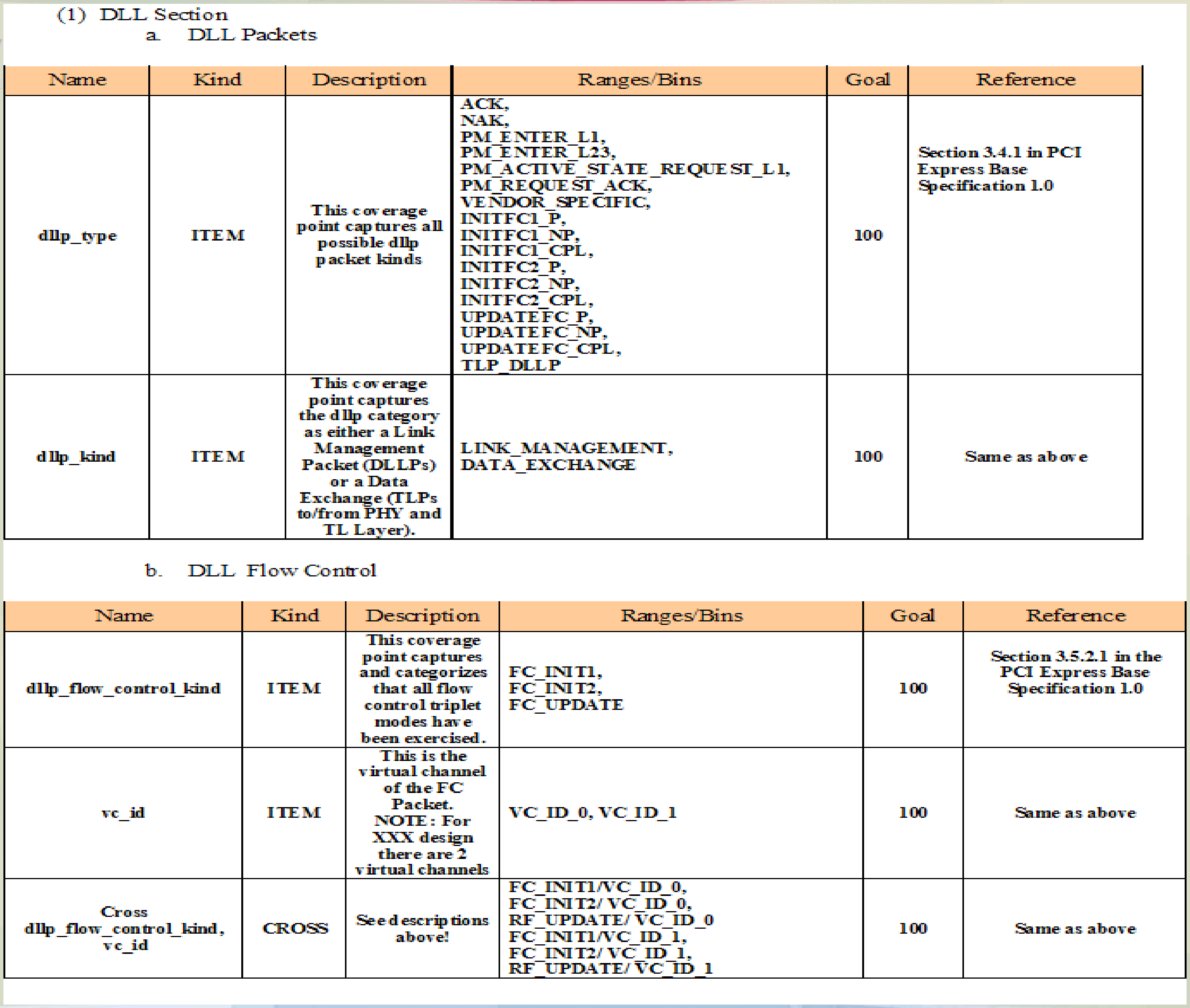
- Single document or a library of documents
  - Testbench Architecture
  - Functional Coverage Plans
  - Doneness Criteria
    - Sign-off functional coverage plans
    - When to merge functional coverage data
    - Expected functional coverage grades
    - Expected code coverage grades
- Why have a verification plan?
  - Puts everyone on the same page
  - Key for identifying tasks for schedule

PARADIGM® WORKS

Functional Coverage Plan

- Multiple documents
- Hierarchal sections that map to design
- Made up of **functional coverage points**
- Identifies a **target** for the randomness to spray
- Assumes coverage points are updated by **random testing**
- Clearly describes the intent of the functional coverage
- Plan at the earliest possible stage of the verification effort
- Reviewed and part of the **Doneness Criteria**

Function Coverage Plan Example



(1) DLL Section

a. DLL Packets

Name	Kind	Description	Ranges/Bins	Goal	Reference
dllp_type	ITEM	This coverage point captures all possible dllp packet kinds	ACK, NAK, PM_ENTER_L1, PM_ENTER_L23, PM_ACTIVE_STATE_REQUEST_L1, PM_REQUEST_ACK, VENDOR_SPECIFIC, INITFC1_P, INITFC1_NP, INITFC1_CPL, INITFC2_P, INITFC2_NP, INITFC2_CPL, UPDATEFC_P, UPDATEFC_NP, UPDATEFC_CPL, TLP_DLLP	100	Section 3.4.1 in PCI Express Base Specification 1.0
dllp_kind	ITEM	This coverage point captures the dllp category as either a Link Management Packet (DLLPs) or a Data Exchange (TLPs to/from PHY and TL Layer).	LINK_MANAGEMENT, DATA_EXCHANGE	100	Same as above

b. DLL Flow Control

Name	Kind	Description	Ranges/Bins	Goal	Reference
dllp_flow_control_kind	ITEM	This coverage point captures and categorizes that all flow control triplet modes have been exercised.	FC_INIT1, FC_INIT2, FC_UPDATE	100	Section 3.5.2.1 in the PCI Express Base Specification 1.0
vc_id	ITEM	This is the virtual channel of the FC Packet. NOTE: For XXX design there are 2 virtual channels	VC_ID_0, VC_ID_1	100	Same as above
Cross dllp_flow_control_kind, vc_id	CROSS	See descriptions above!	FC_INIT1/VC_ID_0, FC_INIT2/ VC_ID_0, RF_UPDATE/ VC_ID_0 FC_INIT1/VC_ID_1, FC_INIT2/ VC_ID_1, RF_UPDATE/ VC_ID_1	100	Same as above

PARADIGM® WORKS

Functional Coverage Plan Options

- weight
  - Typically 1
- goal
  - Typically 100%
  - System Level or Proven Design IP maybe < 100%
- at least count
  - Typically 1
  - Interrupt processing may want > 1
- instance vs. cumulative
  - Instance gives us more confidence
- cross
  - Grow exponentially
  - Ignore unnecessary or unobtainable combinations

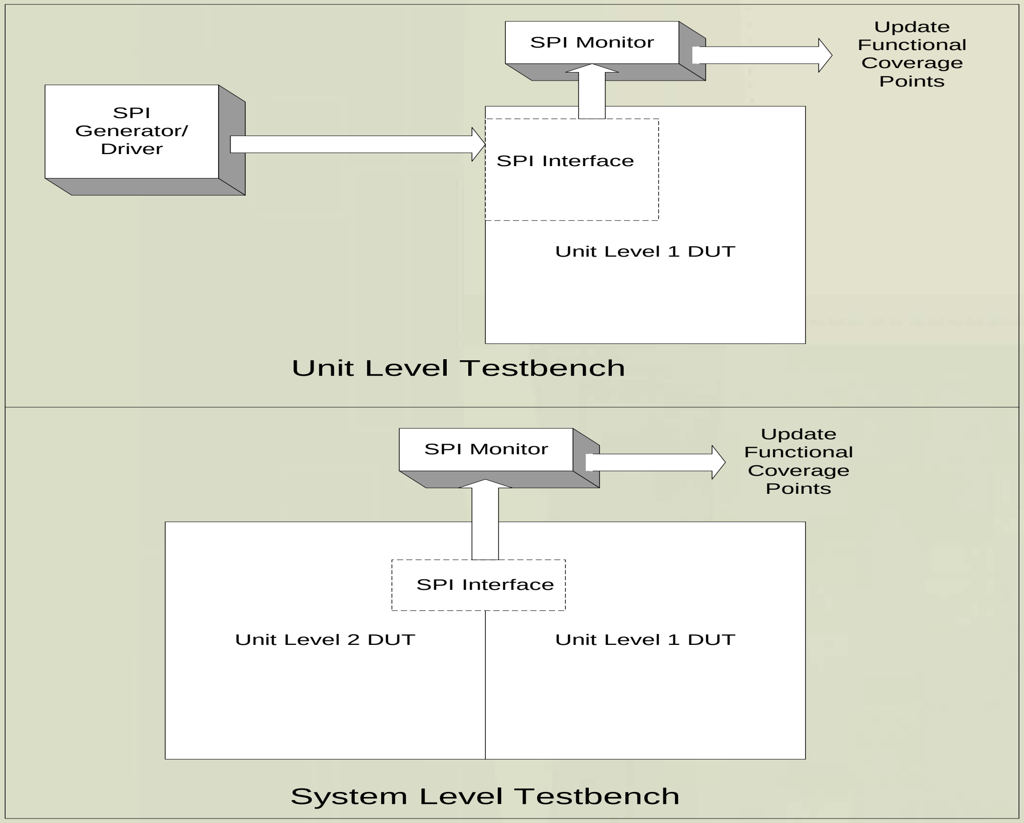
PARADIGM® WORKS

Functional Coverage Planning

- What role does Code Coverage play with Functional Coverage?
- 3rd Party Verification Components and Functional Coverage

Functional Coverage Instrumentation

- Relatively easy to implement
  - Don’t delay till the end ☹
- Implement in Passive code
  - Unit VE to System VE



Functional Coverage Instrumentation Example

```
if (pkt.type != DLLP_REPLAY) {
        error("Design sending unexpected packet type");
}
else {
        update_dllp_functional_coverage(pkt);
}
```

- When to update
  - Help reduce *False Positive* concerns
  - Update at highest level abstraction and update only when necessary
- Where to implement
  - Need scope of multiple objects
  - May need to develop new events
- Suggestions
  - Use enumerators for named constants whenever possible
  - Don't assert errors with functional coverage

PARADIGM® WORKS

Testing and Functional Coverage

- Regress as few *very* random tests as possible
- Keep tests relatively short and execute many tests with random seeds
- Utilize *test ranking*
  - Measure randomness – optimize hitting verification space
- When to develop directed test cases
- Reactive testing?

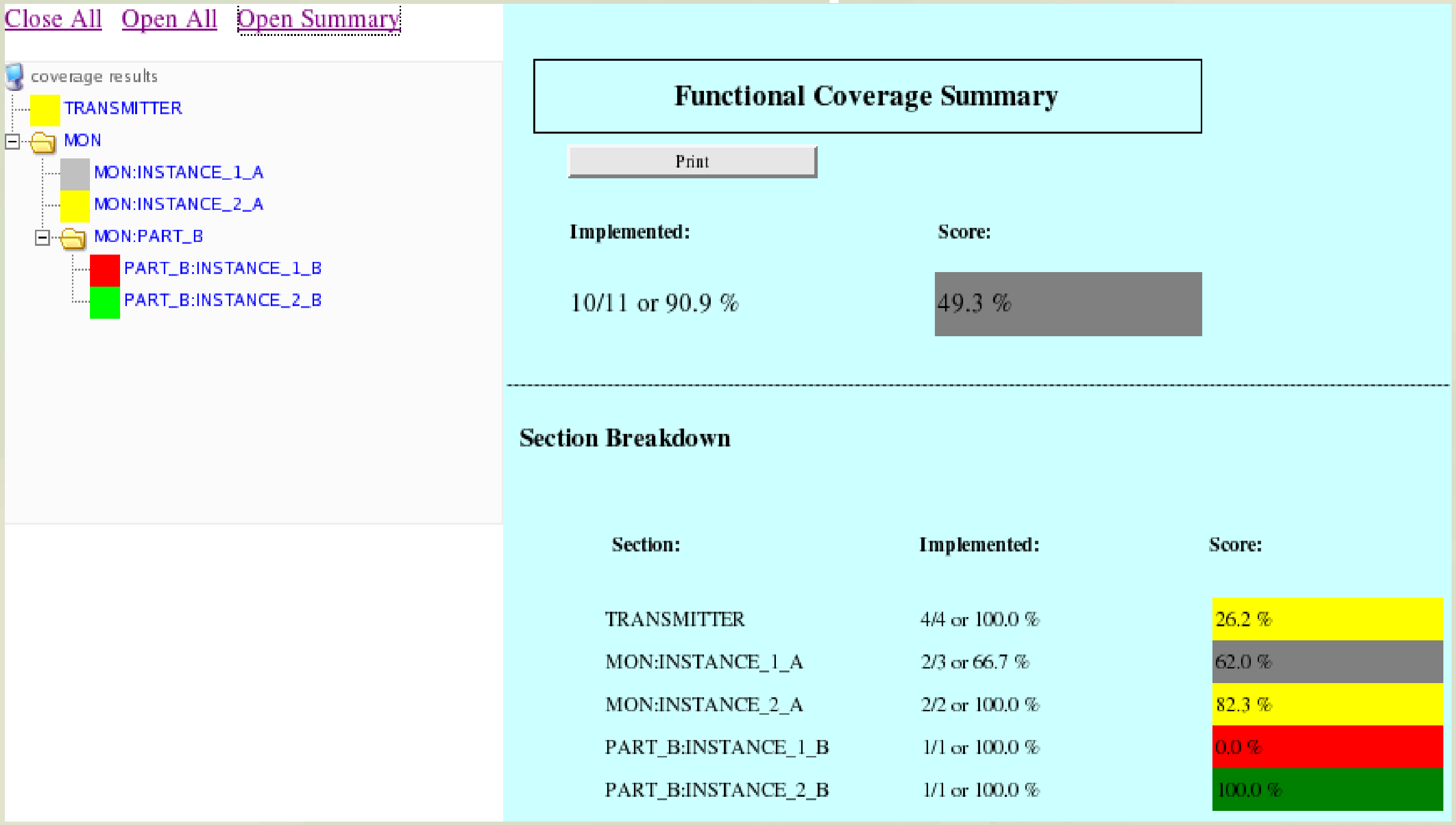
PARADIGM® WORKS

Tracking Functional Coverage

- Closing Functional Coverage Closure
- Identifying holes and calculating grades
- Functional Coverage Plan vs. Functional Coverage Raw Database
- Rerunning tests/regressions
- Tweaking verification environment & adding new focused tests

PARADIGM® WORKS

HTML Summary Output



Close All   Open All   Open Summary

- coverage results
  - 🟨 TRANSMITTER
  - MON
    - ⬜ MON:INSTANCE_1_A
    - 🟨 MON:INSTANCE_2_A
    - MON:PART_B
      - 🟥 PART_B:INSTANCE_1_B
      - 🟩 PART_B:INSTANCE_2_B

Functional Coverage Summary

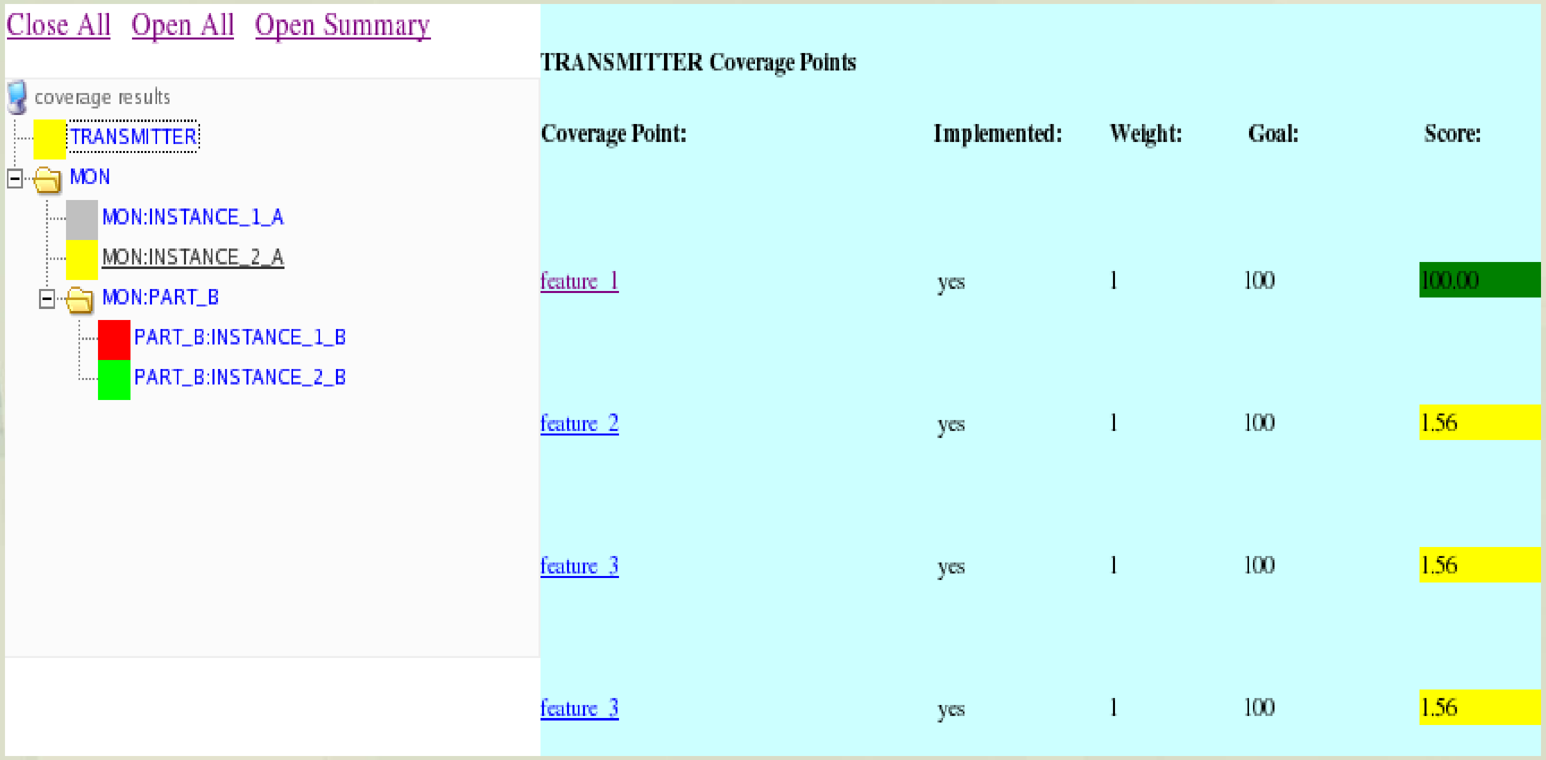
Print

Implemented:	Score:
10/11 or 90.9 %	49.3 %

Section Breakdown

Section:	Implemented:	Score:
TRANSMITTER	4/4 or 100.0 %	26.2 %
MON:INSTANCE_1_A	2/3 or 66.7 %	62.0 %
MON:INSTANCE_2_A	2/2 or 100.0 %	82.3 %
PART_B:INSTANCE_1_B	1/1 or 100.0 %	0.0 %
PART_B:INSTANCE_2_B	1/1 or 100.0 %	100.0 %

HTML Coverage Output



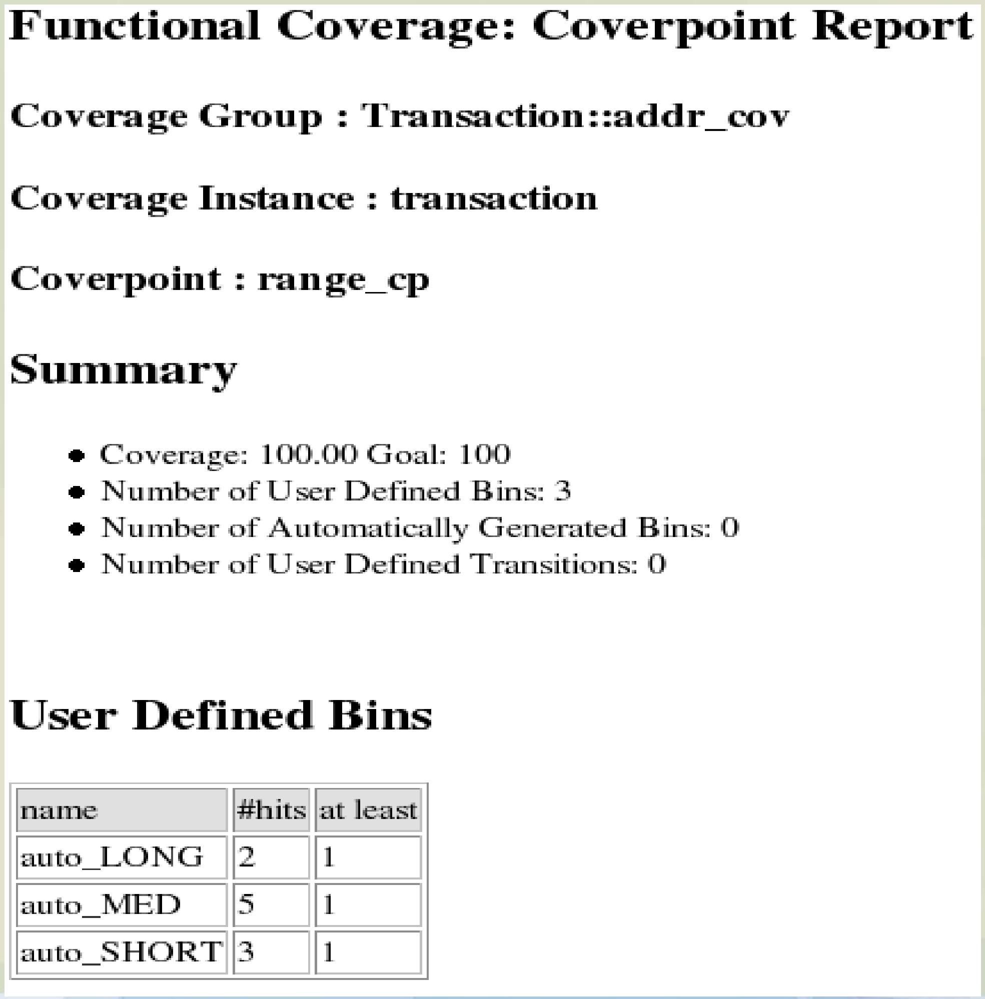
Close All Open All Open Summary

- coverage results
  - 🟨 TRANSMITTER
  - MON
    - ⬜ MON:INSTANCE_1_A
    - 🟨 MON:INSTANCE_2_A
    - MON:PART_B
      - 🟥 PART_B:INSTANCE_1_B
      - 🟩 PART_B:INSTANCE_2_B

TRANSMITTER Coverage Points

Coverage Point:	Implemented:	Weight:	Goal:	Score:
feature_1	yes	1	100	100.00
feature_2	yes	1	100	1.56
feature_3	yes	1	100	1.56
feature_3	yes	1	100	1.56

SVF FC Tool Links to Synopsys Coverage Info



Functional Coverage: Coverpoint Report

Coverage Group : Transaction::addr_cov

Coverage Instance : transaction

Coverpoint : range_cp

Summary

- Coverage: 100.00 Goal: 100
- Number of User Defined Bins: 3
- Number of Automatically Generated Bins: 0
- Number of User Defined Transitions: 0

User Defined Bins

name	#hits	at least
auto_LONG	2	1
auto_MED	5	1
auto_SHORT	3	1

PARADIGM® WORKS

Functional Coverage Conclusions

- Provide *insight* to design features testing in a random verification environment
- Proper *level of abstraction* for verifying design specification
- *Doneness Criteria* should include signed-off *living* functional coverage plans
- *Track* functional coverage based on the functional coverage plan's points

Contact Information:

Paradigm Works, Inc.

300 Brickstone Square

Andover, MA 01810

stephen.donofrio@paradigm-works.com

www.paradigm-works.com