



## Coverage Driven Verification

**Offers a much higher form of predictability to meet coverage goals in an automated fashion**

Verification has emerged as the single biggest bottleneck in complex ASIC, ASSP, and SoC design projects and it consumes 60-70 percent of project effort. SoCs in particular present a massive verification challenge, as not only do they include multiple functional blocks, they often have multiple operating modes. Achieving satisfactory verification requires that the cross product of all functional block interactions and all operating modes be tested with shorter regression cycles covering all test possibilities and eVCs based on **Coverage Driven Verification (CDV)** methodology help achieve this easily.

The verification capabilities can be measured based on verification methodology.

Different methodologies used are,

- Test-based verification: Full directed stimuli generation
- Automated verification: Software to generate stimuli and temporal checking
- Reuse Methodology based verification: In this case, various test scenarios/Reference Models/BFM components can be reused
- Coverage Driven Verification: This is done with combination of random and directed testing with functional/code coverage tool to confirm the maximum coverage

The biggest challenge in verification of a large SoC is to write a lot of test cases to cover all combinations. Problems in test-based verification are:

- Outlining all possible combinations
- Finding out optimum way to write test cases to drive DUT to a desired state
- Verify that DUT reached the desired state
- Check for correct DUT response

Here verification relies on manual effort and therefore, is time-consuming. All efforts are to trace expected bugs and in the process the unexpected bugs escape.

More efforts will be required in reconfirmation and may be in reconstruction, if the device changes. The management of test based verification process is difficult in the sense that it cannot track the progress, which results in less confidence in the product.

The problems in the test-based verification create demand for new verification methodology which

- needs less manual efforts
- takes lesser time to verify
- finds hidden bugs
- offers measurable progress and hence more confidence in the product



Verity's Coverage Driven Verification (CDV) methodology addresses all the problems of test-based verification. The key here is coverage measurement – **“what you can't measure, you can't manage”**. The coverage report shows how far are we from

completion and the holes show how many more simulations are required to cover all possibilities.

### **Using CDV**

The CDV development process includes:

- Defining a verification plan  
In verification plan focus is to identify important attributes based on analysis of device inputs, their most important values and important combinations of device input values. Following are meaningful:
  - Values at the pin level
  - Values within data structures that pass through the pins
  - Value abstracted from above values
  - Time gap between successive values
  
- Defining coverage model and generation scheme  
To define coverage model use both functional coverage and code coverage. Functional coverage measures activity at RTL against goals derived from the specification. As the goals are defined manually, important metrics may be omitted. Functional coverage needs feedback to find the missing metrics. Code coverage measures activity at RTL against goals derived from the RTL. Code coverage alone is not enough as the implementation bugs may influence it.

The best approach here is to use functional coverage as primary measurement technique and then analyze code coverage results to catch omissions in the functional coverage model.

Use sequences to exercise the stimulus required to achieve the functionality

- Running tests, getting feedback from coverage holes  
Run simulation, generate coverage report, analyze report, constrain stimulus to cover holes.

**Yet to Hit**

```

** PHY_2_2_2_VALID (cross pkt_kind, err_kind) **
Samples: 576 Tests: 1 Grade: 0.17 Weight: 1

```

grade	goal	samples	tests	%p/%t	pkt_kind/err_kind
0.50	-	576	1	100/100	NULLTLP
1.00	1	576	1	100/100	NULLTLP/NO_ERROR
0.00	1	0	0	0/0	NULLTLP/OTHER_ERROR
0.00	-	0	0	0/0	TLP
0.00	-	0	0	0/0	DLLP

- Creating checking scheme  
The illegal coverage definition shows invalid combination and thus supplements checking.

**Errors caught in Coverage**


```

** cross__ltssm_kind__order_kind__link_num__lane_num **
Samples: 3512 Tests: 3 Grade: 0.04 Weight: 1

```

grade	goal	samples	tests	%p/%t	ltssm_kind/order_kind/link_num/lane_num
0.04	-	3512	3	100/100	POLLING_CONFIGURATION
0.00	-	216	2	6/6	POLLING_CONFIGURATION/TSL
0.00	-	0	0	0/0	POLLING_CONFIGURATION/TSL/No PAD on Link
0.00	-	216	2	100/6	POLLING_CONFIGURATION/TSL/PAD on Link
0.00	-	0	0	0/0	POLLING_CONFIGURATION/TSL/PAD on Link/No PAD on Lanes
0.00	-	216	2	100/6	POLLING_CONFIGURATION/TSL/PAD on Link/PAD on Lanes
NA	0	216	2	100/6	POLLING_CONFIGURATION/TSL/PAD on Link/PAD on Lanes *** ILLEGAL ***
0.00	1	0	0	0/0	POLLING_CONFIGURATION/TSL/PAD on Link/PAD on Lanes

- Optimizing simulation through test ranking  
The tests are ranked by their coverage efficiency. This highlights redundant tests, which add nothing to the overall coverage. Based on the test ranking an optimized list for next mini regression can be formed.



Maximal Grade: 0.61								
Index	Tests	Cumulative Grade (out of 0.61)	Relative Grade	Cumulative Cost	Cost	Absolute Grade (%)	Efficiency (normalized)	Unique Buckets
<b>Optimized Tests</b>								
	test_dirty2_seq_11	0.86	0.86	0	NA	0.53	NA	35
<input type="checkbox"/>	1 test_dirty2_seq_12	0.94	0.08	0	NA	0.52	NA	127
<input type="checkbox"/>	2 test_phy_recovery_state_19	0.98	0.05	0	NA	0.48	NA	2
<input type="checkbox"/>	3 test_request_to_seq_23	0.99	0.01	0	NA	0.46	NA	11
<input type="checkbox"/>	4 test_dirty_seq_14	0.99	0.01	0	NA	0.45	NA	9
<input type="checkbox"/>	5 test_dirty_hr_disable_sequences_24	0.99	0.01	0	NA	0.48	NA	2
<input type="checkbox"/>	6 test_bug_23	1.00	0.01	0	NA	0.37	NA	-
<b>Redundant Tests</b>								
<input type="checkbox"/>	7 test_demo_13	0.00	0.00	0	NA	0.26	NA	-
<input type="checkbox"/>	8 test_PA2DQ_22	0.00	0.00	0	NA	0.26	NA	-

**Summary:** The coverage measurement provides feedback loops for improving device, environment - stimulus, checking. It also provides information on stimulus effectiveness for progress towards schedule and optimization of regression suite. Coverage driven verification is a promising concept of verification and coverage measurement will be stronger with the introduction of Verisity's Coverage Maximizer. Thus coverage measurement provides the best handle to manage the verification process.