Benefits of PSS coverage at SOC & its limitations

Sundararajan Haran, Principal Engineer, Qualcomm India Private Limited
(sharan@qti.qualcomm.com)
Saleem Khan, Senior Staff Engineer, Qualcomm India Private Limited,
(salekhan@qti.qualcomm.com)

Qualcomm

accellera SYSTEMS INITIATIVE™

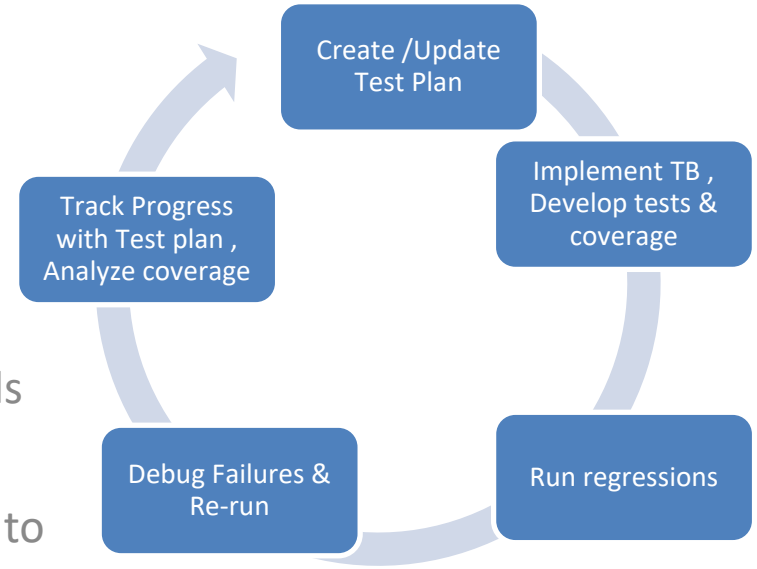
2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Agenda

- Introduction
- PSS coverage overview
- PSS Coverage Advantages
- PSS Gen time coverage – use case
- Current PSS Coverage limitations
- Future Scope
- Summary

Coverage …..

- Coverage Driven Verification (CDV) allows a systematic flow to define and meet the verification goals
- Randomness and automation explore un-anticipated use-cases
- Coverage provides status reports of what was achieved and eventually completeness of goals
- Management can review the produce reports to analyze the progress over-time and take decisions
- Improve overall quality



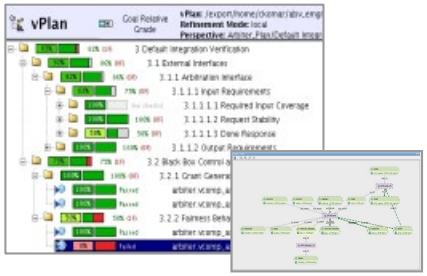
Perspec Portable Stimulus Coverage

- Perspec provides functional coverage on top of the concise behavioral model
  - Allows capturing use-case coverage goals that are impractical in any other way
  - Provides automation to fulfill desired spaces and filter out illegal scenarios (coverage maximization)
  - Can passively prove that a use-case took place without forcing it
- Allows pre-run regression tuning
  - remove redundant tests before running simulation and save a time and resources

Actions model captures the infinite legal scenario space

==

Coverage captures the verification intent and goals



PSS Coverage

Perspec supports PSS coverage collection at pre-run() and post_run()

- Coverage metrics are the same regardless of which collection flow is used

Pre-run() coverage

- Allows projection / approximation of coverage to be collected after test actually runs
- Results are available right after generation is done (no sim needed)
- Can be optionally mirrored for run-time
- **Limitations**:
  - Aware of PSS model only, unaware of "external events"

Post_run() coverage

- Collected from simulation log in post-processing mode ("Mirrored" gen-time coverage)
- Capturing coverage in terms of "external events"

PSS Covergroups

The covergroup construct encapsulates the specification of a coverage model

- Each covergroup is collected separately (by type)

Can include following elements

- A set of coverage points
- Cross coverage between coverage points
- Coverage options

Two forms of covergroup constructs

- Explicit: defined once, instantiated multiple times
  - Can be defined in package, component, action or struct
  - Shall specify formal parameters → shall NOT refer to fields of the scope in which it is defined
  - **All instances of same covergroup type contribute to same single covergroup in the coverage model**
- Implicit (inline)
  - Can be defined in action or struct scope
  - Shall NOT specify formal parameters -> shall refer to fields of the scope in which it is defined
  - **Each inline covergroup definition corresponds to its own covergroup in the coverage model**

Explicit Covergroup

PSS

```
enum color_e {RED, GREEN, BLUE, PINK};
enum shape_e {CIRCLE, TRIANGLE, SQUAR};
component paint_c {
  action draw_shape_1 {
    color_e color;
    shape_e shape;
  };
};
```

PSS

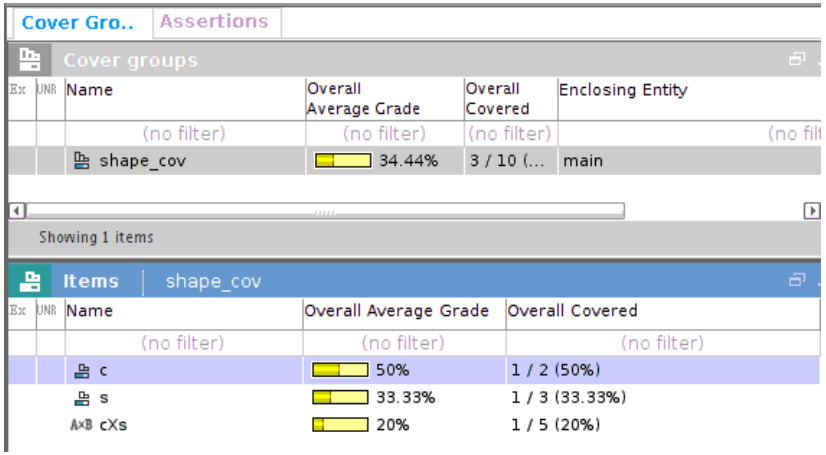
```
covergroup shape_cov (color_e color, shape_e shape) {
  c: coverpoint color {
    ignore_bins i = [PINK, BLUE];
  };

  s: coverpoint shape;

  cXs: cross c, s {
    ignore_bins i = cXs with (c == GREEN and s == TRIANGLE);
  };

};

extend action paint_c::draw_shape_1 {
   shape_cov shape_cov_inst_1 (color, shape);
};
```



Users can define **coverage points** to be sampled after an object which instantiates the covergroup has been solved (as part of scenario solution process)

Users can ignore non-interesting values

Users can cross coverage items, to analyze **combinations** of coverpoint values

Instantiation of cover group in a specific context

Implicit (Inline) Covergroup

PSS

```
enum color_e {RED, GREEN, BLUE, PINK};
enum shape_e {CIRCLE, TRIANGLE, SQUAR};
component paint_c {
  action draw_shape_1 {
    color_e color;
    shape_e shape;
  };
  action draw_shape_2 {...}; //same definition as draw_shape_1
  action draw_shape_3 {...}; //same definition as draw_shape_1

};
```

PSS

```
extend action paint_c::draw_shape_1 {
   shape_cov shape_cov_inst_1 (color, shape);
};

extend action paint_c::draw_shape_2 {
   shape_cov shape_cov_inst_2 (color, shape);
};

extend action paint_c::draw_shape_3 {
  covergroup {
      coverpoint color;
      coverpoint shape;
      cXs: cross color, shape;
  } inline_cov_inst_3;
};
```

Cover Gro.. | Assertions

Cover groups

Ex	UNR	Name	Overall Average Grade	Overall Covered	Enclosing Entity
		(no filter)	(no filter)	(no filter)	(no filt
		shape_cov	34.44%	3 / 10 (...	main
		inline_cov_inst_3	34.44%	3 / 10 (...	draw_shape_3

2 more actions: `draw_shape_2`, `draw_shape_3`

First instance of `shape_cov`

Second instance of `shape_cov`

Inline covergroup instance `inline_cov_inst_3`

`cross` requires label; `coverpoint` label is optional

PSS Coverpoint & Bins

- A covergroup can contain one or more coverage points and their crosses
- Each coverage point and cross is broken down into a set of bins
- Three type of bins. Same syntax is used to define all three types
  - "regular" bins. Number of these bins determines the size of coverage space spanned by respective coverage point or cross
  - ignore bins
  - illegal bins
- Partition into bins ("binning")
  - can be explicitly defined by user, or
  - automatically created by PSS tool
  - important skill to master in order to effectively manipulate coverage spaces !

PSS Covergroup Specification Constructs

```
covergroup {
  //option.per_instance = true;
  option.at_least = 2;

  c_l1 : coverpoint pw_state_t;
  c_l2 : coverpoint freq_mode_t;

  c_l3 : coverpoint idx_data {
    bins idx_v = [0..1,7];
    ignore_bins idx_il = [8,9];
    bins others[] = default;
  }

  c_l4 : coverpoint sid_x iff (is_hs_enabled) {
    illegal_bins not_legal = [3,5];
  }

  all : cross freq_mode_t,sid_x
} cg_implicit;
```

A covergroup can contain one or more coverage points and their crosses

Regular bin

Ignore bin

Default bin

Illegal bin

Cross coverage

Where to Collect Coverage in PSS model ?

Actions

- The natural location for controlled and action-specific coverage definition
- Can refer to fields of tokens produced / consumed by action
- Can refer to component attributes via comp. attribute

Structs

- The coverage is collected as they are being randomized
- May use structs specifically dedicated for coverage collection purpose

Agenda

- Introduction
- PSS coverage overview
- PSS coverage Advantages
- PSS Gen time coverage – use case
- Current PSS Coverage limitations
- Future Scope
- Summary

PSS Coverage Advantages

Perspec Generated coverage shift lefts the project Verification cycle

- Generation and Coverage analysis starts before running first test
- Rank generated tests and optimize testsuite for regression
- Using pre-run regression planning saves the resources, farm utilization

Perspec Scenario Coverage == SoC Quality

- Concurrency coverage
- Temporal relations and shmoo

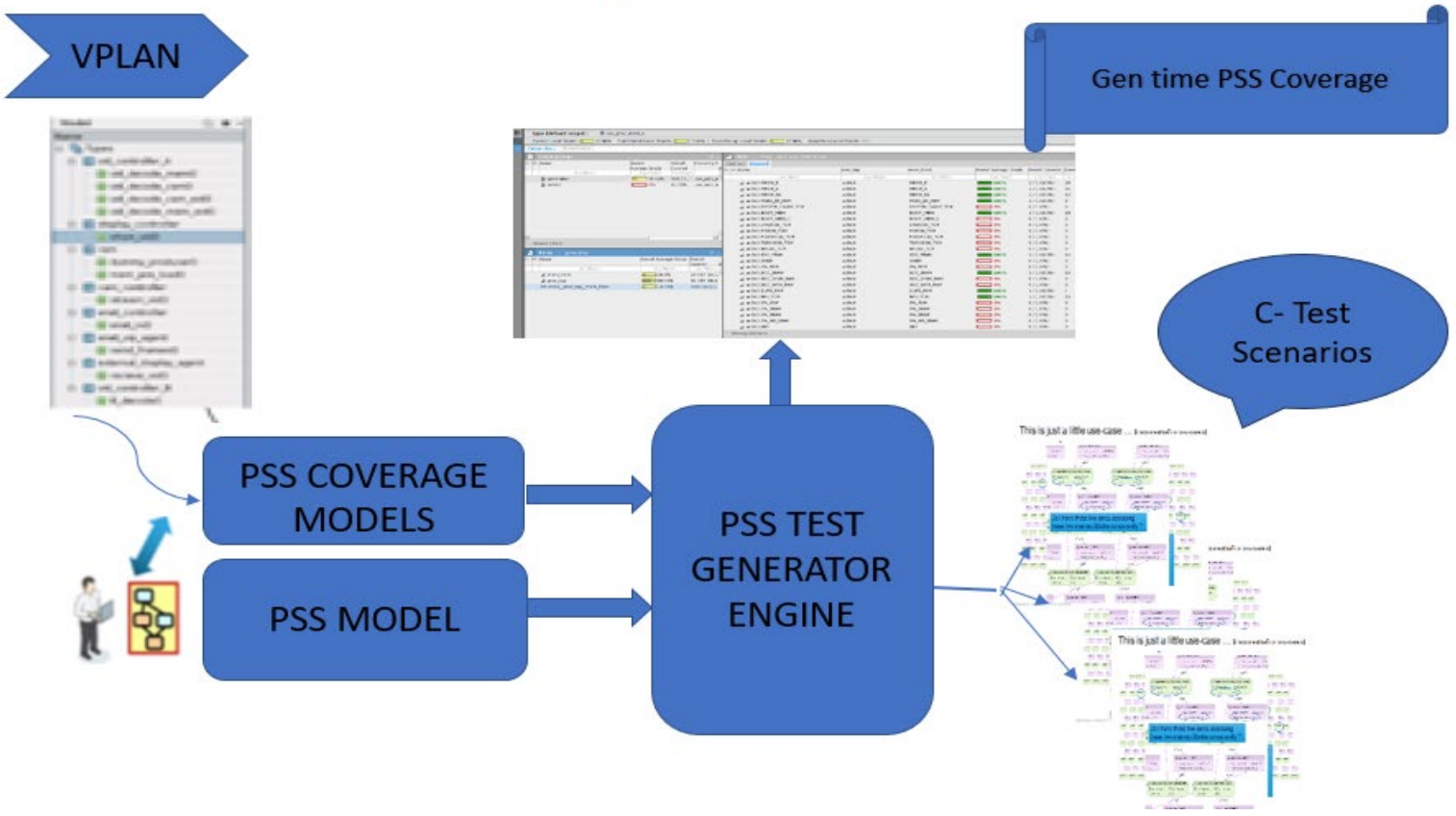
Functional coverage on all platforms including ICE, post-silicon

Proven Coverage Driven Verification methodology (MDV) expanded to SW and HW co verification

Agenda

- Introduction
- PSS Coverage Overview
- PSS coverage Advantages
- PSS Gen time coverage – use case
- Current PSS Coverage limitations
- Future Scope
- Summary

PSS Generation coverage flow



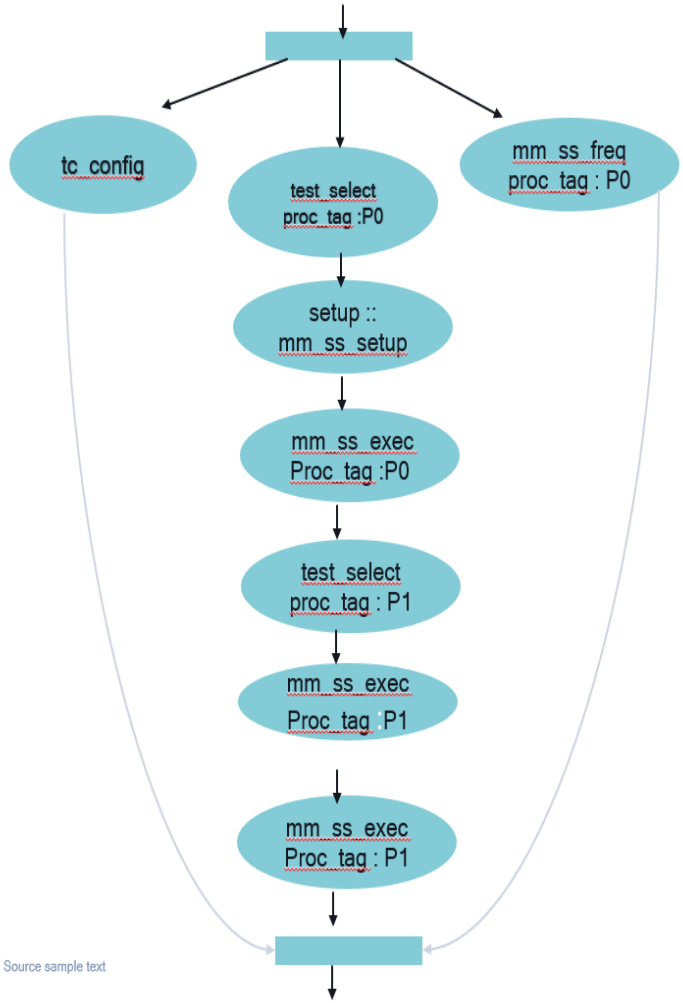
Frequency mode PSS cover group

```
enum frequency_e {freq_mode_1, freq_mode_2, freq_mode_3, freq_mode_4, freq_mode_5};

component pss_top {
  action freq_switch {
    rand frequency_e freq_plan[1];
  };
struct freq_switch_cov {
  rand  frequency_e my_freq_e;
  covergroup {
    coverpoint my_freq_e {};
  } cov_freq;
}

extend action DVE::freq_switch {
  freq_switch_cov freq_switch_l[5];

  exec post_solve {
    foreach (item: freq_plan[i]) {
      freq_switch_l[i].my_freq_e = freq_plan[i];
    };
  };
};
```



Perspec Portable Stimulus Coverage Examples

Ex	UNR	Name	Overall Average Grade	Overall Covered	Score
		(no filter)	(no filter)	(no filter)	
		auto[freq_mode_1]	100%	1 / 1 (100%)	2
		auto[freq_mode_2]	0%	0 / 1 (0%)	0
		auto[freq_mode_3]	0%	0 / 1 (0%)	0
		auto[freq_mode_4]	0%	0 / 1 (0%)	0
		auto[freq_mode_5]	100%	1 / 1 (100%)	3

Covered

Uncovered

Additional tests were generated to cover

Ex	UNR	Name	Overall Average Grade	Overall Covered	Score
		(no filter)	(no filter)	(no filter)	
		auto[freq_mode_1]	100%	1 / 1 (100%)	2
		auto[freq_mode_2]	100%	1 / 1 (100%)	1
		auto[freq_mode_3]	n/a	0 / 0 (n/a)	0
		auto[freq_mode_4]	100%	1 / 1 (100%)	1
		auto[freq_mode_5]	100%	1 / 1 (100%)	3

Excluded/Ignored

Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	
my_freq_e	100%	4 / 4 (100%)

Covered

Agenda

- Introduction
- PSS Coverage overview
- PSS Coverage Advantages
- PSS Gen time coverage – use case
- Current PSS Coverage limitations
- Future Scope
- Summary

PSS Limitations

PSS supports constructs to describe temporal relations between actions executing in sequence or parallel . This is captured in compound actions with the concept of “activity”.

“Activity “ , describe the flow of test to be generated. This is stimulus aspect ,If EDA tool can identify the occurrence between these behaviors patterns in a given test execution this could as well be used to Monitor execution coverage.

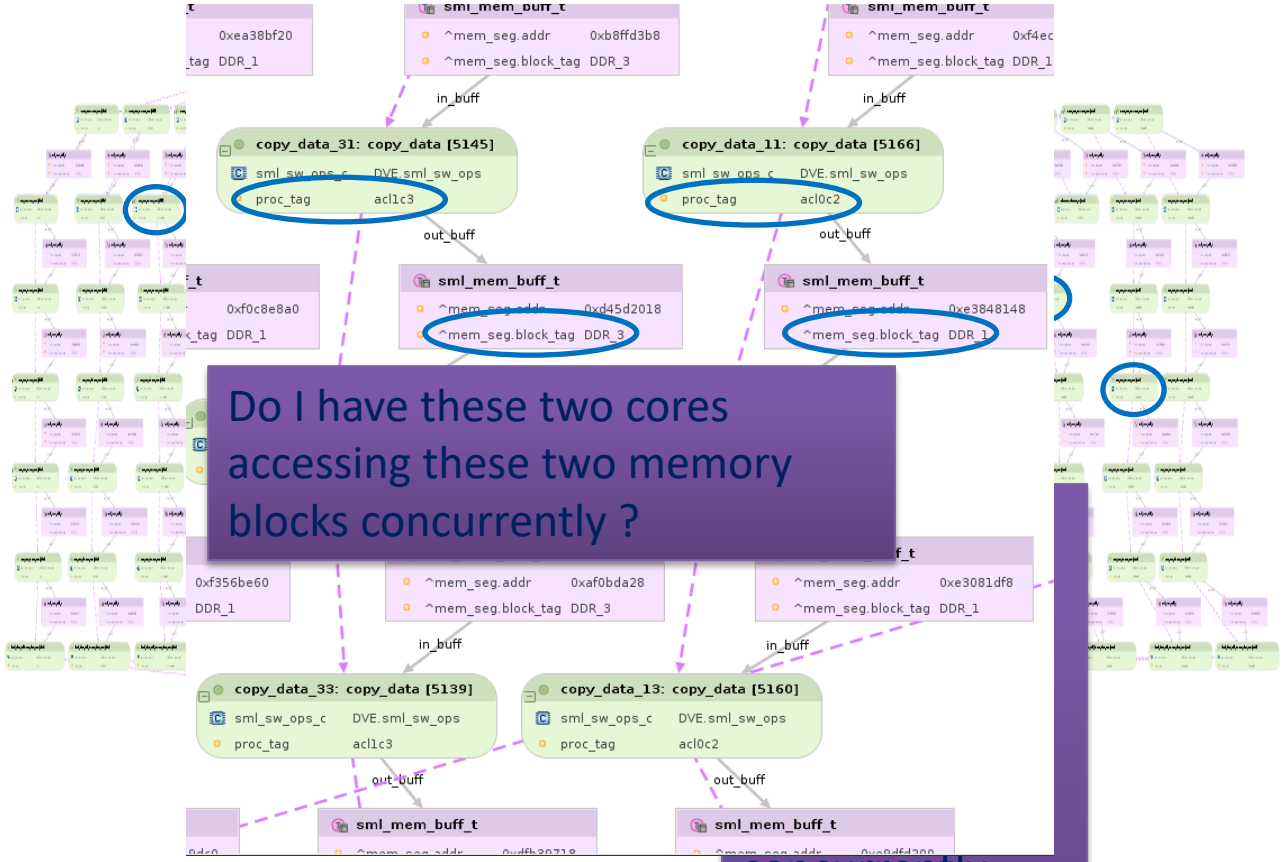
However , PSS seems to be missing a way to bind behavioral primitives with actual terms that can be observed in execution.

Eg : On stimulus side user can define temporal relations between behaviors ,whereas we will not be able to feed events on starting & ending of behaviors . With addition of event support in PSS language it would be ready to take coverage driven verification to next level.

Agenda

- Introduction
- PSS Coverage overview
- PSS Coverage Advantages
- PSS Gen time coverage – use case
- Current PSS Coverage limitations
- Future Scope
- Summary

This is just a little use-case …



Agenda

- Introduction
- PSS Coverage overview
- PSS Coverage Advantages
- PSS Gen time coverage – use case
- Current PSS Coverage limitations
- Future Scope
- Summary

Summary : Benefits of PSS coverage

- PSS Generation coverage analysis is done as standalone without running simulation thereby saves long simulation run time effort.

- PSS Run time coverage support helps to model top level Use case, Performance scenario’s and correlate against the Vplan.

- Future scope: PSS coverage at execution.

Questions

THANK YOU

accellera SYSTEMS INITIATIVE™