2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
FEBRUARY 24-27, 2025

What Just Happened? Behavioral Coverage Tracking in PSS

Tom Fitzpatrick, Wael Mahmoud, Mohamed Nafea

SIEMENS

accellera SYSTEMS INITIATIVE

Authors

Tom Fitzpatrick

Wael Mahmoud

Mohamed Nafea

Agenda

- Motivation
- PSS behavioral coverage in Brief
- Measuring behavioral coverage and scenarios' quality framework
- Use-Cases
  - Use Case I: Simple use-case
  - Use Case II: DMA VIP
- Conclusion

Motivation

- Complex SoCs require testing various scenarios to ensure critical functionality.
- PSS 3.0 introduces Behavioral Coverage to measure defined scenario coverage.
- Randomized solvers in PSS generate varied scenarios, making it hard to predict coverage beforehand.
- Behavioral Coverage allows users to monitor solver decisions, inferred actions, and chosen paths during scenario generation.
- There is no native construct in PSS to track the number of unique generated scenarios.
- Identifying coverage holes helps refine models and constraints to meet coverage goals.
- A methodology is proposed to evaluate coverage, identify missed scenarios, and improve overall test quality.

PSS Behavioral Coverage in Brief

Data Coverage (like SV/UVM)

- Covergroups capture data values
  - Including cross-coverage
- Sampled at end of action traversal by default
- "Transaction coverage," i.e.:
  - size of packet produced by send_packet action
  - addr region of DMA destination
  - channel used for DMA action

Behavioral Coverage

- Capture key action ordering
  - Including data values
- Monitors describe scenarios
- Cover directives capture described scenarios
- "Scenario coverage," i.e.:
  - Did B follow A?
    - Was B.x < 10?
  - Did C and D overlap?

PSS Behavioral Coverage in Brief

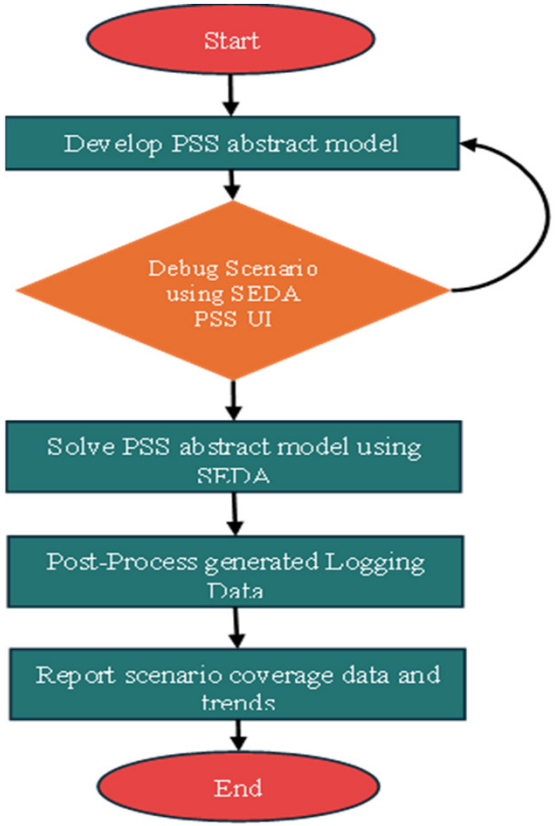
- *Monitor* activity captures behaviors of interest
  - Sequential traversals
    - Including concatenation and eventuality
  - Overlapping traversals
    - The monitor equivalent of parallel
  - Selection of traversals
    - Either A or B
  - Arbitrary combinations of these
- Includes constraints of action/flow object fields

Tracking Behavioral Coverage

- Some coverage *could* be tracked at compile time
  - If it's not subject to scenario randomization
  - Generally not very compelling, but a possible process optimization
- Coverage must really be based on runtime behavior
  - Consider parallel actions in a single-CPU multi-threaded system
  - Required to analyze coverage of reactive scenarios

Measuring Behavioral Coverage

- PSS model defines critical verification intent
  - Optionally debug model in Questa Portable Stimulus Visualizer
- Simulate PSS model in Questa using runtime QPS solver
- QPS Coverage extracts traversed actions post-sim
- QPS Coverage reports
  - Trends and coverage of traversed scenarios
  - Number of occurrences of each unique scenario
  - Trend graphs to measure distribution of generated scenarios



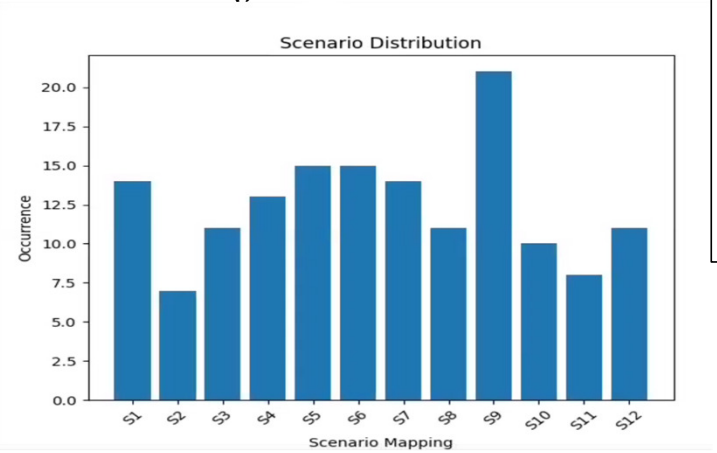
Use Case I: Simple use-case

- Description:
  - 4 simple atomic actions (A, B, C, and D), traversed inside the **`top_act`** compound action, with each pair of actions traversed inside a **`select`** activity statement
- Results:
  - There are four possible traversals of the activity
    - **`{sequence {A}; {C};}`**
    - **`{sequence {A}; {D};}`**
    - **`{sequence {B}; {C};}`**
    - **`{sequence {B}; {D};}`**

```
component pss_top {
    action A { }
    action B { }
    action C { }
    action D { }
    action top_act {
        activity {
            repeat (150) {
                    select {do A; do B;}
                    select {do C; do D;}
            }
        }
    }
}
```

Use Case I: Simple use-case (Cont.)

- Description:
  - Modified version of the previous example by adding a **schedule** statement
- Results:
  - There are 12 possible traversals of the activity, as shown below:
    1. sequence {parallel {A; C;}}
    2. sequence {parallel {A; D;}}
    3. sequence {parallel {B; C;}}
    4. sequence {parallel {B; D;}}
    5. sequence {sequence {A; C;}}
    6. sequence {sequence {A; D;}}
    7. sequence {sequence {B; C;}}
    8. sequence {sequence {B; D;}}
    9. sequence {sequence {C; A;}}
    10. sequence {sequence {C; B;}}
    11. sequence {sequence {D; A;}}
    12. sequence {sequence {D; B;}}



  - All possible scenarios were randomly traversed within the 150 loop iterations

```
component pss_top {
    action A { }
    action B { }
    action C { }
    action D { }
    action top_act {
        activity {
            repeat (150) {
                schedule {
                    select {do A; do B;}
                    select {do C; do D;}
                }
            }
        }
    }
}
```

Turning Stimulus Into Coverage

Sequential Behavior

- Stimulus Activity

```
sequence {do A;do C;}
```

- Monitor Activity

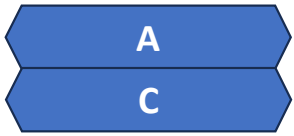
```
sequence {do A;do C;}
concat {do A;do C;}
eventually {do A;do C;}
```

- In this example, the three monitor activities are the same

Parallel Behavior

- Stimulus Activity

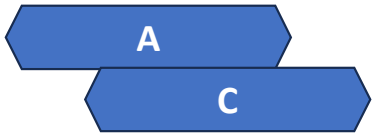
```
parallel {do A; do C;}
```



- Monitor Activity

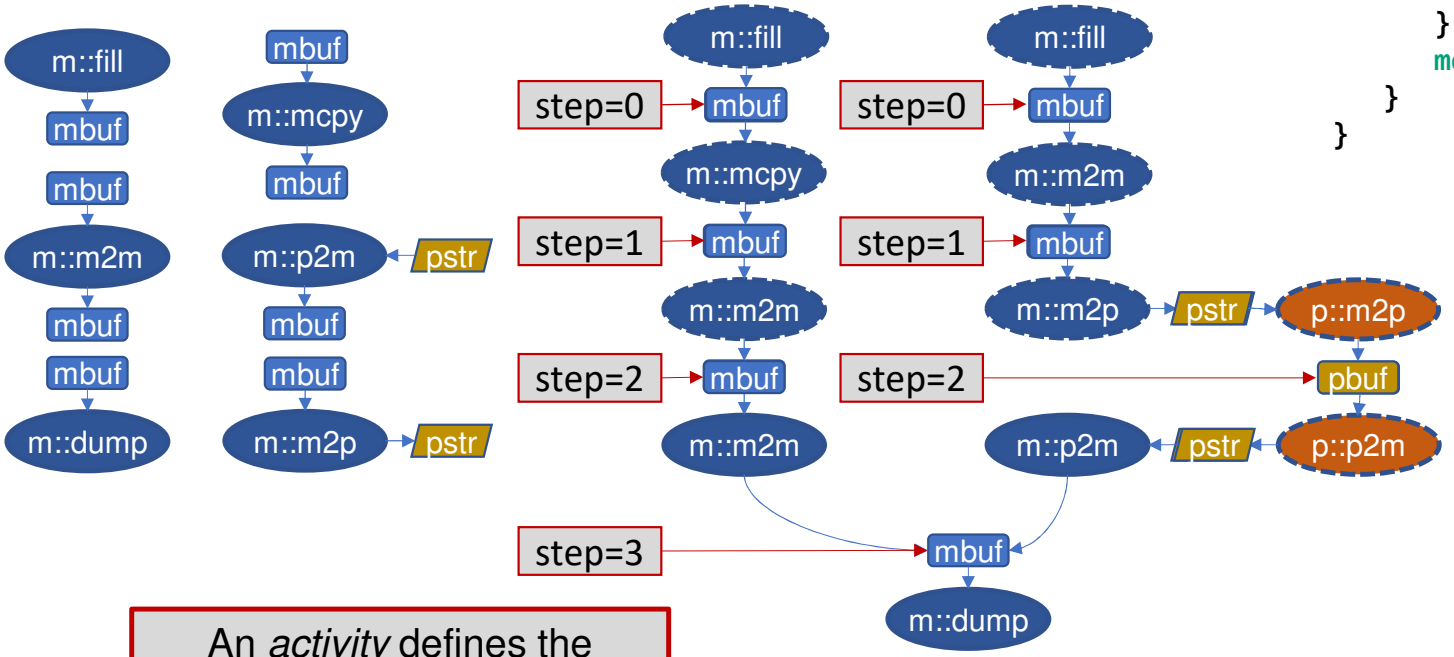
```
overlap {do A; do C;}
```

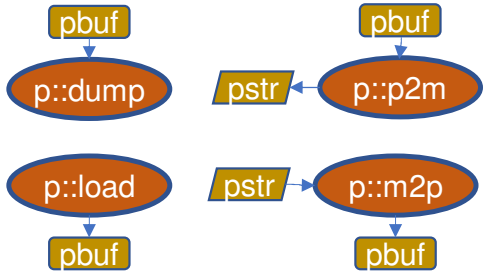
- *overlap* indicates either
  - C starts before A ends, OR
  - A starts before C ends

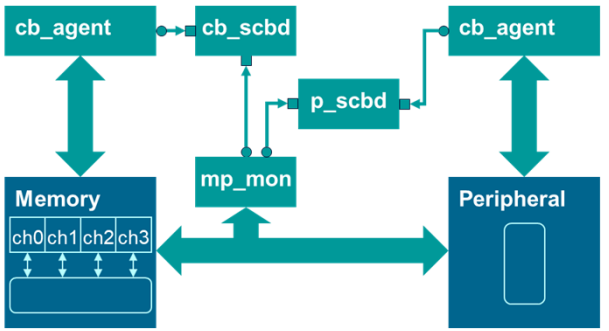


Defining a PSS Scenario

```
constraint mdump.ibuf.step == 3;
activity {
   repeat (4) {
      select {
         p2m with {obuf == mdump.ibuf;};
         m2m with {obuf == mdump.ibuf;};
      }
      mdump;
   }
}
```



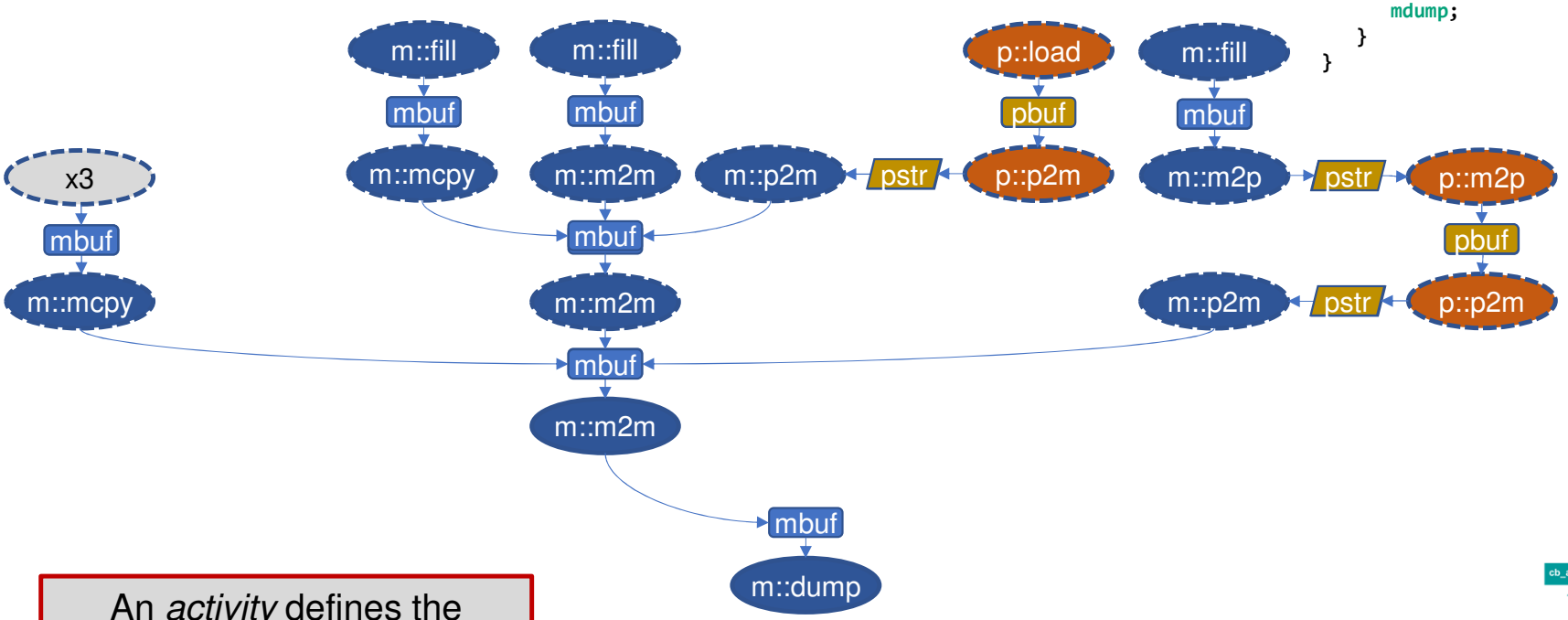




An *activity* defines the schedule of action execution

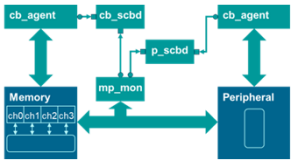
Analyzing the Possibilities

```
constraint mdump.ibuf.step == 2;
activity {
   repeat (4) {
      select {
         p2m with {obuf == mdump.ibuf;};
         m2m with {obuf == mdump.ibuf;};
      }
      mdump;
   }
}
```



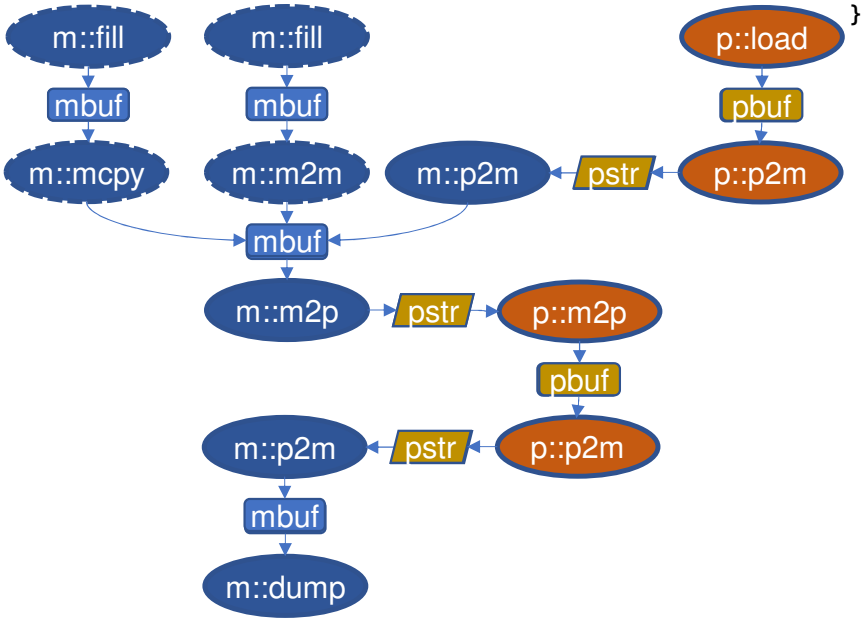
An *activity* defines the schedule of action execution

{m2m;dump}:7

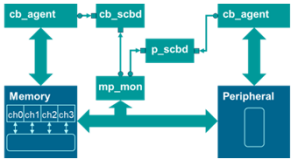


Analyzing the Possibilities

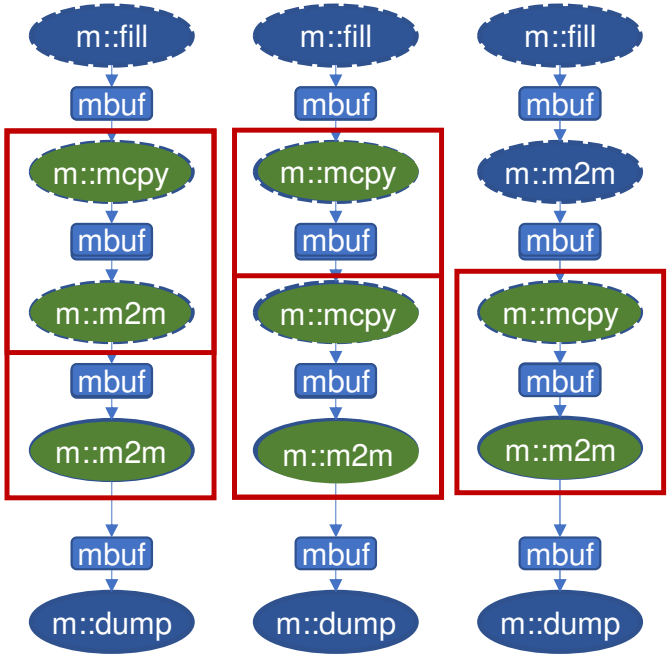
```
constraint mdump.ibuf.step == 2;
activity {
   repeat (4) {
      select {
         p2m with {obuf == mdump.ibuf;};
         m2m with {obuf == mdump.ibuf;};
      }
      mdump;
   }
}
```



{p2m;dump}:3



Use Case II: DMA VIP (Results)



- Behavioral Coverage

```
c1: cover {
  activity {
    sequence {
      do cb_mem_c::cb_mem_mcpy;
      do cb_mem_c::cb_mem_m2m;
    }
  }
}
```

```
c2: cover {
  activity {
    concat {
      do cb_mem_c::cb_mem_mcpy;
      do cb_mem_c::cb_mem_m2m;
    }
  }
}
```

Conclusion

- As SoCs get more complex, PSS models are getting bigger
- Scenario coverage is applicable at block, sub-system, and full system
- Post-processing allows scenario coverage to be extracted
- Questa measures all generated PSS scenarios with a variety of metrics
  - Total number of unique scenarios
  - Distribution of generated scenarios
- Two use cases used to explain the proposed framework

Questions