Portable Test and Stimulus Standard (PSS)

Santeri Laurila
Alper Özaslan

PSS Basics

- Portable Test and Stimulus Standard is used to help with reusing existing verification assets
- Using PSS, verification environments and meaningful tests can be reused across different verification engines, technologies and projects

Why Portable Stimulus?

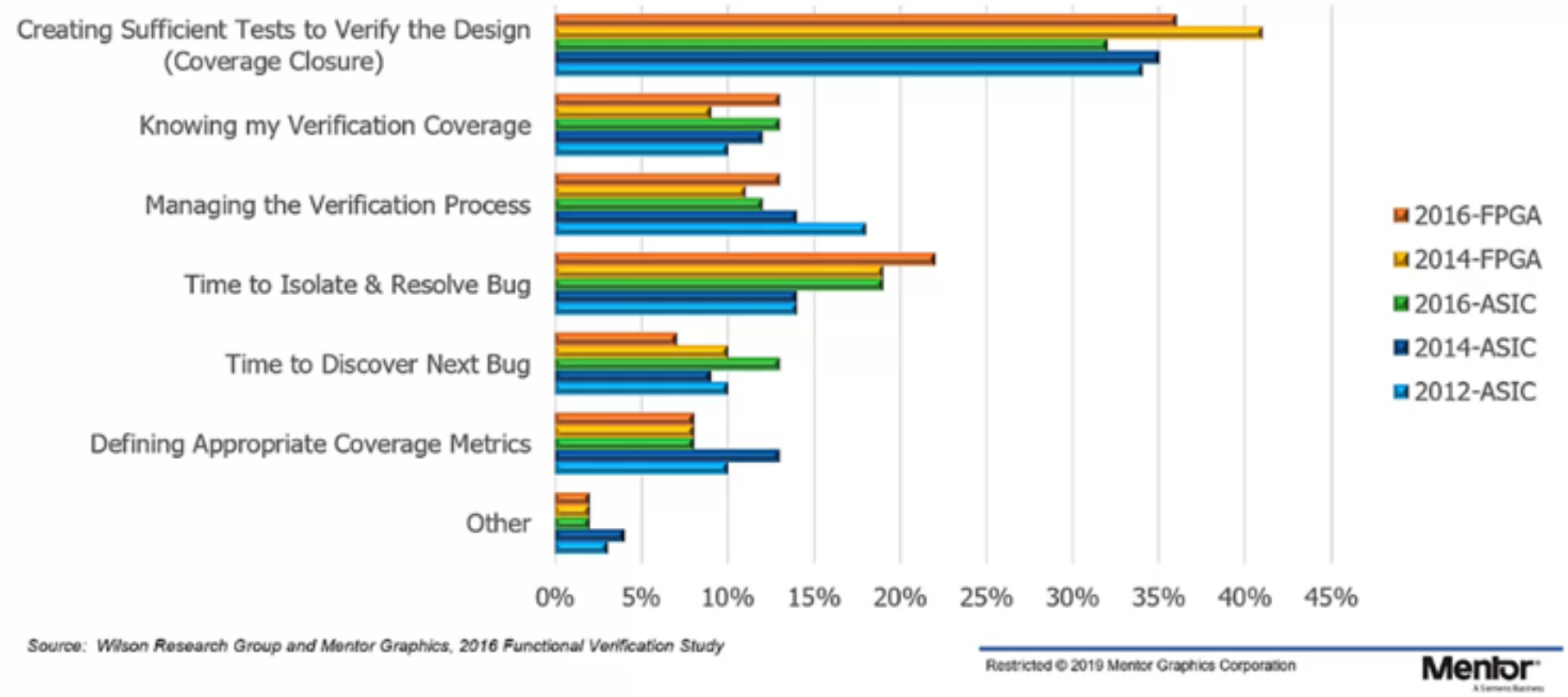


Fig. 1 Biggest Verification Challenges today

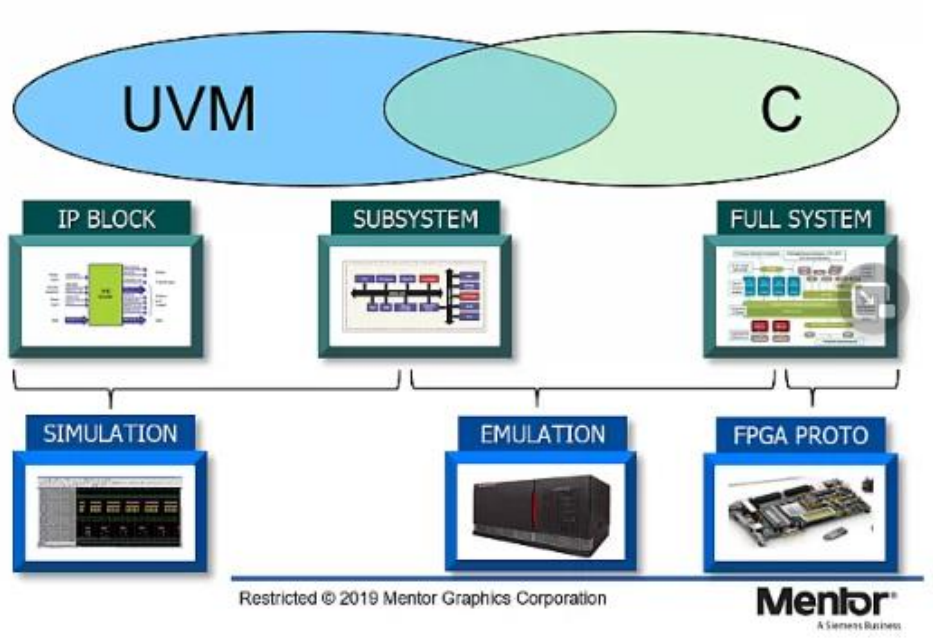
Why Portable Stimulus? (continued)

- Design complexity continues to increace
- Multiple platforms
- Need to reuse Test Intent
  - Higher abstraction
  - Reuse from block to system
  - Map to different platforms

- Verification needs a productivity boost → **PSS**

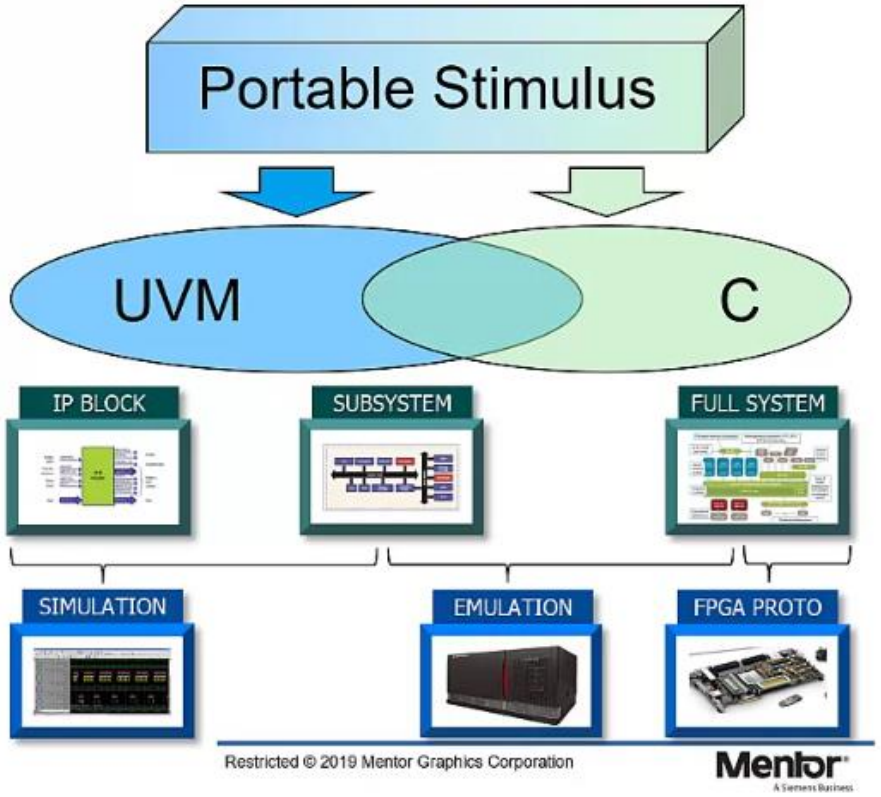
Reuse of Test Intent Across Platforms

- Different tests are used throughout a project
  - Wastes time
  - Error Prone
- UVM constrained-random is limited for SoC-level testing
- C tests are usually directed
  - Hard to create
  - Miss corner cases



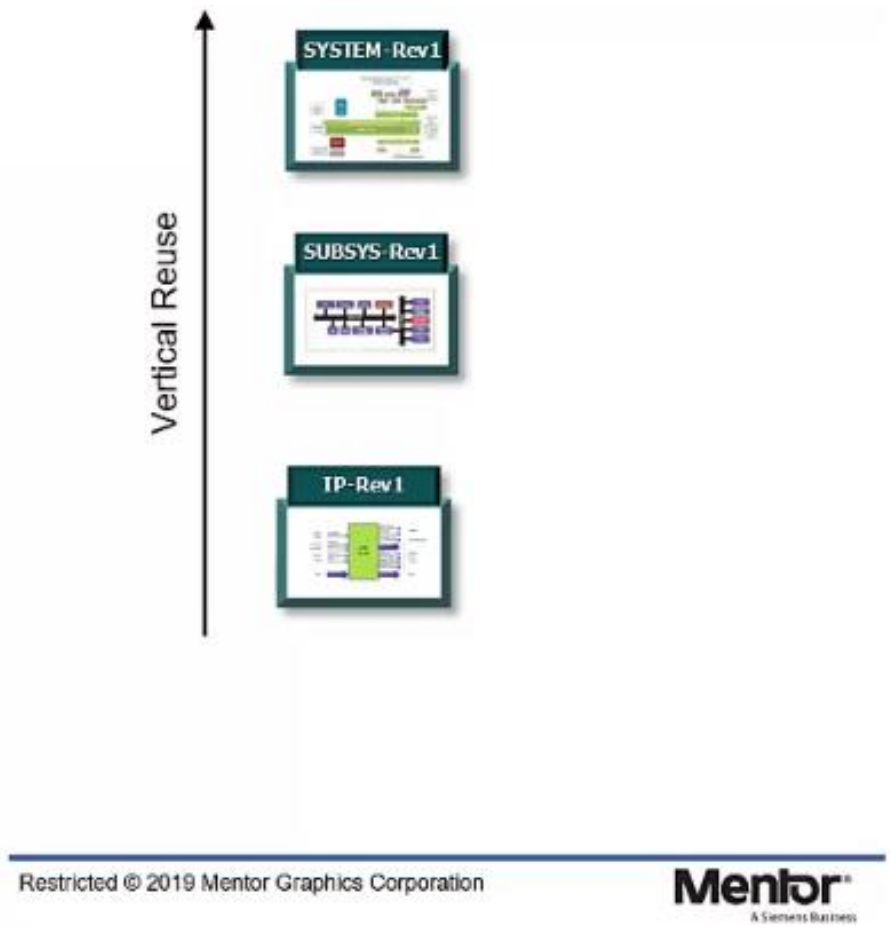
Reuse of Test Intent

- Single specification of test intent is critical
- Define ”scenario space” by capturing
  - interactions dependencies
  - Resource contention
- Abstraction lets tool automate generation
  - Multiple targets
  - Target-specific customization



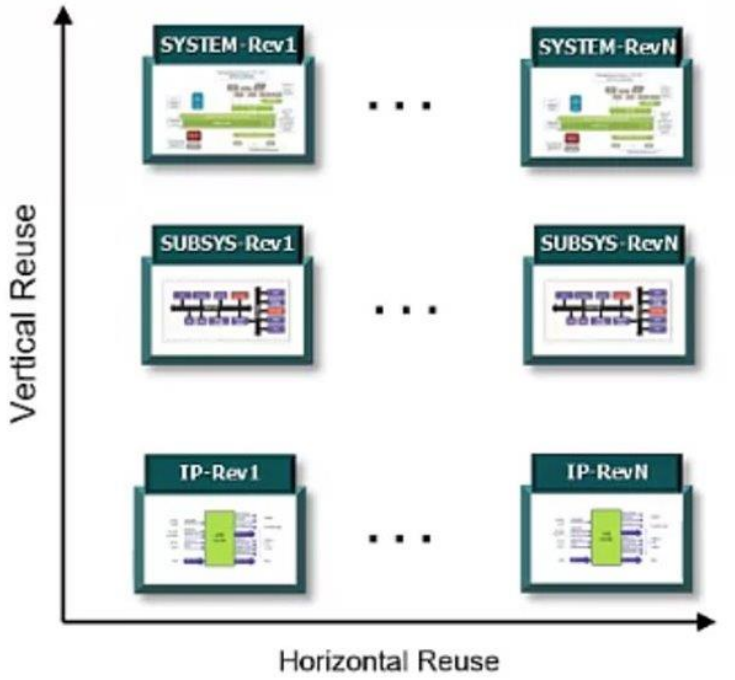
Reusability of tests

- Vertical Reuse
  - Reuse across system levels.
  - Creating in IP level and reuse in Subsystem & System levels.



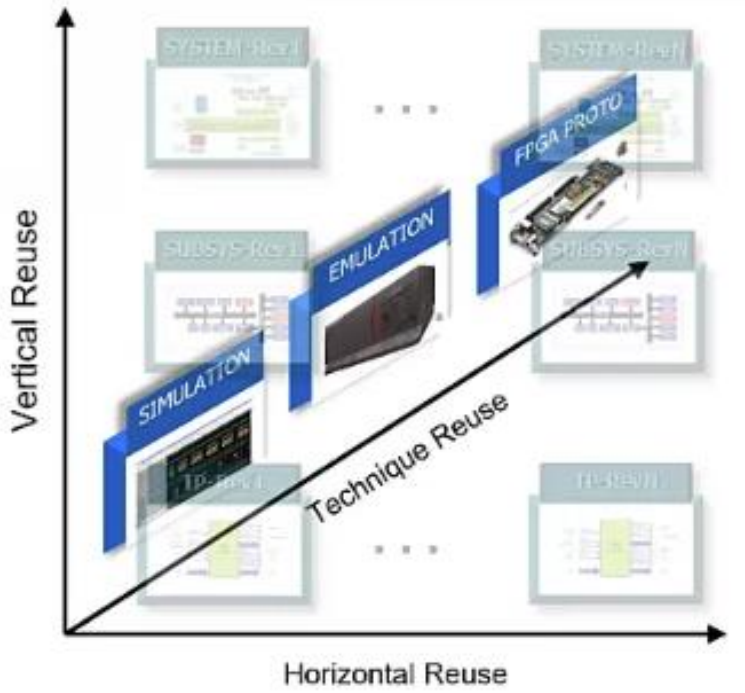
Reusability (cont.)

- Horizontal Reuse
  - Reuse across projects.
  - Develop tests for the first project, customize and reuse/regenerate in different projects.



Reusability (cont.)

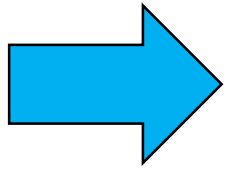
- Technique Reuse
  - Reuse same technique across different platforms.

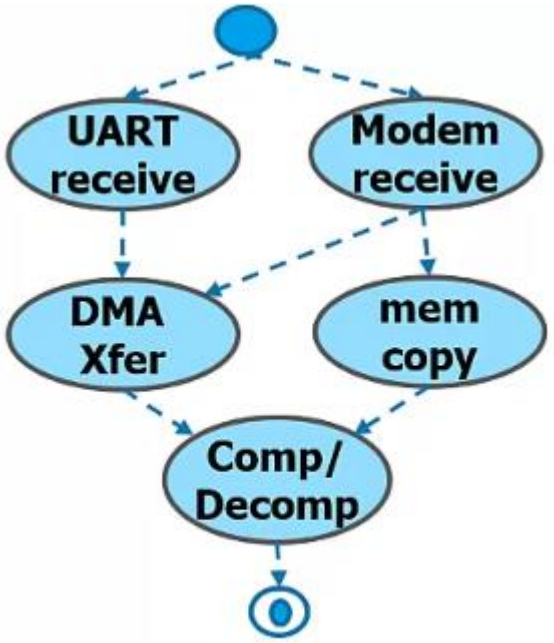


Stimulus at a Higher Level

- Move from transaction to Scenarios

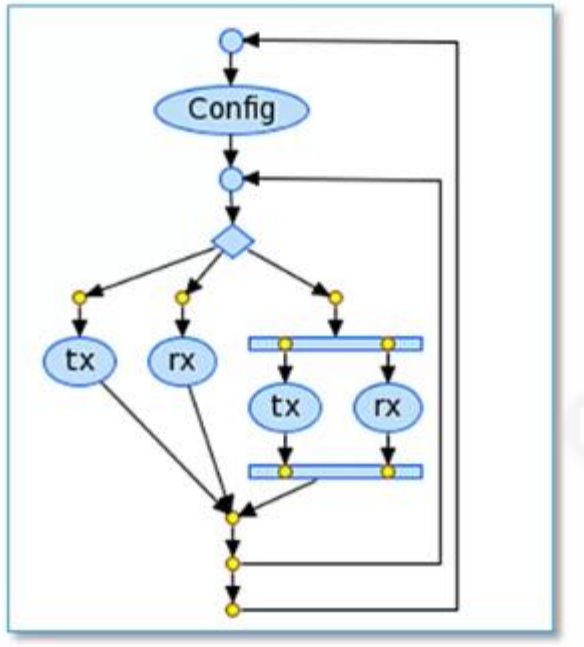
```
class my_seq extends uvm_sequence#(tr_t);
  `uvm_object_utils(my_seq)
  task body;
    for(int i=0; i< 10; i++) begin
      req = tr_t::type_id::create("tr");
      start_item(req);
      req.randomize() with {…};
      finish_item(req);
    end
  endtask
endclass
```





Graph-Based Portable Stimulus Description

- Capture data and control flow
  - Describes legal stimulus scenario space
  - Specify textually, visualize graphically
- Data and data relationships
  - Scalar, composite data types
- Control flow
  - Sequences of operations
  - Choices
  - Loops
- Control flow encapsulation
  - Composite larger scenarios from sub-elements
  - Constrain scenarios from above



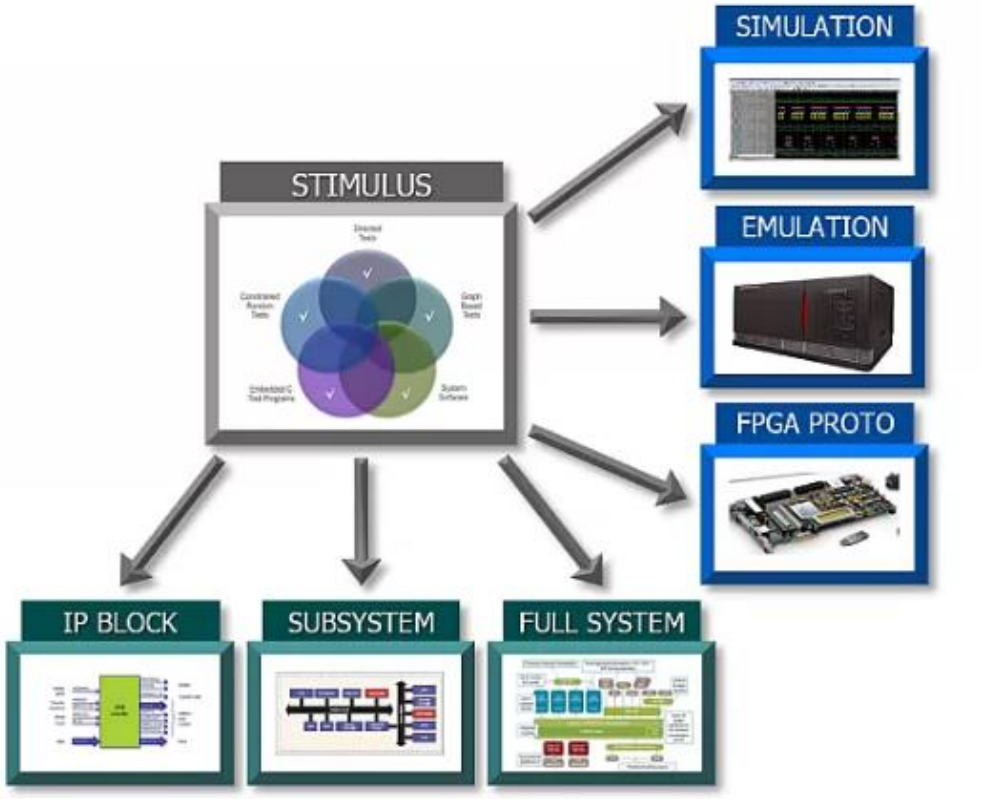
Source: Why Portable Stimulus?
https://verificationacademy.com/sessions/why-portable-stimulus/

Design-Implementation-Specific Tweaks

- Scenario specification defines test intent
- Realization may depend on target resources
  - Multiple CPUs?
  - How many DMA *(Direct memory access)* channels?
- Scenario must be customizable
  - OOP *(Object oriented programming)* inheritance and extension

Summary

- PSS is a single declarative reusable representation of test intent
  - Across levels of integration
  - In a variety of execution platforms
  - Across different configurations
- Focused on scenario-level interactions
- Portable Stimulus is not
  - A single forced level of abstraction
  - A replacement for all current testing activities
  - A monolithic representation



Exam Questions

- Why don't we use UVM to create scenario level test?
- Why using PSS can be beneficial for a company?
- Should you replace all your current testing activities with PSS?

Q&A

Sources & further reading

- Portable Test and Stimulus: https://verificationacademy.com/topics/portable-test-stimulus
- Portable Stimulus Basics: https://verificationacademy.com/courses/portable-stimulus-basics
- Portable Stimulus Concepts & Language Introduction: https://verificationacademy.com/sessions/concepts-and-language-introduction
- Reuse existing verification assets with the Portable Test and Stimulus Standard: https://www.edn.com/reuse-existing-verification-assets-with-the-portable-test-and-stimulus-standard/
- Portable Stimulus Working Group: https://www.accellera.org/activities/working-groups/portable-stimulus