Speakers Bio

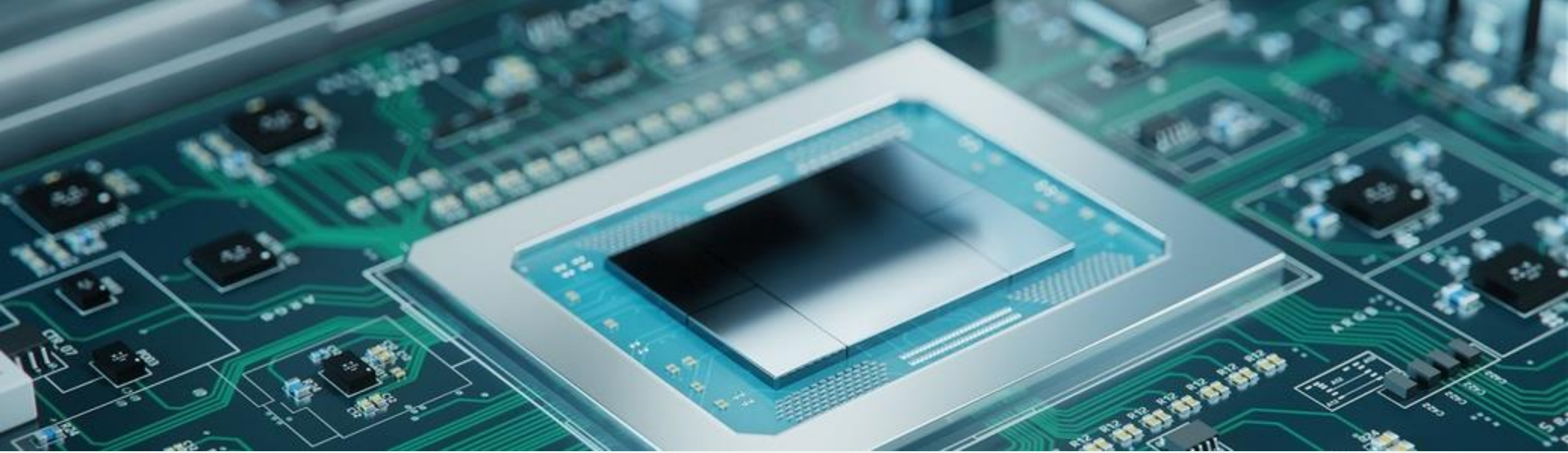
Raghavendra HM is working with Siemens EDA as Team Lead, with overall experience of 15 years in Functional Verification Domain, Prior to working with Siemens had worked with different Service and Product company as methodology development and support engineer in Design verification.

SIEMENS

Speakers Bio

Manoj Manu is an IIT graduate working with Siemens EDA as an Associate Director of VIP team with an overall experience of 18 years in VIP domain. His expertise includes USB, AMBA, Display, MIPI and Serial protocols.

SIEMENS



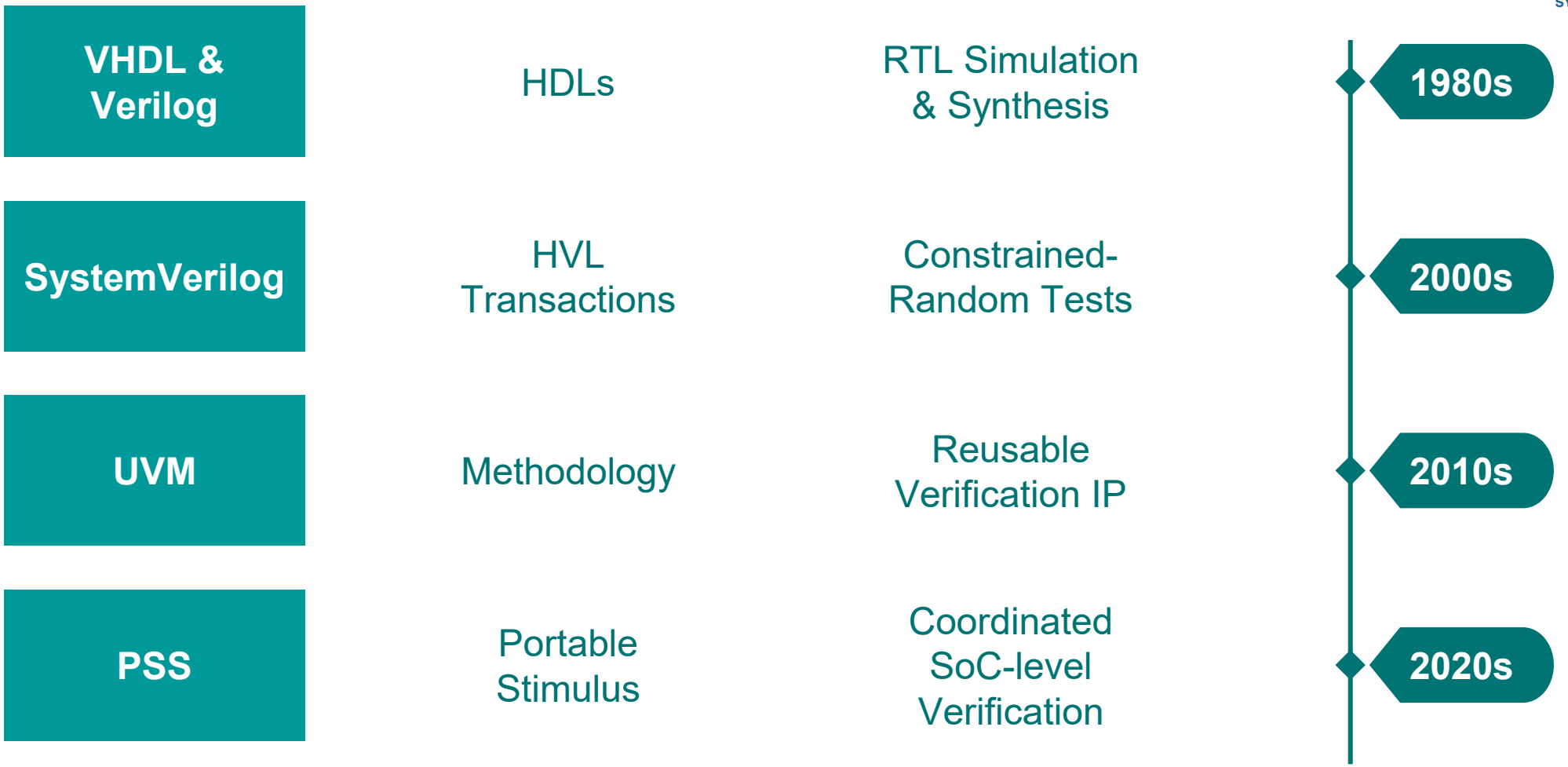
Portable Stimulus is the Next Big Thing. Here's Why

Raghavendra HM
Manoj Manu

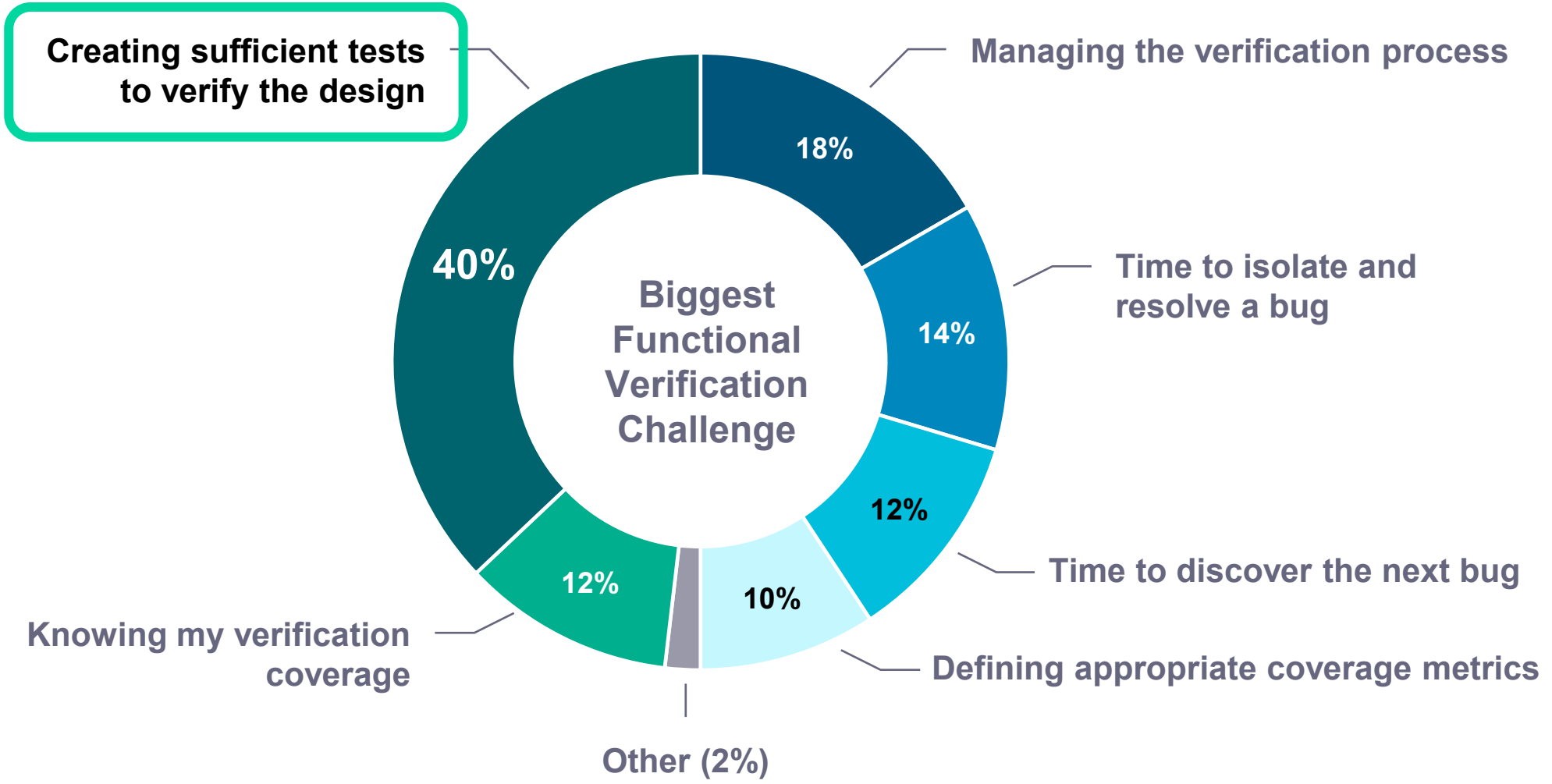
SIEMENS

EDA disrupters come along only every few decades … and define decades

Siemens is making PSS investment to enable the shift



Use PSS to address biggest functional verification challenge

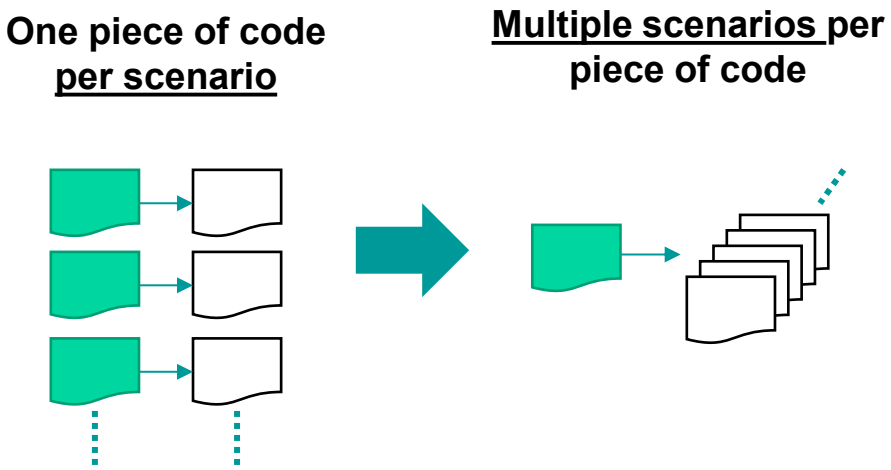


Source: Wilson Research Group and Mentor, A Siemens Business, 2020 Functional Verification Study, ASIC segment

SIEMENS

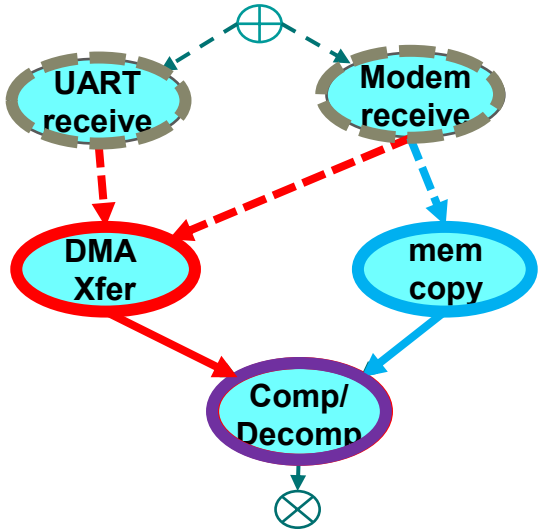
PSS Takes Constrained-Random to the Next Level

Directed testing → Constrained-Random



Scenarios define sets of behaviors

- Define critical verification intent
- Define rules to support critical intent



Simple PSS specification defines multiple scenarios

SIEMENS

Basic UVM Example

m::fill

m::mcpy

m::m2m

m::p2m

m::dump

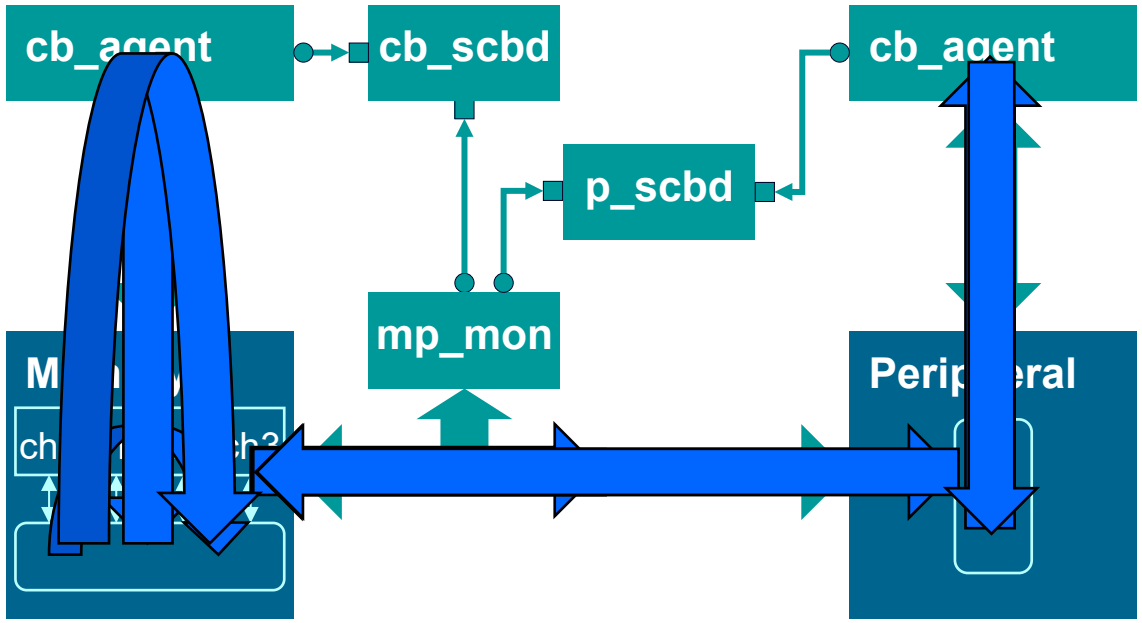
m::m2p

p::dump

p::p2m

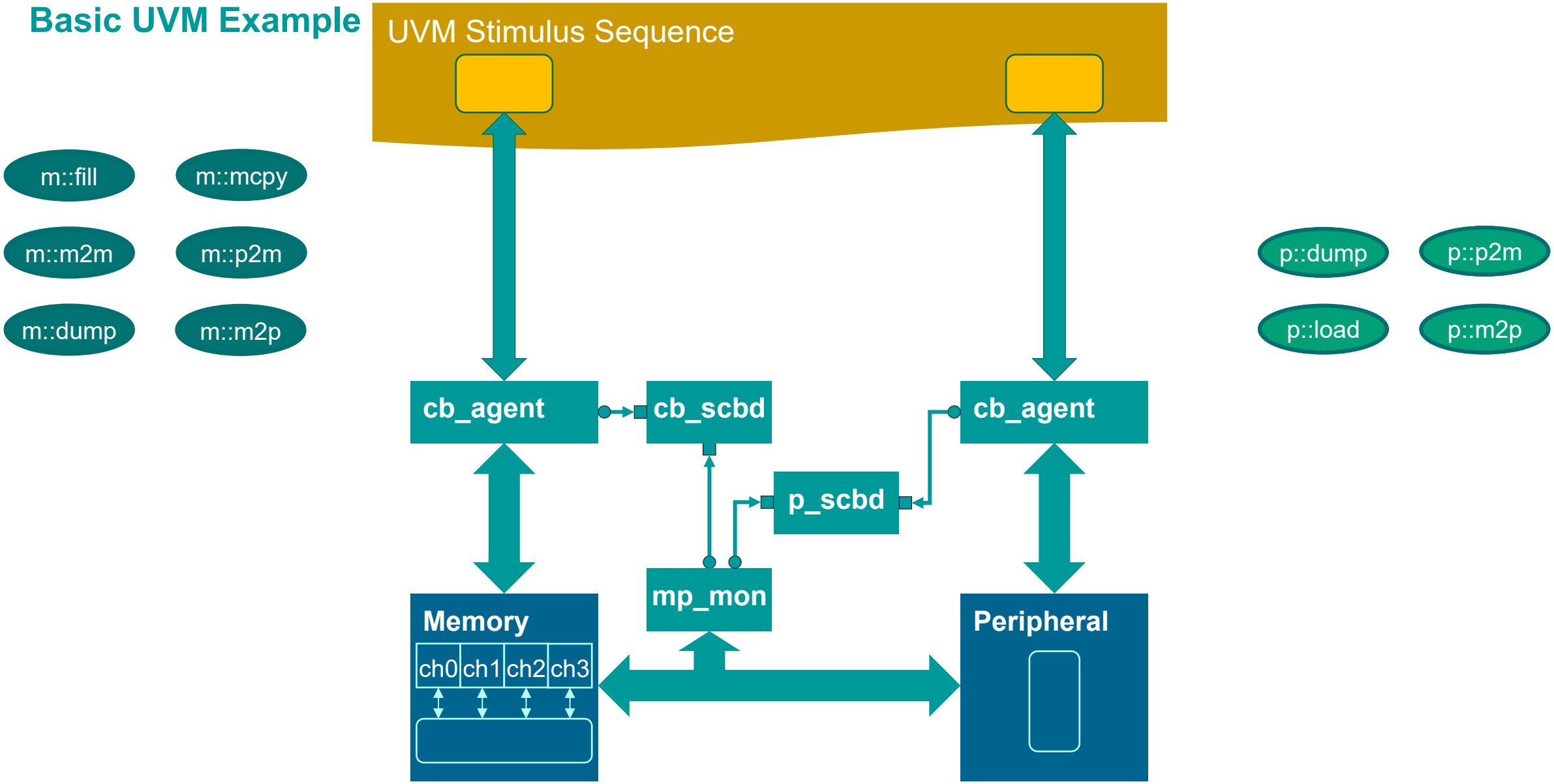
p::load

p::m2p



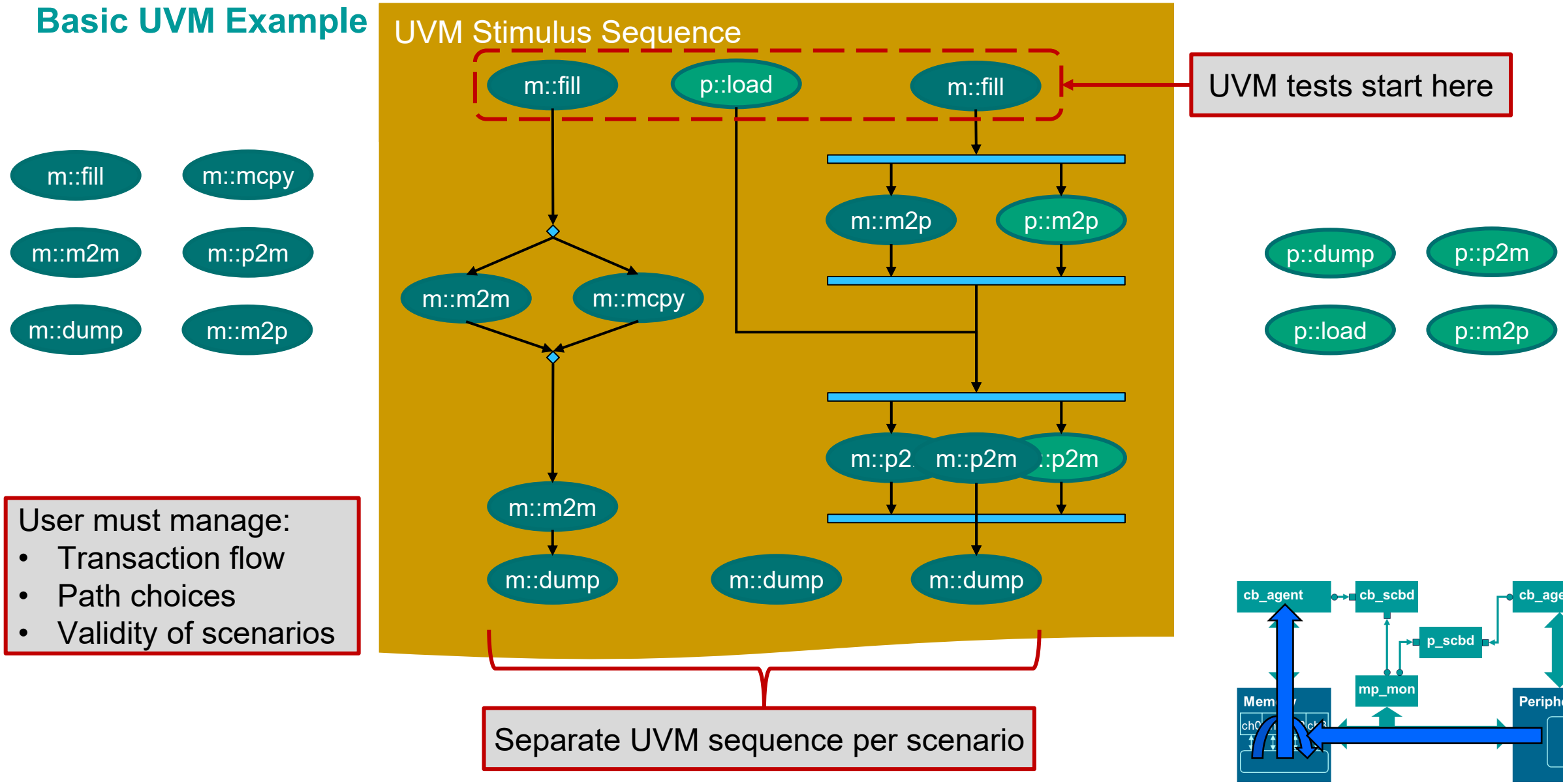
SIEMENS

Basic UVM Example



SIEMENS

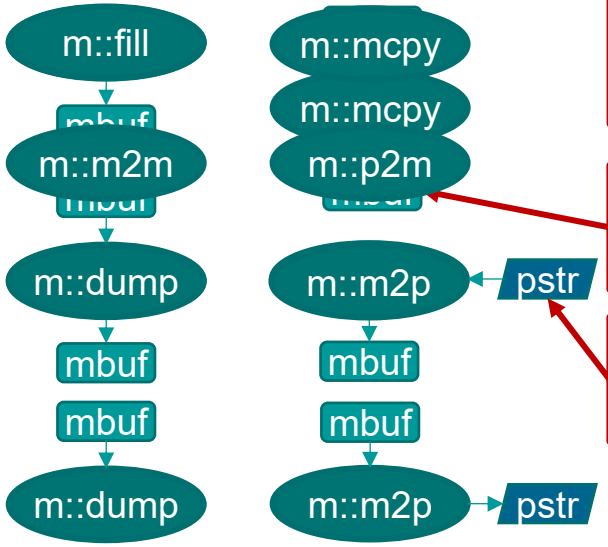
Basic UVM Example



SIEMENS

Basic PSS Concepts

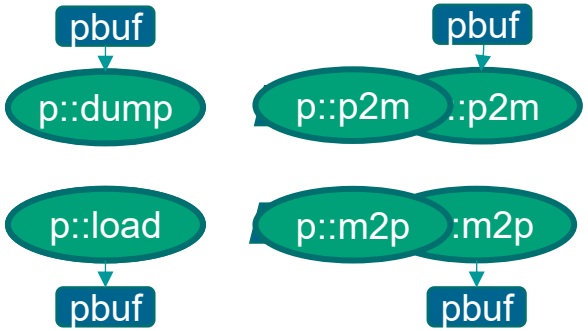
PSS starts by defining behaviors as *actions*



Actions share data through *flow objects*

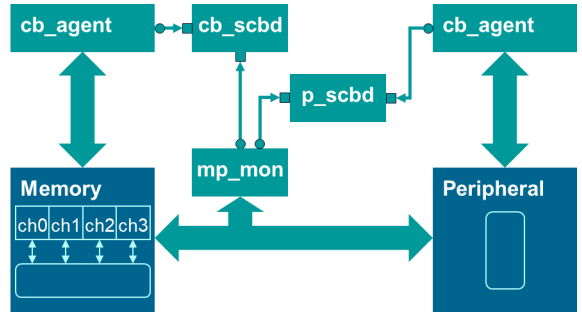
Buffer objects imply sequential behavior

Stream objects imply parallel behavior



Actions execute on a *component*

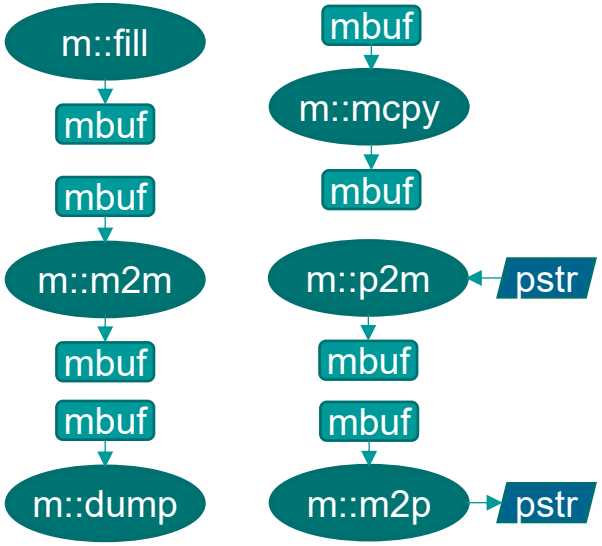
If an action inputs a flow object, then another action **must** provide it

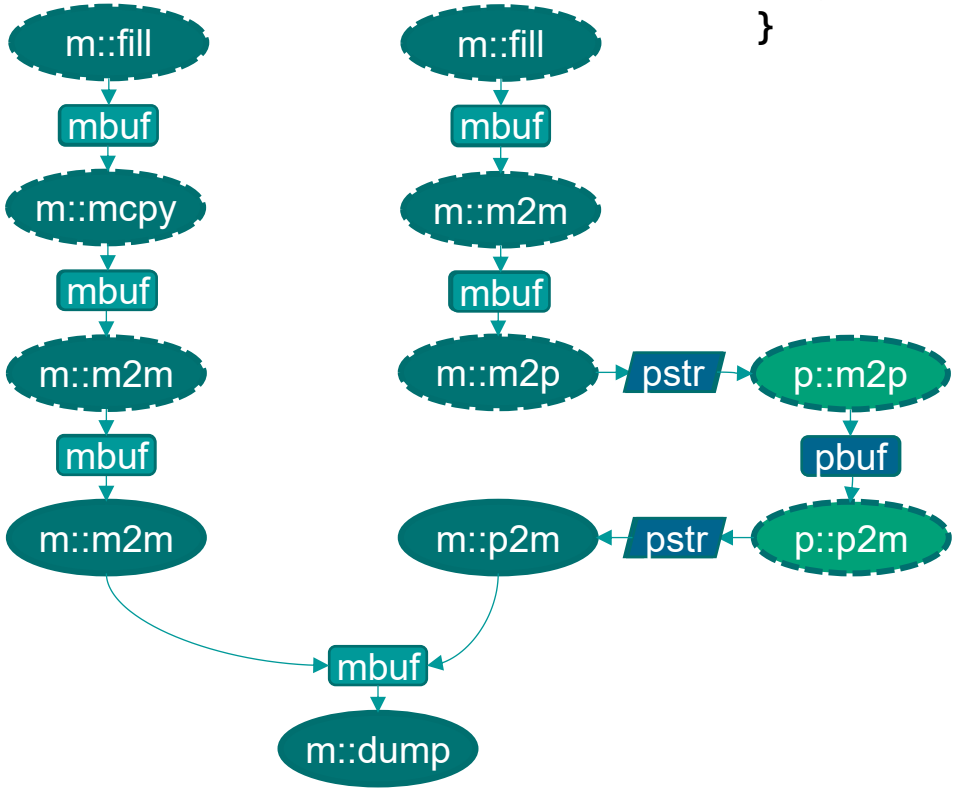


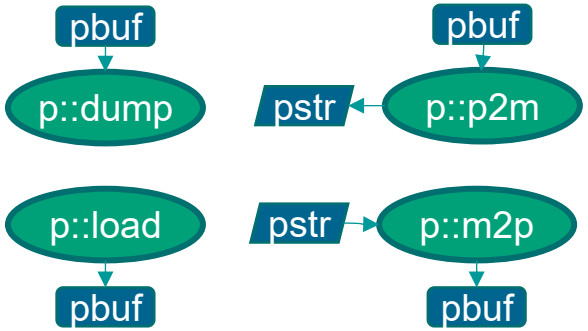
SIEMENS

Defining a PSS Scenario

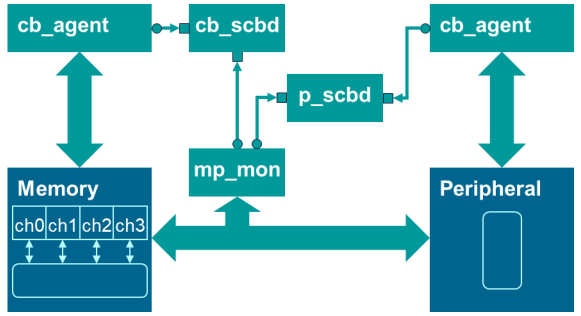
```
constraint mdump.ibuf.step == 2;
activity {
   repeat (4) {
      select {
         p2m with {obuf == mdump.ibuf;};
         m2m with {obuf == mdump.ibuf;};
      }
      mdump;
   }
}
```







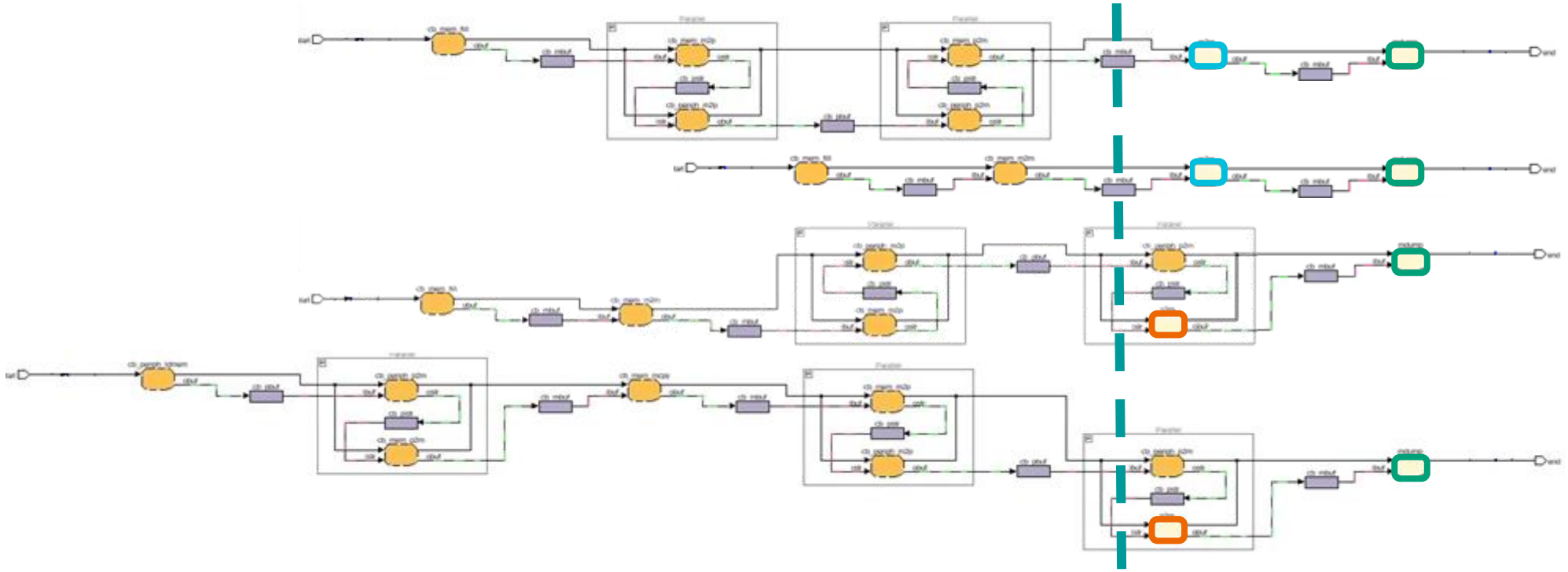
An *activity* defines the schedule of action execution



SIEMENS

Defining a PSS Scenario

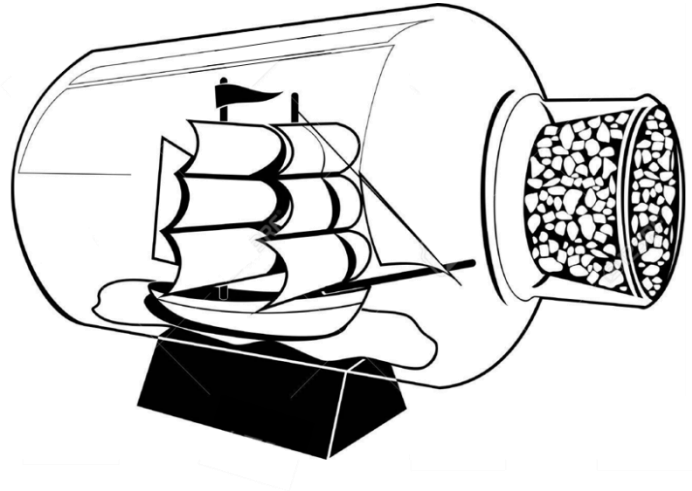
```
constraint mdump.ibuf.step == 2;
activity {
   repeat (4) {
      select {
         p2m with {obuf == mdump.ibuf;};
         m2m with {obuf == mdump.ibuf;};
      }
      mdump;
   }
}
```



What is a Portable Stimulus Model?

The Abstract Model

- *What* it does



The Realization Layer

- *How* it does what it does

SIEMENS

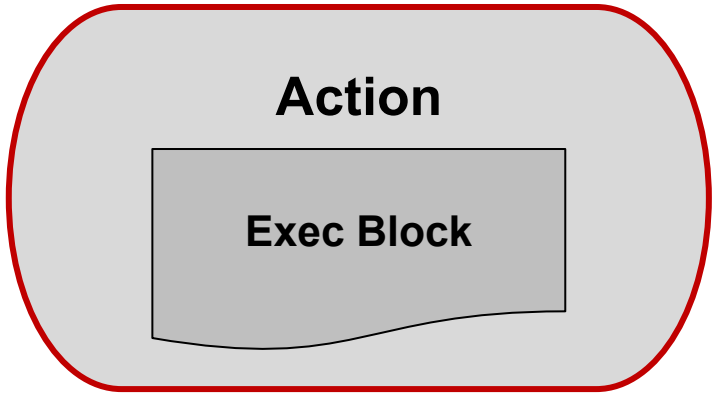
The Rubber Meets the Road

The Abstract Model must be implemented on different targets

Atomic Actions → target code

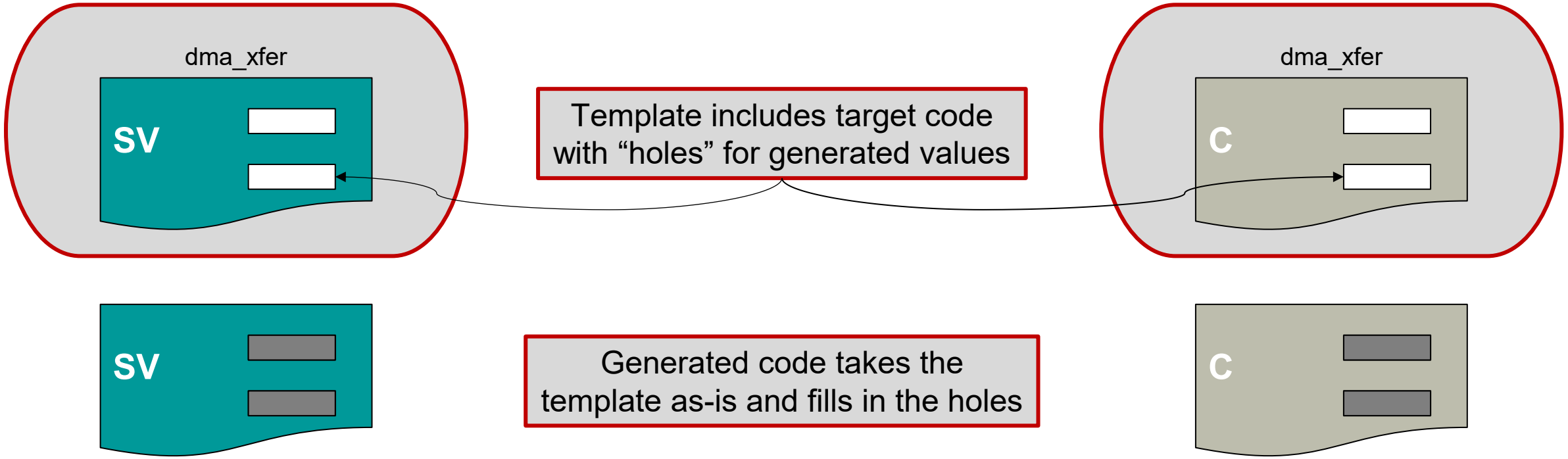
- Target code modeled in *exec* blocks

Generator assembles target code according to *Activity* schedule



SIEMENS

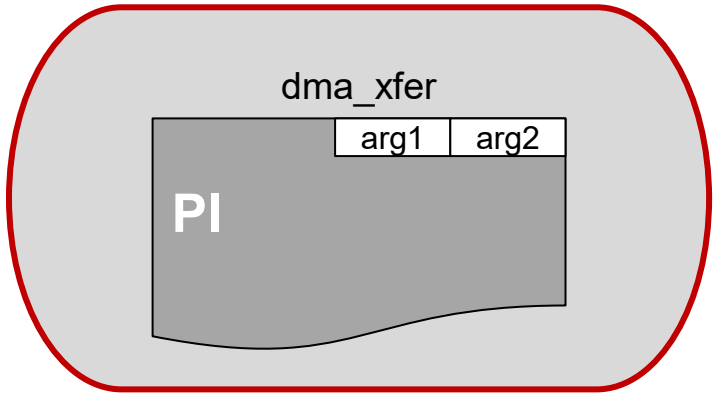
Target Templates Define 1:1 Mapping



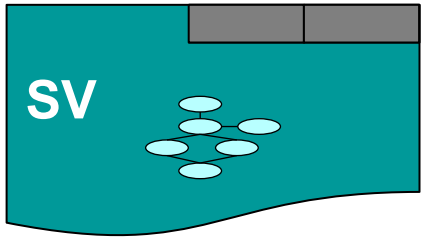
Target templates require a separate exec block for each target language
Managed in PSS via inheritance or extension

SIEMENS

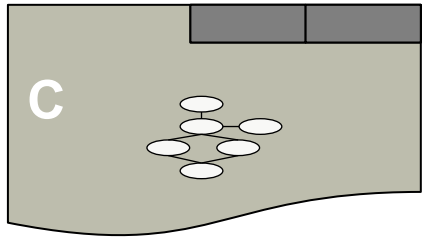
Procedural Interface Isolates Exec Block from Target Language



Procedural Interface defines functions that map to functions(/tasks) in the target language.
Values passed as arguments

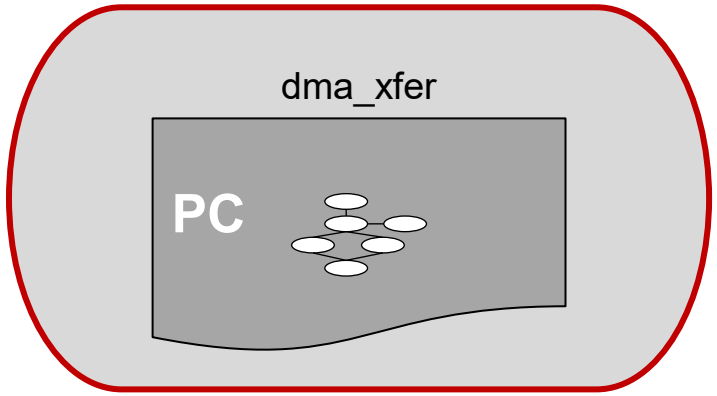


Still have to define possibly complex algorithms in each target language

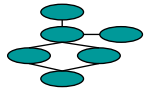


Procedural Interface lets you have one exec block per action type
Simplifies PSS code management

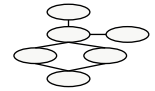
Procedural Constructs Move Complex Flow Control to PSS



Algorithm is specifed in exec block
Imported methods called accordingly

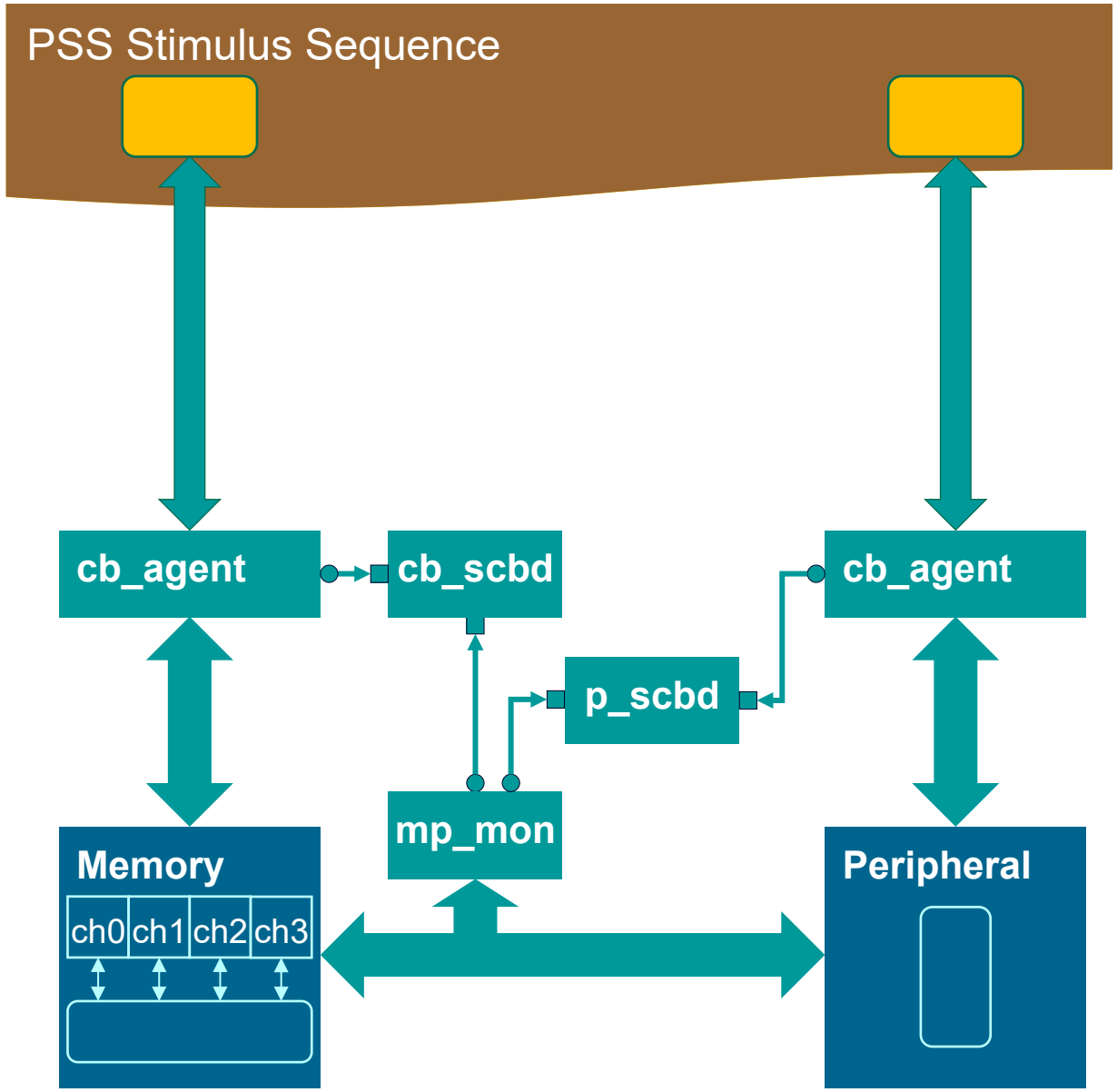
SV 

Complex control flow generated from PSS
Language-specific code is much simpler

C 

Procedural Constructs provide maximum reuse by simplifying the migration between languages

PSS Portability

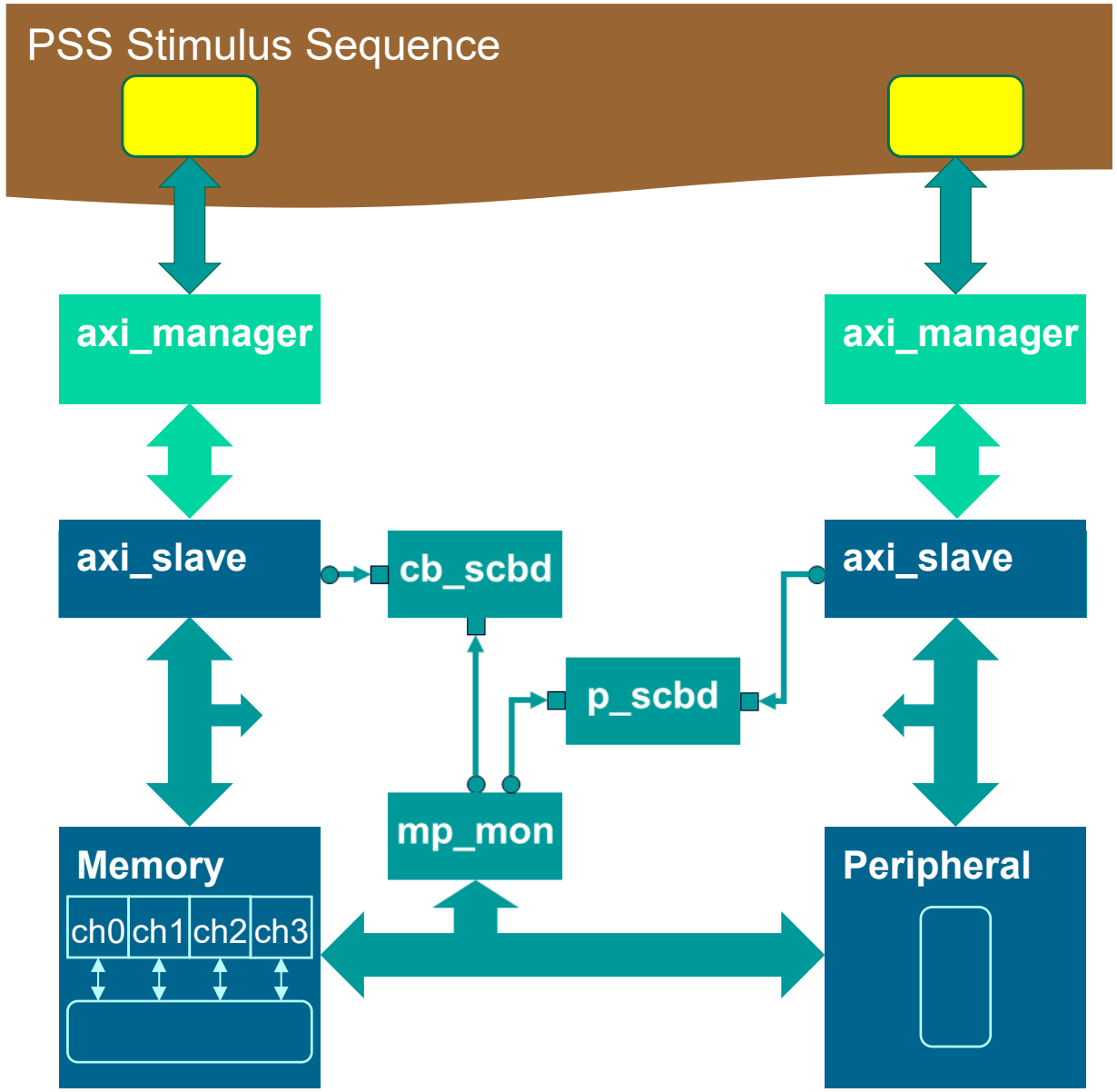


A PSS *package* can import the UVM API to be called from actions

PSS lets you create complex tests more easily than writing UVM virtual sequences

SIEMENS

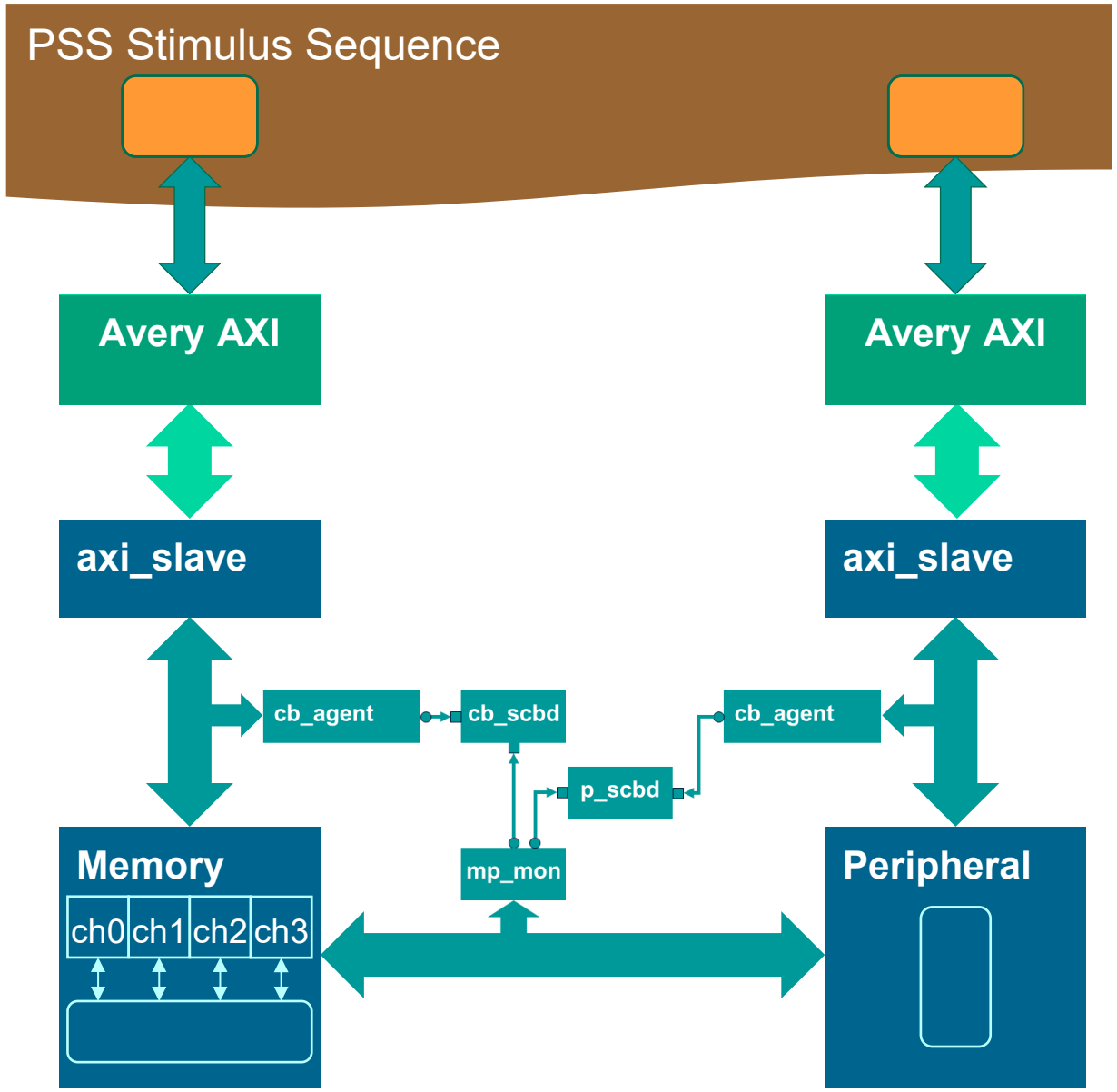
PSS Portability



The same PSS scenario uses the VIP API to implement the actions

SIEMENS

PSS Portability



If you use different VIP, **the same PSS scenario** uses the new VIP API to implement the same actions

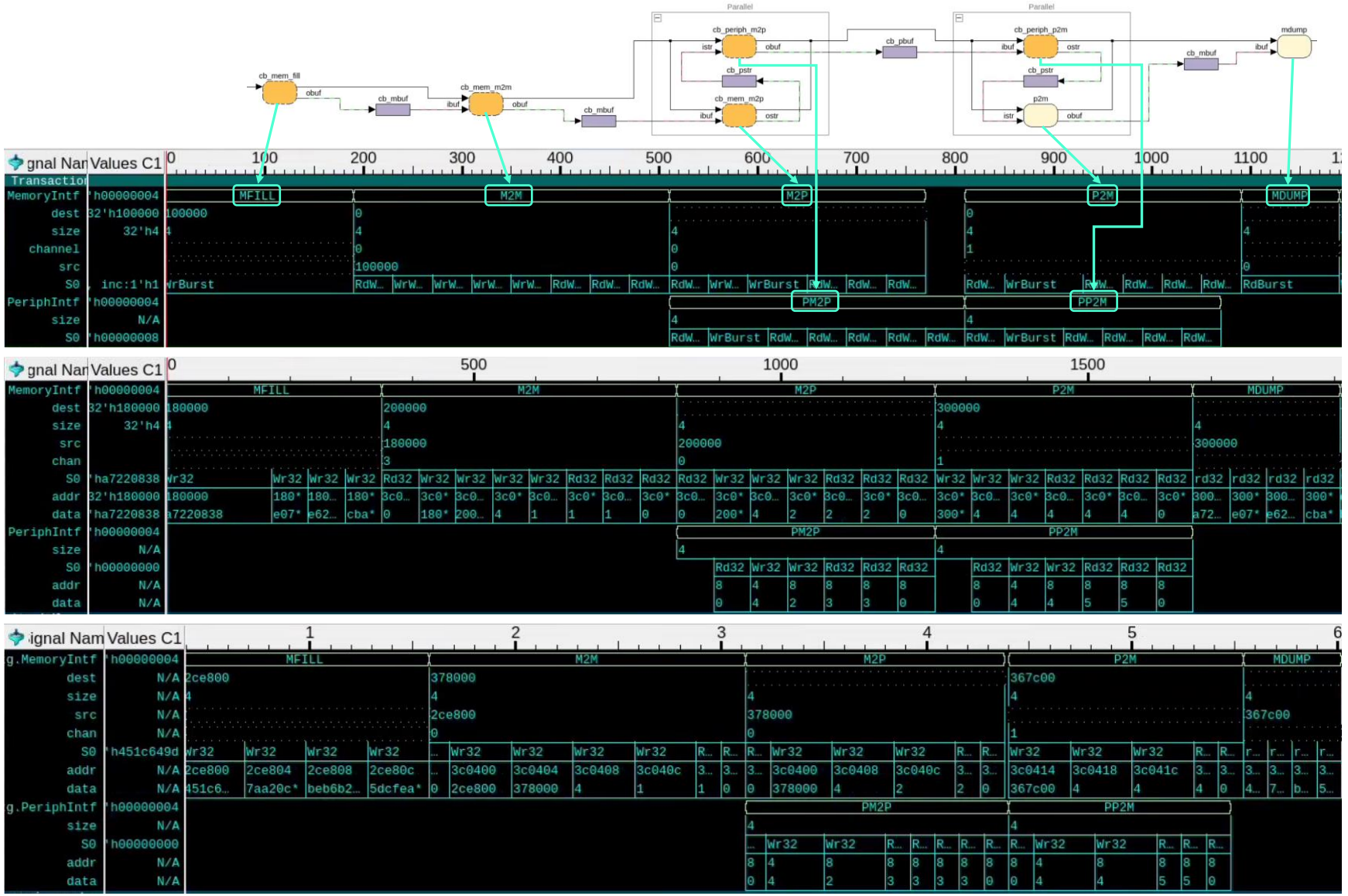
SIEMENS

PSS is Real

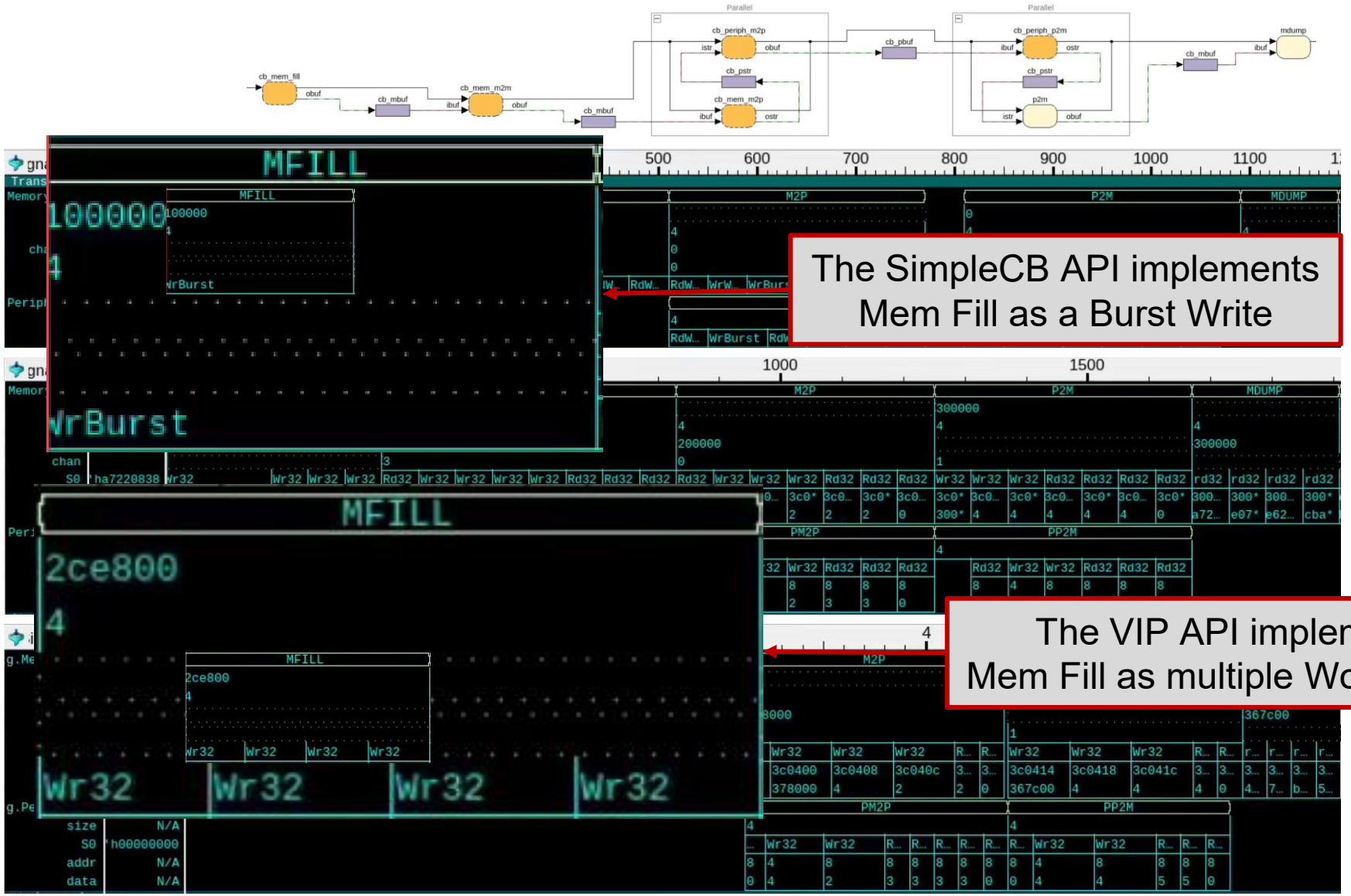
SimpleCB:

Questa VIP:

Avery VIP:



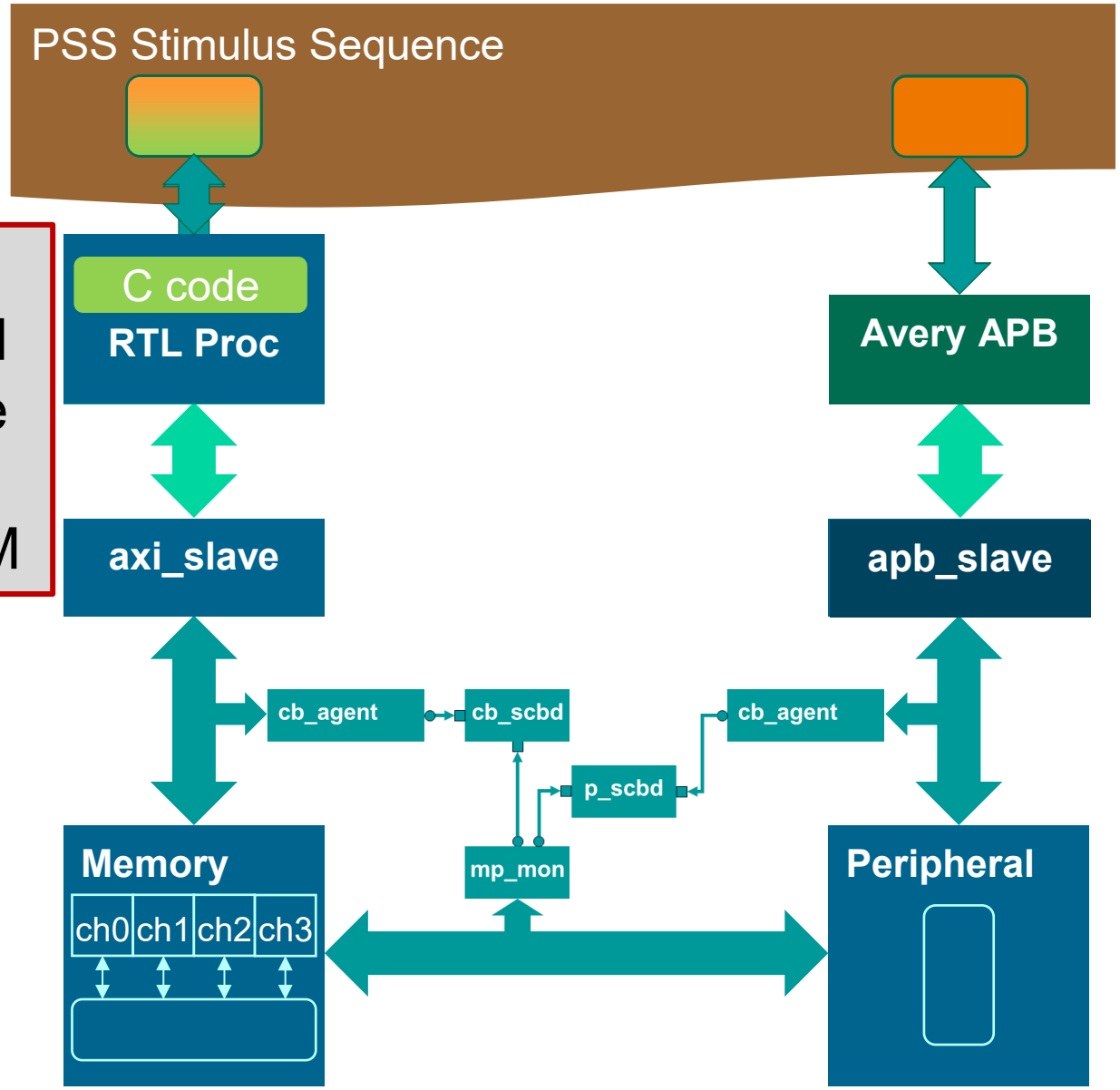
SIEMENS


The SimpleCB API implements
Mem Fill as a Burst Write
The VIP API implements
Mem Fill as multiple Word Writes

SIEMENS

More PSS Portability

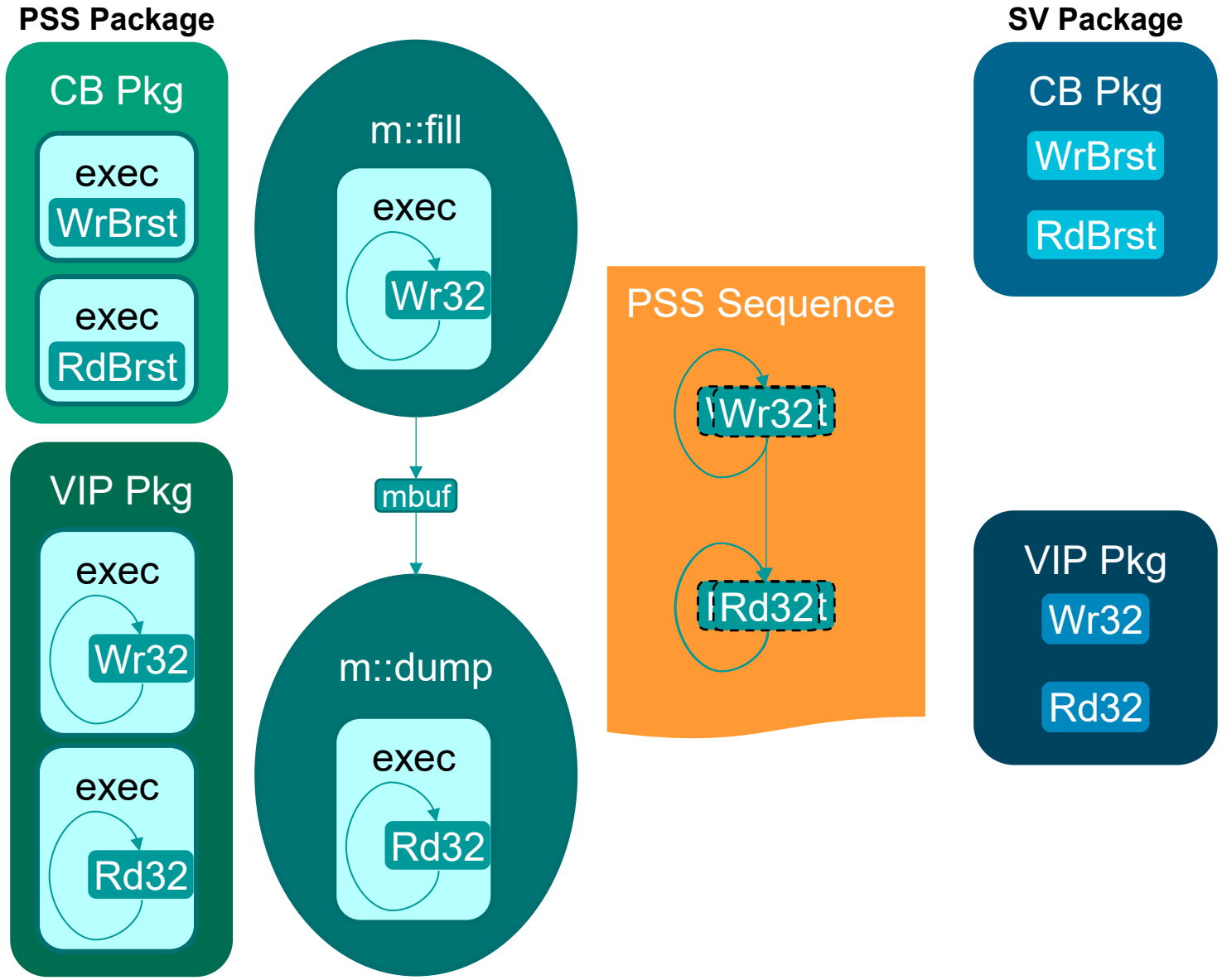
All or part of your test can be realized in C. The sequence will automatically coordinate C & UVM



SIEMENS

It’s All About the Packaging

- PSS scenario is design-specific
  - Actions define what the design does
- Actions implementations are VIP-Specific
  - Low-level actions and/or task/function calls provided from VIP PSS Package
- Generated PSS sequence calls imported VIP tasks/functions
  - Task/function implementations provided from VIP UVM/SV Package
- Change the VIP by changing the package(s)
- PSS scenario definition doesn’t change



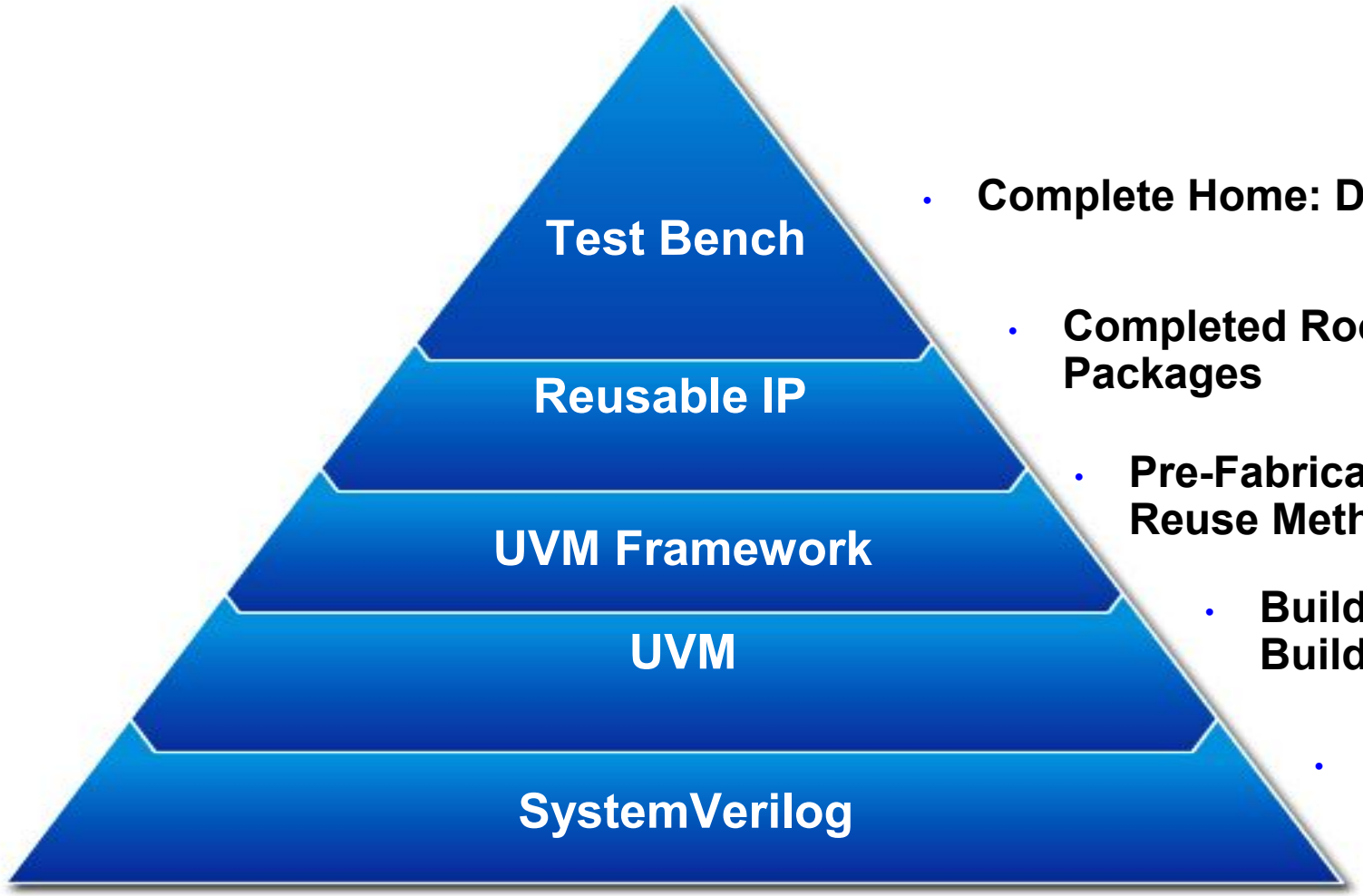
SIEMENS

Adding Automation with the UVM Framework

- UVM Code Generator
  - Interface, environment, test bench
- UVM Jumpstart
  - A layer on top of UVM to hide UVM details
  - Avoid common mistakes at all experience levels
- UVM Reuse methodology
  - Horizontal reuse of components across projects
  - Vertical reuse of environments from block to top
  - Platform reuse from simulation to emulation
  - Allows team to focus on verifying product features

SIEMENS

UVM Framework Reuse Structure



- Complete Home: DUT Specific Bench and Stimulus
- Completed Rooms: Interface and Environment Packages
- Pre-Fabricated Rooms: UVM Use Model and Reuse Methodology
- Building Supplies: Common Verification Building Blocks
- Raw Materials: Abstract Programming Language

SIEMENS

Siemens UVM Framework (UVMF) for Easier UVM

Making the transition to UVM possible

UVM Jumpstart

- Immediately productive while learning UVM
- Layer on top of UVM that hides UVM details
- Allows team to focus on verifying product features
- Promotes reuse increasing productivity

UVM Testbench Generators

- Code generators to create a UVM testbench in minutes
- Saves 3-4 weeks of effort on every project
- Decrease verification schedule up to 25% through infrastructure automation
- Seamlessly integrates PSS into UVM Test Environment

Helped over 40 companies adopt UVM

- 50% are FPGA customers
- Over half are in mil-aero industry
- Majority had little or no SystemVerilog experience
- Used in 100’s of projects

SIEMENS

What Benefits Do UVM Engineers Report?
From DVConUS PSS Tutorial

40%-60% effort reduction for new test benches (abstract model)

60%-80% effort reduction for IP revisions (**improved reuse**)

Great **reduction** in **boring repetitive** work

Reduction in **debug** of complex UVM control flows

Faster **time-to-bug** vs. UVM

Easier to provide **tests to other part of flows**

Flows as **executable documentation**

SIEMENS

Disclaimer

- © Siemens 2024

- Subject to changes and errors. The information given in this document only contains general descriptions and/or performance features which may not always specifically reflect those described, or which may undergo modification in the course of further development of the products. The requested performance features are binding only when they are expressly agreed upon in the concluded contract.

- All product designations may be trademarks or other rights of Siemens AG, its affiliated companies or other companies whose use by third parties for their own purposes could violate the rights of the respective owner.

SIEMENS

Contact

Manoj Manu

E-mail : manoj.manu@siemens.com

Raghavendra HM

E-mail : raghavendra.hm@siemens.com

SIEMENS