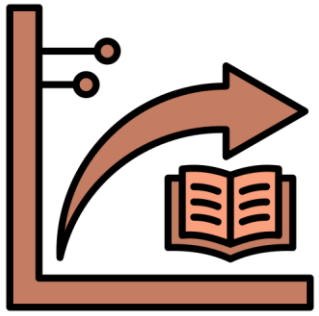
Fear Not! With LLMs, PSS Isn't Scary At All

Tom Fitzpatrick
Strategic Verification Architect
Siemens EDA

SIEMENS

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation sigda acm

PSS Adoption Challenges



Perceived Steep Learning Curve

Difficulty Justifying Upfront Investment

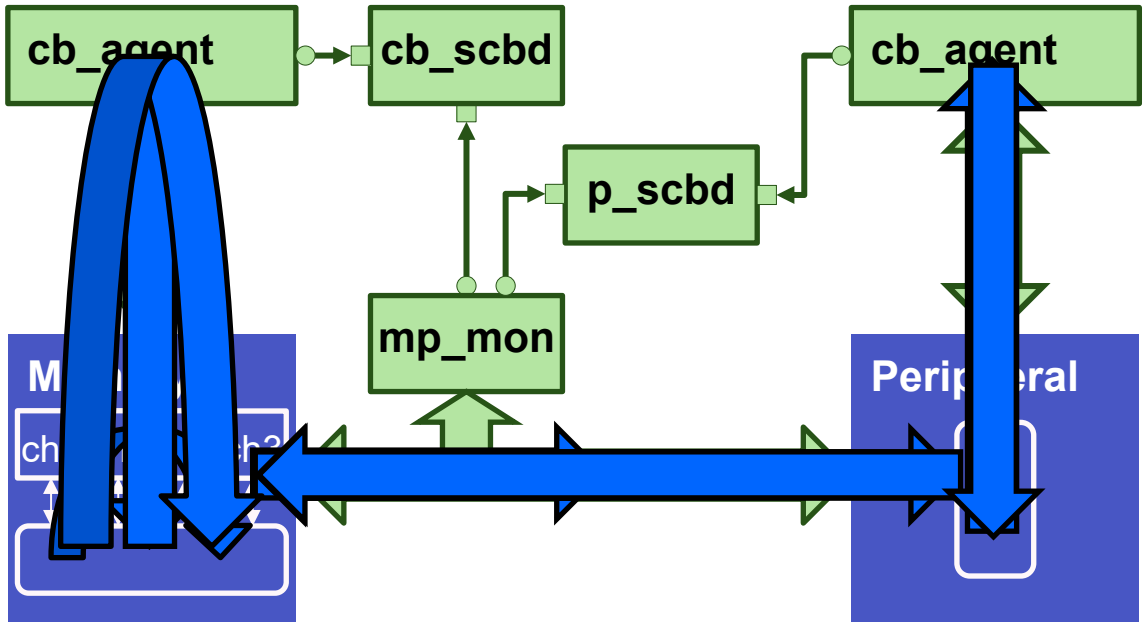
Resistance to Changing Workflows

Limited Ecosystem

Basic UVM Example

Memory Transactions

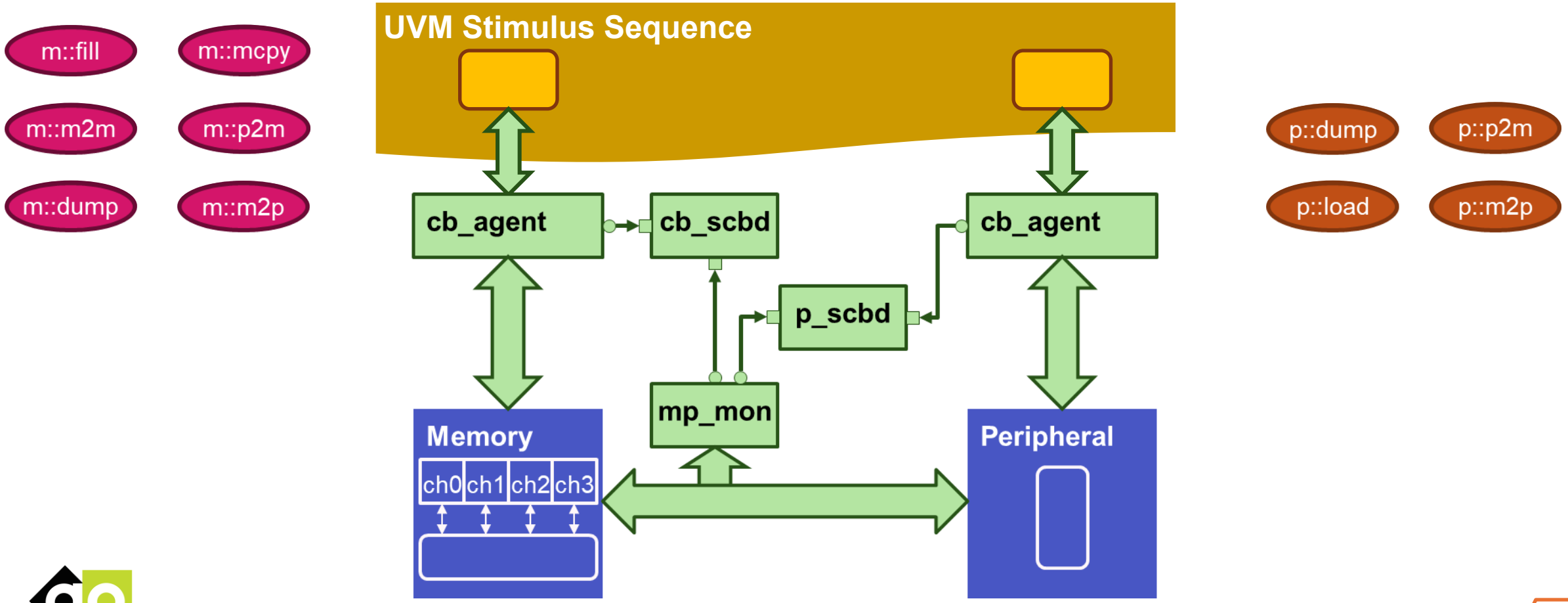
- m::fill
- m::mcpy
- m::m2m
- m::p2m
- m::dump
- m::m2p



Peripheral Transactions

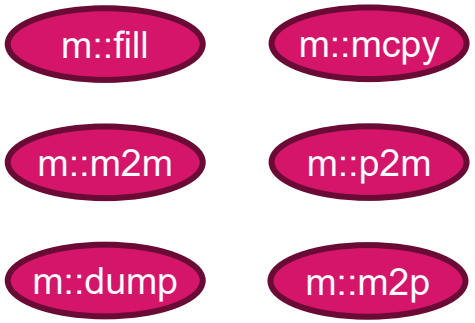
- p::dump
- p::p2m
- p::load
- p::m2p

Basic UVM Example

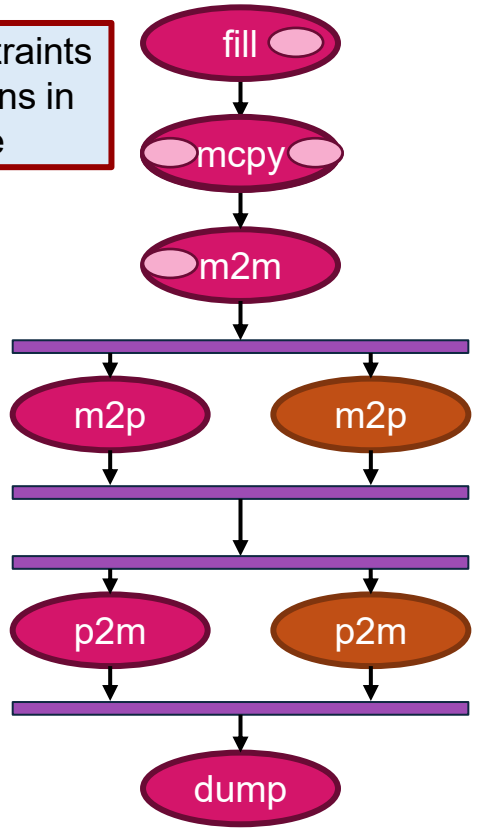


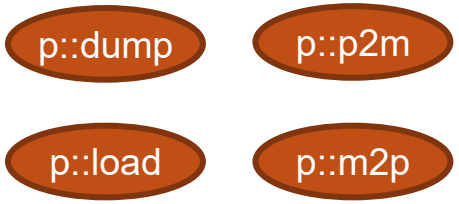
UVM is Based on Transactions

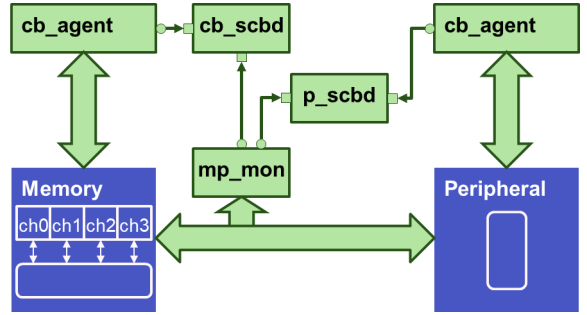
A UVM transaction abstracts ***information*** that must be ***acted upon***



User defines constraints between operations in the sequence

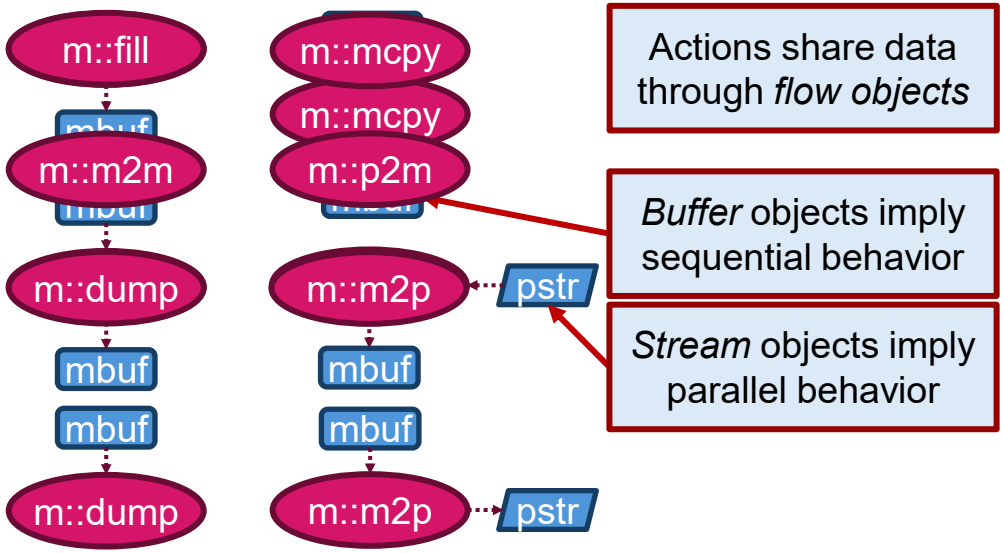






Transactions vs Actions

PSS starts by defining behaviors as *actions*

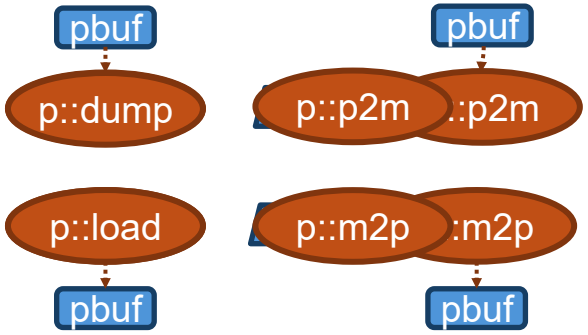


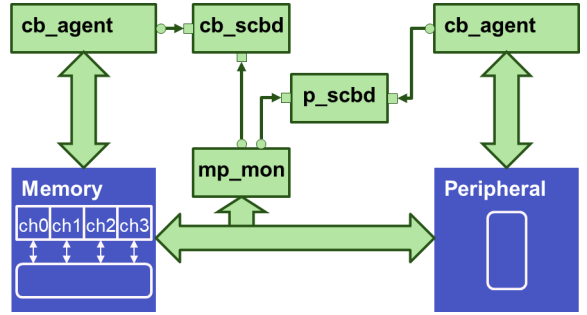
Actions share data through *flow objects*

Buffer objects imply sequential behavior

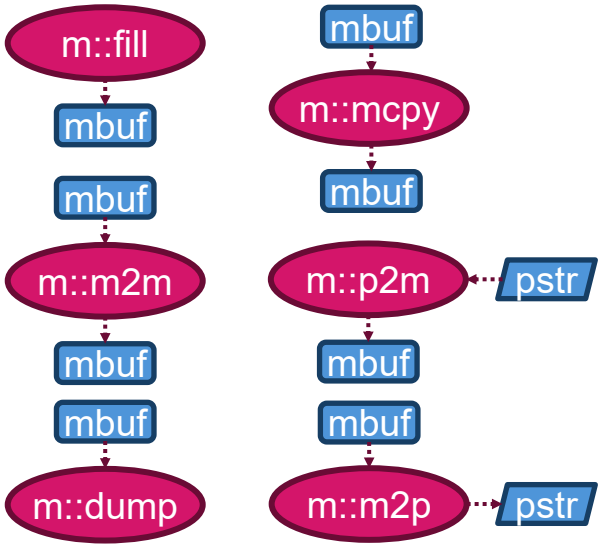
Stream objects imply parallel behavior

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

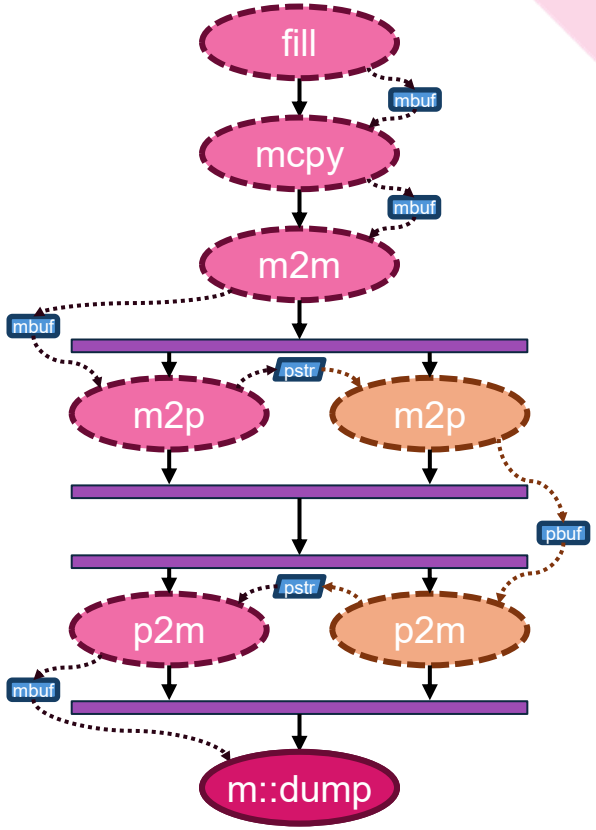




Transactions vs Actions



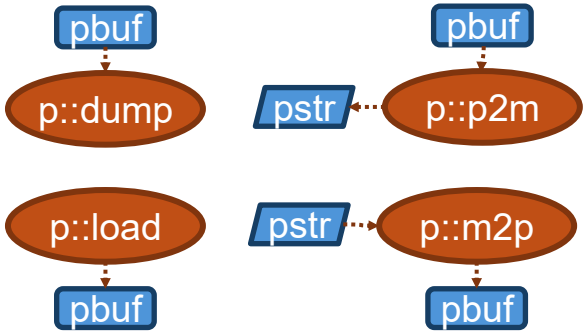
If an action inputs a flow object, then another action **must** provide it

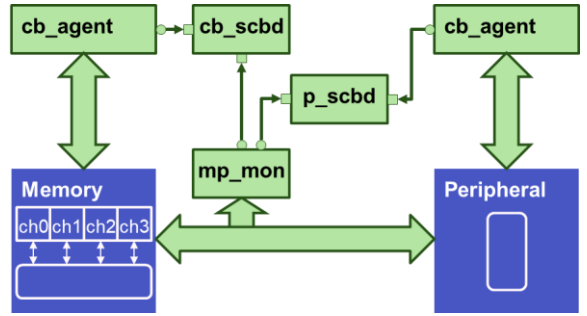


Constraints between transactions are built into the actions

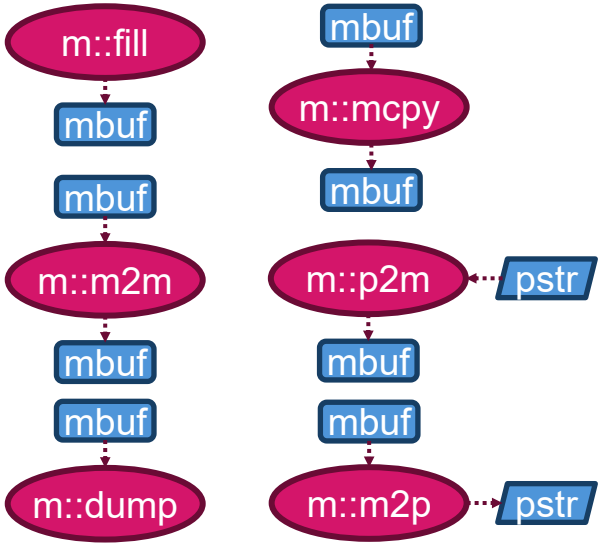
PSS ***actions*** abstract ***behaviors***

PSS Automates scenario building

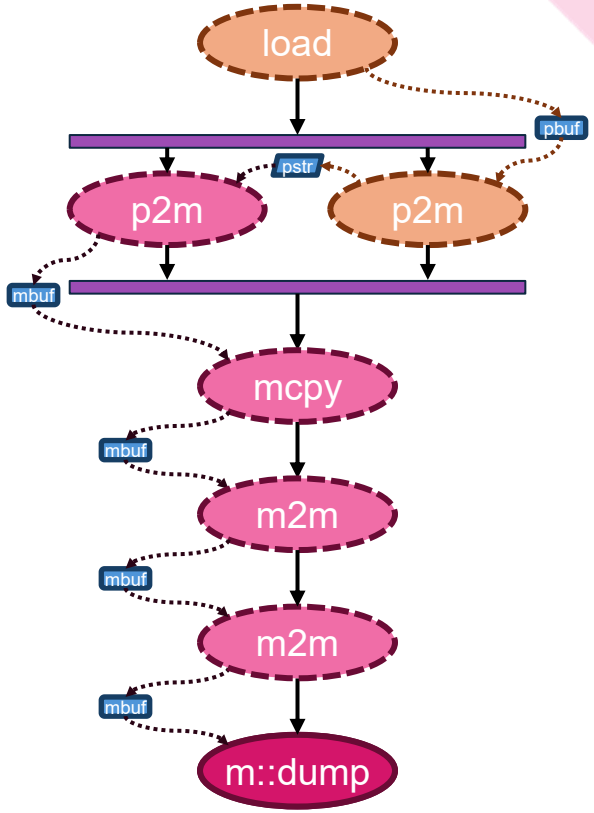




Transactions vs Actions

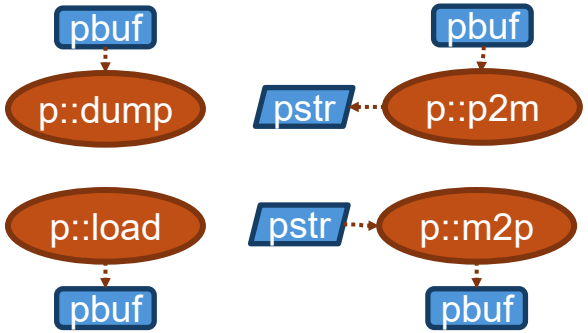


If an action inputs a flow object, then another action **must** provide it

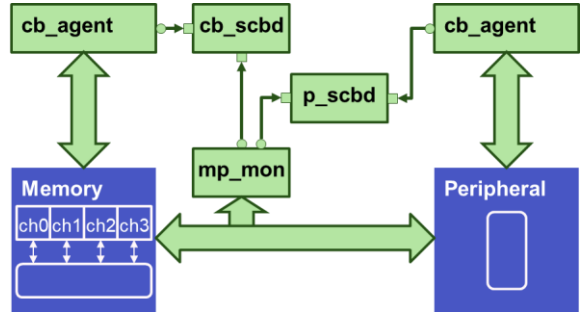


PSS ***actions*** abstract ***behaviors***

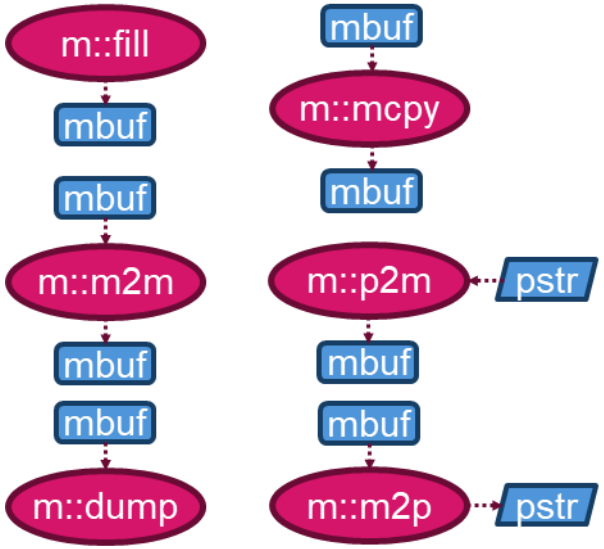
PSS Automates scenario building

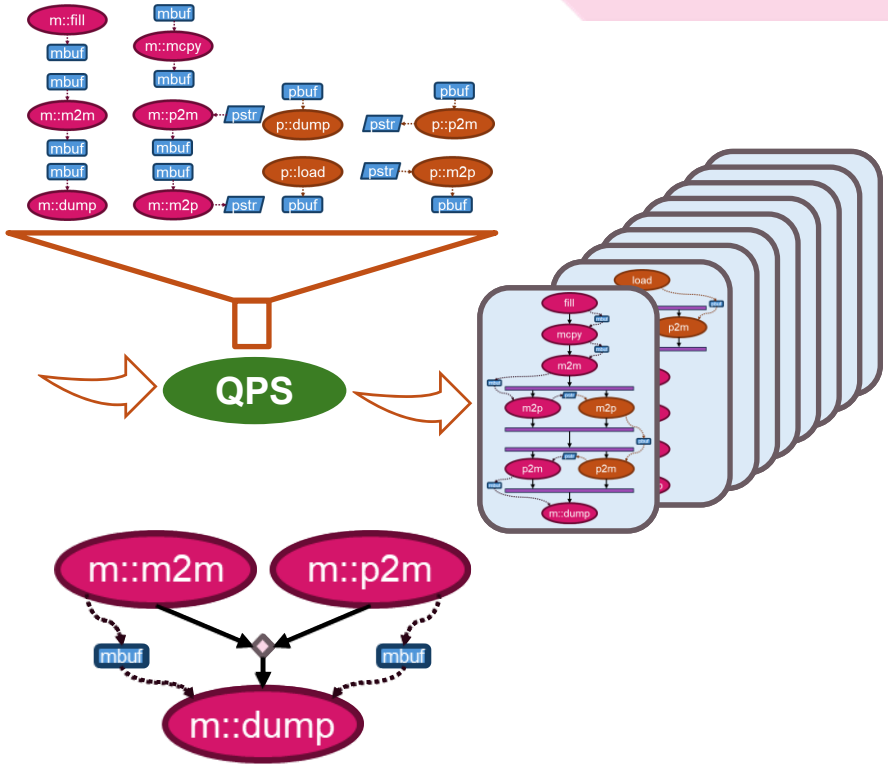


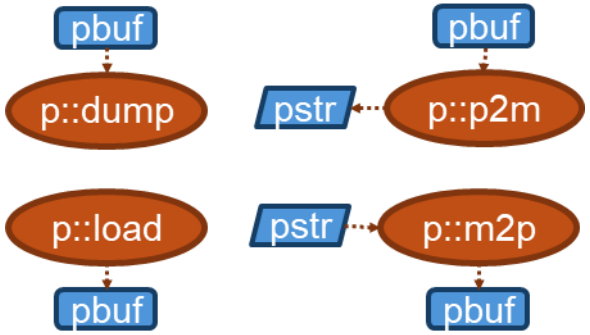
Constraints between transactions are built into the actions



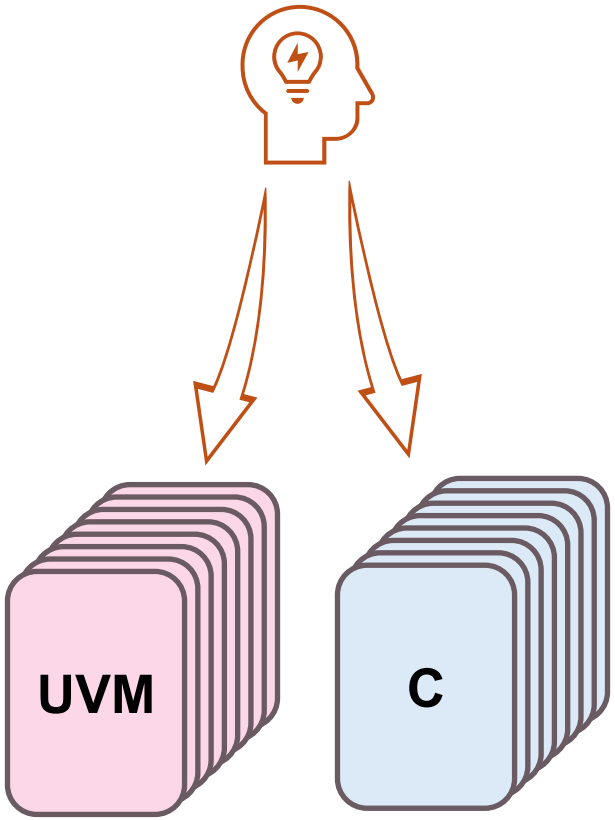
PSS Automates UVM Sequence Writing C Tests Too

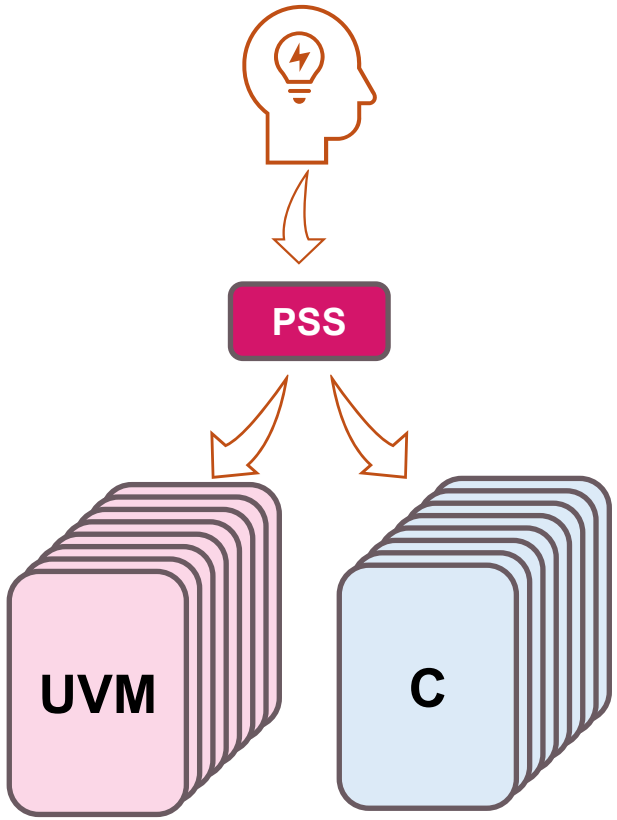




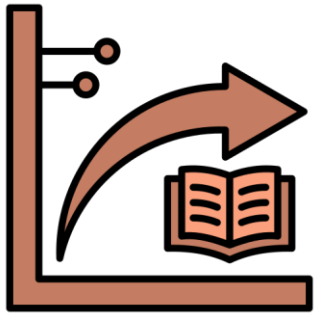


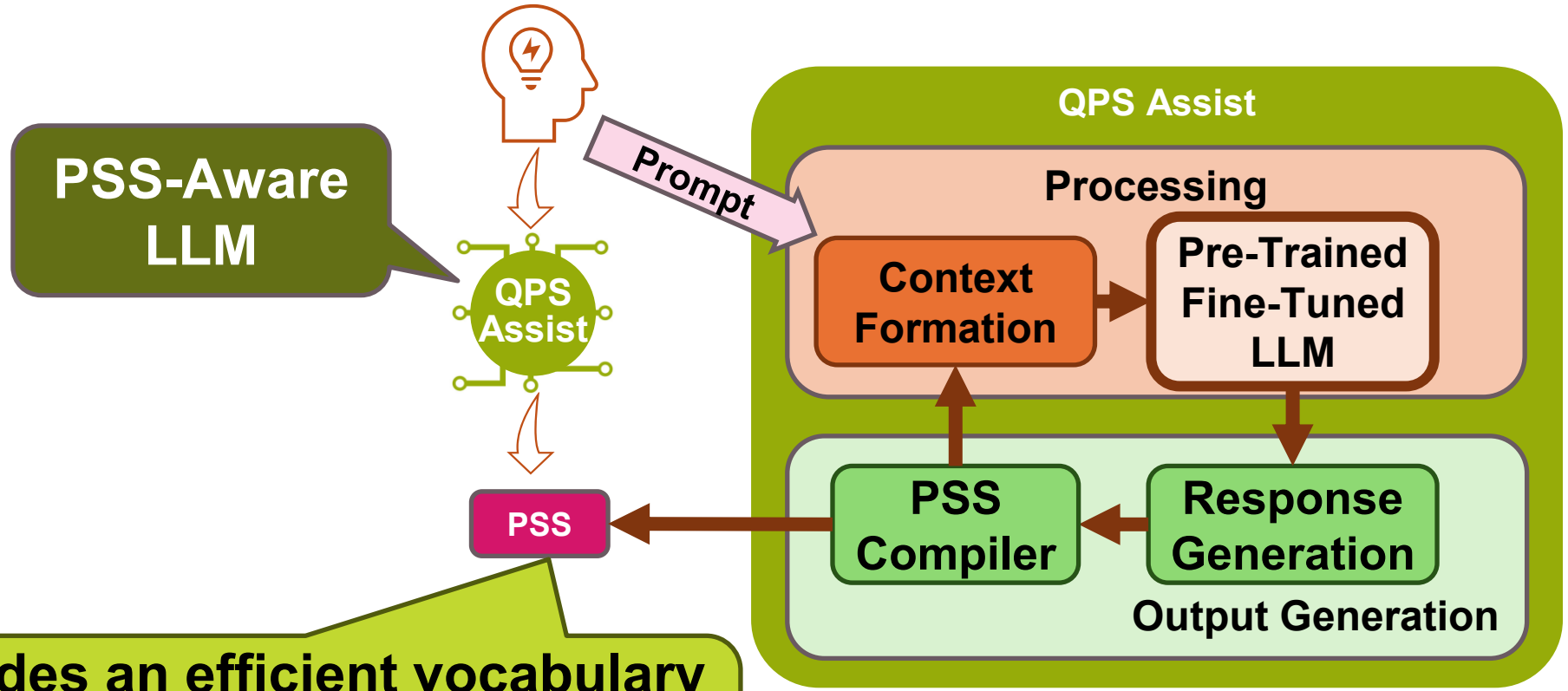
“P” = “Portable”





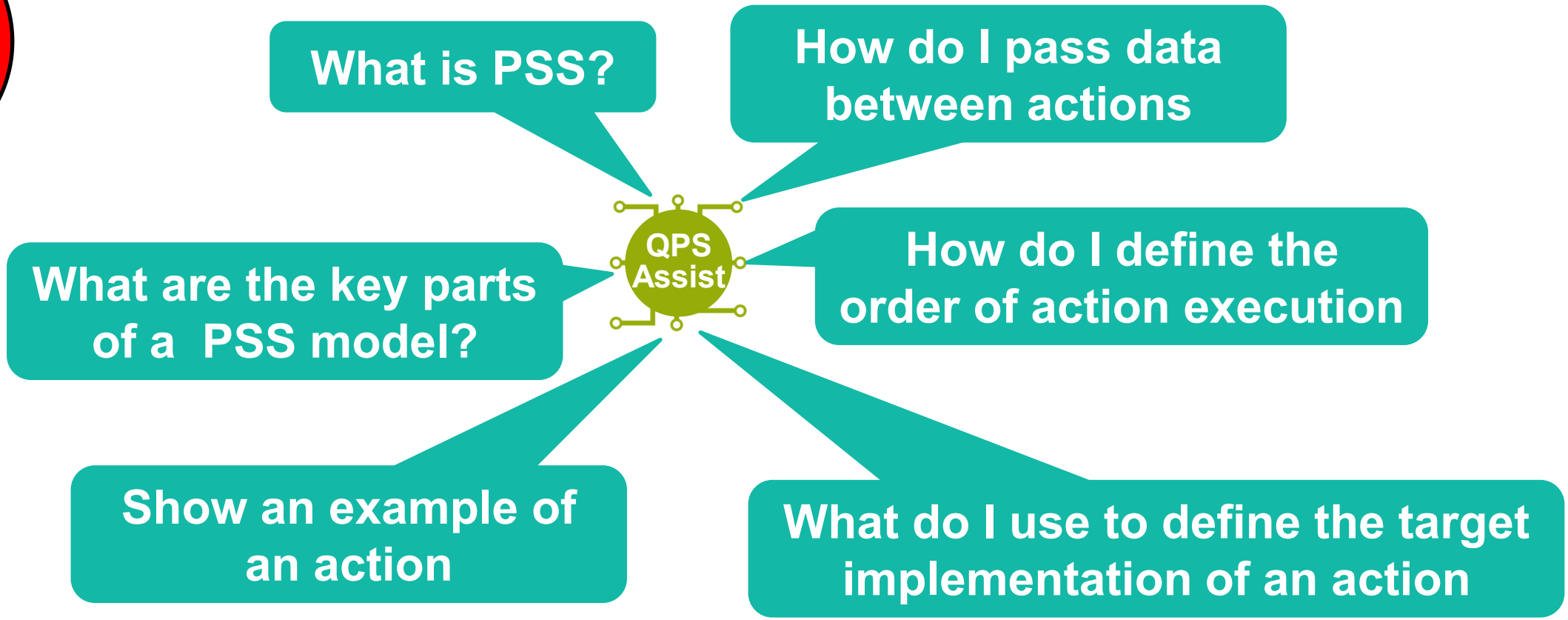
Introducing
Questa One Portable Stimulus Assist





PSS provides an efficient vocabulary for the user to compose prompts and for the LLM to provide answers

Portable Stimulus Assist as Tutor



Portable Stimulus Assist as Partner

define a buffer to model dma transactions

```
buffer mbuf {
  rand bit[32] src;
  rand bit[32] dst;
  rand bit[32] size;
}
```

orm a dma transfer.

```
action m2m {
  input mbuf in_buff;
```

define actions to fill and dump the memory

```
action fill {
  output mbuf out_buff;
}
```

ff.size == in_buff.size;

in the `dma_pkg` package and includes:

the source address.

show a test action that does m2m followed by dump

```
action test_m2m {
  fill fill;
  m2m m2m1;
  dump dump;

  activity {
    m2m1;
    dump;
  }
}
```

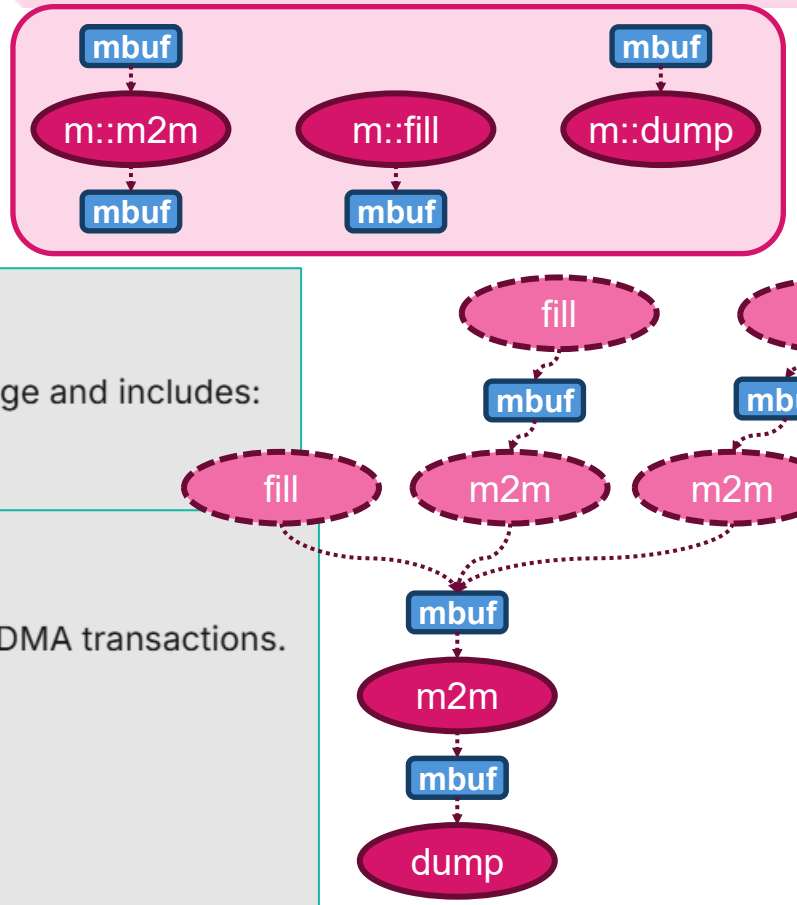
ntains the `mbuf` pool and actions related to DMA transactions.

emory-to-memory transfer with:

representing the source buffer.

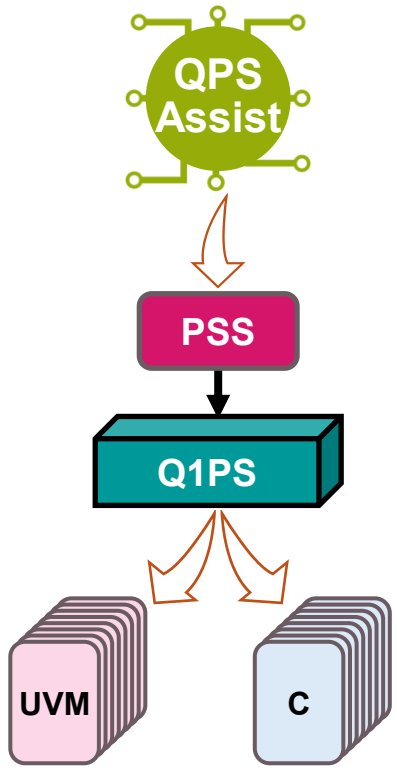
buf representing the destination buffer.

A constraint ensuring the size of the data remains consistent between input and output.

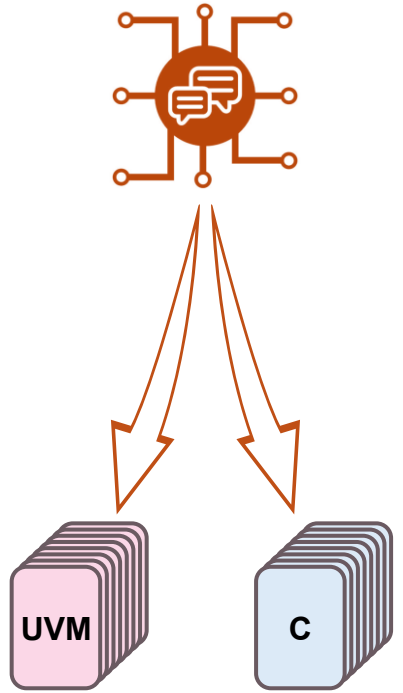


LLM➔PSS vs LLM➔UVM/C

$$Cost_{PSS} = Tokens_{model} + 0_{generation}$$



$$Cost_{Direct} = \sum_{i=1}^{n} (Tokens_{UVM_i} + Tokens_{C_i})$$



Mohamed Moselhy

Technical Architect
Siemens EDA

Sarah Hesham

Machine Learning Engineer
Siemens EDA

Mohamed Nafea

QA Technical Lead
Siemens EDA

Waseem Raslan

AI/ML Technical Director
Siemens EDA

Dan Yu

AI/ML Solutions Manager
Siemens EDA

AI

Security

Systems

EDA

Design

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation

sigda acm