A Unified Framework for Multilanguage Verification IPs Integration

Surinder Sood, Selvakumar Krishnamoorthy, Gaurav Jalan
SmartPlay Technologies

accellera SYSTEMS INITIATIVE™

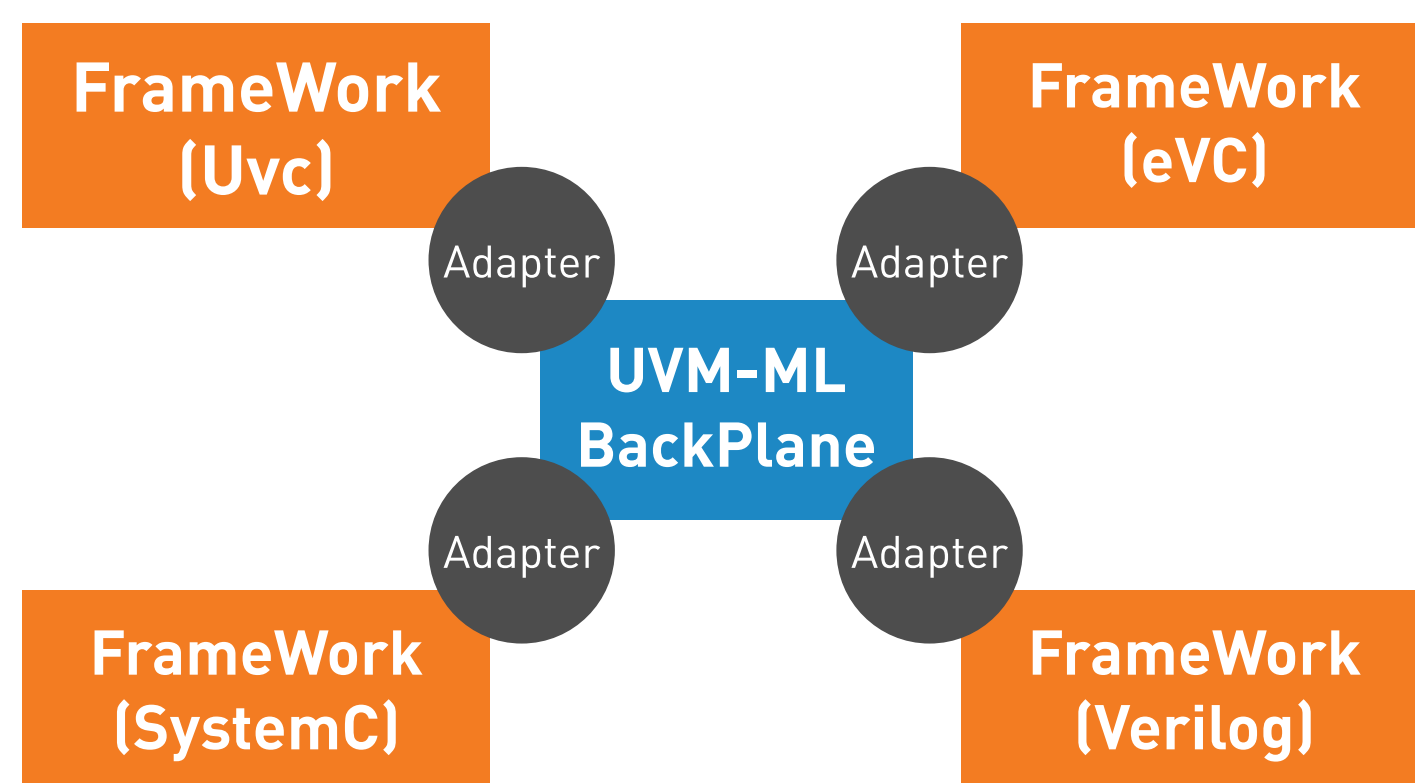2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Introduction

UVM –ML framework combines multi language IP and make them work together for any verification task. The basic ML setup is as shown below:

FrameWork (Uvc) — Adapter
FrameWork (eVC) — Adapter
UVM-ML BackPlane
FrameWork (SystemC) — Adapter
FrameWork (Verilog) — Adapter
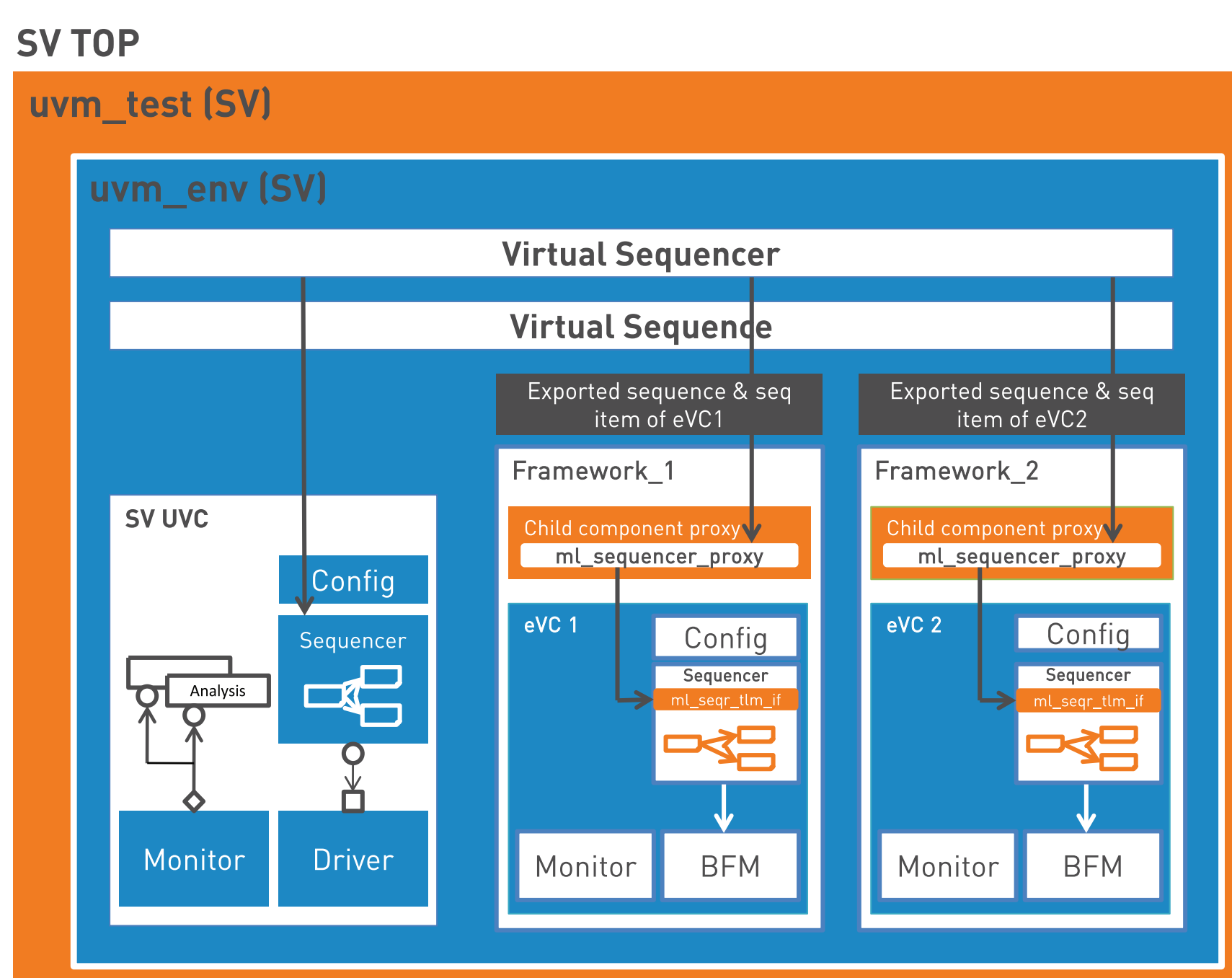
Objectives

Deploying ML framework to achieve –
• Reuse existing verification IPs irrespective of HVL used for representation
• Enabling faster bring-up of SoC testbench
• More focus on verification than testbench development at SoC level
• Optimized code with proven verification IPs
• Minimize verification environment bugs
• Faster time to market

Framework to support –
• Heterogeneous Objection Mechanism
• Configuration mechanism across multiple eVCs and UVCs
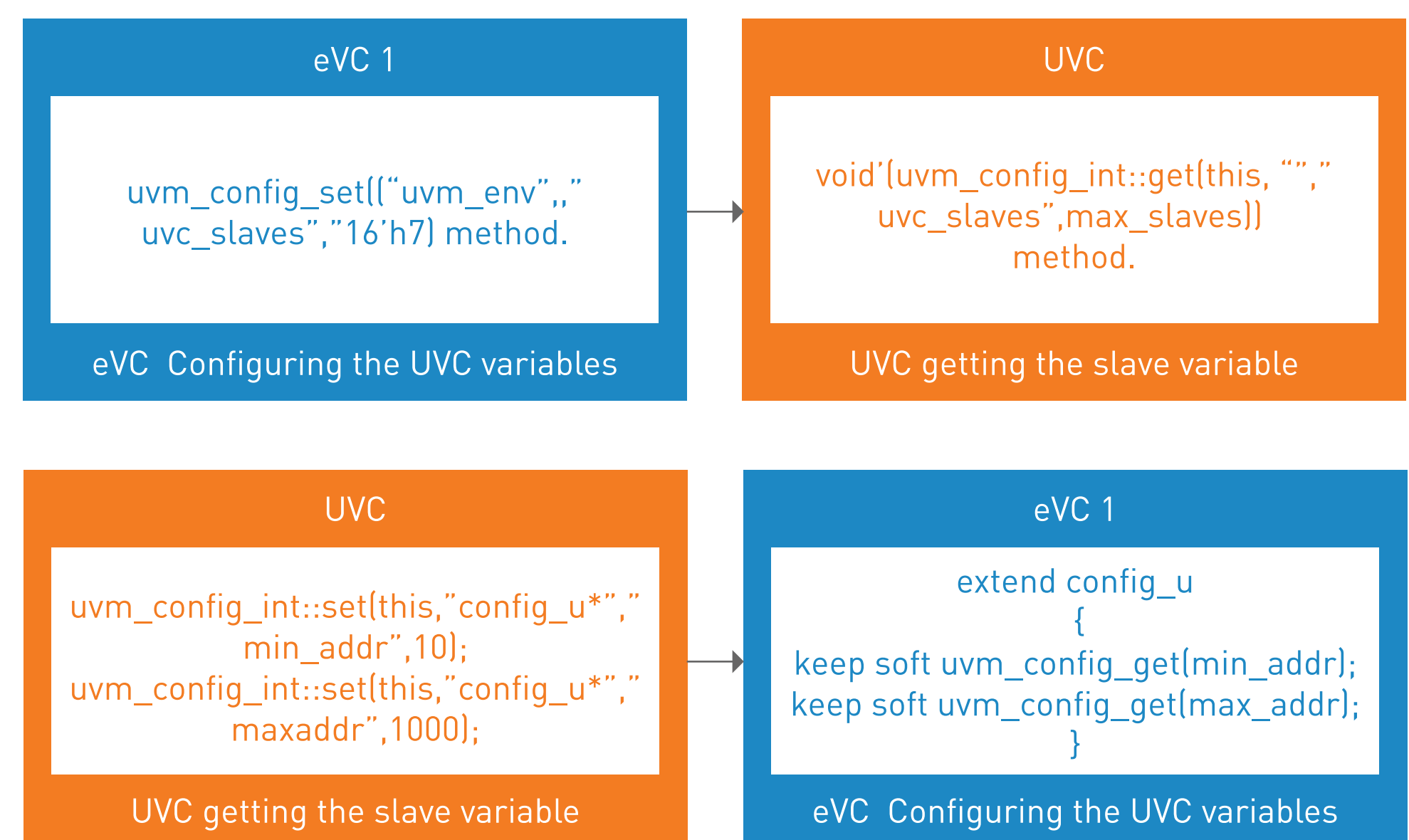• Reuse of existing sequences, assertions, coverage and scoreboard

The Setup

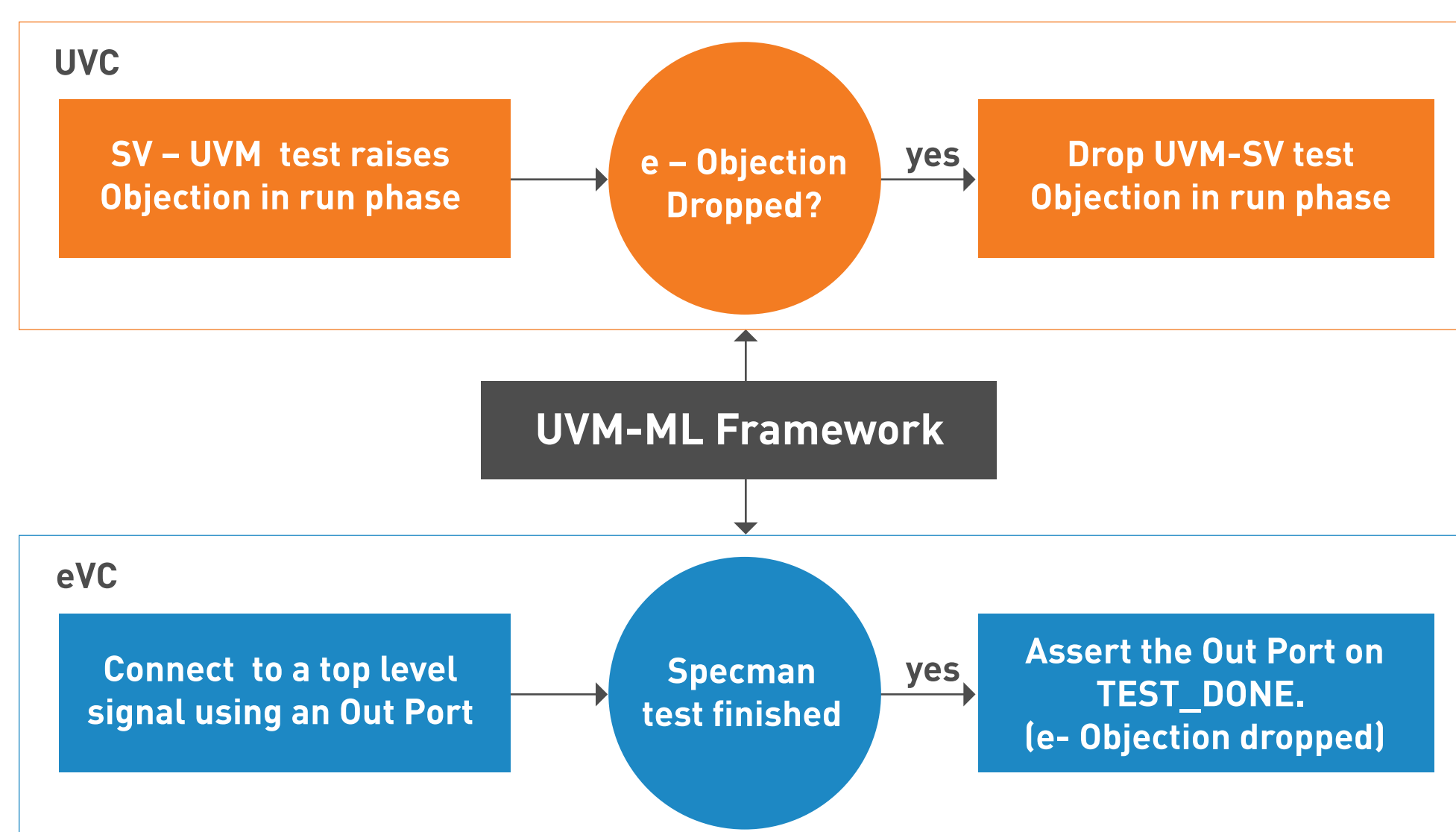Example demonstrating 2 + 1 IP integration: 1 UVC and 2 eVC

SV TOP

uvm_test (SV)

uvm_env (SV)

Virtual Sequencer
Virtual Sequence

Exported sequence & seq item of eVC1
Exported sequence & seq item of eVC2

SV UVC
Config
Sequencer
Analysis
Monitor
Driver

Framework_1
Child component proxy ml_sequencer_proxy
eVC 1
Config
Sequencer
ml_seqr_tlm_if
Monitor
BFM

Framework_2
Child component proxy ml_sequencer_proxy
eVC 2
Config
Sequencer
ml_seqr_tlm_if
Monitor
BFM

Configuration Mechanism: eVC to UVC and Vice Versa

The Heterogeneous configuration mechanism

eVC 1
uvm_config_set(("uvm_env",," uvc_slaves","16'h7) method.
eVC Configuring the UVC variables

UVC
void'(uvm_config_int::get(this, "","" uvc_slaves",max_slaves)) method.
UVC getting the slave variable

UVC
uvm_config_int::set(this,"config_u*"," min_addr",10);
uvm_config_int::set(this,"config_u*"," maxaddr",1000);
UVC getting the slave variable

eVC 1
extend config_u
{
keep soft uvm_config_get(min_addr);
keep soft uvm_config_get(max_addr);
}
eVC Configuring the UVC variables

Objection Mechanism

UVC
SV – UVM test raises Objection in run phase → e – Objection Dropped? → yes → Drop UVM-SV test Objection in run phase

UVM-ML Framework

eVC
Connect to a top level signal using an Out Port → Specman test finished → yes → Assert the Out Port on TEST_DONE. (e- Objection dropped)

Heterogeneous Objection Mechanism

Conclusions

UVM-ML framework helps in achieving –
1. SV based Virtual sequencer to control the multi-eVC sequences
2. Phase propagation across frameworks, e.g. build phase from UVC to underlying eVC
3. Configuration of the eVC from UVM-SV environment both statically and dynamically
4. Phase Synchronization
5. Graceful termination of tests from top-level

Proposed future work –
1. Enhanced Debugging and tracing capabilities
2. Messaging service mechanism at the ML level
3. Propagating Errors to the underlying slaves (eVC in this case)
4. A global objection mechanism

References:

1. B. Sniderman, V. Yankelevich, "Techtorial: UVM Multi-Language: Technology and Reference Application", CDNLive Users Conference, Santa Clara CA, March 2013, http://www.cadence.com/cdnlive/na/2013/Pages/default.aspx.

2. G. Leshem, V. Yankelevich, B. Sniderman. "UVM-ML Whitepaper. A Modular Approach for Integrating Verification Frameworks" (available for download at http://forums.-accellera.org/files/file/65-uvm-ml-open- architecture.

SMARTPLAY™ An Aricent Company