2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Ultimate Shift Left: Unleash the Power of UVM Virtual LAB Methodology upon SoC Verification

Roman Wang

Advanced Micro Devices, Inc. Shanghai, China

AMD

1. Abstract

The project execution schedule is the one of the important factors for a project to be successful in the market. The challenges of SoC verification execution typically come from the complexity of design and flow, stability of the database, resources, reuse, simulation performance, debugging turnaround time, and so on. Therefore, how to meet the schedule with high verification quality is always important for verification engineers.

In this paper, we promote a new practical Universal Verification Methodology (UVM) virtual laboratory(vLAB) methodology to address these challenges. It supports the requirements of reusability, scalability, and flexibility. Powered by it, the IP/SoC verification teams gain great benefits from finding issue, reporting for fix early and quick bring up. It has been widely adopted in many SoC projects for more than two years, and has successfully demonstrated a ‘shift left’ to pull in schedules with high quality.

2. The Challenges

- Complexity of the SoC design, environment, and flow.
- Stability of the SoC database.
- IP integration has semi-pipeline rule in the SoC.
- Lots of integration re-spins between the IP and the SoC.
- The bring-up of IP verification works as the serial mode in the SoC.
- Simulation performance and debugging turnaround time.

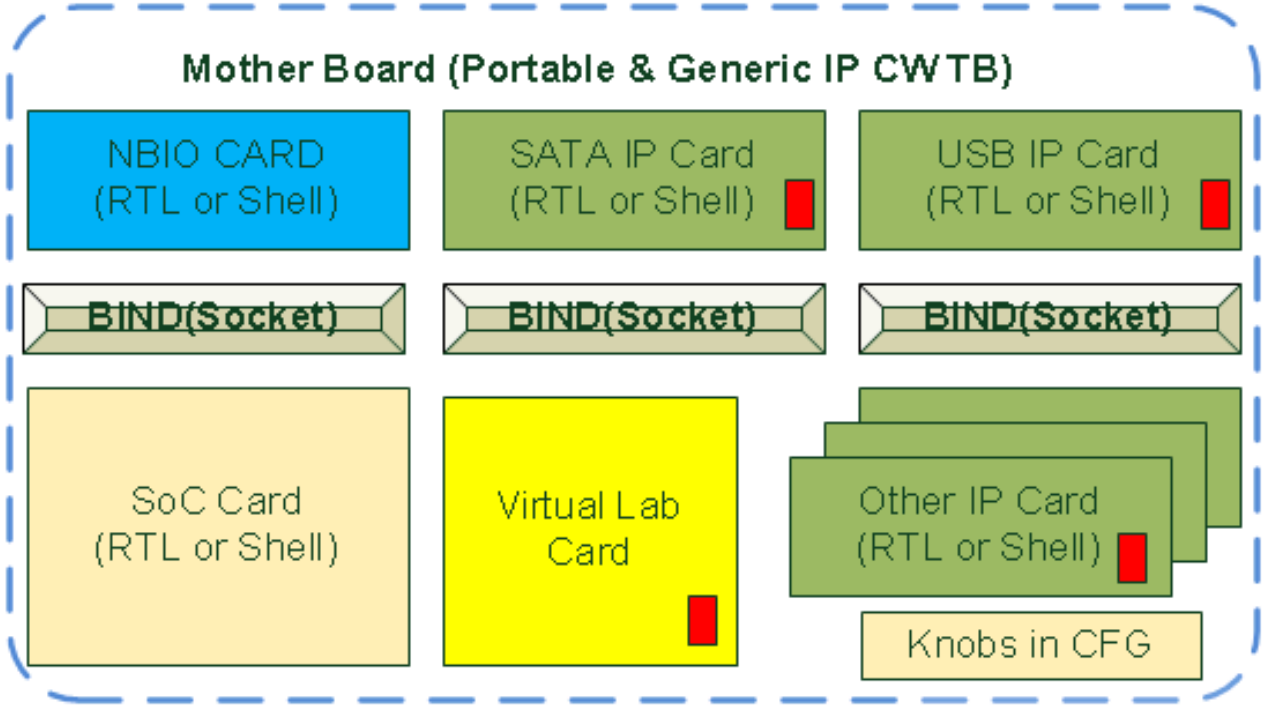
Combo Whacker (CW) is a sub-system verification environment based on SoC database.

3. Question Ourselves

- How to decouple the verification complexity?
- How to reduce the verification dependency?
- How to bring up verification environment in parallel?
- How to reuse design verification (DV) stimulus and test benches without any change?
- How to reduce or eliminate the re-spins because of the reuse?

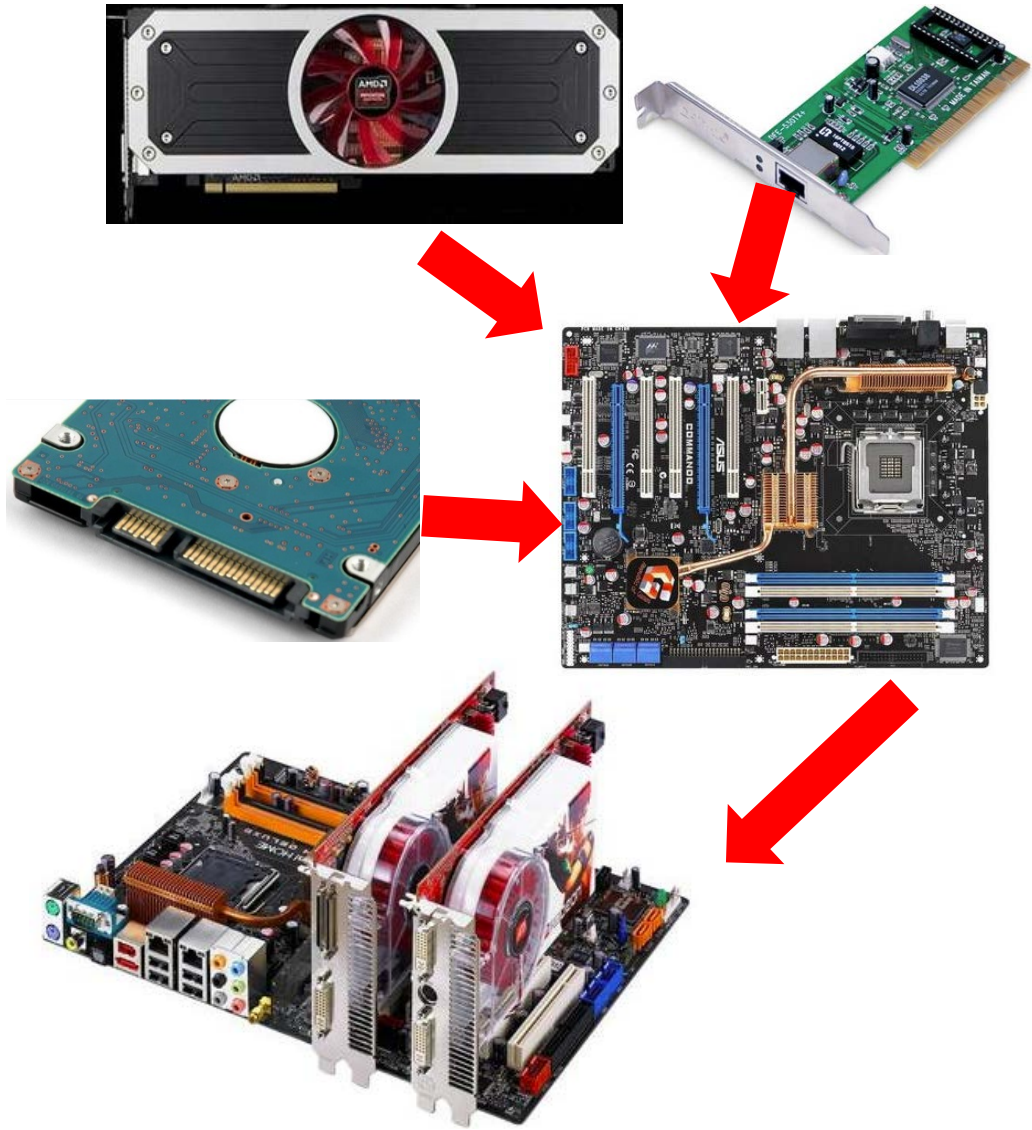
How can we do things a little earlier and better?

4. Brainstorming



Shell *is "Card Remove"* ***RTL*** *is "Card Insert"*

Indicates the verification properties reuse



5. UVM Virtual Lab Solution

- The virtual Lab (vLAB) architecture is looking like a complex laboratory which is a highly configurable, scalable UVM framework.
- It contains several different types of UVM Verification IP (VIP), reusable base testbench and generic UVM abstract layering.
- It defines different bring up modes to boost verification schedule.
- It is also powered by our UVM verification solution library to handle the shared verification challenges across multiple IPs and SoCs.

6. The Concept of Plug-Play

Design Point of View

- Motherboard (SoC RTL database), daughter cards (IPs).
- Configure the IP as RTL view to represent plug-in design under test (DUT).
- Configure the IP as Shell view to represent plug-out DUT.
- The shell should drive lower strength value by default to avoid the X propagation. The UVM VIP will drive the port of shell in the testbench.

Verification Point of View

- Mother board (the SoC level Verification database and flow).
- Daughter cards (the IP level UVM environment, including Testbench, Register Abstract Model (RAL), UVM memory model, sequence, test, etc.).
- Daughter cards are attached on DUT via System Verilog interface binds methodology.
- Configure daughter cards as passive mode to represent the plug-out.
- Configure Daughter cards as active mode to represent the plug-in.

7. The UVM vLAB Bring Up Modes

❑*vLAB Upper Link Mode*

It's intent to bring up the upper layer DUT complex data path where the IP is attached on.

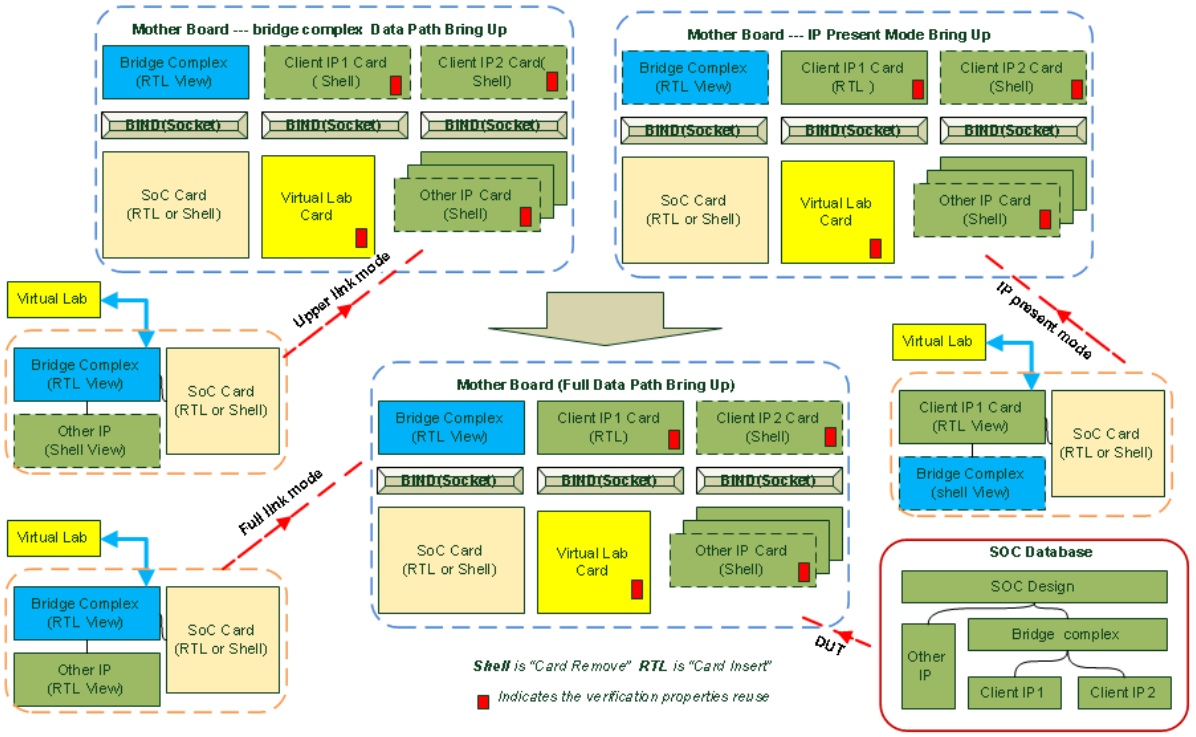
❑*vLAB IP Present Mode*

It's intent to bring up the IP itself DE/DV quality in SOC. Especially for different connection between IP standalone and SOC.

❑*vLAB* Full *Link Mode*

It's intent to bring up full data path including upper DUT complex & IP (SATA, USB3,etc.) .

It's out final goal!

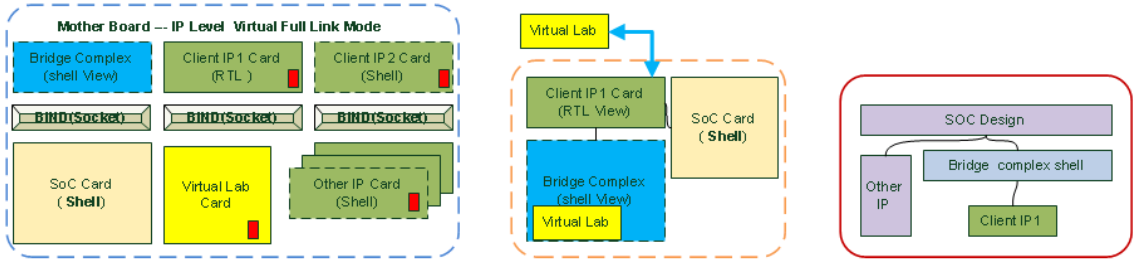


8. Virtual Prototyping and New Modes

To make good use of the time window between the IP- and SoC-level verification, we create several virtual UVM models to support Virtual CW Prototyping and qualify the IP2SoC reuse earlier.

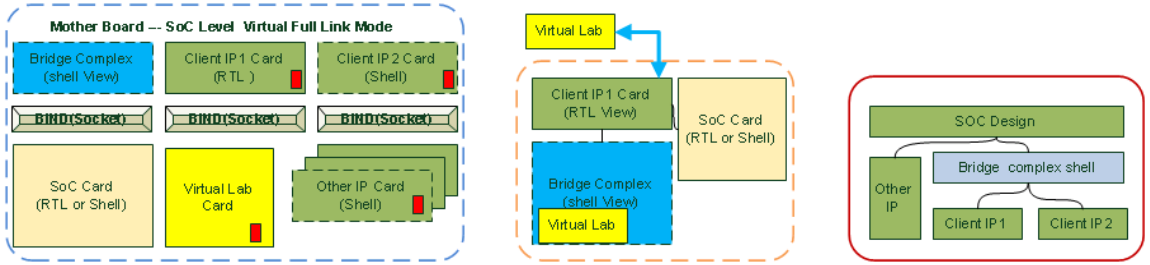
❑*vLAB IP* Virtual Full *Link Mode*

It's only intent to qualify the vLAB framework and IP environment/binds/sequence/test reuse in vCW platform.

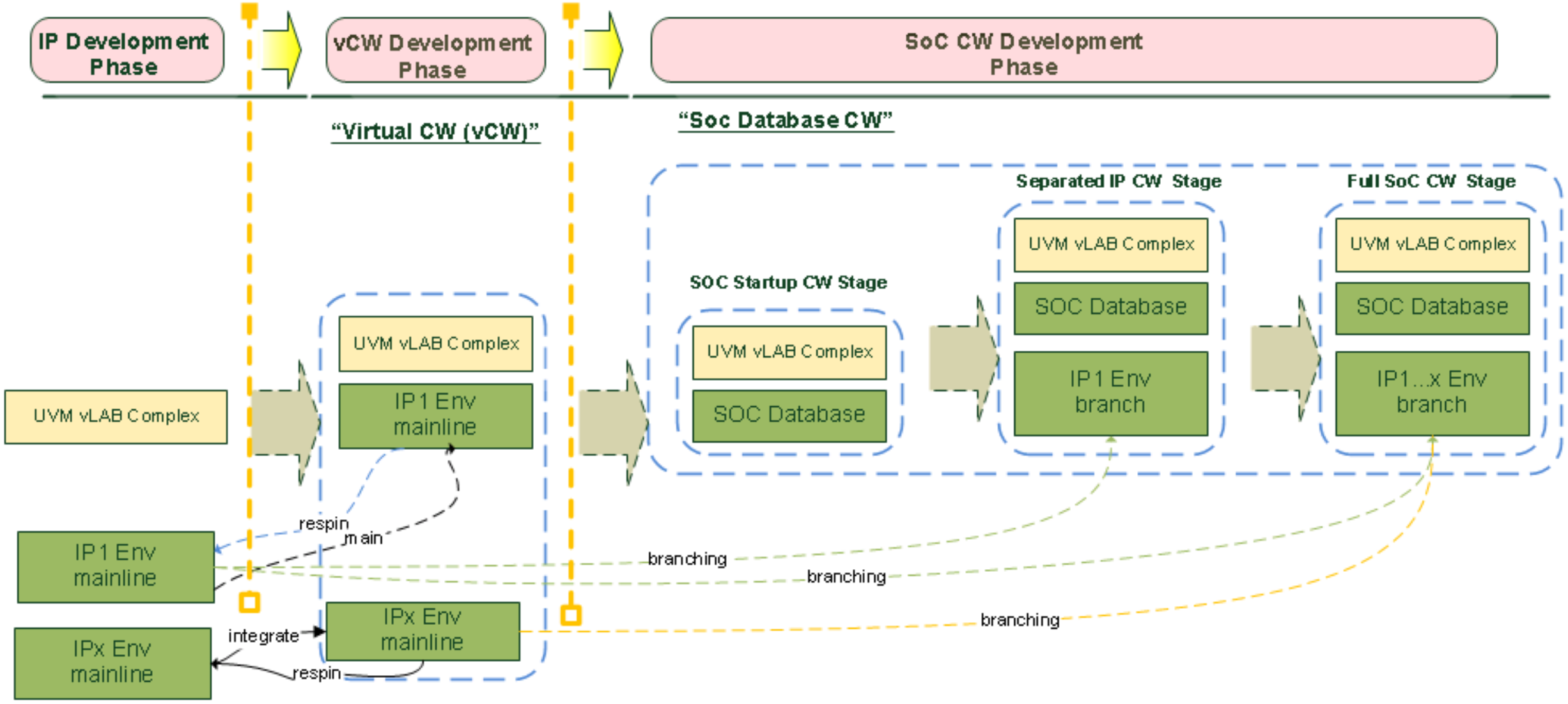


❑*vLAB SoC* Virtual Full *Link Mode*

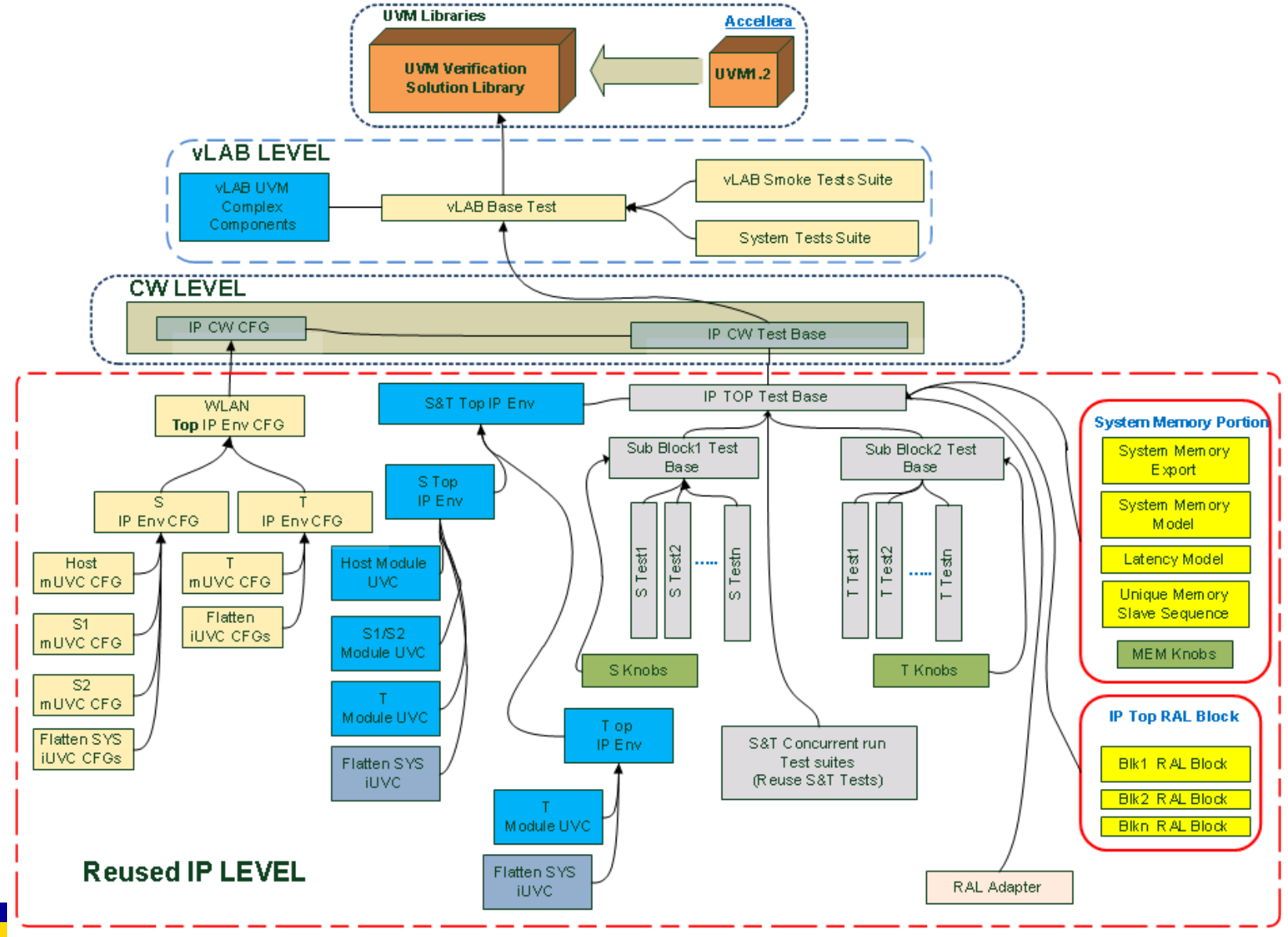
When the SoC database is ready and the upper layer design complex is not fully functional, replace the DUT with vLAB virtual UVM models.



9. The UVM vLAB Methodology Execution Workflow in Real Projects



10. Scalable vLAB UVM Abstract Layering



11. Conclusions

- Successfully deployed in multiple SoC projects for 3+years, and has been successful in achieving a ‘shift left’ to pull in projects’ schedules.

Projects/IP	New SoC (First adoption)	1st derived SoC	2nd derived SoC
IP1 vLAB	Tune 1st sanity pass > 4 weeks **@ project phase two**	Tune 1st sanity pass 2 weeks **@ project phase two**	Tune 1st sanity pass 1 week **@ project phase two**
Regression	Regression pass rate > 80% > 4 weeks after sanity pass	Regression pass rate > 80% 2 weeks after sanity pass	Regression pass rate> 80% 1 week after sanity pass
IP2 without vLAB	Tune 1st sanity pass >5 weeks **@ project phase three**	Tune 1st sanity pass 3~4 weeks **@ project phase three**	Tune 1st sanity pass 2~3 weeks **@ project phase three**
Regression	Regression pass rate > 80% > 6 weeks after sanity pass	Regression pass rate > 80% 4~5 weeks after sanity pass	Regression pass rate > 80% 3~4 weeks after sanity pass

roman.wang@amd.com