Verification of a Multi-language Components

A case Study: Specman E Env with SV UVM VIP

Eran Lahav
eranl@veriests.com

accellera SYSTEMS INITIATIVE™

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

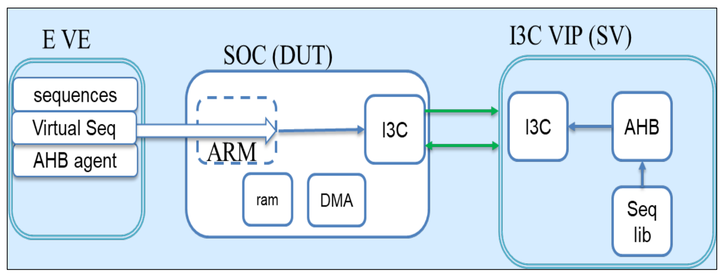
Introduction

Multi-language Verification requires special considerations: Synchronization of test phases and events and moving data objects between the language's worlds.

In some cases, re-writing part of the code, (either manually or automatically) could be a more reasonable solution. In this paper, we discuss the trade-off between the two alternatives: re-write the code and co-running with original languages.

The System

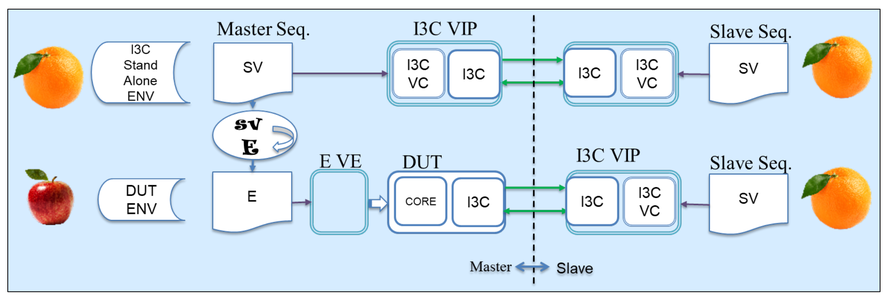
- SOC with peripheral I3C, Driven by ARM trough DMA
- Full E environment for SOC, including configurations, registers model, sequence , interrupt handling, bus drivers, etc.
- I3C UVM VIP from external vendor.



Configuration and Execution Options

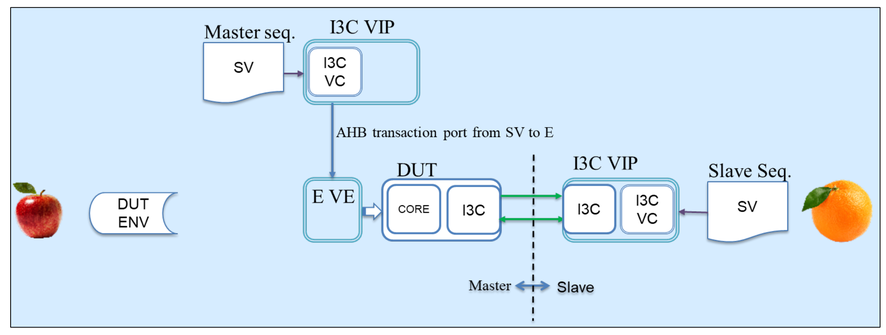
Option 1: Translate

Translate SV-UVM VIP sequences to E and use them within the SoC virtual sequences



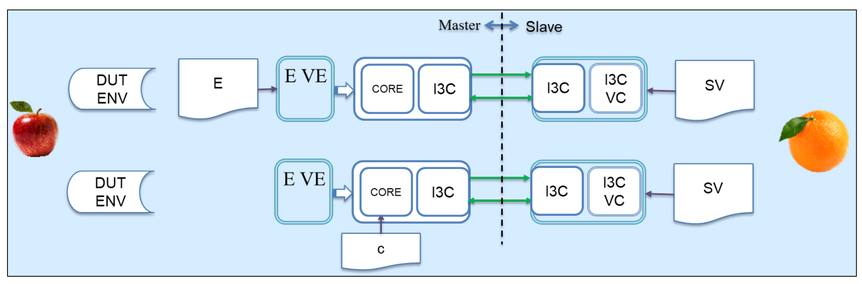
Option 2: Run UVM sequences (both peers), Port Memory accesses

Run SV-Sequences and call E methods from SV for each memory accesses.



Options 3,4: Write from Spec

Write the sequences in E/C from scratch



Consideration

Compare Translation/Rewrite vs. co-run

		Notes	Translate to E	UVM execute	Rewrite from Spec	Write in C
1	Coding Effort	Of Sequences	MED	LOW	HIGH	HIGH
2	Env Effort	Test flow	LOW	HIGH	LOW	LOW
3	Tests Debug Effort		MED	LOW	HIGH	HIGH
4	Preserve Test Suit	Run exact Vendor scenario and checks	HIGH	FULL	LOW	LOW
5	Flexibility	Running a different scenario. Post silicon debug	LOW	LOW	HIGH	MED
6	SW reusability	Using sequences as reference to SW	LOW	LOW	MED	HIGH

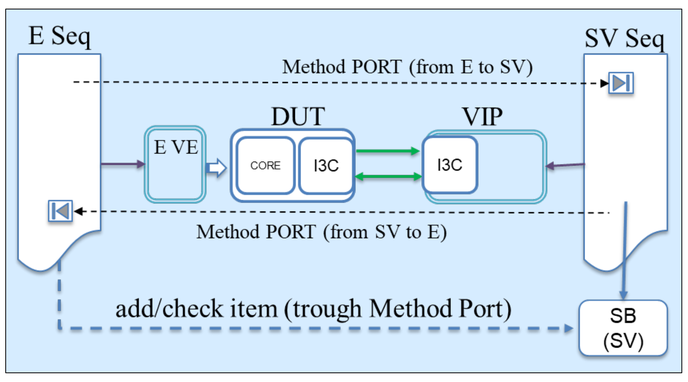
Translation

Sequences translation was done with Python program:

- RAL to Specman register (vr_ad like)
- UVM sequence library to E sequences (mostly body() task to body() TCM)
- It covers only small portion of the Language and uses shortcuts (as macro translation instead of expand)
- Neither all cases nor all commands are supported (less then 10%)
- It took 3-person week to complete
- We estimate a full SV to E Translator to require one person-year

Synchronization

Synchronization between sequences and data objects between languages can be done either by E method ports or by TLM (Transfer Level Model) ports.



Code Examples

- E to SV : Call in E code and executed in SV (UVM object or TB)

```
method_type wb_gamla_set_name(name: string);
.....
i3c_set_uvm_test_name: out method_port of wb_gamla_set_name is instance;
keep i3c_set_uvm_test_name.hdl_path() ==
"~.dte_board_taa0.get_uvm_test_name_from_e";

Test_name: string;
i3c_set_uvm_test_name$(test_name);
```

Define in E

Assign and linked to SV in E

Called in E function

```
function void get_uvm_test_name_from_e(string name);
      sv_test_name = name;
      $display($sformatf("I3C DTE Test is %s",sv_test_name));
endfunction
```

Executed in SV function

Conclusions

Combining a component with a different language into an existing verification environment depends on the characteristics of the system and the verification requirement. In some cases, partially translation can be more efficient and time/cost-effective than applying a standard language-to-language porting. We found that for sequences, particularly for those which produce memory access command, translation of the code is beneficial.

Using UVM-ML Open Architecture is a valuable way to connect and run together components with different languages. Yet, the implementation of these kinds of systems requires a special attention for synchronization. Ports (TLM) between the components should close this gap and enable effective verification.

Veriest

Project was done in collaboration with **Nuvoton** Technologies