2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Integration of HDL Logic inside SystemVerilog UVM based Verification IP

Aleksandra Panajotu
Elsys Eastern Europe, Belgrade, Serbia
aleksandra.panajotu@elsys-eastern.com

accellera SYSTEMS INITIATIVE

Introduction

As complexity of communication between the DUT and the system rises, implementation of the VIPs required for verification becomes more demanding.

In some cases, the nature of the DUT requires that the VIP behaves as similarly to the already implemented Design as possible. If the surrounding Design uses complex logic that is hard to reproduce in any other way than implementing the same logic in the VIP, the development of such VIP can require great effort.

The focus of this poster is on the integration of HDL Design Logic inside a VIP Wrapper which increases the precision with which VIP will model the expected Design behavior in cases where the randomization of the stimuli that VIP should provide is not needed and the exact replication of behavior of surrounding logic is expected.

VIP Wrapper overview

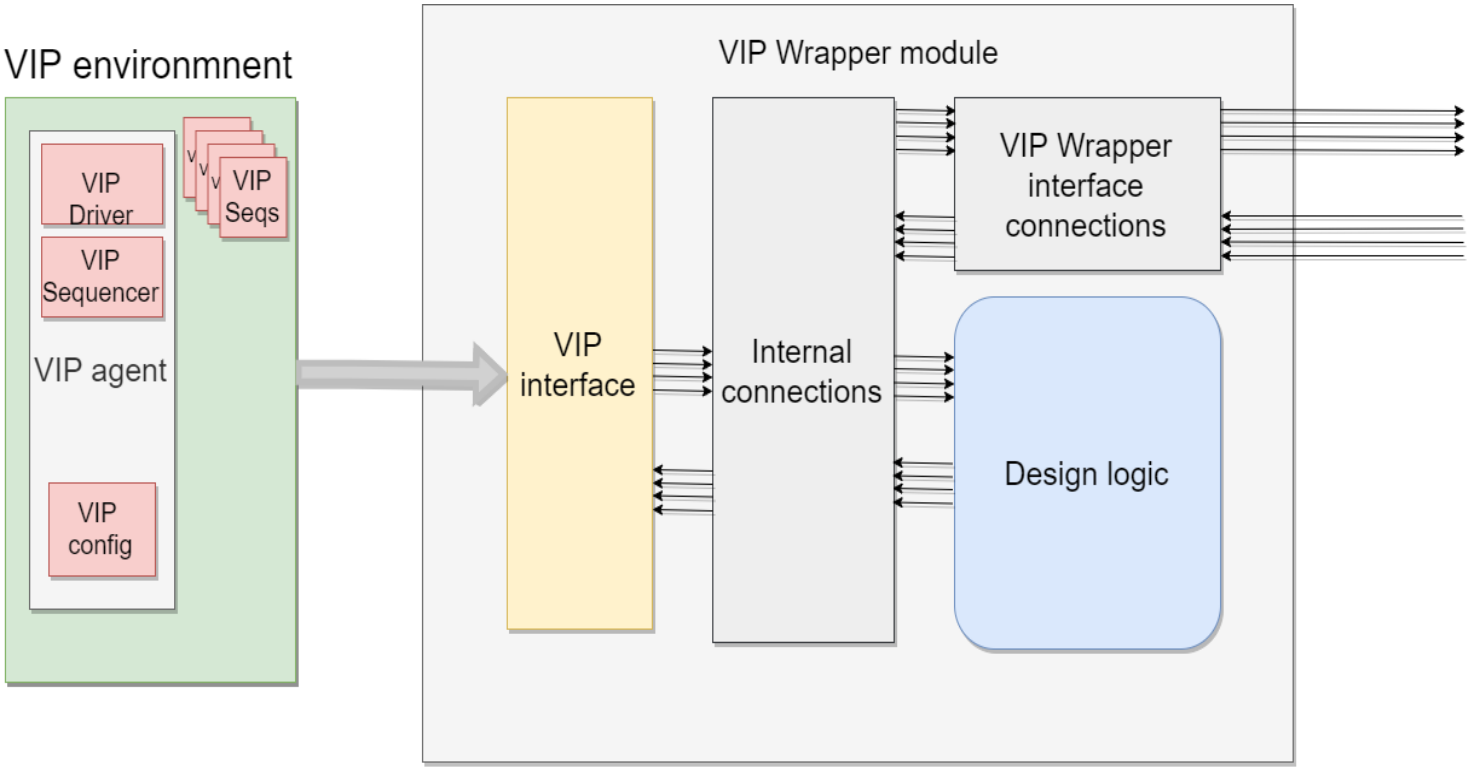
VIP Wrapper implementation proposed here contains three distinct parts: SystemVerilog UVM based VIP, HDL Design logic and the Wrapper Interface. The idea is to integrate the Design and the interface module of the VIP into the Wrapper.

The VIP will keep all its SystemVerilog and UVM features and will represent a part of the system that is driving the input signals of the Design that will process them and use them to stimulate the DUT. This allows the Verification team to develop randomized sequence items and sequences which introduce randomization to the VIP's behavior.

This way the VIP is behaving according to the needs of the DUT and, also, the verification team has a way to control the behavior of the surrounding system that includes the Design to achieve exercise of all possible scenarios.

Wrapper module

The Wrapper will act as an interface to the rest of the UVM environment where it will be used. It will also have internal connections between the VIP interface and the Design.



VIP Wrapper development overview

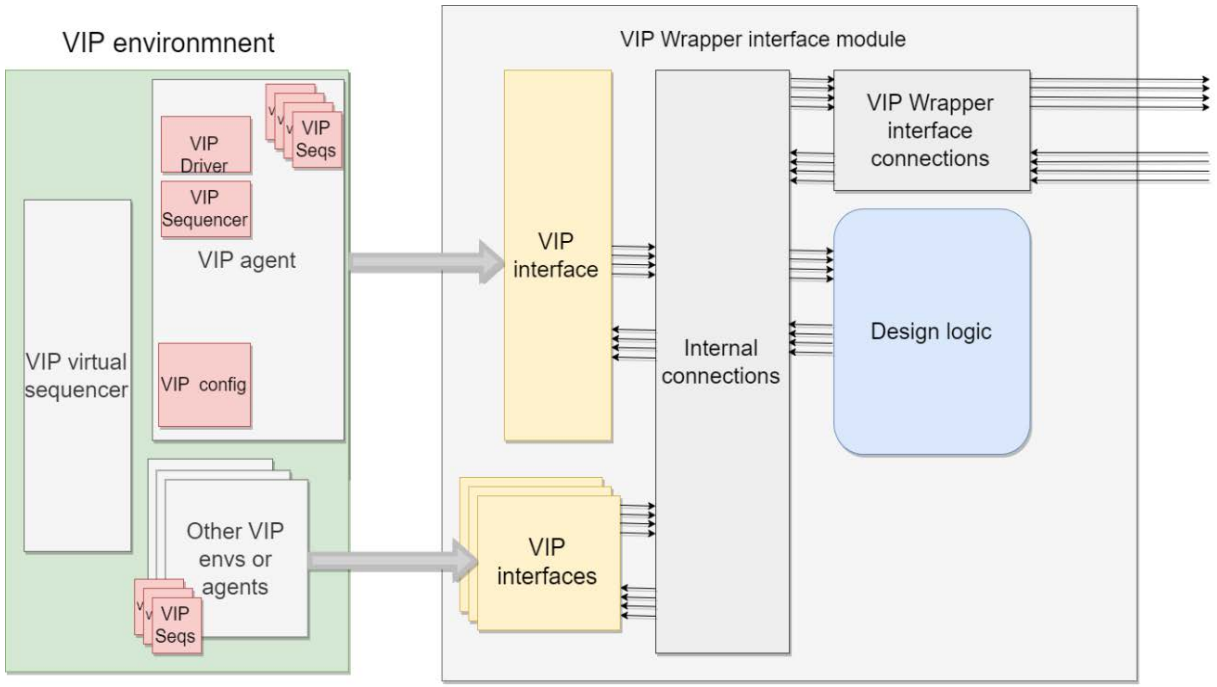
The Wrapper has the Design instance and the VIP interface instance inside it and the connections to the DUT in the Testbench. SystemVerilog does not allow instantiation of a module inside an interface instance, so the Wrapper will be declared as a module. This does not affect its integration or its behavior.

```
module wrapper_if_module
(
//input and output signals of the VIP Wrapper
);

       //internal signals used for connections between the Design and the VIP interface

      design_module    design_instance
      (
       //input and output connections of the Design
      );

      vip_interface    vip_interface_instance
      (
      //input and output connections of the VIP
      );

      //assignments of the internal signals

endmodule : wrapper_if_module
```

VIP implementation variations

The VIP doesn't have to be a single component, it can be a UVM environment with a virtual sequencer and several VIPs instantiated inside the environment that are working independently (i.e. independent request-acknowledge modules) or in coordination (simulating some more complex part of the system for which multiple VIPs can be used instead of using one complex VIP).

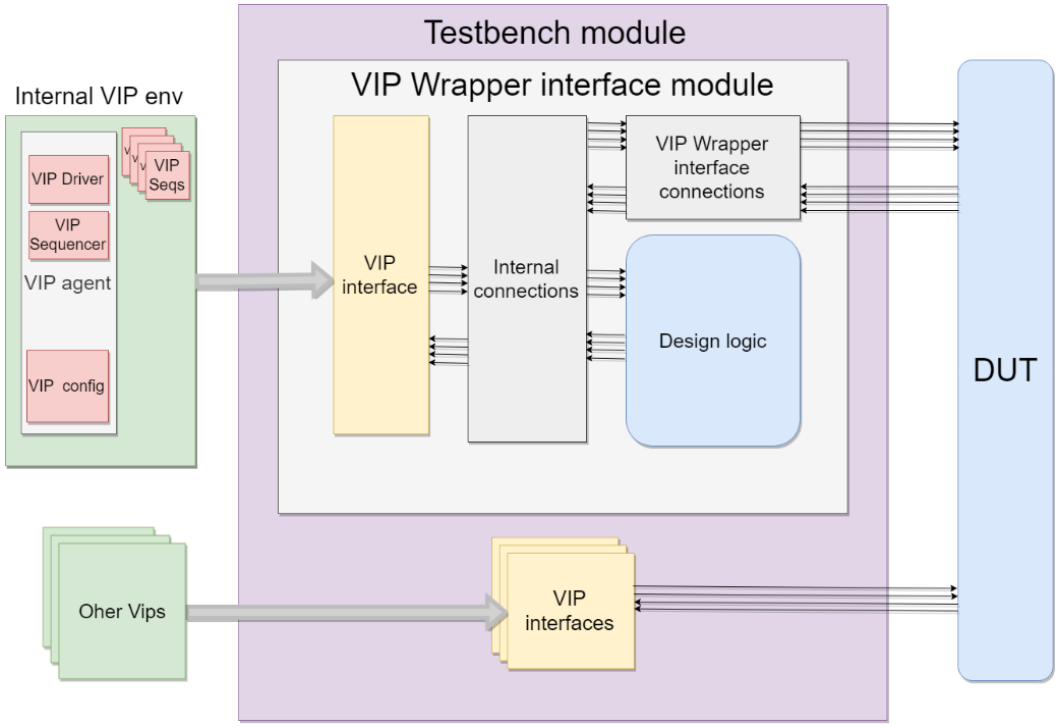
Their interfaces will be connected to the Design and the Testbench in the same manner and their environments or agents will be instantiated inside the environment of the VIP.



Integration overview

The Wrapper is instantiated in the Testbench as a module and is connected to the signals like any other module or interface.

Signals of the VIP interface should not be connected to the Testbench, only the Wrapper interface signals. VIP interface that is instantiated in the Wrapper will be passed through the UVM configuration database from the Testbench and the VIP environment will be able to acquire it.



Integration in Testbench

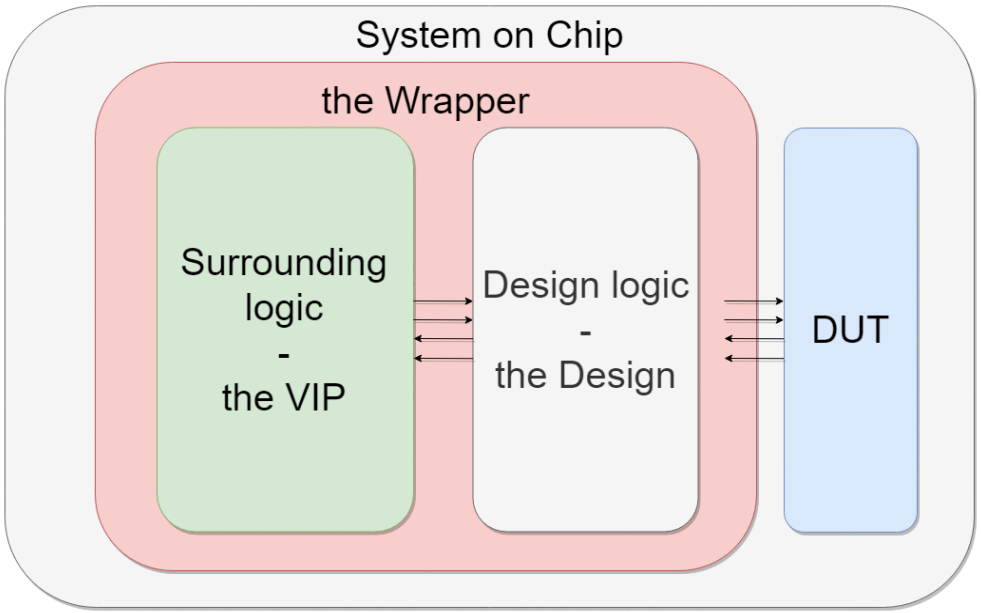
It is important that the VIP interface instance is passed to the UVM configuration database instead of the Wrapper instance as the VIP is not aware that is a part of a wrapper and is expecting the get its own interface through the database.

```
module testbench;

       //internal testbench signals

      dut_top_module   dut;
      (
       //input and output connections of the DUT
      );

      wrapper_if_module   vip_wrapper_instance
      (
      //input and output connections of the VIP Wrapper
      );

      initial begin
            uvm_config_db#(virtual vip_interface)::set(null, "*.vip_wrapper_instance*",
                  "vif", vip_wrapper_instance.vip_interface_instance);

            run_test();
      end

endmodule : testbench
```

Use-cases

This method is most applicable in cases where the DUT should be connected to a part of the system that can be split into two parts:

1) module with complex logic that is difficult to model with high precision but is available for reuse;
2) the surrounding logic, which is generating inputs to that module.



Problems with regular UVM approach

This approach can be used in situations where the exact behavior of a part of the surrounding logic must be reproduced. On projects where multiple IPs that are closely connected are being developed at the same time but by different Verification teams, especially when the IPs are not communicating using a standard protocol it can be very hard to create VIPs that simulate the behavior of the surrounding system.

The solution is to integrate the module of the Design that is responsible for all the calculations and synchronizations and create a VIP that models the feedback the Design needed from the system. This way every time the Design gets updated, the Wrapper would also be updated with the new version of the Design.

Reused Design

It is highly preferable that the Design that is being used inside the Wrapper is fully verified. In case the verification is not fully done, cooperation between Verification teams that are using the Wrapper and the one that is working on the Verification of the Design that is being used is very important.

In this case, it is also recommended that the same Verification team that is doing the verification of the Design that is being reused is developing the VIP Wrapper as it could support eventual changes in the Design implementation.

Positive aspects of this kind of cooperation are that in cases where some corner case behavior was not accurately defined in the specification or was not even discovered.

Conclusion

Development of a VIP Wrapper can be beneficial when exact replication of the surrounding logic is needed. The VIP Wrapper developed using this method has the same core behavior as the design logic that it should model with added possibility of randomization using UVM VIPs wrapped inside.

On the other hand, it introduces a risk of a bug being present inside the Design that is being reused, and for that reason, the Design for integration must be verified.

The randomization of the stimuli that Design inside the VIP Wrapper provides is reduced and comes down to the randomization of the inputs to the Wrapped Design. This approach is recommended for generating stimuli that should not be greatly randomized, as it can provide greater precision and efficiency in VIP development.