A Holistic View of Mixed-Language IP Integration

Pankaj Singh
Infineon Technologies AG
pankaj.singh@infineon.com

Gaurav Kumar Verma
Mentor Graphics Corporation
gaurav-kumar_verma@mentor.com

ABSTRACT

IP reuse has long been touted as one of the key factors in enabling development of today's complex SoC designs. The concept of reuse seems simple and easy in theory, but there are a number of obstacles that design and verification teams must address to be successful, especially in the case of commercial IP cores. One of the significant barriers to IP reuse today is the wide variety of design languages used in IP. SystemC, SystemVerilog, and conventional HDL languages have unique strengths which make them more suitable for writing certain portions of a design or IP. Designers also often choose a language based on their past experience. This frequently leads to portions of designs written in different languages being integrated in a single design. IP available from different sources may also come in different flavors. Connecting such IP requires special skills, in terms of expertise in all the different languages involved, which is not always easy to find. The wide variety of options available to make these connections further increases the complexity of the problem.

The above challenge necessitates the need for a methodology that compares the different ways of making mixed-language connections (such as direct instantiation, the SystemVerilog bind construct, SystemC control/observe, SC Verification-connect, and SC-DPI) and defines their pros and cons, providing precise information to help users select the best approach for their particular SoC.

In the past, most of the work was done by EDA companies to develop tools that understand mixed-language design for simulation but very little has been done toward providing a comprehensive methodology that highlights best practices, thereby minimize mixed-language design integration issues. Only relevant available work written on mixed language IP reuse [1] used direct instantiation and SystemVerilog bind methods without providing solution/benefits of other alternate approaches or comparing the two against each other.

This paper looks at mixed-language design integration from both the EDA tool developers' and designers' perspectives. It describes different approaches and provides useful insights to help users select the best option for integrating two IP blocks in a mixed-language environment. It is an extension of our past work and illustrates the performance comparison of various approaches using 10 Gigabit Ethernet verification environment. A 'mixing methods' approach is also introduced to help designers make their mixed-language connections in scenarios where all methods get eliminated or where using more than one method for integration provides a more optimized connection.

The methodology presented in this paper is further used to develop a utility that takes the two regions, written in different languages, as input and automates the hook-up connection process by suggesting the most suitable approach. It also generates snippets of code that can be automatically inserted in the design to make these connections seamless and less prone to the kind of errors that may result from manual updates. The paper contains a working prototype of this utility, highlighting productivity improvement results from real-world, mixed-language design integrations.

Keywords

1. INTRODUCTION

The wide variety of design languages available today poses a significant barrier to IP reuse. Often designers are not aware of various mixed-language design integration options. Other times the knowledge on various options is available, but it is difficult for a user to choose the best suitable option based on their mixed language design scenario. This difficulty in mixed-language IP integration and reuse often leads to finding issues late during the design cycle, which impacts the overall productivity.

This paper provides a comprehensive methodology that highlights the best practices for mixed-language design integration and automatically comes up with an option for designers to select the optimal method for integration. There are broadly five ways of making mixed-language connections. Pros and cons of each of these approaches and their comparison is described in terms of the usage scenarios, performance implications of using one versus the other, delta cycle value update concerns, and more. A step-by-step guideline based on decision-making trees that designers can follow to help them decide which approach best suits their particular mixed-language integration scenario is also discussed.

The paper starts with details on various options for connections of mixed-language IP blocks, illustrating each method with a common example. It also includes a summary of comparisons between different methods. The third section elaborates on a step-by-step methodical approach for integration. The fourth and fifth sections cover the details about the utility that automates the IP integration. The last section highlights the benefits of this methodology.

2. METHODS TO CONNECT MIXED-LANGUAGE IP BLOCKS

In this section we will introduce each of the five methods for making mixed-language connections and discuss their pros and cons. A

comparison of different approaches will also be provided to help determine the best suitable IP reuse option for a user based on the design scenario.

2.1 Direct Instantiation

In the direct instantiation method, an IP block written in any language is instantiated directly inside the target IP block (written in any language) within the SoC. Here the instantiation statement follows the syntax of the target IP block, as if the instantiated IP block was written in the same language as the target IP block.

This method is the most commonly used for making mixed-language connections as it offers seamless integration with the rest of the code. However, because of the nature of its use-model, it requires that the source code of the target IP block is available. This significantly limits the usability aspect of this otherwise powerful method in real-world IP reuse situations.
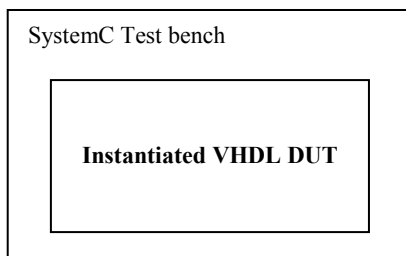

Figure 1. Example of Direct Instantiation

2.2 SystemVerilog Bind Construct

The SystemVerilog bind construct provides an IP block access to both external ports and internal signals in the target IP block. The selected and target IP blocks can be written in any design language. This method provides a powerful capability that, together with a specifically designed use-model, can be used to conveniently connect the two IP blocks independent of their languages.

The SystemVerilog bind construct is increasingly becoming the preferred method for connecting IP blocks in SoC's today, as it offers hook-up connections between two IP blocks without requiring their source code to be present.

Though using SystemVerilog bind construct to bind to a SystemVerilog target scope has been standardized, using this construct to bind to VHDL or SystemC target scopes has not been standardized yet, and as such, it is not fully compatible with all the available simulators. What may work with a simulator from one EDA vendor cannot be guaranteed to work with the simulator of another EDA vendor. Besides this limitation, EDA vendors also differ in their use models.
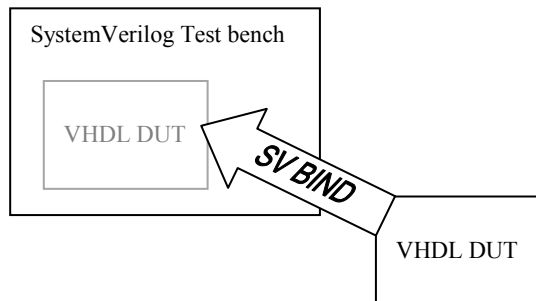

Figure 2. Example of SV Bind Construct

Also, since bind is a SystemVerilog construct, it must be used only in the SystemVerilog regions of the SoC. For instance, if a user wishes to connect a VHDL IP block with a SystemC IP block, an intermediate dummy SystemVerilog wrapper module will have to be created to use the bind construct, which may not be a very efficient approach. These factors somewhat restrict the usage of this otherwise powerful method of making connections.

2.3 SystemC Control/Observe

SystemC control/observe is a powerful construct that allows connection of signals across the hierarchy of a SystemC IP block to any other signal across the hierarchy of another IP block written in SystemVerilog or HDL. It can also be used on pre-compiled SystemVerilog and HDL IP blocks, but the SystemC IP block where SystemC control/observe constructs are used must have source-code visibility. This method cannot be used on compiled IP blocks.

Secondly, it requires the full hierarchical path of the source and destination objects, increasing the complexity. Also, since this method creates a jumper to connect the two signals across IP blocks, specialization and parameterization of IP blocks is not possible when this method is used. All these factors and its requirement for a non-compiled SystemC IP block somewhat restrict the usability of this otherwise useful construct.
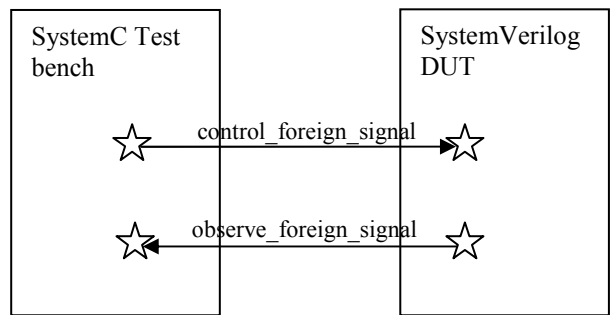

Figure 3. Example of SystemC Control/Observe

2.4 SystemC Verification Connect

SCV-connect is the standard version of SystemC control/observe for IP blocks that include the SystemC Verification Library. It is not as optimized as the SystemC control/observe method and requires the SystemC Verification library to be included in the IP.
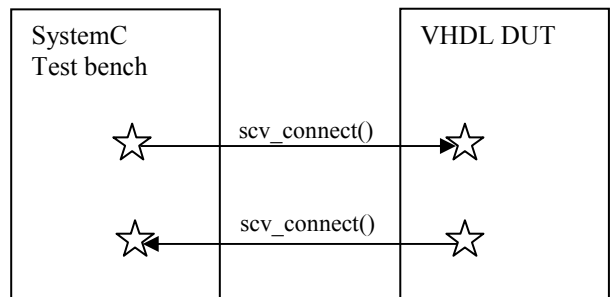

Figure 4. Example of SystemC Verification Connect

2.5 SC-DPI

The SystemC Direct Programming Interface (SC-DPI) method provides an interface between SystemVerilog and SystemC that facilitates inter-language function calls. This means a SystemVerilog

IP can call a function defined in a SystemC IP, and vice versa. It is a fast and suitable technique of connecting SystemVerilog IP blocks with SystemC IP blocks that have their external interfaces defined in the form of methods only.
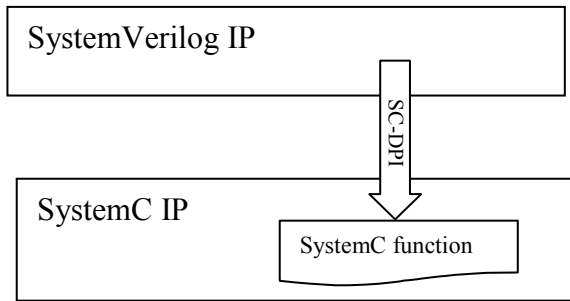

Figure 5. Example of SystemC-DPI

2.6 Comparison between Different Approaches

The table [1] below provides comparison between different approaches for making mixed-language connections. Various important aspects, such as usage scenarios, performance implications of using one versus the other, delta cycle value update concerns, etc. are listed for comparison.

Table 1: Comparison between Different Methods

	A	B	C	D	E	F
Direct Instantiation	Yes	Yes	No	2	Yes	All
SV Bind Construct	Yes	Yes	Yes	3	Yes	All
SystemC Control/Observe	Yes	Yes	No	4	No	1 SC + 1 SV/VHDL
SCV_connect	Yes	Yes	No	4	No	1 SC + 1 SV/VHDL
SC-DPI	Yes	Yes	No	1	No	1 SC + 1 SV

Legend:

> A: Method Works If Source Code Is Available
> B: Method Works If One IP Is Compiled
> C: Method Works If Both the IPs Are Compiled
> D: Performance (Lower Is Faster)
> E: Delta Delay in Data Transfer
> F: Languages Supported

3. THE METHODOLOGY

This section presents a 3-step methodical approach that helps users select the best option for integrating two IP blocks in a mixed-language environment.

STEP 1: Understanding the IP Blocks

As the first step towards deciding which approach to use, designers should:

- identify the design languages of IPs
- check the availability of source code of IPs

At the end of this stage the designer will have a clear understanding of the two IPs that are to be connected.

STEP 2: Understanding the Connections

After gathering information about the two IP blocks, designers should further:

- Analyze and list down the connections that are required to hook-up the two IP blocks together.

All connections can be broadly divided into four categories.

CATEGORY A

All connections are confined to one, and only one, design-unit in the two IP's, respectively.
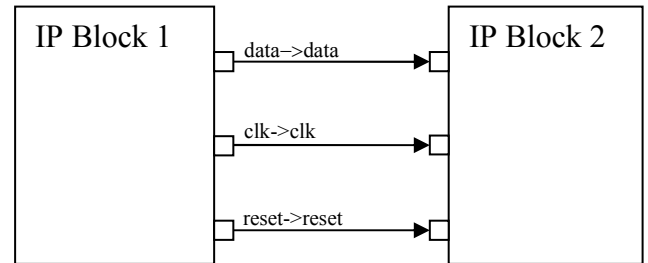

Figure 6. Category A Connections

CATEGORY B

All connections are confined to one, and only one, design unit in one IP block but are distributed across the hierarchies spanning through multiple design units in the other IP block.
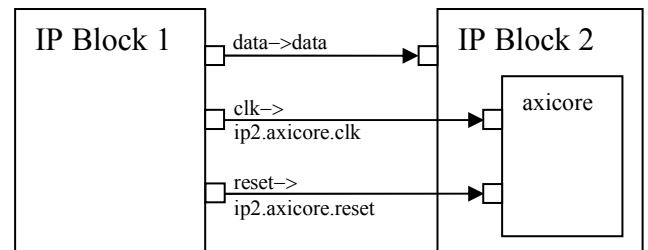

Figure 7. Category B Connections

CATEGORY C

All connections are distributed across the hierarchies spanning through multiple design units in both the IP blocks.
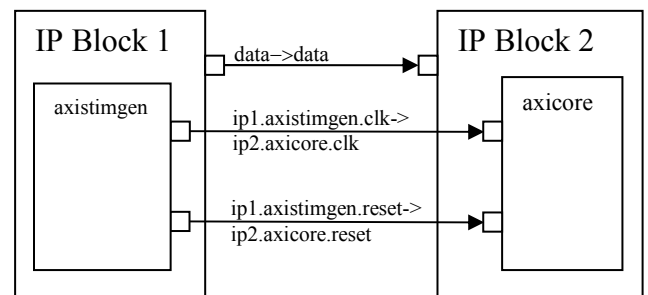

Figure 8. Category C Connections

CATEGORY D
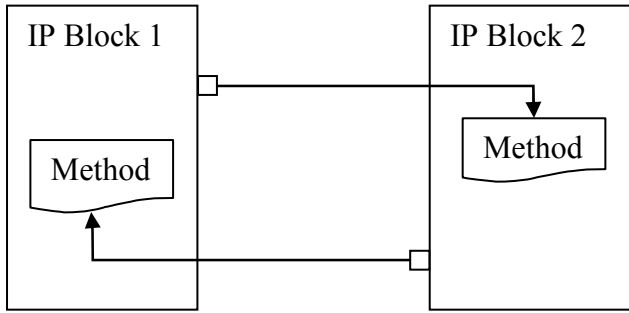
All connections are through method ports only.

Figure 9. Category D Connections

STEP 3: Finalizing the Method

This is the most important step of the proposed methodology where the designers will zero-in on the best method to connect the two IP blocks.

This step goes through a comparison of available methods, elimination of methods, and mixing method routines to arrive at an optimized approach.

Elimination

As the first step, designers should eliminate the choices which cannot be used at all to connect their two IP's together, purely on the basis of the nature of the IP's or the connections required.

The following table can be used for this elimination process:

Table 2: Elimination process for optimal method selection

Method	Elimination Criteria: Eliminate When =>
Direct Instantiation	Both the IP blocks are pre-compiled
	Category C connections are required between IP blocks
SV Bind Construct	Category C connections are required between IP blocks
SystemC Control/Observe	One of the IP blocks is not SystemC
	All SystemC IP blocks are pre-compiled
SCV-CONNECT	One of the IP blocks is not SystemC, which uses SCV
	All SystemC IP blocks are pre-compiled
SC-DPI	One of the IP Blocks is not SystemVerilog
	One of the IP Blocks is not SystemC
	Connections between IP blocks does not involve method ports only

Mixing Methods

There may be situations where it is desirable to use more than one method to connect two IP blocks. Mixing methods is especially useful in the following scenarios:

All Choices Eliminated: There could be situations where elimination process results in elimination of all the available choices. Mixing of modes may be the last hope of making connections in such scenarios.

As an example, consider the case where a precompiled SystemC IP needs to be connected to a SystemVerilog IP whose source code is available, and the connection involves a mixture of Category A and

Category D connections. In such a situation, elimination process will result in elimination of all the available choices.
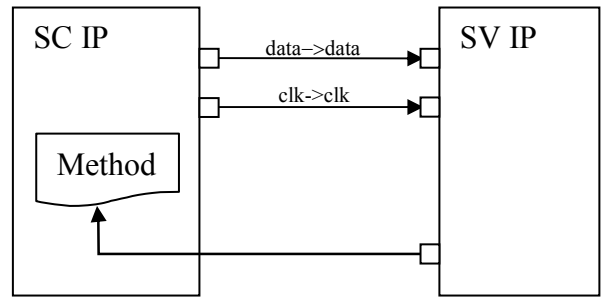

Figure 10. Connecting IP Blocks with Both Category A and Category D Connections

For connecting these two IP blocks, a mixture of direct instantiation and SC-DPI will have to be used.

More Optimized Solutions: Sometimes mixing multiple methods of connections can provide more optimal results as compared to one single method.

As an example, consider the scenario where a precompiled SystemVerilog IP needs to be connected to a SystemC IP whose source code is available, and the Category C connections are involved with a majority of connections being confined to one, and only one, design-unit in both the IP blocks.
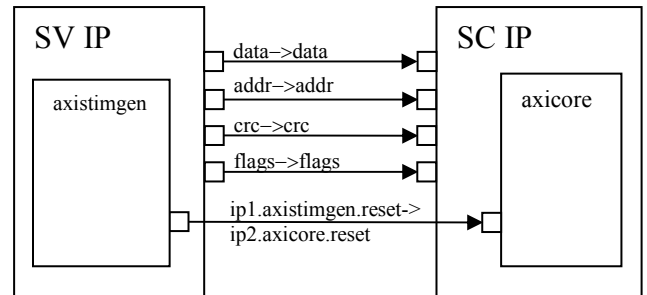

Figure 11. Connecting IP Blocks with a Special Case of Category C Connections

In such a situation, the elimination process will result in SystemC control/observe or scv_connect(). However, it may be more efficient to use SV bind construct, or direct instantiation for connections that are confined to a single design-unit in the two IPs, and use SystemC control/observe or scv_connect() for the rest of the connections.

The steps for mixing methods process are as follows:
- Search for a method for each connection separately.
- Select the most optimized method for the connection.
- Once methods have been finalized for all individual connections, create a list sorted by the methods used.
- Finally use the selected methods to make connections.

Comparison of Choices

After the elimination and mixing methods steps, designers will be left with one or more choices. They can then use details provided in Table [1], Section 2 (Methods to Connect Mixed Language IP Blocks) to select the best approach for their SoC.

4. PROPOSED MIXED LANGUAGE INTER CONNECT UTILITY

The methodology presented in this paper is used to develop a mixed language **I**nter **C**onnect **U**tility (ICU) that takes the two IP blocks written in different languages as input and automates the hook-up connection process by suggesting the most suitable approach, while taking the minimum input from the user. It also generates snippets of code that can be automatically inserted in the design to make these connections seamless and less prone to errors that may result from manual updates.

Designer Configuration File

A **D**esigner **C**onfiguration **F**ile (DCF) can be provided with the utility to automate it even further. This designer configuration file (DCF) will have information about the two regions to be connected, port-maps of the two regions, name of the output generated, and so on.

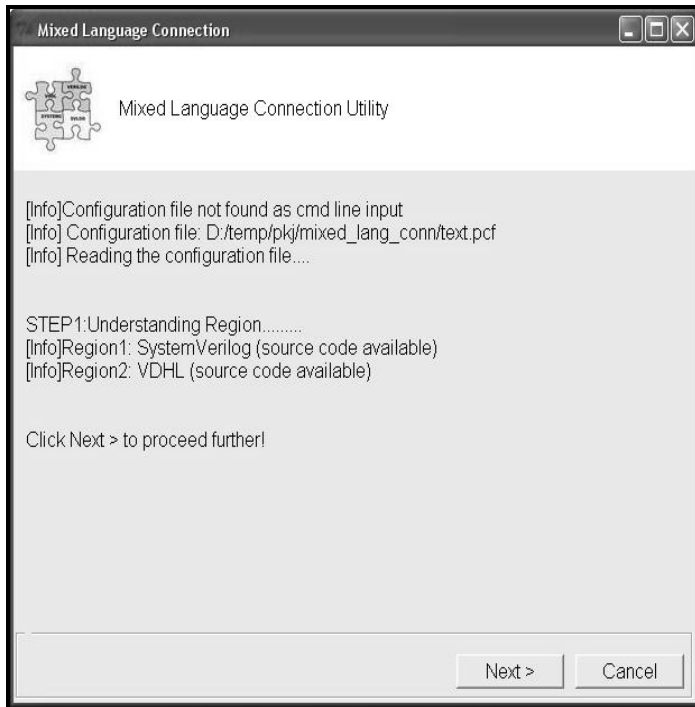Figure 12 depicts Inter connect Utility Step 1; parsing and identifying the design

Figure 12: Snapshot of ICU Step1

At the end of the ICU run, an option will be provided to store the DCF containing all the settings that were gathered or requested from the user in this run. The stored DCF can also be edited by the user to make small changes in his bindings as per the requirement for subsequent CIU runs.

Snapshots of Inter Connect Utility below show the Step 2 and Step3 process (described in section III) of analyzing and making IP connections.
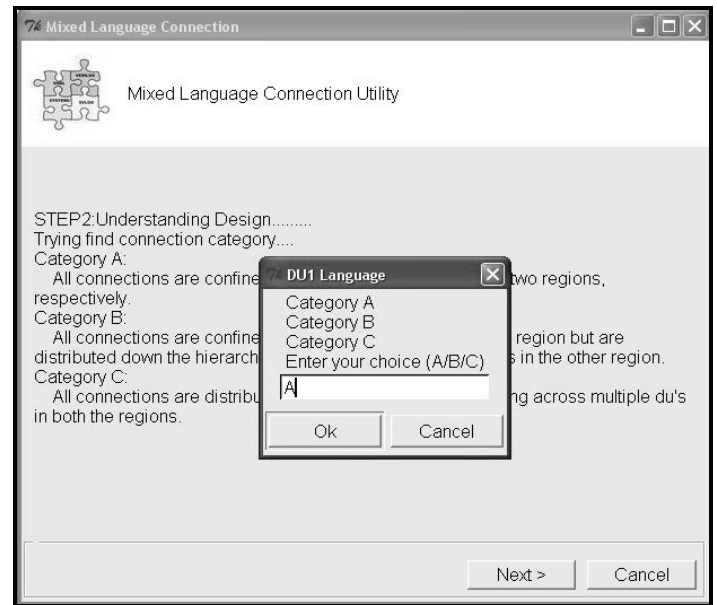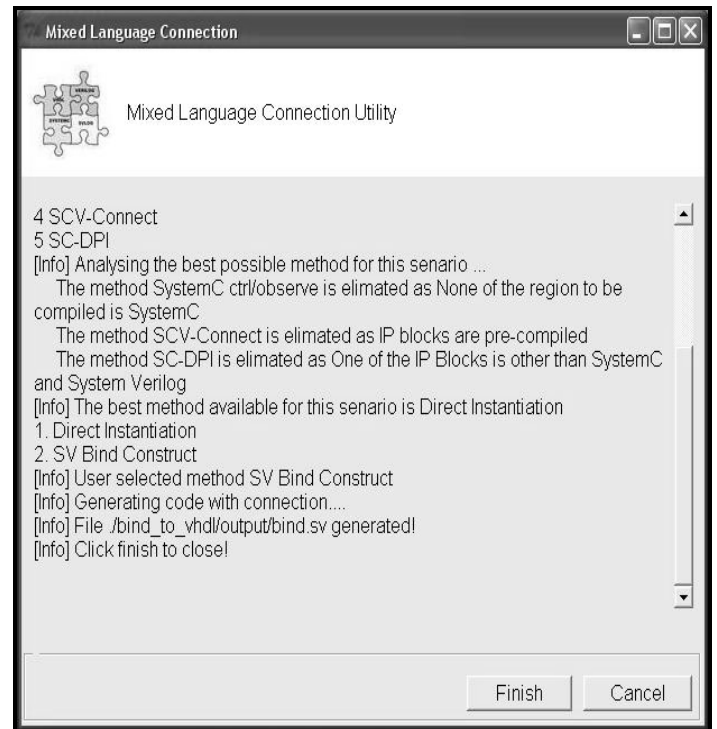

Figure 13: Snapshot of ICU Step2

Figure 14: Snapshot of ICU Step3

5. VALIDATING ICU ON A REAL WORLD DESIGN

Initial results on a small design indicate time savings in comparison to manual integration effort, with the additional benefit of removing the dependency on user's know-how of various integration methods by automatically analyzing and proposing the best option.

The methodology described in this paper is validated on standard 10 Gigabit Ethernet protocol as shown in Figure 15.

The 10 Gigabit Ethernet (10GbE) aims at promoting the use and availability of Ethernet in the WAN environment. It defines a version of Ethernet with a nominal data rate of 10 Gbit/s, ten times as fast as Gigabit Ethernet.

10GbE supports only full duplex links which can be connected by switches. Half Duplex operation and CSMA/CD (carrier sense multiple access with collision detect) are not supported.

The 10GbE standard encompasses a number of different physical layer (PHY) standards. A networking device may support different PHY types by means of pluggable PHY modules.
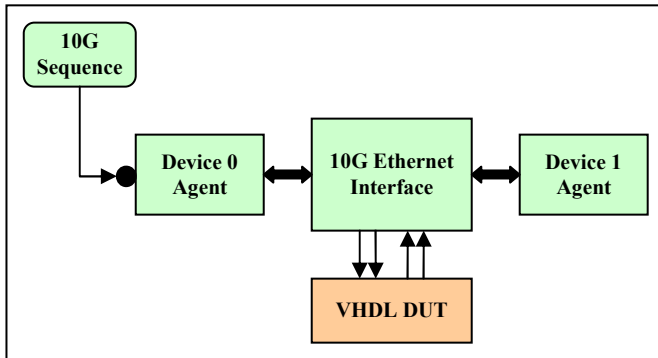

Figure 15: 10 Gigabit Ethernet Environment

We used the methodology proposed in this paper to connect VHDL DUT to the SystemVerilog wrapper. The proposed methodology indicates direct instantiation and SV bind construct as the two most optimized methods of connection for our 10 Gigabit Ethernet example. The performance details of different approaches for connecting mixed language IP is listed in Table 1 (column D).

6. CONCLUSION

The step by step methodology presented in this paper eliminates the limiting factor of IP reuse due to the complexity of mixed language designs. A new 'mixing methods' approach is also introduced to help

designers make their mixed-language connections in scenarios where all methods get eliminated or where using more than one method for integration provides a more optimized connection.

The main benefits of the proposed methodology are twofold:

1. Removing the basic issue with "how to" interconnect mixed-language IP's by analyzing several standardized/non-standardized methods and proposing the best option with the highest benefit and minimal risk.

2. Improving the productivity by minimizing issues found late during the design cycle due to incorrect interconnect approach or manual error in IP integration. The proposed utility selects and automates most of the process of mixed language IP hook-up connection. It also provides flexibility to the user to select his choice of method.

7. REFERENCES

[1] Rudra Mukherjee and Sachin Kakkar, "System Verilog – VHDL Mixed Design Reuse Methodology", DVCon 2006.
[2] Rich Edelmen, Mark Glassar, et al. "Inter Language Function Calls Between SystemC and SystemVerilog", DVCon 2007
[3] Rudra Mukherjee, Gaurav Kumar Verma, et al. "SystemC Mixed-HDL IP Reuse Methodology", IP-07
[4] Gaurav Kumar Verma and Rudra Mukherjee, "Adding New Dimensions to Verification IP Reuse", DVCon 2009
[5] Rajeev Ranjan, Homayoon Akhiani, et al. "Towards Harnessing the True Potential of IP Reuse", DesignCon 2009
[6] SystemC LRM IEEE 1666-2005 (www.systemc.org)
[7] SystemVerilog LRM IEEE 1800-2005 (www.systemverilog.com)
[8] VHDL LRM IEEE 1076-2002 (www.vhdl.org)

IMPORTANT NOTE

The utility presented in this paper is a standalone tool to help users automate connecting their mixed-language IP's. As of today, there are no plans to add this utility in Questa.