AXI4 2x2 Fabric Example

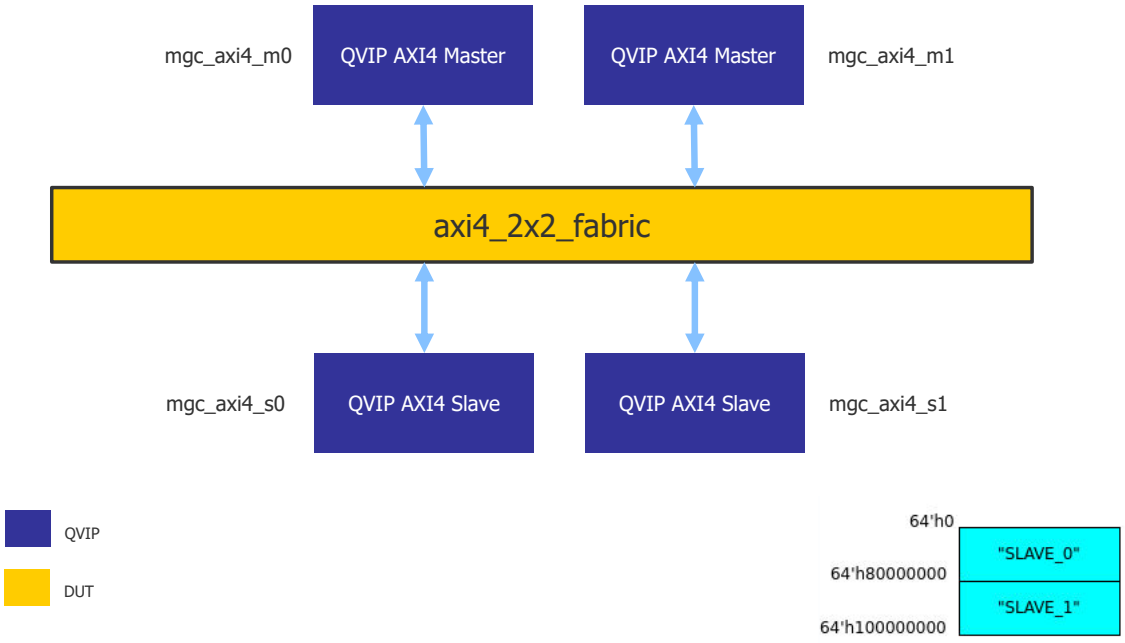

Figure 1: AXI4_2x2_Fabric DUT Connections Block Diagram

Introduction

This example is intended to demonstrate how easily one can generate and integrate Mentor Questa Verification IP (QVIP) into a UVM Framework (UVMF) environment. It consists of four AXI4 QVIPs, two Masters and two Slaves, communicating over an AXI4 Fabric. We will detail how to generate the QVIP components using our QVIP Configurator Tool, incorporate the output into the YAML to specify the environment and bench, and finally how to generate and complete the UVMF Environment and Bench.

Quick Note – Introductory Videos

Prior to getting started, it is recommended to review the 'UVM Framework - One Bite at a Time' course videos which describe the architecture, flow, generation, and use of UVM Framework testbenches. For more information, please refer to the Reference Documentation section at the end of this document.

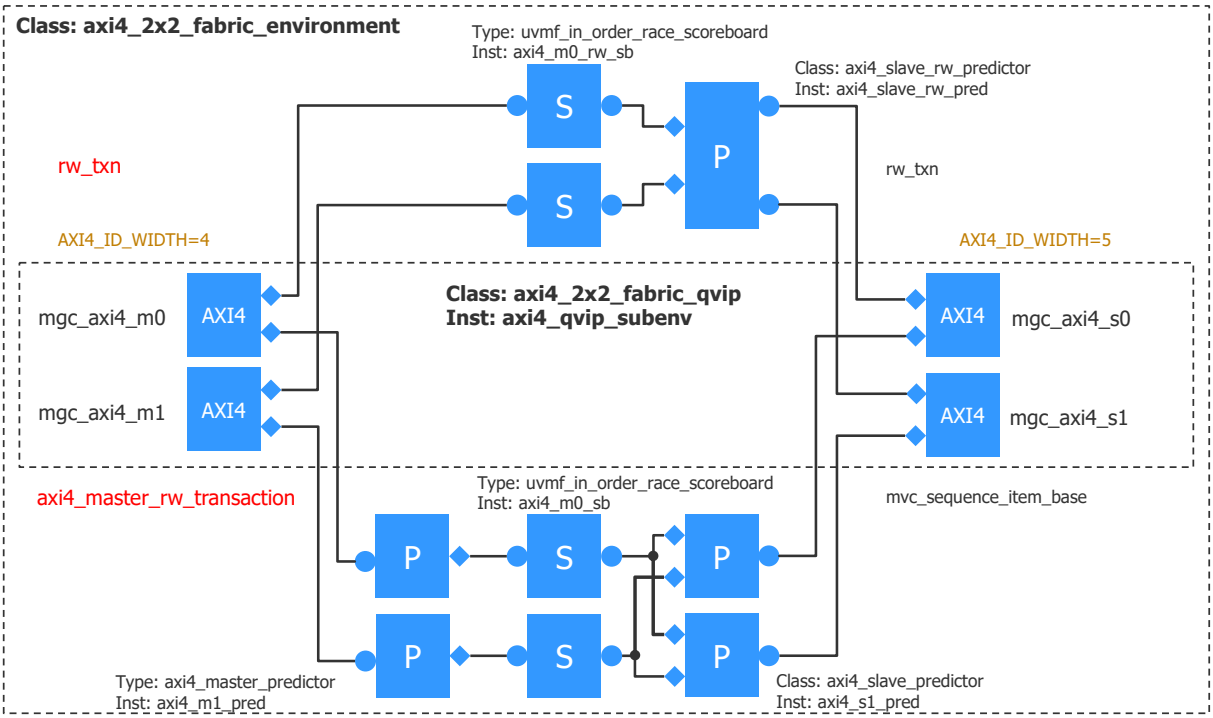

Figure 2: AXI4_2x2_Fabric UVMF Environment Block Diagram

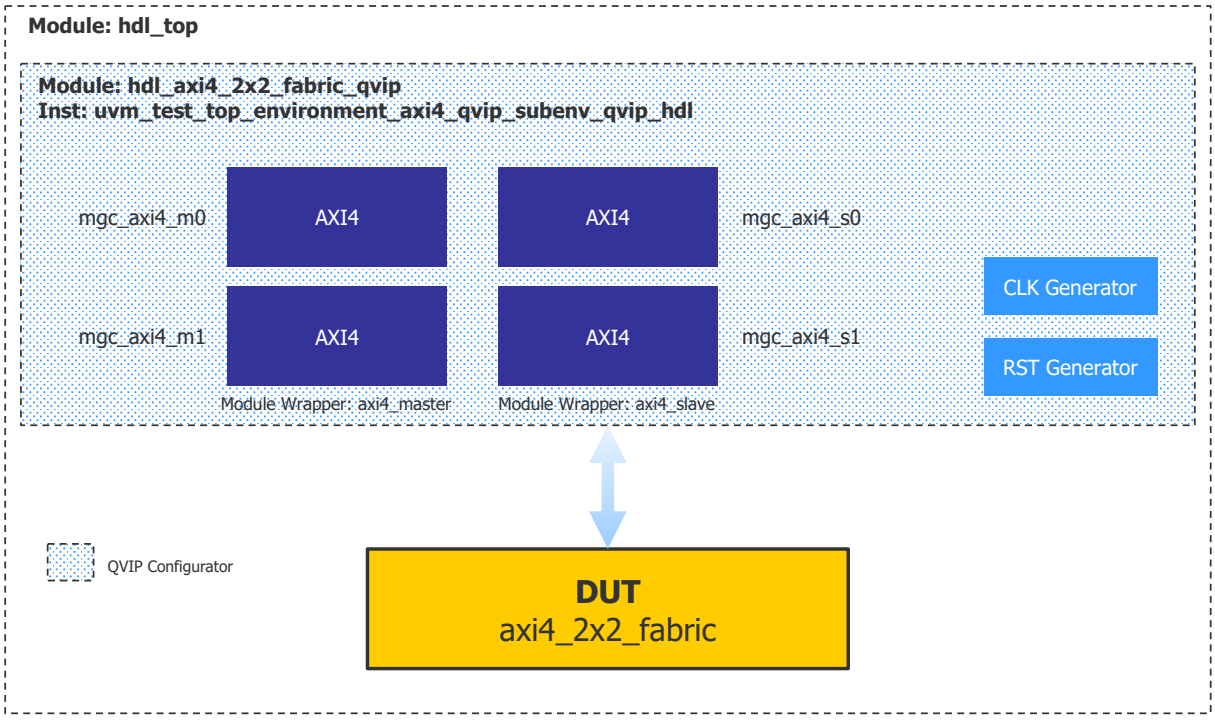

Figure 3: AXI4_2x2_Fabric UVMF Testbench Block Diagram

The advantage of using the QVIP Configurator in conjunction with the UVMF is that it provides us with the ability to quickly and easily configure, generate, and incorporate QVIP instances, as well as to define, instantiate, and connect predictors, and leverage (built-in) UVMF scoreboards

within a UVMF environment. It is also worth mentioning that the resulting bench will be block-to-top ready and contain all the means necessary for a large set of Mentor tool integrations.

Generating the QVIP Sub-Environment

Before we are able to start writing the YAML to describe the UVMF project, we first need to generate the QVIP Components that will make-up the environment. This is because in order to generate the UVMF environment, the YAML needs to know how to create, instantiate, and connect the QVIP Configurator generated code. In order to simplify this, the output of the QVIP Configurator encapsulates all of its class-based output within what we call a QVIP Sub-Environment as shown in the figure below.

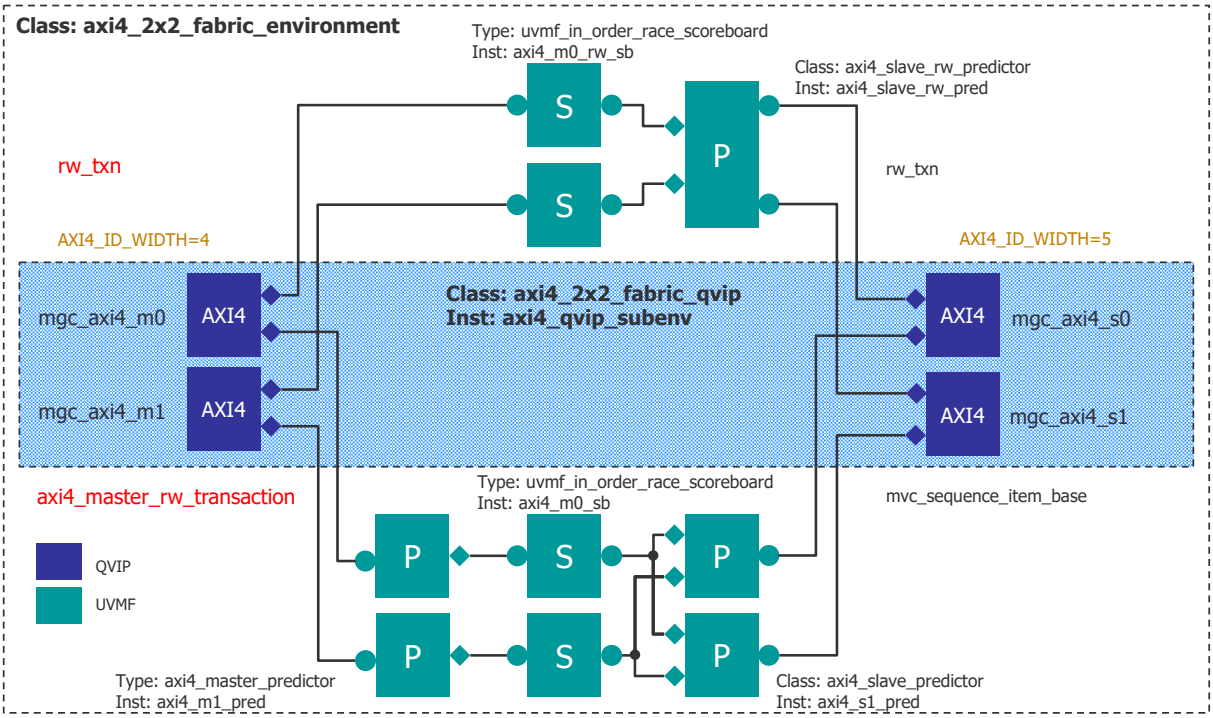

Figure 2: AXI4_2x2_Fabric Environment Showing QVIP Configurator vs UVMF Generated Components

Using the QVIP Configurator

The QVIP Configurator is located within the QVIP installation at the following location; start by launching the application.

- *$QUESTA_MVC_HOME/bin/qvip_configurator*

For more information on the QVIP Configurator, please refer to the Reference Documentation section at the end of this document.

Using the Reply Command File

Every time the QVIP Configurator is launched, it automatically creates a Command File in order for a user to quickly and easily re-generate a specific configuration. This is also extremely helpful when making slight modifications in order to avoid having to manually re-enter your commands to get back to a point where you left off.

For your convenience, we have included a QVIP Configurator Replay Command file within this example to quickly and easily re-generate the QVIP Sub-Environment. It is located in the following directory.

- *$UVMF_HOME/vip_examples/verification_ip/environment_packages/axi4_2x2_fabric_qvip_dir/axi4_2x2_fabric_qvip_dir.cmd*

For more information on the AXI4 QVIP Protocol, including instantiation and configuration settings, please refer to the Reference Documentation section at the end of this document.

Quick Note – QVIP Configurator Output

Please note that this step has already been performed and the resulting output directory has been placed in the following location. Naturally, you can always manually re-create this step and/or re-generate the sub-environment if you wish to see how it works. Also, feel free to open the Replay Command File if you wish to learn more.

- *$UVMF_HOME/vip_examples/verification_ip/environment_packages/axi4_2x2_fabric_qvip_dir*

Creating the YAML to Generate the Environment

Now that the QVIP Sub-Environment has been generated, we can move-on to defining the YAML in order to generate the project Environment and Bench.

YAML Reference Materials

As this guide is not intended to teach the user how to write YAML, we will simply direct your attention to the YAML files that have already been written for this example. For more information on how to write YAML files, please refer to the Reference Documentation section at the end of this document.

YAML AXI4_2x2_Fabric Reference Files

Please feel free to open each YAML file for a better understanding of how the environment has been defined, paying special attention to how the QVIP Sub-Environment and connections for it have been incorporated. Remember, these are files that you will normally need to define, but have been completed for you in this example.

- *$UVMF_HOME/vip_examples/project_benches/axi4_2x2_fabric/docs*
  - *axi4_2x2_fabric_bench.yaml*
  - *axi4_2x2_fabric_env.yaml*
  - *axi4_2x2_fabric_util.yaml*

Quick Note – Parameterization in YAML

Recently the capability to add parameterization to Analysis Components has been added to the YAML. This is extremely useful when dealing with parameterized transaction items. There are two main ways to use parameterization with Analysis Components.

1. Individual parameters
2. Type parameters

The first is to declare individual parameters which can each be overridden upon the instantiation of the component. The second is to declare a “type” parameter which can be overridden by a type. Overriding a type parameter is much cleaner and easier to use - and is also the recommend approach - when working with parameterized transaction items within an Analysis Component. It makes the code much cleaner and easier to understand.

Gotchas

Please note that you are declaring the parameters of an Analysis Component within its declaration within the 'util_components' YAML tag. Here you can also give each parameter a default value. You then have the option to override these parameters upon the components instantiation within the 'analysis_components' YAML tag.
Furthermore, the name of each parameter must match within the YAML between its declaration and definition/instantiation.
Finally, if you have defined parameters within the environment itself, and wish for these environment level parameters to override parameters within your Analysis Component(s), you can simply name them the same.

Quick Note – QVIP Configurator YAML

It is important to note that when generating a UVMF Environment/Bench from YAML that contains output from the QVIP Configurator, you must also include the YAML file that is automatically produced as part of the Configurator output. In this example, it is located in the following location for your reference.

- *$UVMF_HOME/vip_examples/verification_ip/environment_packages/axi4_2x2_fabric_qvip_dir/uvmf/axi4_2x2_fabric_qvip_subenv_config.yaml*

Generating the Environment & Bench

Now we are ready to generate the AXI4_2x2_Fabric Environment and Bench using the YAML files that we have created above. This is done by using the 'yaml2uvmf.py' script which is located within the install. For more information on the UVM Framework and YAML to UVMF code generation, please refer to the Reference Documentation section at the end of this document.

- *$UVMF_HOME/scripts/yaml2uvmf.py*

Running the Generators

Below is an example of how we would use the 'yaml2uvmf.py' script to generate the UVMF Environment and Bench for this specific example. Please note how we have also included the YAML output file from the QVIP Configurator as mentioned above.

- *yaml2uvmf.py axi4_2x2_fabric_bench.yaml axi4_2x2_fabric_env.yaml axi4_2x2_fabric_util.yaml axi4_2x2_fabric_qvip_subenv_config.yaml*

Adding User Code

At this point you have created a UVMF project! However, in order for the project to compile and simulate, we need to tell the scripts where it can find the QVIP Configurator generated code that we created earlier. Please look for and set the following environment variable to the 'uvmf' directory located within the QVIP Configurator output generated code.

- *$UVMF_HOME/vip_examples/project_benches/axi4_2x2_fabric/sim/Makefile*
  - *export AXI4_2X2_FABRIC_QVIP_DIR_NAME ?=*

With the above change, you should now have a working simulation. However, it isn't very interesting as it isn't really doing anything useful. In other words, it is now time for you to add what we call your "User Code". Without going into too much detail, here is a high-level summary of what needed to be done to complete this example.

- Update the Makefile to point to the QVIP Configurator generated output directory for compilation (setting the environment variable as detailed above) as well as to define an existing target to compile your RTL
- Place your RTL in the correct location to be referenced and compiled by the Makefile target defined above
- Instantiate and connect your DUT in the top-level testbench module
- Create stimulus
- Add prediction logic for each predictor

Putting It all Together

For a complete list of files that were modified/added after YAML generation, please refer to the following text document. These are all of the changes required in order to complete this example. At this point I will not go into specifics, but rather leave it as an exercise to the user to explore further. I suggest performing a 'diff' to see and understand the changes that were made for each of these files.

- *$UVMF_HOME/vip_examples/project_benches/axi4_2x2_fabric/docs/user_modified_files.txt*

Running the Example

As with all UVMF examples, this example can be run in CLI, Classic GUI, or Visualizer Interactive from within the simulation directory as follows.

- *$UVMF_HOME/vip_examples/project_benches/axi4_2x2_fabric/sim*
  - CLI Mode
    - *make cli*
  - Classic GUI
    - *make debug*
  - Visualizer Interactive
    - *make debug USE_VIS=1*
      - *Mentor recommends using Visualizer 10.7c or later for Visualizer Live-Sim Mode.*

Quick Note – Environment Variables

Don't forget to set the environment variable named $UVMF_HOME to point to the UVMF installation directory.

Conclusion

Congratulations! You now have a complete working UVMF project which includes Mentor QVIP. This example has demonstrated how to use the QVIP Configurator along with the YAML to create a UVMF+QVIP project quickly and easily. It has also highlighted some of the things that the user will need to take into consideration, including what is required after the project has been generated; adding user code. Hopefully this has been helpful and have fun exploring!

Reference Documentation

UVM Framework – One Bite at a Time (Course/Videos)

- *https://verificationacademy.com/courses/UVM-Framework-One-Bite-at-a-Time*

Questa Verification IP Configurator User Guide

- *$QUESTA_MVC_HOME/docs/pdfdocs/qvip_configurator_user.pdf*

Questa Verification IP User Guide for the ARM AMBA 4 AXI Protocol

- *$QUESTA_MVC_HOME/docs/pdfdocs/qvip_axi4_user.pdf*

UVM Framework Code Generator YAML Reference

- *$UVMF_HOME/docs/UVMF_Code_Generator_YAML_Reference.pdf*

UVM Framework User's Guide

- *$UVMF_HOME/docs/UVM_Framework_Users_Guide.pdf*