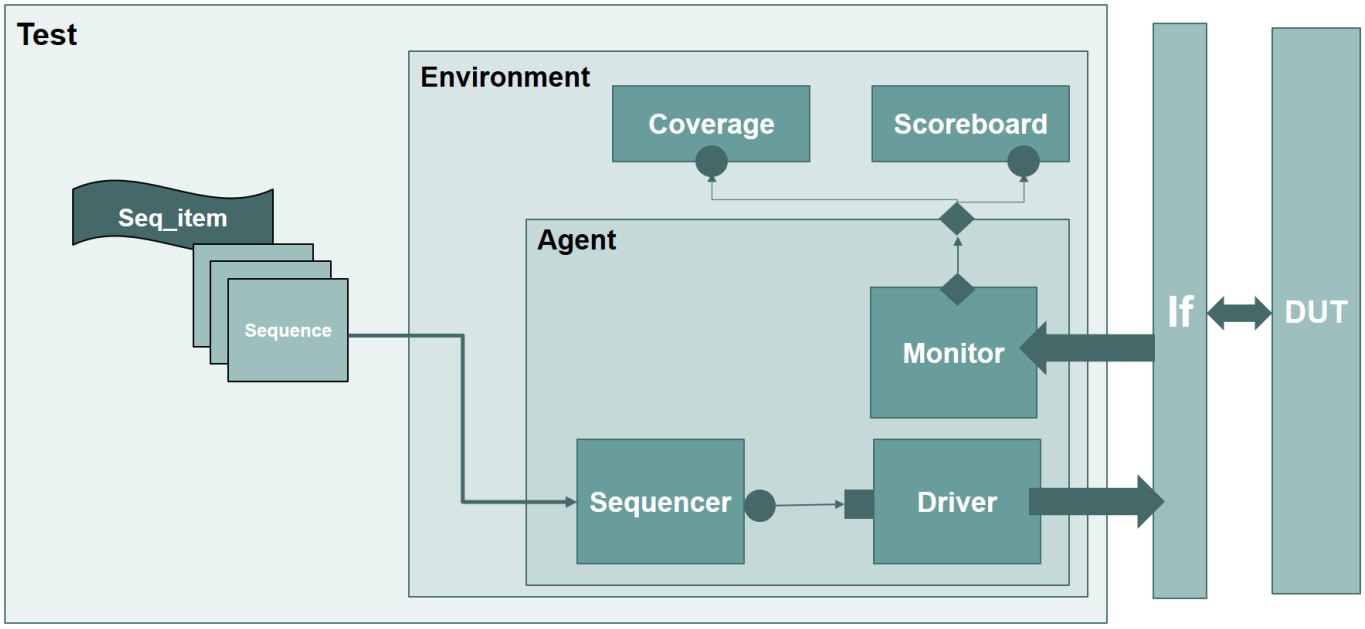
AXI4-Compliant Memory-Mapped Slave Verification Using UVM

According to the previously provided specifications for AXI4 Compliant Memory design in the following Link: **AXI_Specs**

Requirement

- It is required to build a full UVM Verification environment for the AXI-Memory Mapped design using all learned components and objects for the UVM and applying all concepts we walked through during the UVM learning phase.
- U can use the same design you used with SystemVerilog project after your fixes for the detected bugs in the previous project.
- For simplicity and to have a chance to go smoother with UVM you need to test the basic functionalities for the design of **READ** and **WRITE** operations without the **Brust** feature.
- U have to reach 100% Pass rate, 100% functional and code Coverage with giving acceptable justifications in case you miss the full percentage for the coverage.
- Practice Assertions with the built UVM Environment
- You have to implement full structure for the UVM and all components and objects you learned.
- Use the appropriate UVM Phasing and reporting with your environment.
- U have to draw the UVM Structure u used and built in your Verification process like below



Extra Requirement (Optional)

- Verifying the **Brust** Feature for The AXI-Memory mapped design.

- U can practice the Factory overrides feature by making a sequence called **error_sequence** which inherits from the base sequence you implemented and that error sequence is meant to apply stimulus exceeding the address margins to test the response error behavior. And create a new test that called error test in which you will override the base sequence type by the error sequence type and trigger the sequence to send the error stimulus and check the error behavior.

- Practicing the Passive Agents and making a passive agent for has a monitor that monitors the input data only and the other active agent has the monitor that monitor the output data only and both monitors pass the value to scoreboard through two analysis exports which are connected to two tlm_fifo. The SCB will handle them in different ways, for the analysis export that reads the data from the passive agent monitor it send its data to the golden model to evaluate the expected output, and for the analysis export that is connected to output monitor in the active agent it takes the output value from it and use the above collected info the actual and expected and make the proper checks with proper uvm reporting.

- Any other enhancements on the UVM structure you are building it is up to you and you are free to practice all the topics you learnt take the project as a chance for strengthening your knowledge.

Deliverables:

U have to deliver **two separated files**

- Zip Folder contains:
  - All the implemented uvm files and the **design files** and **run.do file**.

- **Separated PDF** contains:
  - Snippets for the implemented code.
  - Snippet for the waveform shows the different testcases clearly.
  - Snippets for the logs show the all printed values using UVM reporting.
  - Snippets for functional coverage report.
  - Snippets for code coverage report for all parameters [Line / toggle / branch /condition/…]
  - Snippets for Assertions Coverage report.
  - Snippet for Do file you have used for automating the process.

The delivered zip file must be named like **your_name_uvm_project.rar** for example:
Hassan_Khaled_uvm_project.rar also the PDF file **Hassan_Khaled_uvm_project.pdf**

Good Luck