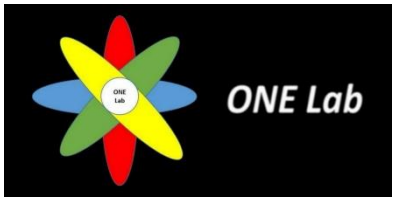

Faculty of Engineering
Cairo University

Project title:

Generic UVM for Soft Processors

Task #4: Implementation (1st Milestone)

Under the supervision of:

Dr. Khaled Salah (Mentor Graphics)

Dr. Hassan Mostafa (Cairo University)

By:

- Kholoud Mahmoud
- Randa Ahmed
- Karim Ayman
- Mostafa Ayman
- Waleed Samy Taie
- Yasser Ibrahim

Table of Contents:

Table of figures:

Abstract:

At the end this task, we've successfully reached our first milestone of the implementation of our generic UVM: a fully functional generic UVM to test the functionality of an instruction with our three different DUTs.

The following milestone will include the rest of the instructions to finish the current layer of our general verification plan: Functionality Testing.

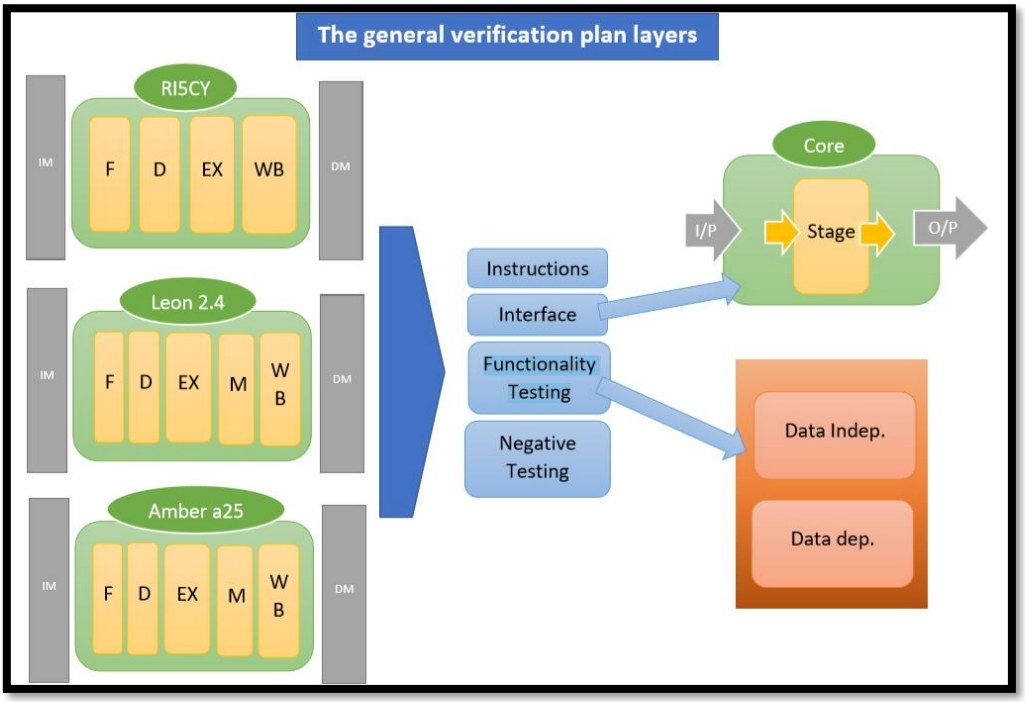

Figure #1: The general verification plan layers

To summarize the implementation of our generic UVM, the user can use our generic UVM with any core (soft processor) of our three cores after attaching only 2 things to the test bench:

- The desired **core package**: includes all the core instructions and its format mapping).
- **The interface** of the chosen core: (1) includes the ports of the top-level module (core interface), (2) each interface has functions that the driver use to drive instructions or data to the DUT and (3) deals with clk and timing to deal with each processor regarding the timing constraints.

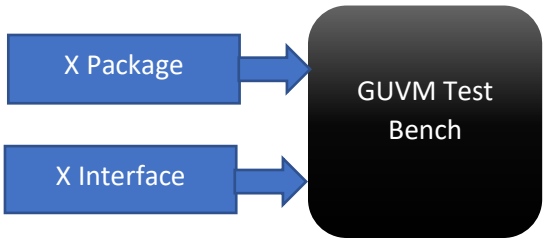

Figure #2: How the user can use our generic UVM

Generic UVM Test Bench Architecture:

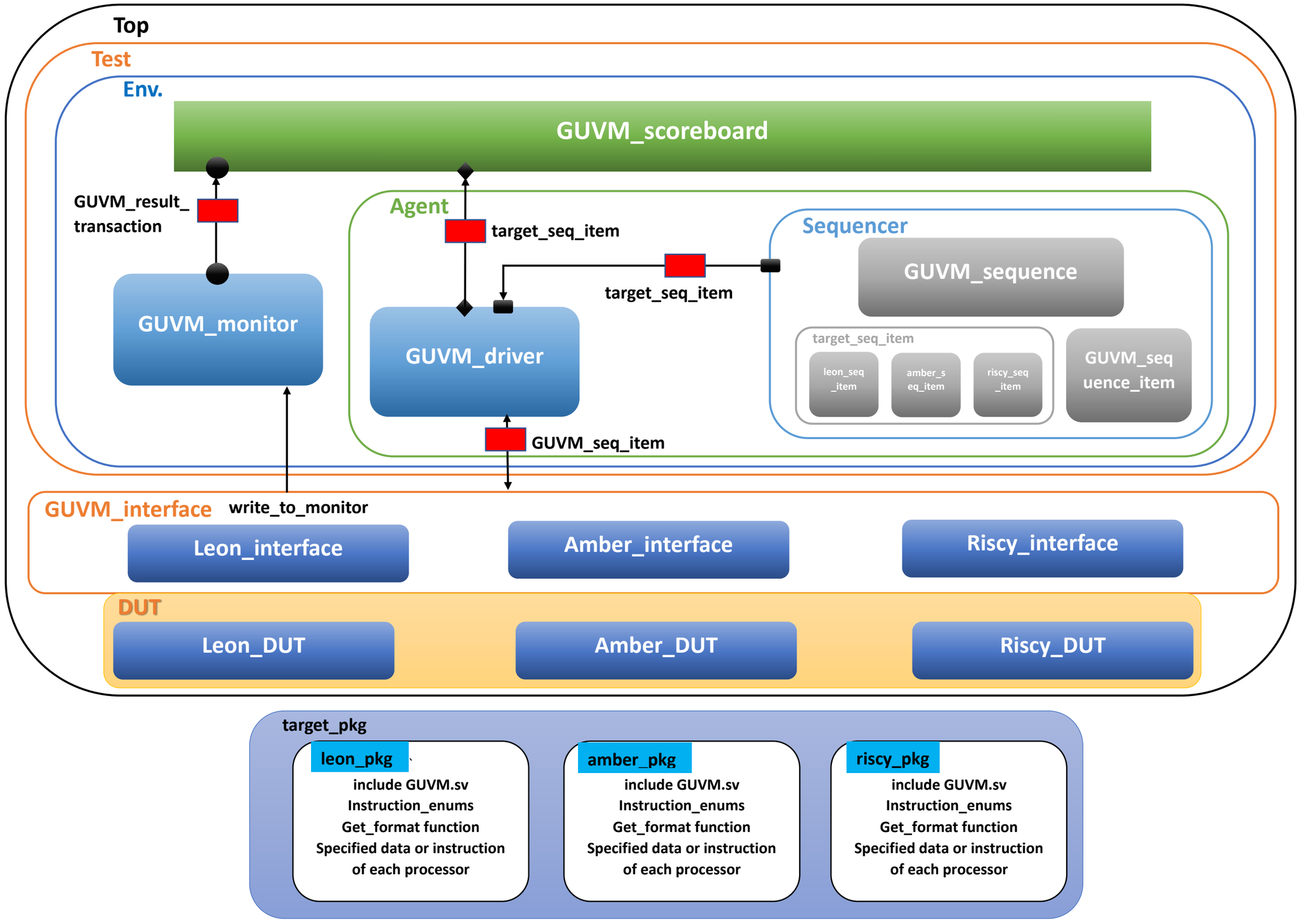

Figure #3: Generic UVM Test Bench Architecture

Generic UVM Components:

Test (GUVM_test)

The GUVM_test is the top-level UVM Component in the UVM Test bench.
The GUVM_test Instantiates GUVM_env, configures it (via factory overrides or the configuration database), and applies stimulus by invoking Sequences through the environment to the DUT.

Environment (GUVM_env)

The environment is the higher-level component of the generic UVM test bench. The configuration of the environment enables customization of its topology and behavior.
The GUVM environment is the generation of the constrained random traffic to stimulate the DUT processor, monitoring of the DUT processor response and checking of the ongoing traffic.

Agent (GUVM_agent)

The agent is an active agent, such that it stimulates the DUT by driving transactions according to the specified test scenario.

Sequencer

GUVM_sequence

A simple sequence was made to test arithmetic instructions, first the instruction is randomized, then the sequence starts sending load instructions to fill the source registers. After loading the registers, the sequence sends the instruction under test, then sends a store instruction to get the result to the memory pins, which the monitor can read from.

GUVM_sequence_item

It handles the randomization of instruction and data.
It has variables to save the data of instruction's input operands.
It has randomization functions.

target_seq_item

There are three target sequence items one for each processor but only one of them will be compiled in the simulation.
Each one has the three cores instruction format variables.
Each one helps to construct loads and store instructions, which needed to deal with instruction data.

Driver (GUVM_driver)

The driver receives the sequence item transactions from the Sequencer and send it on the DUT Interface. GUVM_driver first resets the DUT, then sends the data and the instruction to the DUT, as it converts the transaction-level stimulus into pin-level stimulus.

Monitor (GUVM_monitor)

The monitor samples the DUT interface and captures the information (converting pin-level activity to transactions) there in transactions that are sent out (broadcasting) to the rest of the UVM test bench for analysis using a TLM analysis port. The UVM Monitor delegate the transactions produced to the scoreboard connected to the monitor's analysis port.

Scoreboard (GUVM_scoreboard)

The scoreboard checks the behavior of the DUT processor. It receives the transactions that carries inputs and outputs of the DUT processor through GUVM_driver and GUVM_monitor analysis ports, then it checks the signals and reports the result.
GUVM Scoreboard gets the instruction from the driver, the randomized data from the sequencer, and the output signals of the DUT from the monitor.

Figure #4: Generic UVM Components