# Lab. 1. Development of the test environment in the SystemVerilog language.

## Introduction

Create a new directory "~/VDIC" (for the whole course). Create a GIT repository in VDIC directory, on any server where the lab results will be stored.

A link to the repository should be included in the report. If the repository is private, it should be made available to the user robert.szczygiel@agh.edu.pl

In the repository, we create separate branches for each exercise named "lab01", "lab02", etc.

Synchronization with the selected branch should allow you to run the simulations required for a given exercise.

## Exercise topic

Exercises include the development of the test environment (testbench) in the SystemVerilog language for the given module of the ALU arithmetic and logic unit with serial input and output. The testbench should be developed from the enclosed example.

## The course of the exercise

The exercises consist of the following elements:

- analysis of the presented example of the test environment; starting the simulation,

- developing a test specification for ALU,

- development of a testbench for ALU,

- uploading the results to UPEL.

## Analysis of the test environment

Create another director for the exercise named "VDIC/lab01". Create the git branch "lab01". Download the file lab01_example_2020.zip from UPEL, unpack and view the content. The simulation start file is **run_xrun.sh**. Check it out. Please make sure you understand the functions of all the attached files.

To set the paths to the simulator, execute the **source** ... command specified at the beginning of the run_xrun.sh file in the terminal.

Run the script with no arguments and see what results you get for **code coverage** and **functional coverage.**

Note that the run_xrun.sh script runs two tools:

- xrun - Cadence simulator from the XCELIUM package, and

- imc - Incisive Metric Center from VMANAGER package, the coverage data analyzer

Run the simulation in the graphical mode as described in the file run_xrun.sh. Review the waveforms and the schematic.

View coverage data using Incisive Metric Center (imc) - the run command is printed each time the run_xrun.sh script is run without graphics mode.

## ALU module analysis

Download the spec file for ALU from UPEL and study it. Also download the ALU model (mtm_Alu.vp file) and review. Note what modules the entire unit consists of. In order not to suggest specific solutions when building the tests, the ALU sub-modules were encrypted (which does not interfere with the simulation).

## *Development of the test specification*

*Based on the ALU specification and the example given in the lecture, develop a test specification. Write down a numbered list of the tests you think need to be performed to consider the ALU fully tested. Pay special attention to testing all functionality (all arithmetic and logical operations, running all flags, etc.), checking the results for the extreme argument values, and checking the error handling in data transmission.*

## *Opracowanie testbench'u ALU*

*Develop the alu_tb.sv testbench for the ALU module. According to the example, it should contain at least three independent parts:*

*- **tester** - responsible for the generation of test patterns, based on randomly generated operations; develop appropriate functions for generating random numbers in such a way as to increase the probability that the arguments of the operation will have minimum and maximum values,*

*- **coverage** – containing covergroup declarations covering the tests, following the previously developed specification. ATTENTION: This module must not use ALU outputs! The functional coverage of the test must be 100%.*

*- **scoreboard** – monitoring the correct operation of ALU; Use SystemVerilog queues to model the expected ALU response NOTE: the module must not use data from either the tester or coverage; can only use data from ALU input and output lines.*

***All these parts of the testbench are to be completely independent (they cannot use their own variables).***

*Create a **run_alu.sh** script for your testbench to run the simulation based on the script run_xrun.sh*

*Note: the testbench should not generate on the output:*

*- unjustified warnings or errors,*

*- data for debugging.*

*The testbench should display "PASSED" or "FAILED", depending on the simulation result.*

## *Exercise results*

*Upload the file **lab01_alu.zip** to UPEL, containing:*

*- **alu_test_spec.pdf** - test specification for ALU; in the specification, please include a link to the GIT repository,*

*- **alu_tb.sv** - developed testbench,*

*- **run_alu.sh** - script to run simulations.*

***Note: results should be uploaded to UPEL before the next lecture. Any delays will result in a score of 0 points.***