2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA
SHANGHAI | MAY 26, 2021

accellera SYSTEMS INITIATIVE

inspur 浪潮

Unified Automation Verification Management Approach

Liu Wenbo, Xi'an R&D Center, Inspur, Xi’an, China (liuwenbo03@inspur.com)
Tian Libo, Xi'an R&D Center, Inspur, Xi’an, China (tianlibo@inspur.com)
Shao Haibo, Xi'an R&D Center, Inspur, Xi’an, China (shaohaibo@inspur.com)

Introduction

- Overview of the main contents: What does this paper talk about?
- Overview of background : Why do this work?
- Unified Verification Management Tool
  - Basic steps for manage verification process
  - The basic functions of the tool
- Reusable testcase build method
  - How to implement this method
  - Show some sample code

The Main Contents of This Paper

- This paper introduces a verification management tool that unifies the process construction of front-end verification, prototype verification and post-silicon validation in order to free verification engineers from the complicated process development and enable them to focus on the chip function itself so as to better complete the most important work. It realizes the functions of test case management and automatic execution of test cases.

The Main Contents of This Paper

- Present a method of reusing test cases between front-end verification, prototype verification and post-silicon. The use of test case reuse method can significantly improve the efficiency of use case management in different verification stages and reduce the workload of test case development on the basis of the verification management tools described in this paper.

Background

- SoC technology has made great progress with the progress of semiconductor manufacturing technology. The size and complexity of chips have increased exponentially.
- The current verification of SoC chips is facing more difficulties and challenges, and the verification technology of SoC chips based on FPGA is one of the important ways to solve this problem.
- FPGA can accelerate verification and truly simulate the working state of the actual SoC chip, so it can make up for the defects in the simulation phase, thus greatly improving the success rate of taping.

Background

- The prototype verification and post-silicon verification of the chip design process can effectively ensure the quality of the chip.
- Improving the work efficiency and automation of these two stages can free verification engineers from the complicated process development and enable them to focus on the chip function itself so as to better complete the most important work.
- Considering that there are basically the same requirements for test case management and regression testing process in different stages of verification, a verification management tool is developed.

Background

- This paper focuses on the regression test automation.
- Introduces this verification management tool, which unifies the process of front-end verification, prototype verification and post-silicon validation.
- It realizes the functions of test case management and automatic execution of test cases.

Background

- Because the prototype verification and post-silicon verification sometimes are very consistent with the front-end verification in the test scenario, a test case reuse design method is developed from the point of view of increasing the reusability of test cases.
- This paper will introduce some designs to improve reusability in the front-end verification process based on a unified method. The front-end test cases designed based on this method can be reused in the prototype verification and post-silicon verification phase. After all, it can significantly save the time of verification use case development.

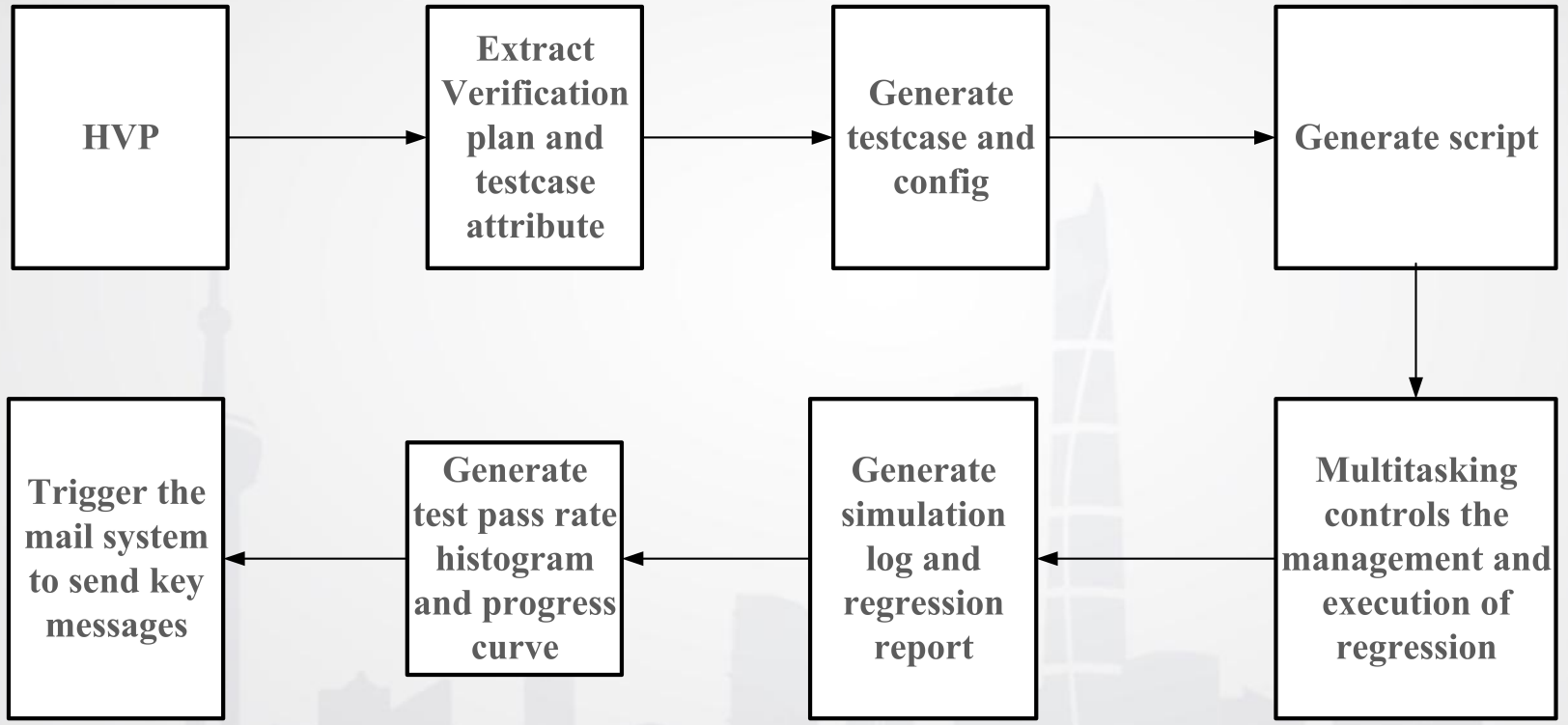
Unified Verification Management Tool

- It uses a unified verification management tool to deal with different stages of verification work.
- Based on this method, the management and automatic execution of test cases can be realized.
- This method divides the execution process of test cases into three stages: verification preparation, verification execution and post-verification processing, which has a high degree of expansibility.
- Personalized processes and functions can be flexibly expanded for different stages of different test cases.

Unified Verification Management Tool

- The detail include:
- Define the attribute information of each test case
- Automatically generating the template and the configuration file of the test case according to these attribute information.
- Build a visual operating environment, and uniformly managing the test cases of multiple modules.
- Complete automatic regression testing, analyze the result log, and generate verification report.

Unified Verification Management Tool



a. *Figure1. Testcase regression system flow*

Unified Verification Management Tool

- ***Define the basic configuration information of the tool***
- It is necessary to define some path information in advance to execute and store files when using the tool. Its form is shown in the pseudo code below. For example, *GL_SRC_DIR* represents the source file location of the test case. The details are as follows:

```
GL_SRC_DIR = $WORKAREAD/verify_repo/soc
GL_HVP_DIR = $WORKAREAD/verify_repo/etc/vplan
GL_SIM_DIR = $WORKAREAD/playground/toolrun/simulation/soc
```

Unified Verification Management Tool

- ***Definite HVP***
- HVP, as the only input source, contains the necessary attributes of the test case definition.The detail is as follows:

```
plan CPU_VPLAN;
attribute string owner       = “Zhang San”;
attribute string module_name   = “cpu”;
feature VO_CPU_IRAM_TEST;
description = “test mem size”
    measure test_status, test_status.percent.PASSED test_status;
        source = “cpu_iram_size_test”
    endmeasure
endfeature
```

Unified Verification Management Tool

- ***Extract test cases and add corresponding methods***
- They automatically extract the information from the HVP and build a data structure for all the test cases it contains, in order to save the properties of each test case and display it in the tool window when users select the specified HVP through the tool interface.
- In the meanwhile, a variety of methods are provided for each test case, including test case file generation, use case execution, log viewing, and so on.

Unified Verification Management Tool

- ***Generate test case simulation configuration file***
- The simulation is divided into three stages:
  - pre-processing
  - Running
  - post-processing.

Unified Verification Management Tool


```
pre_cfg:
target_name:
  - target_name: example_target_name
pre_cmd_list:
  - pre_cmd_list:
- example_pre_cmd1
- $(sim_dir)/example_pre_cmd2
run_cfg:
run_opts:
  - run_opts: “example_opt1 example_opt2”
post_cfg:
post_cmd_list:
  - post_cmd_list:
    - $(sim_dir)/example_post_cmd1
    - example_post_cmd
```

Unified Verification Management Tool

- ***Generate Shell script***
- Each test case generates a corresponding Shell file.

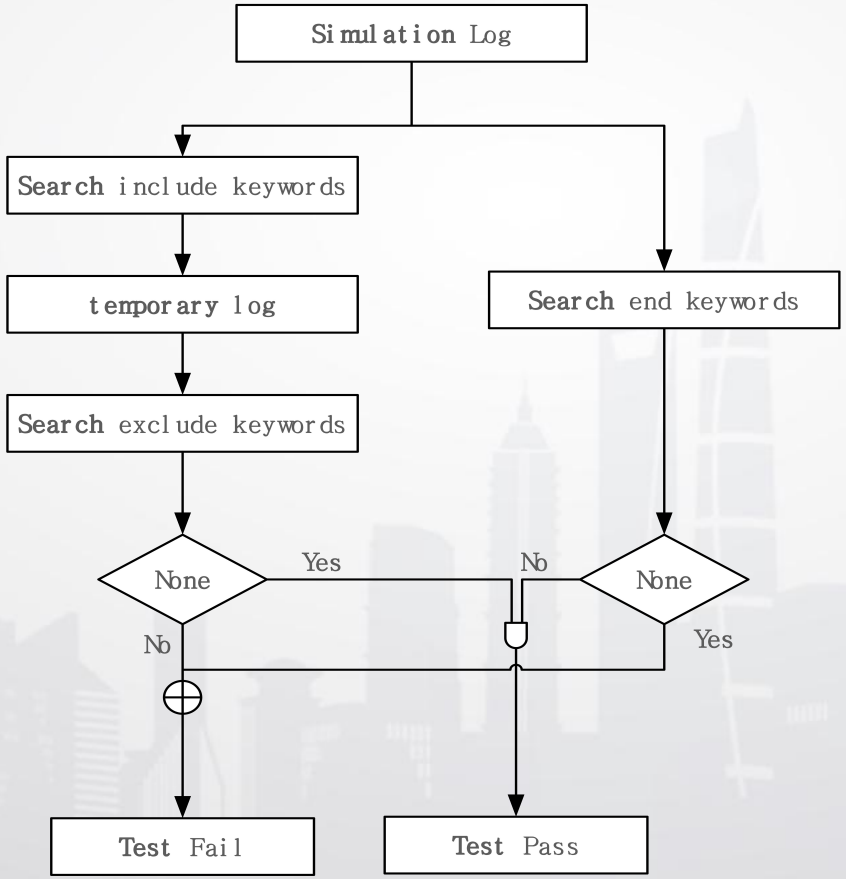
```
echo “pre-processing”
module load python/3.7.5
cp –fr /home/cadadm/scripts/* /nfs/simulation/example
ln –sf /home/cadadm/etc/*.sh /nfs/simulation/example
echo “runing”
python ./run_test.py
echo “post-processing
perl ./check_result.pl
```

Unified Verification Management Tool

- ***Regression management***
- When multiple test cases are submitted, the number of tasks to be run can be limited according to the maximum number of tasks set.
- The status of each test case can be monitored in real time. Once a task is completed, the remaining tasks to be submitted will be executed until all regression test cases are completed.

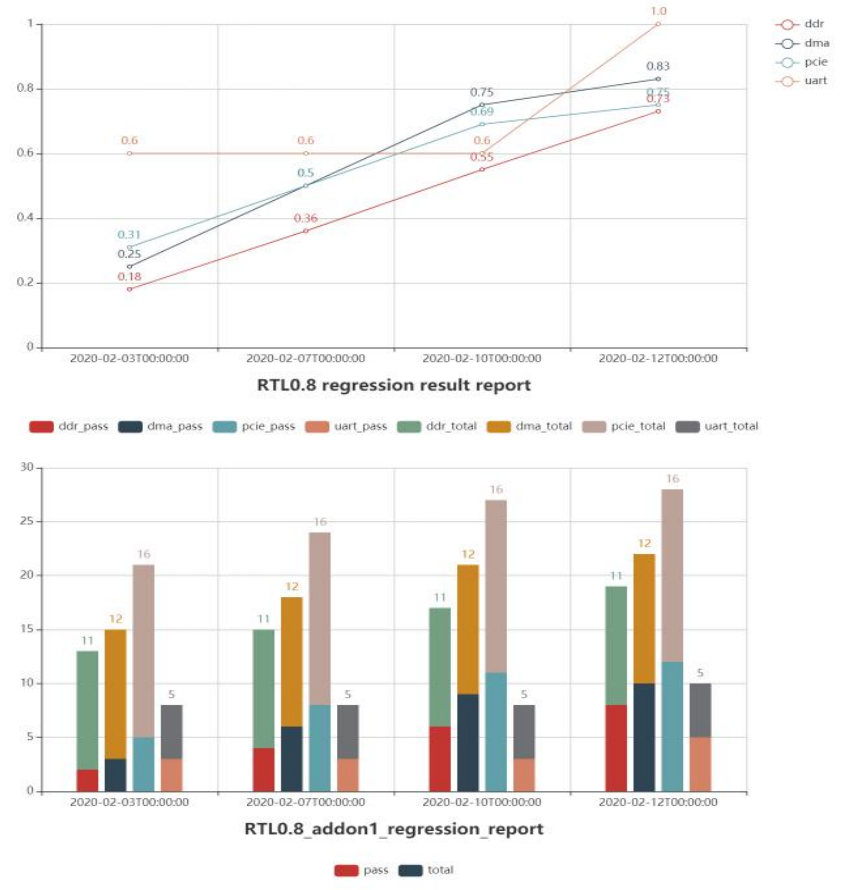
Unified Verification Management Tool

- ***Check the simulation log to determine the result***



Unified Verification Management Tool

- ***Generate reports***



- Users can generate the results of all test cases through the UI, including the HVP, maintainer, run time, results and other information.
- According to the log information, users can draw the bar chart, pie chart to show the proportion of different priorities, and the test pass ratio of different modules, and compare the results of each regression curve.

Reusable testcase build method

- ***WHAT & WHY***
- This paper intends give a method to improve the reusability of test cases in view of the consistency of verification scenarios in the three stages of front-end verification, prototype verification and post-silicon validation.
- This method can complete most of the development of prototype verification test cases during the front-end verification, and the test cases developed at this stage will be reused in the post-silicon verification phase.
- This method can significantly reduce the workload of test case development in prototype verification and post-silicon validation phase.

Reusable testcase build method

- ***HOW***
- We need to find something in common among the verification environments of front-end verification, prototype verification and post-silicon verification by considering the reusability of test cases.
  - Prototype & post_silicon validation:
    - Send the test incentive to the DUT through the real interface.
    - Load FW from external storage and executes it.
  - Front-end verification:
    - Simulate the incentive process, such as issuing various types of control/data plane configuration according to the software flow.

Reusable testcase build method

- ***HOW***
- In this process, the incentive content of front-end verification is highly consistent with prototype verification and post-silicon verification.
- We can unify the implementation methods of different verification stages in this process, to achieve the reuse of test cases in different verification stages.

Reusable testcase build method

- There are many ways to realize this stimulation process. C language is used to write test incentives in front-end verification from the point of view of maintainability and expansibility.
- The method of DPI provided by SystemVerilog language is adopted to combine the test incentives written in C language with the verification platform written by SystemVerilog.
- The following is a simple example to illustrate this design idea and the basic implementation method. The code example can be seen as follows.

Reusable testcase build method

SystemVerilog Code:

```
import “DPI-C” context task
dpi_c_thread();
class dpi_test extends base_test;
  …
 task
dpi_test::run_phase(uvm_phase
phase);
   …
   dpi_c_thread();
   …
 endtask : run_phase
endclass
```

C Code:

```
void dpi_c_thread(){
  config()
}

Void config(){
  reg_write(addr_1, value_1)
  …
  reg_write(addr_n, value_n)
}
```

Reusable testcase build method

- The basic function task, method written by SystemVerilog is also called using DPI in dpi_c_thread as shown below.

```
export “DPI-C” reg_write = task reg_write(int addr, int
value);

task reg_write (int addr, int value);
  /*configuration procedure*/
  …
endtask
```

Conclusion

- This paper implements a unified verification management tool. It can not only improve the work efficiency, but also reduce the repeated investment in verification management in different verification stages.
- And this paper also intends give a method to improve the reusability of test cases. It can reduce the workload of test case development for verification engineers at different verification stages.
- We will consider how to further improve the methods and tools of cooperation between different verification stages based on the view that unifying the tools and methods of different verification stages can reduce repetitive investment and improve work efficiency.

Thanks!