Simulation Guide

SystemC Verification Environment: Simulation Guide

(ver1.5)

Summary

This document describes the method to run simulation and check simulation result using SystemC verification environment.

Relative Document

Test-pattern creating guide
Verification Environment Generator Guide

Contents

Index of Figure

Index of Tables

1 Introduction

This guide helps users run simulation and check the simulation results and generate reports by using tools/scripts of SystemC verification environment.

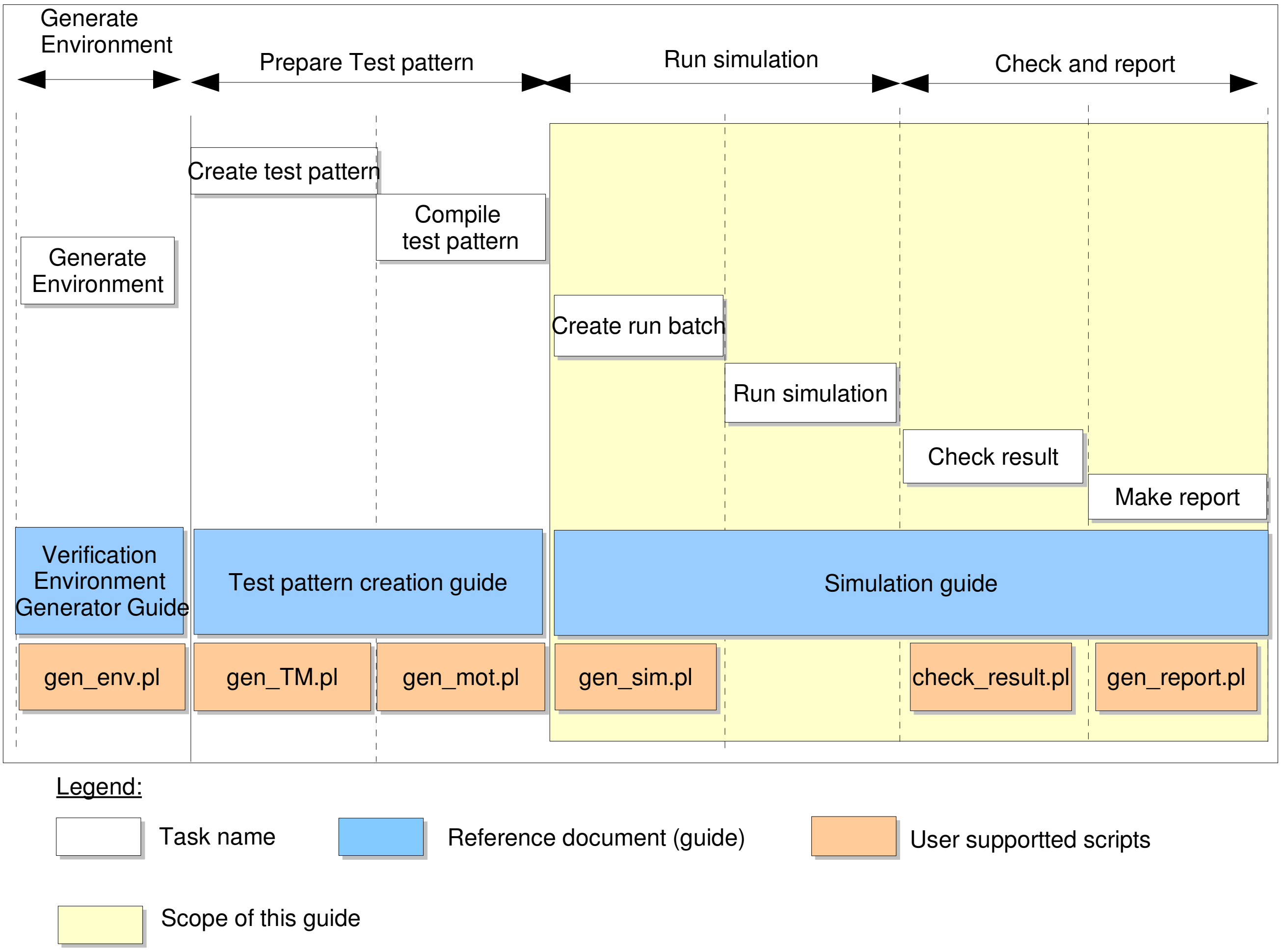


Figure 1.1: Verification Operation flow

Explanation: After creating test pattern data, designer needs to run and check simulation result in the next phases: “Run Simulation” and “Check and report”. These phases include: Create run batch, Run simulation, Check result and Make report. There are some tools along with each step that can support designer to boost the implementation time. The detail of each tool is described in the following chapters.

2 Operation flow

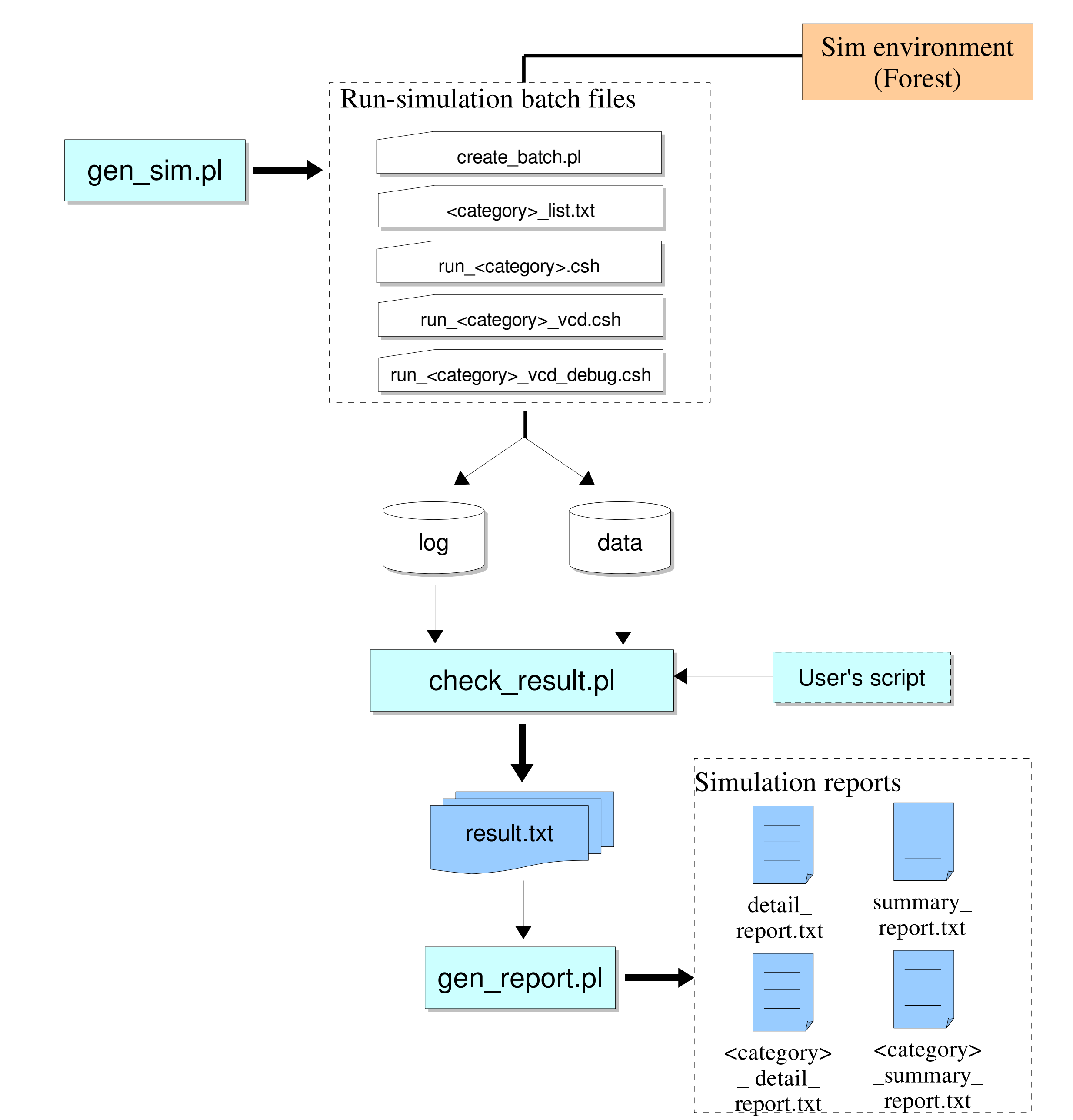


Figure 2.1: Run simulation Flow.

Explanation:

- At the beginning this operation, gen_sim.pl will receive TM_path.txt info file which is an output of gen_tm.pl script (test-pattern generating script). gen_sim.pl will create run-simulation batch files for running simulation. These batch files will base on list of enable TMs to generate run simulation batch file and run this file automatically. The run simulation batch file contains command line to run TMs of each category or all of test pattern which are enable. The input of gen_sim.pl are the links to simulation execution file, configuration file (cci, Forest_SCI.txt). Simulation can be done by running these batch files. The result of running simulation are log files and dump files (if any) that are stored in [log] and [results] directory respectively.
- After running simulation, check_result.pl script will be run in background automatically to

check the simulation result. The operation of check_result.pl script file are searching the PC break point in log file and comparing between expected data and detected data (if any) to judge the result pass or fail. The results are recorded into “result.txt” files which are stored in [results/category] directory. User can define user script to check log files and dump files (will be described later).

- If user wants to make reports from the result, he needs to run gen_report.pl script for this purpose. gen_report.pl based on “result.txt” files to create reports which are stored in [reports] directory. Some report contents can also be added to some sheets in the checklist.

1 How to run simulation

“gen_sim.pl” script supports user to generate run-simulation batch files.

1.1 gen_sim.pl description

Type	**Content**
Script name	gen_sim.pl
Language	Perl version 5.8.5 or later
Purpose	This script is used to create run simulation batch file.
Input file	TM_path.txt, this file is automatically read by this script
Output file	Batch file to run all test pattern “run_all.csh” stored in [sim] directory and batch file to run each of category (TM set) “run_<category>.csh” stored in [sim/<category>]” e.g run_initialize.csh is used for checking Initialize test pattern set and it is stored in [sim/Initialize] directory, two kind of running will be generated , batch file for normal running e,g: run_all.csh and batch file for waveform dumping running e.g: run_all_vcd.csh <category>_list_file.txt : list contains enable test pattern create_batch.pl : script to generate run simulation batch file
Usage	perl gen_sim.pl -exe **<sim_exe_path_fn>** -cci **<cci_path_fn> -[option]**
Argument	<sim_exe_path_fn> : path to simulator execution file **from the “sim” folder**. e.g : ../../tb/SCForest/forestRP1.exe <cci_path_fn>: path to cci file e.g : ../../tb/SCForest/config/RP1.cci
Option	**“-help”** : option for help – Default value: none **“-stim”** : indicates that Simulation Environment is Stimulus Environment (Stimulus model plays a role as CPU)

Table 1.1: gen_sim.pl script description

Note:

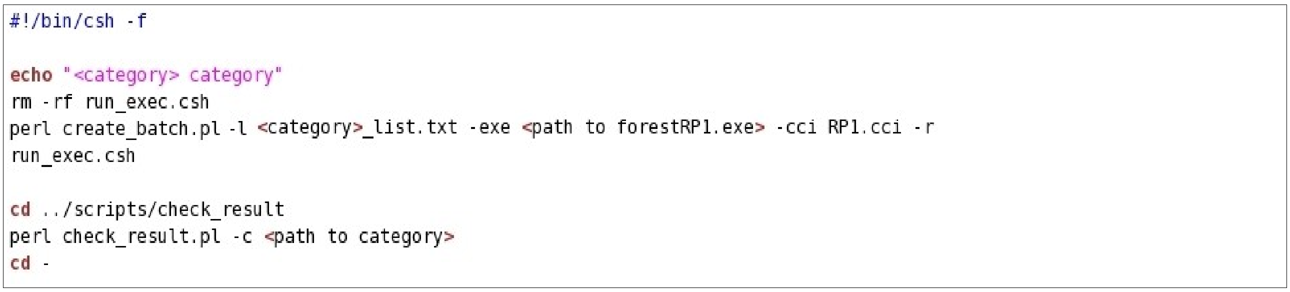
If input paths (<sim_exe_path_fn> and <cci_path_fn>) are provided as relative paths (paths contain “**../**“), they should be the relative paths **from the “sim” directory** (where run_all... batch files are located) point to the execution file and cci file, not from the current directory where “gen_sim.pl” script is located.

If there is no “**../**“ in the paths, input paths are treated as absolute paths to the execution file and cci file.

1.1 Output file description

Sim folder will contain these files :

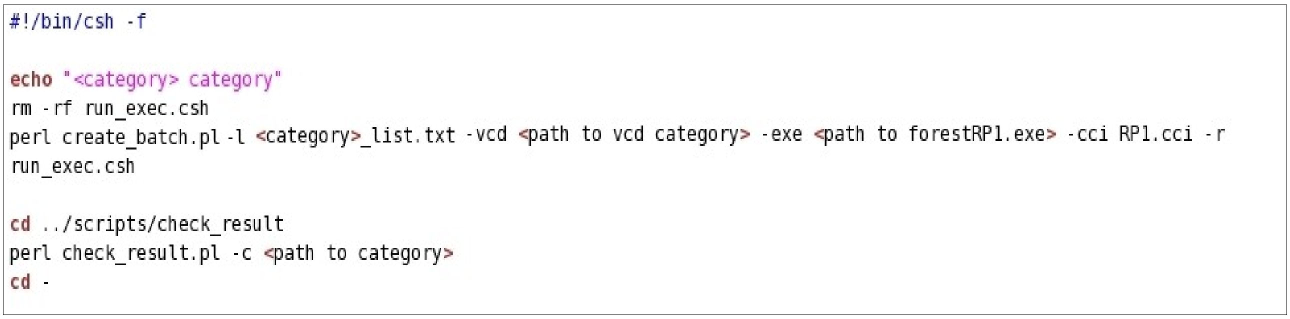
- All necessary files for running a test pattern : test pattern files, simulation files, configuration files...
- *<category>_list.txt* : the list file contains the list of running test patterns (commented TM_name to disable).
- *create_batch.pl* : script to create run simulation batch file *run_<category>_exec.csh*. *run_<category>.csh* (or *run_<category>_vcd.csh*, *run_<category>_vcd_debug.csh*) will run this script to generate batch file based on *<category>_list.txt* then run this file to operate simulation. Finally step is checking result. This script has five options :
  - **-vcd** *<path to vcd category>* : option -vcd and vcd folder of category
  - **-debug** : option -debug
  - **-exe** *<path to forestRP1.exe>* : indicates the path to forest executing file.
  - **-cci** <RP1.cci> : indicates the cpu configuration file.
  - **-r** : indicates that input paths are relative path.
  - **-l** *<<category>_list.txt>* : indicates list of test pattern.
- *run_<category>.csh* : to run simulation for Tms in category and enable in *<category>_list.txt*.



```
#!/bin/csh -f

echo "<category> category"
rm -rf run_exec.csh
perl create_batch.pl -l <category>_list.txt -exe <path to forestRP1.exe> -cci RP1.cci -r
run_exec.csh

cd ../scripts/check_result
perl check_result.pl -c <path to category>
cd -
```

Figure 1.1: Content of run_<category>.csh

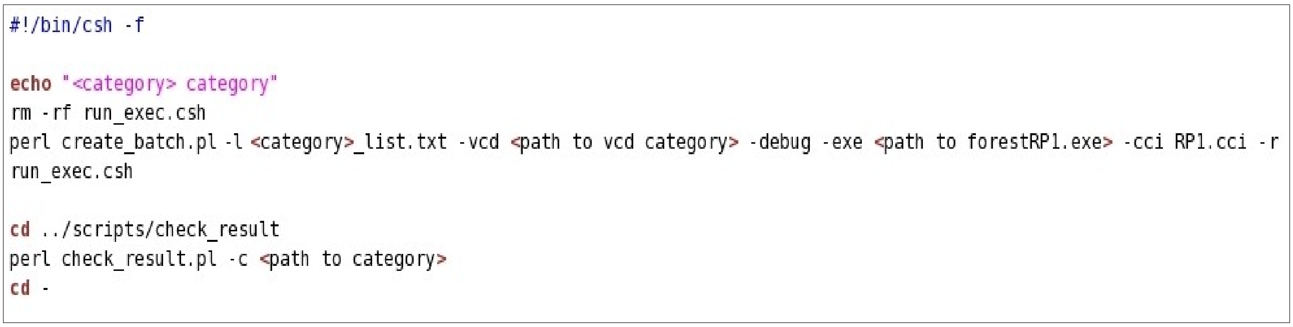
- *run_<category>_vcd.csh* : to run simulation for Tms in category and enable in *<category>_list.txt* with option *-vcd <path to vcd category>*.



```
#!/bin/csh -f

echo "<category> category"
rm -rf run_exec.csh
perl create_batch.pl -l <category>_list.txt -vcd <path to vcd category> -exe <path to forestRP1.exe> -cci RP1.cci -r
run_exec.csh

cd ../scripts/check_result
perl check_result.pl -c <path to category>
cd -
```

Figure 1.2: Content of run_<category>_vcd.csh

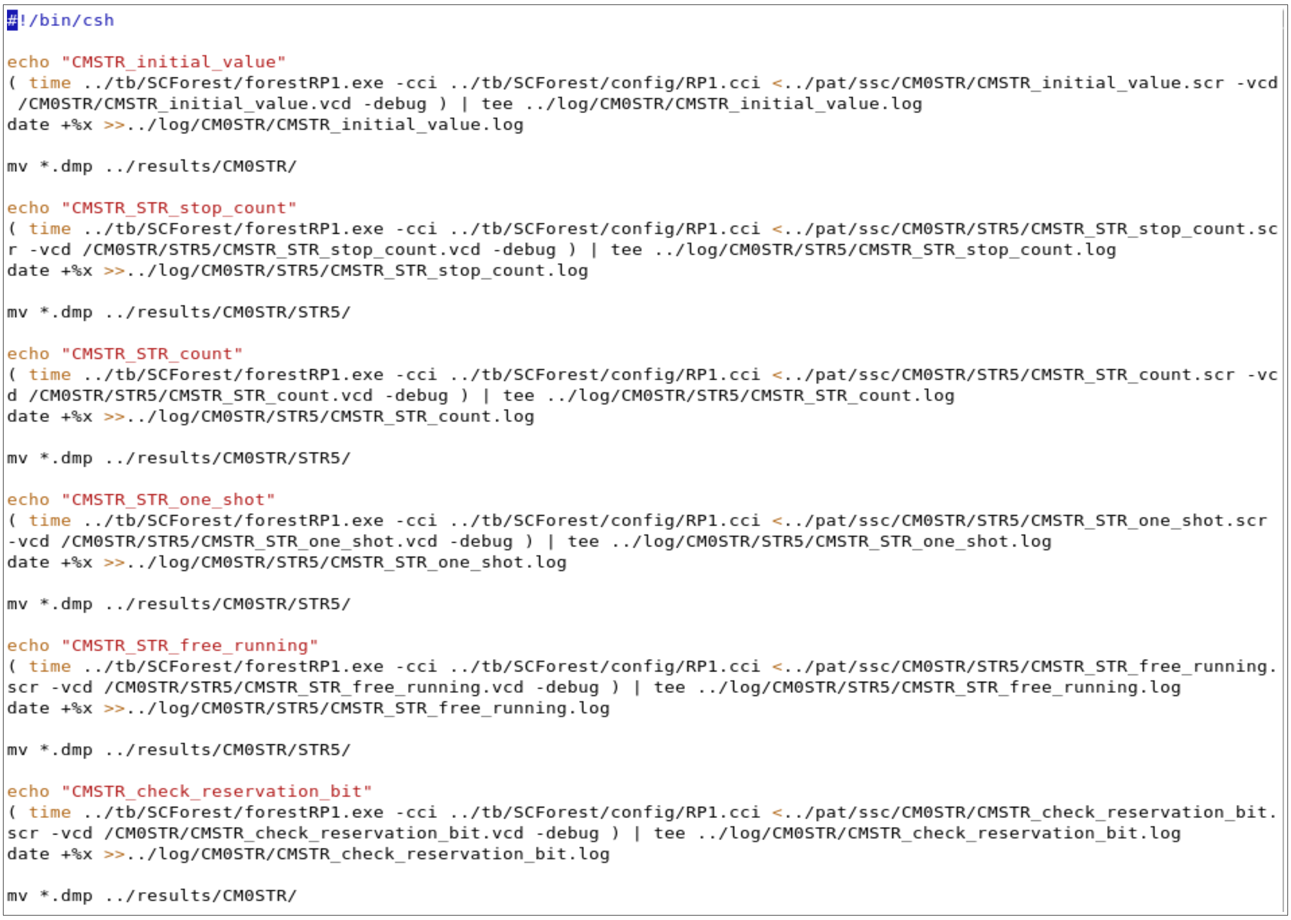
- *run_<category>_vcd_debug.csh* : to run simulation for Tms in category and enable in *<category>_list.txt* with options *-vcd <path to vcd category>* and *-debug*.



```
#!/bin/csh -f

echo "<category> category"
rm -rf run_exec.csh
perl create_batch.pl -l <category>_list.txt -vcd <path to vcd category> -debug -exe <path to forestRP1.exe> -cci RP1.cci -r
run_exec.csh

cd ../scripts/check_result
perl check_result.pl -c <path to category>
cd -
```

Figure 1.3: Content of run_<category>_vcd_debug.csh

- *run_<category>_exec.csh* : This is run simulation batch file is created based on TM list and run automatically. This file contains command lines to run simulation for certain TMs. It calls Forest execution file, provide with cci configuration file, and various kind of options for configure simulation output (log files, waveform, ...).



```
#!/bin/csh

echo "CMSTR_initial_value"
( time ../tb/SCForest/forestRP1.exe -cci ../tb/SCForest/config/RP1.cci <../pat/ssc/CM0STR/CMSTR_initial_value.scr -vcd
 /CM0STR/CMSTR_initial_value.vcd -debug ) | tee ../log/CM0STR/CMSTR_initial_value.log
date +%x >>../log/CM0STR/CMSTR_initial_value.log

mv *.dmp ../results/CM0STR/

echo "CMSTR_STR_stop_count"
( time ../tb/SCForest/forestRP1.exe -cci ../tb/SCForest/config/RP1.cci <../pat/ssc/CM0STR/STR5/CMSTR_STR_stop_count.sc
r -vcd /CM0STR/STR5/CMSTR_STR_stop_count.vcd -debug ) | tee ../log/CM0STR/STR5/CMSTR_STR_stop_count.log
date +%x >>../log/CM0STR/STR5/CMSTR_STR_stop_count.log

mv *.dmp ../results/CM0STR/STR5/

echo "CMSTR_STR_count"
( time ../tb/SCForest/forestRP1.exe -cci ../tb/SCForest/config/RP1.cci <../pat/ssc/CM0STR/STR5/CMSTR_STR_count.scr -vc
d /CM0STR/STR5/CMSTR_STR_count.vcd -debug ) | tee ../log/CM0STR/STR5/CMSTR_STR_count.log
date +%x >>../log/CM0STR/STR5/CMSTR_STR_count.log

mv *.dmp ../results/CM0STR/STR5/

echo "CMSTR_STR_one_shot"
( time ../tb/SCForest/forestRP1.exe -cci ../tb/SCForest/config/RP1.cci <../pat/ssc/CM0STR/STR5/CMSTR_STR_one_shot.scr
-vcd /CM0STR/STR5/CMSTR_STR_one_shot.vcd -debug ) | tee ../log/CM0STR/STR5/CMSTR_STR_one_shot.log
date +%x >>../log/CM0STR/STR5/CMSTR_STR_one_shot.log

mv *.dmp ../results/CM0STR/STR5/

echo "CMSTR_STR_free_running"
( time ../tb/SCForest/forestRP1.exe -cci ../tb/SCForest/config/RP1.cci <../pat/ssc/CM0STR/STR5/CMSTR_STR_free_running.
scr -vcd /CM0STR/STR5/CMSTR_STR_free_running.vcd -debug ) | tee ../log/CM0STR/STR5/CMSTR_STR_free_running.log
date +%x >>../log/CM0STR/STR5/CMSTR_STR_free_running.log

mv *.dmp ../results/CM0STR/STR5/

echo "CMSTR_check_reservation_bit"
( time ../tb/SCForest/forestRP1.exe -cci ../tb/SCForest/config/RP1.cci <../pat/ssc/CM0STR/CMSTR_check_reservation_bit.
scr -vcd /CM0STR/CMSTR_check_reservation_bit.vcd -debug ) | tee ../log/CM0STR/CMSTR_check_reservation_bit.log
date +%x >>../log/CM0STR/CMSTR_check_reservation_bit.log

mv *.dmp ../results/CM0STR/
```

Figure 1.4: Example of content of run_<category>_exec.csh.

In this environment, TMs are subdivided in different categories with different levels (one category may contains sub-categories). For the convenience of the user, for each TM category, a set of batch files will be generated by “gen_sim.pl” script to run simulation for all TMs in the category and also all TMs in all sub-categories of that category. This script also generates batch files to run all TMs. Batch files are C-shell scripts.

Once batch files are generated, user can run simulation easily by running these batch files. Each set of batch files are composed of 3 batch files. With each kind of batch file, the output files of the simulation process are different:

- Standard batch files (file name is "**run_<category>.csh**"): run this kind of batch file to have log file of the simulation process and check result .
- Waveform batch files ("**run_<category>_vcd.csh**"): run this kind of batch file to have log file and simulation waveform and check result .
- Waveform+debug mode batch file ("**run_<category>_vcd_debug.csh**"): run this kind of batch file to simulate in debug mode, get the log file and simulation waveform and check result.

All batch files are put in [sim] directory. Batch files to run all TMs (run_all.csh, run_all_vcd.csh, run_all_vcd_debug.csh) are put right under [sim] directory. Based on the category subdivision, batch files of a category are also put in the corresponding subdirectory of [sim] directory.

Additional information:

- Subdirectories in [sim], [sim/vcd], [log] and [results] directories are also created by this script. If any directory among [sim], [log] and [results] already contains files or subdirectories, the script will prompt for user's decision to remove the old content of that directory.
- This script will also make link to the "Forest_SCI.txt" in the [sim/category] directory, and also make links to all files in the corresponding [pat/category] directory (.cpp, .txt, .ssc, .mot, ...) in the [sim/category] directory for the convenience of running simulation.

1 How to check simulation result

"check_result.pl" script supports user to judge the result of the corresponding TMs is pass or fail. After running simulation, simulation output results (log files, dump data, waveform) of each TMs are generated. These output are checked to judge the result of the TM is pass or fail. This checking process is based on 2 operations: check log file and check dump data.

- check log file: log files are generated in [log] directory. The PC break-point (where simulation is terminated) in the log file is used to judge that TM is pass or fail. The break-points which tell the result is pass or fail are loaded in Address_map_info.txt. check_result.pl reads this input file, and find for these break-point in log files to decide the pass/fail result. Besides, in case that user wants to check the log file with a script defined separately, "check_result.pl" script also supports user to call his script to check the log file to judge the result, not using default script.
- check dump data: dump files (if any) are generated and copied to [results] directory. By default, for each TM, "check_result.pl" script finds all the dump files generated when running a TM, then compare those files (called detected data) with corresponding

expected data to judge that TM is pass or fail. Besides, in case that user wants to check the dump data with a script defined separately, “check_result.pl” script also supports user to call his script to judge the result, not using default script.

When run with a category, “check_result.pl” script will check the result for all TMs in that category and also all TMs in all sub-categories of that category, then output result to “result.txt” files. Other simulation information (CPU time) is also included in “result.txt”. This information is get from log files.

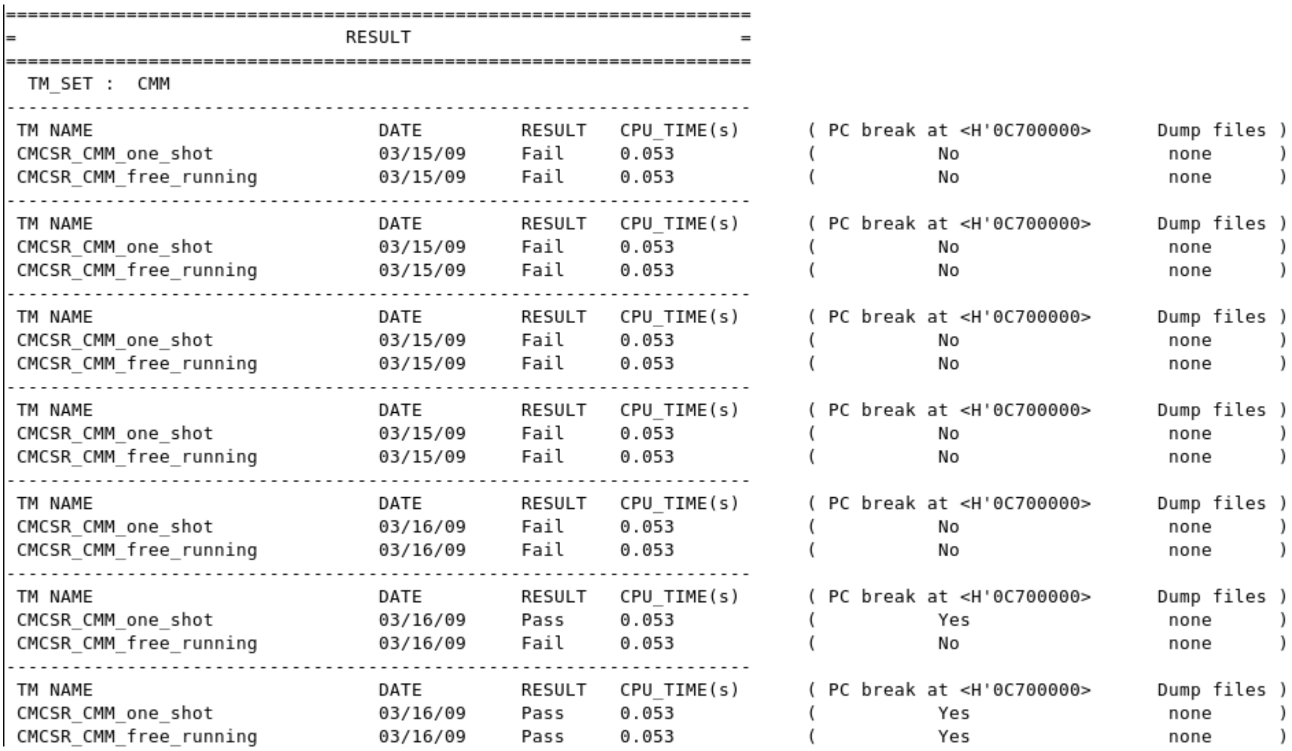
1.1 check_result.pl description

Type	**Content**
Script name	check_result.pl
Language	Perl version 5.8.5 or later
Purpose	This script is used to check the simulation result. check_result.pl based on the PC break point in log file and the address of P_PASS, P_FAIL address in Address_map_info.txt file and also compare the expected data and detected data (if any) to judge the result pass or fail.
Input file	Address_map_info.txt (same with gen_test part), test pattern log file and dump result data (if any) in [.../log/category] and [.../results/category] folders, these file are automatically read by this script.
Output file	In each [.../results/category], result.txt files will be created that stores the result of this “category”.
Usage	perl check_result.pl **-[option]**
Argument	None
Option	**“-c” <input>:** category checking, using this option <input> is category name that is needed to check e.g “-c Initialize” is for checking all test pattern in Initialize set or “-c Operation/count_check” is for checking all test pattern in count_check category of Operation parent category. **Note:** input category must be the last sub category in structure hierarchy. When option is omitted, all test pattern (categories) will be checked. **“-help”** : option for help

Table 1.1: check_result.pl description.

1.1 result.txt description

“result.txt” files contains pass/fail judgment, CPU time and some judgment information for running simulation TMs. With the judgment information, user may know why the TM is passed or failed, e.g the PC break-point is correct or not, the dump files are match or not.



```
=================================================================
=                                RESULT                          =
=================================================================
  TM_SET :  CMM
-----------------------------------------------------------------
 TM NAME                          DATE        RESULT   CPU_TIME(s)      ( PC break at <H'0C700000>      Dump files )
 CMCSR_CMM_one_shot               03/15/09    Fail     0.053            (           No                   none      )
 CMCSR_CMM_free_running           03/15/09    Fail     0.053            (           No                   none      )
-----------------------------------------------------------------
 TM NAME                          DATE        RESULT   CPU_TIME(s)      ( PC break at <H'0C700000>      Dump files )
 CMCSR_CMM_one_shot               03/15/09    Fail     0.053            (           No                   none      )
 CMCSR_CMM_free_running           03/15/09    Fail     0.053            (           No                   none      )
-----------------------------------------------------------------
 TM NAME                          DATE        RESULT   CPU_TIME(s)      ( PC break at <H'0C700000>      Dump files )
 CMCSR_CMM_one_shot               03/15/09    Fail     0.053            (           No                   none      )
 CMCSR_CMM_free_running           03/15/09    Fail     0.053            (           No                   none      )
-----------------------------------------------------------------
 TM NAME                          DATE        RESULT   CPU_TIME(s)      ( PC break at <H'0C700000>      Dump files )
 CMCSR_CMM_one_shot               03/15/09    Fail     0.053            (           No                   none      )
 CMCSR_CMM_free_running           03/15/09    Fail     0.053            (           No                   none      )
-----------------------------------------------------------------
 TM NAME                          DATE        RESULT   CPU_TIME(s)      ( PC break at <H'0C700000>      Dump files )
 CMCSR_CMM_one_shot               03/16/09    Fail     0.053            (           No                   none      )
 CMCSR_CMM_free_running           03/16/09    Fail     0.053            (           No                   none      )
-----------------------------------------------------------------
 TM NAME                          DATE        RESULT   CPU_TIME(s)      ( PC break at <H'0C700000>      Dump files )
 CMCSR_CMM_one_shot               03/16/09    Pass     0.053            (           Yes                  none      )
 CMCSR_CMM_free_running           03/16/09    Fail     0.053            (           No                   none      )
-----------------------------------------------------------------
 TM NAME                          DATE        RESULT   CPU_TIME(s)      ( PC break at <H'0C700000>      Dump files )
 CMCSR_CMM_one_shot               03/16/09    Pass     0.053            (           Yes                  none      )
 CMCSR_CMM_free_running           03/16/09    Pass     0.053            (           Yes                  none      )
```

Figure 1.1: "result.txt" content.

Based on category subdivision, "result.txt" files are put in the corresponding subdirectory of [results] directory. Unlike run-simulation batch files, where batch files in a category will run all TMs in that category and also all TMs in all sub-categories, with "check_result.pl", the "result.txt" file in a category only contains simulation result of TMs in that category, not of any TMs in any sub-categories of that category. The result of TMs in sub-categories will be contained in corresponding "result.txt" file in each sub-category. That means the result of one TM is put in only one "result.txt" file.

When this script run, if a "result.txt" file has not been existed, it will be created. If it has already existed, its content will be updated by added newest simulation result.

1.1 Rule for checking log and dump files

For dump files checking, some rules should be raised for the easy operation of the script. *For each TM*, check_result.pl script will search in the corresponding [pat/category] directory. If the user's script (follow the filename rule described below) is not found, log files, dumped files (also follow the filename rule described below) will be compared by default subroutine of check_result.pl.

In case of default checking: the file name of dump files of a TM must follows the below rule for easily searched by the script.

- Detected dump file's filename: <TM's name>_**Det**[number].dmp
- Expected dump file's filename: <TM's name>_**Exp**[number].dmp

number is used in case of multiple dump files for one TM. The value of *number* is 1, 2, 3, ...

Example: *TM_1_1_1_Det.dmp, TM_1_1_1_Exp.dmp;*

TM_1_2_Det1.dmp, TM_1_2_Det2.dmp, TM_1_2_Exp1.dmp, TM_1_2_Exp2.dmp.

This rule must be considered when developing models. If the dump files are not intended to be checked by default script of “check_result.pl” but by a user's script, this rule can be neglected.

When checking log file, check_result.pl will base on Section PASS/FAIL address in Address_map_info.txt to check pass/fail of each TM.

In case of checking by user-defined script: the filename and output of this script must be fixed as the following rule for easily called by “check_result.pl”:

- Filename: **TM_name_check_dump.pl** (used for checking dump file)

  TM_name_check_log.pl (used for checking log file)

  Location: the corresponding [pat/category] directory.

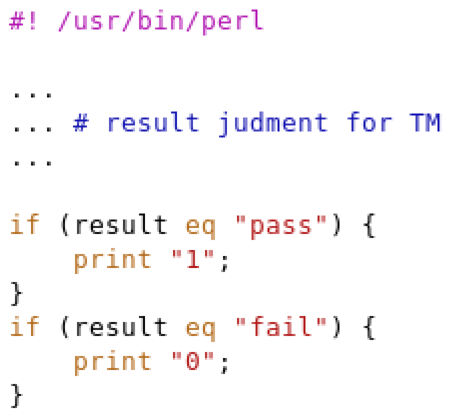
  Example: *CMCSR_CMM_free_running_check_dump.pl* (stored in pat/CM0CSR/CMM/)

  CMSTR_check_reservation_bit_check_dump.pl (stored in pat/CM0STR)

  CMCSR_CMM_free_running_check_log.pl (stored in pat/CM0CSR/CMM/)

  CMSTR_check_reservation_bit_check_log.pl (stored in pat/CM0STR)
- Output: pass/fail result of specified TM, represented by a number 0/1 (**0 for fail and 1 for pass**). These values must be printed out directly to standard output (see the below example).

  Example:



```
#! /usr/bin/perl

...
... # result judment for TM
...

if (result eq "pass") {
    print "1";
}
if (result eq "fail") {
    print "0";
}
```

1 How to generate reports

“gen_report.pl” supports user to make many kinds of report on simulation result. Reading reports is the easiest way for the user to check the simulation result of TMs instead of accessing "result.txt" files.

1.1 gen_report.pl script description

Type	**Content**
Script name	gen_report.pl
Language	Perl version 5.8.5 or later
Purpose	This script is used to generate report based on simulation result.
Input file	results file in [results/category] directory, When user choose to report follow category the result.txt file will be automatically read in user specified category. When user choose to report all category, this script will search each category in results directory and automatically read result.txt file in each category that it found.
Output file	Detailed report, summary report, daily report from the simulation results. Some information for the report is test pattern name,test pattern set, number of pass/fail, average CPU running time and total memory usage for running simulation. All info getting for report is from result.txt file.
Usage	perl gen_report.pl **-[option]**
Argument	None
Option	**"-c" <category>:** category report, using this option script will generate detailed and summary, daily report for a specified category into text file. e.g: "-c operation/convert" for generating report for "convert" category in parent "operation" category. **Note:** input category must be the last subcategory in structure hierarchy. **"-l" <checklist>:** create reports into checklist, using this option script will add Summary_Report sheet and Detailed_report sheet in input checklist and put summary report and detailed report content into these sheet. e.g "-l ../abc/VRF-SLD-08009.ods" When user does not declare "-c" option, this script will generate reports for all running test pattern (category) into text file. **"-help"** : option for help

Table 1.1: gen_report.pl description

1.2 Reports description

When gen_report.pl script reports all project, two reports are created : detail report (*detail_report.txt*), summary report (*summary_report.txt*).

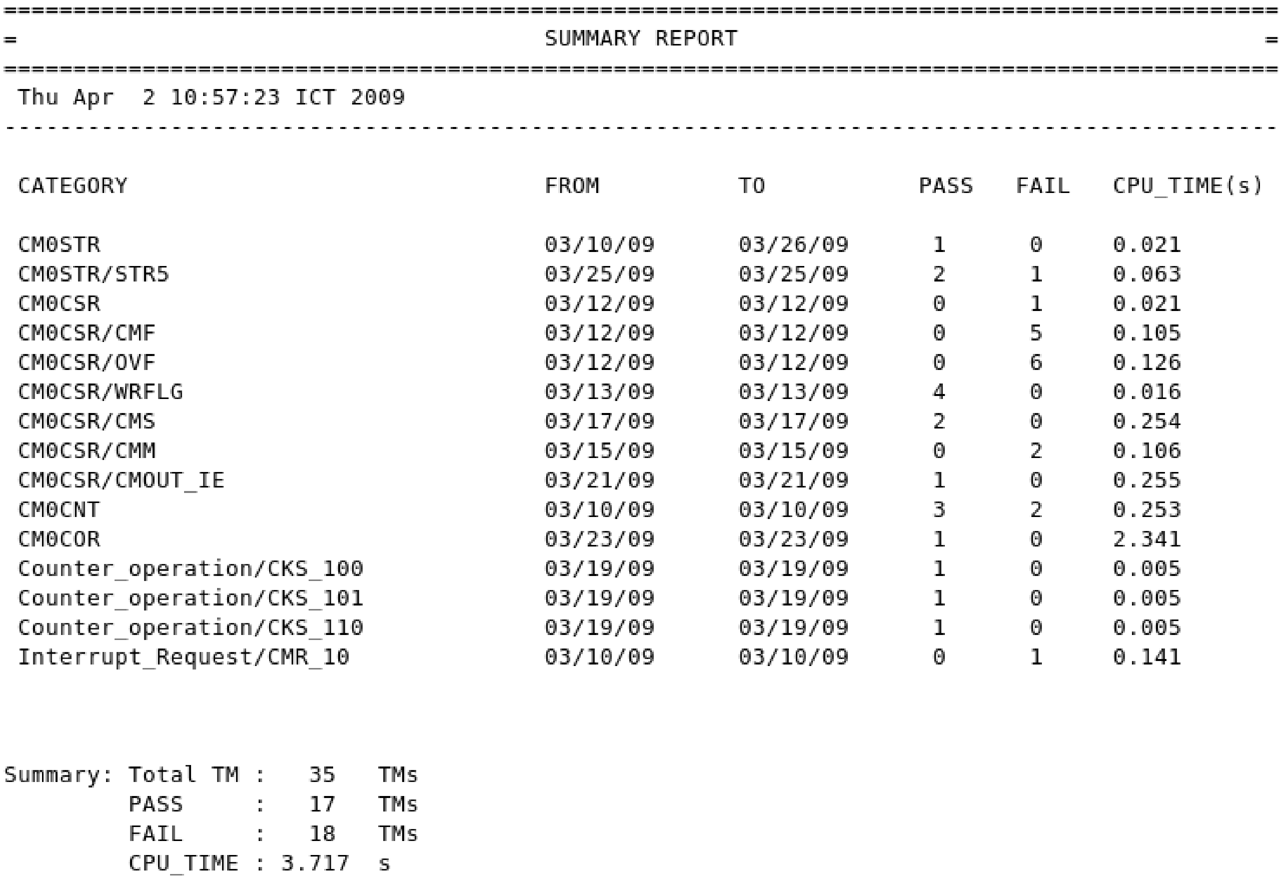
When gen_report.pl script reports one category, two reports are created : category detail report (*<category>_detail_report.txt*), category summary report (*<category>_summary_report.txt*).

If users run script to report many categories, each category will be reported by two reports.

If users report again one category,the reports of this category will be overwritten.

Reports generated by “gen_report.pl” are:

- Summary report (**summary_report.txt**): this report summarizes simulation result of TMs in the category. Information in this report include: number of pass TMs / fail TMs, CPU time, the earliest day and the latest day when TMs in that category run.

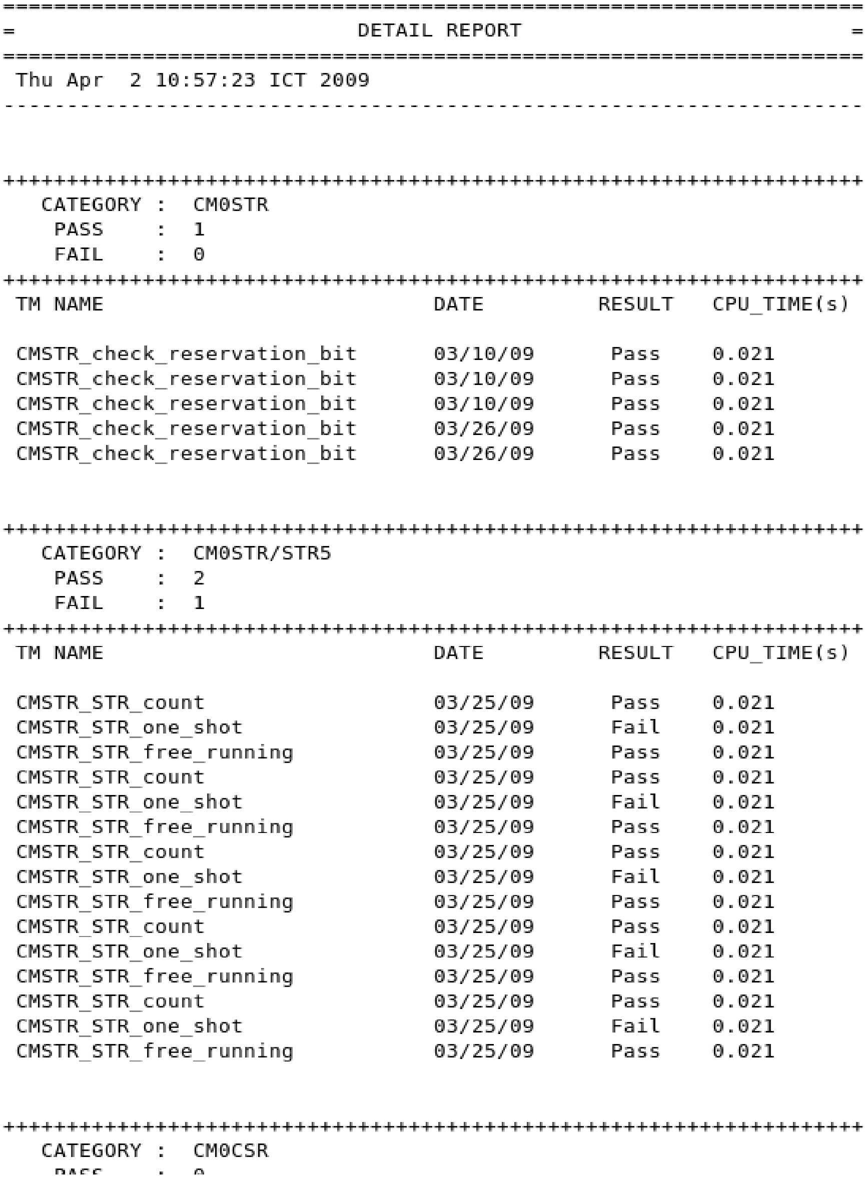


```
==========================================================================================
=                                      SUMMARY REPORT                                    =
==========================================================================================
 Thu Apr  2 10:57:23 ICT 2009
------------------------------------------------------------------------------------------

 CATEGORY                              FROM          TO            PASS    FAIL    CPU_TIME(s)

 CM0STR                                03/10/09      03/26/09       1       0      0.021
 CM0STR/STR5                           03/25/09      03/25/09       2       1      0.063
 CM0CSR                                03/12/09      03/12/09       0       1      0.021
 CM0CSR/CMF                            03/12/09      03/12/09       0       5      0.105
 CM0CSR/OVF                            03/12/09      03/12/09       0       6      0.126
 CM0CSR/WRFLG                          03/13/09      03/13/09       4       0      0.016
 CM0CSR/CMS                            03/17/09      03/17/09       2       0      0.254
 CM0CSR/CMM                            03/15/09      03/15/09       0       2      0.106
 CM0CSR/CMOUT_IE                       03/21/09      03/21/09       1       0      0.255
 CM0CNT                                03/10/09      03/10/09       3       2      0.253
 CM0COR                                03/23/09      03/23/09       1       0      2.341
 Counter_operation/CKS_100             03/19/09      03/19/09       1       0      0.005
 Counter_operation/CKS_101             03/19/09      03/19/09       1       0      0.005
 Counter_operation/CKS_110             03/19/09      03/19/09       1       0      0.005
 Interrupt_Request/CMR_10              03/10/09      03/10/09       0       1      0.141


Summary: Total TM :   35   TMs
         PASS     :   17   TMs
         FAIL     :   18   TMs
         CPU_TIME : 3.717  s
```

Figure 1.1: Example of summary_report.txt

- Detail report (**detail_report.txt**): this report contains the result of every TMs in the category, for all times run-simulation. Information in this report include: the day when the

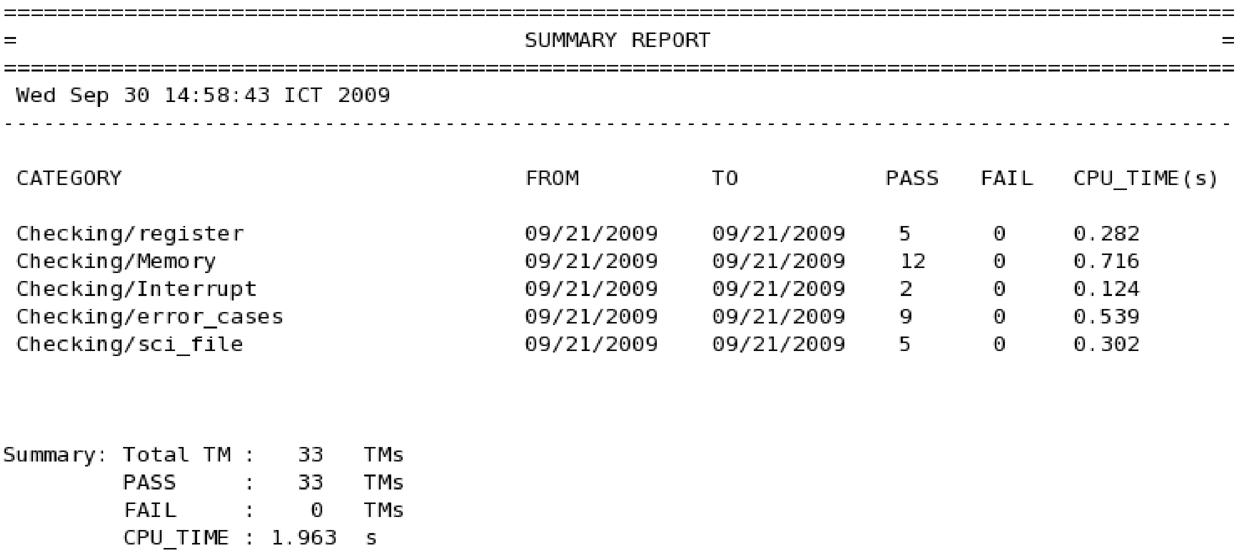
TM run, pass/ fail result, and CPU time for that TM.



```
====================================================================
=                            DETAIL REPORT                         =
====================================================================
 Thu Apr  2 10:57:23 ICT 2009
--------------------------------------------------------------------


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   CATEGORY :  CM0STR
    PASS    :  1
    FAIL    :  0
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 TM NAME                          DATE         RESULT   CPU_TIME(s)

 CMSTR_check_reservation_bit      03/10/09      Pass    0.021
 CMSTR_check_reservation_bit      03/10/09      Pass    0.021
 CMSTR_check_reservation_bit      03/10/09      Pass    0.021
 CMSTR_check_reservation_bit      03/26/09      Pass    0.021
 CMSTR_check_reservation_bit      03/26/09      Pass    0.021


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   CATEGORY :  CM0STR/STR5
    PASS    :  2
    FAIL    :  1
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 TM NAME                          DATE         RESULT   CPU_TIME(s)

 CMSTR_STR_count                  03/25/09      Pass    0.021
 CMSTR_STR_one_shot               03/25/09      Fail    0.021
 CMSTR_STR_free_running           03/25/09      Pass    0.021
 CMSTR_STR_count                  03/25/09      Pass    0.021
 CMSTR_STR_one_shot               03/25/09      Fail    0.021
 CMSTR_STR_free_running           03/25/09      Pass    0.021
 CMSTR_STR_count                  03/25/09      Pass    0.021
 CMSTR_STR_one_shot               03/25/09      Fail    0.021
 CMSTR_STR_free_running           03/25/09      Pass    0.021
 CMSTR_STR_count                  03/25/09      Pass    0.021
 CMSTR_STR_one_shot               03/25/09      Fail    0.021
 CMSTR_STR_free_running           03/25/09      Pass    0.021
 CMSTR_STR_count                  03/25/09      Pass    0.021
 CMSTR_STR_one_shot               03/25/09      Fail    0.021
 CMSTR_STR_free_running           03/25/09      Pass    0.021


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   CATEGORY :  CM0CSR
    PASS    :  0
```

Figure 1.2: Example of detail_report.txt

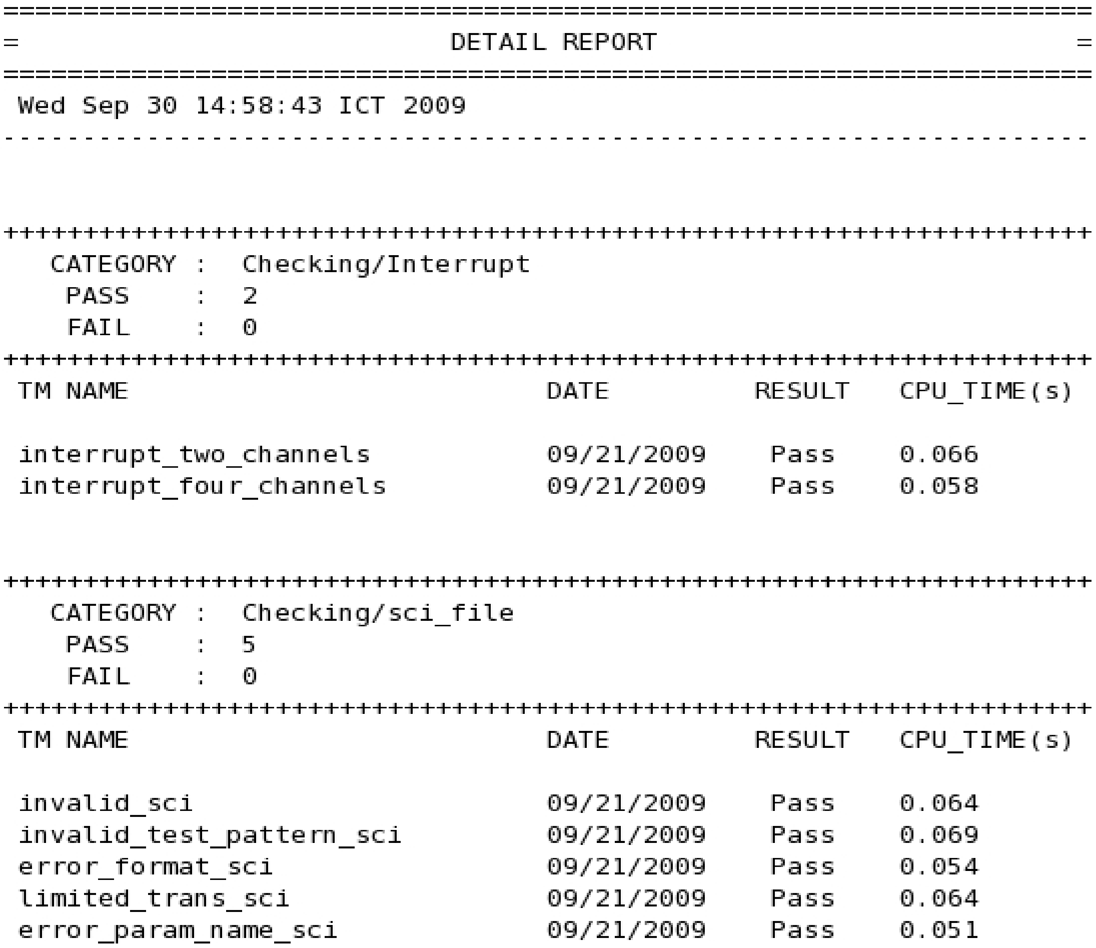
- Category summary report (**<category>_summary_report.txt**): this report summarizes simulation result of TMs in **specify** category. Information in this report include: number of pass TMs / fail TMs, CPU time, the earliest day and the latest day when TMs in that category run.



```
=============================================================================================
=                                     SUMMARY REPORT                                        =
=============================================================================================
 Wed Sep 30 14:58:43 ICT 2009
---------------------------------------------------------------------------------------------

 CATEGORY                             FROM          TO            PASS    FAIL    CPU_TIME(s)

 Checking/register                    09/21/2009    09/21/2009     5       0      0.282
 Checking/Memory                      09/21/2009    09/21/2009     12      0      0.716
 Checking/Interrupt                   09/21/2009    09/21/2009     2       0      0.124
 Checking/error_cases                 09/21/2009    09/21/2009     9       0      0.539
 Checking/sci_file                    09/21/2009    09/21/2009     5       0      0.302



Summary: Total TM :    33   TMs
         PASS     :    33   TMs
         FAIL     :     0   TMs
         CPU_TIME : 1.963   s
```

Figure 1.3: Example of <category>_summary_report.txt

- Category detail report (**<category>detail_report.txt**): this report contains the result of every TMs in the **specify** category, for all times run-simulation. Information in this report include: the day when the TM run, pass/ fail result, and CPU time for that TM.



```
======================================================================
=                           DETAIL REPORT                            =
======================================================================
 Wed Sep 30 14:58:43 ICT 2009
----------------------------------------------------------------------



++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   CATEGORY :  Checking/Interrupt
    PASS    :  2
    FAIL    :  0
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 TM NAME                          DATE         RESULT    CPU_TIME(s)

 interrupt_two_channels           09/21/2009    Pass     0.066
 interrupt_four_channels          09/21/2009    Pass     0.058



++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   CATEGORY :  Checking/sci_file
    PASS    :  5
    FAIL    :  0
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 TM NAME                          DATE         RESULT    CPU_TIME(s)

 invalid_sci                      09/21/2009    Pass     0.064
 invalid_test_pattern_sci         09/21/2009    Pass     0.069
 error_format_sci                 09/21/2009    Pass     0.054
 limited_trans_sci                09/21/2009    Pass     0.064
 error_param_name_sci             09/21/2009    Pass     0.051
```

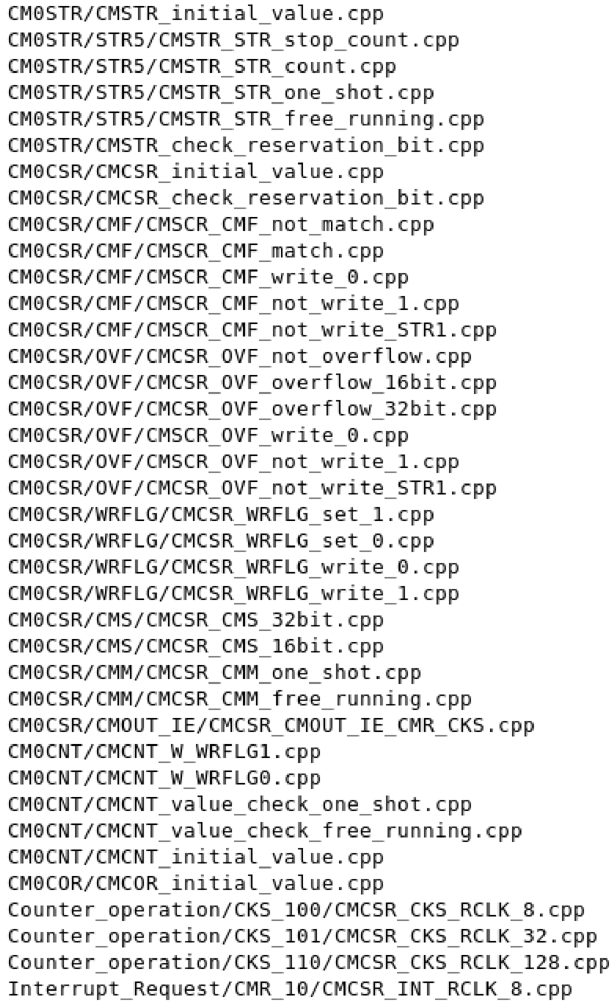
Figure 1.4: Example of <category>_detail_report.txt

1 Scripts running guide

1.1 “gen_sim.pl” running guide

This part guides user to run this script to generate batch files with an example.

Input: The content of “TM_path.txt” file:



```
CM0STR/CMSTR_initial_value.cpp
CM0STR/STR5/CMSTR_STR_stop_count.cpp
CM0STR/STR5/CMSTR_STR_count.cpp
CM0STR/STR5/CMSTR_STR_one_shot.cpp
CM0STR/STR5/CMSTR_STR_free_running.cpp
CM0STR/CMSTR_check_reservation_bit.cpp
CM0CSR/CMCSR_initial_value.cpp
CM0CSR/CMCSR_check_reservation_bit.cpp
CM0CSR/CMF/CMSCR_CMF_not_match.cpp
CM0CSR/CMF/CMSCR_CMF_match.cpp
CM0CSR/CMF/CMSCR_CMF_write_0.cpp
CM0CSR/CMF/CMSCR_CMF_not_write_1.cpp
CM0CSR/CMF/CMSCR_CMF_not_write_STR1.cpp
CM0CSR/OVF/CMCSR_OVF_not_overflow.cpp
CM0CSR/OVF/CMCSR_OVF_overflow_16bit.cpp
CM0CSR/OVF/CMCSR_OVF_overflow_32bit.cpp
CM0CSR/OVF/CMSCR_OVF_write_0.cpp
CM0CSR/OVF/CMSCR_OVF_not_write_1.cpp
CM0CSR/OVF/CMCSR_OVF_not_write_STR1.cpp
CM0CSR/WRFLG/CMCSR_WRFLG_set_1.cpp
CM0CSR/WRFLG/CMCSR_WRFLG_set_0.cpp
CM0CSR/WRFLG/CMCSR_WRFLG_write_0.cpp
CM0CSR/WRFLG/CMCSR_WRFLG_write_1.cpp
CM0CSR/CMS/CMCSR_CMS_32bit.cpp
CM0CSR/CMS/CMCSR_CMS_16bit.cpp
CM0CSR/CMM/CMCSR_CMM_one_shot.cpp
CM0CSR/CMM/CMCSR_CMM_free_running.cpp
CM0CSR/CMOUT_IE/CMCSR_CMOUT_IE_CMR_CKS.cpp
CM0CNT/CMCNT_W_WRFLG1.cpp
CM0CNT/CMCNT_W_WRFLG0.cpp
CM0CNT/CMCNT_value_check_one_shot.cpp
CM0CNT/CMCNT_value_check_free_running.cpp
CM0CNT/CMCNT_initial_value.cpp
CM0COR/CMCOR_initial_value.cpp
Counter_operation/CKS_100/CMCSR_CKS_RCLK_8.cpp
Counter_operation/CKS_101/CMCSR_CKS_RCLK_32.cpp
Counter_operation/CKS_110/CMCSR_CKS_RCLK_128.cpp
Interrupt_Request/CMR_10/CMCSR_INT_RCLK_8.cpp
```

Figure 1.1: Content of TM_path.txt

This file has been generated by “gen_TM.pl” script (refer to “Test-pattern Guide” for more information).

Command:

For CPU Environment

perl gen_sim.pl -exe ../tb/SCForest/forestRP1.exe -cci ../tb/SCForest/config/RP1.cci

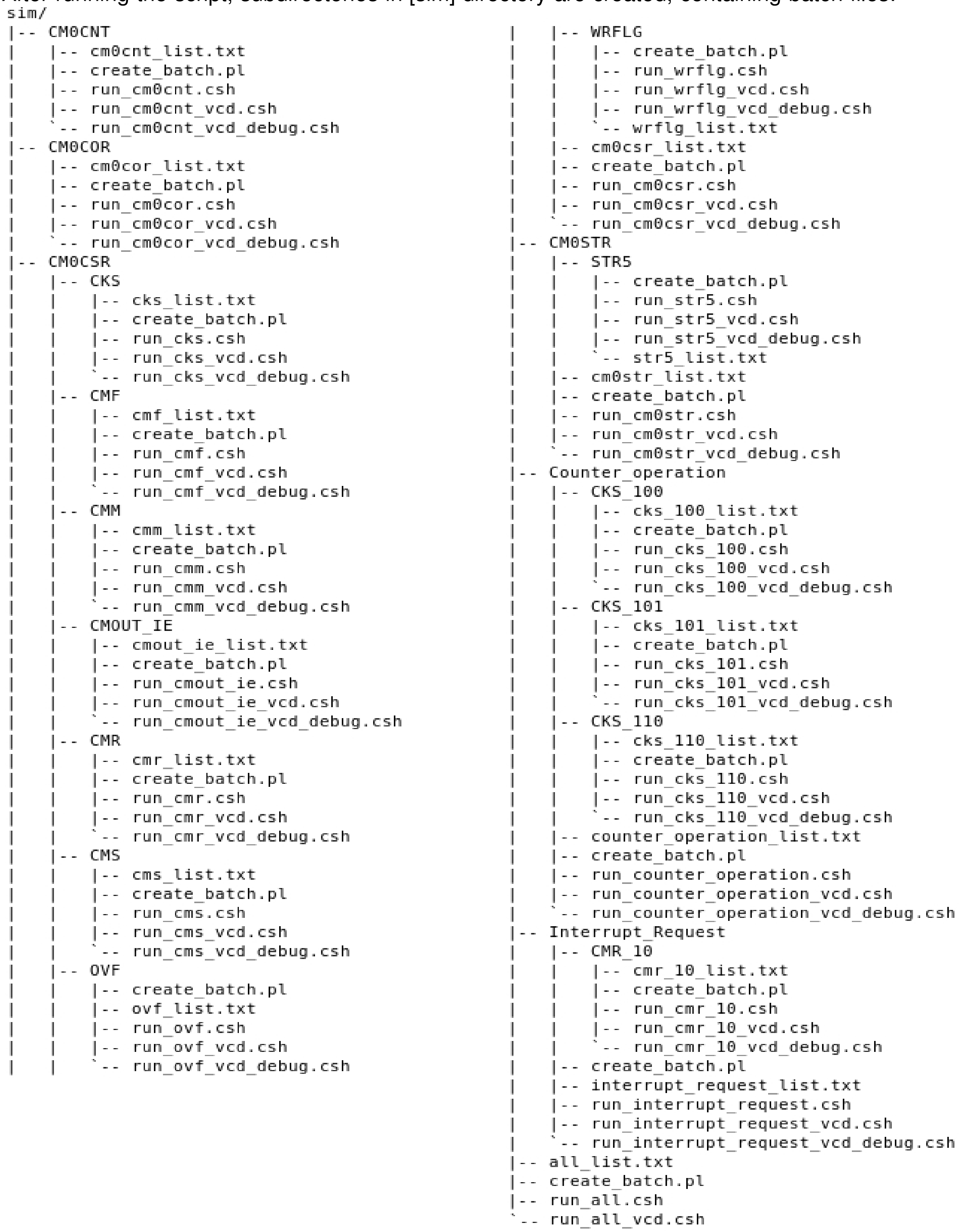
For Stimulus Environment

perl gen_sim.pl -exe ../tb/tlmforest_sm/tlmforest.exe -stim

../tb/SCForest/forestRP1.exe:	path from sim folder to simulation execution file.
../tb/SCForest/config/RP1.cci:	path from sim folder to cci file.

Result:

After running the script, subdirectories in [sim] directory are created, containing batch files:



```
sim/
|-- CM0CNT
|   |-- cm0cnt_list.txt
|   |-- create_batch.pl
|   |-- run_cm0cnt.csh
|   |-- run_cm0cnt_vcd.csh
|   `-- run_cm0cnt_vcd_debug.csh
|-- CM0COR
|   |-- cm0cor_list.txt
|   |-- create_batch.pl
|   |-- run_cm0cor.csh
|   |-- run_cm0cor_vcd.csh
|   `-- run_cm0cor_vcd_debug.csh
|-- CM0CSR
|   |-- CKS
|   |   |-- cks_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cks.csh
|   |   |-- run_cks_vcd.csh
|   |   `-- run_cks_vcd_debug.csh
|   |-- CMF
|   |   |-- cmf_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cmf.csh
|   |   |-- run_cmf_vcd.csh
|   |   `-- run_cmf_vcd_debug.csh
|   |-- CMM
|   |   |-- cmm_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cmm.csh
|   |   |-- run_cmm_vcd.csh
|   |   `-- run_cmm_vcd_debug.csh
|   |-- CMOUT_IE
|   |   |-- cmout_ie_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cmout_ie.csh
|   |   |-- run_cmout_ie_vcd.csh
|   |   `-- run_cmout_ie_vcd_debug.csh
|   |-- CMR
|   |   |-- cmr_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cmr.csh
|   |   |-- run_cmr_vcd.csh
|   |   `-- run_cmr_vcd_debug.csh
|   |-- CMS
|   |   |-- cms_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cms.csh
|   |   |-- run_cms_vcd.csh
|   |   `-- run_cms_vcd_debug.csh
|   |-- OVF
|   |   |-- create_batch.pl
|   |   |-- ovf_list.txt
|   |   |-- run_ovf.csh
|   |   |-- run_ovf_vcd.csh
|   |   `-- run_ovf_vcd_debug.csh
|   |-- WRFLG
|   |   |-- create_batch.pl
|   |   |-- run_wrflg.csh
|   |   |-- run_wrflg_vcd.csh
|   |   |-- run_wrflg_vcd_debug.csh
|   |   `-- wrflg_list.txt
|   |-- cm0csr_list.txt
|   |-- create_batch.pl
|   |-- run_cm0csr.csh
|   |-- run_cm0csr_vcd.csh
|   `-- run_cm0csr_vcd_debug.csh
|-- CM0STR
|   |-- STR5
|   |   |-- create_batch.pl
|   |   |-- run_str5.csh
|   |   |-- run_str5_vcd.csh
|   |   |-- run_str5_vcd_debug.csh
|   |   `-- str5_list.txt
|   |-- cm0str_list.txt
|   |-- create_batch.pl
|   |-- run_cm0str.csh
|   |-- run_cm0str_vcd.csh
|   `-- run_cm0str_vcd_debug.csh
|-- Counter_operation
|   |-- CKS_100
|   |   |-- cks_100_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cks_100.csh
|   |   |-- run_cks_100_vcd.csh
|   |   `-- run_cks_100_vcd_debug.csh
|   |-- CKS_101
|   |   |-- cks_101_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cks_101.csh
|   |   |-- run_cks_101_vcd.csh
|   |   `-- run_cks_101_vcd_debug.csh
|   |-- CKS_110
|   |   |-- cks_110_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cks_110.csh
|   |   |-- run_cks_110_vcd.csh
|   |   `-- run_cks_110_vcd_debug.csh
|   |-- counter_operation_list.txt
|   |-- create_batch.pl
|   |-- run_counter_operation.csh
|   |-- run_counter_operation_vcd.csh
|   `-- run_counter_operation_vcd_debug.csh
|-- Interrupt_Request
|   |-- CMR_10
|   |   |-- cmr_10_list.txt
|   |   |-- create_batch.pl
|   |   |-- run_cmr_10.csh
|   |   |-- run_cmr_10_vcd.csh
|   |   `-- run_cmr_10_vcd_debug.csh
|   |-- create_batch.pl
|   |-- interrupt_request_list.txt
|   |-- run_interrupt_request.csh
|   |-- run_interrupt_request_vcd.csh
|   `-- run_interrupt_request_vcd_debug.csh
|-- all_list.txt
|-- create_batch.pl
|-- run_all.csh
`-- run_all_vcd.csh
```

Figure 1.2: Example of batch file structure in [sim] folder

The content of batch file is described in previous chapter.

1.1 “check_result.pl” running guide

This part guides user to run this script to check result for one category or all categories.

1.1.1 Running with all categories

Input: After user run a batch file to run simulation for TMs, simulation log files for TMs will be generated and stored in [log] directory following the category subdivision:

```
log
├── CMOCNT
│   ├── CMCNT_W_WRFLG0.log
│   ├── CMCNT_W_WRFLG1.log
│   ├── CMCNT_initial_value.log
│   ├── CMCNT_value_check_free_running.log
│   └── CMCNT_value_check_one_shot.log
├── CMOCOR
│   └── CMCOR_initial_value.log
├── CMOCSR
│   ├── CMCSR_check_reservation_bit.log
│   ├── CMCSR_initial_value.log
│   ├── CMF
│   │   ├── CMSCR_CMF_match.log
│   │   ├── CMSCR_CMF_not_match.log
│   │   ├── CMSCR_CMF_not_write_1.log
│   │   ├── CMSCR_CMF_not_write_STR1.log
│   │   └── CMSCR_CMF_write_0.log
│   ├── CMM
│   │   ├── CMCSR_CMM_free_running.log
│   │   └── CMCSR_CMM_one_shot.log
│   ├── CMOUT_IE
│   │   └── CMCSR_CMOUT_IE_CMR_CKS.log
│   ├── CMS
│   │   ├── CMCSR_CMS_16bit.log
│   │   └── CMCSR_CMS_32bit.log
│   ├── OVF
│   │   ├── CMCSR_OVF_not_overflow.log
│   │   ├── CMCSR_OVF_not_write_STR1.log
│   │   ├── CMCSR_OVF_overflow_16bit.log
│   │   ├── CMCSR_OVF_overflow_32bit.log
│   │   ├── CMSCR_OVF_not_write_1.log
│   │   └── CMSCR_OVF_write_0.log
│   └── WRFLG
│       ├── CMCSR_WRFLG_set_0.log
│       ├── CMCSR_WRFLG_set_1.log
│       ├── CMCSR_WRFLG_write_0.log
│       └── CMCSR_WRFLG_write_1.log
├── CMOSTR
│   ├── CMSTR_check_reservation_bit.log
│   ├── CMSTR_initial_value.log
│   └── STR5
│       ├── CMSTR_STR_count.log
│       ├── CMSTR_STR_free_running.log
│       ├── CMSTR_STR_one_shot.log
│       └── CMSTR_STR_stop_count.log
├── Counter_operation
│   ├── CKS_100
│   │   └── CMCSR_CKS_RCLK_8.log
│   ├── CKS_101
│   │   └── CMCSR_CKS_RCLK_32.log
│   └── CKS_110
│       └── CMCSR_CKS_RCLK_128.log
├── Interrupt_Request
    └── CMR_10
        └── CMCSR_INT_RCLK_8.log
```

Command:

perl check_result.pl

Result:

After running this script, “result.txt” files will be updated, containing results for all categories which contains TMs. These files are put in [results] directory, also follows the category subdivision:

```
results
├── CM0CNT
│   └── result.txt
├── CM0COR
│   └── result.txt
├── CM0CSR
│   ├── CMF
│   │   └── result.txt
│   ├── CMM
│   │   └── result.txt
│   ├── CMOUT_IE
│   │   └── result.txt
│   ├── CMS
│   │   └── result.txt
│   ├── OVF
│   │   └── result.txt
│   ├── WRFLG
│   │   └── result.txt
│   └── result.txt
├── CM0STR
│   ├── STR5
│   │   └── result.txt
│   └── result.txt
├── Counter_operation
│   ├── CKS_100
│   │   └── result.txt
│   ├── CKS_101
│   │   └── result.txt
│   └── CKS_110
│       └── result.txt
└── Interrupt_Request
    └── CMR_10
           └── result.txt
```

The content of “result.txt” file is described in previous chapter.

1.1.2 Running with one category

User can use this script check result for some certain categories.

Command:

perl check_result.pl -c CM0CSR

-c CM0CSR : check result for CM0CSR category.

Result:

This script will check the result for CM0CSR category and all of its subcategories, and update the “result.txt” files of corresponding categories. The below “result.txt” file will be updated:

```
├── CM0CSR
│   ├── CMF
│   │   └── result.txt
│   ├── CMM
│   │   └── result.txt
│   ├── CMOUT_IE
│   │   └── result.txt
│   ├── CMS
│   │   └── result.txt
│   ├── OVF
│   │   └── result.txt
│   ├── WRFLG
│   │   └── result.txt
```

```
|    └── result.txt
```

1.2 “gen_report.pl” running guide

This part guides user to run this script to generate report for one category or all categories.

1.2.1 Running with all categories

Command:

perl gen_report.pl

Result:

This script will generate reports for all categories. The content of reports is described in previous chapter.

1.2.2 Running with one category

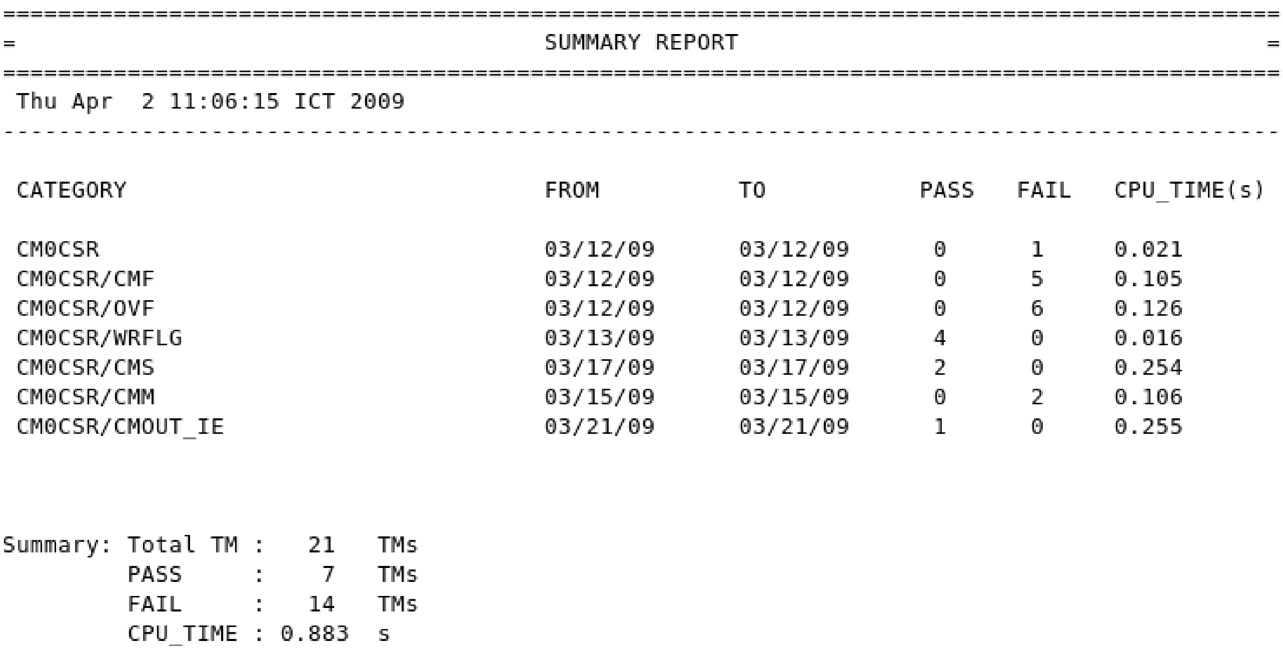
Command:

perl gen_report.pl -c CM0CSR

-c CM0CSR : generate reports for CM0CSR category.

Result:

This script will generate reports for the input category, and also for all of its subcategories.



```
==============================================================================================
=                                      SUMMARY REPORT                                        =
==============================================================================================
 Thu Apr  2 11:06:15 ICT 2009
----------------------------------------------------------------------------------------------

 CATEGORY                              FROM          TO            PASS    FAIL    CPU_TIME(s)

 CM0CSR                                03/12/09      03/12/09       0       1      0.021
 CM0CSR/CMF                            03/12/09      03/12/09       0       5      0.105
 CM0CSR/OVF                            03/12/09      03/12/09       0       6      0.126
 CM0CSR/WRFLG                          03/13/09      03/13/09       4       0      0.016
 CM0CSR/CMS                            03/17/09      03/17/09       2       0      0.254
 CM0CSR/CMM                            03/15/09      03/15/09       0       2      0.106
 CM0CSR/CMOUT_IE                       03/21/09      03/21/09       1       0      0.255



Summary: Total TM :    21   TMs
         PASS     :     7   TMs
         FAIL     :    14   TMs
         CPU_TIME : 0.883   s
```

Figure 1.3: CM0CSR_summary_report.txt for CM0CSR category.

1.2.3 Update reports to checklist

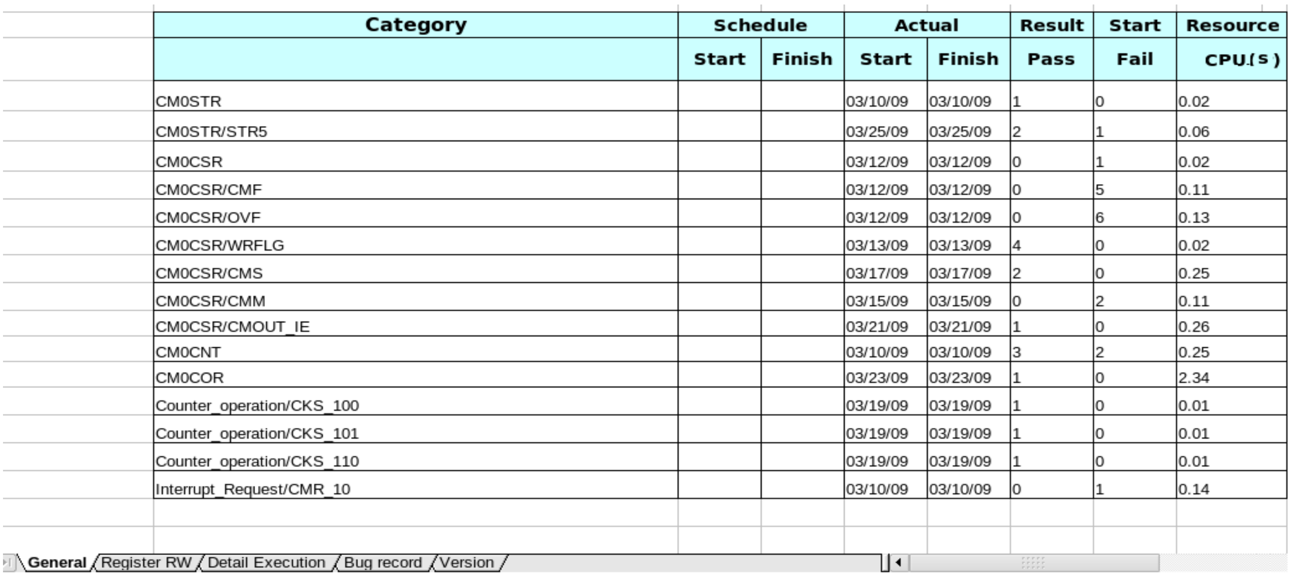
Command:

perl gen_report.pl -I ./VRF-SLD-08008-1.xls

Result:

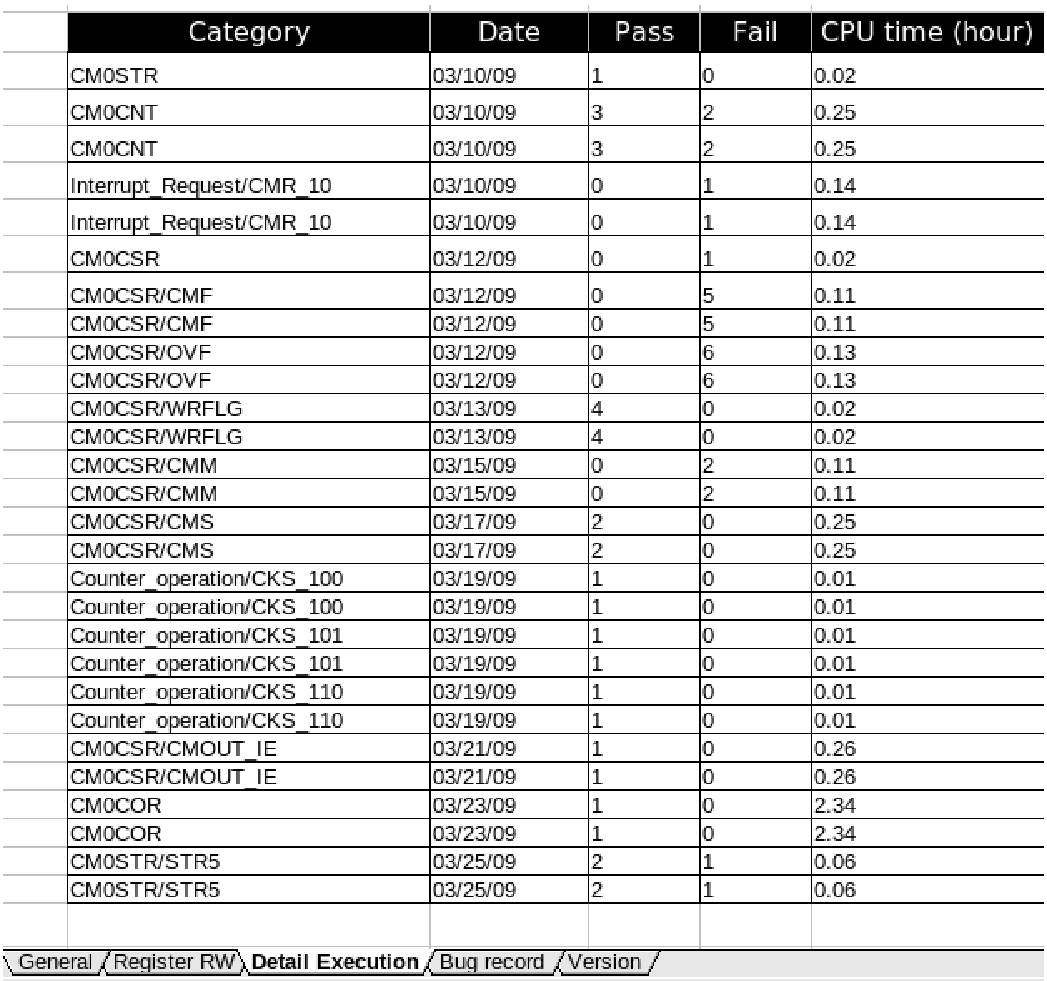
Two report sheets in the checklist will be updated.

- "General" sheet: just like summary report.



Category	Schedule		Actual		Result	Start	Resource
	Start	Finish	Start	Finish	Pass	Fail	CPU.(s)
CM0STR			03/10/09	03/10/09	1	0	0.02
CM0STR/STR5			03/25/09	03/25/09	2	1	0.06
CM0CSR			03/12/09	03/12/09	0	1	0.02
CM0CSR/CMF			03/12/09	03/12/09	0	5	0.11
CM0CSR/OVF			03/12/09	03/12/09	0	6	0.13
CM0CSR/WRFLG			03/13/09	03/13/09	4	0	0.02
CM0CSR/CMS			03/17/09	03/17/09	2	0	0.25
CM0CSR/CMM			03/15/09	03/15/09	0	2	0.11
CM0CSR/CMOUT_IE			03/21/09	03/21/09	1	0	0.26
CM0CNT			03/10/09	03/10/09	3	2	0.25
CM0COR			03/23/09	03/23/09	1	0	2.34
Counter_operation/CKS_100			03/19/09	03/19/09	1	0	0.01
Counter_operation/CKS_101			03/19/09	03/19/09	1	0	0.01
Counter_operation/CKS_110			03/19/09	03/19/09	1	0	0.01
Interrupt_Request/CMR_10			03/10/09	03/10/09	0	1	0.14

- "Detail Execution" sheet: like detail report, but summarize for each time running TMs.



Category	Date	Pass	Fail	CPU time (hour)
CM0STR	03/10/09	1	0	0.02
CM0CNT	03/10/09	3	2	0.25
CM0CNT	03/10/09	3	2	0.25
Interrupt_Request/CMR_10	03/10/09	0	1	0.14
Interrupt_Request/CMR_10	03/10/09	0	1	0.14
CM0CSR	03/12/09	0	1	0.02
CM0CSR/CMF	03/12/09	0	5	0.11
CM0CSR/CMF	03/12/09	0	5	0.11
CM0CSR/OVF	03/12/09	0	6	0.13
CM0CSR/OVF	03/12/09	0	6	0.13
CM0CSR/WRFLG	03/13/09	4	0	0.02
CM0CSR/WRFLG	03/13/09	4	0	0.02
CM0CSR/CMM	03/15/09	0	2	0.11
CM0CSR/CMM	03/15/09	0	2	0.11
CM0CSR/CMS	03/17/09	2	0	0.25
CM0CSR/CMS	03/17/09	2	0	0.25
Counter_operation/CKS_100	03/19/09	1	0	0.01
Counter_operation/CKS_100	03/19/09	1	0	0.01
Counter_operation/CKS_101	03/19/09	1	0	0.01
Counter_operation/CKS_101	03/19/09	1	0	0.01
Counter_operation/CKS_110	03/19/09	1	0	0.01
Counter_operation/CKS_110	03/19/09	1	0	0.01
CM0CSR/CMOUT_IE	03/21/09	1	0	0.26
CM0CSR/CMOUT_IE	03/21/09	1	0	0.26
CM0COR	03/23/09	1	0	2.34
CM0COR	03/23/09	1	0	2.34
CM0STR/STR5	03/25/09	2	1	0.06
CM0STR/STR5	03/25/09	2	1	0.06

Revision History

Rev.	Modified Contents	Approval	Reviewed by	Created by
1.0	New creation		Chau Nguyen 03/17/09	Duc Duong 03/17/09
1.1	- Add General operation flow chart - Modify option part for gen_sim.pl		Chau Nguyen 03/27/09	Duc Duong 03/27/09
1.2	- Remove memory usage information from report. - Modify the rule of user-defined script for comparing dumped files - Add section 6.3.3		Chau Nguyen 04/02/09	Duc Duong 04/02/09
1.3	- Change “building test pattern” to “compile test pattern” in figure 1.1.	A.Imoto 04/10/09	Thai M. Ng. 04/08/09	Duc Duong 04/08/09
1.4	- Update Figure 1.1 - Update Figure 2.1 and explanation - Update table 3.1 : Update option and output file - Update chapter 3.2 : Describe new structure of batch file - Update chapter 4.3 : Add user script definition to check log files - Update chapter 5.2 : Add explanation about report for specify category - Update chapter 6.1 : Add command for generate Stimulus Environment - Updated figure 6.2 : New batch file structure		Chau Nguyen 10/06/09	Son Tran 09/30/09
1.5	- Add version of Perl - Modify Figure 1.1 : Change name of reference document of Generate Environment phase. - Add relative documents - Update name of figure 3.1	A.Imoto 10/14/09	Chau Nguyen 10/13/09	Son Tran 10/07/09