**NIUEZ1**

**Coverage Plan**

Ramya Nerabetla

# COVERAGE INTENT

1. To fully-automate the coverage flow from running the weekly regression to providing coverage reports showing progress at various stages of the project.
2. To use project coverage as one of the metrics to determine coverage from the DV team.
3. To be able to extract coverage for individual subsystems and cumulative set of blocks with in the SOC.
4. To be able to highlight areas that have low-coverage.
5. To have one central area to look up the test list and exclusions per block.
6. To be able to incrementally review coverage and rope in tests and exclusions week after week.

# COVERAGE PLAN FOR NIUE

Below is the list of steps required to execute coverage driven validation on Niue.

1. Pre-coverage run exclusions (mostly include RAMs and IPs) are at –

*$SOS\_CLIENT/vsys/proj/niue1z1/asic/dv/top/tb/coverage/cover.mti.f*

1. Regression is run with coverage enabled by default.

Add ‘-coverage’ switch to simrun/alpha\_test

1. Regressions are run by multiple users and the results are dumped to the shared regression directory. As mentioned in Section 3, there was a problem with this approach where the merge tool crashed when the paths to the source files of identical names are not matching. Thus, the following script was developed to override the usernames of the file sources in <test.ucdb> once the test terminates.

*$SOS\_CLIENT/vsys/proj/niue1z1/asic/dv/top/tb/coverage/cover\_rename\_srcfilestring.sh*

1. This script is called automatically run once the test completes and is invoked by .all.cmd -> .post.cmd

.all.cmd is as follows:

*umask 2 && ./.gcc.cmd && ./.vlog.cmd && ./.vopt.cmd && ./.vsim.cmd ; ./.post.cmd*

.post.cmd is as follows:

*post\_process -c top:ospi\_test.basic -s 596779270 -r -p*

*grep -ir "/wa/.\*/rtl/.\*vopt-2718.\*\|/wa/.\*/rtl/.\*vopt-2685.\*\|/wa/.\*/rtl/.\*vopt-2241.\*" logs/vopt.log.warnings > rtl\_PROJ\_lint\_issues.rpt 2>/dev/null*

*grep -ir "/vsysip/.\*/rtl/.\*vopt-2718.\*\|/vsysip/.\*/rtl/.\*vopt-2685.\*\|/vsysip/.\*/rtl/.\*vopt-2241.\*" logs/vopt.log.warnings > rtl\_VSYSIP\_lint\_issues.rpt 2>/dev/null*

*cp -sf /sswork/gf22/niue1z1/rkalyana/regr/vsys/proj/niue1z1/asic/dv/top/tb/coverage/cover\_rename\_srcfilestring.sh .*

*./cover\_rename\_srcfilestring.sh*

|  |  |
| --- | --- |
| Test Result | *cover\_rename\_srcfilestring.sh* script Invocation |
| Pass | yes |
| Fail | No (ucdb is renamed – optional) |
| Killed | yes |
| Running | No |

1. Once all the tests complete, the corresponding ucdb’s are available in respective run directories. Merge the ucdb files using the following command –

*ignrc vcover parallelmerge -runmode local -j 16 -genlist -suppress 6821 -stats=perf -verbose -l parallelmerge.log -outname merge.ucdb*

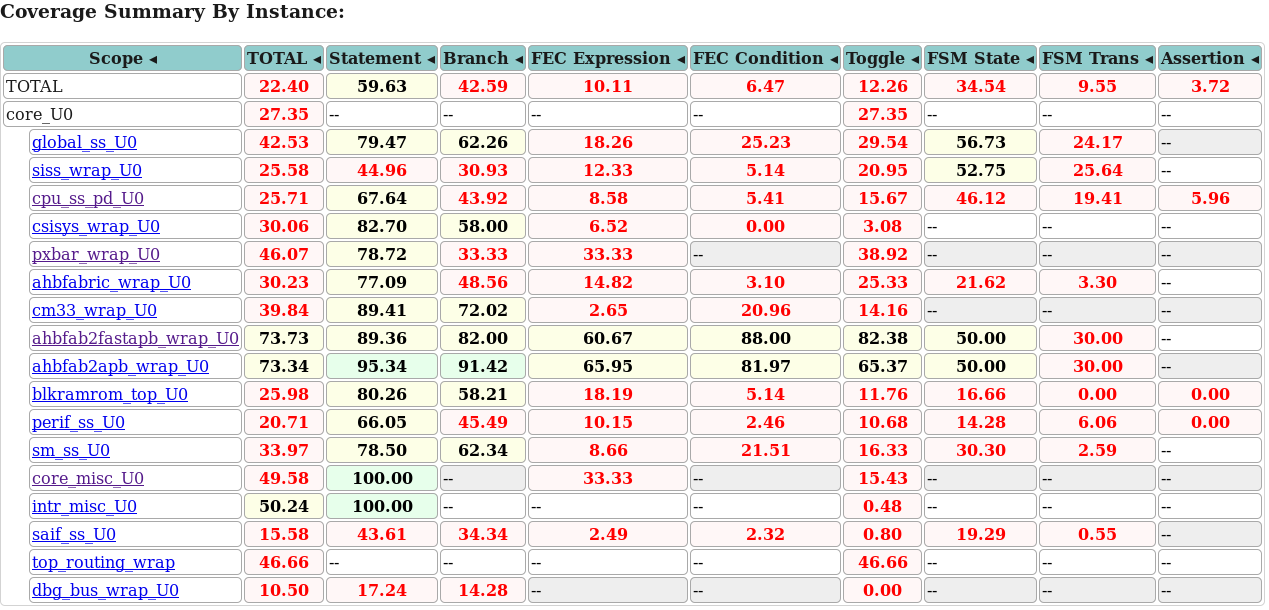
*ignrc vcover report -html -htmldir cov\_report -source -details merge.ucdb*

*firefox cov\_report/index.html&*

1. The html report is circulated to the DV team for review.

It includes coverage data for all the blocks inside the DUT.

For Example:



1. The DV owner Analysis the respective block coverage,
   * + 1. Adds new tests to the respective <block>\_regression.list
       2. Identifies new exclusions, reviews them (usually few/week) with the designer and gets a sign-off.
       3. Appends reviewed exclusions to cover.mti.f in a pre-defined format.

Format of user specific DV exclusions is as follows –

#<Block>\_<Date>\_Exclusions\_SignedOffBy\_<DesignOwner>

#Justification: <……..>

Exclusion1

Exclusion2

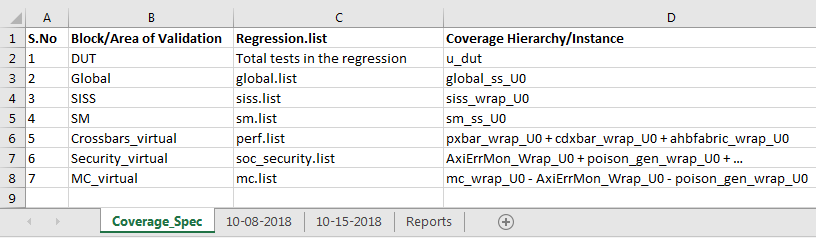
**NOTE**: Line exclusions are allowed only during ECO stage.

* + - 1. Checks in the exclusion file, the new tests and the regression list.

1. Next regression automatically picks the exclusions and new tests and the process continues.

# COVERAGE FLOW AUTOMATION

* + - 1. Prior to enabling coverage on a project, the following table must be clearly laid out. It is used as a spec by the coverage script.



**Block/Area of Validation:**

* Indicates the blocks or areas that would require a high-level coverage report. The final Coverage report spreadsheet will have the data reported only for these blocks.
* If the block has a suffix of – ‘\_virtual’, it means it doesn’t truly represent any hierarchical block in the DUT but could be a cumulative result of multiple sub-blocks spread-out across the SOC. A few simple examples are crossbars, security, MC etc.

**Regression.list**

* Indicates the regression list that represents the list of tests for that block/area of Validation.
* The coverage script will parse this list to evaluate the number of tests accounted for this regression.
* Any test that was specifically added for coverage purpose alone needs to have a suffix of ‘\_cov’. This will enable maintaining one single list / block and is easy for scripting the coverage flow. The script can always report the tests that were added for coverage separately.

**Coverage Hierarchy/Instance**

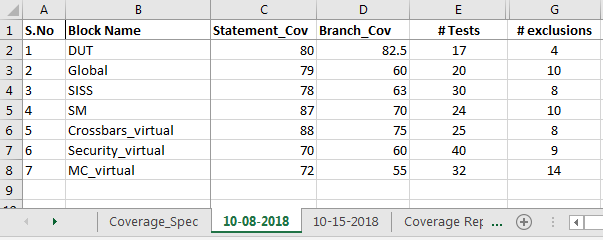
* Indicates keyword / Hierarchy in the html that the coverage script uses to parse and report the coverage data.
  + - 1. **Coverage Script:**

The primary inputs to the script are –

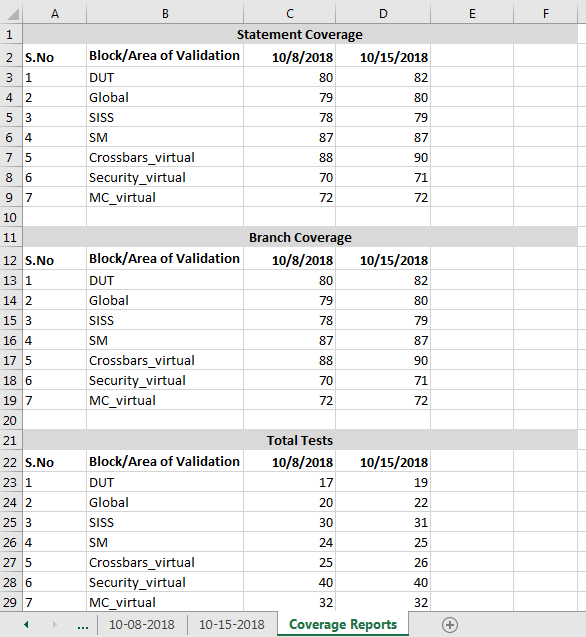
1. Coverage\_Spec: i.e. The table above
2. HTML file that has the coverage data.
3. Access to the tests/regressions/\* in-order to count the number of tests in the <block>.list
4. Access to *dv/top/tb/coverage/cover.mti.f* to count the number of exclusions.

Listed below are the coverage script capabilities -

**STEP-1**: To parse the html and append the coverage data to a new worksheet as shown below -



**STEP-2**: Append data to the Coverage report worksheet as follows to enable graph representation –



**STEP 3:** Draw graphs for statement coverage, Branch coverage, # of tests, # of exclusions per week as shown below –

