Accelerating SoC Verification Signoff using Save & Restart/Dynamic Test Reload Enhanced Regression flow

Vanshlata B, Samsung Semiconductor India R&D Ltd., Bangalore, India (b.vanshlata@samsung.com)
Divya M, Samsung Semiconductor India R&D Ltd., Bangalore, India (d.madhavarap@samsung.com)
Garima S, Samsung Semiconductor India R&D Ltd., Bangalore, India (s.garima@samsung.com)
Seonil Brian Choi, Samsung Electronics Co., Ltd., Seoul, Korea (seonilb.choi@samsung.com)

***Abstract* - With increasing complexity of SOC designs, Design and Verification flow faces tremendous challenges. Improvement in fabrication technology puts stress on Power, Performance and Area concerns, aggravating the tight schedules, and lesser time for Verification and Coverage. Although it's never possible to exercise 100% scenarios for an SoC for verification closure, it is pertinent that major bugs and loopholes are caught before MTO.**
This paper will introduce the Cadence vManagerCS integrated CR script that uses Save and Restart methods to significantly reduce simulation times, improve license utilization, increasing debug efficiency and bug identification and closure, and finally, early Verification Signoff. This paper will also describe the results of our usage of the tool on live projects and the productivity boost that was seen.

I. INTRODUCTION

Design and Verification flow is replete with multiple challenges that significantly impact the product cycle and time to market. Some of them are highlighted below:

- Use case identification
- Increasing Design Complexity (longer simulation times)
- Shorter time to market and verification schedules
- Regression and Coverage Closure
- Debug efficiency (simulation time vs active debug time)
- Power management challenges
- Security and functional safety
- Limited IP/VIP licences and LSF queue

A survey by Wilson Research Group[1] suggested debug cycles take 44% of engineers' time. This is shown in Fig 1. and Fig. 2.

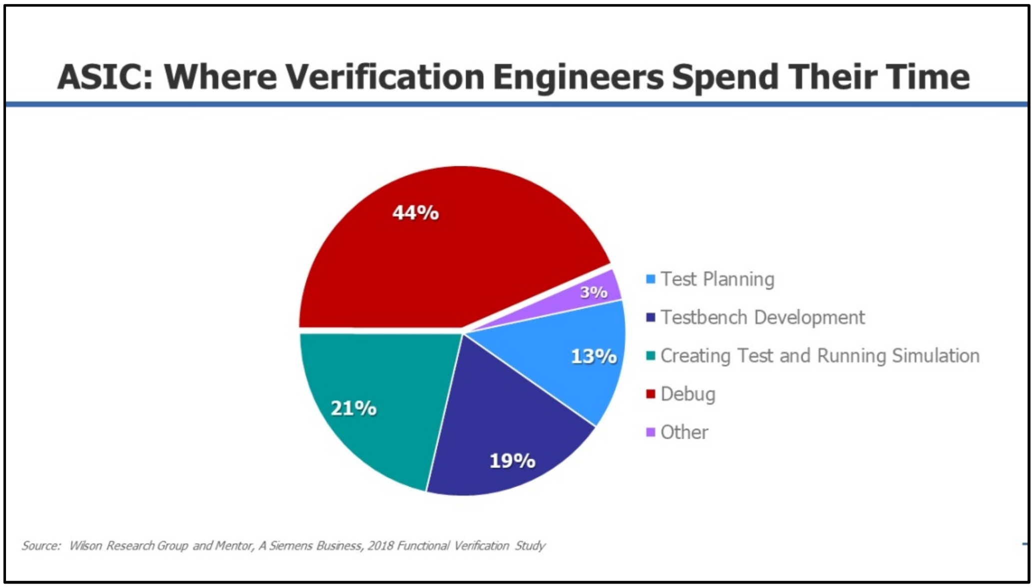


Fig. 1. Time spent by verification engineers

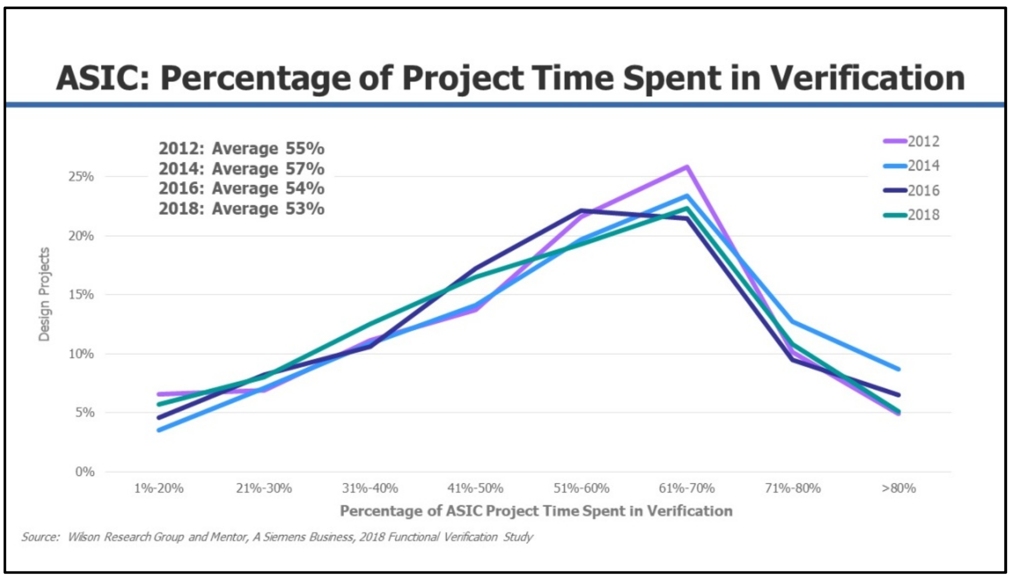


Fig. 2. Percentage of project time spent in verification

To prevent derailing time to market goals, efficient execution should be prioritized. Automation and EDA tool solutions have cut down a lot of development time, but longer simulation times result in lesser license efficiency (higher per test association time) and higher engineer wait times.

II. BACKGROUND

The idea behind Save and Restart (SnR) is that, all tests have to do some initial setup before starting a test scenario. Running this phase multiple times for every test takes a lot of time. If we can use the data and state of common sequence for multiple tests, this would minimize the time used by each test to do initial setup. This is illustrated in Fig. 3.

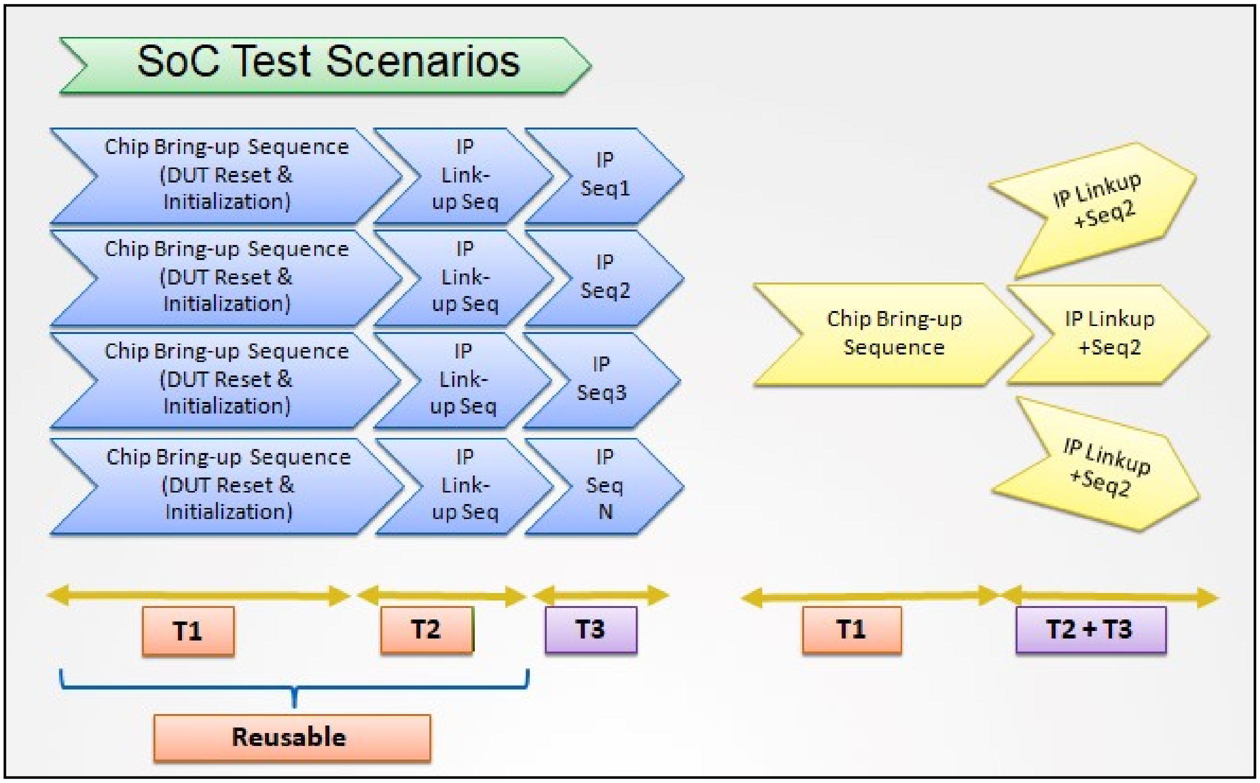


Fig. 3. Illustration for Save and Restart concept

Verilog provides inbuilt syntax for save and restore. Even Specman e had these capabilities. Different vendors have incorporated it to benefit their customers in suitable manner.

A. Existing Simulators With Save And Restart Feature

The VCS tool from Synopsys and Modelsim/Questasim from Mentor Graphics use the inherent *$save* task of verilog to provide this functionality.

The VCS syntax for existing UBUS example available in the standard UVM installation is shown below in Fig. 4 and 5.

```
// 1 Master - 1 Slave Test
class test_generic_1m_1s extends ubus_example_base_test;
  `uvm_component_utils(test_generic_1m_1s)
   ….
  virtual task main_phase(uvm_phase phase);
    uvm_sequence_base exec_seq;
    string seq_name;

    `uvm_info("main_phase", "Entering...", UVM_NONE)

    phase.raise_objection(this);
    super.main_phase(phase);

    if($test$plusargs("seq")) begin
      $value$plusargs("seq=%s", seq_name);
      $display("The value of sequence::%s", seq_name);
    end
    $cast(exec_seq, factory.create_object_by_name(seq_name));
    exec_seq.start(ubus_example_tb0.ubus0.masters[0].sequencer);

    phase.drop_objection(this);
    `uvm_info("main_phase", "Exiting...", UVM_NONE)
  endtask

endclass : test_generic_1m_1s
```

Fig. 4. Illustration for Test code

```
//Compilation
vcs -sverilog -ntb_opts uvm +incdir+../sv ubus_tb_top.sv -debug

//Simulate to a common state
./simv +UVM_TESTNAME=test_generic_1m_1s -ucli -i makerestore.in
```

```
run 100
## saving the state in saveddesign after 100 timeunits
save saveddesign
quit
```

```
//Simulate different stimulus
./simv +seq=read_modify_write_seq -ucli -i runrestore.in
./simv +seq=r8_w8_r4_w4_seq -ucli -i runrestore.in
```

```
## Restoring saved design
restore saveddesign
## simulation proceeds after 100 time units
run
```

Fig. 5. VCS Run command

B. Limtations With Exiting Simulators That Prevent Effective SnR Implementation

- Only saved Verilog & VHDL states, external states say C codes is skipped.
- External models states requires PLI,VPI mechanisms which has to be done manually.
- Dependency on machine configuration for save and Restart.
- Did not work with all built-in system tasks.

III. PROPOSED APPROACH

Due to previously mentioned limitations, we have used the **"Process Based Save and Restart (PBSR)"** provided by Cadence. It reduces the bring-up and debug time during development, and can also be integrated to run regressions and to achieve higher degree of functional coverage by getting to more meaningful portions of simulation faster and reseed the remaining run, with same benefits[2].

PBSR integrates with itself the **DTR (Dynamic Test Reload)** methodology. Here, a Package-based Testbench is used to add new code to the existing environment, which considerably reduces the testbench development time. A **DBS (Dynamic Base Snapshot)** is created and can be dynamically referenced by a **DTS (Dynamic Test Snapshot)** with new sequences during the restart portion of the test. DBS is the time-zero snapshot that contains full model (DUT, TB, initial UVM test classes). DTS contains a model of SystemVerilog package and a DBS that could be linked to it. The same has been illustrated in Fig. 6.

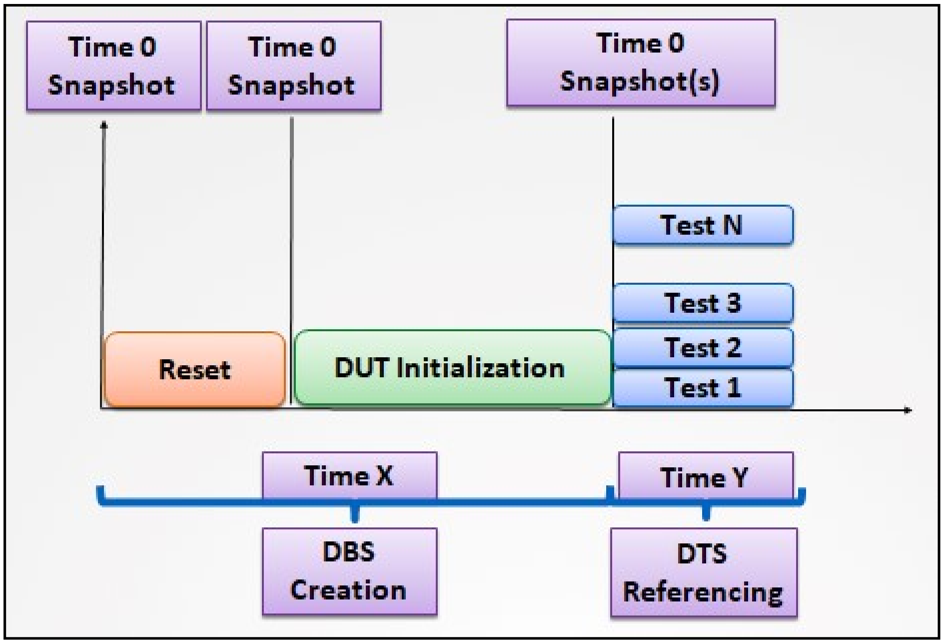


Fig. 6. Illustration for DBS and DTS concept

Without SnR :
Total regression time = $nX + Y_1 + Y_2 + \ldots\ldots + Y_n$

With SnR:
Total Regression time = $X_1 + X_2 + \ldots\ldots + X_m + Y_1 + Y_2 + \ldots\ldots + Y_n = mX_a + Y_1 + Y_2 + \ldots\ldots + Y_n$

Total Time saving = $nX - mX_a = (n-m)X$, if $X \sim X_a$

Where,
n = total no. of tests
m = total no. of boot snapshots
X = Time taken for a specific snapshot
Y = Time taken to complete remaining portion of a specific test
X_a = Average time taken for common point of interest (like DUT initialization)
$X_1, X_2, \ldots X_m$ = Time taken by \multiple snapshots
$Y_1, Y_2, \ldots. Y_n$ = Time taken to complete remaining portion of tests

Here, the magnitude of Y is in 2-10 hours whereas, $Y_1, Y_2, \ldots. Y_n$ are in the order of minutes to few hours.The magnitude of time saving increases with increasing no. of tests.

There is also a custom integrated Central Regression script that uses data from an input excel file to aide in identifying tests enabled for SnR, creating regressions suites with SnR syntax, creating SnR compatible builds, user defined snapshots, and enabling restart of multiple tests from the saved point of interest using UVM factory override for multiple test scenarios. The time saved is attributable to the common checkpoint that forms major portion of the test initialization (50-70% of simulation time). A sample input excel sheet is shown in Fig. 7.

	A	B	C	D	E	F	G	H	I
1	**Sl No.**	**Test ID**	**IP**	**Feature**	**Description**	**Test Command**	**SnR Status**	**Snapshot**	**Package Name**
2	1	A1	IP1	Feature1					
3	2	A2	IP1	Feature2					
4	3	A3	IP2	Feature1					
5	4	A4	IP2	Feature2					
6	5	A5	Common	Feature3					
7									
8									
9									
10									
11									
12	Block Name								
13									
14									

BLK_A BLK_B BLK_C

Fig. 7. Illustration for Regression Script input excel file

IV. IMPLEMENTATION

Save and Restart can be implemented by either of the two methods show in Fig. 8. The save and Restart methods are interchangeable.

Fig. 8. Illustration for SnR Implementation

A. Syntax for creating the DBS

xrun –top <other options>-mkdbssnap -name <dbsname> -process_save –clean -l -uvmnoautocompile +CHECKPOINT="snapshot_name" -input <save_file_name>.tcl -l <log_file_name>.log -access +rw +UVM_TESTNAME=<save_test_name>

1) *-input <save_file_name>.tcl* contains conditions for stop as shown in Fig. 9.

```
#set a breakpoint to stop at the end of configure_phase
uvm_phase -stop_at configure_phase -end
#run until the breakpoint is reached
run
#save the snapshot
save init_done -overwrite
exit
```

Fig 9. Illustration for Save tcl file

2) Tcl commands are used for saving, using *stop -object object_name, stop -line line_number, stop -time, stop - condition* etc. for synchronization

B. Syntax for creating DTS

xrun -r top <other options>-dbsname <name>[@<directory>] -dbssnap <snapshot_name>-process_save -uvmnoautocompile -input <restart_file_name>.tcl -l <log_file_name>.log -uvm -access +rwc +UVM_TESTNAME=<test_name><files_for_new_package>

1) *-process_save* is mandatory to enable SnR during elaboration and simulation.
2) *-uvmnoautocompile* option used when building the DTS to avoid recompiling design units used by DBS.
3) *-input <restart_file_name>.tcl* contains uvm_factory override commands as shown in Fig. 10.

```
# Command for restart
uvm_factory -override -by_type <base_sequence> <overriding_sequence>
uvm_factory -print
```

Fig 10. Illustration for Restart tcl file

V. PRACTICAL RESULTS

Fig. 11-13 show the comparison in time taken by tests before and after SnR implementation.

Time taken for Snapshot creation = 103 minutes.

```
xcelium> exit
xmsim: Memory Usage - Final: 14118.5M, Peak: 14118.6M, Peak virtual: 14488.8M
xmsim: CPU Usage - 19.0s system + 5813.2s user = 5832.2s total (5872.9s, 99.3% cpu)
TOOL: xrun(64)  18.09-a001: Exiting on Apr 18, 2020 at 17:32:47 KST  (total: 01:43:24)
```

Fig 11 Illustration showing time taken to create snapshot

Test 1:

Time taken = 137 minutes

Saving = (137-(7+103))*100/137 = 19.7% per test

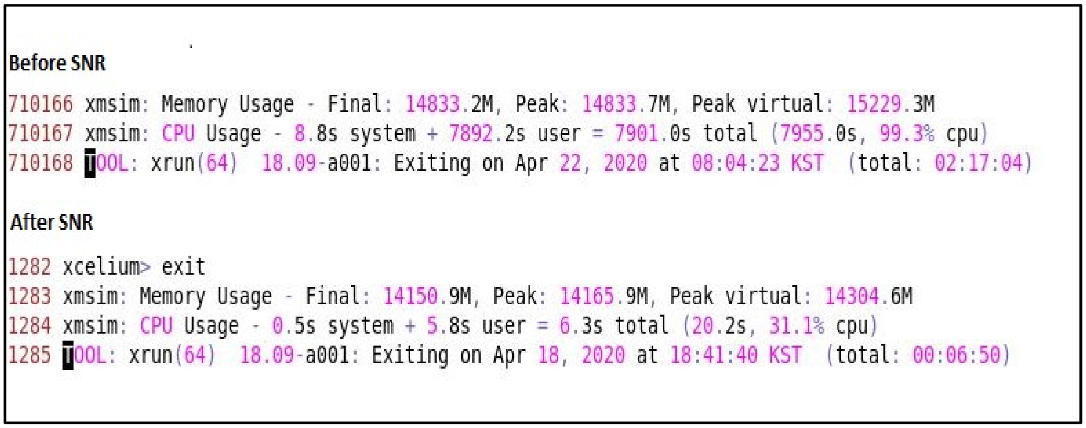


Fig 12. Illustration showing time taken by Test 1

Test 2:

Time taken = 162 minutes

Saving = (162-(30+103))*100/162 = 18.5% per test

Cumulative Saving = (162+137 –(30+7+103))*100/(162+137) =53.17%

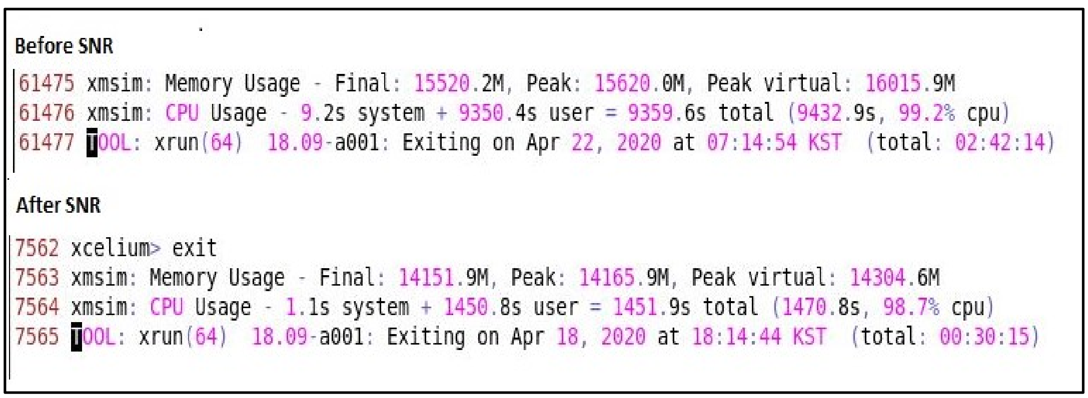


Fig 13. Illustration showing time taken by Test 2

As the number of tests per snapshot increases, the cumulative time saving increases gradually. Also, with LSF queue and multiple license slots, parallel runs reduce runtime considerably. With long-running tests, the time saving is drastically more with SnR. Fig. 14. shows the comparison of time taken before and after SnR implementation for few blocks of a live SoC Project based on data from Table 1.

Table 1. Block wise time saving

Block Name	No. of SNR tests	Total time for non SNR tests	Total time for SNR tests	Total Time saving (hrs)	Avg Time saving per test (hrs)	%age saving
Block A	3	7.45	1.68	5.77	1.92	77.48
Block B	15	57.12	8.11	38.82	2.59	85.80
Block C	41	748.89	546.25	202.63	4.94	27.06
Block D	57	140.11	34.02	106.10	1.86	75.72
Block E	97	252.64	93.74	158.90	1.64	62.89
Block F	37	66.49	15.86	66.49	1.80	76.14
Block G	11	26.99	6.72	20.27	1.84	75.10
Block H	43	121.96	45.15	76.81	1.79	62.98
Total	304	1421.65	751.54	675.79	2.22	47.14

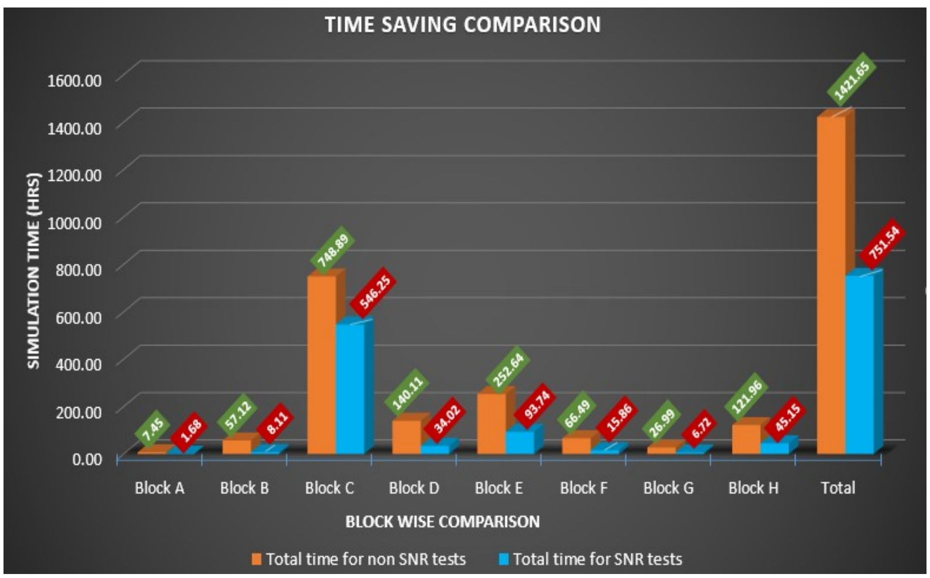


Fig. 14. Block wise time saving comparison

VI. CONCLUSION

Productivity improvement with SnR enabled regression compared to normal regression on a live project was significant.

A. Benefits

1) Faster incremental test development through Dynamic test pkg. (DTS technique)
2) Optimizes regressions to efficiently utilize LSF farm and licenses.
3) Deployment in current live project has led to saving of weeks of man hours and early bug closures with reduced failure triage time.
4) Per test saving of 1-10 hours (10-90% time saving).
5) Per block (with 60-400 tests and 2-3 snapshots) improvement of 50-60% regression time in a mixed regression (2-3X gain).
6) Per block (with 60-400 tests and 2-3 snapshots) improvement of 60-80% regression time in complete SnR regression (3-4x gain).

7) 2-3 man weeks of time saving on every label running full Central regression with incremental runs (containing 4K -6K tests with 30-40 snapshots) with 80-90% time saving (3-4x gain).
8) Majority of CR bugs caught and resolved within 1st to 2nd CR run as debug cycle times are reduced, leading to lesser incremental runs per label.

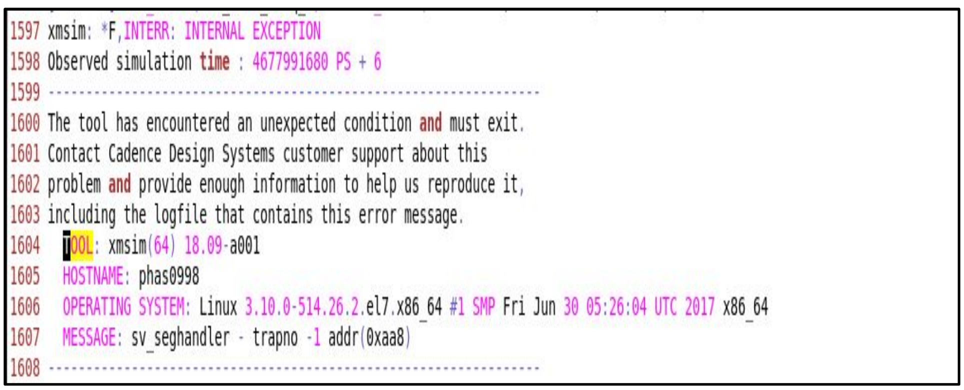
B. Limitations

1) Tests are uvm_component objects, and hierarchy can't be changed once UVM phasing has started. So, uvm_sequence is used to create test object to enable on the fly factory override in PBSR, needs environmental changes. One-time effort.
2) Package based environment required to facilitate DTR to modify sequences on the go.
3) Make sure that the Tcl override command (in restart.tcl) is executed before the sequence is created (in base_sequence).
4) Sequence overriding is possible only by the derived classes over their base classes.
5) Cannot use RTL hierarchical references in TB, instead interface connections need to be defined.

In spite of limitations to this method, the productivity improvements far outweigh the time/effort invested for environmental modifications, and can be carried forward to any project without much effort.

C. Issues faced during Implementation

1) vManager session log files (local_log.log) with missing simulation log caused vm_scan.pl to detect fake failure/pass(failure with negative filters). (Resolved)
2) Internal exception due to sv_seghandler error as shown in Fig. 15. (Resolved in 18.09-t001)



```
xmsim: *F,INTERR: INTERNAL EXCEPTION
Observed simulation time : 4677991680 PS + 6
-----------------------------------------------------------------
The tool has encountered an unexpected condition and must exit.
Contact Cadence Design Systems customer support about this
problem and provide enough information to help us reproduce it,
including the logfile that contains this error message.
  TOOL: xmsim(64) 18.09-a001
  HOSTNAME: phas0998
  OPERATING SYSTEM: Linux 3.10.0-514.26.2.el7.x86_64 #1 SMP Fri Jun 30 05:26:04 UTC 2017 x86_64
  MESSAGE: sv_seghandler - trapno -1 addr(0xaa8)
-----------------------------------------------------------------
```

Fig. 15. Image showing sv_seghandler error

3) Core dumped error with override failure (Resolved in 18.09-t001)

VII. REFERENCES

[1] https://www.mentor.com/products/fv/events/2020-wilson-research-group-verification-survey-results

[2] Improving Constrained Random Testing by Achieving Simulation Verification Goals through Objective Functions, Rewinding and Dynamic Seed Manipulation" - Eldon G. Nelson, Intel Corp at DVCon US 2017.

[4] Ed Powell, Ron Thurgood and Aneesh Samudrala. Hewlett Packard Enterprise "Using Save/Restore is Easy, Right? A User's Perspective on Deploying Save/Restore in a Mature Verification Methodology" in DVCON US 2019

[5] Accellera "Universal Verification Methodology (UVM) 1.1 User's Guide", May 18, 2011

[6] IEEE Computer Society and the IEEE Standards Association Corporate Advisory