Optimization techniques to improve simulation memory performance

Aditya Musunuri
Freescale Semiconductor, Inc.

Narayana Koduri
Synopsys

September 18th, 2015
SNUG Austin

Agenda

Introduction

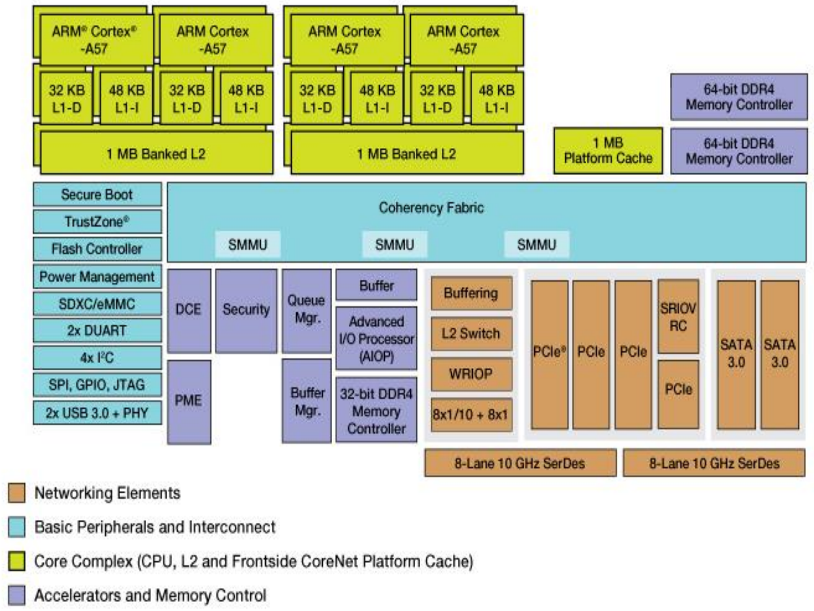
Introduction

- With increase in SoC Design size & complexity
  - The complexity of its verification is increasing tremendously
    - A more complex testbench environment
    - Increased number of testcases
- This effected our verification performance goals and prompted us looking at optimizing the same
  - Performance metrics are:
    - The time to perform the task, whether it is compilation or simulation
    - The memory utilization while doing it
- Our focus here is on the runtime memory utilization for running regression
  - LSF resources required us to limit the memory to 6GB
  - Secondarily, we also looked at optimizing runtime memory with dumping

SoC Design & Environment

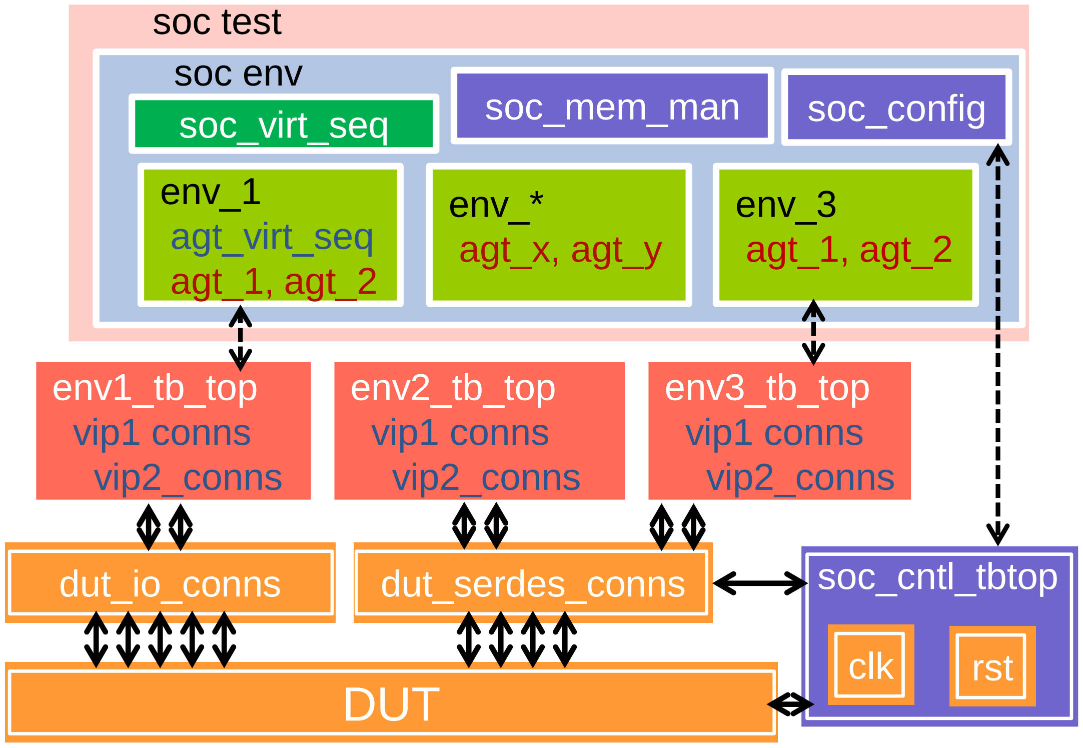
SoC Design

- The QorIQ® LS2 processor
  - 8 general purpose ARM Cortex®-A57 cores
  - Freescale’s second generation data path acceleration architecture (DPAA2)
  - Advanced IO Processing with in-line or fully autonomous networking functions
  - Upto eight 10 Gb/s and eight 1 Gb/s Ethernet interfaces
  - Next-generation SATAIII, USB3 and PCIe® controllers (supporting SR-IOV) controllers



The SoC Design

Testbench Environment



SoC Testbench Block Diagram

Methodology

- Verilog, System Verilog, UVM, C++, ROCOO (Freescale Internal - Random On Chip Object Oriented Language)

Module Part

- DUT_*_CONNECTION : MUX for the IO/SERDES pins in the DUT
- <env_name>_TB_TOB : VIP connections to DUT. An env may contain sub-envs

Program Part

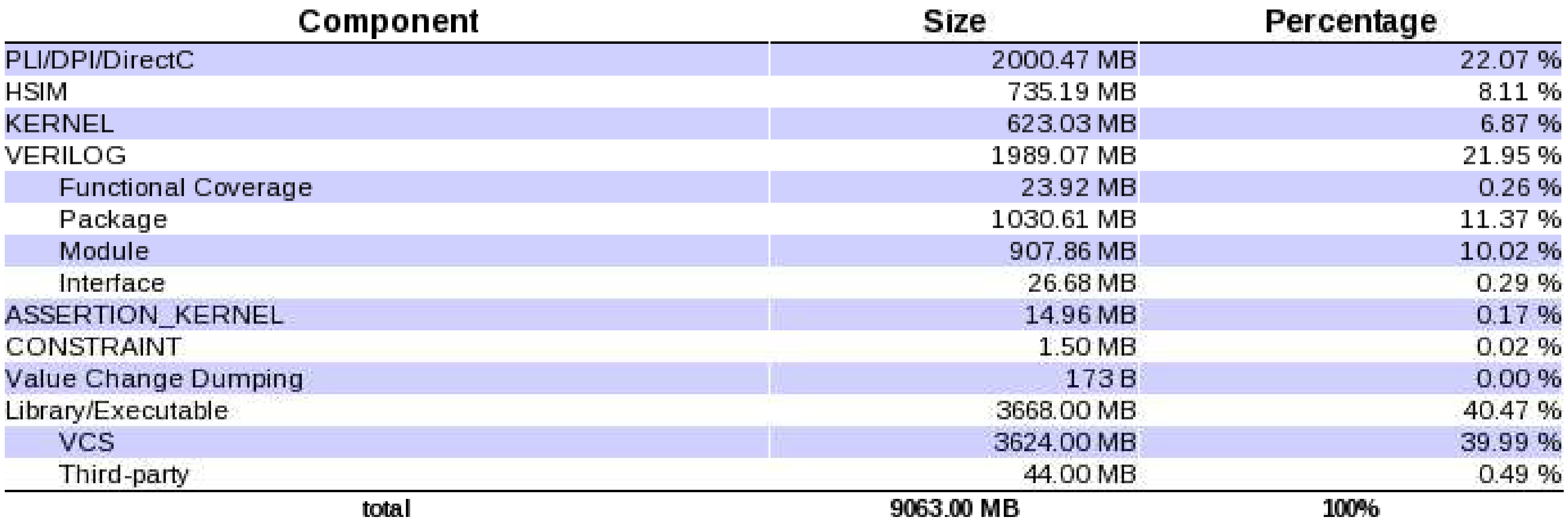
- SOC_TEST : Executes a Virtual Sequence on Virtual Sequencer for each phase
- SOC_ENV : Wrapper for all the various envs. Also includes SOC related classes
- <env_name>_ENV : A specific env for each block/driver/VIP

Findings & Results

Regression Setup

VCS Simprofiler

- Enable VCS Simprofiler
  - Add "`-simprofile`" compile option
  - Add "`-simprofile time`" runtime option
  - Add "`-simprofile mem`" runtime option
  - Generate profile reports using command "`profrpt`"
- Simprofile is generated in text and html formats

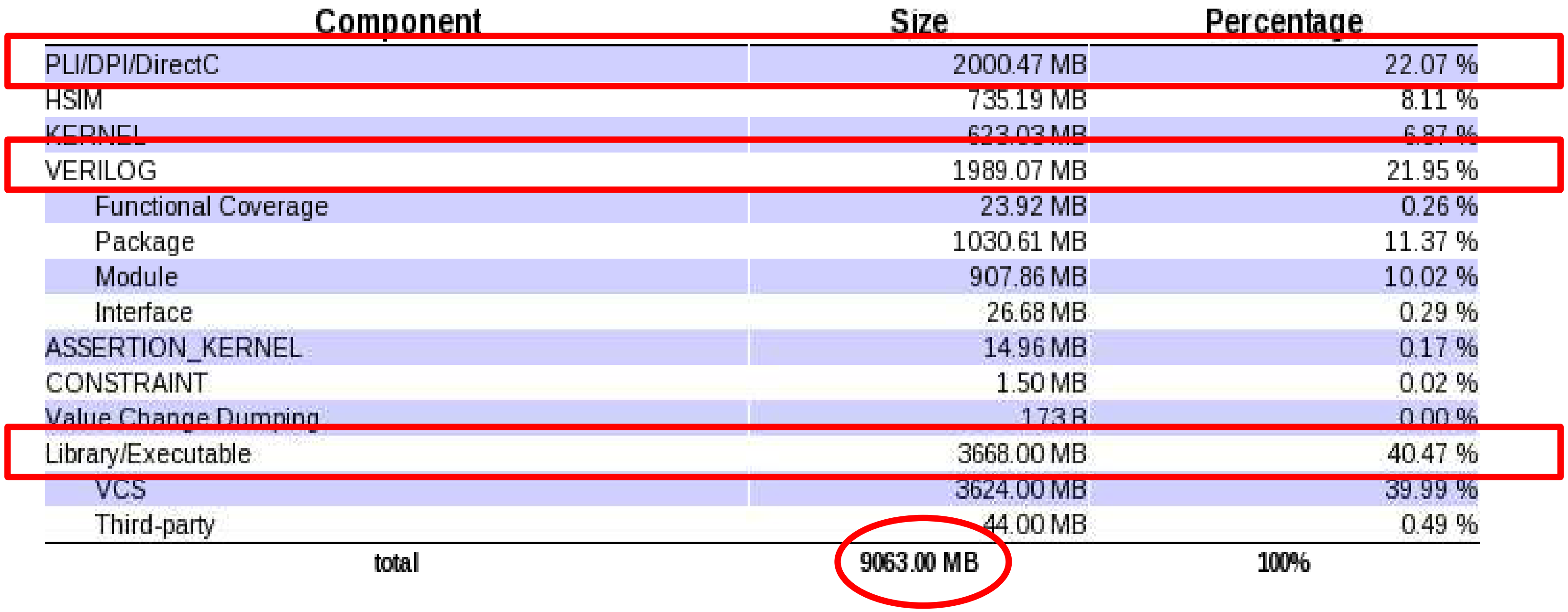


Component	Size	Percentage
PLI/DPI/DirectC	2000.47 MB	22.07 %
HSIM	735.19 MB	8.11 %
KERNEL	623.03 MB	6.87 %
VERILOG	1989.07 MB	21.95 %
Functional Coverage	23.92 MB	0.26 %
Package	1030.61 MB	11.37 %
Module	907.86 MB	10.02 %
Interface	26.68 MB	0.29 %
ASSERTION_KERNEL	14.96 MB	0.17 %
CONSTRAINT	1.50 MB	0.02 %
Value Change Dumping	173 B	0.00 %
Library/Executable	3668.00 MB	40.47 %
VCS	3624.00 MB	39.99 %
Third-party	44.00 MB	0.49 %
total	9063.00 MB	100%

Findings: Simprofiler

Original Setup

- Memory consumption is 9GB



Component	Size	Percentage
PLI/DPI/DirectC	2000.47 MB	22.07 %
HSIM	735.19 MB	8.11 %
KERNEL	623.03 MB	6.87 %
VERILOG	1989.07 MB	21.95 %
Functional Coverage	23.92 MB	0.26 %
Package	1030.61 MB	11.37 %
Module	907.86 MB	10.02 %
Interface	26.68 MB	0.29 %
ASSERTION_KERNEL	14.96 MB	0.17 %
CONSTRAINT	1.50 MB	0.02 %
Value Change Dumping	173 B	0.00 %
Library/Executable	3668.00 MB	40.47 %
VCS	3624.00 MB	39.99 %
Third-party	44.00 MB	0.49 %
total	9063.00 MB	100%

- The key contributors from the report are :
  - PLI/DPI/DirectC – 2 GB
  - Verilog (Module, Package) – 1.9 GB
  - Library Executable – 3.6 GB

Investigation: PLI/DPI/DirectC

- The memory consumed in this category is due to:
  - Debug capabilities enabled with PLI tab, DPI and DirectC Interfaces
- Our setup doesn’t have major DPI/DirectC interfaces. Hence, we focused on PLI based debug capabilities
- Looking at our setup, we found “`-debug`” option, this enables :
  - Several debug access capabilities such as forcing, writing or depositing nets, registers, etc.
  - Setting value and time breakpoints
- Debug capabilities are redundant for performing regression which has 7000 testcases

Action : PLI/DPI/DirectC

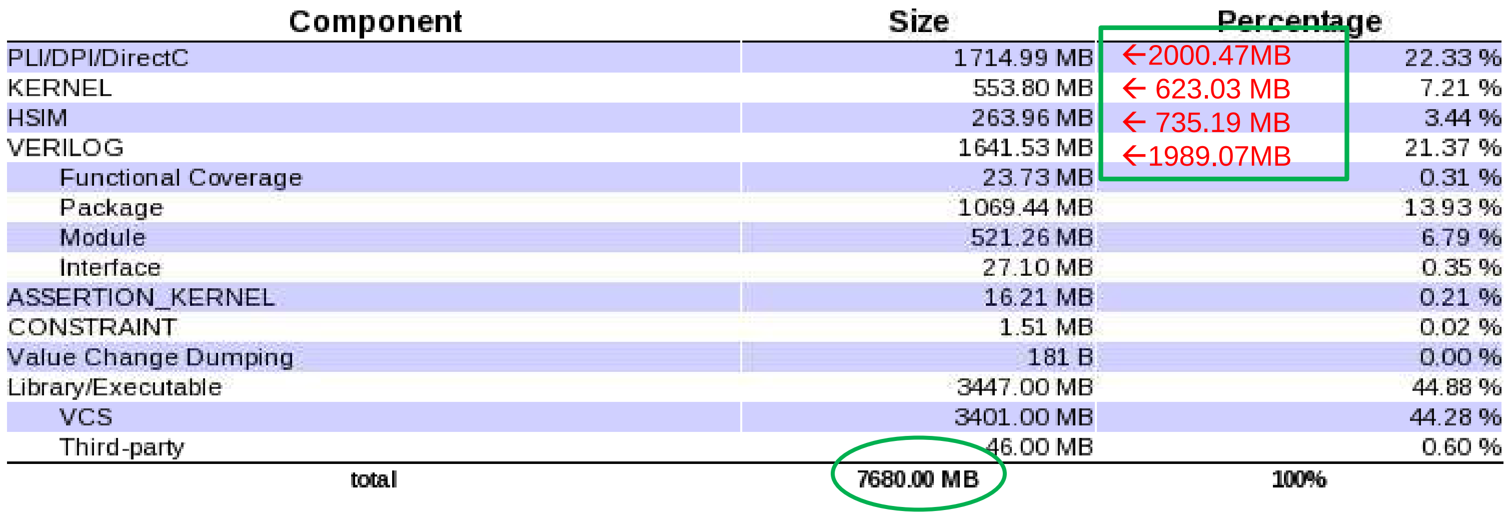
- Replace `“-debug”` option with `“-debug_access+r”` which only allows read capabilities
  - But, as a SoC team we were not sure what debug capabilities are needed

- We leveraged VCS PLI learn capability:
  - This can learn what debug capabilities are needed during simulation. Enabled by runtime switch `“+vcs+learn+pli”`
  - Generates `learn_pli.tab` file which can be specified with either `“+applylearn+learn_pli.tab”` or `“-P learn_pli.tab”` at compile time

Results (1)

Limiting Debug Capabilities

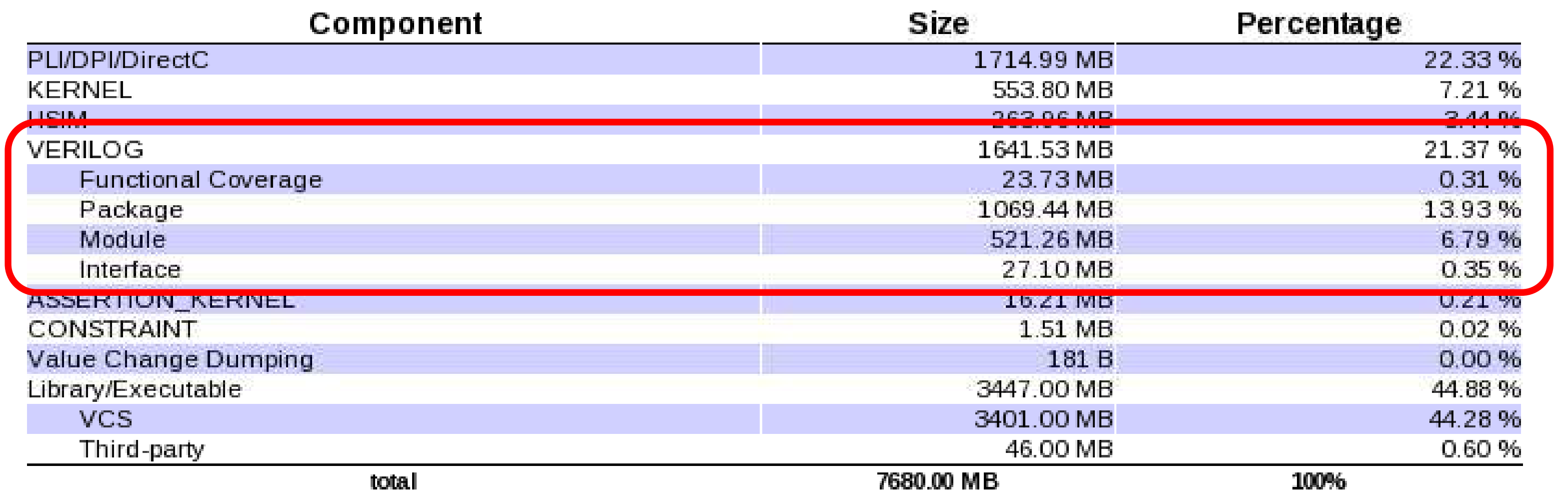
- Simprofiler shown below is generated after limiting debug capabilities
  – Improved memory consumption by 15% down to 7.6GB from 9GB



Component	Size		Percentage
PLI/DPI/DirectC	1714.99 MB	←2000.47MB	22.33 %
KERNEL	553.80 MB	← 623.03 MB	7.21 %
HSIM	263.96 MB	← 735.19 MB	3.44 %
VERILOG	1641.53 MB	←1989.07MB	21.37 %
Functional Coverage	23.73 MB		0.31 %
Package	1069.44 MB		13.93 %
Module	521.26 MB		6.79 %
Interface	27.10 MB		0.35 %
ASSERTION_KERNEL	16.21 MB		0.21 %
CONSTRAINT	1.51 MB		0.02 %
Value Change Dumping	181 B		0.00 %
Library/Executable	3447.00 MB		44.88 %
VCS	3401.00 MB		44.28 %
Third-party	46.00 MB		0.60 %
total	7680.00 MB		100%

Findings: Simprofiler

- The other component is Verilog
  - This component shows memory consumed due to modules, packages, interfaces and functional coverage



Component	Size	Percentage
PLI/DPI/DirectC	1714.99 MB	22.33 %
KERNEL	553.80 MB	7.21 %
USIM	263.06 MB	3.44 %
VERILOG	1641.53 MB	21.37 %
Functional Coverage	23.73 MB	0.31 %
Package	1069.44 MB	13.93 %
Module	521.26 MB	6.79 %
Interface	27.10 MB	0.35 %
ASSERTION_KERNEL	16.21 MB	0.21 %
CONSTRAINT	1.51 MB	0.02 %
Value Change Dumping	181 B	0.00 %
Library/Executable	3447.00 MB	44.88 %
VCS	3401.00 MB	44.28 %
Third-party	46.00 MB	0.60 %
total	7680.00 MB	100%

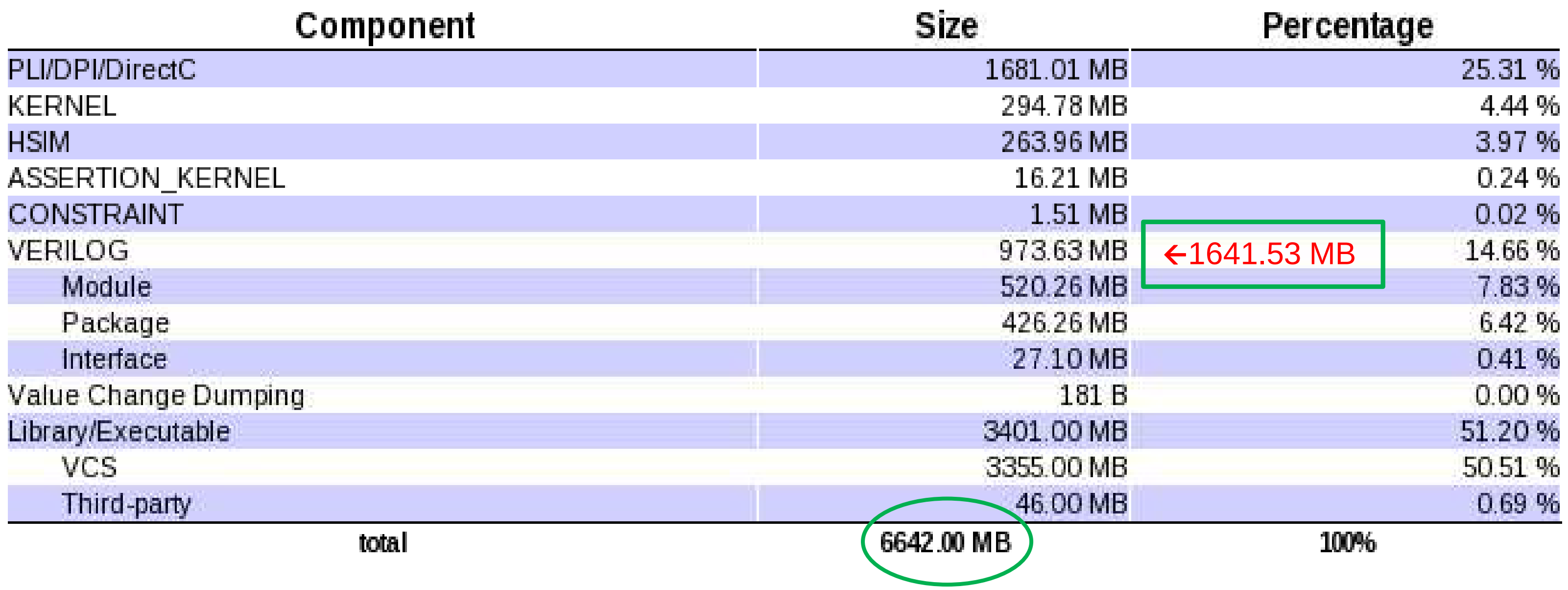
Investigation & Action: Verilog

- Identified that some of our VIPs are enabled with transaction coverage
  - This is leading to higher memory consumption
  - This transaction coverage data is not required as part of regression setup
- Disabled coverage data by adding a VCS option `“-covg_disable_cg”`

Results(2)

Disabling coverage data

- Improved memory consumption by another 13%, down to 6.6GB from 7.6GB

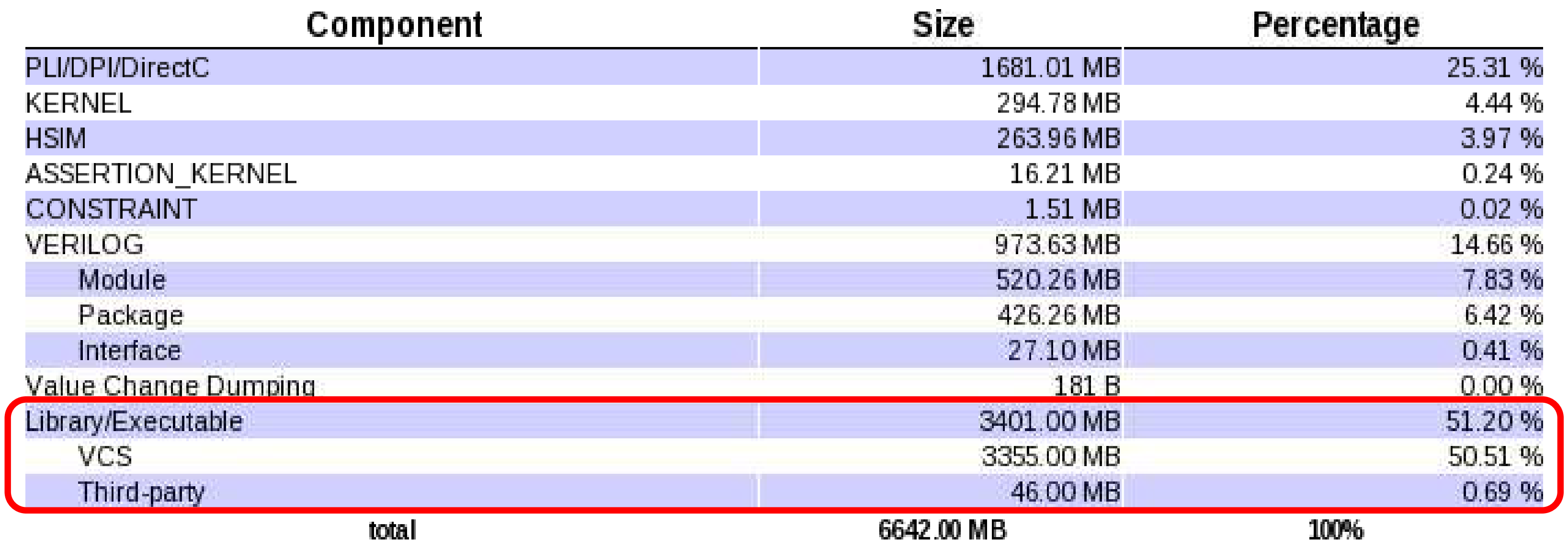


Component	Size	Percentage
PLI/DPI/DirectC	1681.01 MB	25.31 %
KERNEL	294.78 MB	4.44 %
HSIM	263.96 MB	3.97 %
ASSERTION_KERNEL	16.21 MB	0.24 %
CONSTRAINT	1.51 MB	0.02 %
VERILOG	973.63 MB	14.66 %
Module	520.26 MB	7.83 %
Package	426.26 MB	6.42 %
Interface	27.10 MB	0.41 %
Value Change Dumping	181 B	0.00 %
Library/Executable	3401.00 MB	51.20 %
VCS	3355.00 MB	50.51 %
Third-party	46.00 MB	0.69 %
total	6642.00 MB	100%

←1641.53 MB

Findings: Simprofiler

- The other major component is Library Executable
  - This is related to *vcs* executable and any 3rd party programs part of the executable



Component	Size	Percentage
PLI/DPI/DirectC	1681.01 MB	25.31 %
KERNEL	294.78 MB	4.44 %
HSIM	263.96 MB	3.97 %
ASSERTION_KERNEL	16.21 MB	0.24 %
CONSTRAINT	1.51 MB	0.02 %
VERILOG	973.63 MB	14.66 %
Module	520.26 MB	7.83 %
Package	426.26 MB	6.42 %
Interface	27.10 MB	0.41 %
Value Change Dumping	181 B	0.00 %
Library/Executable	3401.00 MB	51.20 %
VCS	3355.00 MB	50.51 %
Third-party	46.00 MB	0.69 %
total	6642.00 MB	100%

Investigation & Action : Library Executable, Setup

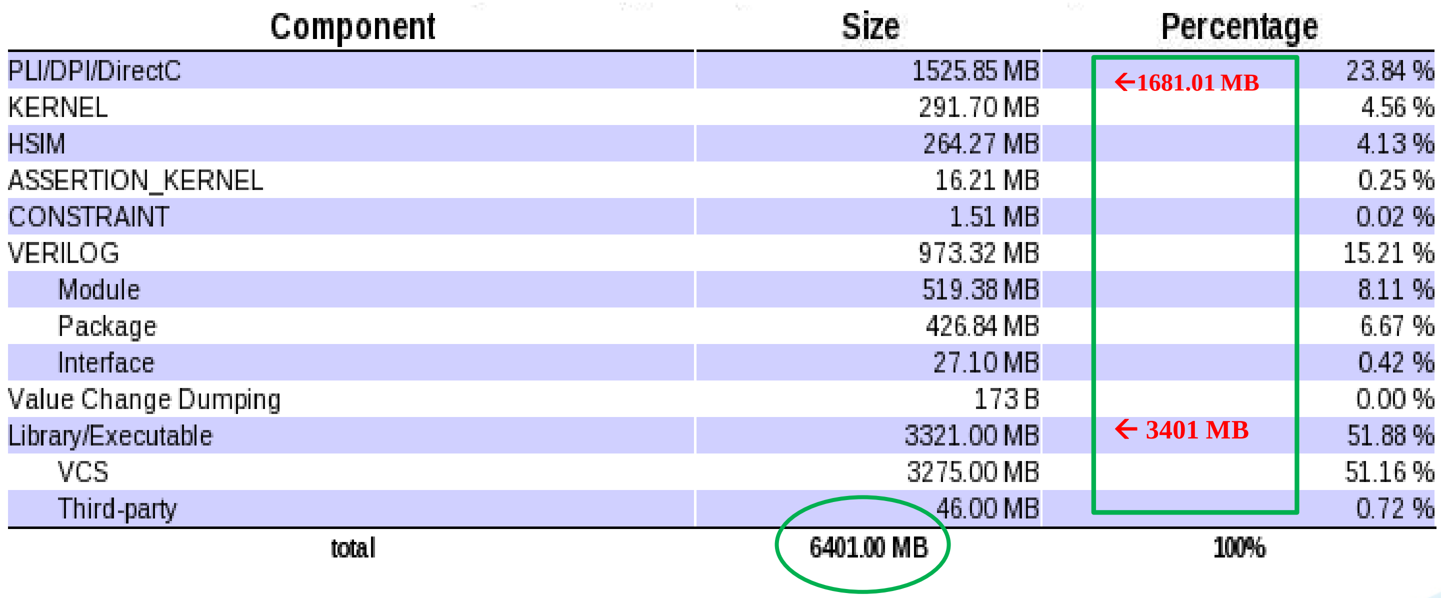
- This needed additional investigation with Synopsys® team
  - We found that this is an overhead caused by a message logger of some specific VIPs in the testbench environment
  - We replaced VCS option `“+vpi”` with `“+vpi+1”` option to eliminate the memory overhead
  - This option limits the behavioural information at compile-time, but preserves the structural information
- After further investigation with Synopsys team, identified and removed following redundant legacy options :
  - `${NOVAS_HOME}/share/PLI/VCS/LINUXAMD64/pli.a`
    `-P ${NOVAS_HOME}/share/PLI/VCS/LINUXAMD64/novas.tab`
  - `+memcbk`
  - `+vcsd`

FSDB dump is enabled by option “-debug_access”

Results (3)

Final setup

- Simprofiler shown below is generated with:
  - Replace `“+vpi”` → `“+vpi+1”`
  - Removed `+memcbk, +vcsd, ${NOVAS_HOME}/share/PLI/VCS/LINUXAMD64/pli.a -P ${NOVAS_HOME}/share/PLI/VCS/LINUXAMD64/novas.tab`
  - Improved memory consumption by another 3%, down to 6.4GB from 6.6GB



Component	Size		Percentage
PLI/DPI/DirectC	1525.85 MB	←1681.01 MB	23.84 %
KERNEL	291.70 MB		4.56 %
HSIM	264.27 MB		4.13 %
ASSERTION_KERNEL	16.21 MB		0.25 %
CONSTRAINT	1.51 MB		0.02 %
VERILOG	973.32 MB		15.21 %
Module	519.38 MB		8.11 %
Package	426.84 MB		6.67 %
Interface	27.10 MB		0.42 %
Value Change Dumping	173 B		0.00 %
Library/Executable	3321.00 MB	← 3401 MB	51.88 %
VCS	3275.00 MB		51.16 %
Third-party	46.00 MB		0.72 %
total	6401.00 MB		100%

Final Results

Summary for regression setup

- Following table summarizes the performance results observed with original and regression setup

2014.12-SP1	Base Numbers	Final Numbers (Regression setup)	Progress
Setup	Original	Updated based on Simprofiler findings	
Runtime (secs)	806	583	28%
Runtime Mem (GB)	9	6.0	33%

- Regression Setup (Compile Time Options):
  - Replace "`-debug`" with "`-debug_access+r -P learn_pli.tab`"
  - Replace "`+vpi`" with "`+vpi+1`"
  - Enable `-covg_disable_cg`
  - Remove legacy options "`novas tab`", "`+memcbk`" and "`+vcsd`"

Findings & Results

Dump Enabled

Waveform Dumping

- Dumping is only enabled to debug the test failures
  - By default, regression setup does not enable waveform dumping

- Flow has been setup to dump either vpd or fsdb
  - Flow triggers appropriate UCLI script and dumps either vpd or fsdb based on user's requirement
  - Dump is enabled for complete testbench, design and all hierarchies

Dump Enabled

- Following table summarizes the performance results observed with original and regression setup

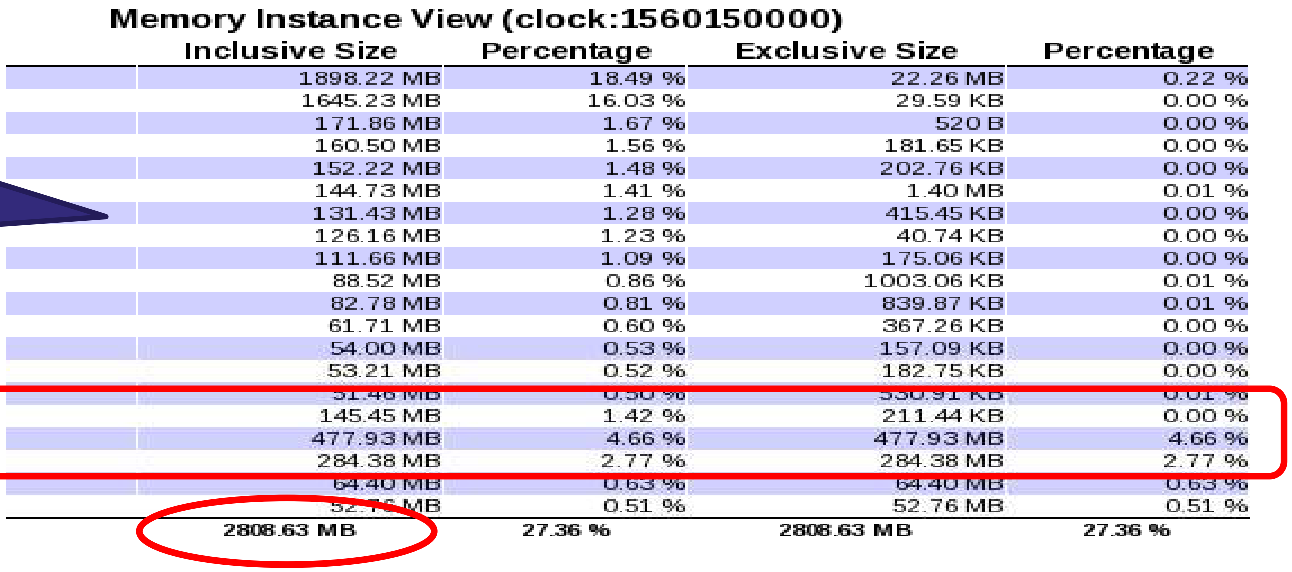
2014.12-SP1	Base (vpd)	Final (vpd)	Final (fsdb)	Progress
Setup	Original	Regression	Regression + FSDB setup	
Runtime (secs)	1469	1336	1005	31%
Runtime Mem (GB)	13.2	10	9.6	27%
Dump Size (MB)	149	149	120	19%

- FSDB Setup, Runtime :
  - `setenv FSDB_DUMPER_VDI 1` (Default starting 2015.09)
  - `+fsdb+skip_cell_instance=3, +fsdb+packedmda+struct`

Findings: Simprofiler

Dump Enabled

- Simprofiler, memory instance view after enabling dump



Hierarchy removed for confidential reasons

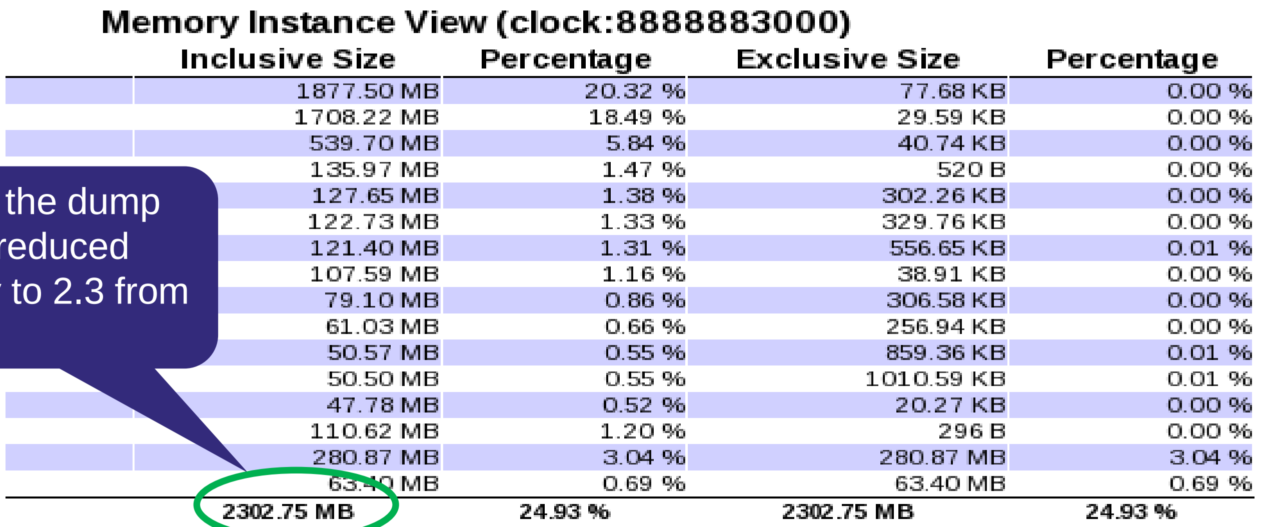
Memory Instance View (clock:1560150000)

	Inclusive Size	Percentage	Exclusive Size	Percentage
	1898.22 MB	18.49 %	22.26 MB	0.22 %
	1645.23 MB	16.03 %	29.59 KB	0.00 %
	171.86 MB	1.67 %	520 B	0.00 %
	160.50 MB	1.56 %	181.65 KB	0.00 %
	152.22 MB	1.48 %	202.76 KB	0.00 %
	144.73 MB	1.41 %	1.40 MB	0.01 %
	131.43 MB	1.28 %	415.45 KB	0.00 %
	126.16 MB	1.23 %	40.74 KB	0.00 %
	111.66 MB	1.09 %	175.06 KB	0.00 %
	88.52 MB	0.86 %	1003.06 KB	0.01 %
	82.78 MB	0.81 %	839.87 KB	0.01 %
	61.71 MB	0.60 %	367.26 KB	0.00 %
	54.00 MB	0.53 %	157.09 KB	0.00 %
	53.21 MB	0.52 %	182.75 KB	0.00 %
	51.46 MB	0.50 %	530.91 KB	0.01 %
	145.45 MB	1.42 %	211.44 KB	0.00 %
	477.93 MB	4.66 %	477.93 MB	4.66 %
	284.38 MB	2.77 %	284.38 MB	2.77 %
	64.40 MB	0.63 %	64.40 MB	0.63 %
	52.76 MB	0.51 %	52.76 MB	0.51 %
	2808.63 MB	**27.36 %**	**2808.63 MB**	**27.36 %**

- From the simprofile, we can see that memory is being consumed for testbench, uvm, etc.
- We can restrict the dump to registers and DUT using FSDB Dump commands

Results

Experiment with limiting the dump to DUT



Memory Instance View (clock:8888883000)

Inclusive Size	Percentage	Exclusive Size	Percentage
1877.50 MB	20.32 %	77.68 KB	0.00 %
1708.22 MB	18.49 %	29.59 KB	0.00 %
539.70 MB	5.84 %	40.74 KB	0.00 %
135.97 MB	1.47 %	520 B	0.00 %
127.65 MB	1.38 %	302.26 KB	0.00 %
122.73 MB	1.33 %	329.76 KB	0.00 %
121.40 MB	1.31 %	556.65 KB	0.01 %
107.59 MB	1.16 %	38.91 KB	0.00 %
79.10 MB	0.86 %	306.58 KB	0.00 %
61.03 MB	0.66 %	256.94 KB	0.00 %
50.57 MB	0.55 %	859.36 KB	0.01 %
50.50 MB	0.55 %	1010.59 KB	0.01 %
47.78 MB	0.52 %	20.27 KB	0.00 %
110.62 MB	1.20 %	296 B	0.00 %
280.87 MB	3.04 %	280.87 MB	3.04 %
63.40 MB	0.69 %	63.40 MB	0.69 %
2302.75 MB	**24.93 %**	**2302.75 MB**	**24.93 %**

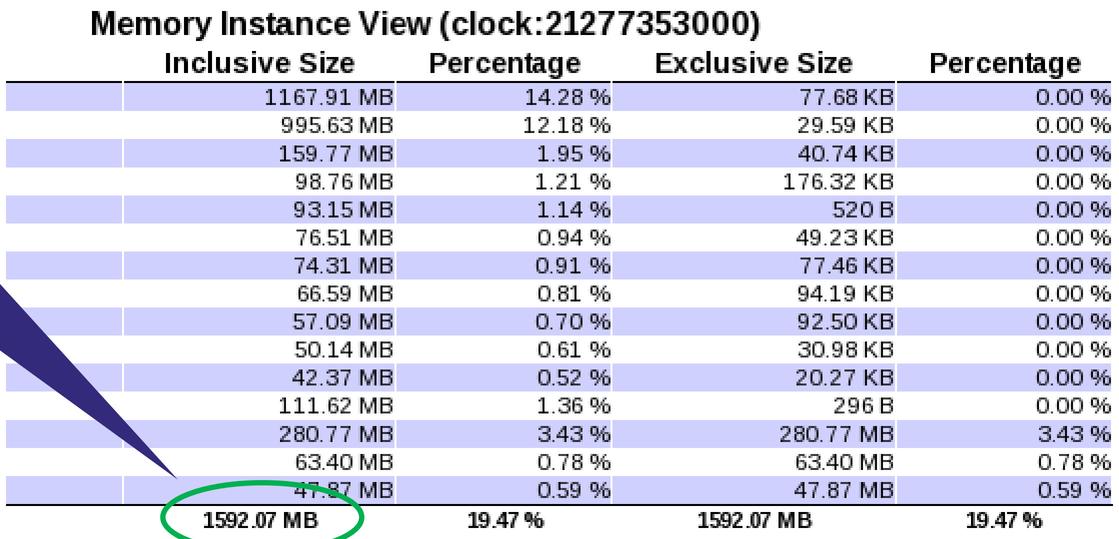
Limiting the dump to DUT reduced memory to 2.3 from 2.8

2014.12-SP1	Final (fsdb)	Ver1 (fsdb)	Progress
Setup	Regression	Regression+ Limit to DUT	
Runtime (secs)	1005	1115	-10%
Runtime Mem **Inst View** (GB)	2.8	2.3	18%
Runtime Mem (GB)	9.6	9.2	5%
Dump Size (MB)	120	65	46%

Results

Experiment limiting the dump to Registers & DUT

Limiting the dump to registers in DUT reduced memory to 1.6 from 2.8



Memory Instance View (clock:21277353000)

	Inclusive Size	Percentage	Exclusive Size	Percentage
	1167.91 MB	14.28 %	77.68 KB	0.00 %
	995.63 MB	12.18 %	29.59 KB	0.00 %
	159.77 MB	1.95 %	40.74 KB	0.00 %
	98.76 MB	1.21 %	176.32 KB	0.00 %
	93.15 MB	1.14 %	520 B	0.00 %
	76.51 MB	0.94 %	49.23 KB	0.00 %
	74.31 MB	0.91 %	77.46 KB	0.00 %
	66.59 MB	0.81 %	94.19 KB	0.00 %
	57.09 MB	0.70 %	92.50 KB	0.00 %
	50.14 MB	0.61 %	30.98 KB	0.00 %
	42.37 MB	0.52 %	20.27 KB	0.00 %
	111.62 MB	1.36 %	296 B	0.00 %
	280.77 MB	3.43 %	280.77 MB	3.43 %
	63.40 MB	0.78 %	63.40 MB	0.78 %
	47.87 MB	0.59 %	47.87 MB	0.59 %
	1592.07 MB	19.47 %	1592.07 MB	19.47 %

2014.12-SP1	Final (fsdb)	Ver1 (fsdb)	Ver3 (fsdb)	Progress
Setup	Regression	Regression+ Limit to DUT	Regression+ Limit to DUT, Reg	
Runtime (secs)	1005	1115	983	3%
Mem **Inst View** (GB)	2.8	2.3	1.6	43%
Runtime Mem (GB)	9.6	9.2	8.1	16%
Dump Size (MB)	120	65	15	87%

Recommendations: Dump Enabled

- Dump based on debug requirements
  - Can we limit the number of levels dumped ?
    - Here is an example command to limit the dump to DUT:

      ```
      fsdbDumpvars 0 "testbench.top"
      ```
  - Can we limit the dump to IOs, registers, etc. ?
    - Here is an example command to limit the dump only registers

      ```
      fsdbDumpvars 0 "testbench.top“  “+Only_Reg”
      ```

- Verdi has several commands to limit the dump or suppress specific instances or signals

Future Work and Enhancements

Enhancements

- VCS enhancements requested
  - Limit the memory overhead with Simprofiler
  - Categorize the memory overhead caused by transaction coverage under coverage instead of showing it under package

Future Work

- The current analysis and optimization was done for simulation runs in regression. Similar analysis needs to be performed for coverage related simulations as well
- Evaluate +rad feature for further performance optimization
- Try new VCS feature, redirect simprofile database to a specific location
  - Analyze performance for different RTL versions
    - `%simv -simprofile mem -simprofile_dir_path /central_loc/rel_xyz`

Conclusions

Conclusions (1)

- Improved the regression runtime memory by 33% and runtime by 28%
- Improved dump runtime memory by 27% and runtime by 31%

Phase	Option Name	Action
Compile Time	`-debug_access+r`	Use instead of `-debug`
	`-P learn_pli.tab`	Add the generated `learn_pli.tab`
	`-covg_disable_cg`	Add to disable cover group collection
	`NOVAS tab file`	Remove this option (Redundant with `-debug_access`)
	`+vpi+1`	Use instead of `+vpi`
	`+vcsd, +memcbk`	Remove these options
Run Time	`setenv FSDB_DUMPER_VDI 1`	Add this env variable (Default in 2015.09)
	`+fsdb+skip_cell_instance=3`	Add to disable cell dumping
	`+fsdb+packedmda+struct`	Add to dump packed mda and struct data types

Conclusions (2)

- Recommendations
  - Run the profiler periodically as it is a good way to keep an eye on the memory footprint
  - Review compile and runtime options to make sure they are up to date in between the projects
  - Work with Synopsys application engineers to learn about the latest VCS features in new versions, especially performance related options
  - Review the debug requirements and dump only required information by leveraging FSDB dump commands

Thank You