Presentation to LGE, South Korea

SystemC Verification Strategy

Test and Verification Solutions

Delivering Tailored Solutions for
Hardware Verification and Software Testing

Agenda

- **Verifying SystemC hardware designs**
- **What is an “Advanced Verification Strategy”?**
- **Advanced test benches**
  - Structure
  - Features

Objectives for SystemC Models

- **Architectural Modelling**
  - Investigate potential architectures
  - For example, performance modelling
- **Software Development**
  - Allow early development of software
    - In parallel to hardware development
    - Need to model to software visible state
- **Hardware Design Language**
  - Write the design in SystemC
  - EDA tool flow from SystemC to Gate netlist

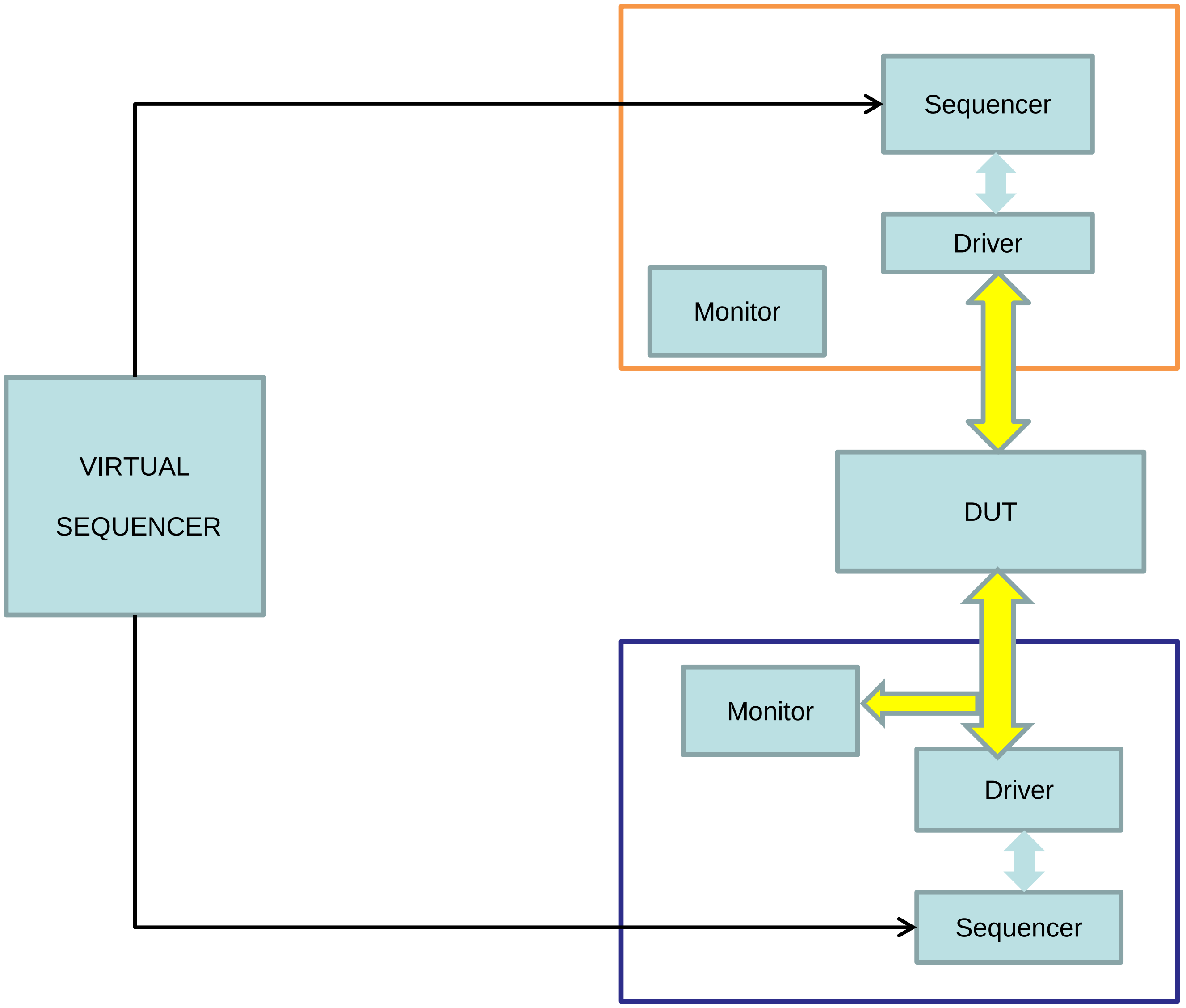
Verification Strategies for SystemC

- **Architectural Modelling**
  - Need to verify accuracy of model
- **Software Development**
  - Need to verify accuracy of model
- **Hardware Design Language**
  - Need a full verification strategy in place
  - This is the topic of this talk

An Advanced Verification Strategy

DUT Specification	
	Requirements Understanding
	Feature Extraction
	Feature List Delivery
	Test Bench Architecture
	Driver, Basic Sequencer & Test
	Monitor & Basic Protocol Checker
	Scoreboard (if required)
	Protocol Functional Coverage Model
	Full Sequences and Tests
	Complete Protocol Checkers
Signoff	

Structure of an Advanced Testbench



Some Features of an Advanced Testbench

- **Base class for verification components such as**
  - driver, sequence, sequencer, monitor, scoreboard, environment, test case
- **Functional coverage library**
- **Factory class for registering component**
- **Configuration class for configuring Test bench**
- **Verbosity Control of messages**
- **Default sequence support**
- **On time Randomization**
- **Raise Objection and Drop Objection**
- **Timeout mechanism**
  - From test case, we can override timeout value.

TVS SystemC Verification Environment

- **Development Environment**
  - SystemC and C++
- **CRAVE library**
  - Used for the randomization
- **TVM Function Coverage library**
  - Generates functional coverage in xml format
  - Can have multiple cover groups in single instance
  - Supports Conditional Coverage
  - Have support for Exclude or Include cross bins by specific cover bin scope specific or bin specific
  - Supports Transition Bins (one series of transition) and Auto Bins
- **TVM library**
  - The TVM library is the methodology is same as that of UVM Library with less features
- **GTKWAVE**
  - Waveform viewing.
- **LCOV & GCOV**
  - Code coverage (generating the output in html format)
- **asureSign**
  - TVS verification management tool

CRAVE Library

- **An advanced constrained random verification (CRV) environment for SystemC**
- **Overcomes the following deficiencies of CRV via SystemC Verification Library (SCV):**
  - Poor dynamic constraint support, hence no control of constraint effects at run-time
  - No constraint specification for dynamic data-structures
  - Low usability when specifying constraints for composed data structures
  - Poor information in case of over-constraining
  - Limits in complexity of constraints, since constraint-solving is based on Binary Decision Diagrams (BDDs) only

How CRAVE addresses SCV limitations

- New constraint specification API
- Dynamic constraints and data structures
- Improved usability
- Parallel constraint-solving

Sample CRAVE Constraints

A basic Constrained Random packet

```
struct packet : public rand_obj
{
  randv< unsigned int > src_addr;
  randv< sc_uint<16> > dest_addr;

packet() : src_addr(this), dest_addr(this) {

constraint(src_addr() <= 0xFFFF);

constraint("diff", src_addr() != dest_addr());

soft_constraint(dest_addr() % 4 == 0);
 }
 };
```

Sample CRAVE Constraints

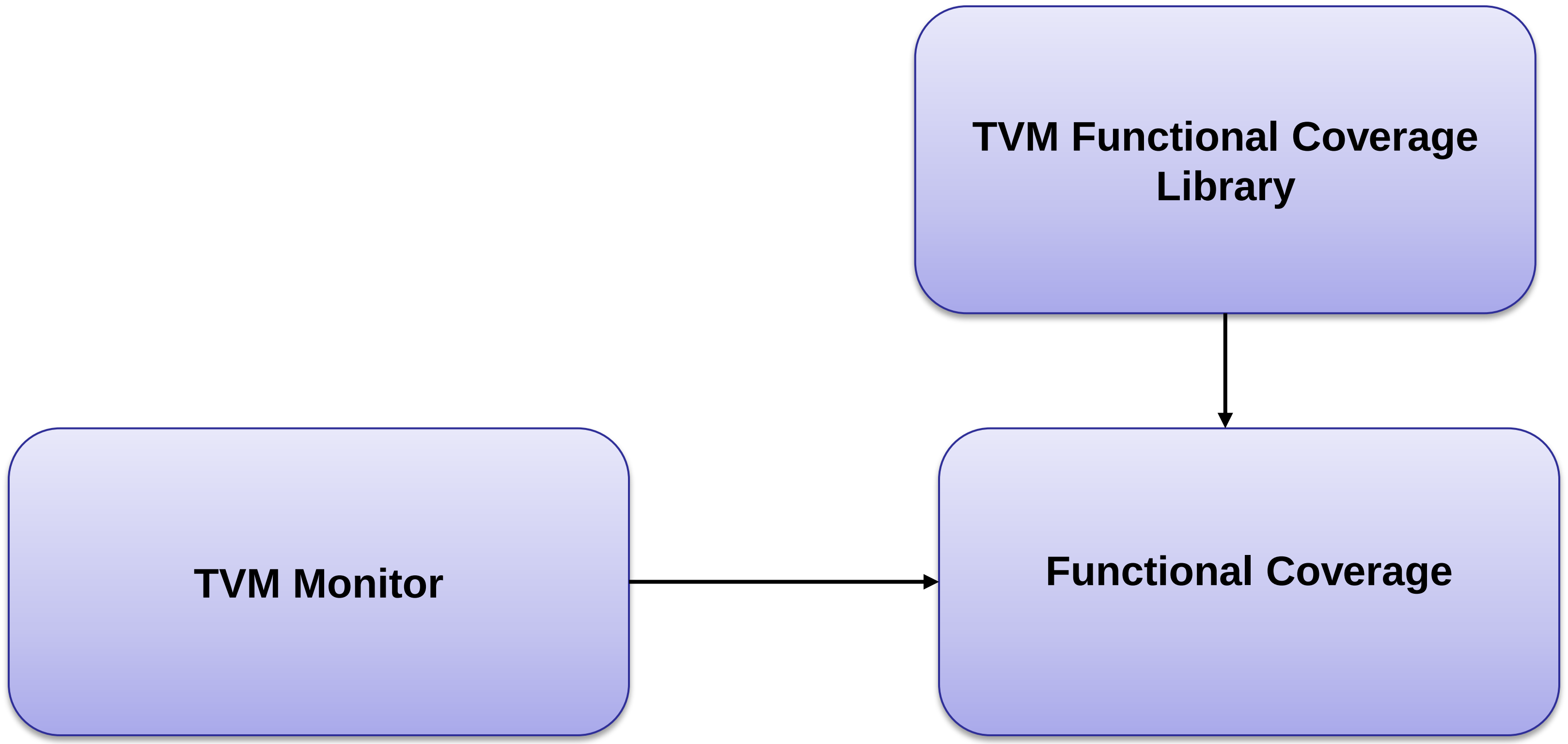
An inherited packet

```
struct packet1 : public packet {
randv< char > data;
packet1() : data(this) {
constraint('a' <= data() && data() <= 'z');
constraint(dest_addr() % 2 == 1);
}
 };
```

TVM Function Coverage library

- **To support hardware verification by C++ and SystemC**
- **Features:**
  - Generates functional coverage in xml format
  - Can have multiple cover groups in single instance
  - Supports Conditional Coverage
  - Have support for Exclude or Include cross bins by specific cover bin scope specific or bin specific
  - Supports Transition Bins (one series of transition), Auto Bins
  - Can cross up-to 4 cover points

TVM Function Coverage library



Sample Code - Coverpoints

```
COVER_POINT_ST(ADDR)
   BINS(aa,"[0:2]")
   BINS(xx,"[3:5]")
   IGNORE_BINS(bb,"(6,7)")
   ILLEGAL_BINS(cc,"[7:8]")
   TRANSITION(tx1,1,2)
 COVER_POINT_END
```

```
COVER_POINT_ST(DATA)
   BINS(dd,"[0:3]")
   IGNORE_BINS(ee,"[5:6]")
   ILLEGAL_BINS(ff,"a1")
   TRANSITION(tx2,3,1)
 COVER_POINT_END
```

Sample Code - Coverpoints

```
COVER_POINT_ST(NCP)
  AUTO_BINS(gg)
 COVER_POINT_END
```

```
CROSS_COVERPOINT(ADDR_DATA)                                    // create class for this
   INVALID_CROSS_BINS(4,"ADDR","(0,2)","DATA","(0,1,2)") // values passed here
   must already be in the valid bins
 CROSS_COVERPOINT_END
```

```
CROSS_COVERPOINT(ADDR_DATA_NCP)                                // create class for this
     VALID_CROSS(3,"ADDR.aa","DATA.dd","NCP.gg")
 CROSS_COVERPOINT_END
```

TVM library

- **TVM provides the best framework to achieve coverage-driven verification (CDV)**
- **TVM testbench components:**
  - Transaction
  - Driver (BFM)
  - Sequencer
  - Monitor
  - Agent
  - Environment (env)

Developing reusable verification components for TVM library

- **Modelling Data Items for Generation**
- **Transaction-Level Components**
- **Creating the Driver**
- **Creating the Sequencer**
- **Creating the Monitor**
- **Instantiating Components**
- **Creating the Agent**
- **Creating the Environment**
- **Enabling Scenario Creation**
- **Managing End of Test**
- **Implementing Checks and Coverage**

GTKWAVE

- Analysis tool used to perform debugging on Verilog or VHDL
- Not intended to be run interactively with simulation
- Relies on a post-mortem approach through the use of dump files
- Supports various dump file formats like VCD (Value Change Dump)

GCOV

- Code coverage analysis and statement-by-statement profiling tool
- Comes as a standard utility with the GNU Compiler Collection (GCC) suite
- Produces a test coverage analysis of a specially instrumented program
- Takes source files as command-line arguments and produces an annotated source listing
- Each line of source code is prefixed with the number of times it has been executed; lines that have not been executed are prefixed with "#####"

LCOV

- Graphical front-end for gcov
- Collects gcov data for multiple source files and creates HTML pages containing the source code annotated with coverage information
- Reads all the .gcno files and creates the .info file for generating HTML report

Usage:

```
lcov -c --no-external -d .  --output-file coverage.info
```

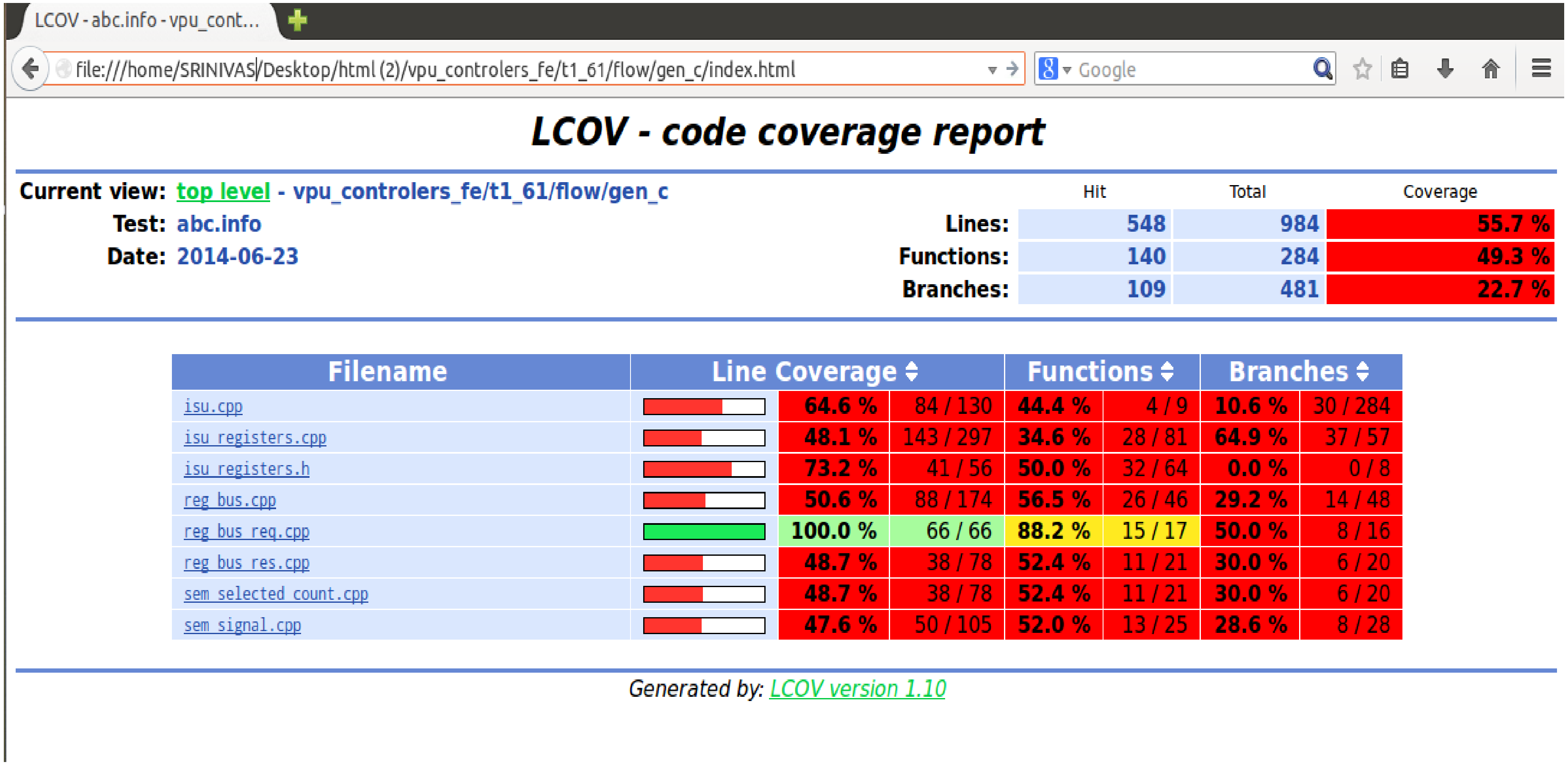
Create the coverage.info it contains all the files coverage information.

```
 lcov -r coverage.info --rc lcov_branch_coverage=1 /tools/crave-bundle/\*
/usr/\* tvm/\*  --output-file abc.info
```

Remove unwanted coverage information of files such as library files and creates the abc.info file

LCOV

- **Sample LCOV report**


LCOV - abc.info - vpu_cont...
file:///home/SRINIVAS/Desktop/html (2)/vpu_controlers_fe/t1_61/flow/gen_c/index.html
Google

LCOV - code coverage report

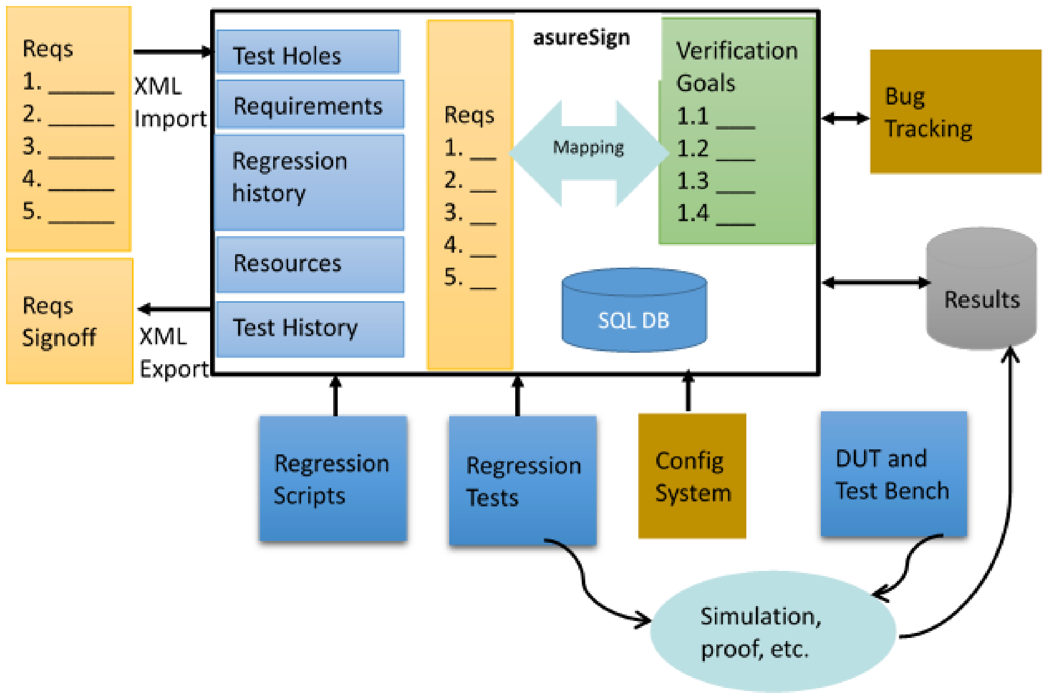
Current view: top level - vpu_controlers_fe/t1_61/flow/gen_c
Test: abc.info
Date: 2014-06-23

	Hit	Total	Coverage
Lines:	548	984	55.7 %
Functions:	140	284	49.3 %
Branches:	109	481	22.7 %

Filename	Line Coverage			Functions		Branches	
isu.cpp		64.6 %	84 / 130	44.4 %	4 / 9	10.6 %	30 / 284
isu_registers.cpp		48.1 %	143 / 297	34.6 %	28 / 81	64.9 %	37 / 57
isu_registers.h		73.2 %	41 / 56	50.0 %	32 / 64	0.0 %	0 / 8
reg_bus.cpp		50.6 %	88 / 174	56.5 %	26 / 46	29.2 %	14 / 48
reg_bus_req.cpp		100.0 %	66 / 66	88.2 %	15 / 17	50.0 %	8 / 16
reg_bus_res.cpp		48.7 %	38 / 78	52.4 %	11 / 21	30.0 %	6 / 20
sem_selected_count.cpp		48.7 %	38 / 78	52.4 %	11 / 21	30.0 %	6 / 20
sem_signal.cpp		47.6 %	50 / 105	52.0 %	13 / 25	28.6 %	8 / 28

Generated by: LCOV version 1.10

asureSign

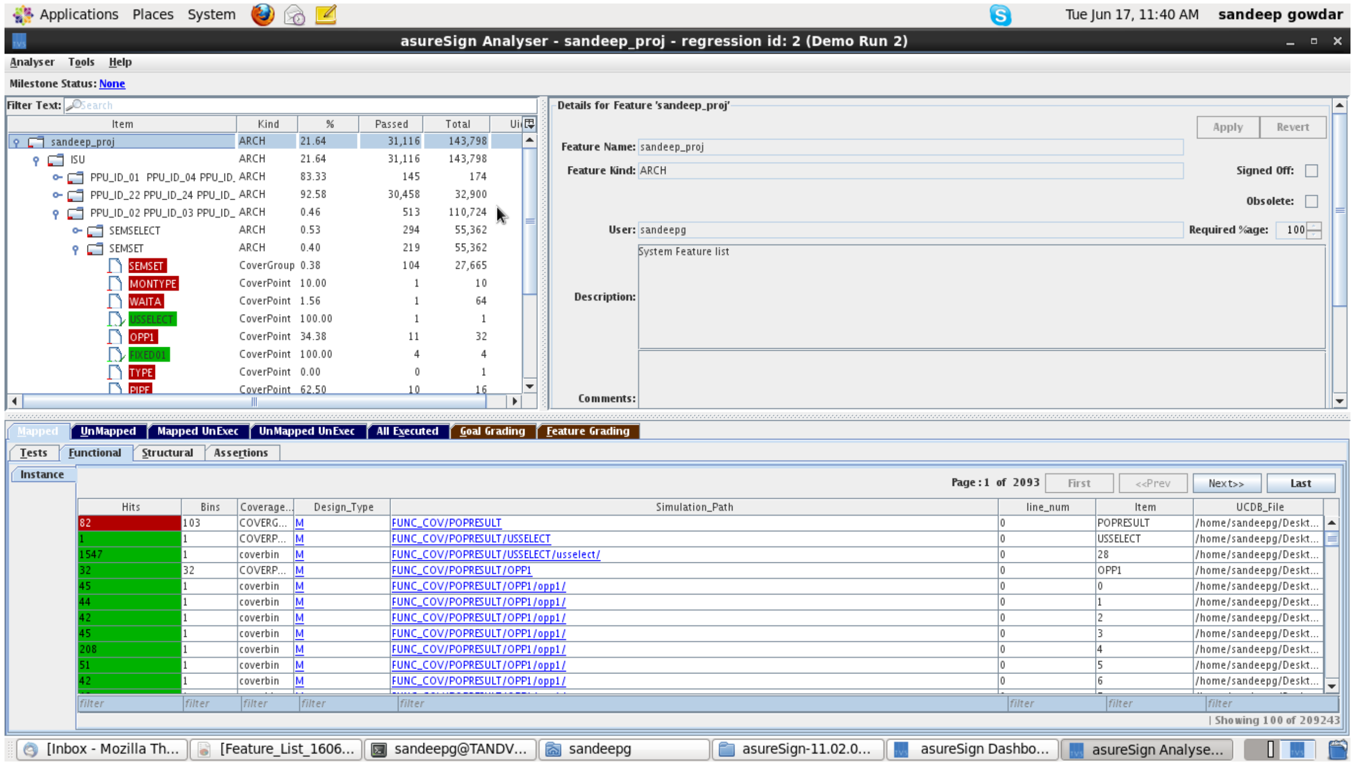


Simulator independent

- Can independently work with your test environment with no dependency on any EDA tool chain
- Combine data from multiple tools and disciplines

asureSign

- TVM_FUNCTIONAL_COVERAGE library dumps the functional coverage data into an XML file
- Using asuresign we can read these xml files and verify the functional coverage



asureSign Analyser - sandeep_proj - regression id: 2 (Demo Run 2)

Milestone Status: None

Filter Text: Search

Item	Kind	%	Passed	Total
sandeep_proj	ARCH	21.64	31,116	143,798
ISU	ARCH	21.64	31,116	143,798
PPU_ID_01 PPU_ID_04 PPU_ID_	ARCH	83.33	145	174
PPU_ID_22 PPU_ID_24 PPU_ID_	ARCH	92.58	30,458	32,900
PPU_ID_02 PPU_ID_03 PPU_ID_	ARCH	0.46	513	110,724
SEMSELECT	ARCH	0.53	294	55,362
SEMSET	ARCH	0.40	219	55,362
SEMSET	CoverGroup	0.38	104	27,665
MONTYPE	CoverPoint	10.00	1	10
WAITA	CoverPoint	1.56	1	64
USSELECT	CoverPoint	100.00	1	1
OPP1	CoverPoint	34.38	11	32
FIXED01	CoverPoint	100.00	4	4
TYPE	CoverPoint	0.00	0	1
PIPE	CoverPoint	62.50	10	16

Details for Feature 'sandeep_proj'

Feature Name: sandeep_proj

Feature Kind: ARCH

Signed Off: ☐

Obsolete: ☐

User: sandeepg

Required %age: 100

Description: System Feature list

Comments:

Page : 1 of 2093

Hits	Bins	Coverage...	Design_Type	Simulation_Path	line_num	Item	UCDB_File
82	103	COVERG...	M	FUNC_COV/POPRESULT	0	POPRESULT	/home/sandeepg/Deskt...
1	1	COVERP...	M	FUNC_COV/POPRESULT/USSELECT	0	USSELECT	/home/sandeepg/Deskt...
1547	1	coverbin	M	FUNC_COV/POPRESULT/USSELECT/usselect/	0	28	/home/sandeepg/Deskt...
32	32	COVERP...	M	FUNC_COV/POPRESULT/OPP1	0	OPP1	/home/sandeepg/Deskt...
45	1	coverbin	M	FUNC_COV/POPRESULT/OPP1/opp1/	0	0	/home/sandeepg/Deskt...
44	1	coverbin	M	FUNC_COV/POPRESULT/OPP1/opp1/	0	1	/home/sandeepg/Deskt...
42	1	coverbin	M	FUNC_COV/POPRESULT/OPP1/opp1/	0	2	/home/sandeepg/Deskt...
45	1	coverbin	M	FUNC_COV/POPRESULT/OPP1/opp1/	0	3	/home/sandeepg/Deskt...
208	1	coverbin	M	FUNC_COV/POPRESULT/OPP1/opp1/	0	4	/home/sandeepg/Deskt...
51	1	coverbin	M	FUNC_COV/POPRESULT/OPP1/opp1/	0	5	/home/sandeepg/Deskt...
42	1	coverbin	M	FUNC_COV/POPRESULT/OPP1/opp1/	0	6	/home/sandeepg/Deskt...

| Showing 100 of 209243

Conclusion

- **TVS has developed an advanced verification flow for SystemC models**
  - It is license free
  - Provides all of the features expected in advanced test benches
  - It fits well with the TVS Verification Strategy
- **It is being used on first client project**