



*Experience of
using SystemC
for design and
verification*

Andy Lunness

Developed in collaboration with TVS



Content

- ✓ Background and challenges
- ✓ The Verification Solution
- ✓ Advantages
- ✓ Case Study



BluWireless Objectives

❑ ***Reusability***

- across projects using SystemC designs

❑ ***Cost effectiveness***

- reducing dependency on proprietary licences

❑ ***Advanced verification support***

- ability to deploy advanced verification techniques

❑ ***IP quality***

- demonstrable improved IP quality via metrics and bug discovery rates



Challenges in Development

- SystemC does not have class libraries equivalent to the System Verilog UVM class library
- SystemC class libraries such as SCV and CRAVE provide constrained random verification features but are less mature than System Verilog
- Replication of coverage driven verification features supported by UVM is more of a challenge in SystemC
- Avoid proprietary tools from EDA vendors



The Development Approach

- ❑ Developing the C++ equivalent of UVM
 - C++ Class library (Called TVM)
- ❑ Enabling constrained random verification
 - through CRAVE
- ❑ Enabling Coverage
 - through a Functional Coverage Library



C++ Class library

- ❑ Library of base methodology classes and functions in C++ enabling easy conversion from TVM->UVM or UVM->TVM
- ❑ The TVM library mimicked UVM phases such as compile, build, run, check etc.
- ❑ The library includes base classes for Constraint Driven Randomization using CRAVE
- ❑ Functional coverage is enabled through a TVM Functional Coverage Library
- ❑ Code Coverage is enabled through gcov & lcov tools

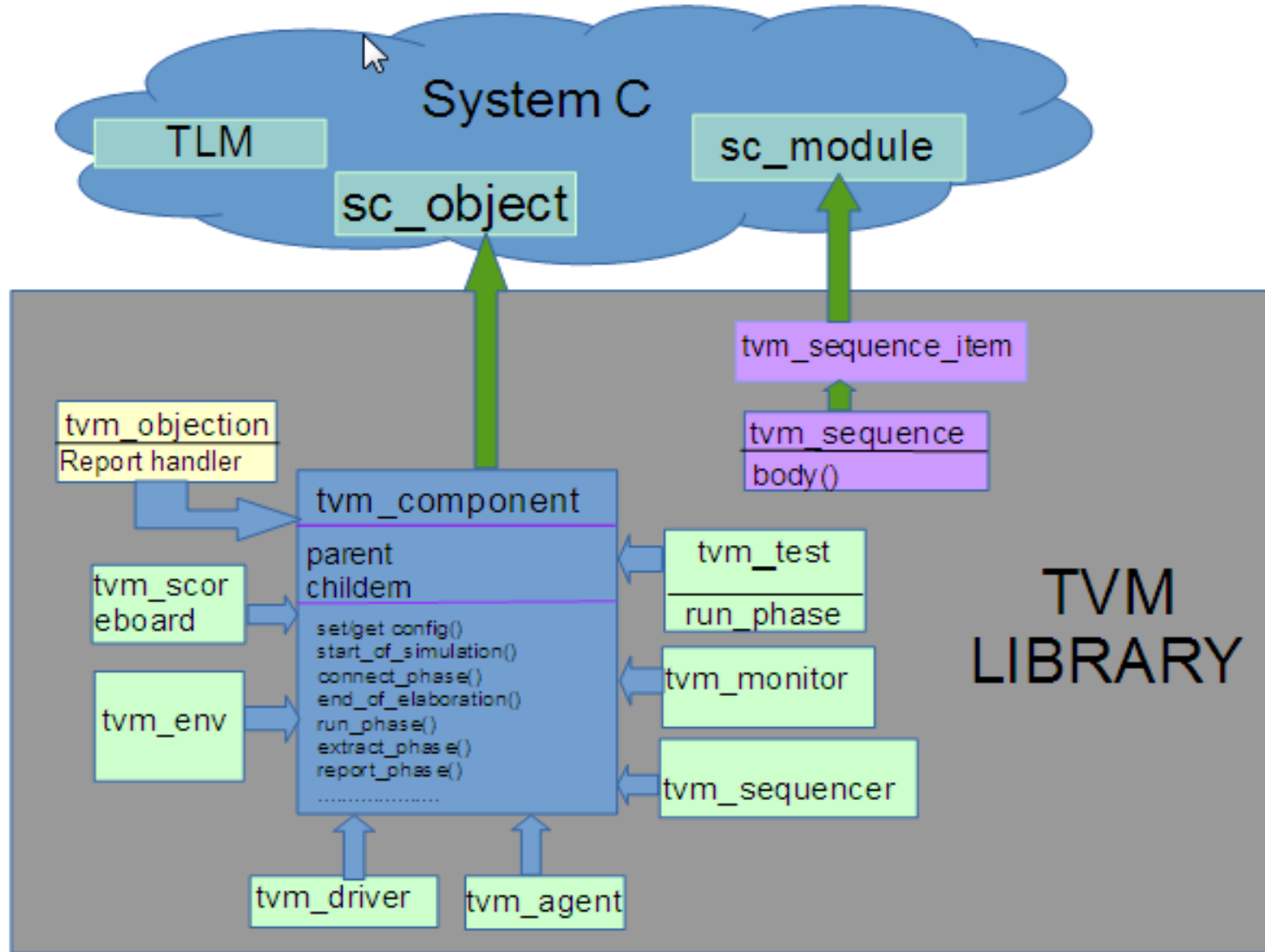


Some example CRAVE constraints

```
randv<int> x,y;
Generator gen;
gen(x() != y());
for (int i =0 ; i < 1000; ++i) {
gen.next();
run_test(x,y);
}
gen( x()*x() == y() );
for (int i =0 ; i < 500; ++i) {
gen.next();
run_test(x,y) ;
}
gen( y()%2 == 0 );
for (int i =0 ; i < 500; ++i) {
gen.next();
run_test(x,y) ;
}
```



TVMM Library Structure



Constrained Random Verification

- ❑ Enabled through use of external randomization library called CRAVE
- ❑ The CRAVE library allows the users to set constraints in a manner similar to System Verilog UVM
- ❑ CRAVE allows inline constraints, which can be formulated and changed incrementally at run-time



Example Functional Coverage



Func cov.h

```
COVER_GROUP_START(CG1)
  int ph_ncp;
  int ph_addr;

  COVER_POINT_ST(ADDR)
    EX_AUTO_BINS(xx)
  COVER_POINT_END

  ADD_COVER_POINT
  COVER_POINT(ADDR,ph_addr,8)
  END_COVER_POINT

  SAMPLE_COVER_POINT
  SAMPLE(ADDR)
  END_SAMPLE_COVER_POINT

COVER_GROUP_END
```

Func cov imp.cpp

```
int main() {
  CG1 obj("CG1");
  obj.set_db_name("HELLO.TVS"
);
  obj.add_cover_point();
  obj.ph_addr = 41;
  obj.sample();
  obj.ph_addr = 81;
  obj.sample();
  obj.display();
  return 0;
}
```



Achieving Coverage

- ❑ TVS developed a Functional Coverage Library (FCL) to enable coverage driven verification
- ❑ Coverage is achieved through TVM functional coverage library and asureSIGN*
 - Can be used either with TVM or individually on C++ based environments
 - Supports report generation
 - Support for different coverage types

*asureSign is TVS' in-house requirements driven management and verification tool



Sample asureSIGN report

asureSign Analyser - BubbleSort - regression id: 1 (v1)

Analyser Tools Help

Milestone Status: **None**

Filter Text:

Item	Kind	%	Passed	Total	Uid
BubbleSort	ARCH	100.00	578	578	1
BUBBLESORT	ARCH	100.00	578	578	2
BubbleSort	CoverGroup	100.00	258	258	3
TypeOfSort	CoverPoint	100.00	2	2	4
OrderAsc	CoverPoint	100.00	1	1	5
OrderDsc	CoverPoint	100.00	1	1	6
IntdataAsc	CoverPoint	100.00	47	47	7
IntdataDsc	CoverPoint	100.00	47	47	8
Chandata	CoverPoint	100.00	52	52	9
UnalignedDataA	CoverPoint	100.00	31	31	10
UnalignedDataD	CoverPoint	100.00	31	31	11
TypeOfSort_Ch	Cross	100.00	104	104	12
TypeOfSort_Or	Cross	100.00	2	2	13
TypeOfSort_Or	Cross	100.00	2	2	14

Details for Verification Goal 'BubbleSort'

Goal Kind: **CoverGroup**

Signed Off: Obsolete: Manual Signed Off:

Design Type: Instance Module None

Goal Name: BubbleSort

Mapping String: *BubbleSort*

Item Name: Line Number:

User: srinivas Required %age:

Specification:

Mapped UnMapped Mapped UnExec UnMapped UnExec All Executed Goal Grading Feature Grading

Tests Functional Structural Assertions

Instance

Page: 1 of 3

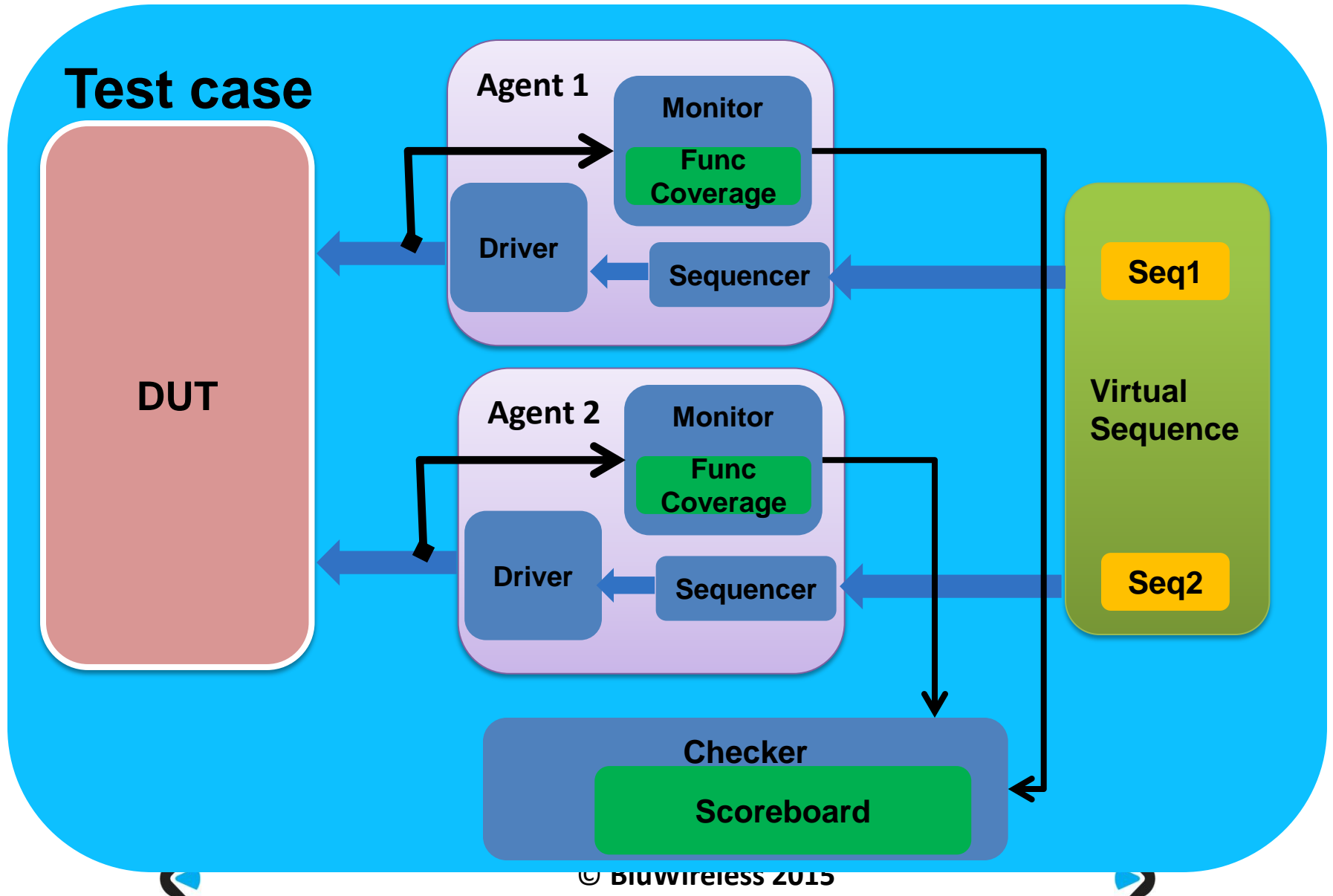
Hits	Bins	Coverage_Kind	Design_Type	Simulation_Path	line_num	Item	UCDB_File
258	258	COVERGROUP		bbs_name/BubbleSort	0	BubbleSort	/Projects/bluwireless/sri...
2	2	COVERPOINT		bbs_name/BubbleSort/Ty...	0	TypeOfSort	/Projects/bluwireless/sri...
4532	1	coverbin		bbs_name/BubbleSort/Ty...	0		/Projects/bluwireless/sri...
4120	1	coverbin		bbs_name/BubbleSort/Ty...	1		/Projects/bluwireless/sri...
1	1	COVERPOINT		bbs_name/BubbleSort/Or...	0	OrderAsc	/Projects/bluwireless/sri...
56	1	coverbin		bbs_name/BubbleSort/Or...	0		/Projects/bluwireless/sri...
1	1	COVERPOINT		bbs_name/BubbleSort/Or...	0	OrderDsc	/Projects/bluwireless/sri...
84	1	coverbin		bbs_name/BubbleSort/Or...	0		/Projects/bluwireless/sri...
16	16	COVERPOINT		bbs_name/BubbleSort/Int...	0	IntdataAsc	/Projects/bluwireless/sri...
11	1	coverbin		bbs_name/BubbleSort/Int...	0	0_1	/Projects/bluwireless/sri...

filter filter filter filter filter filter filter filter

Showing 100 of 270



Sample Implementation



Advantages of TVS approach

- TVM supports many UVM features
 - Constructs for agents, monitors, drivers, sequencers and scoreboard
 - Factory registration and overriding
 - Randomization control using constraints
 - Ability to raise and drop objections
 - config_db mechanism
 - Dynamic casting using DCAST similar to System Verilog's \$cast
 - Support for time out mechanism
 - Reporting severity similar to UVM
 - Uses TLM ports (interface method class)



Advantages of TVM approach

- TVM supports System Verilog features
 - Verbosity control of debug messages
 - Fine process control (like fork-join, fork-join none, fork-join_any , tracking process using process handle)
 - Interface class



Conclusion

- ❑ This project was executed over 8 months by a 3 member team.
- ❑ The following blocks were verified:
 - ISU
 - DFE_ISU
 - 4 V.Ports (MAC, DFE, DFE_MAC, DFE_DFE)
- ❑ At block level:
 - Functional coverage of 100% was achieved for all blocks.
 - In Code Coverage, line coverage of 100% with exclusions was achieved.
- ❑ The TVM library developed by TVS, along with Functional Coverage Library and asureSign is available for verification of C++ and System C based verification projects.

