BLUWIRELESS TECHNOLOGY

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

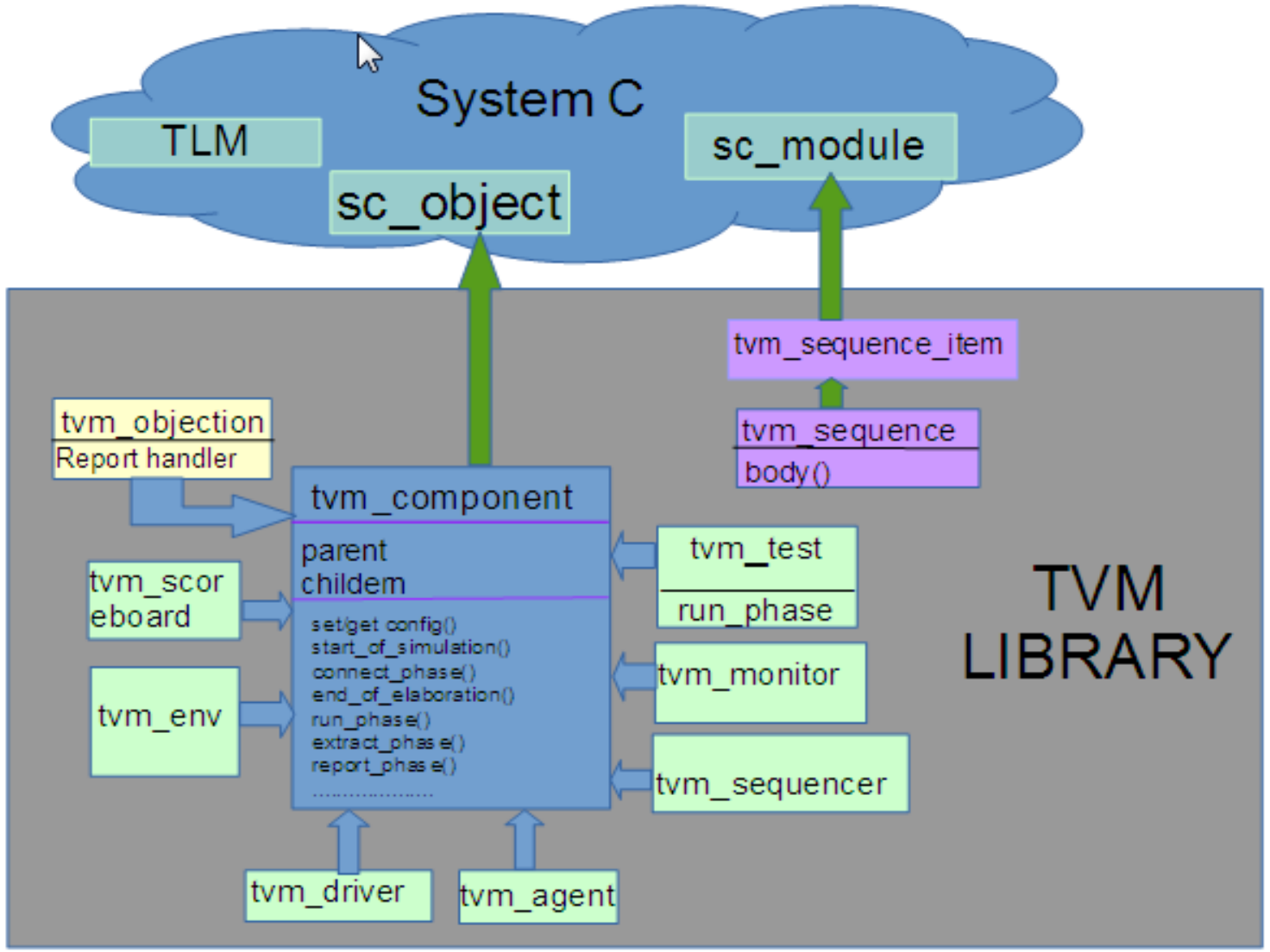
C++ Class library

- 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) ;
}
```

TVM 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

CRAVE is freely available under the MIT license
CRAVE is available via public repositories at GitHub.

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

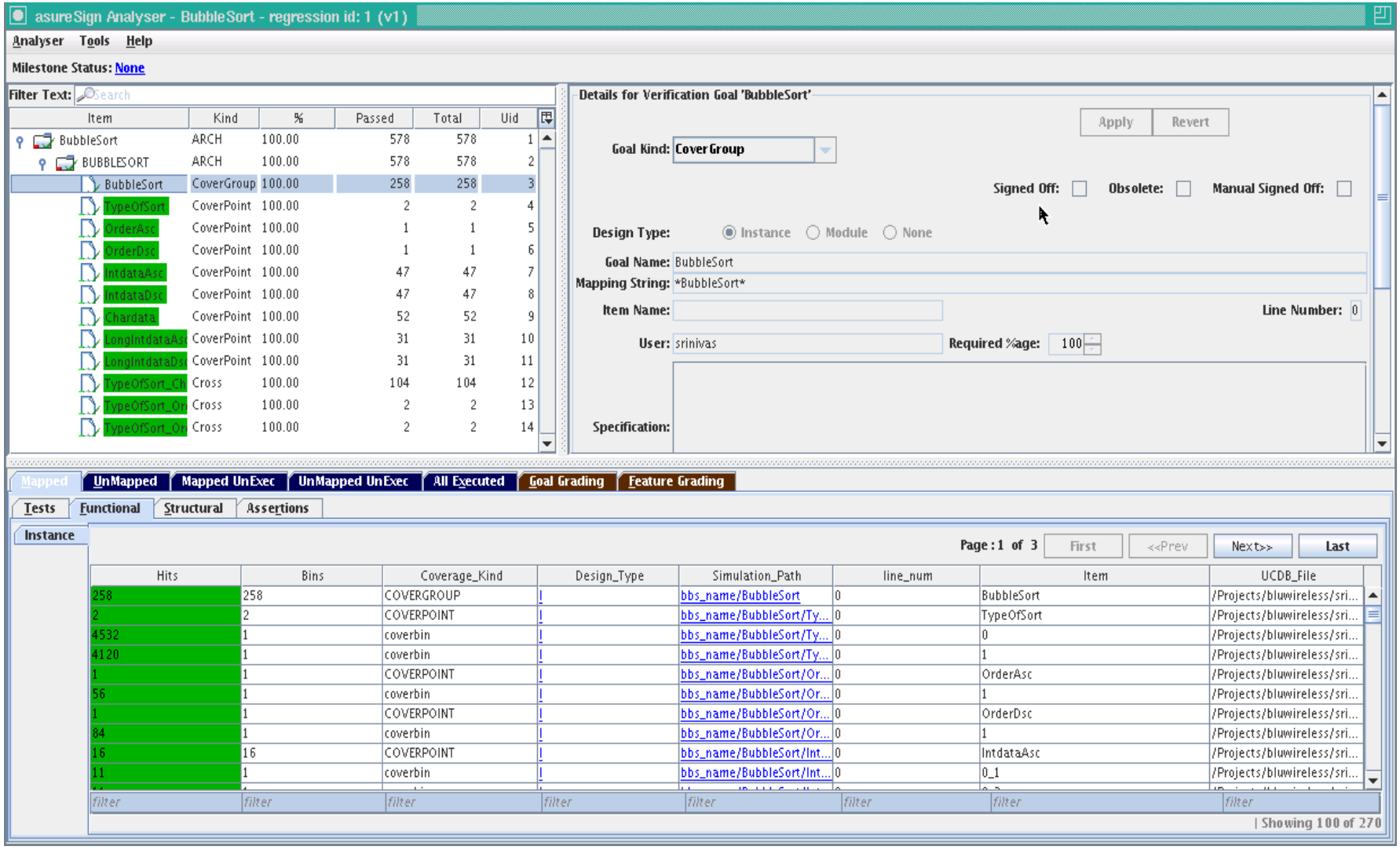
```
int main() {
   CG1 obj("CG1");
   obj.set_db_name("HELLO.TVS"
);
   obj.add_cover_point();
   obj.ph_addr = 41;
   obj.sample();
   obj.ph_addr = 81;
   obj.sample();
   obj.display();
   return 0;
 }
```

Achieving Coverage

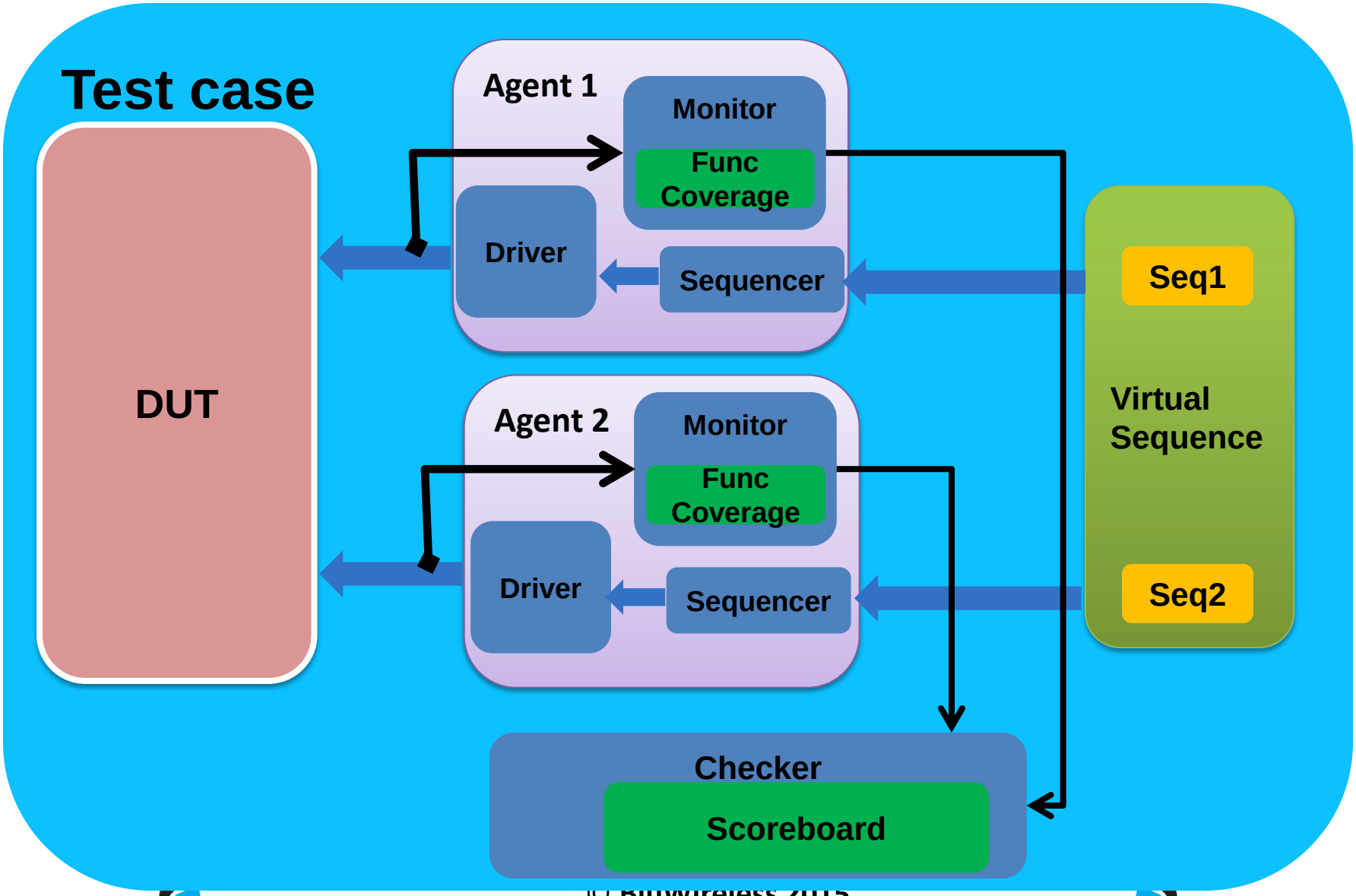
- TVS developed a Functional Coverage Library (FCL) to enable coverage driven verification
- Coverage is achieved through TVM functional coverage library and asureSIGN*
  - Can be used either with TVM or individually on C++ based environments
  - Supports report generation
  - Support for different coverage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

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

Sample asureSIGN report



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.