Enriching UVM in SystemC with AMS extensions for randomization and functional coverage

Thilo Vörtler, Thomas Klotz, Karsten Einwich,Yao Li, Zhi Wang, Marie-Minerve Louërat, Jean-Paul Chaput, François Pêcheux, Ramy Iskander,

Martin Barnasconi

2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Outline

- Motivation
- UVM for SystemC and SystemC-AMS
- Randomization
- Coverage
- Summary

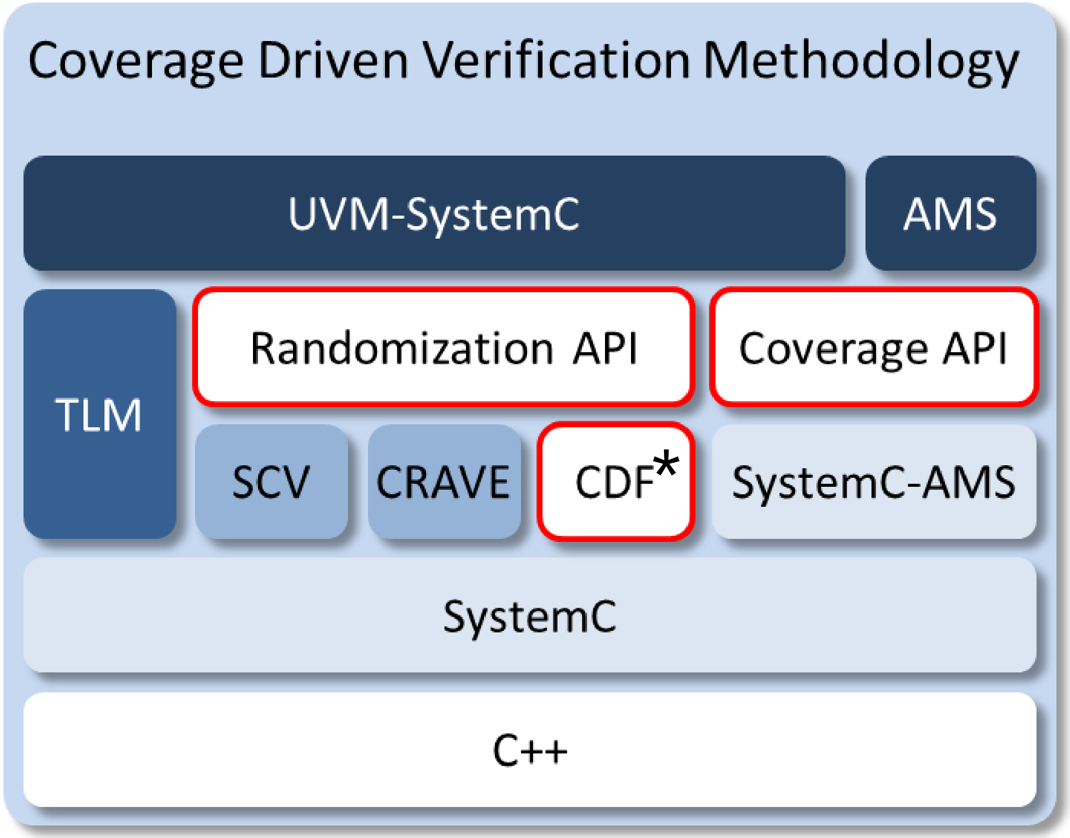
Outline

- Motivation
- UVM for SystemC and SystemC-AMS
- Randomization
- Coverage
- Summary

Motivation (1)

- UVM is currently being available for SystemC-AMS
- The UVM methodology relies heavily on the use of randomization of sequences and functional coverage collection
- Randomization and coverage APIs are not defined in UVM but inherited from System Verilog

Motivation (2)



→ Goal: Definition of randomization and coverage API for UVM in SystemC/AMS

*continuous distribution functions

Outline

- Motivation
- UVM for SystemC and SystemC-AMS
- Randomization
- Coverage
- Summary

UVM for SystemC-AMS

Requirements for randomization and coverage API:

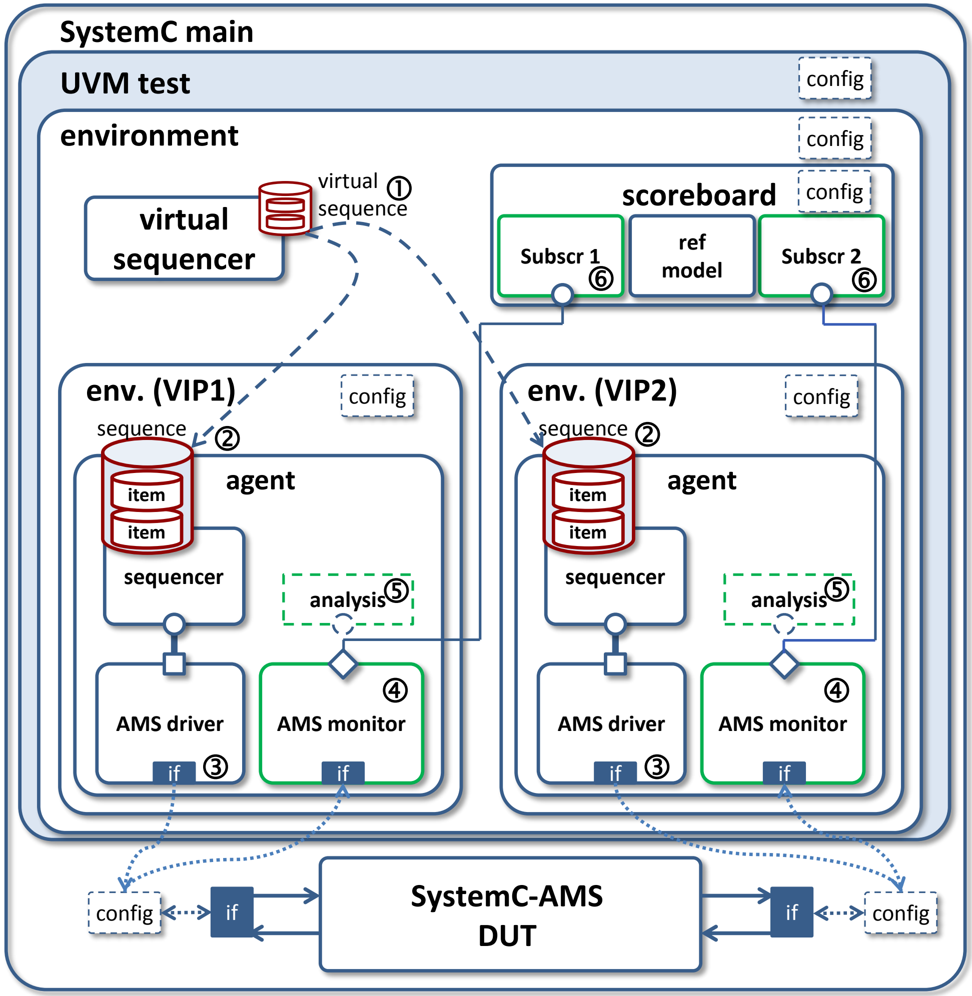
- Current available SystemC libraries don’t offer a simple API that is familiar to UVM users
- Extensions for dealing with real values needed for AMS verification are needed

→Proposal for an SCV extension (scvx) for further standardization within Accellera

UVM for SystemC/AMS

- Sequences that drive stimulus to the DUT are randomized

- Coverage is used in monitors and scoreboards to check the progress of the verification process



Outline

- Motivation
- UVM for SystemC and SystemC-AMS
- Randomization
- Coverage
- Summary

Randomization (1)

Supported constructs (CRAVE 1.0 as backend)

Functionality	SystemVerilog	UVM-SystemC (scvx)
Random variable declaration	rand T	scvx_rand<T>
Enable or disable random variable	rand_mode(…)	rand_mode(…)
Constraint block declaration	constraint	scvx_constraint
Enable or disable constraint	constraint_mode(…)	constraint_mode(…)
Randomization container object	-	scvx_rand_object
Randomize method	randomize()	randomize()
Randomize method with inline constraint	randomize() with …	randomize_with(…)

Randomization (2)

```
class simplesum : public
scvx::scvx_rand_object
{
 public :
 scvx::scvx_rand< int > x, y, z ;
 scvx::scvx_constraint c1, c2, c3 ;
 simplesum( scvx::scvx_name name )
 : x("x"), y("y"), z("z"),
    c1("c1"), c2("c2"), c3("c3")
 {
   c1( z() == x() + y() );
   c2( x() == 5 );
   c3( y() > 0 && y() < 10 );
 }
 void print_result() const
 {
  cout << name() << " : "  << z << " == "
         << x << "  + " << y << endl ;
  }
}; // class simplesum
```

```
int sc_main(int, char*[])
{
 simplesum s("simplesum");
 bool result = s.randomize();

 if (result) s.print_result();
 else cout << "No solution found." << endl;
 s.c2.constraint_mode( false );
 result = s.randomize_with( s.x() == 10 );

 if (result) s.print_result();
 else cout << "No solution found." << endl;
 s.y.rand_mode( false );
 result = s.randomize();

 if (result) s.print_result();
 else cout << "No solution found." << endl;
  return 0;
}
```

Randomization (3)

- Console output:

```
$ ./simplesum.exe
constraint c1 registered.
constraint c2 registered.
constraint c3 registered.
simplesum: 13 == 5 + 8
constraint c2 disabled.
in-line constraint ci_0 applied (disabled after use)
simplesum: 13 == 10 + 3
random variable 'y' made inactive (value remains 3).
simplesum: 33556493 == 33556490 + 3
```

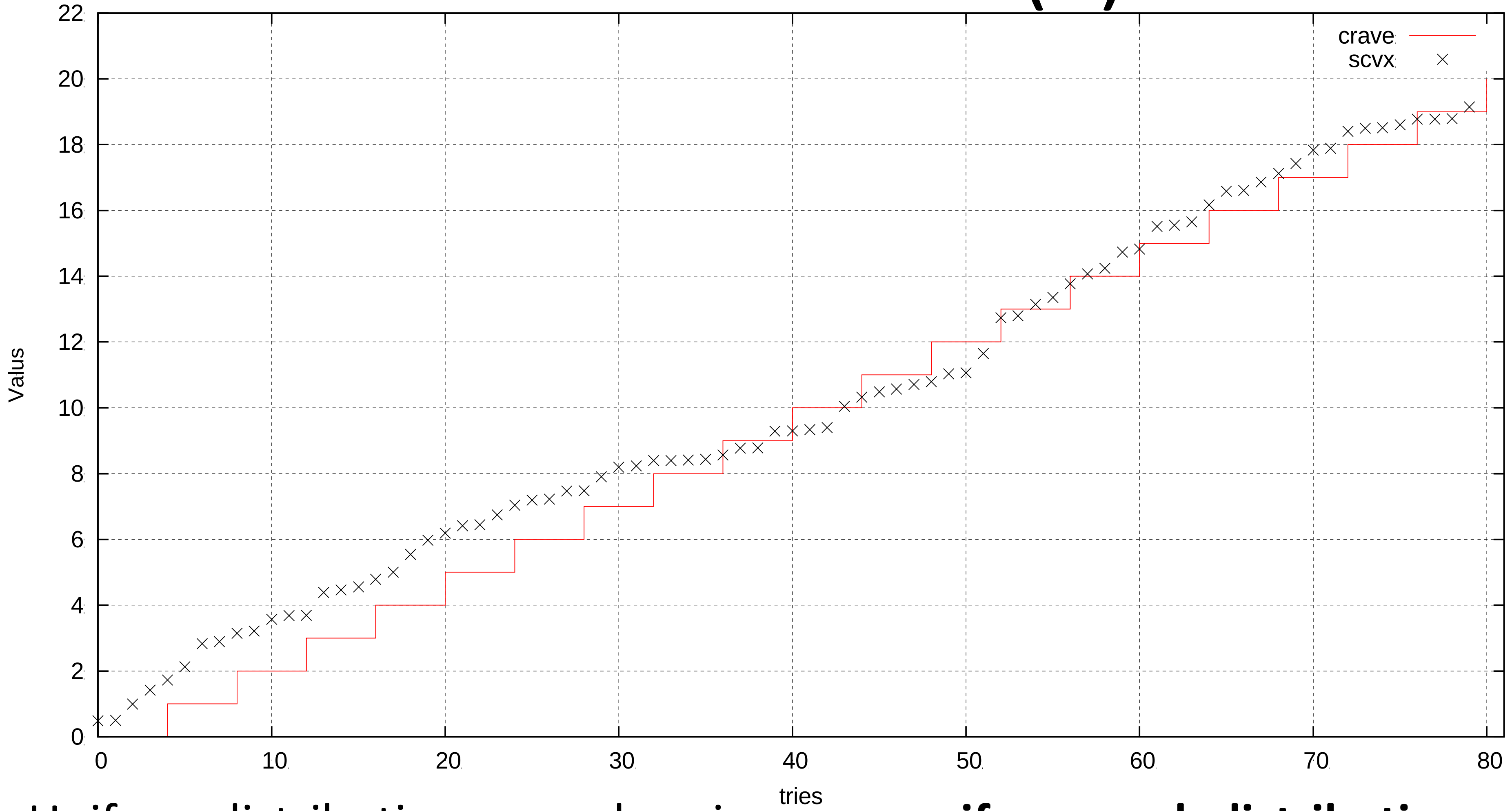
Randomization (4)

- Analog extensions (C++11 based solver)

Distribution function	UVM-SystemC (scvx)
Normal distribution	scvx_normal_distribution
Uniform distribution	scvx_uniform_real_distribution
Bernoulli distribution	scvx_bernoulli_distribution
Piece-wise linear probability distribution function	scvx_piecewise_linear_probability_distribution
Discretized probability distribution function	scvx_discrete_probability_distribution

```
scvx::scvx_rand<real> a
a.set_distribution( dist_type(dist_params) );
```

Randomization (5)



Uniform distribution example using **scvx_uniform_real_distribution**

Outline

- Motivation
- UVM for SystemC and SystemC-AMS
- Randomization
- Coverage
- Summary

Coverage (1)

- Supported features

Functionality	SystemVerilog	UVM-SystemC (scvx)
Coverage model	covergroup	scvx_covergroup
Coverage points	coverpoint	scvx_coverpoint
Coverage state bins	bins	bins()
Illegal bins	illegal_bins	illegal_bins()
Ignore bins	ignore_bins	ignore_bins()
Coverage options	option	option

Coverage (2)

```
class cg: public scvx::scvx_covergroup
{
 public:
 scvx::scvx_coverpoint cp_m;
 scvx::scvx_coverpoint cp_n;
 cg( scvx::scvx_name name, int& m, int& n)
  : cp_m( "cp_m", m ),
    cp_n( "cp_n", n )
  {
    option.auto_bin_max = 16;
    cp_m.bins("bin_a") =
      scvx::list_of(4,  0, 1, 2, 3 );
    cp_m.bins("bin_b", scvx::SINGLE_BIN) =
      scvx::list_of(4,  4, 5, 6, 7 );
    cp_m.ignore_bins("ignore_bins") = 6;
    cp_n.ignore_bins("ignore_bins") = 13;
  }
};
```

```
int sc_main(int, char*[])
{
  int m; // variable to be covered
  int n; // variable to be covered
  int stimuli_m[] =
  { 3, 5, 6, 5, 3, 6, 5, 5, 3, 3 };
  int stimuli_n[] =
  { 13, 1, 6, 3, 16, 12, 8, 3, 13, 3 };
  cg cg_inst("cg_inst", m, n);
  for ( int i = 0; i < 10; i++ )
  {
    m = stimuli_m[i];
    n = stimuli_n[i];
    cg_inst.sample();
  }
  cg_inst.report();
  return 0;
}
```

Coverage (3)

- Coverage statistics are printed at the end of the simulation
- When covering real values ranges are used to identify bins
- Support of Unified Coverage Interoperability Standard (UCIS) under discussion

```
$ ./test.exe

Covergroup: cg_inst
----------------------------------------
VARIABLE      Expected   Covered   Percent
----------------------------------------
cp_m                 7         2     28.57
cp_n                15         5     33.33
----------------------------------------
TOTAL:              22         7     31.82
----------------------------------------
coverpoint: cp_m
---------------------------
Name     Percent     Hitrate
---------------------------
bin_a[0]       0          0
bin_a[1]       0          0
bin_a[2]       0          0
bin_a[3]     100          4
bin_b[4]       0          0
bin_b[5]     100          4
bin_b[7]       0          0
---------------------------
```

Outline

- Motivation
- UVM for SystemC and SystemC-AMS
- Randomization
- Coverage
- Summary

Summary

- Currently UVM for SystemC is discussed as being a standard in SystemC Verification Working Group, however the standard doesn’t describe an coverage and randomization API

- Randomization and coverage API with extensions to real values described. However, constrained randomization for real values is hard!

- API will be submitted to Accellera for standardization

Acknowledgements

The development of the UVM-SystemC methodology and library has been supported by the European Commission as part of the Seventh Framework Programme (FP7) for Research and Technological Development in the project 'VERIFICATION FOR HETEROGENOUS RELIABLE DESIGN AND INTEGRATION' (VERDI). The research leading to these results has received funding from the European Commission under grand agreement No 287562.

Questions