Using Constraints for SystemC AMS Design and Verification

Thilo Vörtler, Karsten Einwich,
COSEDA Technologies GmbH
Muhammad Hassan, Daniel Große,
DFKI GmbH

Agenda

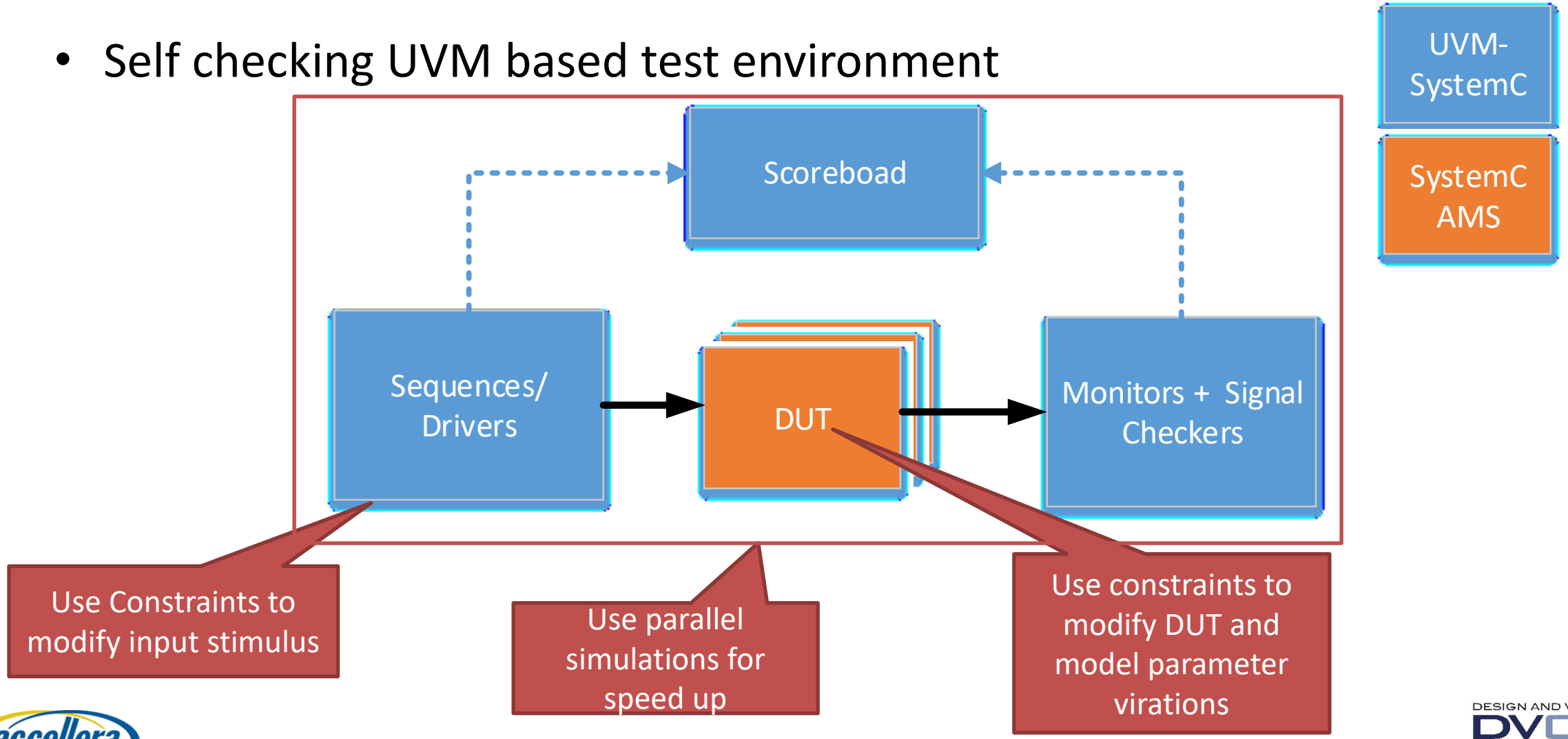
- Introduction
- CRAVE for AMS
- Constraints for SystemC AMS
- Parallel Simulation Framework
- DC-DC Converter Example
- Summary

Introduction

- SystemC AMS allows the creation of high level virtual prototypes of mixed signal systems.
- A major challenge is testing different system parameters for design space exploration and verification.
- Goal:
  - Use constraints to describe variations of a Mixed Signal DUT and the verification input stimulus.
  - Apply and extend existing verification techniques for design space exploration and simulation of parameter variations.

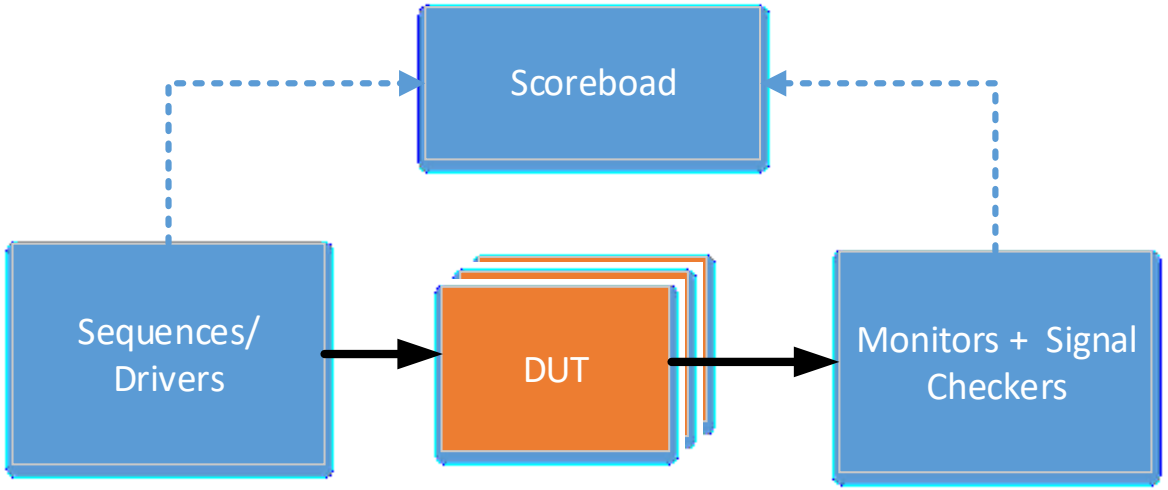
Introduction

- Self checking UVM based test environment



Introduction

- Used tools and Libraries:
  - SystemC AMS to model mixed signal Systems[1,2]
  - UVM SystemC for building a self checking test environment[3]
  - CRAVE Library for constrained randomization of C++/SystemC variables[4,5]
  - CRAVE-UVM adaption layer for creating randomized UVM transactions[4,5]
  - COSIDE® Mixed Signal Checker framework for SystemC AMS

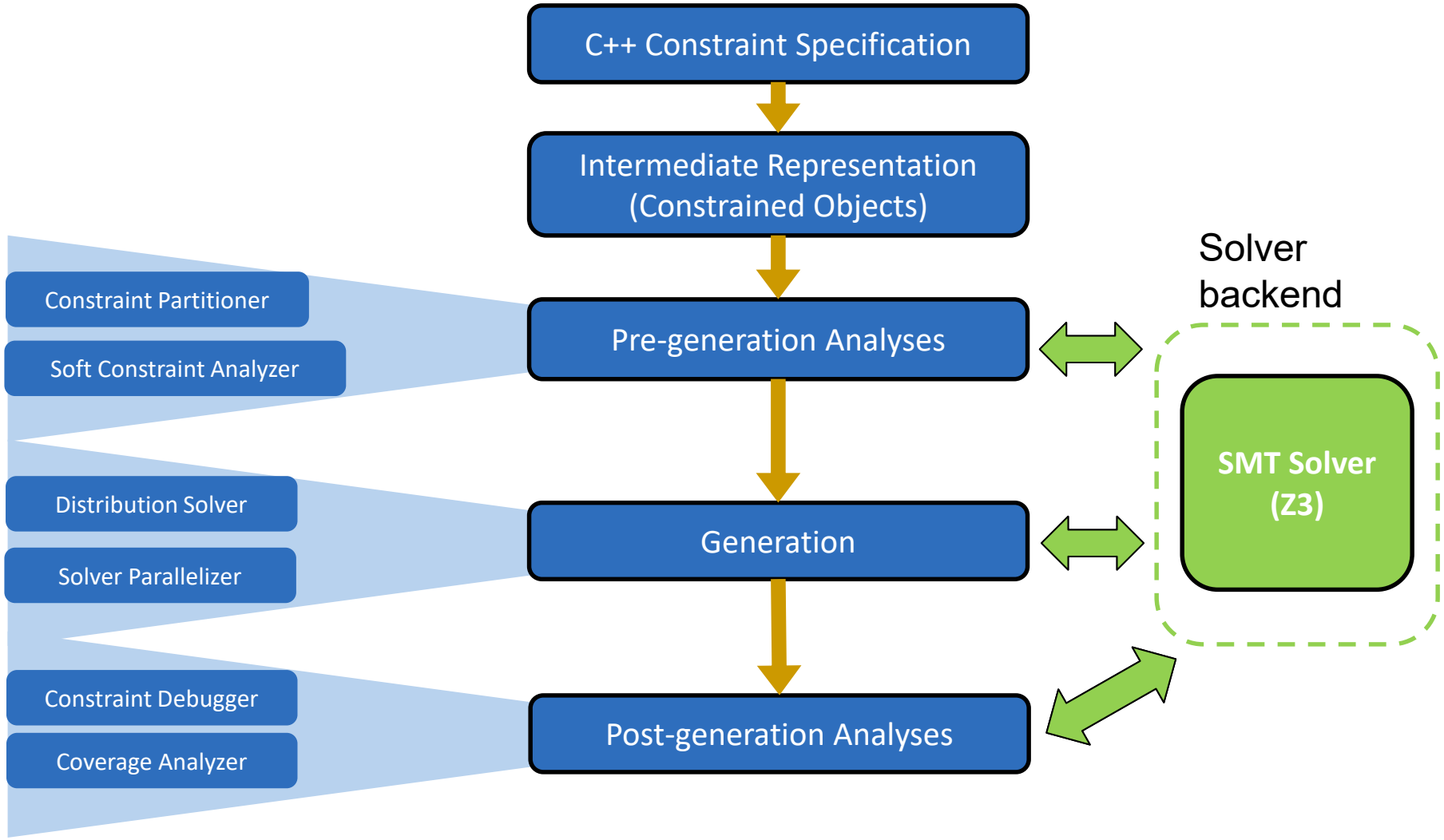


1. http://www.accellera.org/downloads/standards/systemc
2. http://www.coseda-tech.com/systemc-ams-proof-of-concept
3. http://www.accellera.org/activities/working-groups/systemc-verification/
4. http://www.systemc-verification.org/crave
5. https://github.com/agra-uni-bremen/crave
6. https://www.coseda-tech.com/

CRAVE

- CRAVE - **C**onstrained **Ra**ndom **V**erification **E**nvironment
  - Powerful & extensible constrained random stimuli generator
  - Constraining of generated stimuli possible
  - Leverage latest constraint solving technologies
  - C++11 syntax for constraint definition
- CRAVE for Analog Mixed Signal (AMS)
  - Supports digital, and analog functions and data types
  - Scalable, and efficient
  - Hard / soft constraints
  - Constraints partitioning ….

CRAVE for AMS



CRAVE for AMS

- Random variable definition – should support real values (e.g., 1.3, 100.871 …)
  - Data type: **“double” → Was not supported in CRAVE**

```
class item : public crv_sequence_item {
  crv_variable<double> r; //Random Variable - resistor
  crv_variable<double> c; //Random Variable - capacitor
  crv_constraint c1 { r() < 3.3 }; //Constraint resistor < 3.3KOhm
  crv_constraint c2 { c() > 100.5 }; //Constraint capacitor > 100.5 nF
  item(crv_object_name) {}
};
```

- IEEE 754 Floating Point Arithmetic standard compatible

CRAVE for AMS

- Backend: Z3 SMT solver
- Logic: QF_FP
  - All expression constants converted to „declare-const Real“
  - Non-linear artihmetic not supported
- Operators with „double“ datatype

```
crv_variable<double> x0,x1,x2;
crv_constraint plus{x0() == x1() + x2()};
crv_constraint minus{x2() == x0() - x1()};
crv_constraint mul{x2() == x1() * 2};
crv_constraint div{x1() == x2() / 2};
crv_constraint nequal{x1() != x2() || x1() == x2()};
crv_constraint range0{x1() >= 0 && x1() <= 5};
crv_constraint range1{x1() >= 0 && x1() <= x2()};
… …
```

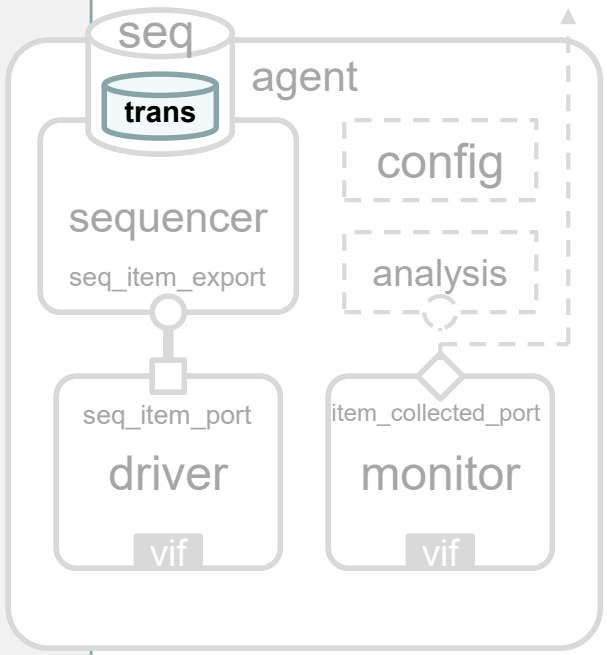
Constraints for SystemC AMS

- Classify constraints depending on usage:
  - to shape the stimulus (in test cases),
  - modify the DUT during construction and elaboration.
- Runtime Constraints:
  - Typically used in UVM runtime sequences
  - Are solved during simulation (run phase) and program *drivers* which shape the stimulus.
  - Examples:
    - setting of register values
    - (digital) input values for interfaces/busses

Constraints for SystemC AMS

```
class vip_trans : public uvm_randomized_sequence_item
{
 public:
  crv_variable<int> addr;
  crv_variable<int> data;
  crv_variable<bus_op_t> op;

  UVM_OBJECT_UTILS(vip_trans);

  vip_trans( const std::string& name = "vip_trans" )
  {
    addr = 0x0;
    data = 0x0;
    op = BUS_READ;
  }

  virtual void do_print( uvm_printer& printer ) const { ... }
  virtual void do_pack( uvm_packer& packer ) const { ... }
  virtual void do_unpack( uvm_packer& packer ) { ... }
  virtual void do_copy( const uvm_object* rhs ) { ... }
  virtual bool do_compare( const uvm_object* rhs ) const { ... }

}; // class vip_trans
```

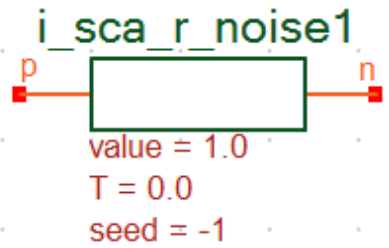
User-defined data items with randomization



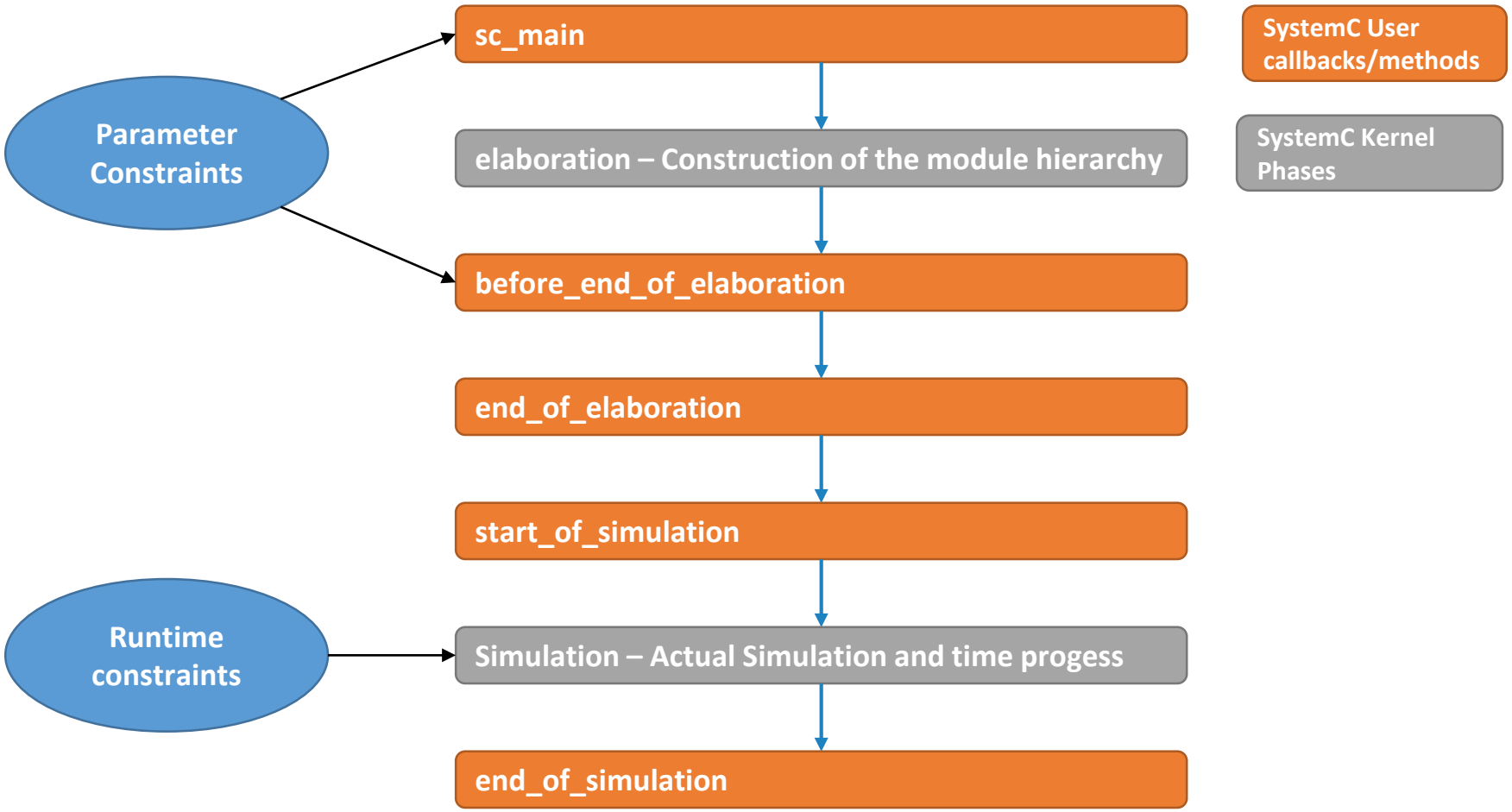
Constraints for SystemC AMS

- Parameter Constraints:
  - Parameterize the DUT and are set statically during elaboration of a SystemC Design. They must not be changed during runtime.
  - Evaluated only once and are created before DUT instantiation
  - Examples:
    - Values of resistors, capacitors, inductors
    - Temperature of a noisy resistor

```
class parameter_constraints : public crv_sequence_item {
  crv_variable<double> R1; //Resistor 1
  crv_variable<double> R2; //Resistor 2
  crv_constraint  R1_val{ 1 < R1 (), R1 () < 3.3  }; // 1KOhm to 3.3Kohm
  crv_constraint R2_val{ 0.5 < R2(), R2() < 4.7 }; // 0.5Kohm to 4.7Kohm
   crv_constraint R_overall{ R2()+R1() < 4.7e3 }; // Restrict sum
  parameter_constraints(crv_object_name) {}
};
```

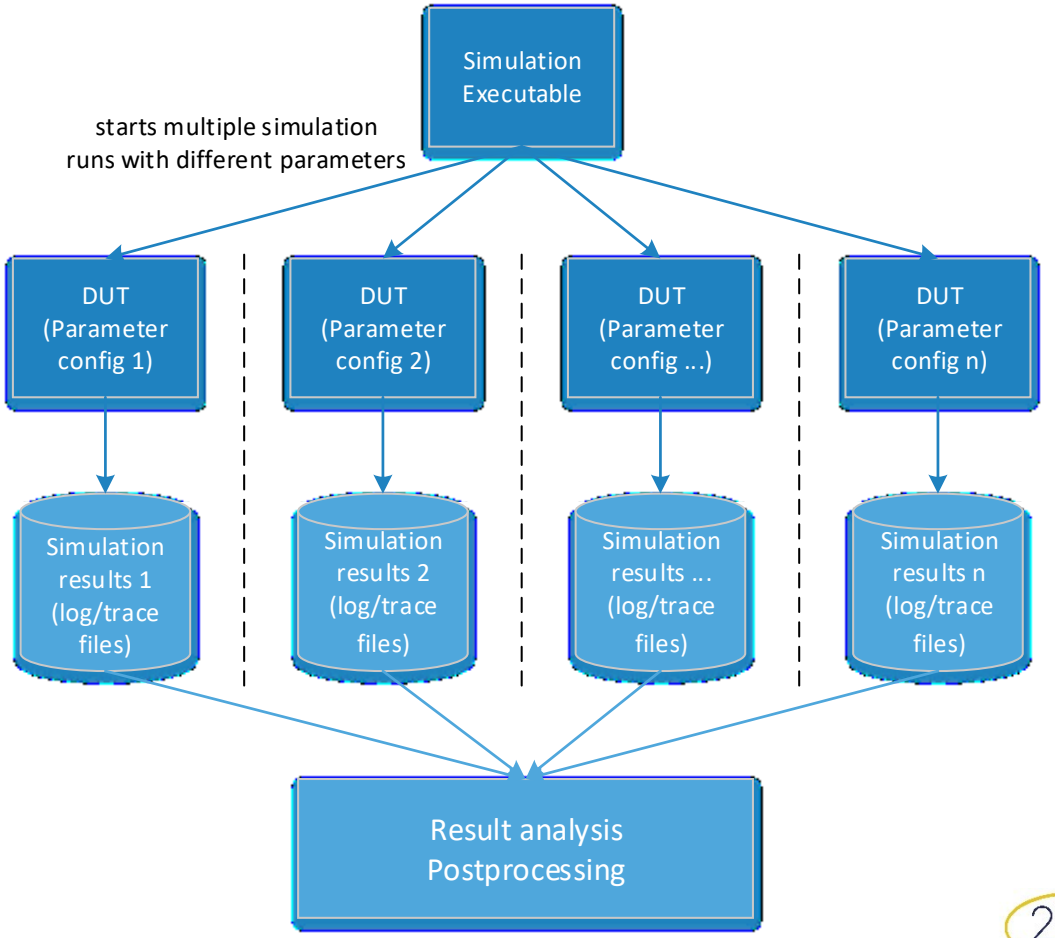


Constraints for SystemC AMS



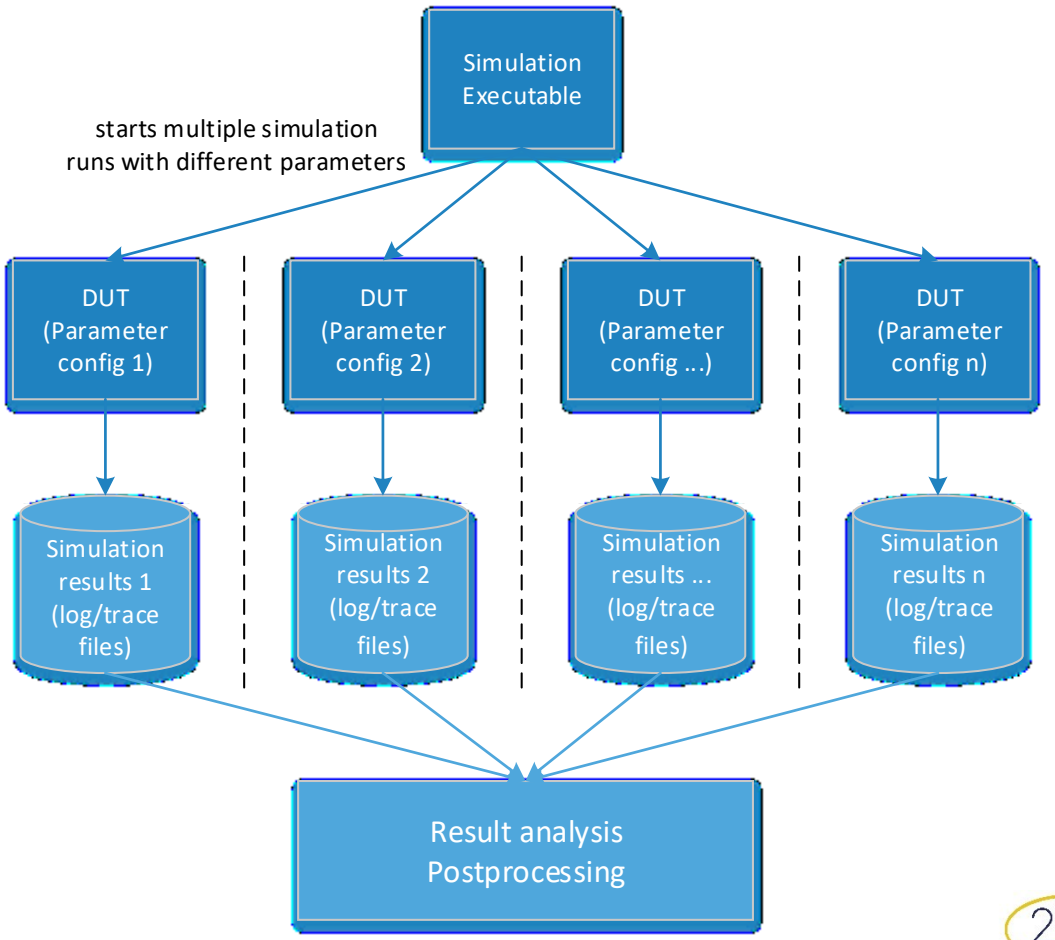
Parallel simulation framework for SystemC

- Spawns several SystemC simulations from the same executable
- Each run gets a unique seed and can generate:
  - Different DUT configurations
  - UVM test sequences
- Number of parallel runs can be specified



Parallel simulation framework for SystemC

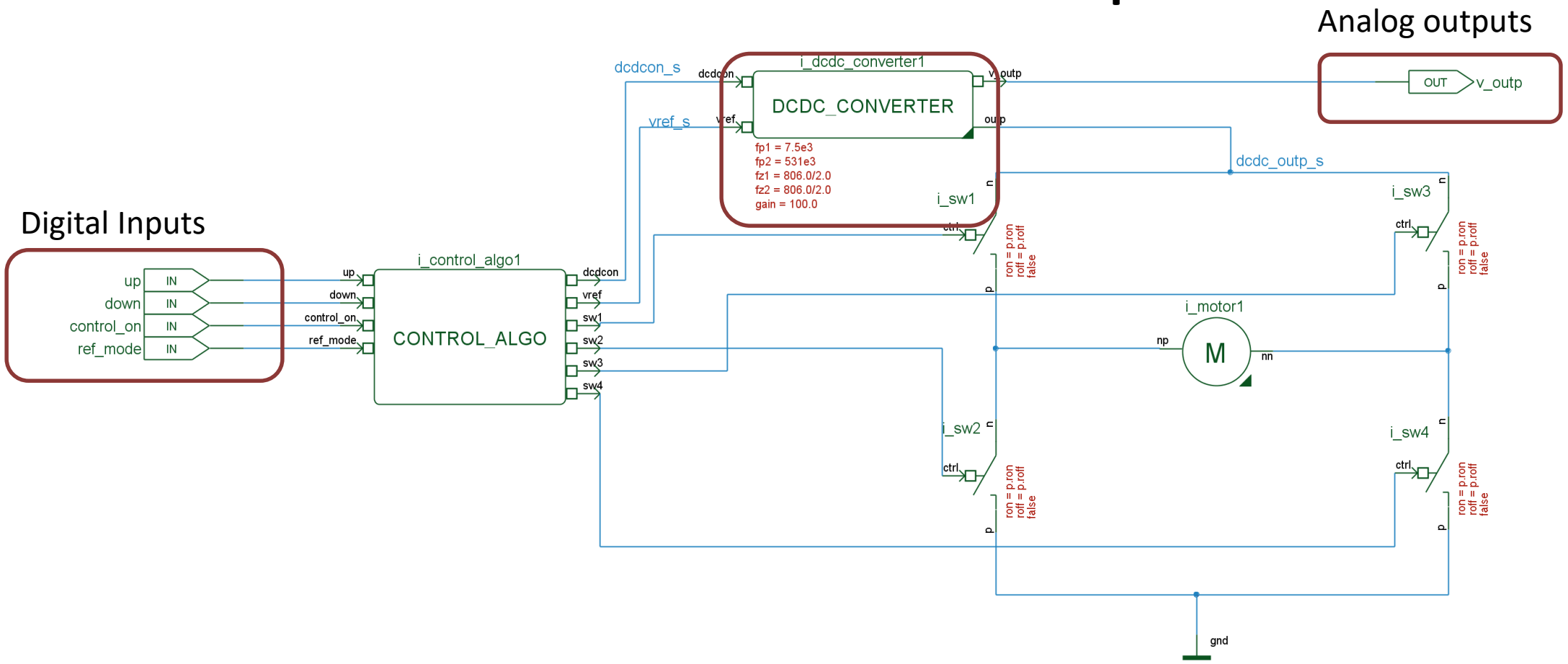
```
int sc_main(int argc, char* argv[]) {
  // spawn 20 runs
  statistics_handle sh = sca_statistics_start(CRV, 20);
  // create trace file for this run
  std::stringstream trace_file_name;
  trace_file_name << "parallel_run_" << sh.run_number();
  parameter_constraints c;
  // instantiate CRAVE constratints object
  parameter_constraints c("parameter_constraints");
  // randomize CRAVE constraints
  c.randomize();
  // Create parameters and assign randomized values
  dut::params p_dut;
  p_dut.p_R1 = c.R1;
  p_dut.p_R2 = c.R2;
  …
  // Bind parameters to DUT
  dut* i_dut;
  i_dut = new dut("i_dut", p_dut);
  …
};
```



DC-DC Converter Example – Set Up

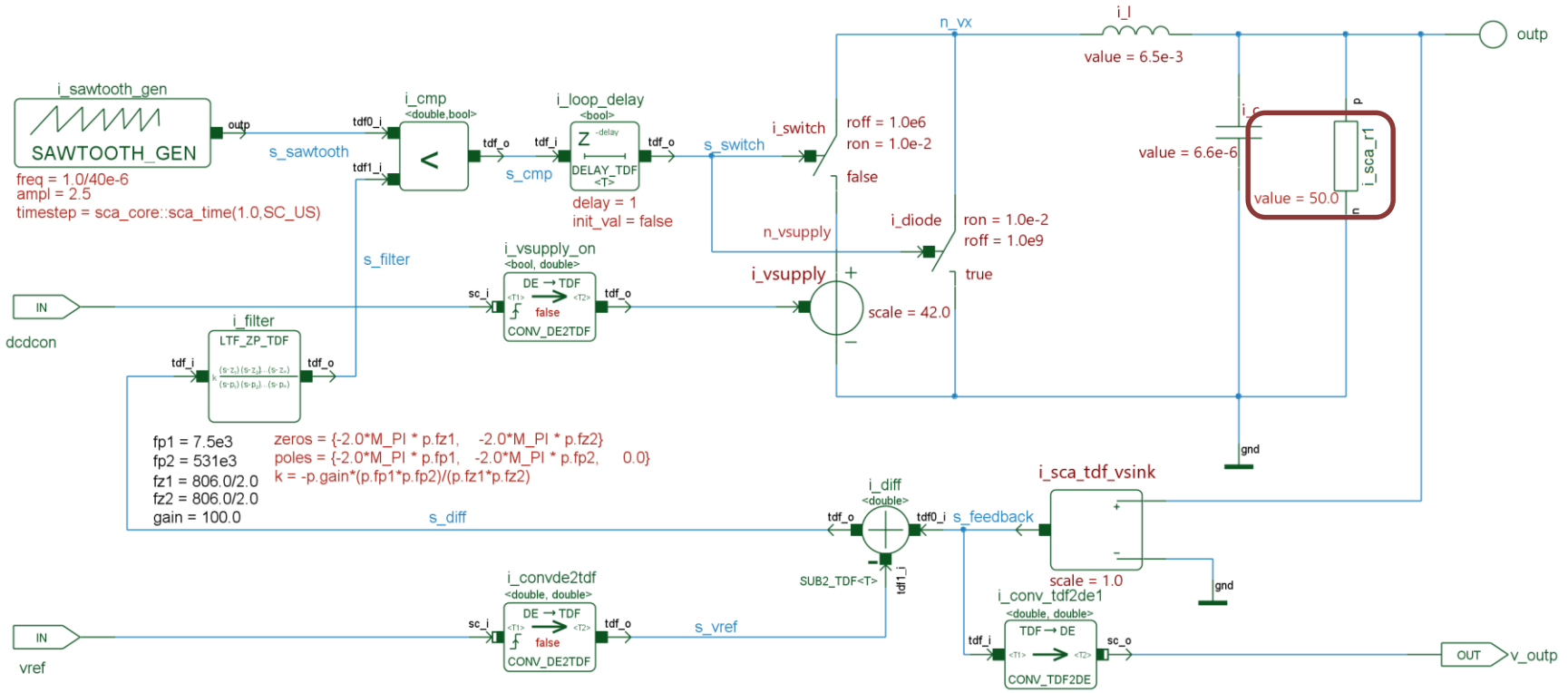
- Example for methodology: SystemC AMS DC-DC converter system
  - Motor controlled by digital controller circuit (SystemC discrete event)
  - DC-DC converter modeled using AMS model of computation (timed data flow (TDF) and electric linear network (ELN))
- UVM-SystemC test bench
  - Randomized control sequences
  - Setting of reference voltage for DC-DC converter
  - Automatic check of settling time for desired voltage
- Randomization of resistor values using CRAVE (parameter constraints)

DC-DC Converter Example

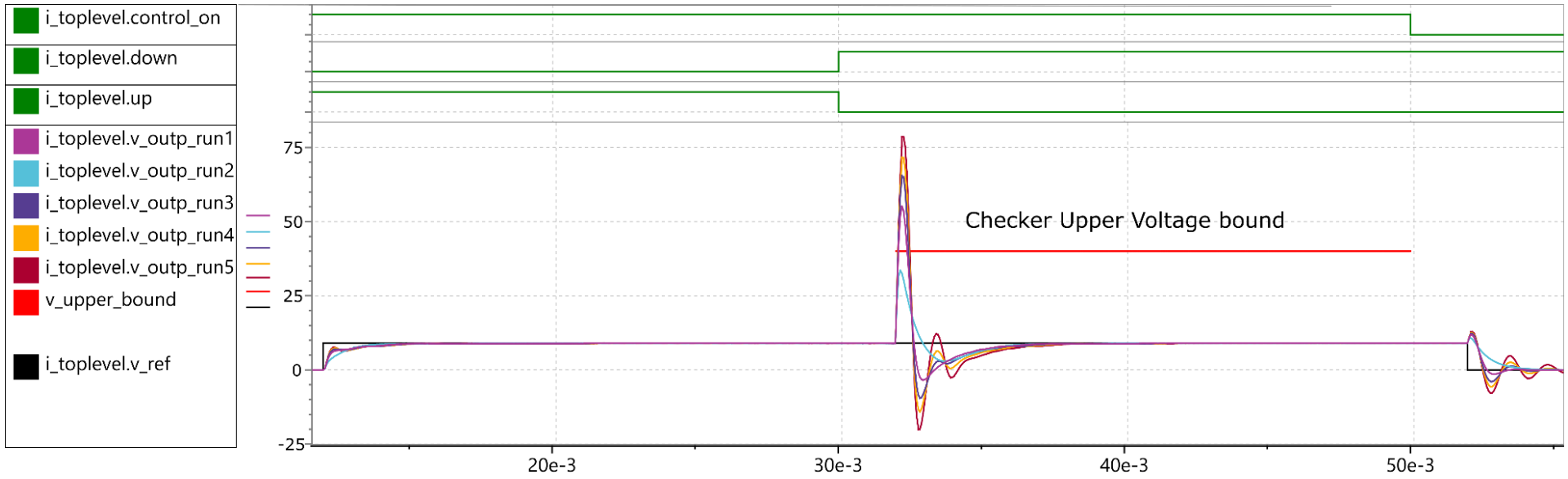


DC-DC Converter Example

Controls output voltage settling time



DC-DC Converter Example – Results



DC-DC Converter Example – Results

- DUT combinations can be created automatically based on constraints
- Parallel simulation of DUT combinations allows to run many simulations
  - nearly ideal speedup
  - 100 simulations in 68 seconds, one test case is simulating 60ms
- Automatic check of settle times possible

Summary

- CRAVE real value extensions will be contributed to Accellera SystemC Verification Working Group.
- UVM – Crave Adaption layer is work in progress, will be also contributed.
- *This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project CONVERS under contract no. 16ES0656.*

Questions