Constraint Randomization with CRAVE

Muhammad Hassan

Cyber Physical Systems,
DFKI GmbH, Bremen, Germany

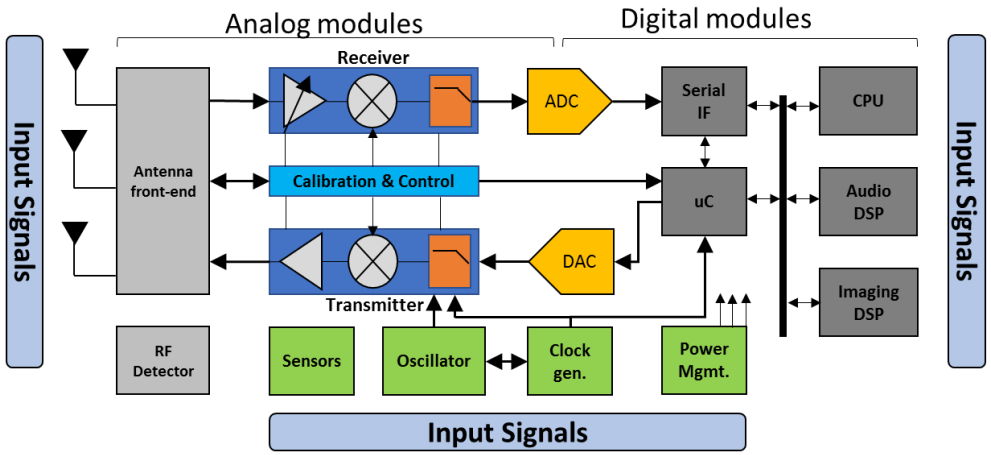
muhammad.hassan@dfki.de

CONVERS

Bundesministerium
für Bildung
und Forschung

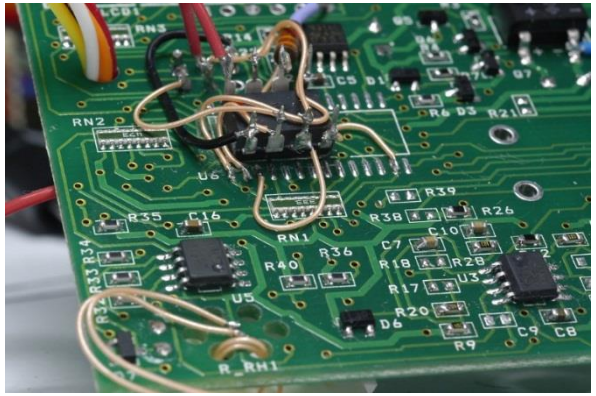
16ES0656

Motivation



Feature rich! Complex design & verification methodologies!

Typical heterogenous system



https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/#59b6ba7a9788
https://www.seguetech.com/how-much-test-coverage-enough-testing-strategy/segue-blog-how-much-test-coverage-for-testing-strategy/
https://zaxrosenberg.com/bugs-arent-wrong/
https://www.flickr.com/photos/kotse_m_/552067112/

Directed Testing

- Tests written with Device Under Test (DUT) specifications in mind
  - Only tests what is stated in datasheet
  - Stimuli with specific data values (hand-written)
  - Focused on to specific features
  - Difficult to maintain
- Coverage metrics
  - Structural
    - Statement
    - Branch
    - Dataflow etc..
  - Functional
- Difficult to catch corner-cases/hidden bugs

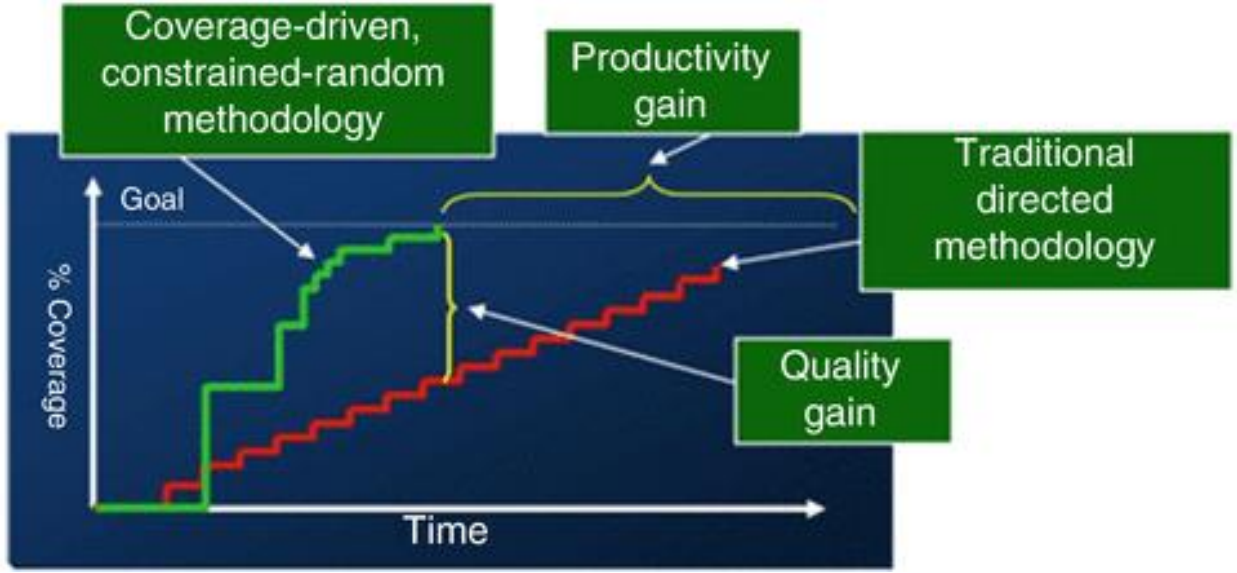
Legend

Branch Coverage			
Line Coverage			
Color	Line	Symbol	Description
Blue	Hit	+	Taken
Red	Not hit	-	Not Taken
		#	Not executed

```
180      Branch    :   Line    :
181 [T  F][T  F]:        2 :     a = r[SRC_A] ;
182              :        2 :     b = r[SRC_B] ;
183              :        2 :     c = r[SRC_C] ;
184              :        2 :     d = r[SRC_D] ;
185              :          :
186        [ + - ]:        2 :     if (a > b)
187 [ +  + ][ -  + ]:      2 :       if ((a > c) && (a > d) )
188              :        0 :         max = a;
189 [ +  + ][ +  - ]:      2 :       else if ((a > c) && (a < d)) //MC4 - replace && with ||
190              :        1 :         max = d;
191 [ +  - ][ +  - ]:      1 :       else if (((a < c) && (a > d))) //MC2 - negate expression
192              :        1 :         max = c;
193              :          :       else
194        [ #  # ]:        0 :         if (c > d)
195              :        0 :           max = c;
196              :          :         else
197              :        2 :           max = d;
198              :          :     else
199 [ #  # ][ #  # ]:      0 :       if ((b > c) && (b > d))
200              :        0 :         max = b;
201 [ #  # ][ #  # ]:      0 :       else if ((b > c) && (b < d))
202              :        0 :         max = d;
203 [ #  # ][ #  # ]:      0 :       else if ((b < c) && (b > d))
204              :        0 :         max = c;
205              :          :       else
206        [ #  # ]:        0 :         if (c > d)
207              :        0 :                 max = c;
208              :          :         else
209              :        0 :                 max = d;
210              :          :
211              :        2 :         r[MAX_VALUE] = max;
```

Constrained Randomization

- Randomization as the basis of verification
  - Constraints on stimulus
  - Automatic
  - Faster and accurate coverage closure
- Bugs detection due to
  - unexpected combinations of inputs
  - extreme input values
- Detects
  - Hidden bugs
  - Corner cases

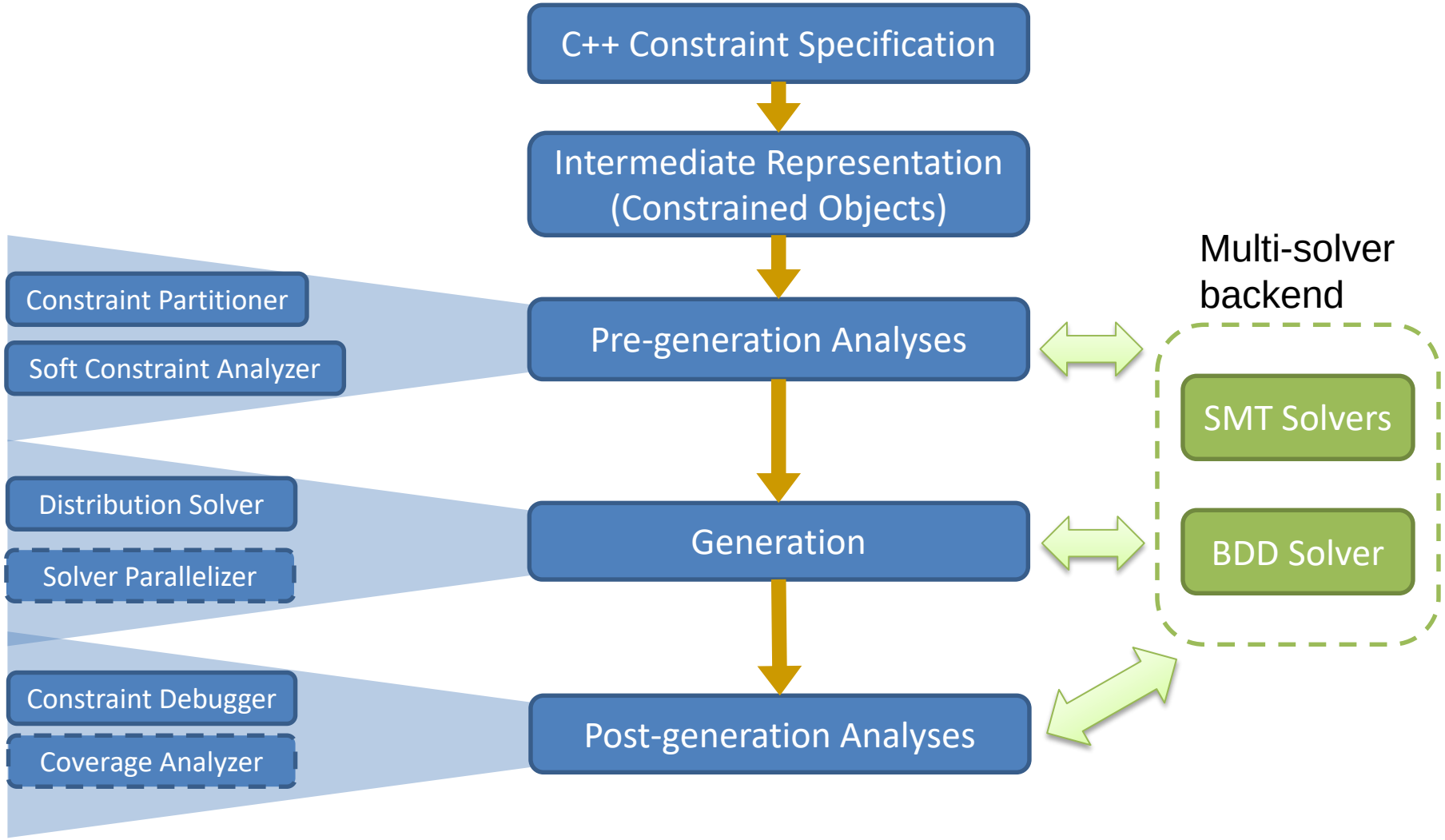


Mehta, Ashok B. "Constrained Random Verification (CRV)." *ASIC/SoC Functional Design Verification*. Springer, Cham, 2018. 65-74.

Basic of CRAVE

- **C**onstrained **Ra**ndom **V**erification **E**nvironment
- Open-source library
- Syntax and semantics closely followed SystemVerilog IEEE 1800 std
- Can be used with C++ and SystemC
- Random objects
- Random variables
- Hard/soft constraints
- Efficient constraint solvers

CRAVE Architecture



Random Variable and Constraints

- Single/Multiple constraints on a single variable

```
class item : public crv_sequence_item {
  crv_variable<int> v; //Random Variable
  crv_constraint c1 { v() < 10 }; //Constraint
  crv_constraint c2 { v() > 0 }; //Constraint
  item(crv_object_name) {}
};
```

CRAVE - Pre-Generation Analysis

- Constraints Partitioning
  - Disjoint set of constraints
  - Separate, and fast execution of constraint solver

- Hard / Soft constraint analyzer
  - Hard constraints – Must always be satisfied
  - Soft constraints – Can be ignored in case of conflict with hard constraints
    - Default behavior specification
    - Can be prioritized

Distribution

```
class item : public crv_sequence_item {
  item(crv_object_name) {}
  crv_variable<int> v; //Variable
  crv_constraint c {dist(v(),
    make_distribution(range<int>(0,9)))};//Distribution
};
```

v	0	1	2	3	4	5	6	7	8	9
Weight	1/10	1/10	1/10	1/10	1/10	1/10	1/10	1/10	1/10	1/10

Weighted Range

```
class item : public crv_sequence_item {
  item(crv_object_name) {}
  crv_variable<int> v; //Variable
  crv_constraint c {dist(v(),
    make_distribution(weighted_range<int>(0, 4, 40),
    weighted_range<int>(5, 9, 60)))}; //Weighted Range
};
```

v	0	1	2	3	4	5	6	7	8	9
Weight	0.08	0.08	0.08	0.08	0.08	0.12	0.12	0.12	0.12	0.12

Ranges and Distribution

```
class item : public crv_sequence_item {
 public:
  item(crv_object_name) {}
  crv_constraint c_src_addr_range{dist(src_addr(),
                make_distribution(range<int>(0, 9), range<int>(90, 99))) };




  crv_variable<int> src_addr, dest_addr;
};
```

src	0	1	2	3	4	5	6	7	8	9
weight	1/20	1/20	1/20	1/20	1/20	1/20	1/20	1/20	1/20	1/20

src	90	91	92	93	94	95	96	97	98	99
weight	1/20	1/20	1/20	1/20	1/20	1/20	1/20	1/20	1/20	1/20

Ranges and Distribution

```
class item : public crv_sequence_item {
 public:
  item(crv_object_name) {}



  crv_constraint c_dest_addr_range{dist(dest_addr(),
              make_distribution(weighted_range<int>(0, 2, 60),
                                weighted_range<int>(10, 12, 30),
                                weighted_range<int>(100, 103, 10))) };
  crv_variable<int> src_addr, dest_addr;
};
```

dest	0	1	2	10	11	12	100	101	102	103
Weight	2/10	2/10	2/10	1/10	1/10	1/10	0.025	0.025	0.025	0.025

Ranges and Distribution

```
class item : public crv_sequence_item {
 public:
  item(crv_object_name) {}
  crv_constraint c_src_addr_range{dist(src_addr(),
              make_distribution(range<int>(0, 9), range<int>(90, 99))) };

  crv_constraint c_dest_addr_range{dist(dest_addr(),
              make_distribution(weighted_range<int>(0, 2, 60),
                                weighted_range<int>(10, 12, 30),
                                weighted_range<int>(100, 103, 10))) };
  crv_variable<int> src_addr, dest_addr;
};
```

```
Output:
src_addr: 94 dest_addr: 12
src_addr: 93 dest_addr: 12
src_addr: 0 dest_addr: 102
...
```

Enum

```
CRAVE_BETTER_ENUM(car_type_enum, AUDI = 1, BMW = 2);
CRAVE_BETTER_ENUM(color_enum, RED, GREEN);
class item : public crv_sequence_item {
  crv_variable<car_type_enum> car;
  crv_variable<color_enum> color;
  crv_constraint car_color{
    if_then(car() == car_type_enum::AUDI, color() != RED),
    if_then(car() == car_type_enum::BMW, color() != BLUE)};
}};
```

Packet Example

```
class Packet : public crv_sequence_item {
 public:
  Packet(crv_object_name) {}
  crv_constraint c { src() > 10, src() < 15, dist(kind(),
                              make_distribution(range<uint8_t>(0, 255)))};
  crv_variable<sc_bv<32>> src, dst, data[8];
  crv_variable<sc_bv<8>> kind;
};
```

```
Output:
00000000000000000000000000001100  01010111
00000000000000000000000000001100  10010111
00000000000000000000000000001100  00100110
00000000000000000000000000001100  11100011
...
```

CRAVE Constraint Operators

```
crv_variable<type> 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 rshift{x2() == x1() << 1};
crv_constraint lshift{x1() == x2() >> 1};
crv_constraint range0{x1() >= 0 && x1() <= 5};
crv_constraint range1{x1() >= 0 && x1() <= x2()};
crv_constraint not{!x0() == x1()};
crv_constraint neg{~x() == x1()};
crv_constraint and{x2() == x0() & x1()};
crv_constraint or{x2() == x0() | x1()};
crv_constraint xor{x2() == x0() ^ x1()};
```

Soft Constraints

```
struct item : public crv_random_obj {
  crv_variable<int> src_addr;
  crv_variable<int> dest_addr;

  crv_constraint hard_c{ src_addr() <= 20, dest_addr() <= 100 };
  crv_soft_constraint soft_c{ src_addr() % 4 == 0 };

  item(crv_object_name) {}
};
```

Summary

- CRAVE library
  - C++, SystemC
  - Efficient solvers
- Real value support is under progress
  - Joint project CONVERS between DFKI & Coseda Technologies

- How to get CRAVE
  - www.systemc-verification.org/crave
  - Download / clone from GitHub!
  - Follow installation instructions given in INSTALL.txt
  - CRAVE is freely available under the **MIT license**.

Constraint Randomization with CRAVE

Muhammad Hassan

Cyber Physical Systems,
DFKI GmbH, Bremen, Germany

muhammad.hassan@dfki.de

CONVERS
Bundesministerium für Bildung und Forschung
16ES0656

Some useful links, and publicataions

- http://systemc-verification.org/
- https://www-cps.hb.dfki.de/home
- http://www.informatik.uni-bremen.de/agra/ger/index.php
- Muhammad Hassan, Daniel Große, Thilo Vörtler, Karsten Einwich, and Rolf Drechsler. Functional coverage-driven characterization of RF amplifiers. In FDL, pages 1-8, 2019. (Best Paper Candidate).
- Mehran Goli, Muhammad Hassan, Daniel Große, and Rolf Drechsler. Security validation of VP-based SoCs using dynamic information flow tracking. *it-Information Technology*, 2019.
- Mehran Goli, Muhammad Hassan, Daniel Große, and Rolf Drechsler. Automated analysis of virtual prototypes at electronic system level. In *GLSVLSI*, 2019.
- Muhammad Hassan, Daniel Große, Hoang M. Le, and Rolf Drechsler. Data flow testing for SystemC-AMS timed data flow models. In *DATE*, 2019.
- Thilo Vörtler, Karsten Einwich, Muhammad Hassan, and Daniel Große. Using constraints for SystemC AMS design and verification. In *DVCon Europe*, 2018. **(Best Paper Award)**.
- Muhammad Hassan, Daniel Große, Hoang M. Le, Thilo Vörtler, Karsten Einwich, and Rolf Drechsler. Testbench qualification for SystemC-AMS timed data flow models. In *DATE*, pages 857-860, 2018.
- Muhammad Hassan, Vladimir Herdt, Hoang M. Le, Daniel Große, and Rolf Drechsler. Early SoC security validation by VP-based static information flow analysis. In *ICCAD*, pages 400-407, 2017
- Muhammad Hassan, Vladimir Herdt, Hoang M. Le, Mingsong Chen, Daniel Große, and Rolf Drechsler. Data flow testing for virtual prototypes. In *DATE*, pages 380-385, 2017.
- Daniel Große, Hoang M. Le, Muhammad Hassan, and Rolf Drechsler. Guided lightweight software test qualification for IP integration using virtual prototypes. In *ICCD*, pages 606-613, 2016.