CRAVE 2.0: The Next Generation Constrained Random Stimuli Generator for SystemC

Hoang M. Le, Rolf Drechsler

University of Bremen, Germany

Agenda

- Motivation
- Basics of CRAVE
- Major new features in CRAVE 2.0
  - Soft Constraints
  - Distribution Constraints
  - Constraint Partitioning
- Wrap-up

Motivation

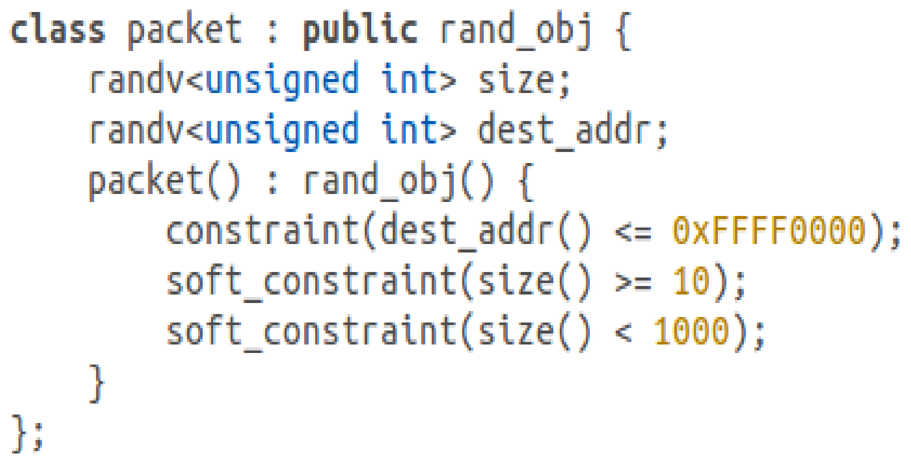
- SystemC
  - IEEE 1666-2011
  - C++ modeling @ multiple levels of abstraction
  - Accellera open-source reference simulator
- Verification of SystemC models
  - In a SystemC-centric design and verification flow
  - Constrained Random Verification (CRV): Accellera open-source SystemC Verification Library (SCV)
  - Lacking in capabilities versus SystemVerilog
    - Limited expressive power
    - Outdated non-scalable constraint solving technology

Our Vision

- Open-source free EDA software
- Powerful & extensible constrained random stimuli generator for SystemC
  - Leverage latest constraint solving technologies
  - Available at github: https://github.com/agra-uni-bremen/crave-bundle
  - Easy integration in a SystemC verification environment
    - DVCon'12: System Verification Methodology (SVM)
    - DVCon'14: Universal Verification Methodology using SystemC (UVM-SystemC)
    - Yours?

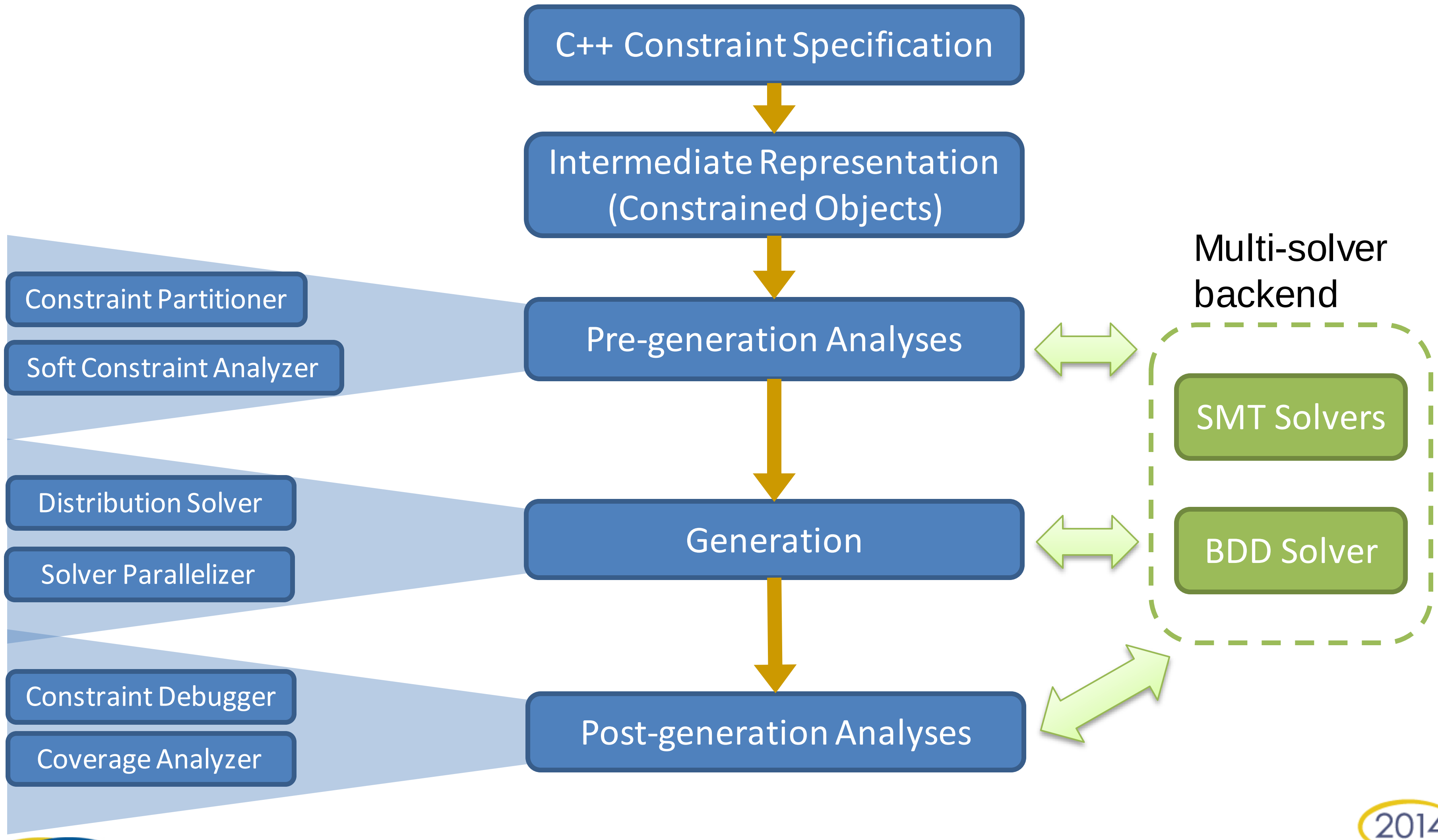
Basics of CRAVE

- **C**onstrained **Ra**ndom **V**erification **E**nvironment
- SystemVerilog-inspired syntax
- Random objects
- Random variables
- Hard/soft constraints



```
class packet : public rand_obj {
    randv<unsigned int> size;
    randv<unsigned int> dest_addr;
    packet() : rand_obj() {
        constraint(dest_addr() <= 0xFFFF0000);
        soft_constraint(size() >= 10);
        soft_constraint(size() < 1000);
    }
};
```

CRAVE Architecture



CRAVE 2.0

- Efficient shaping of the solution space
  - Soft constraints (new feature in SystemVerilog-2012)
  - Distribution constraints
- Faster generation
  - Constraint Partitioning

Soft Constraint

- Hard constraint must always be satisfied!
- Soft constraint shall be satisfied unless contradicted by
  - A hard constraint
  - A soft constraint with **higher priority**
- Typical use case: specify default values, modify later in subsequent specialization.
- In example, size() >= 10 should be overridden by size() >= 5

```
class packet : public rand_obj {
    randv<unsigned int> size;
    randv<unsigned int> dest_addr;
    packet() : rand_obj() {
        constraint(dest_addr() <= 0xFFFF0000);
        soft_constraint(size() >= 10);
        soft_constraint(size() < 1000);
    }
};
```

```
class short_packet : public packet {
    short_packet() : packet() {
        soft_constraint(size() >= 5);
        soft_constraint(size() < 10);
    }
};
```

Soft Constraint

- CRAVE priority scheme
  - Based on creation time of constraint
    - Created later → Higher priority
  - Compactible with SystemVerilog semantics
- But doesn't SCV support soft constraints already (SCV_SOFT_CONSTRAINT)?
  - No priority scheme
  - Drop all soft constraints if contradicted!

```
class packet : public rand_obj {
    randv<unsigned int> size;
    randv<unsigned int> dest_addr;
    packet() : rand_obj() {
        constraint(dest_addr() <= 0xFFFF0000);
        soft_constraint(size() >= 10);
        soft_constraint(size() < 1000);
    }
};
```

```
class short_packet : public packet {
    short_packet() : packet() {
        soft_constraint(size() >= 5);
        soft_constraint(size() < 10);
    }
};
```

Distribution Constraint

- Uniform distribution is not always wanted!
  - a < 10 or b < 10 will be extremely rare → hard to reach coverage closure
- Distribution constraints: user-defined biases to create interesting stimuli (corner cases)

```
class my_rand_obj: public rand_obj {
    randv<unsigned int> a, b, c;
    my_rand_obj() : rand_obj() {
        constraint(if_then(a() < 10, c() == 0));
        constraint(if_then(b() < 10, c() == 1));
        constraint(a() <= 10000000000);
        constraint(b() <= 10000000000);
        constraint(c() <= 10000000000);
    }
};
```

```
class my_rand_obj_ext: public my_rand_obj {
    my_rand_obj_ext() : my_rand_obj() {
        constraint(dist(a(),
            distribution<unsigned int>::create
                (weighted_range<unsigned int>(0, 9, 30)) // 30%
                (weighted_range<unsigned int>(10, 1000000000, 70)) // 70%
        ));
        constraint(dist(b(),
            distribution<unsigned int>::create
                (weighted_range<unsigned int>(0, 9, 50)) // 50%
                (weighted_range<unsigned int>(10, 1000000000, 50)) // 50%
        ));
    }
};
```

Distribution Constraint

- Generated values always in specified ranges (hard constraint)
- Distribution is best-effort but cannot be guaranteed (impossible in many cases)
- Solving algorithm
  - Generate values based on specified distributions
  - Try to solve the constraints with these generated values
  - Drop one randomly chosen value and repeat last step

Distribution Constraint

- But doesn’t SCV also support distributions?
  - They cannot be mixed with other constraints!
  - False report of unsolvable constraints
- CRAVE result

	0	1	2	3	4	5	6	7	8	9	Total	%
a	230	200	230	234	216	214	228	220	219	229	2220	22,2
b	437	420	446	451	408	406	421	413	410	413	4225	42,3

Constraint Partitioning

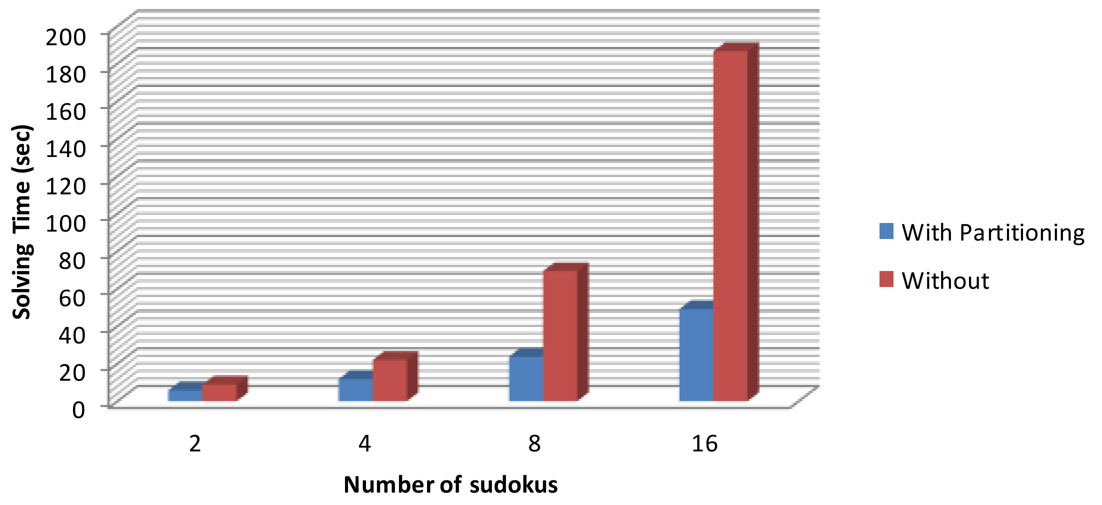
- Observation: multiple disjoint constraint sets in a big set of constraints
- Performance gain by solving disjoint constraint sets independently
- Very efficient implementation possible (e.g. with union-find data structure)

```
class packet : public rand_obj {
    randv<unsigned int> size;
    randv<unsigned int> dest_addr;
    packet() : rand_obj() {
        constraint(dest_addr() <= 0xFFFF0000);
        soft_constraint(size() >= 10);
        soft_constraint(size() < 1000);
    }
};
```

```
class multicast_packet : public packet {
    randv<unsigned int> other_dest_addr[16];
    multicast_packet() : packet() {
        for (int i = 0; i < 16; i++)
            constraint(other_dest_addr[i]() <= 0xFFFF0000);
    }
};
```

Constraint Partitioning – Experiments

- Multicast packet
  - SCV (no CP available) timeouts
  - CRAVE (both with and without CP) returns instantly
- Sudoku puzzle



Wrap-up

- CRAVE 2.0
  - Open-source constrained random stimuli generator for SystemC/C++
  - Powerful constraint solving technologies
  - Extensible architecture
  - Frequently added new features, recently: soft constraints, distribution constraints, constraint partitioning
- What‘s next?
  - Integration of coverage models
  - Graph-based specification

Acknowledgement

SPONSORED BY THE

Federal Ministry
of Education
and Research

EffektiV

DFG

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project EffektiV under contract no. 01IS13022E and by the German Research Foundation (DFG) within the Reinhart Koselleck project DR 287/23-1.

Thanks for your time!
Questions?