Discover Over-Constraints by Leveraging Formal Tool

Dongsheng Ouyang, Ray Zhang, Lucas Liu, Doris Yin, Wayne Ding
NVIDIA Semiconductor Technology (Shanghai) Co., Ltd.

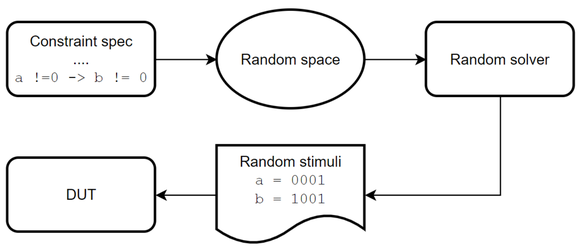
2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

INTRODUCTION

Constrained Random Testing (CRT) is a dominant and powerful stimulus generation methodology. One of the most challenging problems of CRT is over-constraint. Over-constraints are constraints that are too narrow, ruling out certain legal input scenarios.

Over-constraints will cause RTL bug escape and project schedule slip. Traditionally, verification engineers rely on code review or simulation with coverage to find over-constraints, but this workflow is very time and resource-consuming and does not work very well.

We propose to use formal to identify over-constraints. With formal, we can report random variables' reachable space, identify unreachable expressions in constraints, and report unreachable bins in stimulus coverage. With the help of these reports, we identify over-constraints in a short turnaround time during the constraint composition stage.



RANDOM SPACE

A random variable's random space is the feasible region that satisfies all constraints on this variable.

After we get the random space, we can check if this variable is over-constrained or not by comparing the intended space against the random space. Once we get the list of over-constrained variables, we can discover over-constraints by a simple analysis of constraint expressions.

But when a set of complex constraints limit random variables, it is impossible to manually figure out a variable's random space. We should rely on a tool to exhaust a variable's random space.

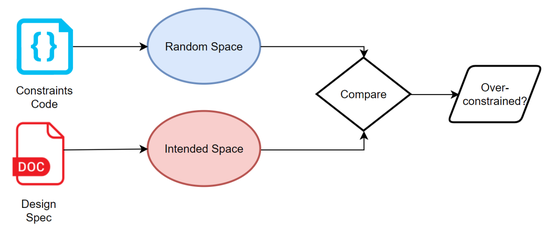
```
rand bit [2:0] a;
rand bit [2:0] b;
rand bit [2:0] c;

constraint cst {
  a >=2;
  b > a;
  4'(a+b) == c;
}
```

Variable	Reachable space
a	{2,3}
b	{3,4,5}
c	{5,6,7}

Solution table

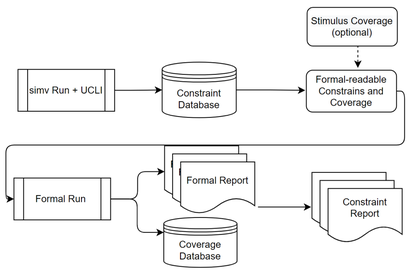
a/b	0	1	2	3	4	5	6	7
0	X	X	X	X	X	X	X	X
1	X	X	X	X	X	X	X	X
2	X	X	X	c=5	c=6	c=7	X	X
3	X	X	X	X	c=7	X	X	X
4	X	X	X	X	X	X	X	X
5	X	X	X	X	X	X	X	X
6	X	X	X	X	X	X	X	X
7	X	X	X	X	X	X	X	X



TECHNICAL SOLUTION

We propose a workflow to use formal to discover over-constraints. With formal, a series of reports are generated, which help to identify over-constraints quickly.

1. Use Synopsys UCLI (Unified Command-line Interface) to generate a constraint random database.
2. Convert random constraints and optional stimulus coverage into formal readable.
3. Run a formal tool to explore reachable space accurately. Formal reports will be generated after formal runs.
4. Generate constraints qualify reports. The review can find over-constraints from these reports quickly.



random constraints	SVA properties
`v inside {5, 10};`	`v inside {5, 10};`
`x dist {100 := 1, 200 := 2, 300 := 5}; *`	`x inside {100, 200, 300};`
`mode == little -> len < 10;`	`mode == little |-> len < 10;`
`if (mode == little) len < 10; else if (mode == big) len > 100;`	`if (mode == little) len < 10; else if (mode == big) len > 100;`
`unique {a, b, c};`	Unsupported
`solve s before d;`	Ignored
`soft length inside {32,1024};`	Ignored

CONSTRAINT REPORTS – I

Unreachable code report. Formal can help to prove the reachability of constraint expressions. These unreachable expressions are sources of over-constrained and under-constrained stimuli.

```
rand bit[3:0] a;
rand bit[3:0] b;
rand bit[1:0] error;

constraint cons {
  if(a+b > 4'hf)
    error == 1;
  else if(a+b < 4'h2)
    error == 2;
  else
    error == 0;
}
```

Unreachable Precondition of Constraints

Precondition Constraint	Source
(a + b > 4'hf) cons	test.sv:10

Random space report. Unreachable spaces will be marked red, and reachable spaces will be marked green. We develop a smart algorithm to divide random variables into 2 categories: enum variables with all possible values, and continuous variables with a min~max range. Reviewers can check variables' random space with design spec requirements to find over-constrained variables quickly.

ENUM Variable Stimulus Space Review Table

Variable	Supported Possibilities	Actual Possibilities	Gap	Diversity
winownet[0]	16	9	7	56.25
winownet[1]	16	9	7	56.25
winownet[2]	16			
winownet[3]	16			
winownet[4]	16			
winownet[5]	16			
winownet[6]	16			
winownet[7]	16			
winownet[8]	16			
winownet[9]	16			
winownet[10]	16			
winownet[11]	16			
winownet[12]	16			
winownet[13]	16			
winownet[14]	16			
winownet[15]	16			

winowner[0]

Bin	Reachability	Distribution
[0:0]	reachable	10.30%
[1:1]	reachable	10.26%
[2:2]	reachable	11.00%
[3:3]	reachable	10.93%
[4:4]	reachable	11.71%
[5:5]	reachable	12.64%
[6:6]	reachable	12.47%
[7:7]	reachable	12.36%
[8:14]	unreachable	0.00%
[15:15]	reachable	9.33%

CONTINOUS Variable Stimulus Space Review Table

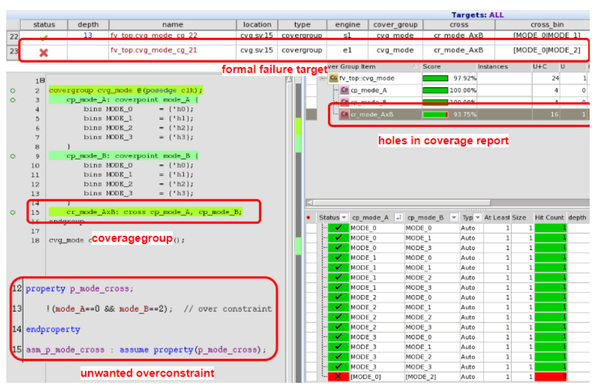
Variable	Theory S	Diversity
[model.Core[0].SetClockFrequency.Hertz][0]	0	0%
[model.Core[1].SetClockFrequency.Hertz][0]	0	0%
[model.Core[2].SetClockFrequency.Hertz][0]	0	0%
[model.Core[3].SetClockFrequency.Hertz][0]	0	0%
[model.localRSize][0]	0	0%
[model.localRSize][1]	0	0%
[model.localRSize][2]	0	0%
[model.localRSize][3]	0	0%

model.localRSize[0]

Bin	Reachability	Distribution
[0:2107]	unreachable	0.00%
[2108:2108]	reachable	0.00%
[2109:133297]	reachable	99.99%
[133298:133298]	reachable	0.01%
[133299:4294967295]	unreachable	0.00%

CONSTRAINT REPORTS – II

Unreachable stimulus coverage report. User-defined stimulus coverpoints can be optionally added as random state targets, these coverpoints can feed into formal, and formal will report all its unreachable bins. Reviewers can judge whether those unreachable bins are over-constrained cases or can be added in a waiver.



CONCLUSIONS

We proposed a new solution to over-constraints detection by leveraging the formal tool in the simulation world. It creates a working model with a speed-of-light turnaround from constraint directly to report. It is much more advanced than the original turnaround from constraint writing to test creating, to coverage regression, and finally to coverage report analysis to fix possible constraint issues.

REFERENCES

[1] Systematic Constraint Relaxation (SCR): Hunting for Over-Constrained Stimulus, Debarshi Chatterjee, DVCon US 2022.
[2] Synopsys, VCS® Unified Command Line Interface User Guide R-2020.12, December 2020, https://www.synopsys.com/
[2] https://www.jenkins.io/

We would like to thank your colleagues Liang Li, Mark Zhang, and Farmer Wang at NVIDIA for their valuable suggestions during this effort.

For any questions, please contact us through email at douyang@nvidia.com