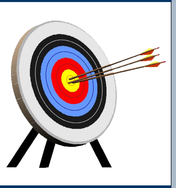
Tackling Random Blind Spots with Strategy-Driven Generation

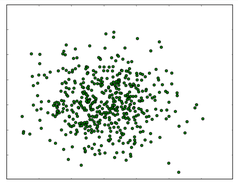
Matthew Ballance
matt_ballance@mentor.com

Engineer-Directed Tests

Verify known cases
Targeted
Focused
Functional Coverage Tracked



Open-Loop Random Tests



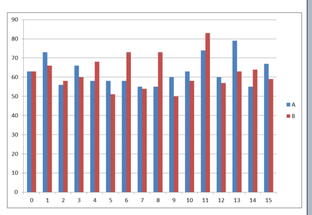
Scatter-shot
Find lurking bugs
Generate un-envisioned cases
Get to edges of constraint space

Open-Loop Random Challenges

No metrics
Random resistant corner cases

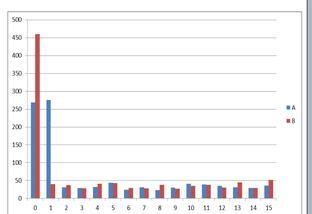
Unconstrained Random: Even Value Distribution

```
class unconstrained;
  rand bit[3:0]      A;
  rand bit[3:0]      B;
endclass
```



Unconstrained Random: Uneven Value Distribution

```
class unconstrained;
  rand bit[3:0]      A;
  rand bit[3:0]      B;
  constraint c {
    if (A > 1) {
      B == 0;
    }
  }
endclass
```



Key cases likely to be missed!

Strategy-Driven Generation

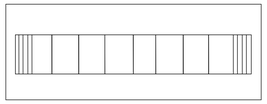
Automatically hits random-resistant cases
Identifies high-value tests from constraints
Identifies key value / variable combinations
Targets selected values during simulation

Pattern-Based Target Value Selection

Spreads stimulus across reachable space

Uniform Ranges
- Divides reachable domain into N ranges
- Spreads values across reachable domain

Edge Ranges
- Place single-value bins at min/max
- Min/max values are often corner cases



Constraint-Guided Target Value Selection

Ensures constraint branches are exercised

Condition-inferred target values
- Equality expression in if/implication
- Inside expression in if/implication

```
rand bit[15:0]              A;
rand bit[15:0]              B;

constraint c {
  if (A inside {[1:5]}) {
    B < 10;
  }
}
```

→ Target range A 1..5

Body Constraint-inferred target values
- Equality expression in if/implication body
- Inside epxression in if/implication body

```
rand bit[15:0]              A;
rand bit[15:0]              B;

constraint c {
  if (A < 2) {
    B == 20;
  }
}
```

→ Target value B=20

Individual Variable Combination Strategy

Independently target variables
- Interactions occur randomly
- Enables targeting more value ranges

All-Pairs Variable Combination Strategy

Approach from the software test domain
Selects pairs, triples, quads of variables

Case study data suggests
- 70% bugs triggered with pairs
- No bugs required 6 or more to trigger

Constraint-Guided Combination Strategy

Targets constraint-related variables

```
rand bit[15:0]              A;
rand bit[15:0]              B;

constraint c {
  if (A < 2) {
    B == 20;
  }
}
```

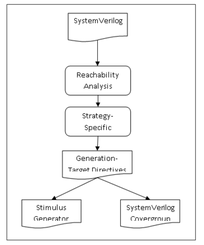
→ A and B are related and should be crossed

Works well with high constraint density
Less useful with low constraint density

Proof of Concept Implementation

Uses an intelligent testbench automation solver
- Constraint-based stimulus description
- Goal-based value generation

Process
- Import SV class fields/constraints
- Analyze variables for reachability
- Identifies strategy-specific goals
- Creates stimulus generator class
- Creates strategy-specific SV covergroup



Results

Compare strategy-driven and random
- Use Ethernet packet class
- Apply pairwise strategy
- Target maximum 64 target ranges

```
class ethmac_tx_seq_item;
  rand frame_fmt_e       frame_fmt;
  rand bit                     pad;
  rand bit               crc;
  rand bit               has_tag;
  rand bit[15:0]         len;
  rand bit[15:0]         payload_len;
  constraint c {
    len inside {[4:4096]};
    if (len < 46) { pad == 0; }

    if (frame_fmt == FRAME_FMT_ETH) {
      crc == 1;
      if (has_tag) {
        payload_len inside {[42:1500]};
        len == (payload_len + 6+6+2+4);
      } else {
        payload_len inside {[46:1500]};
        len == (payload_len + 6+6+2);
      }
    } else {
      len == payload_len;
    }
  }
endclass
```

Compare Random and Strategy-Driven
- Random progress stops at 85%
- Strategy-driven easily hits all cases