BUG 2 ⇐ Baud x #characters

data.randomize() with {band< 1000; band > 900;}

```
vcs
simv +ntb_random_seed=5
simv
```

$random

Functional Coverage

Look at sample Midterm

- Interfaces
- clking blcks
- Mod ports
- constraints
- assertions

□ Line coverage:

```
  if (x)
☑   y<=c;
  else
☑   y<=d;
```

Toggle coverage:

```
          ↑ ↑
logiz 2   0 1
```

Definitions

Coverage

% of verification objective that has been met

Two types of Coverage

Objectives that can be automatically inferred (ie : line coverage, toggle coverage)

User specified, correlates with some design intent (functional coverage)

(i.e. the baud_rate?

Coverage Model in SV

The covergroup construct encapsulates the specification of a coverage model. Each covergroup specification can include the following components:

A clocking event that synchronizes the sampling of coverage points

A set of coverage points

Cross coverage between coverage points

Every entry in the functional spec of a project should correlate to some covergroup

Key Aspect

The key aspects of functional coverage are as follows:

It is user-specified and is not automatically inferred from the design.

It is based on the design specification (i.e., its intent) and is thus independent of the actual design code or its structure.

Simple Example

```
class DATA_t;
   rand logic [7:0] addr;
   rand logic [7:0] data;
   rand logic rw;
endclass
covergroup cg @(posedge clk);
   coverpoint data.addr;
   coverpoint data.data;
   coverpoint data.rw;
endgroup
```

256

sampling @ every clock cycle

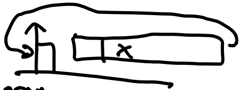
bin[0] 0-3
bin[1] 4-7

coverpoint → bins

addr	~~0→255~~	64
data	~~0→255~~	64
rw	~~0→1~~	2

```
cg  cg1;
DATA_t data;
initial begin
  cg1 = new;
  data = new;
repeat (100) begin
 data.randomize();
 @(posedge clk) ;
 $display("%d ,cg1.get_coverage();
end
$finish;
end
```

random baud
chars
send_to_ uart
wait(fifo_empty);



prog

should sample once per loop

Example Results

logic [31:0] x

coverpoint x;

2^{32} bins Very expensive

After 10 random samples, %coverage = 42%

Explanation
urg –dir simv.vdb –format text

```
VARIABLE  EXPECTED UNCOVERED COVERED PERCENT GOAL WEIGHT AT LEAST
AUTO BIN MAX COMMENT
                         -hit none      hit once
data.addr 64       54          10      15.62   100  1      1        64
data.data 64       56          8       12.50   100  1      1        64
data.rw   2        0           2     100.00  100  1      1        2
```

%coverage is the average(15.6,12.5,100) = 42%

“bins” are automatically created

User Specified Bins

```
covergroup cg @(posedge clk);
  coverpoint data.addr  {
    bins low = { [0:63] };
    bins med = { [100:150] };
    bins high = { [200:255] };
  }
  coverpoint data.data  {
    bins low = { [0:63] };
    bins med = { [100:150] };
    bins high = { [200:255] };
  }
  coverpoint data.rw;
endgroup
```

May want to create bin for specific value

Cross Coverage

```
covergroup cg @(posedge clk);
   coverpoint data.addr  {
      bins low = { [0:63] };
      bins med = { [100:150] };
      bins high = { [200:255] };
   }
   coverpoint data.data  {
      bins low = { [0:63] };
      bins med = { [100:150] };
      bins high = { [200:255] };
   }
   coverpoint data.rw;
   cross data.addr, data.data, data.rw;
endgroup
```

$256 \times 256 \times 2$

$\approx 131k$

default binning

$64 \times 64 \times 2$

$\approx 15k$

Cross Coverage

```
18 bins in all  (3 x 3 x 2)

Summary for Group   test::cg

CATEGORY  EXPECTED UNCOVERED COVERED PERCENT
Variables 8       1        7       88.89
Crosses   18       16       2       11.11


Variables for Group  test::cg


VARIABLE  EXPECTED UNCOVERED COVERED PERCENT GOAL WEIGHT AT LEAST AUTO BIN MAX COMMENT
data.addr 3       1        2       66.67   100  1      1       0
data.data 3       0        3       100.00  100  1      1       0
data.rw   2       0        2       100.00  100  1      1       2


Crosses for Group  test::cg


CROSS EXPECTED UNCOVERED COVERED PERCENT GOAL WEIGHT AT LEAST PRINT MISSING COMMENT
cg_cc 18       16       2       11.11   100  1      1       0
```

UART

baud

num chars.

character tree

#bits 5,6,7,8

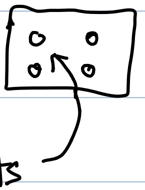
#stop bits

#start " "

parity

Random Stimulus

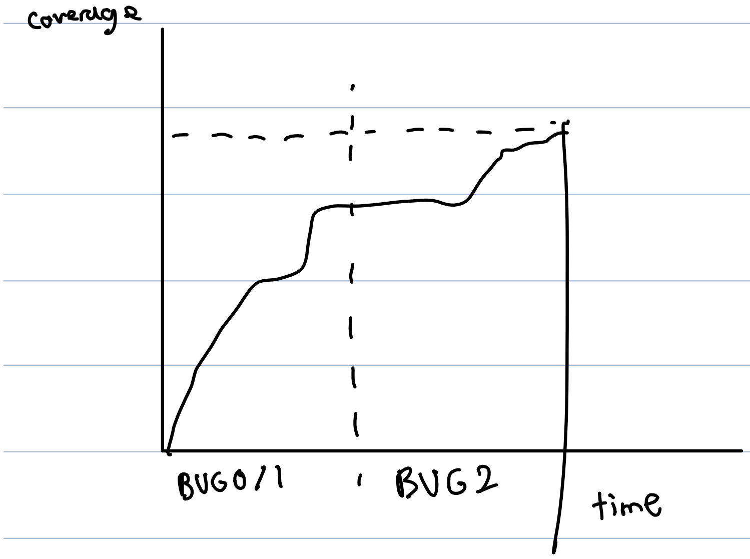
Coverage Holes



Biased Directed Tests

① Finding Bugs

② Coverage


coverage
BUG0/1
BUG2
time