COVERGATE: Coverage Exposed

Rich Edelman, Mentor, A Siemens Business

Fremont, CA

Mentor® A Siemens Business

Functional Coverage

A Coverage Collector

```
236 class coverage_collector extends uvm_subscriber # (transaction);
237   `uvm_component_utils (coverage_col
238
239   transaction t;
240
241   covergroup cg;
242     rw_cp:    coverpoint t.rw;
243     addr_cp:  coverpoint t.addr {
244       bins regular [] = {[0:99]};
245       ignore_bins ignore_addr = {[100:$]};
246     }
247     data0_cp:  coverpoint t.data [7:0] {
248       bins regular [] = {[0:99]};
249       ignore_bins ignore_data = {[100:$]};
250     }
251     data1_cp:  coverpoint t.data [15:8] {
252       bins regular [] = {[0:99]};
253       ignore_bins ignore_data = {[100:$]};
254     }
255     data2_cp:  coverpoint t.data [23:16] {
256       bins regular [] = {[0:99]};
257       ignore_bins ignore_data = {[100:$]};
258     }
259     data3_cp:  coverpoint t.data [31:24] {
260       bins regular [] = {[0:99]};
261       ignore_bins ignore_data = {[100:$]};
262     }
263
264      addr_X_rw: cross addr_cp, rw_cp;
265     data0_X_rw: cross data0_cp, rw_cp;
266   endgroup
267
268   virtual function void sample (transaction t);
269     this.t = t;
270     cg.sample ();
271   endfunction
272
273   function new (string name = "coverage_collector",
274       uvm_component parent = null);
275     super.new (name, parent);
276     cg = new ();
277   endfunction
278
279   function void write (transaction t);
280     sample (t);
281     `uvm_info ("COVERAGE", $sformatf ("Coverage=%0d%% (t=%s)",
282       cg.get_inst_coverage (), t.convert2string ()), UVM_MEDIUM)
283   endfunction
284 endclass
```

Covergroup

Coverpoints

Coverbins

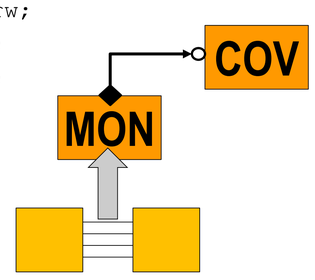
Cross Coverage

The sample() function

```
# UVM_INFO tb.sv(281) @ 717190: .a2.cc [COVERAGE] Coverage=55% (t=[] rw=READ, addr=499, data=500)
# UVM_INFO tb.sv(281) @ 717190: .a2.cc [COVERAGE] Coverage=55% (t=[] rw=READ, addr=499, data=500)
# UVM_INFO tb.sv(281) @ 717410: .a2.cc [COVERAGE] Coverage=55% (t=[] rw=READ, addr=699, data=700)
```

A Simple Monitor

```
task run_phase(uvm_phase phase);
  forever begin
    @(posedge mif.CLK);
    if ((mif.VALID == 1) &&
        (mif.READY == 1)) begin
      t = transaction::type_id::create("t");
      if (mif.rw == READ) begin
        t.rw = mif.rw;
        t.addr = mif.addr;
        @(negedge mif.READY);
        t.data = mif.rd;
      end
      else if (mif.rw == WRITE) begin
        t.rw = mif.rw;
        t.addr = mif.addr;
        t.data = mif.wd;
      end
      else if (mif.rw == IDLE) begin
        t.rw = mif.rw;
        t.addr = 'z;
        t.data = 'z;
      end
      ap.write(t);
    end
  end
endtask
```



Bad Functional Coverage

A common mistake with coverpoints is how the bins are assigned values.

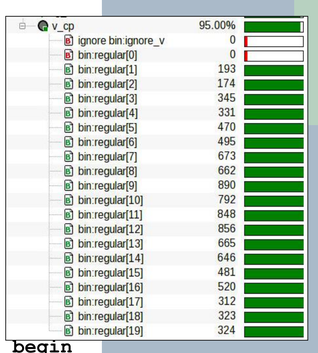
```
bins regular      = {[0:99]}; makes   1 bin
bins regular[10] = {[0:99]}; makes  10 bins
bins regular[]    = {[0:99]}; makes 100 bins
covergroup cg;
    rw_cp:    coverpoint t.rw;
    addr_cp:  coverpoint t.addr {
      bins regular[] = { [0:99] };
      ignore_bins  ignore_addr = { [100:$]
};
```

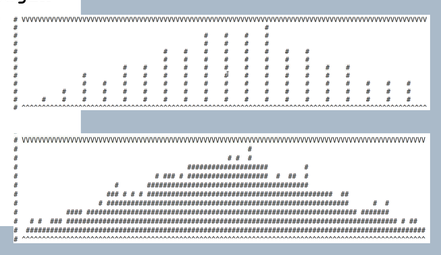
Random Number Distribution

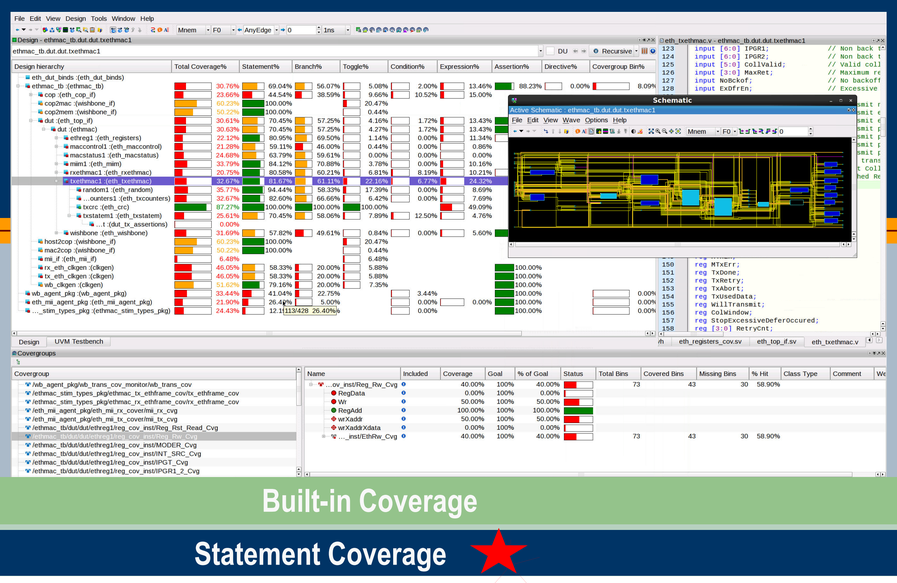
```
class DIST;
  int my_bins[100];

  function void add(int value);
    my_bins[value]++;
  endfunction

  function void print();
    int largest = 0;
    int per_y;
    for (int x = 0; x < 100; x++)
      if (my_bins[x] > largest)
        largest = my_bins[x];
    per_y = largest / 10;
    for (int y = 10; y >= 1; y--) begin
      for (int x = 0; x < 100; x++)
        if (my_bins[x] >= y*per_y)
          $write("#");
        else
          $write(" ");
      $write("\n");
    end
  endfunction
endclass
```







Built-in Coverage

Statement Coverage

What statements were executed? What statements were NOT executed?

```
eth_macstatus.v - ethmac_tb.dut.dut.macstatus1
216   // LatchedMRxErr
217✓  always @ (posedge MRxClk or posedge Reset)
218   begin
219✓    if(Reset)
220✓      LatchedMRxErr <= 1'b0;
221    else
222 X Xf  if(MRxErr & MRxDV & (RxStatePreamble | RxStateSFD | (|RxStateData) | RxStateIdle & ~Transmitting))
223 Xs    LatchedMRxErr <= 1'b1;
224✓    else
```

Code Coverage Analysis

Show coverage type: Multiple GoTo Line

Line#	Coverage ty	Hit	Coverage%	Details
220	Statement	✓	100.00%	LatchedMRxErr <= 1'b0;
222	Condition	X	0.00%	if(MRxErr & MRxDV & (RxStatePreamble | RxStateSFD | (|RxStateData) | RxStateIdle & ~Transmitting))
222	Branch	X	0.00%	if(MRxErr & MRxDV & (RxStatePreamble | RxStateSFD | (|RxStateData) | RxStateIdle & ~Transmitting))
223	Statement	X	0.00%	LatchedMRxErr <= 1'b1;

Branch Coverage

What branches were executed? What branches were NOT executed?

Condition Coverage

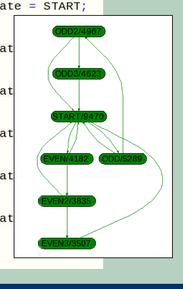
Each input to a condition must be covered.
For an input to be covered, it must be able to control the output to 0 and 1
If B is always zero, then A will not be covered, since it cannot control the output

A & B

Finite State Machine Coverage

Every state, every transition

```
dut.sv - top.dut
24✓  always @(posedge CLK) begin
25✓    even_or_odd = addr & 2'b01;
26
27 X X  case (state)
28✓    START:
29✓      if (even_or_odd == 0) state = EVEN;
30✓      else state = ODD;
31✓    ODD:
32✓      if (even_or_odd == 0) state = START;
33✓      else state = ODD2;
34✓    EVEN:
35✓      if (even_or_odd == 0) stat
36✓      else state = START;
37✓    ODD2:
38✓      if (even_or_odd == 0) stat
39✓      else state = ODD3;
40✓    EVEN2:
41✓      if (even_or_odd == 0) stat
42✓      else state = START;
43✓    ODD3:
44✓      if (even_or_odd == 0) stat
45✓      else state = START;
46✓    EVEN3:
47✓      if (even_or_odd == 0) stat
48✓      else state = START;
49     endcase
50   end
```



Expression Coverage

Similar to Condition Coverage – with a truth table

```
Coverage Details
Instance: ethmac_tb.dut.dut.txethmaci.txcrc
File:                                uvm-ethmac/rtl/eth_crc.v
Line: 103
Expression Coverage for:
((Enable & (((Data_Crc[3] ^ (Data_Crc[2] ^ Data_Crc[0])) ^ (Crc[28] ^ Crc[30])) ^ Crc[31])) ^ Crc[3])
Expression Coverage: 4 out of 8 input terms covered = 50.00%

Input Terminal        Covered
--------------        -------
Enable                N
Data_Crc[3]           N
Data_Crc[2]           N
Data_Crc[0]           N
Crc[28]               Y
Crc[30]               Y
Crc[31]               Y
Crc[3]                Y

Rows:      Hits(->0)   Hits(->1)   Target          Non-Masking Condition(s)
-----      ---------   ---------   ------          ------------------------
Row: 1           0           0   Enable_0        (((Data_Crc[3] ^ (Data_Crc[2] ^ Data_Crc[0])) ^ (Crc[28] ^ Crc[30])) ^ Crc[31])
Row: 2           1           1   Enable_1        (((Data_Crc[3] ^ (Data_Crc[2] ^ Data_Crc[0])) ^ (Crc[28] ^ Crc[30])) ^ Crc[31])
Row: 3           1           1   Data_Crc[3]_0   Enable
Row: 4           0           0   Data_Crc[3]_1   Enable
Row: 5           1           1   Data_Crc[2]_0   Enable
Row: 6           0           0   Data_Crc[2]_1   Enable
Row: 7           1           1   Data_Crc[0]_0   Enable
Row: 8           0           0   Data_Crc[0]_1   Enable
Row: 9           0           1   Crc[28]_0       Enable
Row: 10          1           0   Crc[28]_1       Enable
Row: 11          0           1   Crc[30]_0       Enable
Row: 12          1           1   Crc[30]_1       Enable
Row: 13          0           1   Crc[31]_0       Enable
Row: 14          1           1   Crc[31]_1       Enable
Row: 15          0           1   Crc[3]_0        -
Row: 16          1           1   Crc[3]_1        -
```

Toggle Coverage

Did the signal toggle? 0 → 1 and 1 → 0

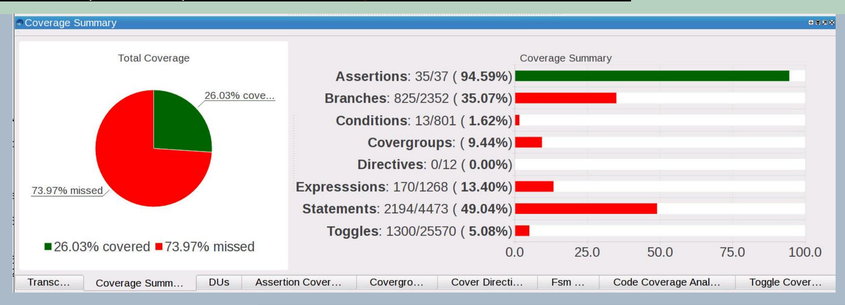
Toggle Coverage

Show: All Filter:

Name	Kind	1H->0L	0L->1H	0L->Z	Z->0L	1H->Z	Z->1H	#Nodes	#Toggled	% Toggled	% 01	% full	Include	Mode
ByteCnt	Packed Array	0	0	0	0	0	0	16	0	0.00%	0.00%	0.00%		internal
ByteCntMax	Net	0	0	0	0	0	0	1	0	0.00%	0.00%	0.00%		internal
DlyCrcCnt	Packed Array	0	0	0	0	0	0	3	0	0.00%	0.00%	0.00%		internal
ExcessiveDeferCnt	Net	0	0	0	0	0	0	1	0	0.00%	0.00%	0.00%		internal
IncrementByteCnt	Net	0	0	0	0	0	0	1	0	0.00%	0.00%	0.00%		internal
IncrementNibCnt	Net	1	1	0	0	0	0	1	1	100.00%	100.00%	33.33%		internal
NibCnt	Packed Array	5	5	0	0	0	0	16	5	31.25%	31.25%	10.41%		internal
ResetByteCnt	Net	0	0	0	0	0	0	1	0	0.00%	0.00%	0.00%		internal
ResetNibCnt	Net	1	1	0	0	0	0	1	1	100.00%	100.00%	33.33%		internal

Transcript | Coverage Summary | DUs | Assertion Coverage | Covergroups | Cover Directives | Fsm List | Code Coverage Analysis | Toggle Coverage

Coverage Debug



Contact information

Rich Edelman
Mentor, A Siemens Business
rich_edelman@mentor.com