Calculating FSMs coverage

Ben Cohen
http://www.systemverilog.us/   ben@systemverilog.us

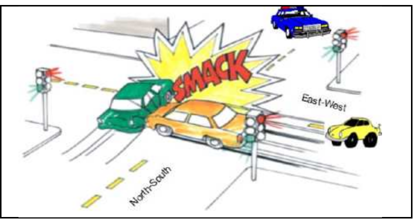
Abstract

There are many techniques to calculate the coverage of states for FSM machines. One technique is to use the SystemVerilog **cover property** or **`cover sequence`** statements. However, to cover all possible combinations of an FSM machine can be very tedious when done manually because of the many possible combinations. There are tools that can automate this process. However, an alternate approach is to use the **covergroup** statement that performs coverage between the previous state of the FSM and the current state, effectively computing the coverage of the state transitions. In addition cross coverage between states of different FSMs can also be computed. This paper demonstrates this concept by example.

The model

Consider a traffic light controller that controls two traffic lights: a *north-south* and *east-west*. The states of the lights can be expressed with an enumerated type and with module variables as follows:

```
typedef enum {ALL_OFF, RED, YELLOW, GREEN,
              RED_FLASH, YELLOW_FLASH, ALL_FLASH  } lights_e;
 module traffic;
    lights_e ns, ew; // noth-south, east-west
```



Using `cover  sequence` statement

To obtain coverage of the FSM states for each set of lights (i.e., `ns, ew`), one could use a set of **`cover sequence`** statements. For example,

```
  default clocking cb_ck1 @ (posedge(clk));
  cq_ns_RED2GREEN:  cover sequence(ns==RED ##1 ns==GREEN);
  cq_ns_GREEN2YELLOW: cover sequence(ns==GREEN ##1 ns==YELLOW);
  // …
```

Another coverage of interest for this controller is the cross coverage between the *north-south* and *east-west* lights. This can again be done with the cover sequence; for example,

```
  cq_nsRED_ewRED: cover sequence(ns==RED ##0 ew==RED);
  cq_nsGREEN_ewGREEN: cover sequence(ns==GREEN ##0 ew==GREEN);
  // …
```

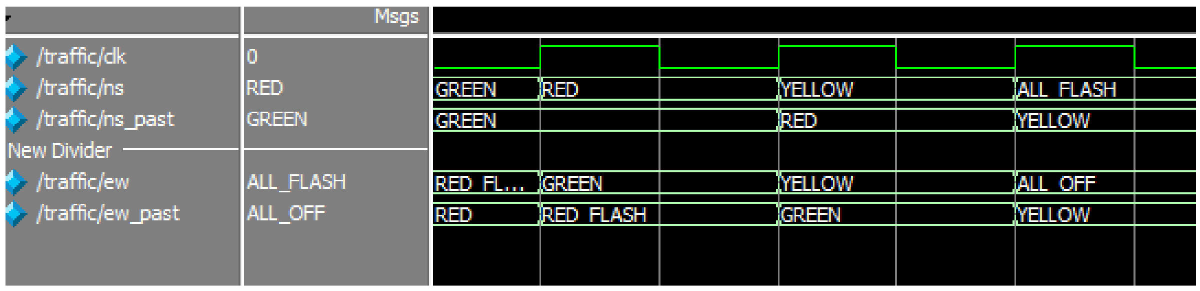
To cover all the states can be very tedious, even for an FSM that has as few as seven states because of the number of possibilities.

Using `covergroup` statement

The **`covergroup`** with cross coverage is a simpler approach to obtain coverage and cross coverage for FSM machines. The next state sequence of the FSM can be computed into a variable that copies the current state at the clock edge. For example,

```
  lights_e ns, ew, ns_past, ew_past; // noth-south, east-west, and next states
  always_ff @ (posedge clk)  begin : aly
        ns_past <= ns;
        ew_past <= ew;
    end : aly
```

Thus for a *north-south* light, the coverage from its past state to the current state (i.e., the state transition) can be expressed as: `ns_pastXns: cross ns_past, ns;`
Below is a waveform of the signals in a simulation.



The coverage of the FSM can thus be expressed using the covergroup as follows:

```
    covergroup tlights_cg;
        type_option.merge_instances = 0;
        option.per_instance = 1;
        option.get_inst_coverage = 1;
        // ns_cp: coverpoint ns;   // redundant because of the cross
        // ew_cp: coverpoint ew;
        ns_pastXns: cross ns_past, ns;
        ew_pastXew: cross ew_past, ew;
        nsXew: cross ns, ew;
    endgroup
```

The complete model and results is shown below. For simplicity, because the actual traffic light FSM constructions are of little importance to demonstrate the coverage, they are modeled using weighted random values.

```
import uvm_pkg::*; `include "uvm_macros.svh"
typedef enum {ALL_OFF, RED, YELLOW, GREEN, RED_FLASH, YELLOW_FLASH, ALL_FLASH  } lights_e;
class t_c;
    rand lights_e ns, ew; // noth-south, east-west, and next states
    constraint NS_cst {ns dist{ALL_OFF:=1, RED:=20, YELLOW:=6, GREEN:=20,
                               RED_FLASH:=3, YELLOW_FLASH:=3, ALL_FLASH:=3};}
endclass : t_c
module traffic;
    bit clk;
    lights_e ns, ew, ns_past, ew_past; // north-south, east-west, and next states
    t_c t=new();
    initial forever #10 clk=!clk;

    // emulation of traffic light FSMs
    always_ff @ (posedge clk)  begin : aly
        if(!t.randomize())  `uvm_error("run_phase", "seq randomization failure");
        ns <= t.ns;
        ew <= t.ew;
        ns_past <= ns;
        ew_past <= ew;
        t_cg.sample(); // Sampling of covergoup
    end : aly

    covergroup tlights_cg;
        type_option.merge_instances = 0;
        option.per_instance = 1;
        option.get_inst_coverage = 1;
        // ns_cp: coverpoint ns;   // redundant because of the cross
        // ew_cp: coverpoint ew;
        ns_pastXns: cross ns_past, ns;
        ew_pastXew: cross ew_past, ew;
        nsXew: cross ns, ew;
    endgroup
    tlights_cg t_cg = new; // instantiation of covergroup
endmodule : traffic
```

Sample results of the cross coverage and cross coverage is shown below:

```
COVERGROUP COVERAGE:
-------------------------------------------------------------------------------------------
Covergroup                                              Metric      Goal/ Status
                                                                    At Least
-------------------------------------------------------------------------------------------
Covergroup instance \/traffic/t_cg                       84.3%        100 Uncovered
    Coverpoint ns_cp                                    100.0%         100 Covered
        covered/total bins:                                  7          7
        missing/total bins:                                  0          7
        bin auto[ALL_OFF]                                    1          1 Covered
        bin auto[RED]                                       16          1 Covered
        bin auto[YELLOW]                                     2          1 Covered
        bin auto[GREEN]                                     17          1 Covered
        bin auto[RED_FLASH]                                  6          1 Covered
        bin auto[YELLOW_FLASH]                               4          1 Covered
        bin auto[ALL_FLASH]

Cross ns_pastXns                                      46.9%         100 Uncovered
        covered/total bins:                                 23         49
        missing/total bins:                                 26         49
        bin <auto[ALL_OFF],auto[ALL_OFF]>                    1          1 Covered
        bin <auto[RED],auto[RED]>                           10          1 Covered
        bin <auto[GREEN],auto[RED]>                          6          1 Covered
        bin <auto[RED],auto[YELLOW]>                         1          1 Covered
        bin <auto[GREEN],auto[YELLOW]>                       1          1 Covered
        bin <auto[ALL_OFF],auto[GREEN]>                      1          1 Covered
        bin <auto[RED],auto[GREEN]>                          2          1 Covered
        bin <auto[YELLOW],auto[GREEN]>                       1          1 Covered
…
     bin <auto[ALL_FLASH],auto[YELLOW_FLASH]>           2          1 Covered
        bin <auto[RED],auto[ALL_FLASH]>                      2          1 Covered
        bin <auto[YELLOW],auto[ALL_FLASH]>                   1          1 Covered
        bin <auto[GREEN],auto[ALL_FLASH]>                    1          1 Covered
        bin <auto[RED],auto[ALL_OFF]>                        0          1 ZERO
        bin <auto[YELLOW],auto[ALL_OFF]>                     0          1 ZERO
        bin <auto[GREEN],auto[ALL_OFF]>                      0          1 ZERO
        bin <auto[RED_FLASH],auto[ALL_OFF]>                  0          1 ZERO
```

The comple model code and results are availabel at
http://systemverilog.us/trafficlight_coverage.zip

---

http://www.systemverilog.us/
* SystemVerilog Assertions Handbook, 3rd Edition, 2013 ISBN 0-9705394-3-6 ** NEW BOOK **
* A Pragmatic Approach to VMM Adoption 2006 ISBN 0-9705394-9-5
* Using PSL/SUGAR for Formal and Dynamic Verification 2nd Edition, 2004, ISBN 0-9705394-6-0
* Real Chip Design and Verification Using Verilog and VHDL, 2002 isbn 0-9705394-2-8
* Component Design by Example, 2001 ISBN 0-9705394-0-1
* VHDL Coding Styles and Methodologies, 2nd Edition, 1999 ISBN 0-7923-8474-1
* VHDL Answers to Frequently Asked Questions, 2nd Edition ISBN 0-7923-8115
