Functional Coverage Collection for Analog Circuits
— Enabling Seamless Collaboration between Design and Verification

Zhipeng Ye, Honghuang Lin and Asad Khan
Texas Instruments
Outline

• Background
• Motivation
• Analog Coverage Collector (ACC)
• Proposed Analog Coverage Collection Flow
• Example
• Conclusions
Why do we need Coverage?

• From manager: When are we really done with verification?

• From designer: Are my blocks being verified thoroughly at chip level?
Types of Coverage

• Two types of coverage:
  1. Code coverage
     A. Line coverage
     B. Path coverage
     C. Toggle coverage
     D. FSM coverage
  2. Functional coverage
Types of Coverage (cont.)

• Functional coverage is a measure of which design features have been exercised by the tests


• High quality Covergroups need to be constructed
Coverage in AMS Domain

- Coverage adoption in AMS domain has been in place
- Code coverage is not available for analog circuits
- Functional coverage is potentially available
- \( \text{value_of_coverage} \propto \text{quality_of_covergroup} \)
Challenge #1

- A lot of nets and nodes are in the analog circuits, even in a simple level shifter circuit
Challenge #2

• Multiple thresholds are available for each analog signal
Challenge #3

- More than one signal may need to be monitored at the same time to construct interesting scenarios.

**Scenario A**
- VDD0
- VDD1

**Scenario B**
- VDD0
- VDD1

V_{OUT} may become high even when V_{IN} is low in scenario A!
Challenge #4

- Documentation about analog circuit design is often times lacking details since historically analog designers are the verifiers as well

\[ BW = 1.5 \text{MHz} \]
\[ SR = 0.7 \text{V/\mu s} \]
Where are we today?

• For verification engineers, it is hard to construct high quality covergroups at the chip level testbench for analog circuits

• For analog design engineers, there are limited ways to transfer design knowledge
What is the solution?

- The issues mentioned can be easily addressed if the analog designers *have an efficient way to pass the design knowledge to the verification engineers.*
• ACCs are implemented with three views
  • Symbol
  • VerilogAMS
  • VerilogA
Analog Coverage Collector (cont.)

- Information like net name and checking conditions can be specified
- The output becomes high when a specified trigger condition is met
module voltage_condition(output logic condition);

parameter string vnet = "";
parameter string symbol = "<" from '{"<", ",>", ",="};

Initial begin
    if (symbol == ">") dir = 1;
    else if(symbol == "<") dir = -1;
    else dir = 0;
end

analog begin
    v_val = $cds_get_analog_value(H_full, "potential");
end
endmodule
Analog Coverage Collection Flow

Verification Engineer

Build ACC Repository

Instantiate ACCs into schematic

Specify necessary information in ACCs

Identify all the ACCs in the design netlist

Analog Designer
Identify ACCs from Netlist

```perl
foreach my $line (read_file("$netlist_file")) {
    if ($line =~ m/^module.*$/) {
        $line =~ s/\S+\s(\S+)\s.*$/\1/;
        chomp($line);
        $module_name = $line;
    } elsif ($line =~ m/^voltage_condition.*$/) {
        print("$module_name.voltage_condition\n");
    } elsif ($line =~ m/^current_condition.*$/) {
        print("$module_name.current_condition\n");
    } elsif ($line =~ m/^voltage_diff_condition.*$/) {
        print("$module_name.voltage_diff_condition\n");
    } else {#do nothing}
}
```

Result:

```
>./grep_acc.pl
LS.voltage_condition
LS.voltage_condition
LS.current_condition
LS.voltage_diff_condition
```
Analog Coverage Collection Flow

1. **Build ACC Library**
2. Instantiate ACCs into schematic
3. Specify necessary information in ACCs
4. Identify all the ACCs in the design netlist
5. Construct coverage groups
6. Collect coverage and feedback to analog designer

Verification Engineer

Analog Designer

Adjust ACCs if needed
Example Circuit – Level Shifter

Absolute voltage ACCs

Absolute current ACC

Relative voltage ACC

Schematic Based Checkers
Example chip level Schematic
module tb_sample_design();
electrical vddldo;
logic c_in1;
logic c_in2;
...

sample_design (*integer library_binding = "<library>") DESIGN (.vddldo(vddldo),
.vdd1p8v(vddlp8v),
);

sample_cov (*integer library_binding = "<library>")_covergroup (.in1(c_in1),
.in2(c_in2),
);

assign c_in1 = DESIGN.I1.I1.condition;
assign c_in2 = DESIGN.I1.I2.condition;

...
module sample_cov(in1, in2, ...);
input in1;
input in2;
...
covergroup sample_cg_in;
    option.per.instance = 1;
    cov_in1 : coverpoint in1;
    cov_in2 : coverpoint in2;
    cov_cross: cross cov_in1, cov_in2;
endgroup
...
initial begin
    sample_cg_in cg_in;
    cg_in = new();
    ...
end
endmodule
Coverage Results

Instance (default scope): tc_sample_design Bringup

Overall Covered Grade: 87.5%  Functional Covered Grade: 87.5%

Cover groups

<table>
<thead>
<tr>
<th>Name</th>
<th>Overall Average Grade</th>
<th>Overall Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no filter)</td>
<td>(no filter)</td>
<td>(no filter)</td>
</tr>
<tr>
<td>unblkl.cg_in</td>
<td>100%</td>
<td>0 / 0 (100%)</td>
</tr>
<tr>
<td>unblkl.cg_out</td>
<td>100%</td>
<td>8 / 8 (100%)</td>
</tr>
<tr>
<td>unblkl.cg_r0</td>
<td>100%</td>
<td>5 / 0 (62.5%)</td>
</tr>
<tr>
<td>unblkl.cg_MP1</td>
<td>93.67%</td>
<td>7 / 0 (87.5%)</td>
</tr>
</tbody>
</table>

Showing 4 items

Items | unblkl.cg_r0

<table>
<thead>
<tr>
<th>Name</th>
<th>Overall Average Grade</th>
<th>Overall Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no filter)</td>
<td>(no filter)</td>
<td>(no filter)</td>
</tr>
<tr>
<td>cov_i1</td>
<td>100%</td>
<td>2 / 2 (100%)</td>
</tr>
<tr>
<td>cov_i2</td>
<td>50%</td>
<td>1 / 2 (50%)</td>
</tr>
<tr>
<td>aw cov_cross</td>
<td>50%</td>
<td>2 / 4 (50%)</td>
</tr>
</tbody>
</table>
Conclusions

• ACC is an efficient method to pass information from analog designers to verification engineers

• Meaningful Covergroups can be easily constructed for analog circuits to track verification progress

• ACCs stay with schematic, make them reusable and portable
Questions & Answers