2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

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

Zhipeng Ye, Honghuang Lin and Asad Khan

Texas Instruments

Outline

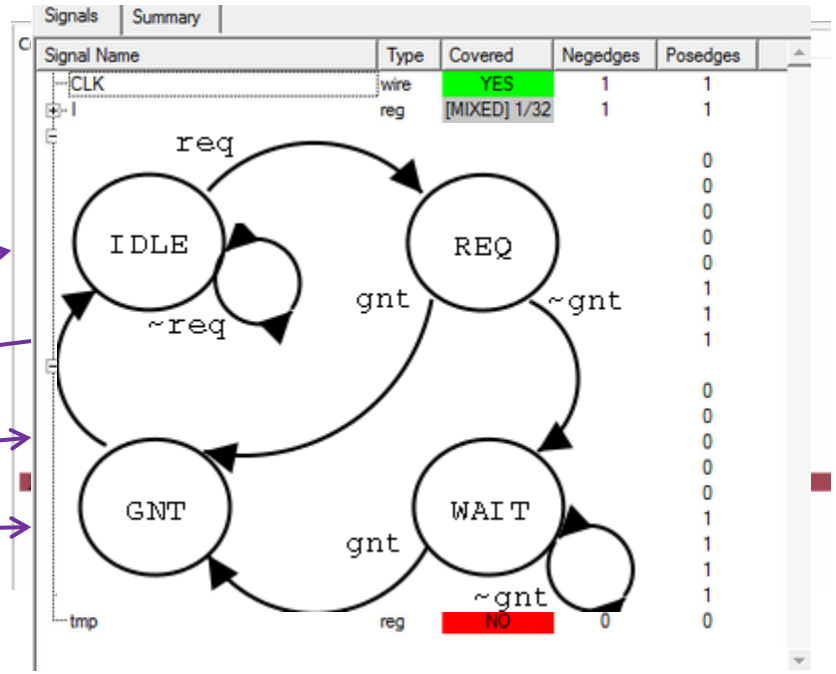
- Background
- Motivation
- Analog Coverage Collector (ACC)
- Proposed Analog Coverage Collection Flow
- Example
- Conclusions

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?

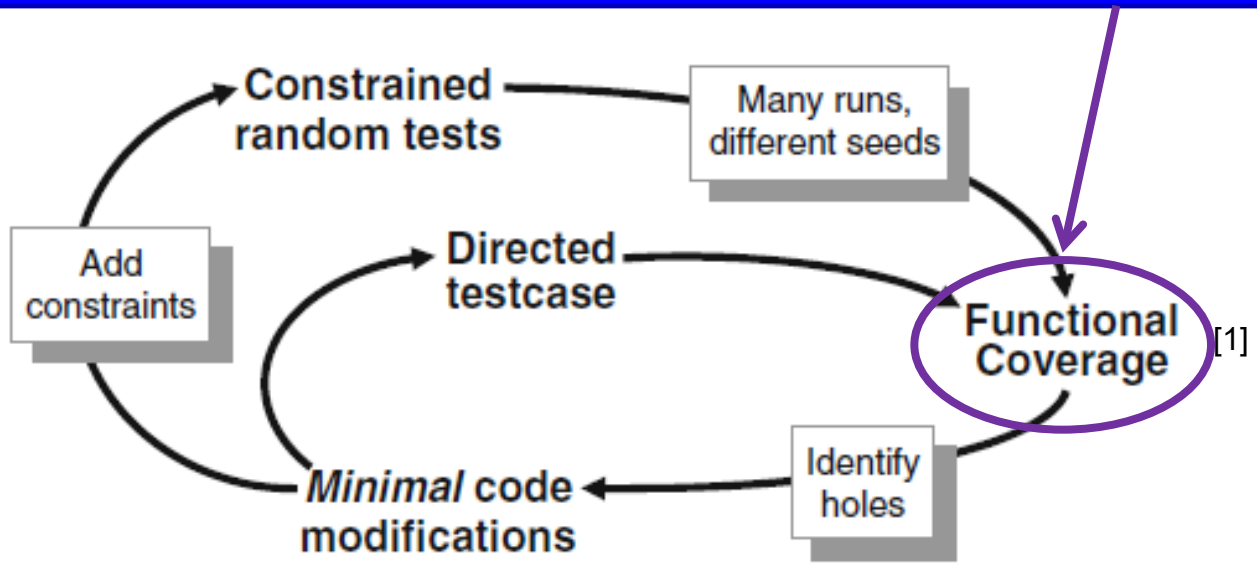
Types of Coverage

- 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

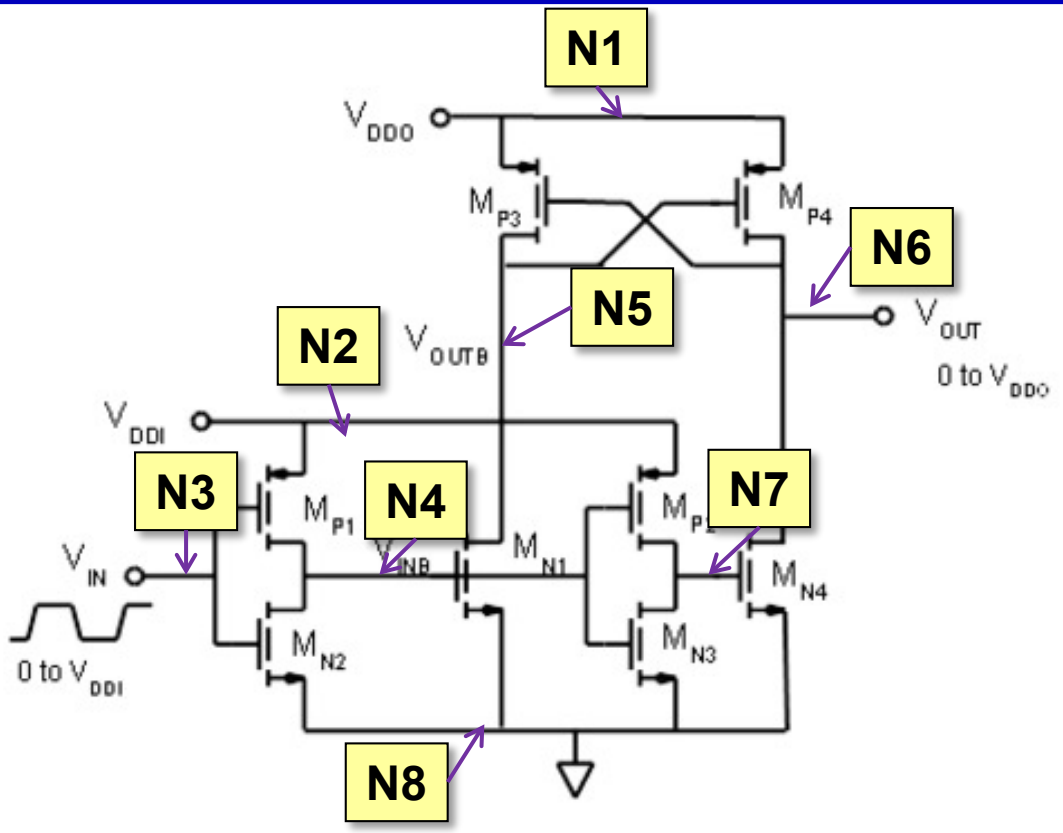
[1] C. Spear and G. Tumbush, "*SystemVerilog for Verification*", Springer, 2012.

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
- *value_of_coverage* $\propto$ *quality_of_covergroup*

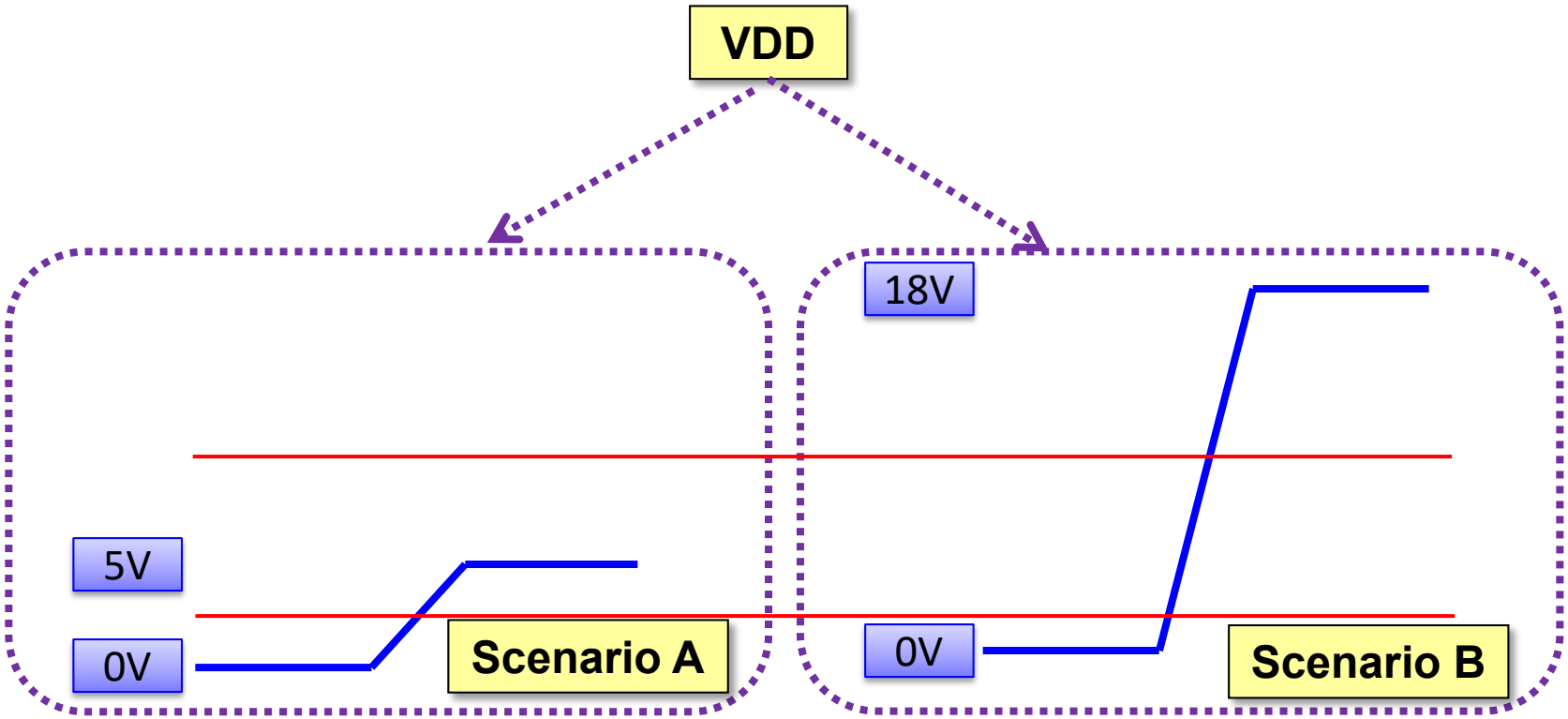
Challenge #1

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



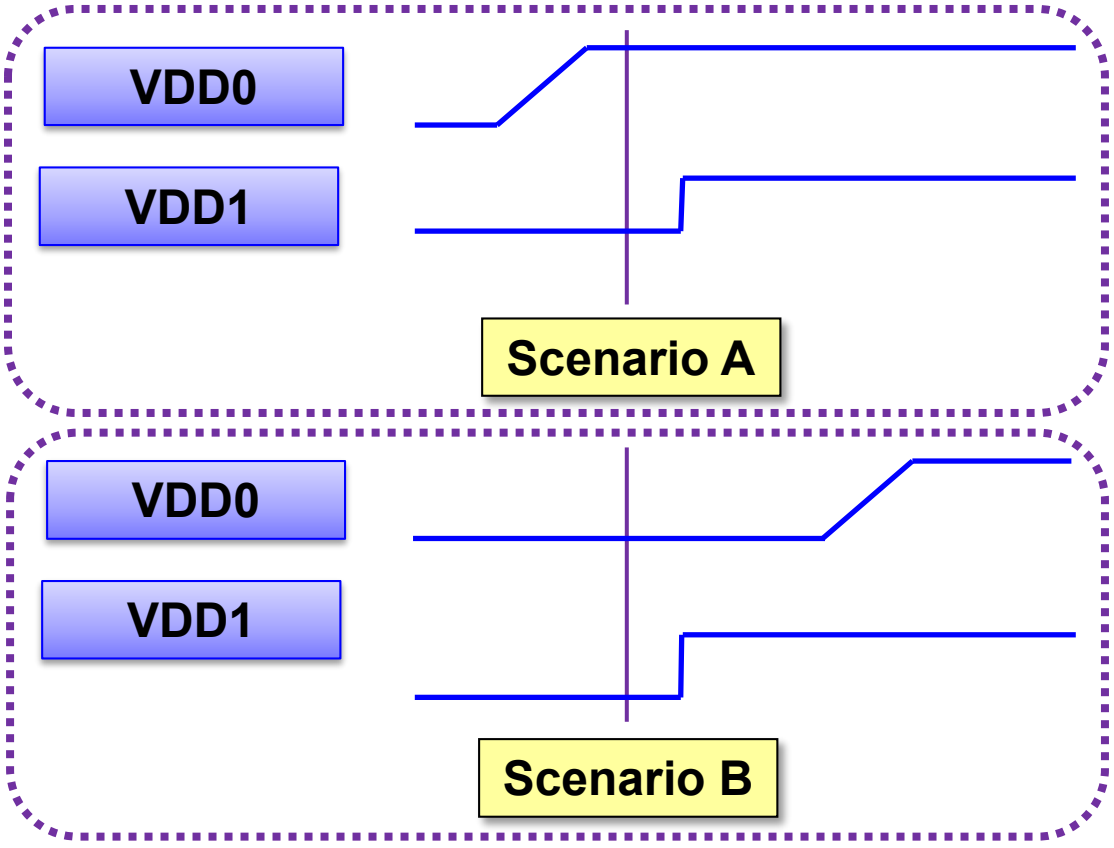
Challenge #2

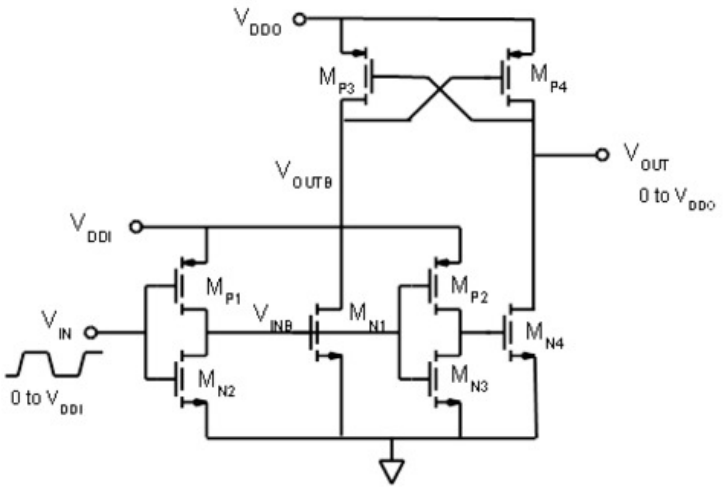
- Multiple thresholds are available for each analog signal



Challenge #3

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

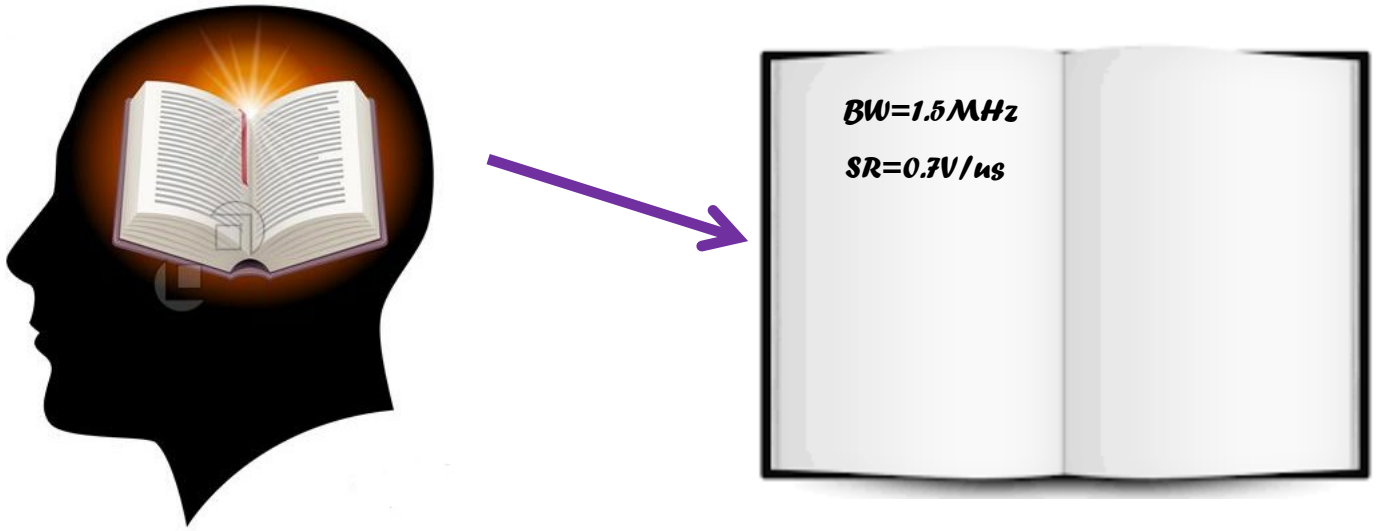




V_{OUT} may become high even when V_{IN} is low in scenario A !

Challenge #4

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



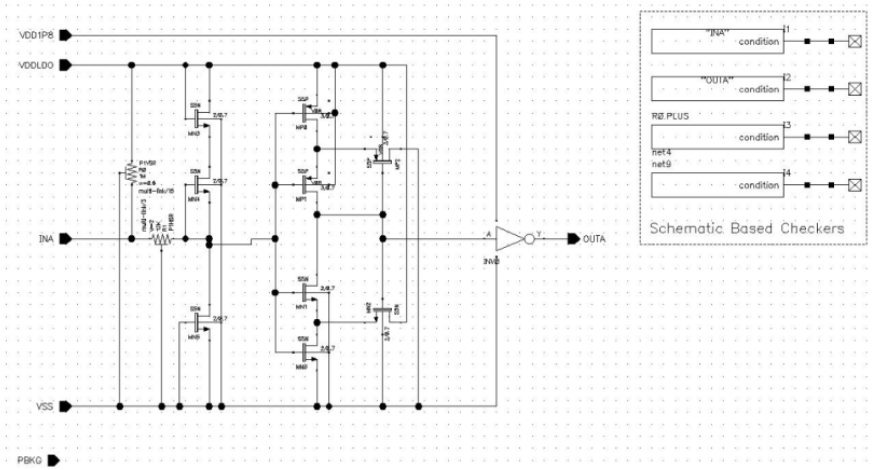
Where are we today?

- 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?

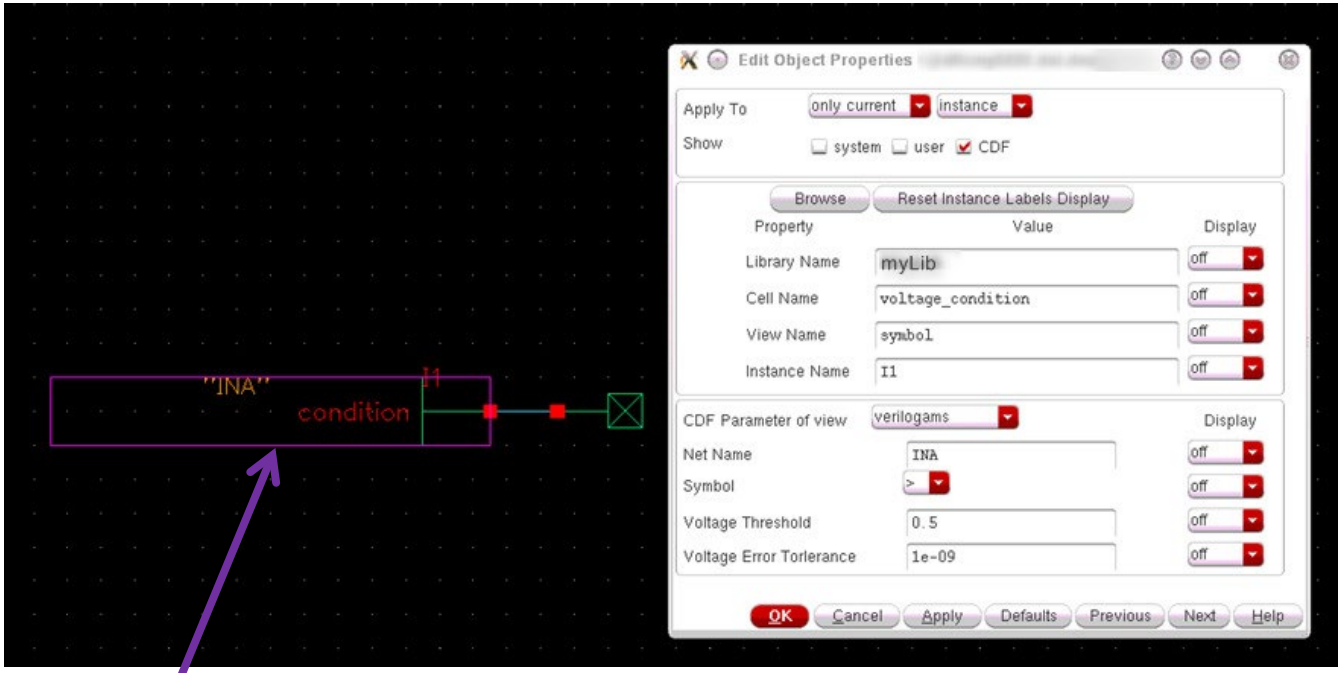
- The issues mentioned can be easily addressed if the analog designers *have an efficient way to pass the design knowledge to the verification engineers*



document

schematic

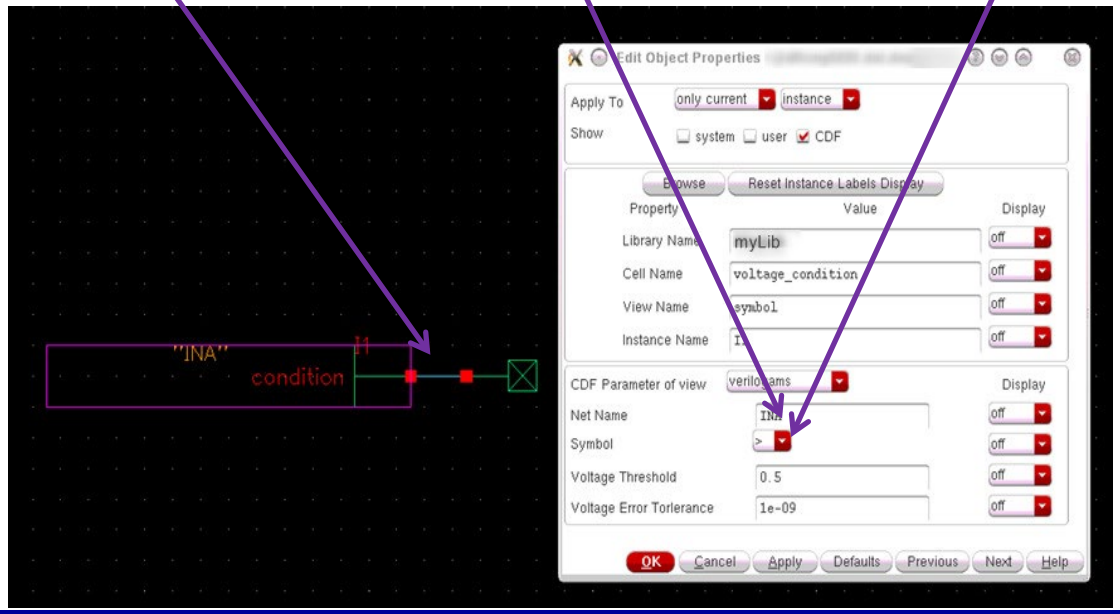
Analog Coverage Collector



- ACCs are implemented with three views
  - Symbol
  - VerilogAMS
  - VerilogA

Analog Coverage Collector (cont.)

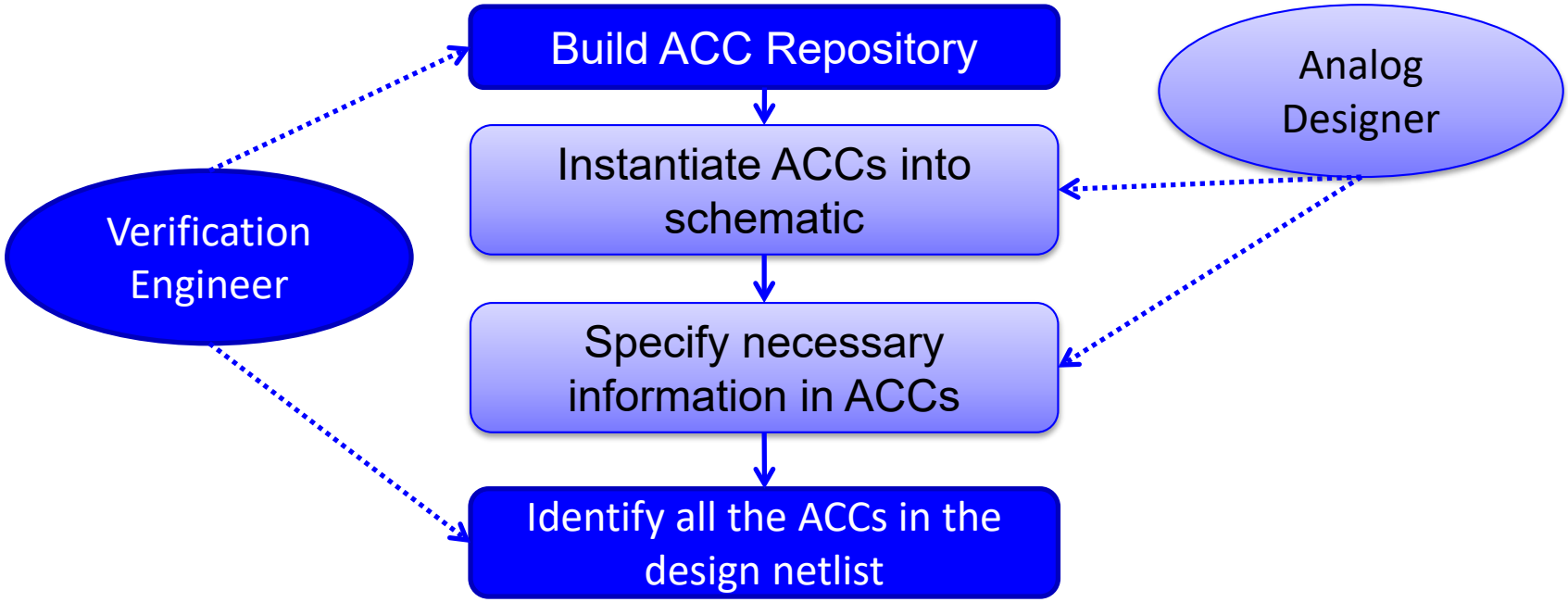
- Information like net name and checking conditions can be specified
- The output becomes high when a specified trigger condition is met



ACC- Implementation

```
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



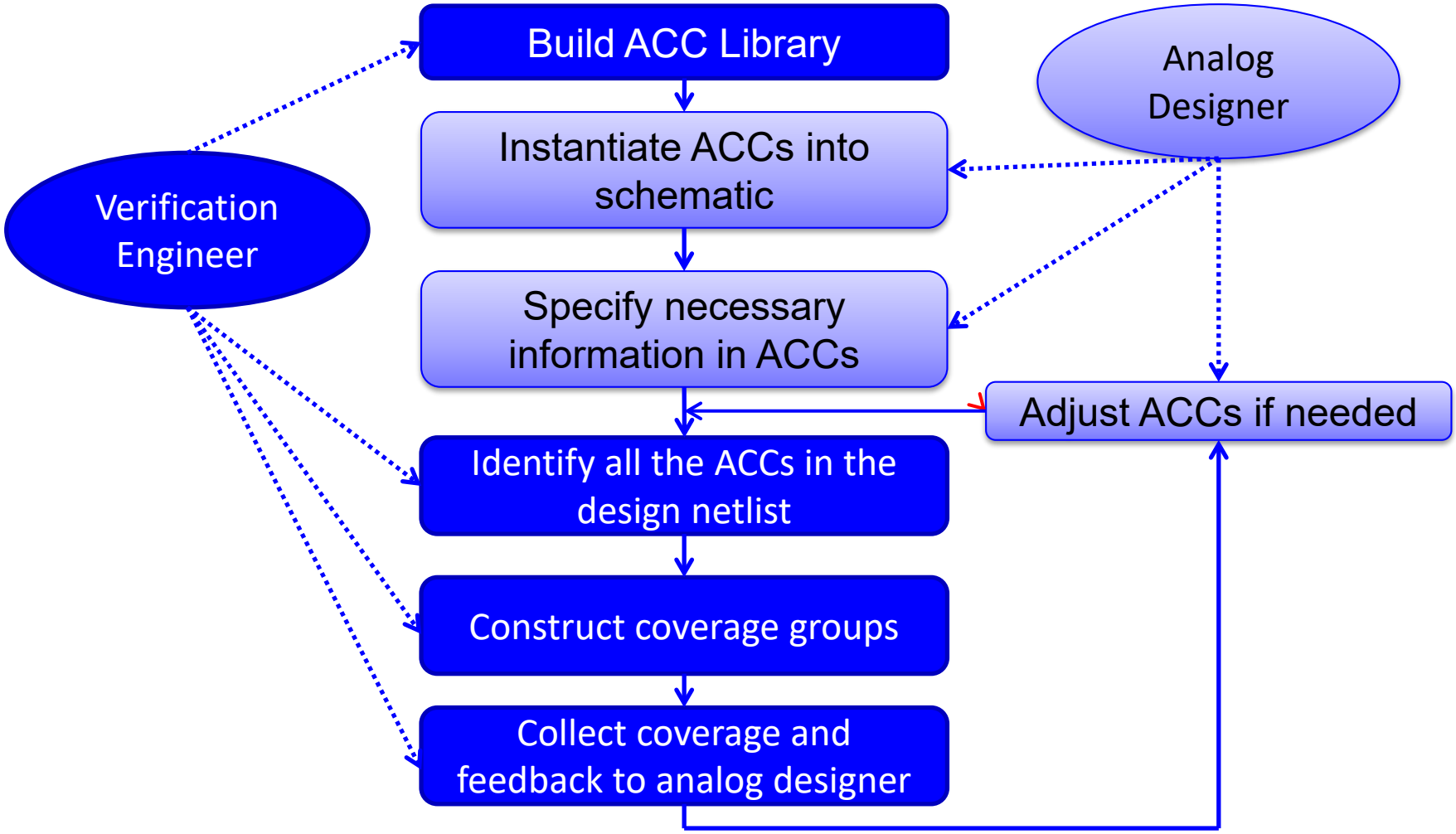
Identify ACCs from Netlist

```
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} }
```

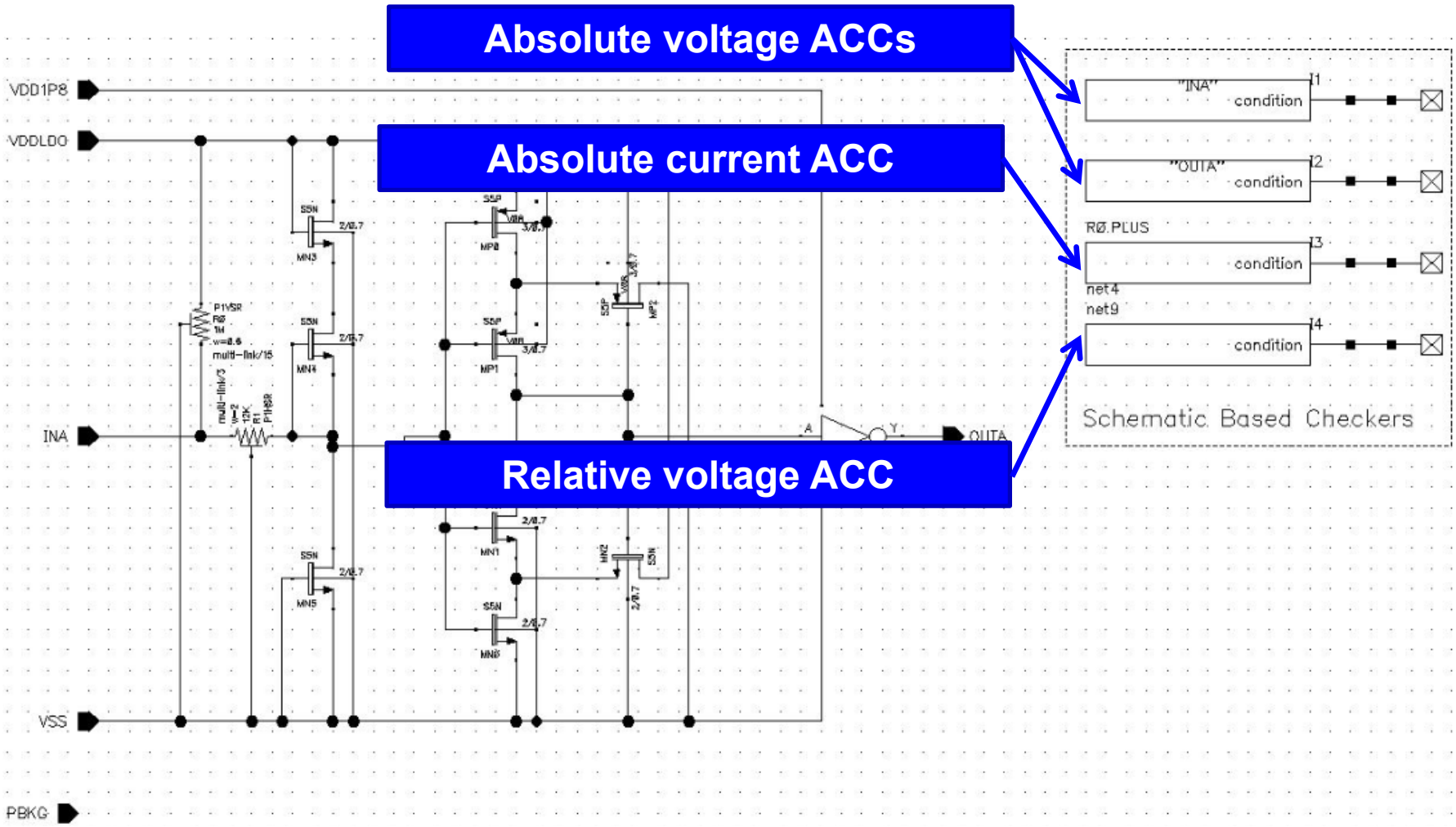
Result:

```
>./grep_acc.pl
LS.voltage_condition
LS.voltage_condition
LS.current_condition
LS.voltage_diff_condition
```

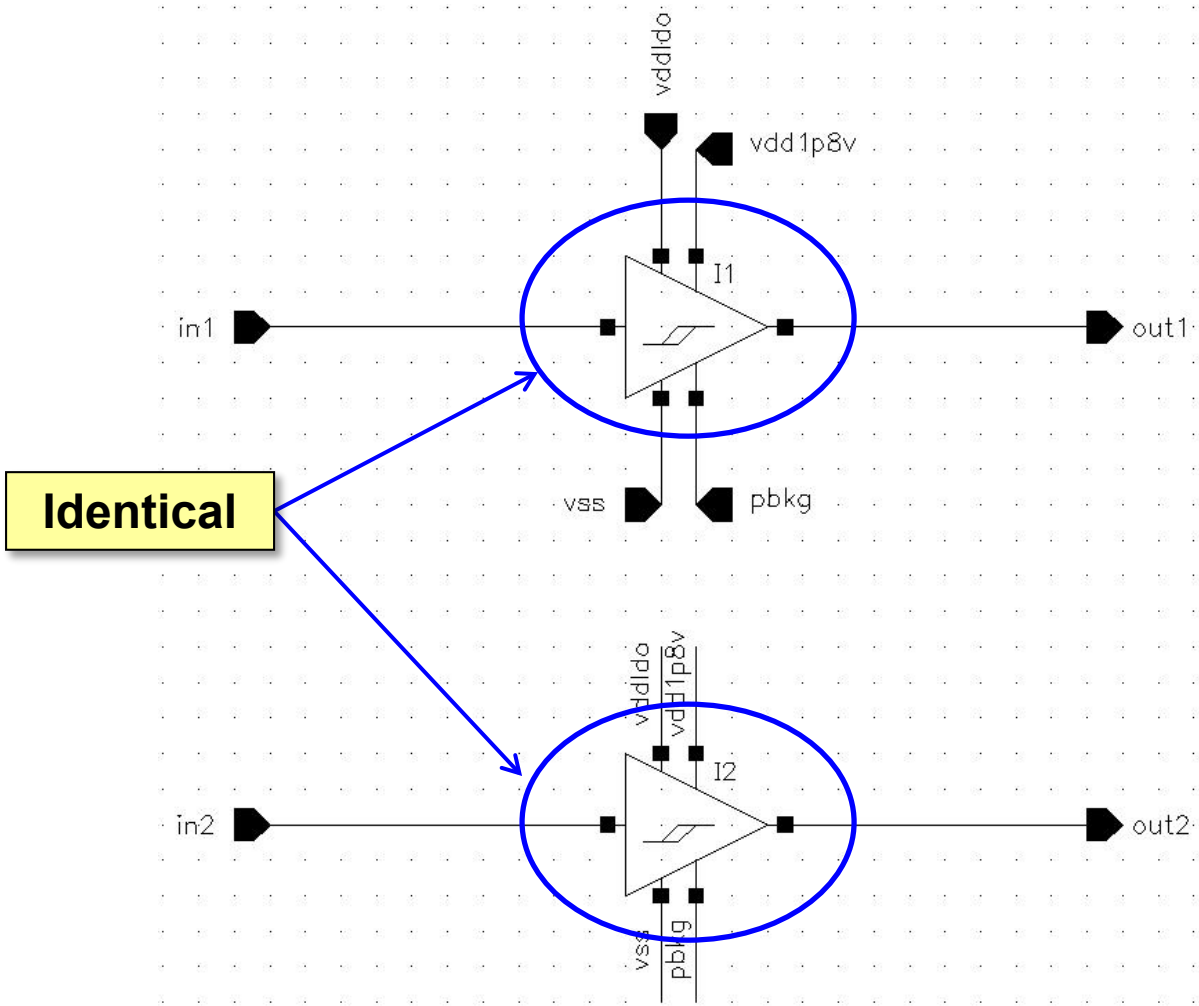
Analog Coverage Collection Flow



Example Circuit – Level Shifter



Example chip level Schematic



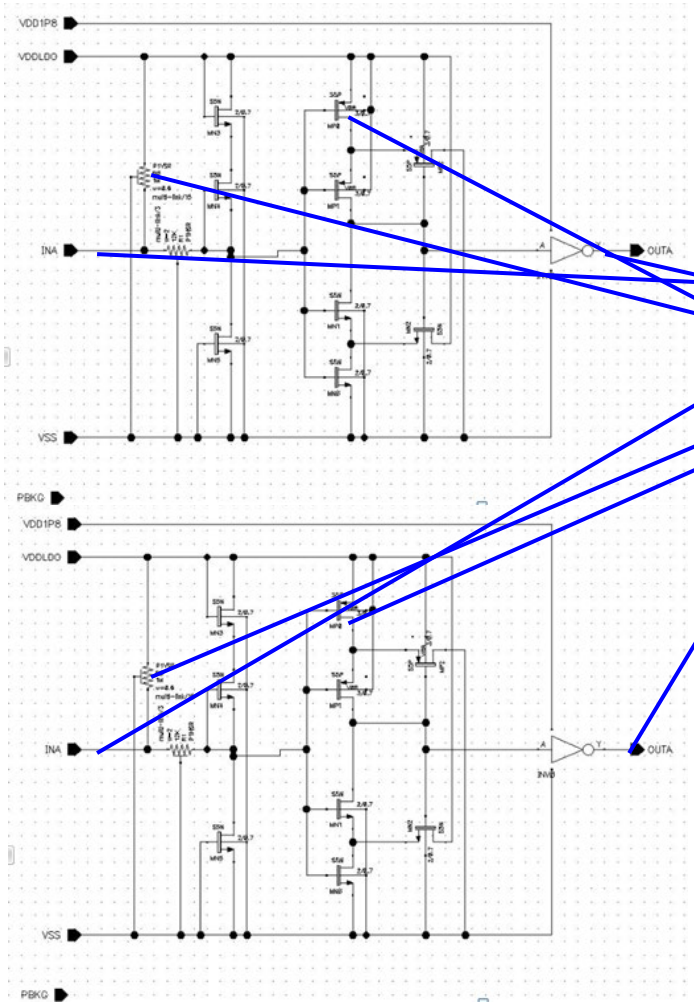
Sample Code of TB

```
module tb_sample_design();
electrical vddldo;
logic      c_in1;
logic      c_in2;
…
sample_design (*integer library_binding = "<library>"*)DESIGN
(.vddldo(vddldo),
.vdd1p8v(vdd1p8v),
…
);
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;
…
```

Sample Code of Covergroup

```
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% | Cov

Cover groups

Ex	UNR	Name	Overall Average Grade	Overall Covered
		(no filter)	(no filter)	(no filter)
		unmblk1.cg_in	100%	8 / 8 (100%)
		unmblk1.cg_out	100%	8 / 8 (100%)
		unmblk1.cg_r0	66.67%	5 / 8 (62.5%)
		unmblk1.cg_MP1	91.67%	7 / 8 (87.5%)

Showing 4 items

Items | unmblk1.cg_r0

Ex	UNR	Name	Overall Average Grade	Overall Covered
		(no filter)	(no filter)	(no filter)
		cov_i1	100%	2 / 2 (100%)
		cov_i2	50%	1 / 2 (50%)
		A×B cov_cross	50%	2 / 4 (50%)

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