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

Z. Ye, H. Lin and A. M. Khan
Texas Instruments
12500 TI Blvd,
Dallas, TX 75243

***Abstract*-In this paper, Analog Coverage Collector (ACC) is proposed to serve as a tool that analog designers can utilize to pass the analog design information, even when this information is deep inside the schematic hierarchy, to verification engineers who would handle chip level AMS/functional testbench for mixed signal designs. Based on that information, verification engineers are able to construct meaningful covergroups at the top level testbench in order to measure the chip-level functional coverage as accurately as possible. We show that analog designers can easily use ACC, and an example is presented to demonstrate the flow to collect analog coverage.**

I. Introduction

"When are we really done with verification?" This is one of the questions often asked by management to the verification engineers when it comes to scoping the completion percentage of an assigned task. In the meantime, more and more analog designers start asking verification engineers tough questions such as "Are my blocks being verified thoroughly at top level?" and "When will my block be verified 100%?". Without a proper coverage methodology, these questions become tough to answer.

Historically, there have been two kinds of verification coverage in digital domain, code coverage and functional coverage. Code coverage is one useful way to measure verification progress since it doesn't require additional manual work on part of the user, and is collected automatically by the simulation tool. Through this code coverage, we can measure how many lines of code have been executed (line coverage), which paths through the code and expressions have been executed (path coverage), which single-bit variable took the values 0 or 1 (toggle coverage), and which states and transitions in a state machine were exercised (FSM coverage) [1]. However, in the analog domain, this luxury is not available due to the schematic nature of analog circuits.

Functional coverage on the other hand measures what design features are covered through the test suites. Covergroups need to be constructed by verification engineers to collect functional coverage from simulation regression. The quality of covergroups directly impacts the accuracy of the collected coverage. For example, in a design with ten features if the covergroups are only built to cover six of the ten features, even if 100% functional coverage is achieved, the other four features may not work because of the fact that they may never get verified at all.

The concept of functional coverage is well known in the digital verification arena; however, it is relatively new to analog verification. This is mainly due to the complex nature of analog design and design implementation practices. For example, each device in the analog schematic may have its unique voltage or current thresholds. In additional, multiple thresholds may need to be checked for one device. Usually this kind of information is not openly documented or is innate to the designers' implementation. In addition, the coverage collection needs to be applied to selected devices to avoid information overload. Therefore usually it requires deep design knowledge or tedious information alignment sessions with the designer to gather this critical information. As a result, it is hard for the verification engineers to construct high quality covergroups at the top level testbench for analog circuits. On the other hand, there are few straight forward ways for the analog designers, who have full knowledge of their design, to pass this information to the verification engineers, which is a verification challenge.

In light of this challenge, there needs to be a flow or a method which is both design-friendly and verification-friendly that would allow analog designers to pass this essential information to verification engineers so that a complete set of covergroups can be built at top level testbench.

This paper addresses above challenge and provides a solution to enable seamless collaboration between design and verification. Following is a quick overview of the upcoming sections: In section II the concept of analog coverage is introduced and discussed. In section III ACC is presented as an efficient method that analog designers

can utilize to pass the knowledge needed to build analog covergroups for verification engineers. In section IV the analog coverage collection flow is discussed with section V demonstrating an example to show the power of ACC. Finally we draw some conclusions in Section VI.

II. Analog Coverage

The target of Analog coverage is to measure whether the voltage on a certain net or the current on a certain node has reached all the expected pre-defined thresholds. The concept of cross coverage applies when more than one signal needs to be sampled at the same time. Following four reasons make analog coverage collection complex and not trivial.

A. Identifying proper nets and nodes

Even in a relatively simple analog circuit there are many nets and nodes, therefore, design knowledge is needed to identify which nets or nodes need to be monitored in order to collect coverage. For example, in a level shifter circuit shown in Fig.1 there are eight nets and even more nodes in the circuit. Collecting coverage on each of these may not be necessary and we definitely want to identify all the critical ones.

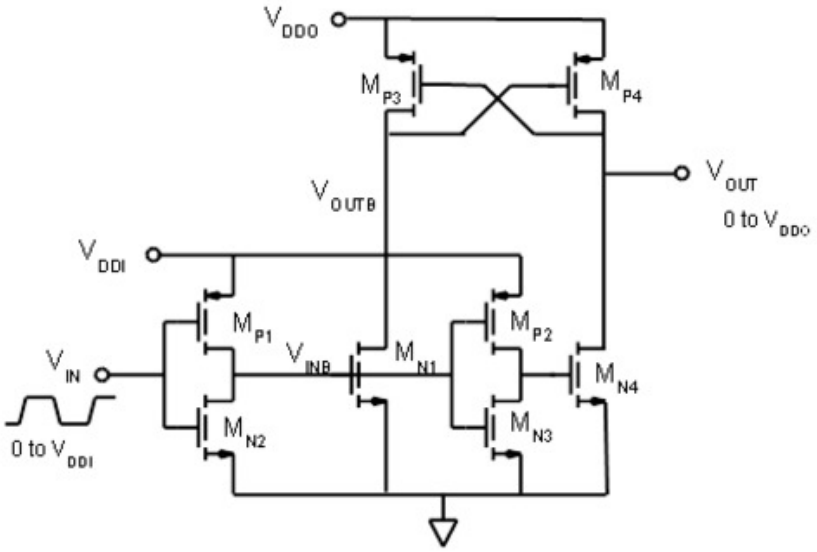

Fig. 1 Generic Level Shifter Circuit [2]

B. Determining proper threshold(s)

For digital signals the threshold is usually fixed because the expected values are often either "0" or "1". On the other hand in analog domain things are more complicated. For example, in order to fully exercise a particular circuit, the threshold for a voltage signal can be anywhere between ground and power supply, and sometimes even beyond the limitation of ground or power supply. In addition, usually more than one threshold may need to be defined for a certain net or node; therefore, smart decisions have to be made to determine proper thresholds for analog signals.

C. More than one signal may need to be monitored at the same time

Just as the concept of cross coverage for digital signals, more than one analog signal may need to be monitored simultaneously in order to create interesting covergroups. As an example, for the circuit shown in Fig.1, VDD1 and VDD0 may be combined to collect the coverage of power up sequencing.

D. Documentation about analog circuit design is often times lacking details

When writing covergroups for digital signals, verification engineers can reference to the register map, product specification and digital design specification, since most of the digital signals are function related and the standards of documenting digital signals are more mature. On the other hand, due to the nature of the analog circuits, the documentation is not in such detail since implementation specifics are mostly retained by designers at block level. One of the reasons is that historically analog designer is the verifier of the circuit he/she designed and there hasn't

been any standardized practice in place to share this information. Therefore it is hard for verification engineers to extract similar information that is needed to build proper covergroups.

All the four issues mentioned above can be easily addressed if the analog designers have an efficient way to pass the design knowledge to the verification engineers. Documenting everything is one way to do it. But we have found out that most of the analog designers prefer to work on their designs in schematic editor than documenting separately. In light of these, we propose a schematic way to transfer analog design knowledge from analog designers to verification engineers – a seamless and efficient approach of handling such information.

III. Analog Coverage Collector

All of the ACCs are implemented with three views, symbol, VerilogAMS and VerilogA. The source code is implemented in both VerilogAMS and VerilogA so that users can have options to choose from depending on where the analog circuit is being simulated, i.e. in pure analog block level or at top level mixed signal environment. The symbol view is for analog designers to place them into the schematics, just like any other cells that are instantiated in schematics.

The symbol and its object properties of an absolute voltage ACC are shown in Fig. 2 as an example below. By editing its CDF parameters, information such as net name, checking conditions, thresholds and error tolerance can be specified. The output "condition" becomes high when a specified trigger condition is met, which can then be utilized to collect coverage.

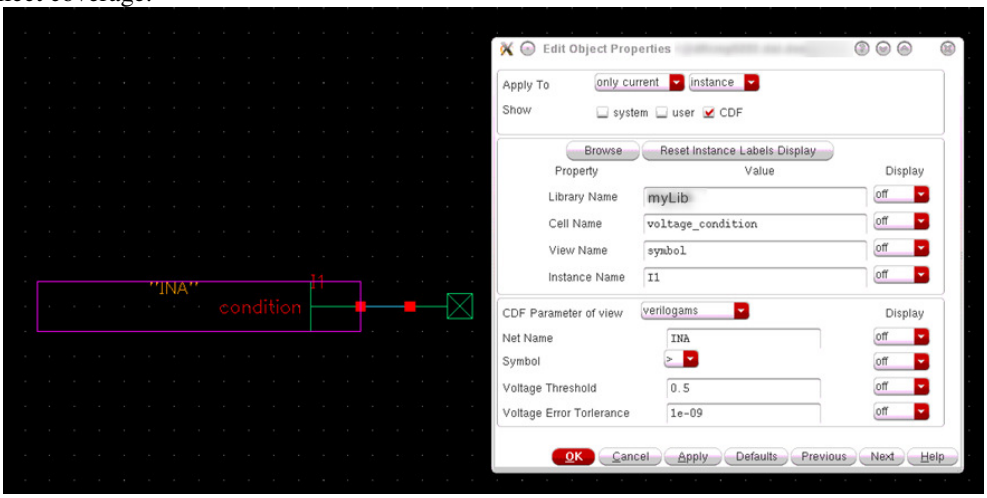

Fig.2 Absolute Voltage ACC along with its Properties

The pseudo code for the implementation of the absolute voltage ACC is shown in Table.1.

Table 1 Pseudo Code for Absolute Voltage ACC

```
module voltage_condition(condition);
  //*********************************************
   //Declare output
   //*********************************************
   output condition;
   logic   condition;
   //*********************************************
   //Declare user configurable CDF parameters
   //*********************************************
   parameter string vnet = "";
   parameter string symbol = "<" from '{"<", ">", "="};
   …
   //*********************************************
   //Initialization
   //*********************************************
```

```
    initial begin
      //Convert the format of hierarchical path
      H = …
     //Construct the full path of the signal to be monitored, H_full = H.vnet
     H_full = …;
     //determine the threshold checking direction from symbol entered by user
     if (symbol == “>”)
       dir = 1;
     else if …
       …
    end
    //*********************************************
    //Main Program to Calculate output “condition”
    //*********************************************
    always @ (above <threshold 1>)
     cond1 = 1’b1;
    end
    always @ (above <threshold 2>)
     cond1 = 1’b0;
    //other always blocks to calculate cond2 and cond3
    …
    //*********************************************
    //Construct output “condition”
    //*********************************************
    assign condition = <combination of cond1, cond2 and cond3>.
    //*********************************************
    //Obtain analog value of the monitored signal
    //*********************************************
    analog begin
     v_val = $cds_get_analog_value(H_full, “potential”);
    end
endmodule
```

Similarly a repository of ACCs including current ACCs, relative voltage ACCs etc. can be implemented also.

IV. Analog Coverage Collection Flow

With a repository of coverage collectors available, the analog designers can then instantiate them in the design schematics by pointing them to the signals of interest and specifying correct threshold voltage and other pertinent information. A flow chart describing this process is shown in Fig. 3 below indicating seamless collaboration between analog designers and verification engineers thereby allowing robust execution.

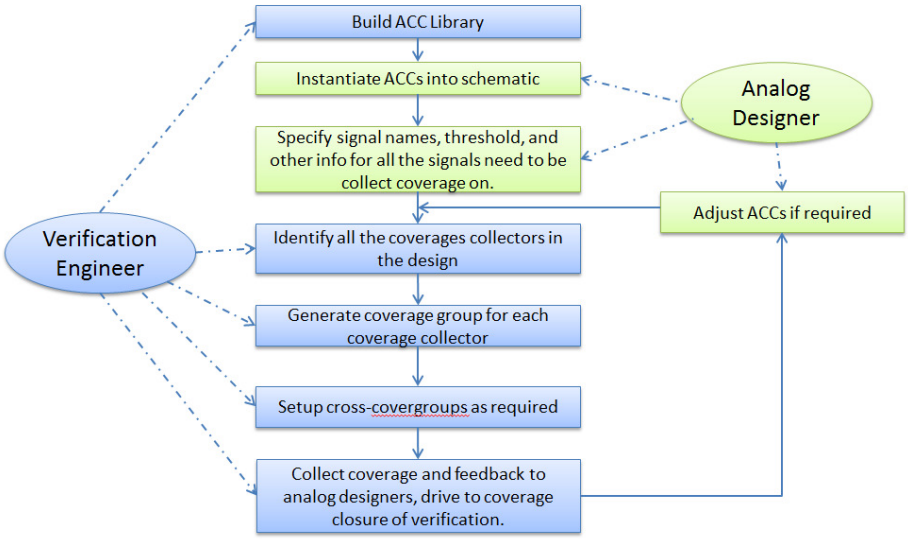

Fig.3 Analog Coverage Collection Flow

Once this flow is setup, it benefits both design and verification engineers. From analog designer's point of view, the question of "whether my design block is being verified exhaustively at top level" can be answered. If the answer is yes, he/she would be confident that the block would work at chip level in terms of functionality. From verification engineer's point of view, comprehensive covergroups for analog blocks can be easily established with the help of ACCs. If the coverage result is less than expected, more interesting stimulus need to be identified and added.

V. Example

An example level shifter design along with the ACCs shown in Fig.4, where I1 and I2 are the absolute voltage ACCs, I3 is the absolute current ACC and I4 is the relative voltage ACC.

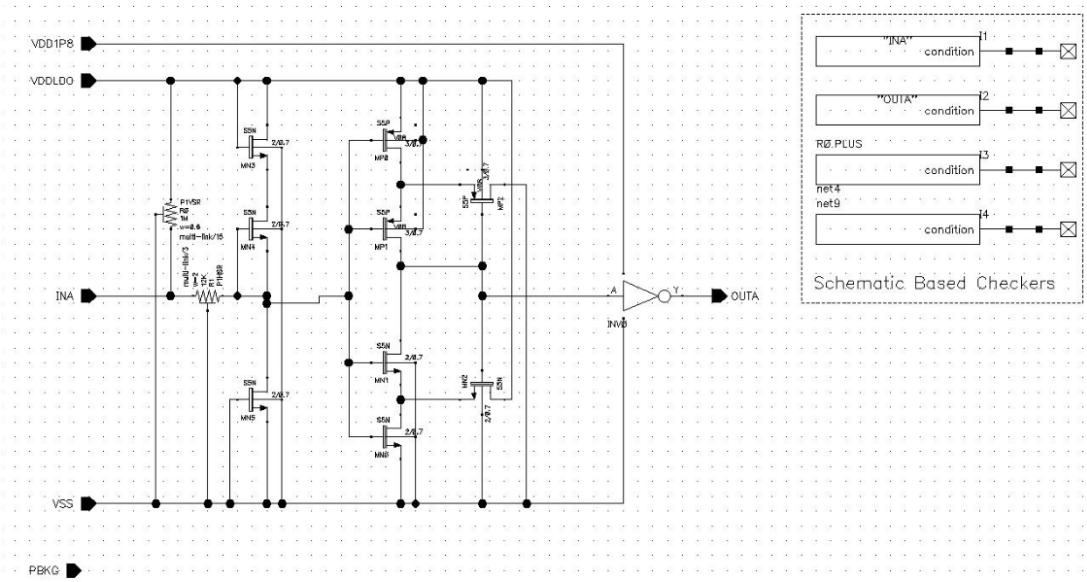

Fig.4 Example Design along with Coverage Collectors

In the top level schematic, two level shifters are instantiated, as shown in Fig. 5. As a result, cross-coverage across these two cells can be monitored, if required.

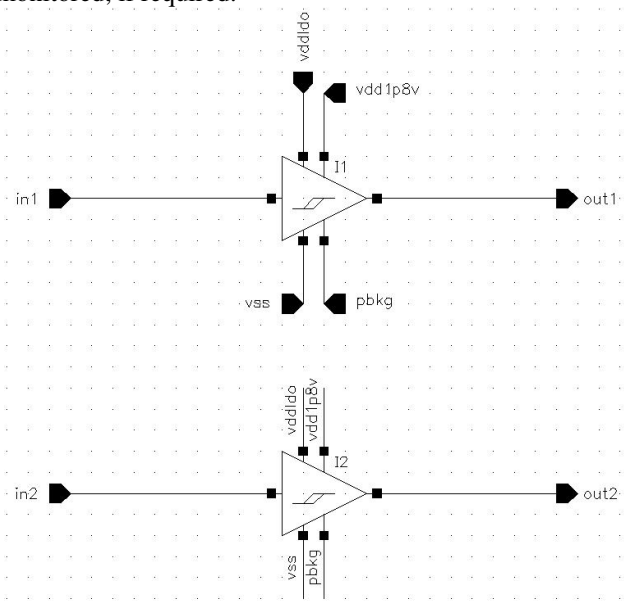

Fig.5 Example Top Level Design

Partial code for the Verilog-AMS based top level testbench is shown in Table 2. In the testbench, Design Under Test (DUT), covergroup and other drivers/monitors are instantiated.

Table 2 Partial Code For Testbench in VerilogAMS

```
module tb_sample_design();
  //***********************************************
   //Declare signals
   //***********************************************
   electrical vddldo;
   electrical vdd1p8v;
   …
   logic       c_in1;
   logic       c_in2;
   …
  //***********************************************
   //Instantiate Design
   //***********************************************
   sample_design (*integer library_binding = "<library>"*) DESIGN
     (.vddldo(vddldo),
      .vdd1p8v(vdd1p8v),
       …
     );
   //***********************************************
   //Instantiate Covergroup
   //***********************************************
   sample_cov (*integer library_binding = "<library>"*) _covergroup
     (.in1(c_in1),
      .in2(c_in2),
       …
     );
  //***********************************************
```

```
    //Obtain coverage signals
    //*******************************************
    assign c_in1 = DESIGN.I1.I1.condition;
    assign c_in2 = DESIGN.I1.I2.condition;
    …
   //*******************************************
    //Instantiate other driver/monitors
    //*******************************************
    …
endmodule
```

Since ACCs are embedded within the schematic, a script can be written to grab all the ACCs in design from the top level netlist for large designs. Once all the ACCs are identified, covergroups can be written around the ACCs to collect analog coverage. An example code for covergroup implementation is shown in Table 3.

Table 3 Partial Code for Covergroup in SystemVerilog

```
module sample_cov(in1, in2, …);
    //*******************************************
    //declare inputs
    //*******************************************
    input in1;
    input in2;
    …
    //*******************************************
    //Construct Covergroups
    //*******************************************
    covergroup sample_cg_in;
       option.per_instance = 1;
       cov_in1 : coverpoint in1;
       cov_in2 : coverpoint in2;
       cov_cross: cross cov_in1, cov_in2;
    endgroup
    …
    //*******************************************
    //Declare the covergroups and sample them
    //*******************************************
    initial begin
    sample_cg_in cg_in;
    ...
    cg_in = new();
    …
forever begin
    fork
       begin
          @(in1 or in2)
          cg_in.sample();
       end
    …
    join
end
endmodule
```

An example analog verification coverage result is shown in Fig.6 using IMC tool from Cadence. It can be seen that we can easily identify coverage holes in the target test suite. For example, in covergroup cg_r0, coverpoint cov_i2 is not fully exercised yet.

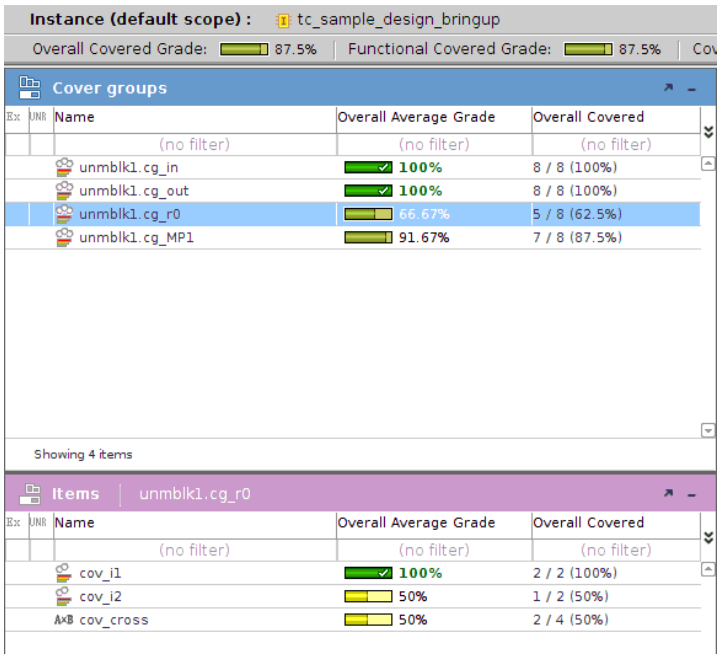

Fig.6 Example Coverage Results

VI. CONCLUSIONS

In this paper, ACC is proposed as an efficient method to pass information from analog designers to verification engineers. As a result, meaningful covergroups can be constructed for collecting the coverage information of analog circuits. Another benefit for the ACC is that since it stays with the design schematic, it is reusable and portable without any overhead of environment management.

REFERENCES

[1] C. Spear and G. Tumbush, "*SystemVerilog for Verification*", Springer, 2012.
[2] https://wiki.analog.com/university/courses/electronics/electronics-lab-voltage-level-shifter