2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
OCTOBER 14-15, 2025

A UVM Testbench for Exploring Design Margins of Analog/Mixed-Signal Circuits: A PCI-Express Receiver Detection Circuit Example

Jaeha Kim, Seoul National University, Seoul, Korea (jaeha@snu.ac.kr)

***Abstract*— This paper presents a UVM testbench for characterizing the design margins of analog/mixed-signal (AMS) circuits by finding the worst-case deviation of the circuit's response across a continuous-valued parameter space. The testbench combines a reactive stimulus technique with a Bayesian optimization algorithm to efficiently and adaptively explore the parameter space. Using a PCI Express receiver detection circuit as a case study, of which analog components are modeled in SystemVerilog with XMODEL primitives, the paper demonstrates how the testbench can identify design points that maximize margin and assess their sensitivity to secondary operating conditions. This approach enables adaptive, coverage-driven AMS verification, supporting more automated and scalable margin analysis in complex mixed-signal systems.**

Keywords—analog/mixed-signal verification; universal verification methodology (UVM); SystemVerilog; XMODEL; design margin analysis; Bayesian optimization.

I. INTRODUCTION

Often, the functionality of analog/mixed-signal (AMS) circuits must be verified over a continuous range of operating conditions to ensure that the circuit has sufficient design margins and can maintain robust operation even when the condition changes. This paper presents a UVM testbench for characterizing the design margins of an AMS circuit using reactive stimulus generation and optimization techniques to efficiently explore the condition parameter space.

As a concrete example, we target a PCI-Express (PCIe) receiver detection circuit shown in Figure 1. This circuit determines whether a receiver is present or absent on the other end of an AC-coupled channel, and operates by forming a relaxation oscillator, measuring its oscillation period in digital counts (N_p), and comparing the result against a predetermined threshold value ($N_{p,thres}$). Since the value of N_p also changes with the operating conditions of the circuit, including the changes in the AC coupling capacitance (C_c), equivalent channel capacitance (C_{ch}), and transmitter/receiver termination resistances ($R_{term,tx}$ and $R_{term,rx}$), it is desirable to maximize the separation

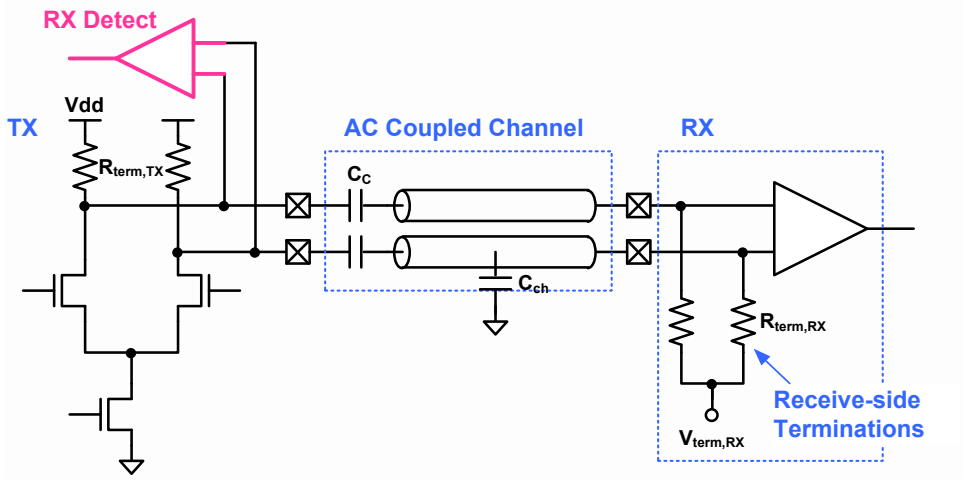

Figure 1. A PCI-Express transmitter-receiver configuration with the condition parameters shown: AC coupling capacitance (C_c), equivalent channel capacitance (C_{ch}), and transmitter- and receiver-side termination resistances ($R_{term,TX}$ & $R_{term,RX}$).

between the minimum value of N_p observed when the receiver is present (*min* ($N_{p,w/RX}$)) and the maximum value when the receiver is absent (*max*($N_{p,w/oRX}$)), and to choose a threshold value $N_{p,thres}$ that satisfies the inequality *min*($N_{p,w/RX}$) > $N_{p,thres}$ > *max*($N_{p,w/oRX}$) across a wide range of other condition parameters, such as the reference voltages (V_{refL} and V_{refH}).

This paper presents a UVM testbench for measuring *min*($N_{p,w/RX}$) and *max*($N_{p,w/oRX}$) of a PCIe receiver detection circuit modeled in SystemVerilog, across specified ranges of the condition parameters, C_c, C_{ch}, $R_{term,tx}$, and $R_{term,rx}$. Instead of exhaustively sweeping over all possible parameter combinations, the testbench leverages a reactive stimulus technique in conjunction with a Bayesian optimization algorithm to efficiently locate the point that yields the minimum or maximum N_p within the defined parameter space.

II. Receiver Detection Circuit for PCI-Express Transmitter

A PCIe transmitter must determine whether a receiver is connected before initiating data transmission [1]. Figure 1 illustrates a typical configuration, where a PCIe transmitter is connected to a PCIe receiver via an AC-coupled channel. The transmitter employs a differential current-mode driver, and both the transmitter and receiver sides include termination resistors to minimize signal reflections caused by impedance discontinuities. The PCIe specification requires that the receiver detection circuit correctly identify the presence or absence of a receiver for a range of condition parameters: AC coupling capacitance (C_c) between 76-265 nF, equivalent channel capacitance (C_{ch}) up to 3 nF, and transmitter- and receiver-side termination resistances ($R_{term,TX}$ and $R_{term,RX}$) each ranging within 40-60 Ω.

One example of the receiver detection circuit operates by transforming the transmitter into a relaxation oscillator [2], as illustrated in Figure 2. The circuit turns on the transmitter's pull-down currents when the common-mode voltage (V_{cm}) of the transmitter outputs exceeds a high reference threshold ($V_{ref,H}$), and turns them

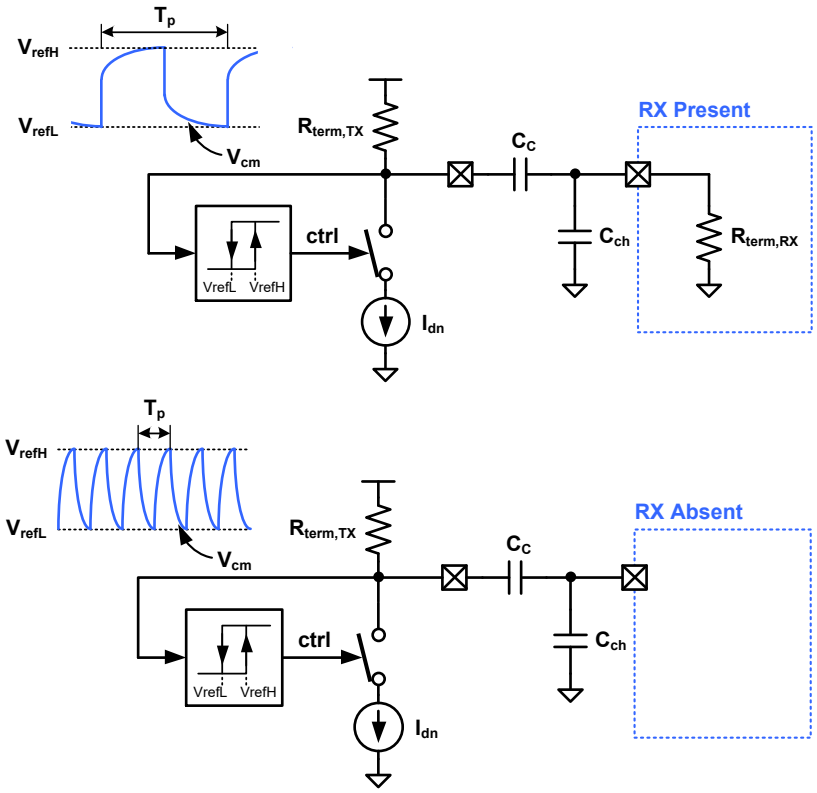

Figure 2. Operation of the PCI-Express receiver detection circuit when the receiver is present (top) and absent (bottom).

off when V_{cm} falls below a low reference threshold ($V_{ref,L}$). This produces a periodic waveform of V_{cm} oscillating between $V_{ref,L}$ and $V_{ref,H}$, and the period of this oscillation can serve as an indicator of receiver presence. When the receiver is present, the setting time constant (τ) is longer due to the termination resistors on both ends: $\tau \approx (R_{term,TX}+R_{term,RX})\cdot C_c$. On the other hand, when the receiver is absent, the time constant is shorter: $\tau \approx (R_{term,TX})\cdot C_{ch}$.

Figure 3 illustrates one possible implementation of this receiver detection circuit. The design forms a mixed-signal feedback loop consisting of an output common-mode (V_{cm}) sensor, a 3-level quantizer, a digital controller, and a pull-down current (I_{dn}) driver. The pull-down currents drawn from the transmitter's outputs are controlled by the 2-bit digital output of the 3-level quantizer (comp[1:0]), which compares their common-mode voltage V_{cm} against two reference voltages, $V_{ref,H}$ and $V_{ref,L}$. Specifically, the pull-down currents are enabled when V_{cm} rises above $V_{ref,H}$ and disabled when V_{cm} falls below $V_{ref,L}$. This is basically the operation of a relaxation oscillator, producing periodic oscillations in both the analog voltages (e.g. V_{cm}) and the digital outputs (e.g. comp[1:0]). The digital controller counts the oscillation period (N_p) using a 100MHz external clock and compares the result against a pre-defined threshold ($N_{p,thres}$) to determine the receiver's presence.

Achieving robust operation of this receiver detection circuit requires careful tuning of the threshold values ($V_{ref,H}$, $V_{ref,L}$, and $N_{p,thres}$). It is because the measured value of N_p depends not only on the presence or absence of the receiver, but also on the values of C_c, C_{ch}, $R_{term,tx}$, and $R_{term,rx}$. To ensure reliable detection, it is desirable to maximize the separation between the minimum value of N_p observed when the receiver is present (*min* ($N_{p,w/RX}$)) and the maximum value when the receiver is absent (*max*($N_{p,w/oRX}$)), and to choose a threshold value $N_{p,thres}$ that satisfies the inequality *min*($N_{p,w/RX}$) > $N_{p,thres}$ > *max*($N_{p,w/oRX}$) against possible variations in $V_{ref,H}$ and $V_{ref,L}$.

The analog components of the receiver detection circuit are modeled in SystemVerilog using primitives provided by XMODEL from Scientific Analog [3]. XMODEL enables the modeling of analog behavior with efficient, event-driven simulation semantics, fully integrated within the SystemVerilog environment. It supports both signal-flow modeling (e.g., using compare, transition, and and_xbit primitives) and conservative-system modeling (e.g., using isource and switch primitives).

Figure 4 lists the models for the output common-mode sensor (*sens_vcm*), 3-level quantizer (*comp_lev3*), and pull-down current driver (*drv_icm*). Although these models may appear straightforward, their simplicity is a key advantage of XMODEL. In contrast, describing the same circuit using Real-Number Model (RNM) would be significantly more complex, primarily as the circuit draws currents from the transmitter output nodes, which are shared between the main PCIe transmitter circuit and the receiver detection circuit. Capturing this behavior in RNM requires the use of user-defined nets and resolution functions, whereas XMODEL handles it naturally and efficiently.

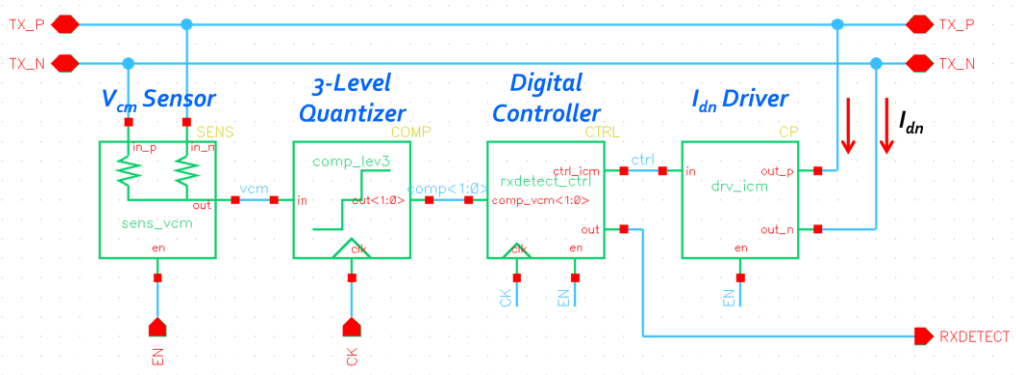

Figure 3. The block diagram of the PCI-Express receiver detection circuit modeled with XMODEL primitives.

```
module sens_vcm (
   output xreal out,
   input xreal in_p, in_n,
   input xbit en
);

xreal p0, n0;
switch       #(.R0(`INFINITY), .R1(10.0e3)) SW1 (.pos(p0), .neg(out), .ctrl(en));
switch       #(.R0(`INFINITY), .R1(10.0e3)) SW2 (.pos(n0), .neg(out), .ctrl(en));

endmodule    // sens_vcm


module comp_lev3 #(
    parameter real VrefH = 1.75,
    parameter real VrefL = 1.35
)(
    output xbit [1:0] out,
    input xreal in,
    input xbit clk
);

xreal vref_H, vref_L;
compare      #(.threshold(VrefL)) COMP0 (.out(out[0]), .in(in), .in_ref(`ground), .trig(clk),);
compare      #(.threshold(VrefH)) COMP1 (.out(out[1]), .in(in), .in_ref(`ground), .trig(clk),);

endmodule    // comp_lev3


module drv_icm #(
    parameter real Idn = 0.0125
)(
    output xreal out_p, out_n,
    input xbit in, en
);

xbit ctrl;
and_xbit     U0 (.out(ctrl), .in({in,en}));
transition   #(.value0(0.0), .value1(Idn)) U1 (.out(i_dn), .in(ctrl));
isource      #(.mode("in")) IP (.pos(out_p), .neg(`ground), .in(i_dn));
isource      #(.mode("in")) IN (.pos(out_n), .neg(`ground), .in(i_dn));

endmodule    // drv_icm
```

Figure 4. SystemVerilog models of the PCIe receiver detection circuit's analog components using XMODEL primitives: the output common-mode sensor (*sens_vcm*), 3-level quantizer (*comp_lev3*), and pull-down current driver (*drv_icm*).

III. UVM TESTBENCH FOR DESIGN MARGIN ANALYSIS

Our objective is to develop a UVM testbench that verifies whether the circuit functions correctly over the full specification space of C_c, C_{ch}, $R_{term,TX}$, and $R_{term,RX}$ and assess its sensitivity to the variations in $V_{ref,H}$, $V_{ref,L}$, and $N_{p,thres}$.

Figure 5 shows the organization of the proposed UVM testbench. Building on the approaches described in [4]–[6], the testbench encapsulates all the analog-specific details in a fixture module, enabling the rest of the testbench to be constructed using standard UVM components. The fixture module includes the full setup for evaluating the receiver detection circuit: transmitter drivers, AC-coupled channels, and configurations with and without receiver-side terminations. All the condition parameters (C_c, C_{ch}, $R_{term,TX}$, and $R_{term,RX}$) are implemented as signals, not as parameters in SystemVerilog, so that their values can vary during the course of simulation.

The proposed UVM testbench follows a similar organization to the one described in [5], where the driver agent also serves as a monitor agent by employing a reactive stimulus technique [7]—selecting the next stimulus based on prior outcomes. Instead of exhaustively sweeping through a fixed set of condition parameter values, the testbench adaptively searches for parameter combinations that either maximize or minimize the oscillation period count (N_p), leveraging a Bayesian optimization algorithm [8]-[9]. A Bayesian optimizer constructs a surrogate model using the results of N_p obtained from previously evaluated parameter points. It then uses this surrogate model to predict the expected value of N_p and the associated confidence level at unexplored points in the parameter space.

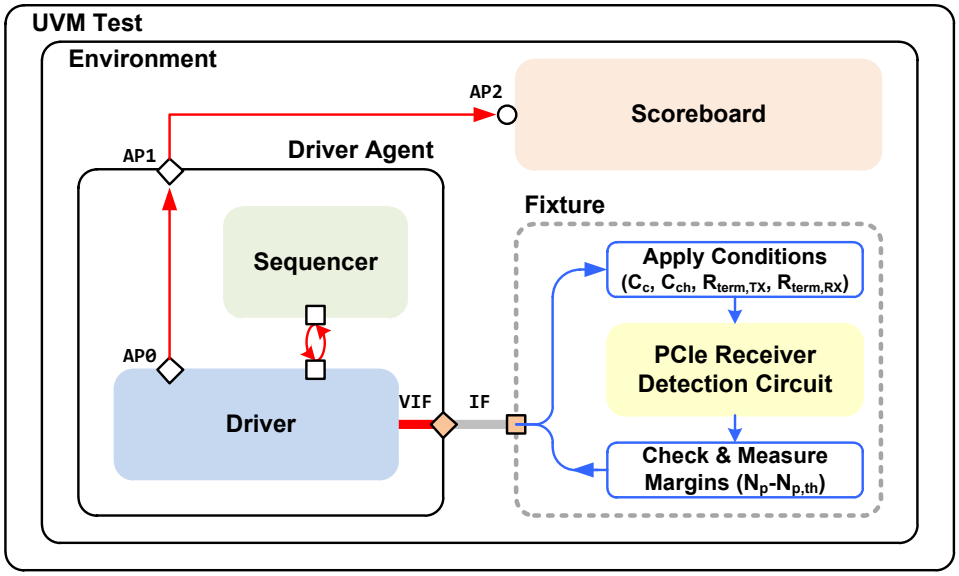

Figure 5. Overview of the UVM testbench that uses reactive stimulus and Bayesian optimization techniques to verify circuits over a continuous space of condition parameters and measure design margins.

Based on these predictions, the optimizer suggests the next parameter point to evaluate, thereby increasing the likelihood of discovering the maximum or minimum N_p within the specified ranges. As previously discussed, the measured minimum value of N_p observed when the receiver is present (*min* ($N_{p,w/RX}$)) and maximum value when the receiver is absent (*max*($N_{p,w/oRX}$)) can be used to assess the design margin of the receiver detection circuit as well as its sensitivity to the variations in $V_{ref,H}$ and $V_{ref,L}$.

The following subsections provide detailed descriptions of each component within this UVM testbench.

A. Sequencer Component

The sequencer component is the key element in this UVM testbench, as it performs the iterative search to find the minimum N_p with receiver present, *min*($N_{p,w/RX}$), and the maximum N_p with receiver absent, *max*($N_{p,w/oRX}$), using the reactive stimulus [7] and Bayesian optimization techniques [8]. Figure 6 lists the simplified code of the sequencer component.

A set of SystemVerilog DPI functions performing each step of the Bayesian optimization are defined using the *BayesOpt* library [8]:

- `BAYESOPT_initialize`(*num_vars*, *lower_bounds*, *upper_bounds*)
  : initializes a *BayesOpt* data model with the number of variables and their lower/upper bounds.
- `BAYESOPT_selectNext`(*opt_data*, *var_values*)
  : select the next trial point of variables in search of the minimum value.
- `BAYESOPT_updateModel`(*opt_data*, *var_values*, *result*)
  : updates the *BayesOpt* data model with the result obtained for a given trial point.
- `BAYESOPT_getOptimum`(*opt_data*, *var_values*)
  : estimates the optimum point of variables using the *BayesOpt* t data model.

Using these DPI functions, the sequencer component initiates a sequence of N_p measurement tests, first to determine the minimum N_p with the receiver present and second to determine the maximum N_p with the receiver absent. In each iteration, the sequencer calls *BAYESOPT_selectNext*() function to select the next set of condition parameters to explore. These parameter values are then sent to the fixture module through the driver component

using the standard UVM methods: *start_item*() and *finish_item*(). The sequencer subsequently waits for a response from the fixture module via the driver, using the *get_response*() method. Upon receiving the response packet *RSP* containing the measured N_p corresponding to the condition parameters, it updates the surrogate data model by calling the *BAYESOPT_updateModel*() function. After completing a predefined number of iterations (e.g. 200), the *BAYESOPT_getOptim*um() function is used to predict the set of condition parameter values that is most likely to yield the optimum result.

```
class SEQ_MARGIN extends uvm_sequence #(PACKET);
    `uvm_object_utils(SEQ_MARGIN)

    // ... some details omitted for brevity ...

    int num_iter = 200;         // number of iterations
    PACKET PKT, RSP;            // packets exchanged with fixture module

    task body();
        PKT = PACKET::type_id::create("PKT");
        lower_bounds = '{75e-9, 0.0, 40, 40};
        upper_bounds = '{265e-9, 3e-9, 60, 60};
        opt_data = BAYESOPT_initialize(4, lower_bounds, upper_bounds);

        // receiver present
        PKT.RX_present = 1;
        for (int i=0; i<=num_iter; i++) begin:LOOP1
            if (i < num_iter) BAYESOPT_selectNext(opt_data, var_value);
            else BAYESOPT_getOptimum(opt_data, var_value);
            PKT.Cc = var_value[0];
            PKT.Cch = var_value[1];
            PKT.Rterm_tx = var_value[2];
            PKT.Rterm_rx = var_value[3];

            start_item(PKT);
            finish_item(PKT);
            get_response(RSP);

            // find minimum Np when receiver is present
            BAYESOPT_updateModel(opt_data, var_value, RSP.Np);
        end: LOOP1

        // receiver absent
        PKT.RX_present = 0;
        for (int i=0; i<num_iter; i++) begin:LOOP2
            if (i < num_iter) BAYESOPT_selectNext(opt_data, var_value);
            else BAYESOPT_getOptimum(opt_data, var_value);
            PKT.Cc = var_value[0];
            PKT.Cch = var_value[1];
            PKT.Rterm_tx = var_value[2];
            PKT.Rterm_rx = var_value[3];

            start_item(PKT);
            finish_item(PKT);
            get_response(RSP);

            // find maximum Np when receiver is absent
            BAYESOPT_updateModel(opt_data, var_value, -RSP.Np);
        end: LOOP2
    endtask: body

endclass: SEQ_MARGIN
```

Figure 6. The sequencer component searching for the minimum or maximum N_p across the parameter space using reactive stimulus and Bayesian optimization techniques.

B. Fixture Module

The fixture module instantiates the PCIe receiver detection circuit model and contains the necessary test setup for measuring the oscillation period count N_p with or without the receiver present for a given set of condition parameters. Figure 7 lists the code of the fixture module. Note that it is the only module that contains the XMODEL primitives.

The fixture module communicates with the driver component via a handshake protocol using *START* and *DONE* signals, enabling multiple N_p measurements within a single simulation run. When the driver intends to initiate a new N_p measurement for a set of C_c, C_{ch}, $R_{term,tx}$, and $R_{term,rx}$ values, it asserts *START* to 1. In response, the fixture module executes the test by configuring the simulation environment with the specified parameter values, applying them to the coupling capacitors, equivalent channel capacitors, and transmitter/receiver termination resistors. It also supplies a 100MHz clock (*CK*) and a 1.8V supply voltage (*vdd*). The fixture module initially waits for a period of time prescribed by *t_init* before enabling the receiver detection circuit. It then allows for the N_p measurement to proceed, subject to a time-out window specified by *t_meas*. Upon completion, the fixture module sends the measured N_p value back to the driver and asserts *DONE* to 1 to indicate the completion.

```
module FIXTURE (IF_t IF);
    // ... parameter definitions and signal declarations omitted for brevity ...

    // DUT model for the PCIe receiver detection circuit
    rxdetect    #(.VrefH(VrefH), .VrefL(VrefL), .Np_thres(1024))
                RXDET (.CK(CK), .TX_P(TX_P), .TX_N(TX_N), .EN(EN), .RXDETECT(RXDETECT));

    // transmitter and AC-coupled channels
    tx_pcie     TX (.out_p(TX_P), .out_n(TX_N), .vdd(VDD), .Rterm(Rterm_tx), .Cp(Cp_tx));
    channel     CHN_P (.port_1(CH_P), .port_2(RX_P), .Cch(Cch));
    channel     CHN_N (.port_1(CH_N), .port_2(RX_N), .Cch(Cch));
    cap_sw      CP (.pos(TX_P), .neg(CH_P), .C(Cc));
    cap_sw      CN (.pos(TX_N), .neg(CH_N), .C(Cc));

    // add receiver depending on RX_present
    bit_to_xbit RX_CONN (.in(RX_present), .out(RX_EN));
    switch      SW0 (.pos(RX_P), .neg(RX_P0), .ctrl(RX_EN))
    switch      SW1 (.pos(RX_N), .neg(RX_N0), .ctrl(RX_EN))
    rx_pcie     RX (.in_p(RX_P0), .in_n(RX_N0), .Rterm(Rterm_rx), .Cp(Cp_rx));

    // sources feeding clock and supply voltage
    clk_gen     #(.freq(f_clk)) XP0 (.out(CK0));
    xbit_to_bit CK_CONN (.in(CK0), .out(CK));
    dc_gen      #(.value(vdd_val)) XP1 (.out(VDD));

    // interface handshaking
    always begin: LOOP
        @(posedge IF.START);
        IF.DONE = 0;
        RX_present = IF.RX_present;
        Cc = IF.Cc;
        Cch = IF.Cch;
        Rterm_tx = IF.Rterm_tx;
        Rterm_rx = IF.Rterm_rx;

        EN = 0;
        #(t_init);
        EN = 1;
        fork
            @(posedge RXDET.CTRL.DONE);
            #(t_meas);
        join_any
        disable fork;

        IF.Np = (RXDET.CTRL.DONE) ? RXDET.CTRL.Np_val : -1;
        IF.DONE = 1;
    end: LOOP
endmodule: FIXTURE

interface IF_t (input bit RST);
    bit START, DONE;                        // handshaking signals
    bit RX_present;                         // configuration parameter
    real Cc, Cch, Rterm_tx, Rterm_rx;       // condition parameters
    int Np;                                 // measurement result
endinterface: IF_t
```

Figure 7. The fixture module including the DUT model of the receiver detection circuit and the test setup for setting the condition parameters (C_c, C_{ch}, $R_{term,tx}$, and $R_{term,rx}$) and measuring the oscillation period count (N_p).

C. *Scoreboard Component*

The scoreboard component receives all the N_p measurement results broadcast by the driver agent and tracks the minimum value of N_p with the receiver present, *min*($N_{p,w/RX}$), and the maximum value of N_p with the receiver absent *max*($N_{p,w/oRX}$). At the end of the simulation, the scoreboard reports these values for the design margin analysis. Figure 8 lists the code of this scoreboard component.

```
class SCOREBOARD extends uvm_scoreboard;
    `uvm_component_utils(SCOREBOARD)

    uvm_table_printer printer;
    int Np_wRX;             // minimum Np with RX present
    int Np_woRX;            // maximum Np with RX absent

    task run_phase(uvm_phase phase);
        Np_wRX = 0;
        Np_woRX = 0;

        forever begin: SCORING
            FIFO.get(PKT);
            if (PKT.RX_present == 1) begin
                if (PKT.Np == -1 || Np_wRX == -1) Np_wRX = -1;
                else if (Np_wRX == 0 || PKT.Np < Np_wRX) Np_wRX = PKT.Np;
            end
            else begin
                if (PKT.Np == -1 || Np_woRX == -1) Np_woRX = -1;
                else if (Np_woRX == 0 || PKT.Np > Np_woRX) Np_woRX = PKT.Np;
            end
        end: SCORING
    endtask: run_phase

    function void report_phase(uvm_phase phase);
        printer.print_generic("", "", 0, {48{"-"}});
        printer.print_generic("", "", 0, $sformatf("Min Np with RX present = %0d", Np_wRX);
        printer.print_generic("", "", 0, $sformatf("Max Np with RX absent  = %0d", Np_woRX);
        printer.print_generic("", "", 0, {48{"-"}});
    endfunction: report_phase

endclass: SCOREBOARD
```

Figure 8. The scoreboard component keeping track of the minimum N_p with receiver present and maximum N_p with receiver absent.

IV. EXPERIMENTAL RESULTS

The described UVM testbench for the PCIe receiver detection circuit is run with Cadence Xcelium and Scientific Analog XMODEL. Figure 9 shows an example of the final simulation log generated by UVM, listing the results of the minimum N_p with the receiver present (*min*($N_{p,w/RX}$)) and the maximum N_p with the receiver absent (*max*($N_{p,w/oRX}$)). On a 64-bit RHEL7 operating system being emulated on a computer equipped with Apple M4 Pro processor and 48-GB memory, it required a total of 91 seconds to complete 400 iterations—200 iterations for minimizing N_p with the receiver present and 200 for maximizing N_p with the receiver absent.

With the comparator thresholds set to $V_{ref,H}$=1.75V and $V_{ref,L}$=1.35V, the minimum N_p with the receiver was 2,507 and the maximum N_p without the receiver was 223. This yields a substantial separation of 2,507 – 223 = 2,284. Using an $N_{p,thres}$ value of 1,365, the corresponding design margins, defined as *min*($N_{p,w/RX}$) - $N_{p,thres}$ and $N_{p,thres}$ - *max*($N_{p,w/oRX}$) can each reach up to 1,142 counts, indicating robust detection behavior.

```
------------------------------------------------
Min Np with RX present = 2507
Max Np with RX absent  = 223
------------------------------------------------
```

Figure 9. The UVM simulation log listing the minimum N_p with the receiver present and maximum N_p with the receiver absent with $V_{ref,H}$=1.75V and $V_{ref,L}$=1.35V.

Table I lists the design margins and simulated *min*($N_{p,w/RX}$) and *max*($N_{p,w/oRX}$) values for various settings of $V_{ref,L}$ and $V_{ref,H}$. The design margin is calculated as (*min*($N_{p,w/RX}$) - *max*($N_{p,w/oRX}$)) / 2, assuming that the threshold $N_{p,thres}$ is placed at the midpoint of (*min*($N_{p,w/RX}$) + *max*($N_{p,w/oRX}$)) / 2. The receiver detection circuit is considered operational only when *min*($N_{p,w/RX}$) exceeds *max*($N_{p,w/oRX}$). In the table, only the cases highlighted in light green satisfy this condition.

It is important to note that this receiver detection circuit exhibits significant sensitivity to the variations in $V_{ref,L}$ and $V_{ref,H}$. Even small changes—on the order of tens of millivolts—can cause the margin to shrink dramatically or even disappear entirely, leading to circuit failure in edge cases. This sensitivity underscores the need for careful threshold tuning to ensure reliable operation across process, voltage, and temperature (PVT) variations.

Table I. Design margins and simulated *min*($N_{p,w/RX}$) and *max*($N_{p,w/oRX}$) values for various combinations of $V_{ref,L}$ and $V_{ref,H}$. Each cell lists the computed design margin, followed by the corresponding (*min*($N_{p,w/RX}$), *max*($N_{p,w/oRX}$)) pair. The entries with negative margins indicate failed detection conditions where *min*($N_{p,w/RX}$) < *max*($N_{p,w/oRX}$), and the entries with '--' represent cases where periodic oscillation could not be observed under some operating conditions.

VrefL \ VrefH	1.700	1.725	1.750	1.775
1.400	**-68** (15, 151)	**-73** (29, 175)	**-74** (55, 203)	**466** (1187, 255)
1.375	**-76** (15, 167)	**-73** (36, 183)	**224** (663, 215)	**1766** (3791, 259)
1.350	**-76** (27, 179)	**125** (449, 199)	**1142** (2507, 223)	**1912** (4095, 271)
1.325	**64** (335, 207)	**868** (1959, 223)	**1802** (3851, 247)	**1906** (4095, 283)
1.300	-- (- , -)	-- (- , -)	-- (- , -)	-- (- , -)

V. CONCLUSION

This work demonstrates that a UVM testbench can be effectively extended to verify the functionality of analog/mixed-signal (AMS) circuits across a continuous-valued parameter space. To efficiently and adaptively explore this space, the testbench combines a reactive stimulus technique with a Bayesian optimization algorithm to identify worst-case deviations in the circuit's response. These measured response bounds can then be used to determine the design margins of the circuit and assess their sensitivity to the secondary parameter variations.

Using a PCI Express (PCIe) receiver detection circuit as a case study, the testbench evaluates the range of the circuit's output—specifically, the oscillation period (N_p)—to assess whether the circuit operates correctly across the full range of specified condition parameters. The approach not only identifies the worst-case operating conditions, but also determines optimal design parameter settings—such as the oscillation period threshold and comparator reference levels—that maximize robustness.

The testbench encapsulates all the analog behaviors in its fixture module built using XMODEL primitives, while standard UVM components handle sequencing, communication, and result analysis. Overall, this work demonstrates that AMS verification can be made both adaptive and coverage-driven, supporting more automated and scalable margin analysis in complex mixed-signal systems.

VI. ACKNOWLEDGMENT

The EDA tools used in this work were supported by the IC Design Education Center (IDEC), Korea and Scientific Analog, Inc, Palo Alto, CA, U.S.A.

VII. REFERENCES

[1] R. Budruk, D. Anderson, and T. Shanley, "PCI Express System Architecture", MindShare, Inc., 2008.

[2] C. Guo and F. Yang, "Method and Apparatus for Receiver Detection on a PCI-Express Bus", US Patent US7222290B2.

[3] Scientific Analog, Inc. XMODEL. [Online]. Available at: https://www.scianalog.com/xmodel.

[4] C. Dancak, "A UVM SystemVerilog Testbench for Analog/Mixed-Signal Verification: A Digitally-Programmable Analog Filter Example," Design and Verification Conference and Exhibition (DVCON) U.S., Mar. 2021.

[5] J. Kim, "A UVM Reactive Testbench for Jitter Tolerance Measurement of High-Speed Wireline Receivers," Design and Verification Conference and Exhibition (DVCON) U.S., Mar. 2023.

[6] J. Kim, "A UVM Testbench for Checking the Global Convergence of Analog/Mixed-Signal Systems: An Adaptive Decision-Feedback Equalizer Example," Design and Verification Conference and Exhibition (DVCON) Europe, Nov. 2024.

[7] C. E. Cummings, et al., “UVM Reactive Stimulus Techniques,” Design and Verification Conference and Exhibition (DVCON) U.S., Mar. 2020.

[8] R. Martinez-Cantin, "BayesOpt: A Bayesian Optimization Library for Nonlinear Optimization, Experimental Design and Bandits," J. of Machine Learning Research, Nov. 2014.

[9] T. Kim, et al., "Verifying Start-up Failures in Coupled Ring Oscillators in Presence of Variability using Predictive Global Optimization," Int’l Conf. on Computer-Aided Design, Nov. 2013.