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 Systems: A PCI-Express Receiver Detection Circuit Example

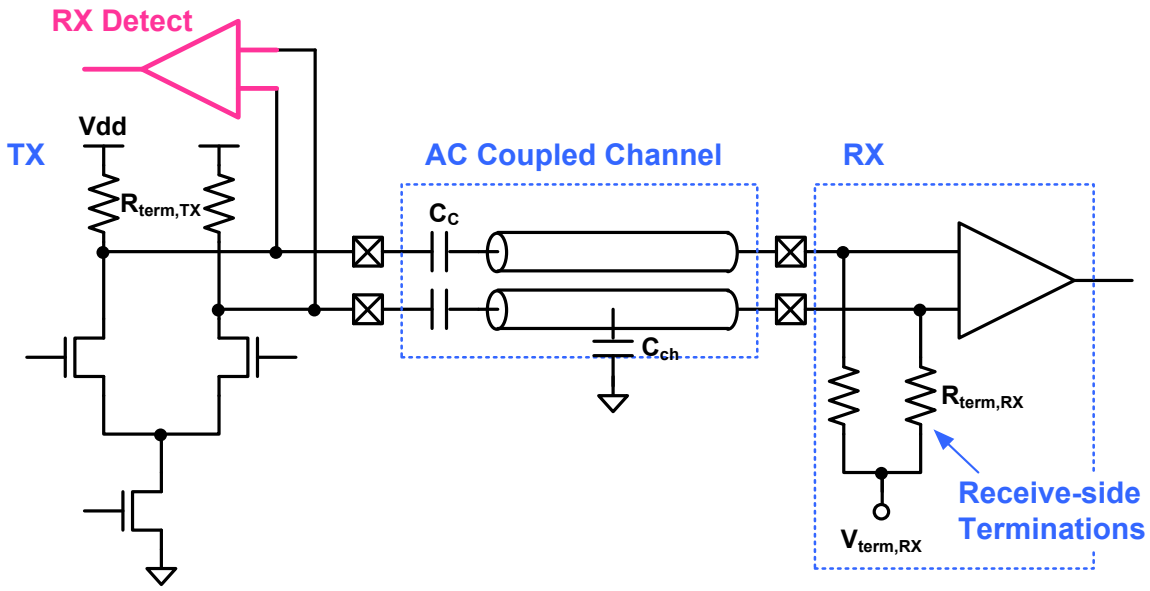
Jaeha Kim
Seoul National University / Scientific Analog, Inc.

scientific analog

accellera SYSTEMS INITIATIVE

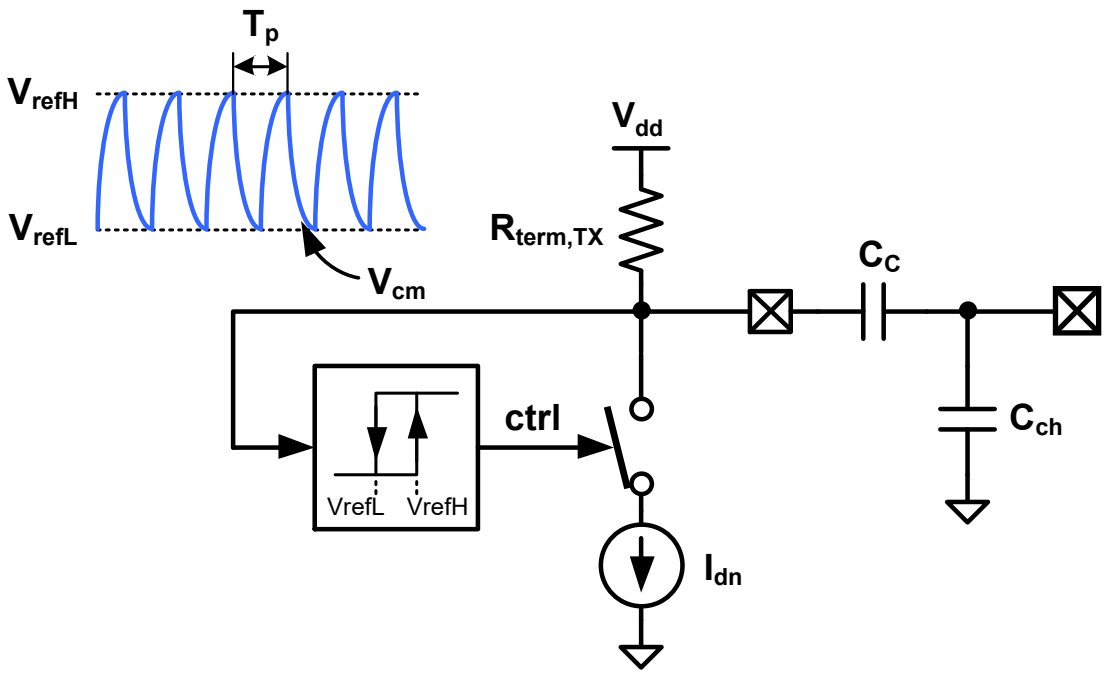
PCIe Receiver Detection Circuit

- A PCIe transmitter (TX) needs to determine whether a receiver (RX) is connected before initiating data transmission
- The receiver detection circuit must function correctly for a range of condition parameters:
  - AC coupling capacitance (C_c) : 76~265nF
  - Equiv. channel capacitance (C_{ch}) : 0~3nF
  - TX-side termination ($R_{term,TX}$) : 40~60Ω
  - RX-side termination ($R_{term,RX}$) : 40~60Ω



RX Detection Circuit Example: Idea

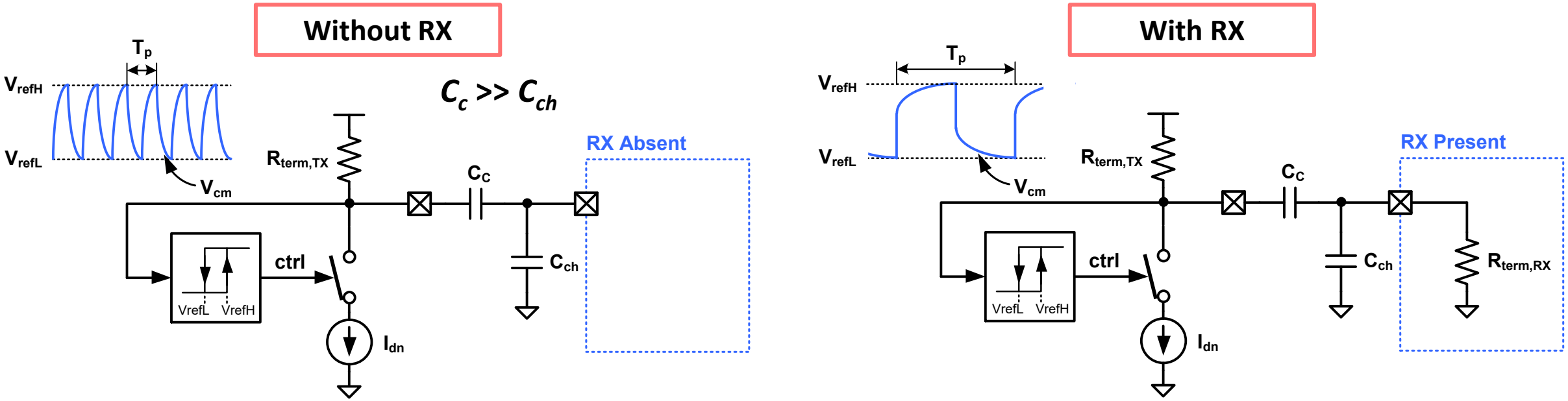
- Transforms the transmitter into a relaxation oscillator and measures its oscillation period
  - With the TX pull-down current (I_{dn}) off, the TX common-mode output (V_{cm}) rises toward the supply voltage V_{dd}
  - When V_{cm} reaches $V_{ref,H}$, I_{dn} is switched on and V_{cm} starts falling
  - When V_{cm} reaches $V_{ref,L}$, I_{dn} is switched off and V_{cm} starts rising again



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

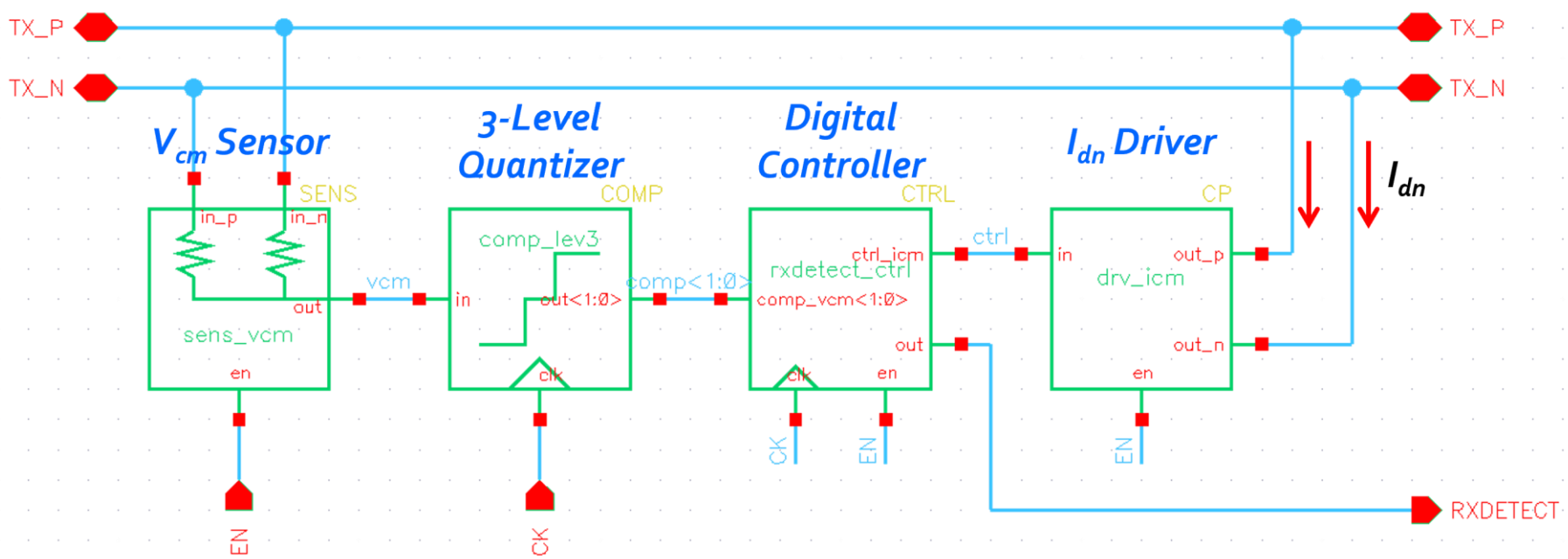
RX Detection Circuit Operation

- The presence of RX can be detected based on the oscillation period
  - Without RX present: V_{cm} settles with $\tau_{w/oRX} \approx R_{term,TX} \cdot C_{ch}$ → *short periods*
  - With RX present: V_{cm} settles with $\tau_{w/RX} \approx (R_{term,TX} + R_{term,RX}) \cdot C_c$ → *long periods*



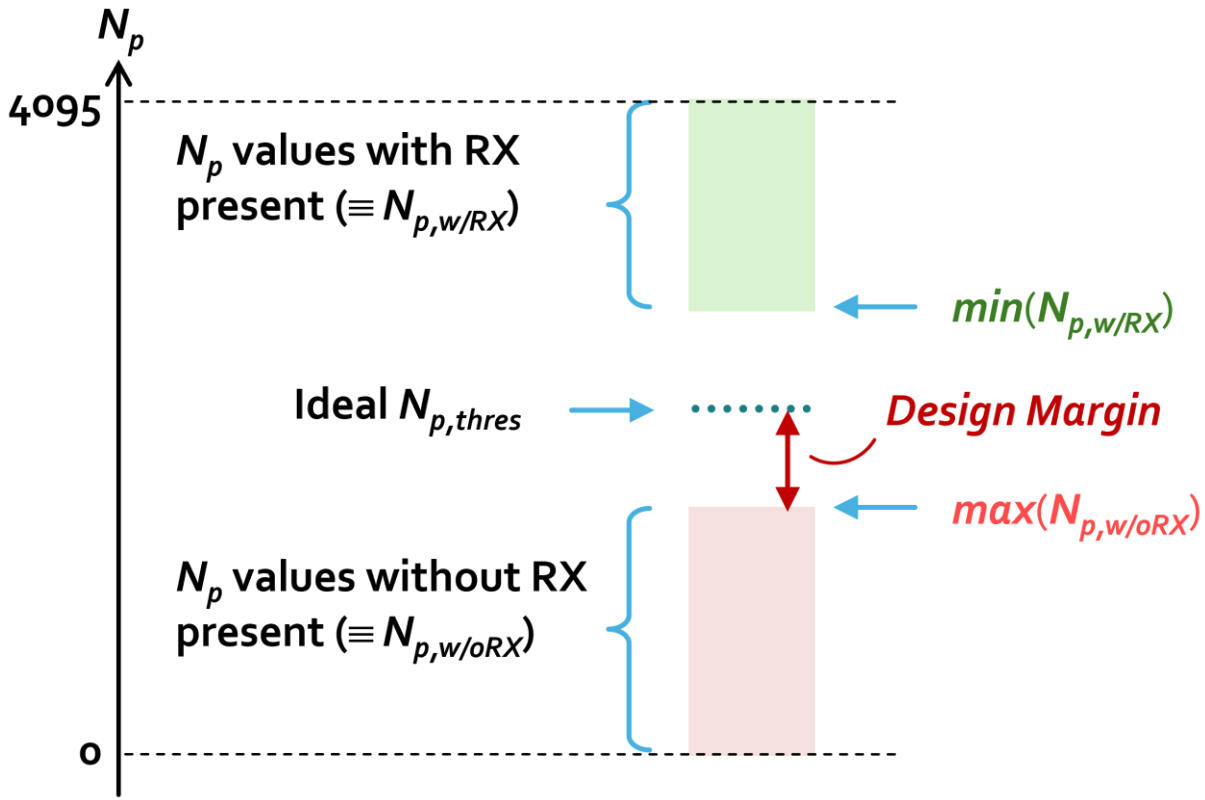
RX Detection Circuit Implementation

- The RX detection circuit is a mixed-signal feedback loop with 4 components
  - A digital controller is inserted to measure the oscillation period in digital counts (N_p)
  - A receiver is considered to be present when $N_p > N_{p,thres}$



Design Margins of RX Detection Circuit

- The RX detection circuit must operate correctly for a range of condition parameter values: (C_{cl}, C_{chl}, $R_{term,TX}$, $R_{term,RX}$)
  - It implies that the oscillation period count (N_p) will vary as well
  - ***Minimum N_p with RX*** must be sufficiently greater than ***maximum N_p without RX***
  - $min(N_{p,w/RX}) > max(N_{p,w/oRX})$



Goal of This Work

- Build a UVM testbench that can characterize the design margin of a PCIe receiver detection circuit
  - What is the **min. N_p value with RX** across the range of condition parameters?
  - What is the **max. N_p value without RX** across the range of condition parameters?

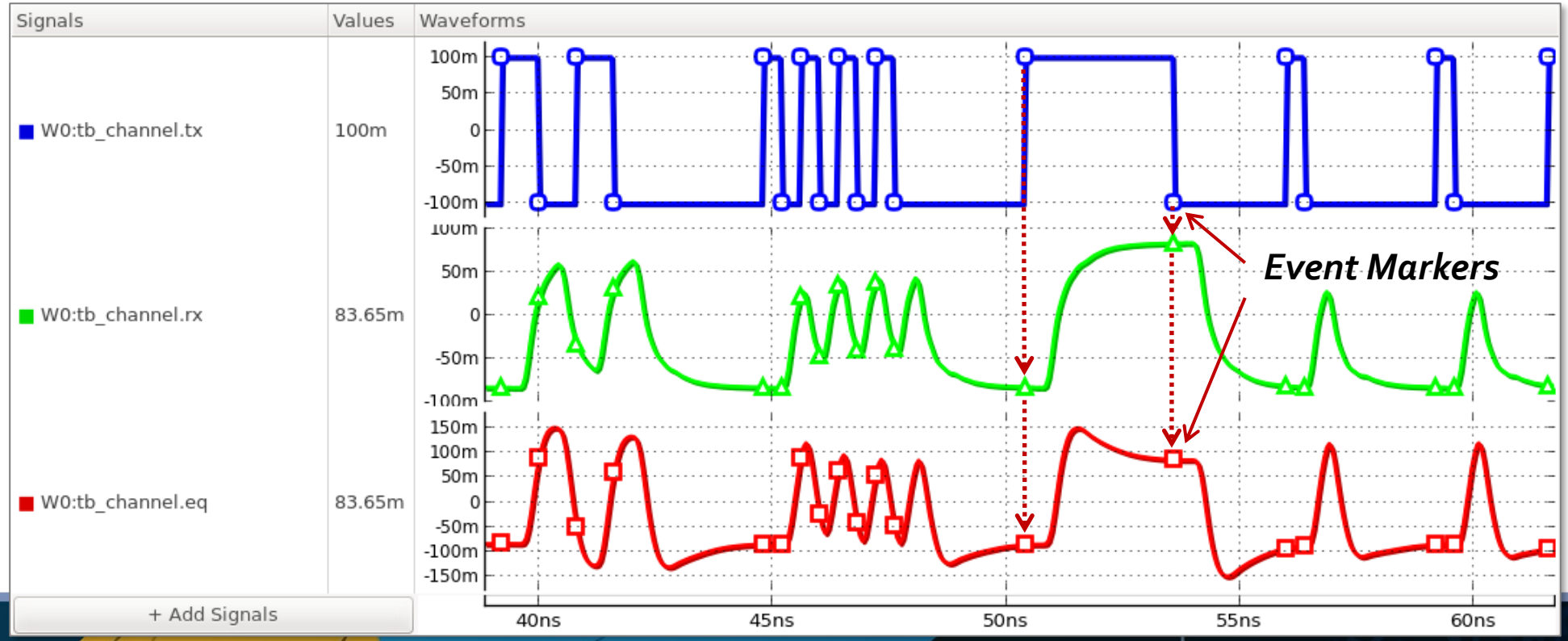
- To achieve this, we need to:
  - Model the analog components of the PCIe transceiver in SystemVerilog; and
  - Find the min/max values of N_p by exploring a continuous-valued parameter space

XMODEL Enables Analog in SystemVerilog/UVM

- *XMODEL* is a plug-in extension enabling ***fast and accurate analog/mixed-signal*** simulation in ***SystemVerilog***
  - ***Event-driven***: delivering 10~100x faster speed than Real-Number Model (RNM)
  - ***Analog***: supporting both functional and circuit-level models
  - ***SystemVerilog***: fully compliant with SystemVerilog-based flows (e.g. UVM)

Event-Driven Simulation of Analog Circuits

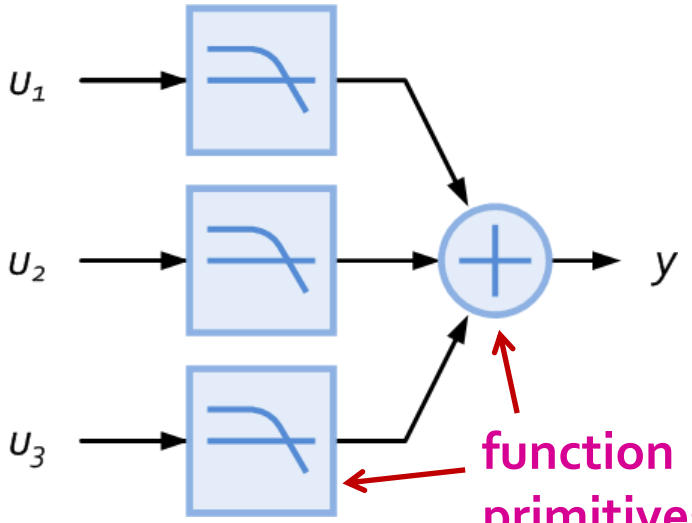
- Using equation-based representation, *XMODEL* performs event-driven simulation of analog circuits in SystemVerilog



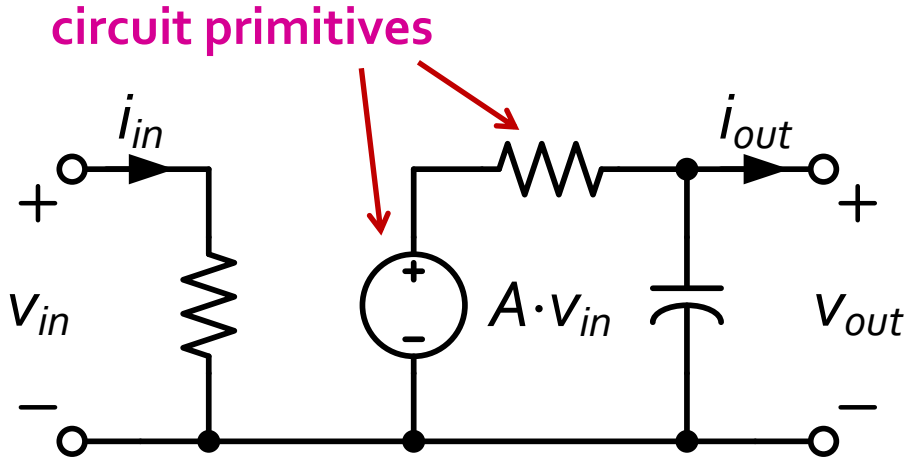
Modeling Analog Circuits with *XMODEL* Primitives

- One can describe both signal-flow and conservative-system models of analog circuits by combining a rich set of *XMODEL* primitives

Signal-flow Model
(Block-Level Model)



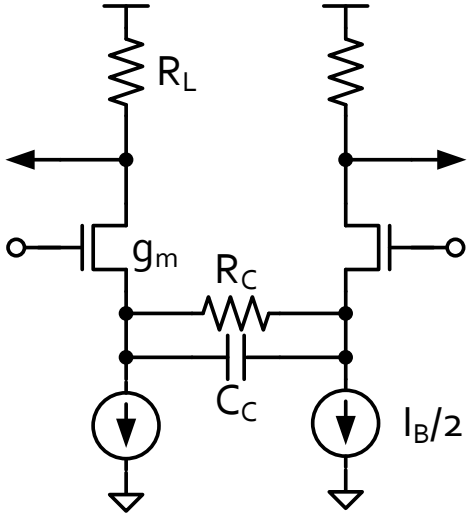
Conservative-System Model
(Circuit-Level Model)



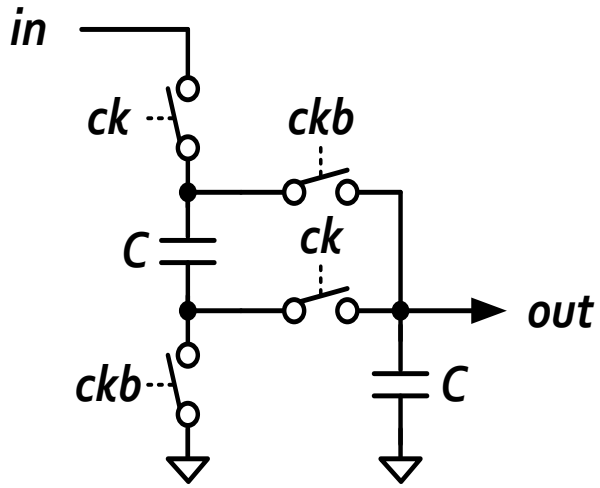
Circuit-Level Modeling with *XMODEL*

- The circuit primitives of *XMODEL* allow one to model analog circuits by listing their elements directly in SystemVerilog
  - It's the most natural way to model nonlinear, switching, and loading effects

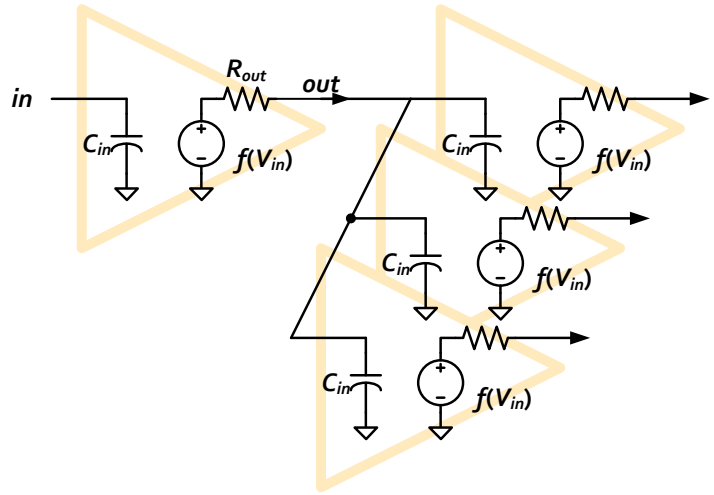
Nonlinear Effects



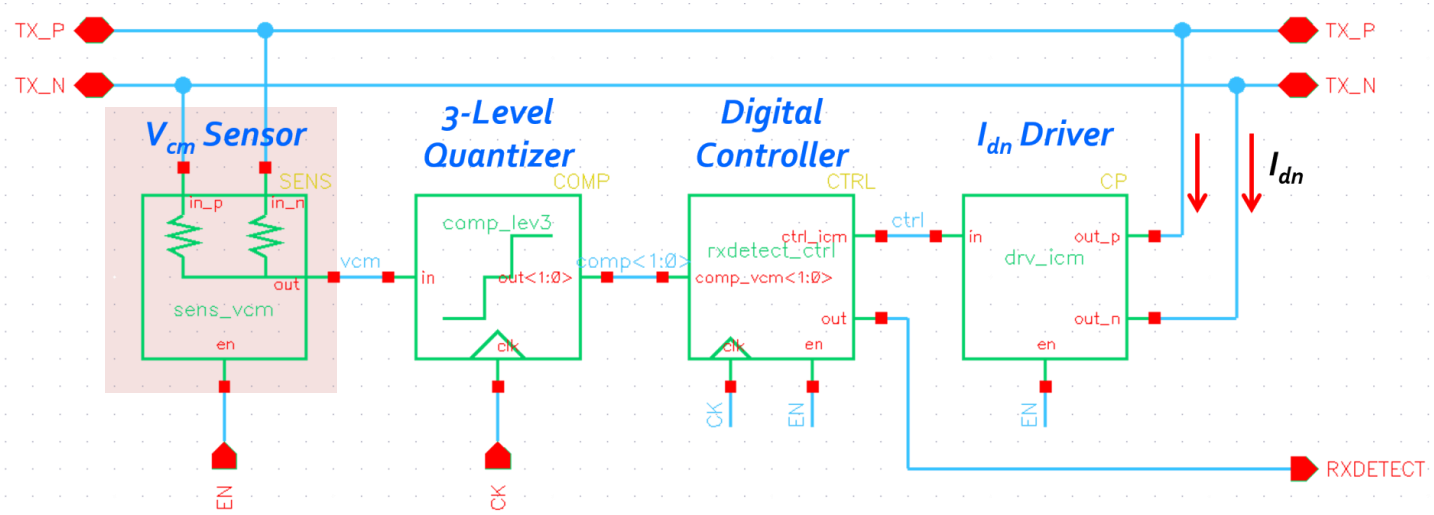
Switching Effects



Loading Effects



RX Detection Circuit Model: V_{cm} Sensor

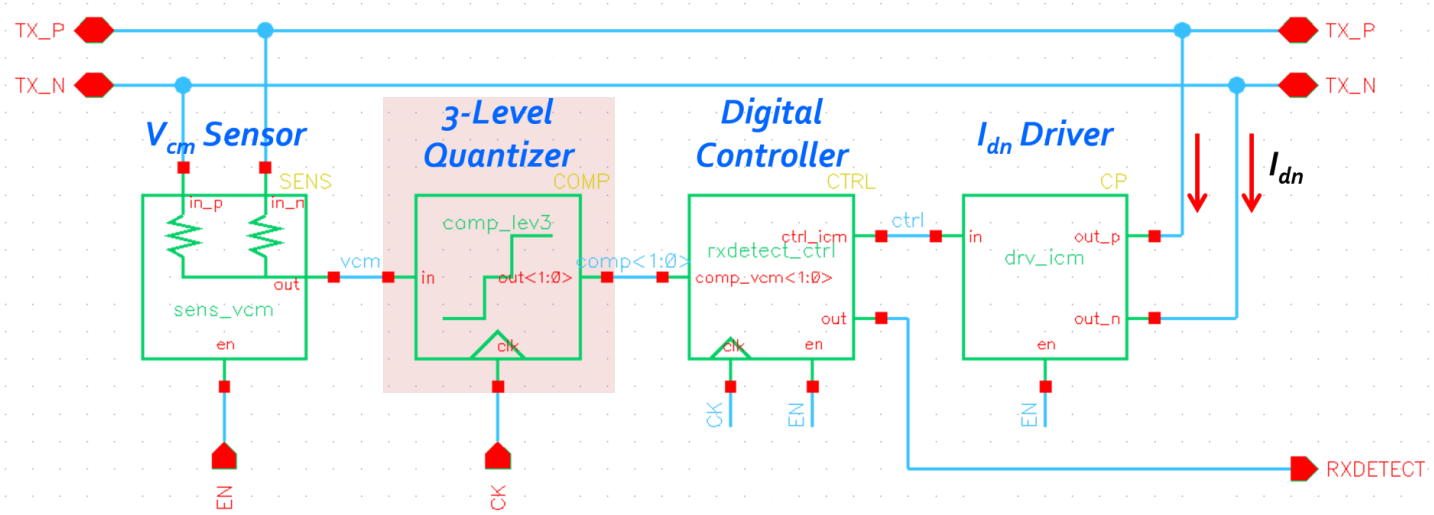


- ***V_{cm} sensor*** measures the common mode of the TX outputs with a pair of resistors when enabled
- Described using ***switch*** primitives

***sens_vcm.sv*: TX common-mode sensor**

```
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
```

RX Detection Circuit Model: 3-Level Quantizer

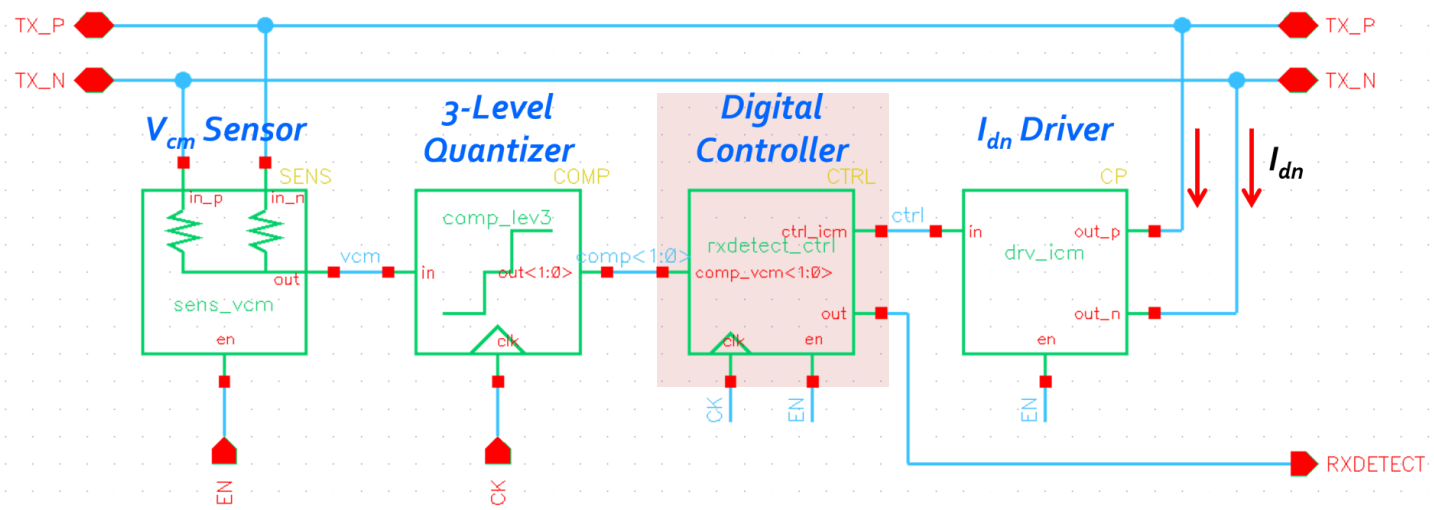


- ***Three-level quantizer*** digitizes V_{cm} into into a 2-bit code 00, 01, 11 by comparing it against V_{refL} and V_{refH}
- Described using ***compare*** primitives

comp_lev3.sv: 3-level quantizer

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

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
```

RX Detection Circuit Model: Digital Controller

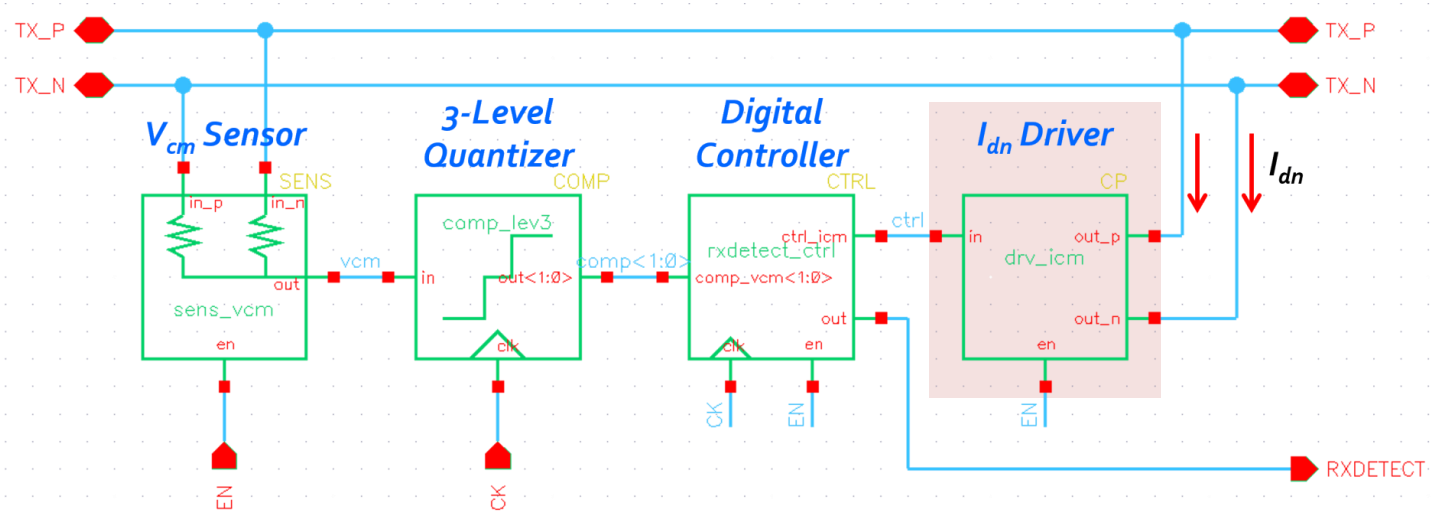


- ***Digital controller*** switches the TX currents (I_{dn}) on or off based on the quantizer output and counts N_p
- Described in pure Verilog

rxdetect_ctrl.sv: digital controller

```
module rxdetect_ctrl #(
    parameter int Np_thres = 1024
)(
    output reg ctrl_icm, out,
    input [1:0] comp_vcm,
    input clk, en
);

always @(posedge clk) begin
    if (!en) {ctrl_prev, ctrl_icm} <= 2'b00;
    else begin
        ctrl_prev <= ctrl_icm;
        if (comp_vcm == 2'b11) ctrl_icm <= 1;
        else if (comp_vcm == 2'b00) ctrl_icm <= 0;
    end
end

always @(posedge clk) begin
    if (!en) {Np_cnt, Tp_cnt, out} <= 16'b0;
    else if (!out) begin
        if (Np_cnt > 0) Tp_cnt <= (Tp_cnt < 12'bfff) ? Tp_cnt
+ 1 : 12'bfff;
        if (ctrl_prev && !ctrl_icm) begin
            if (Np_cnt < Np) Np_cnt <= Np_cnt + 1;
            else {Np_cnt, Tp_cnt, out} <= (Tp_cnt > Tp_thres)
? 16'b1 : 16'b0;
        end
    end
end

endmodule
```

RX Detection Circuit Model: TX Current Driver



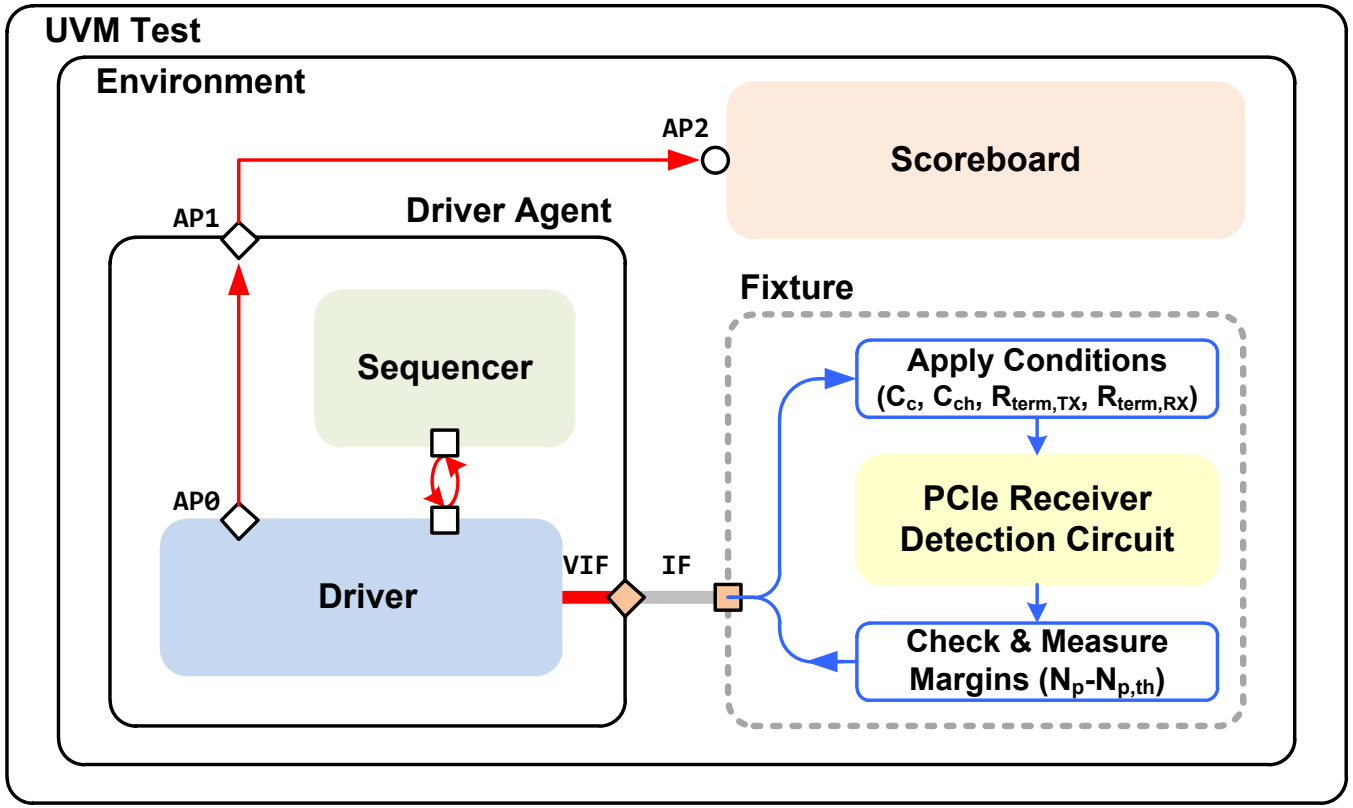
- ***TX current driver*** pulls a common-mode current (I_{dn}) from the TX outputs depending on the control input
- Described using ***transition*** & ***isource*** primitives

drv_icm.sv: TX current driver

```
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
```

UVM Testbench for Design Margin Exploration

- ***Driver agent*** searches for condition parameters that minimize or maximize N_p
- ***Fixture module*** applies the condition parameters to the DUT and measures N_p
- ***Scoreboard*** reports the min. N_p with the receiver present and max. N_p without



Driver Agent

- Driver agent finds the optimal set of C_{cl}, C_{chl}, $R_{term,TXl}$, $R_{term,RX}$ that minimizes or maximizes N_p
  - It uses reactive stimulus technique choosing the next parameter set using a Bayesian optimization
  - With these added DPI functions:
    - **BAYESOPT_initialize()**
    - **BAYESOPT_selectNext()**
    - **BAYESOPT_updateModel()**
    - **BAYESOPT_getOptimum()**

```
class SEQ_MARGIN extends uvm_sequence #(PACKET);
    `uvm_object_utils(SEQ_MARGIN)
    ...
    task body();
        PKT = PACKET::type_id::create("PKT");
        opt_data = BAYESOPT_initialize(4, lower_bounds, upper_bounds);

        // with 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

        // with receiver absent
        ...
    endtask: body
endclass: SEQ_MARGIN
```

Fixture Module

- Fixture module uses a handshaking protocol to enable multiple N_p trials within one simulation run
  - It waits for the driver agent to assert ***START***;
  - Retrieves a new set of C_c, C_{ch}, $R_{term,TX}$, $R_{term,RX}$ values and a flag to connect RX;
  - Enables the RX detection circuit and waits for its response; and
  - Sends the N_p value back to the driver agent and assert ***DONE***

```
module FIXTURE (IF_t IF);
    // 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
```

Fixture Module (2)

- Testbench configurations include:
  - RX detection circuit (DUT)
  - TX driver with AC-coupled channel
  - RX front-end with ideal switches controlling its presence
  - Clock and supply sources
- Using ***cap_sw*** & ***res_sw*** primitives to change C_c, C_{ch}, $R_{term,TX}$, $R_{term,RX}$ as signals (i.e., not parameters) during simulation

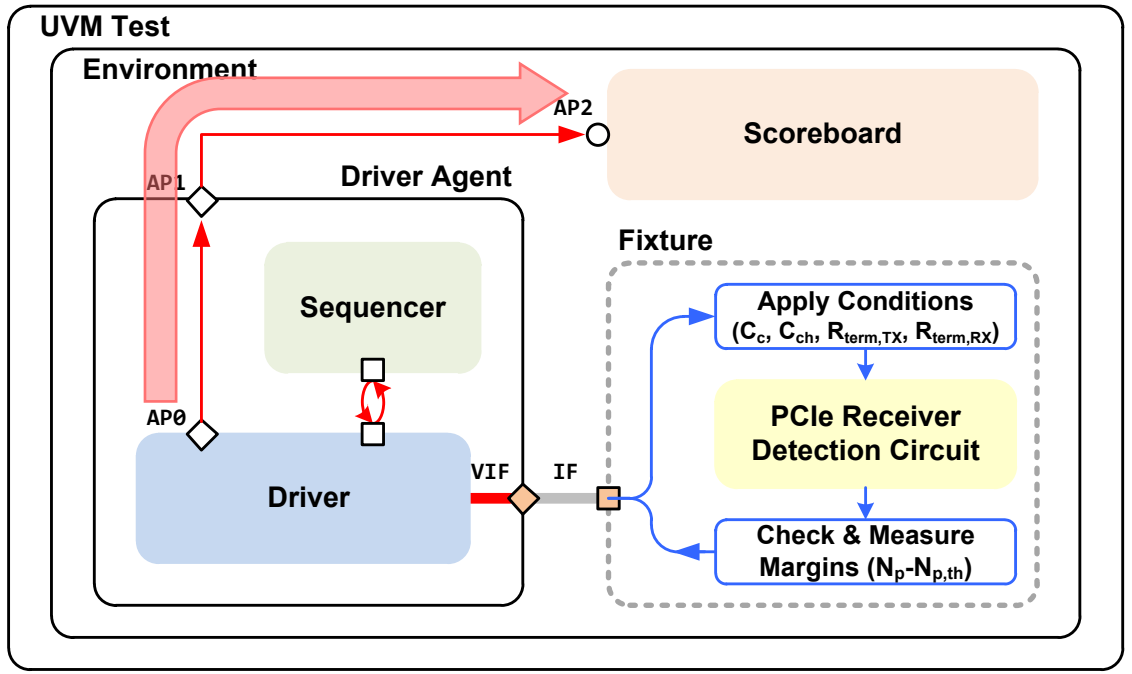
```
    // DUT model for the PCIe receiver detection circuit
    rxdetect    #(.VrefH(VrefH), .VrefL(VrefL), .Np_thres(1024))
                RXDET (.CK(CK), .TX_P(TX_P), .TX_N(TX_N),
                       .EN(EN), .RXDETECT(RXDETECT));

    // 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));

endmodule
```

Scoreboard Component

- It listens to all the N_p results broadcast by the driver agent and keeps track of $min(N_{p,w/RX})$ & $max(N_{p,w/oRX})$

- And reports these values when the simulation is finished

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



Experimental Results: Measuring N_p

- Collecting $\boldsymbol{min}(\boldsymbol{N_{p,w/RX}})$ & $\boldsymbol{max}(\boldsymbol{N_{p,w/oRX}})$ for various $\boldsymbol{V_{refH}}$ & $\boldsymbol{V_{refL}}$ values
- Each simulation takes about 90 seconds to complete 400 iterations

V_{refL} \ V_{refH}	**1.700**	**1.725**	**1.750**	**1.775**
1.400	(15, 151)	(29, 175)	(55, 203)	(1187, 255)
1.375	(15, 167)	(36, 183)	(663, 215)	(3791, 259)
1.350	(27, 179)	(499, 199)	**(2507, 223)**	(4095, 271)
1.325	(355, 207)	(1959, 223)	(3851, 247)	(4095, 283)
1.300	No Oscillations	No Oscillations	No Oscillations	No Oscillations

Experimental Results: Design Margins

- Design margin is calculated as ($min(N_{p,w/RX})$ - $max(N_{p,w/oRX})$) / 2
- Design margin is highly sensitive to the variations in V_{refH} and V_{refL}

V_{refL} \ V_{refH}	1.700	1.725	1.750	1.775
1.400	-68	-73	-74	466
1.375	-76	-73	224	1766
1.350	-76	125	**1142**	1912
1.325	64	868	1802	1906
1.300	-	-	-	-

Summary

- A UVM testbench for characterizing the design margins of an analog/ mixed-signal circuit is presented
- The testbench finds the worst-case deviation of the circuit's response across a continuous-valued parameter space by:
  - Using *XMODEL* primitives to model the analog components of a PCIe receiver detection circuit at the device level; and
  - Combining a reactive stimulus technique with a Bayesian optimization algorithm to efficiently and adaptively explore the parameter space
- This work paves the way for adaptive, coverage-driven AMS verification