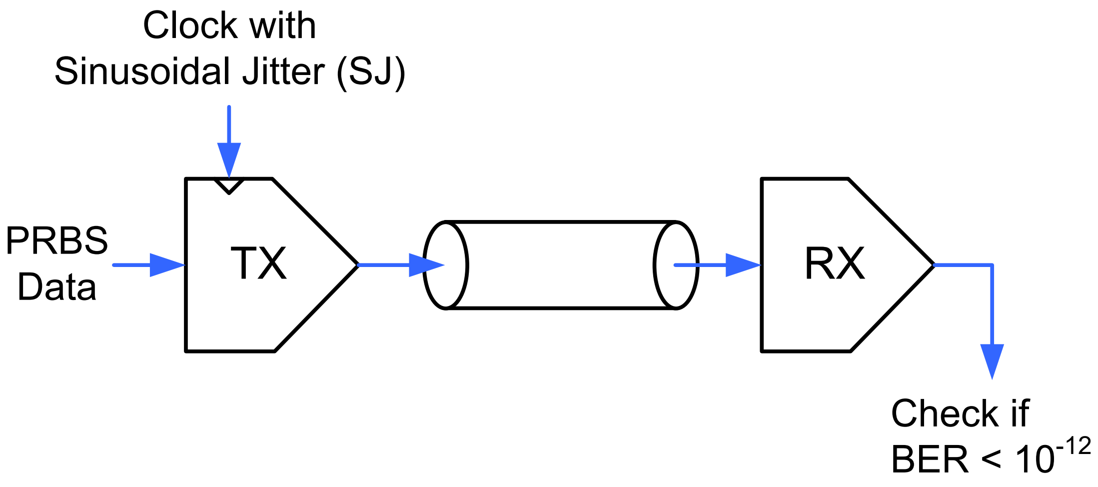
2023
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
FEBRUARY 27-MARCH 2, 2023

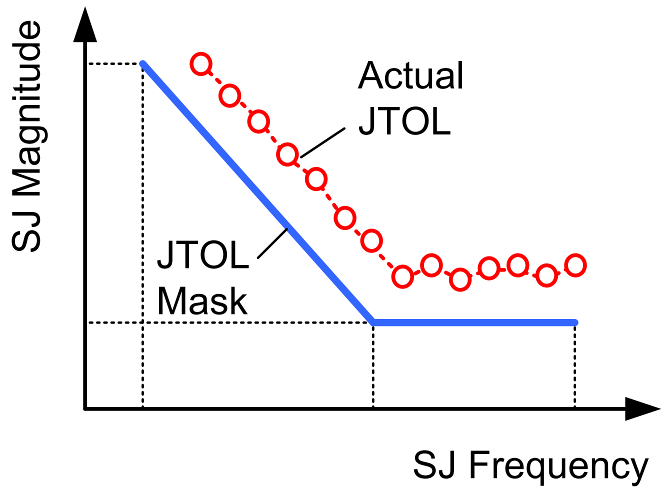
A UVM Reactive Testbench for Jitter Tolerance Measurement of High-Speed Wireline Receivers

Jaeha Kim
Seoul National University / Scientific Analog, Inc.

Jitter Tolerance (JTOL) of High-Speed Receivers

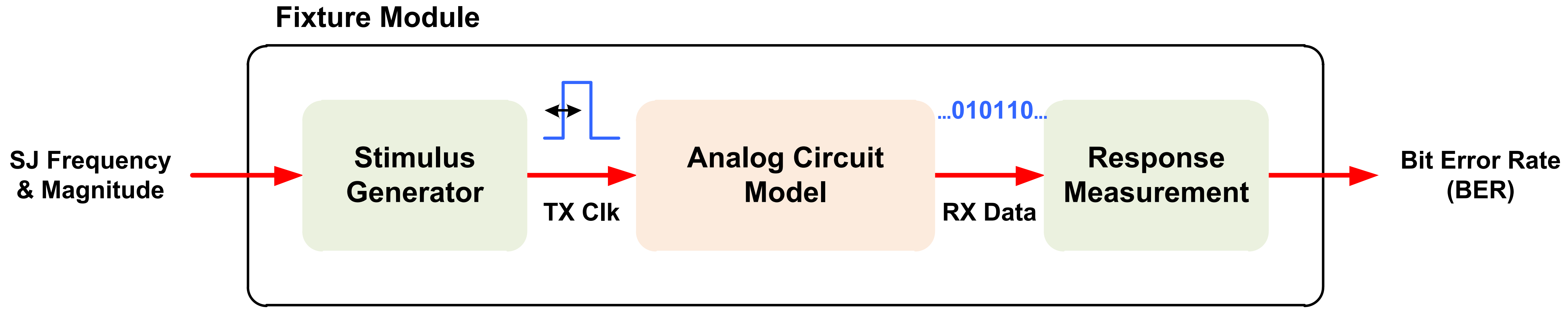
- Jitter tolerance (JTOL) test measures the resilience of a high-speed wireline receiver by finding the maximum magnitude of an additional sinusoidal jitter (SJ) that can be tolerated for a target BER (e.g. 10^{-12})
- This work presents a UVM testbench performing such an iterative search





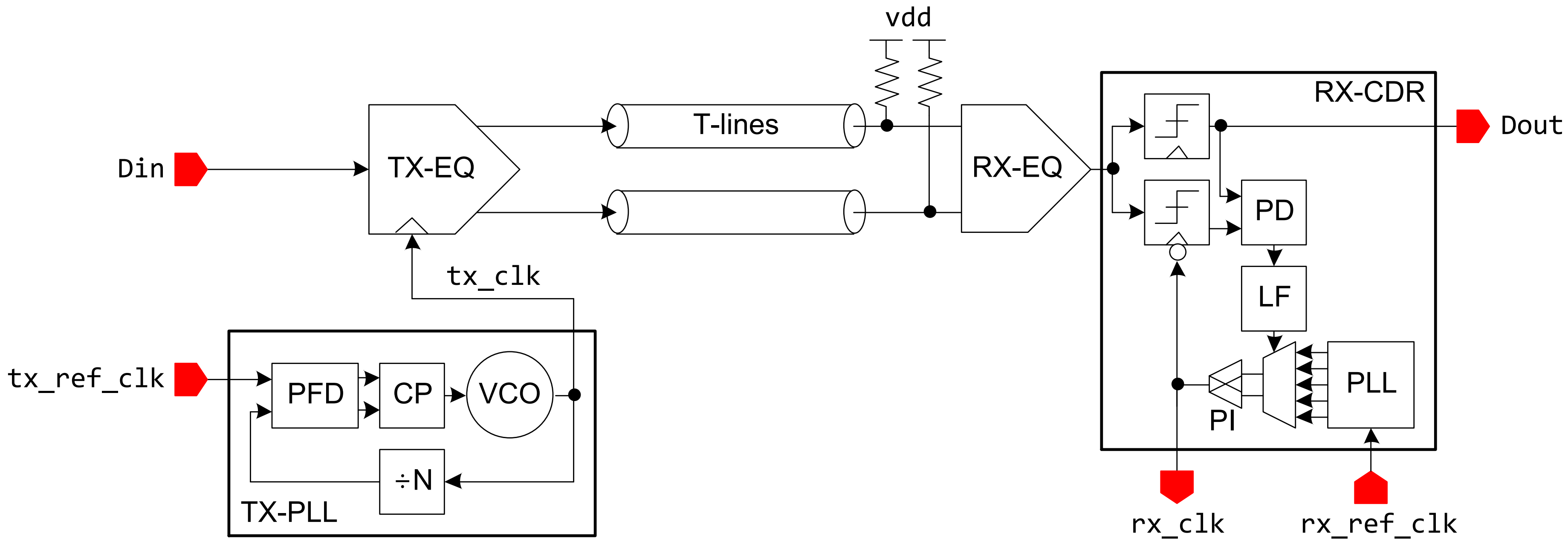
Extending UVM to Analog/Mixed-Signal Verification

- UVM testbenches for AMS circuits can be built with standard components if we use a well-defined ***fixture module*** enclosing these elements:
  - AMS device under verification (DUV) modeled in SystemVerilog
  - Analog instrumentations for generating stimuli and measuring responses



Modeling a High-Speed Wireline Transceiver

- Requires capabilities of simulating analog pulses propagating through a lossy channel, expressing precise timing of PLL & CDR, and modeling analog circuits directly as a network of circuit elements in SystemVerilog

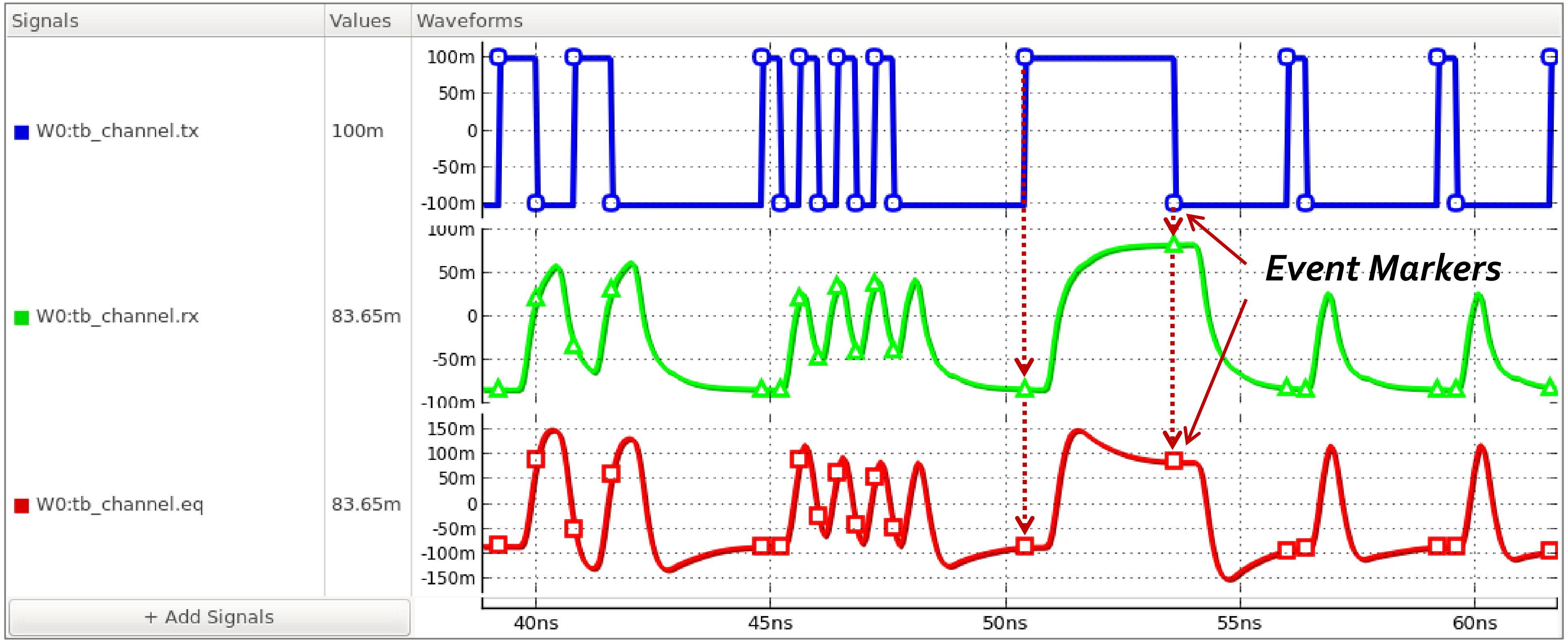


XMODEL Enables Analog in SystemVerilog/UVM

- *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 XMODEL*

- Ideal for simulating data pulses through the lossy channel and equalizer

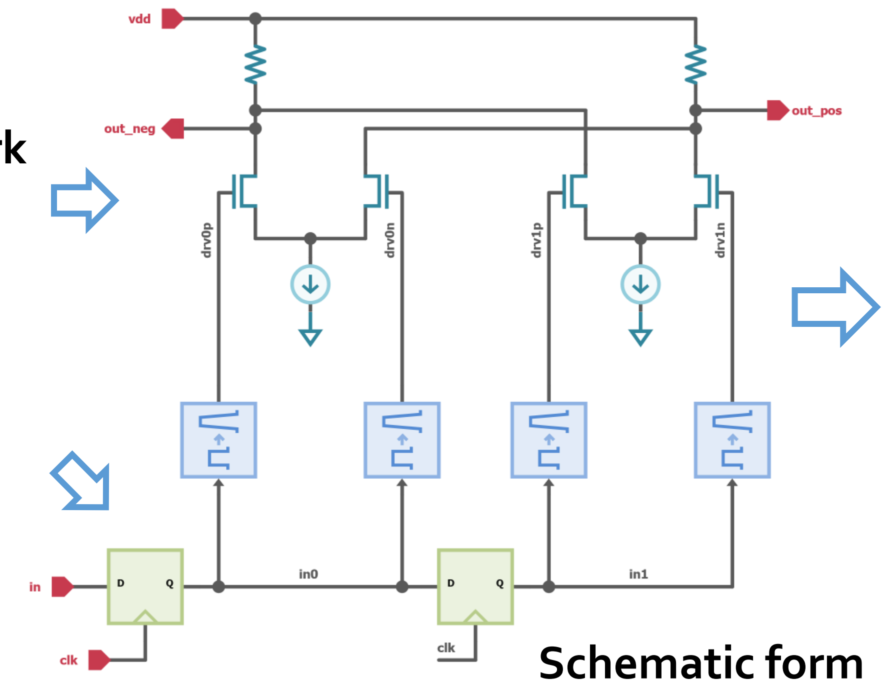


Equalizing Transmitter Model Example

- AMS models are composed by putting together the *XMODEL* primitives

Can describe circuit directly as a network of circuit primitives

Timing accuracy of digital pulses is not limited by timestep



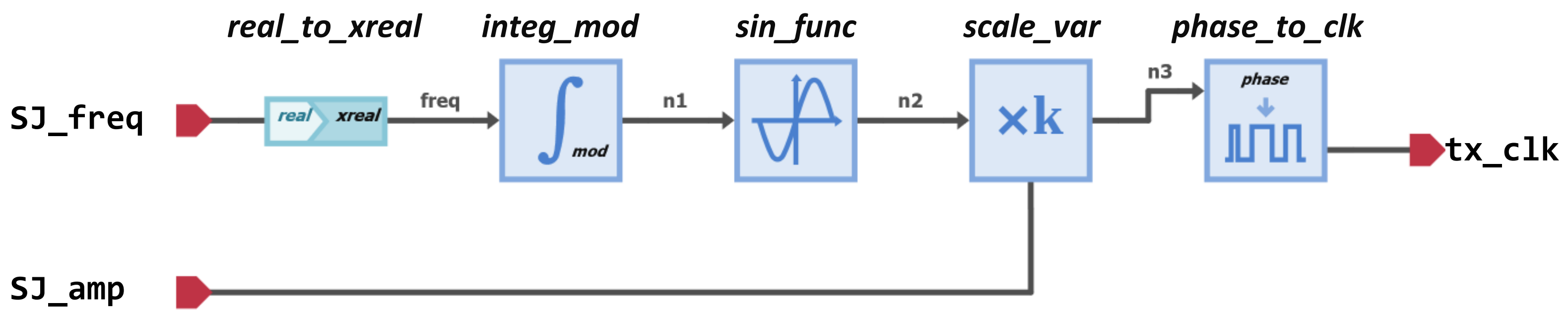
Schematic form

Source form

```
module tx_eq (...);

xreal drv0p, drv0n, drv1p, drv1n;
xreal tail0, tail1;
xbit in0, in1;

isource     #(.mode("dc"), .dc(0.00875))
            I0 (.neg(`ground), .pos(tail0), .in(`ground));
isource     #(.mode("dc"), .dc(0.00125))
            I1 (.neg(`ground), .pos(tail1), .in(`ground));
nmosfet     #(.W(2e-05), .L(4e-08), .Kp(2e-05), .Vth(0.5))
            M0 (.g(drv0p), .d(out_neg), .b(`ground), .s(tail0));
nmosfet     #(.W(2e-05), .L(4e-08), .Kp(2e-05), .Vth(0.5))
            M1 (.g(drv0n), .d(out_pos), .b(`ground), .s(tail0));
nmosfet     #(.W(2e-05), .L(4e-08), .Kp(2e-05), .Vth(0.5))
            M2 (.g(drv1p), .d(out_neg), .b(`ground), .s(tail1));
nmosfet     #(.W(2e-05), .L(4e-08), .Kp(2e-05), .Vth(0.5))
            M3 (.g(drv1n), .d(out_pos), .b(`ground), .s(tail1));
resistor    #(.R(50)) R0 (.neg(out_pos), .pos(vdd));
resistor    #(.R(50)) R1 (.neg(out_neg), .pos(vdd));
transition  #(.value0(0.0), .value1(1.0)) XP0 (.out(drv0p), .in(in0));
transition  #(.value0(1.0), .value1(0.0)) XP1 (.out(drv0n), .in(in0));
transition  #(.value0(1.0), .value1(0.0)) XP2 (.out(drv1p), .in(in1));
transition  #(.value0(0.0), .value1(1.0)) XP3 (.out(drv1n), .in(in1));
dff_xbit    XP4 (.clk(clk), .d(in), .q(in0));
dff_xbit    XP5 (.clk(clk), .d(in0), .q(in1));

endmodule   // tx_eq
```

Fixture Module with Analog Instrumentations

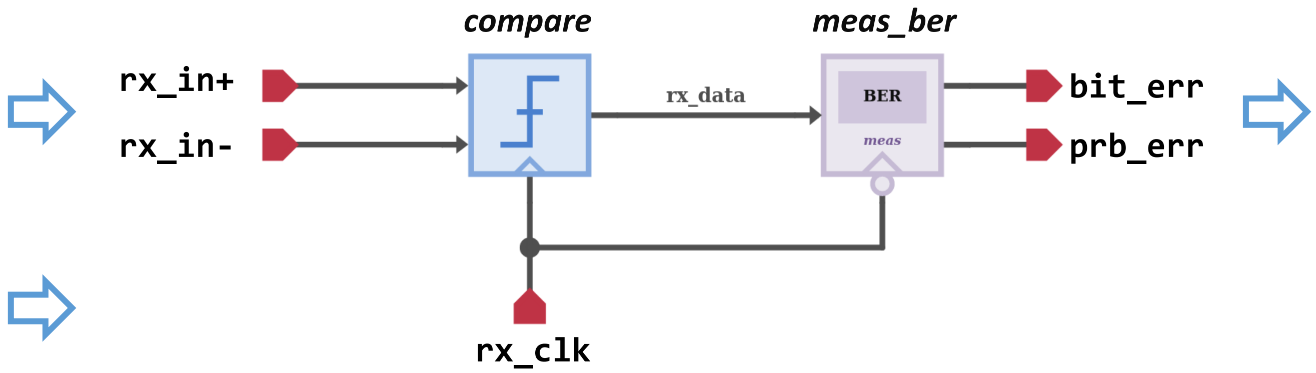
- Generating a clock with SJ from the frequency & amplitude inputs:



- Measuring the BER of the received data using statistical simulation:

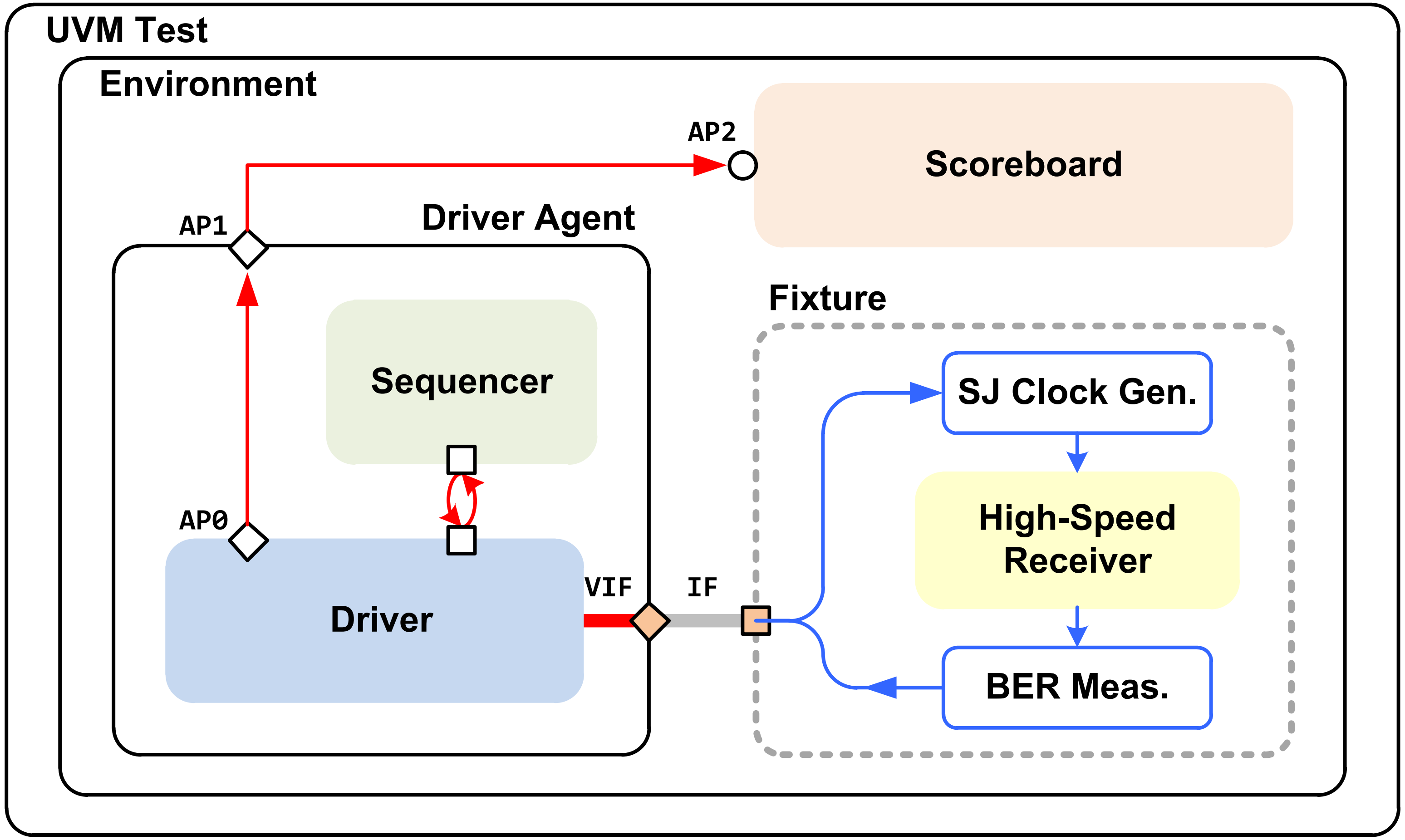
Analog input with voltage noise

Clock input with timing noise (jitter)



Statistical probability of bit-errors

UVM Testbench for JTOL Measurement



- The rest of testbench can be built using the standard UVM components
  - Sequencer
  - Driver
  - Scoreboard

UVM Reactive Stimulus Technique

- A technique to reactively generate the next sequence item based on the result of the previous sequence items [Cummings, et al., 2020]

UVM Sequence

```
class SEQ_JTOL extends uvm_sequence #(PACKET);
    `uvm_component_utils(SEQ_JTOL)
    PACKET PKT, RSP;

    task body();
        PKT = PACKET::type_id::create("PKT");
        forever begin
            start_item(PKT);
            // ... set PKT data values
            finish_item(PKT);

            get_response(RSP);
            // ... read RSP data values
            // ... decide on the next item values
        end
    endtask: body
endclass: SEQ_JTOL
```

Stimulus Data

Response Data

UVM Driver

```
class DRIVER extends uvm_driver #(PACKET);
    `uvm_component_utils(DRIVER)
    PACKET PKT, RSP;

    task run_phase(uvm_phase phase);
        forever begin
            seq_item_port.get_next_item(PKT);
            if (!$cast(RSP, PKT.clone())) `uvm_fatal("DRIVER", "failed");
            RSP.set_id_info(PKT);
            // ... drive PKT data to the FIXTURE

            // ... wait until the FIXTURE response is ready (VIF.BER)
            RSP.BER = VIF.BER;
            seq_item_port.item_done(RSP);
        end
    endtask: run_phase
endclass: DRIVER
```

To/From Fixture

Search Algorithms for JTOL Measurement

```
class SEQ_JTOL extends uvm_sequence #(PACKET);
    task body();
        // ... details omitted for brevity
        flag = 0;

        // phase 1: linear search to find the first failing point
        while (1) begin
            start_item(PKT); finish_item(PKT); get_response(RSP);
            if (RSP.BER < RSP.BER_tol) begin
                mag_min = PKT.SJ_mag;
                PKT.SJ_mag += mag_inc;
                if (flag == -1) break; else flag = 1;
            end
            else begin
                mag_max = PKT.SJ_mag;
                PKT.SJ_mag -= mag_inc;
                if (flag == 1) break; else flag = -1;
            end
        end

        // phase 2: binary search to find the pass/fail boundary
        while (mag_max/mag_min > 1.05) begin
            PKT.SJ_mag = $sqrt(mag_max * mag_min);
            start_item(PKT); finish_item(PKT); get_response(RSP);
            if (RSP.BER < RSP.BER_tol) mag_min = PKT.SJ_mag;
            else mag_max = PKT.SJ_mag;
        end
    endtask: body
endclass: SEQ_JTOL
```

- Using the reactive stimulus technique, the max. SJ magnitude with BER < 10^{-12} is found in two steps:
  - Linear search to identify an interval containing the BER pass/fail boundary
  - Binary search to refine the boundary

UVM Scoreboard

```
class SCOREBOARD extends uvm_scoreboard;
    `uvm_component_utils(SCOREBOARD)
    SCORECARD SCD;
    PACKET PKT;
    uvm_analysis_export #(PACKET) AP2;
    uvm_tlm_analysis_fifo #(PACKET) FIFO;

    task run_phase(uvm_phase phase);
        int N;
        forever begin
            FIFO.get(PKT);
            N = PKT.tag;
            if (SCD.DATA.exists(N)) begin
                if (PKT.BER < PKT.BER_tol && SCD.DATA[N].mag < PKT.SJ_mag)
                    SCD.DATA[N].mag = PKT.SJ_mag;
            end
            else begin
                SCD.DATA[N].freq = PKT.SJ_freq;
                SCD.DATA[N].mag = (PKT.BER < PKT.BER_tol) ? PKT.SJ_mag : 0.0;
            end
            SCD.num_trials++;
        end
    endtask: run_phase

endclass: SCOREBOARD
```

- Driver broadcasts the BER result of each trial to the scoreboard
- Scoreboard keeps the record of the max. SJ magnitude with BER < 10^{-12} for each SJ frequency

UVM Simulation Log

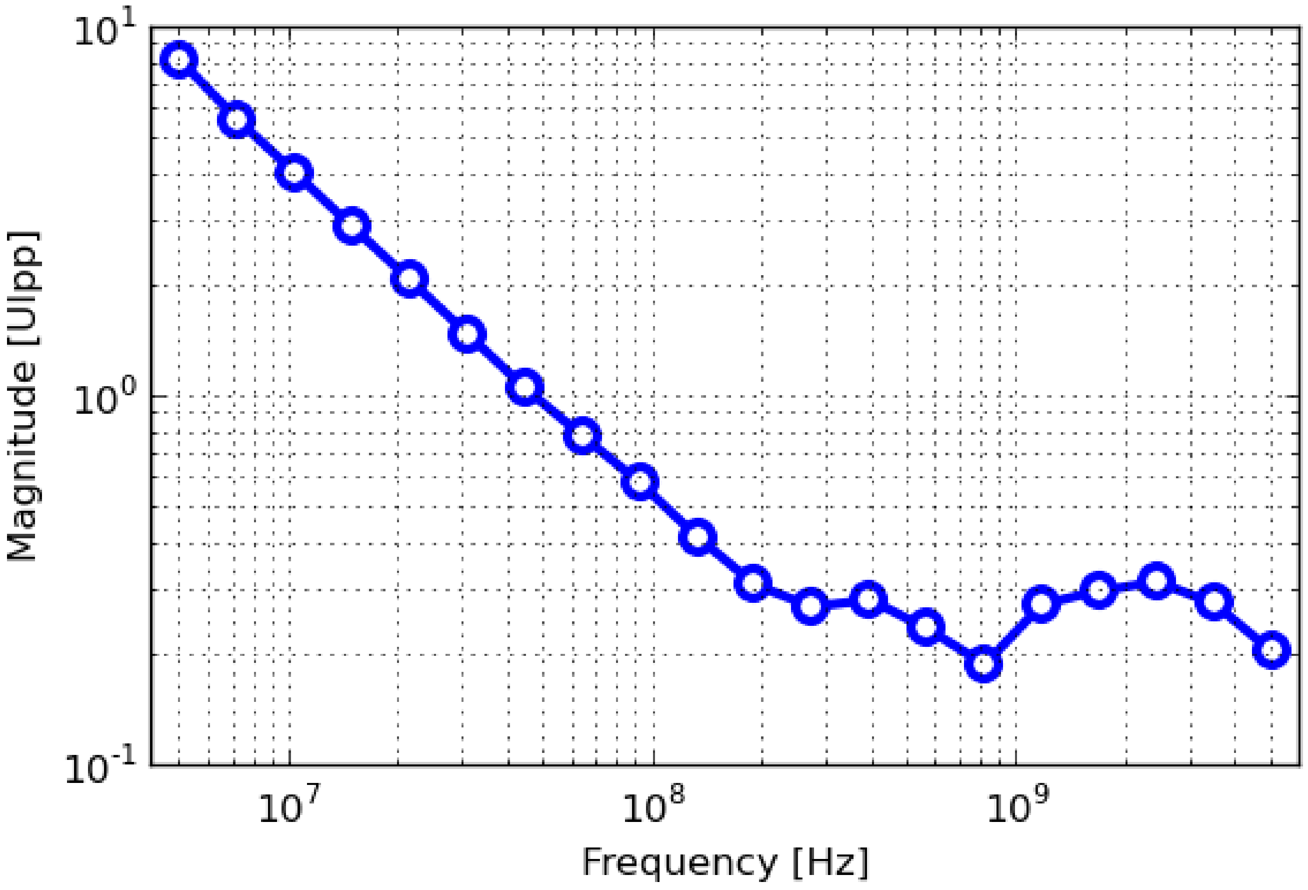
- Simulation takes 41 min. running 106 BER trials for 20 SJ frequencies

```
UVM_INFO @ 2210.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 20 SJ freq=5.000000e+09 Hz, mag=5.000000e-01 UIpp --> BER=6.033595e-04
UVM_INFO @ 4411.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 20 SJ freq=5.000000e+09 Hz, mag=4.000000e-01 UIpp --> BER=4.253071e-06
UVM_INFO @ 6612.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 20 SJ freq=5.000000e+09 Hz, mag=3.000000e-01 UIpp --> BER=4.559507e-09
UVM_INFO @ 8813.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 20 SJ freq=5.000000e+09 Hz, mag=2.000000e-01 UIpp --> BER=4.824476e-13
UVM_INFO @ 11014.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 20 SJ freq=5.000000e+09 Hz, mag=2.449490e-01 UIpp --> BER=3.271928e-11
UVM_INFO @ 13215.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 20 SJ freq=5.000000e+09 Hz, mag=2.213364e-01 UIpp --> BER=3.781672e-12
UVM_INFO @ 15416.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 20 SJ freq=5.000000e+09 Hz, mag=2.103979e-01 UIpp --> BER=1.263946e-12
UVM_INFO @ 17617.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 20 SJ freq=5.000000e+09 Hz, mag=2.051331e-01 UIpp --> BER=7.470467e-13
UVM_INFO @ 19818.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 19 SJ freq=3.475964e+09 Hz, mag=2.051331e-01 UIpp --> BER=2.923724e-15
UVM_INFO @ 22019.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 19 SJ freq=3.475964e+09 Hz, mag=2.461597e-01 UIpp --> BER=8.445417e-14
UVM_INFO @ 24220.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 19 SJ freq=3.475964e+09 Hz, mag=2.871863e-01 UIpp --> BER=2.007209e-12
UVM_INFO @ 26421.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 19 SJ freq=3.475964e+09 Hz, mag=2.658829e-01 UIpp --> BER=4.560037e-13
UVM_INFO @ 28622.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | # 19 SJ freq=3.475964e+09 Hz, mag=2.763294e-01 UIpp --> BER=7.860811e-13
(...)
```

```
(...)
UVM_INFO @ 206903.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  3 SJ freq=1.034569e+07 Hz, mag=4.320903e+00 UIpp --> BER=9.844018e-09
UVM_INFO @ 209104.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  3 SJ freq=1.034569e+07 Hz, mag=4.179040e+00 UIpp --> BER=2.617306e-11
UVM_INFO @ 211305.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  2 SJ freq=7.192249e+06 Hz, mag=4.179040e+00 UIpp --> BER=2.324675e-19
UVM_INFO @ 213506.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  2 SJ freq=7.192249e+06 Hz, mag=5.014848e+00 UIpp --> BER=1.745598e-17
UVM_INFO @ 215707.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  2 SJ freq=7.192249e+06 Hz, mag=5.850657e+00 UIpp --> BER=1.286969e-12
UVM_INFO @ 217908.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  2 SJ freq=7.192249e+06 Hz, mag=5.416655e+00 UIpp --> BER=3.667517e-16
UVM_INFO @ 220109.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  2 SJ freq=7.192249e+06 Hz, mag=5.629475e+00 UIpp --> BER=4.189074e-15
UVM_INFO @ 222310.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  1 SJ freq=5.000000e+06 Hz, mag=5.629475e+00 UIpp --> BER=1.387735e-20
UVM_INFO @ 224511.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  1 SJ freq=5.000000e+06 Hz, mag=6.755370e+00 UIpp --> BER=4.093819e-19
UVM_INFO @ 226712.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  1 SJ freq=5.000000e+06 Hz, mag=7.881265e+00 UIpp --> BER=1.972978e-15
UVM_INFO @ 228913.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  1 SJ freq=5.000000e+06 Hz, mag=9.007160e+00 UIpp --> BER=2.232116e-03
UVM_INFO @ 231114.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  1 SJ freq=5.000000e+06 Hz, mag=8.425427e+00 UIpp --> BER=2.148739e-11
UVM_INFO @ 233315.000ns: uvm_test_top.E.AGNT.DRV [RUN]
  | #  1 SJ freq=5.000000e+06 Hz, mag=8.148805e+00 UIpp --> BER=8.002244e-15
```

Simulated Jitter Tolerance (JTOL) Results

```
-------------------------------------------------
JITTER TOLERANCE (JTOL)
INDEX      FREQUENCY(Hz)    MAGNITUDE(UIpp)
-------------------------------------------------
1          5.0000e+06       8.1488
2          7.1922e+06       5.6295
3          1.0346e+07       4.0418
4          1.4882e+07       2.8870
5          2.1407e+07       2.0728
6          3.0792e+07       1.4806
7          4.4293e+07       1.0576
8          6.3714e+07       0.7851
9          9.1649e+07       0.5858
10         1.3183e+08       0.4184
11         1.8963e+08       0.3106
12         2.7278e+08       0.2709
13         3.9238e+08       0.2834
14         5.6442e+08       0.2362
15         8.1189e+08       0.1894
16         1.1679e+09       0.2756
17         1.6799e+09       0.2996
18         2.4165e+09       0.3168
19         3.4760e+09       0.2763
20         5.0000e+09       0.2051
-------------------------------------------------
TOTAL NUMBER OF TRIALS: 106
-------------------------------------------------
```



Summary

- A UVM testbench for AMS circuits can be built using standard UVM components with a well-defined fixture module
- To measure the jitter tolerance of a high-speed wireline receiver:
  - The fixture module enclosing the transceiver model and its instrumentations can be composed with *XMODEL* primitives
  - Search for maximum SJ can be performed using the reactive stimulus technique
- This work was supported by Samsung Electronics, Co. Ltd. and the EDA tools were supported by IDEC and Scientific Analog, Inc.