2022
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

A UVM SystemVerilog Testbench for 5G/LTE Multi-Standard RF Transceiver

ByeongKyu Kim and Jaeha Kim

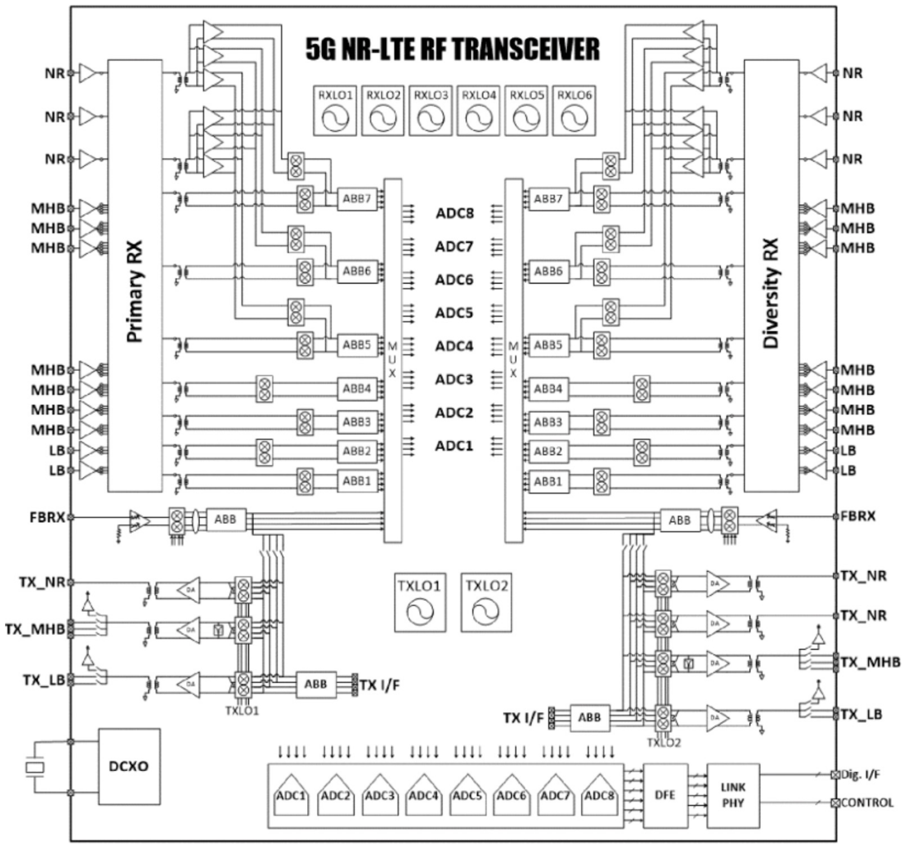
Seoul National University, Seoul, Korea

accellera
SYSTEMS INITIATIVE

Contents

- Motivation
- 5G/LTE RF Transceiver Model
- UVM Testbench for AMS Verification
- Simulation Results
- Summary

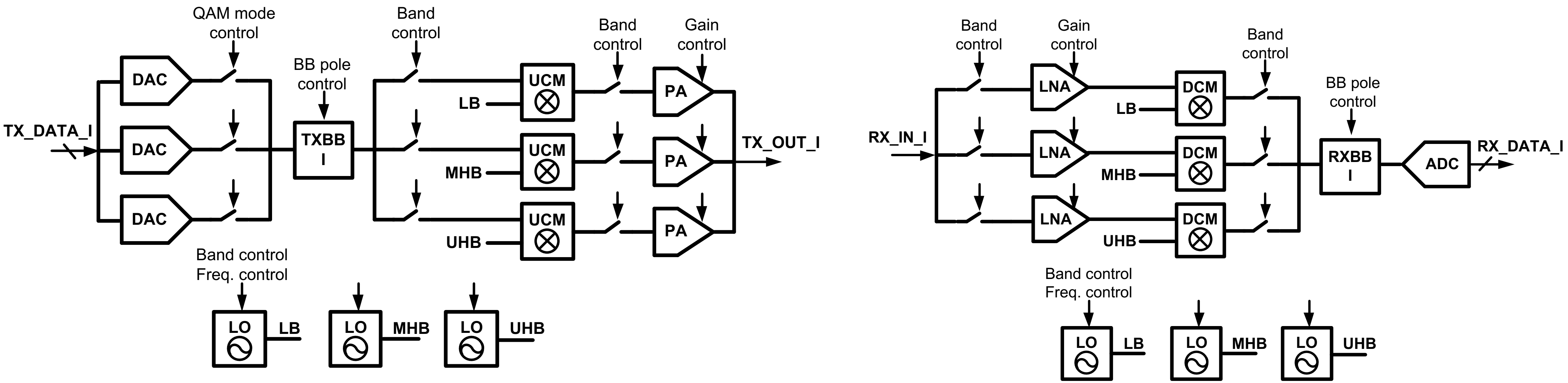
Motivation



- A 5G/LTE RF transceiver is a complex analog circuit with 351 operating modes
  - 117 frequency bands × 3 modulations
- Thorough verification is needed to avoid:
  - Errors in selecting the active blocks
  - Errors in routing the signals
  - Errors in decoding the control bits
- This work aims to use UVM testbench to perform such verification

J. Lee, et al., "A Sub-6GHz 5G New Radio RF Transceiver Supporting EN-DC with 3.15Gb/s DL and 1.27Gb/s UL in 14nm FinFET CMOS," *ISSCC* 02/2019.

Modeling RF Transceiver in SystemVerilog



- To use UVM, we need to first model the RF transceiver in SystemVerilog
- Real-number model (RNM) and baseband equivalent model (BBEQ) are not adequate when simulating RF signals at variable carrier frequencies

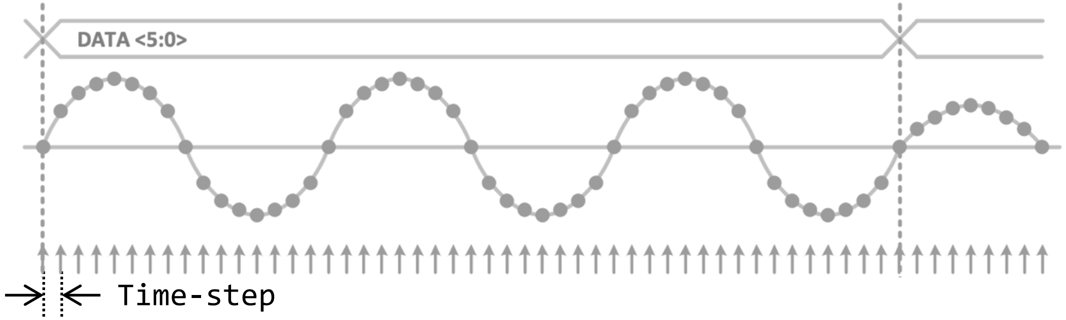
C. Y. Park and J. Kim, "Event-Driven Modeling and Simulation of 5G NR-Band RF Transceiver in SystemVerilog," SMACD 07/2021.

Problems with RNM and BBEQ

Real Number Model (RNM)

- Expressing high-frequency RF signals would require many events triggered at fine time steps, resulting in slow simulation

```
parameter time_step = 10e-12        // time-step size
always @(posedge clk) begin
    t = $realtime;
    sin_out = amp * $sin(2*`M_PI*freq*t*time_step);
end
```



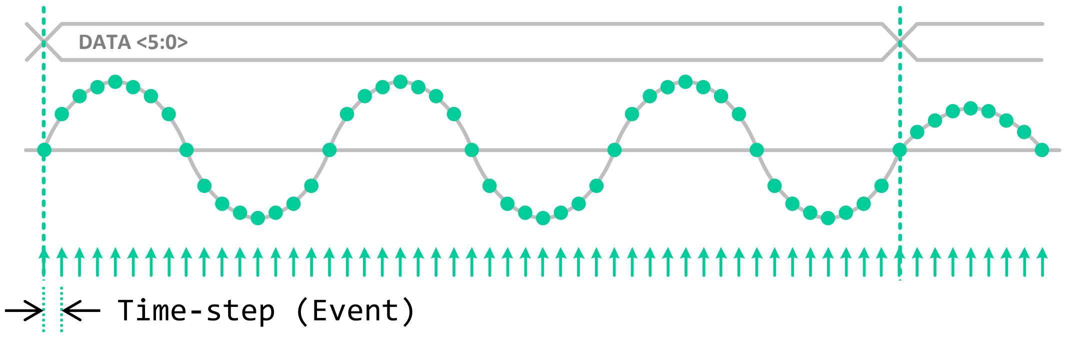
Baseband Equivalent Model (BBEQ)

- Can reduce the events by assuming a fixed-frequency carrier, but cannot express RF signals with variable or multiple carrier frequencies

XMODEL : Efficient Event-Driven Simulation

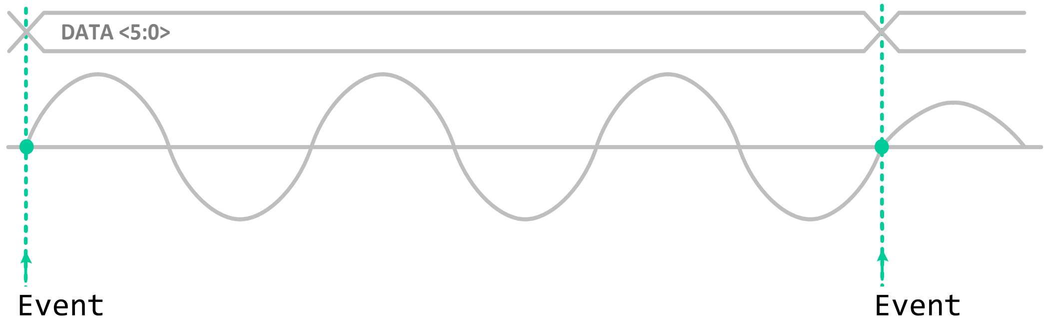
- Fast and accurate analog/mixed-signal simulation in SystemVerilog by expressing analog signals using functional expressions
  - $x(t) = \sum_i c_i t^{m_i - 1} e^{-a_i t} u(t) \rightarrow X(s) = \sum_i \frac{c_i}{(s+a_i)^{m_i}}$

RNM's event-points




```
parameter time_step = 10e-12          // time-step size
always @(posedge clk) begin
    t = $realtime;
    sin_out = amp * $sin(2*`M_PI*freq*t*time_step);
end
```

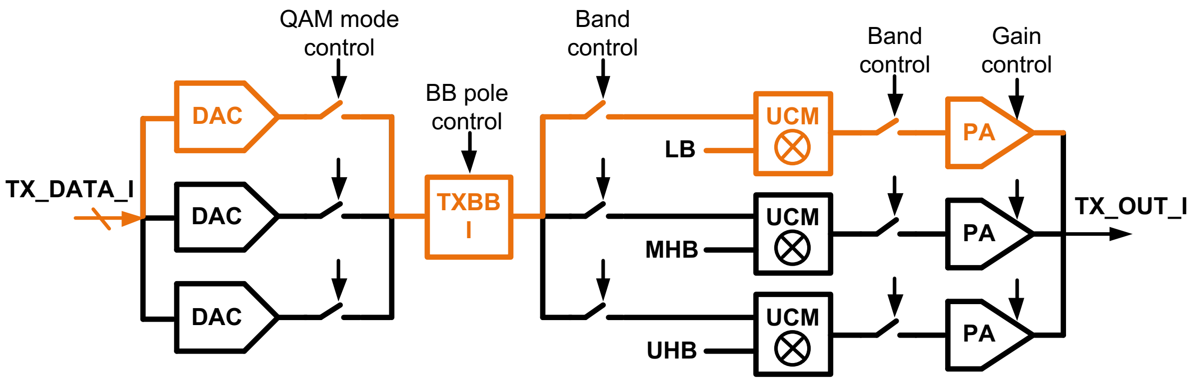
XMODEL's event-point




```
sin_gen  #(.freq(freq)) XP1 (.out(sin_unit));
multiply XP2 (.in(amp, sin_unit), .out(sin_out));
```

Modeling RF Transceiver with XMODEL

- Each component of the RF transceiver is modeled by XMODEL primitives that describe the circuit's functionalities

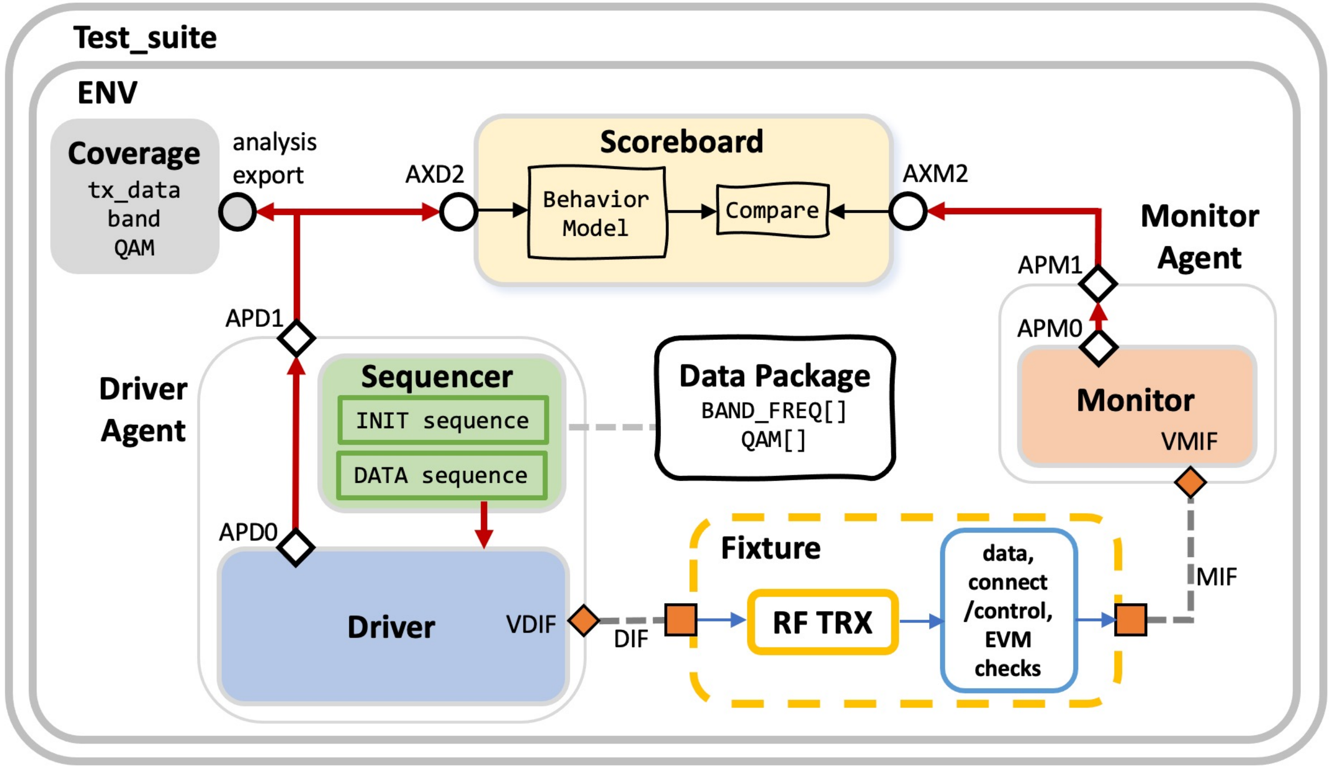


Transmitter I path

```
xreal        DAC_out, BB_in, BB_out, MIX_in, MIX_out,
             PA_in, PA_out;

// D/A converter
dac          #(.num_bit(4))
             XP1 (.out(DAC_out), .in(TX_DATA));
switch       SW1 (.pos(DAC_out), .neg(BB_in), .ctrl(sel_qam[0]));

// TX baseband (BB) filter
filter_var  #(.num_poles(1))
            XP2 (.gain(gain), .poles('{pole,0.0}),
                 .out(BB_out), .in(BB_in));
switch      SW2 (.pos(BB_out), .neg(MIX_in), .ctrl(sel_band[0]));

// Up-conversion mixer
multiply    #(.num_in(2))
            XP3 (.out(MIX_out), .in({net5, MIX_in}));
switch      SW3 (.pos(MIX_out), .neg(PA_in), .ctrl(sel_band[0]));

// Power amplifier
multiply    #(.num_in(2))
            XP4 (.out(PA_out), .in({ctrl_PA_gain, PA_in}));
```

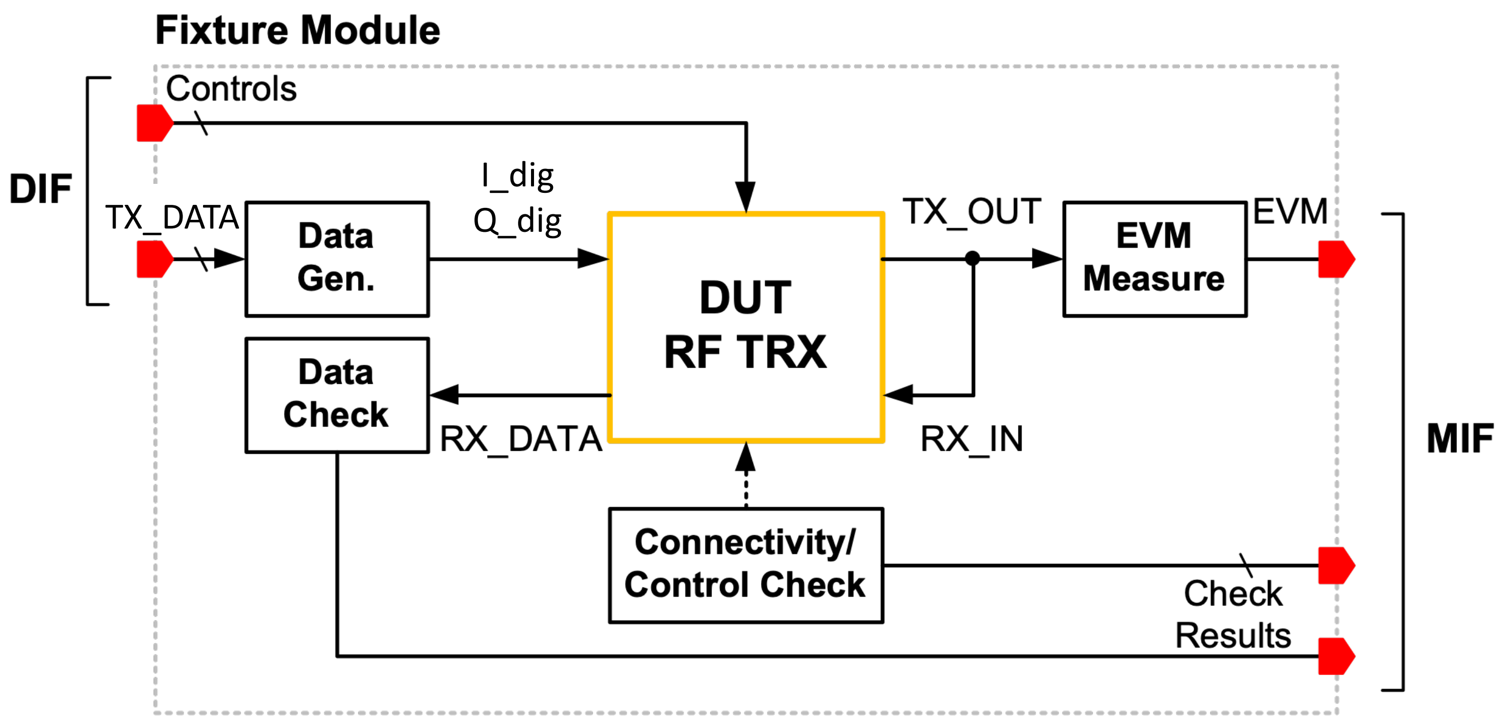
UVM Testbench for Analog/Mixed-Signal DUT

- By encapsulating all the analog instrumentations within a fixture module, the rest of the testbench can be built using standard UVM components



Fixture Module

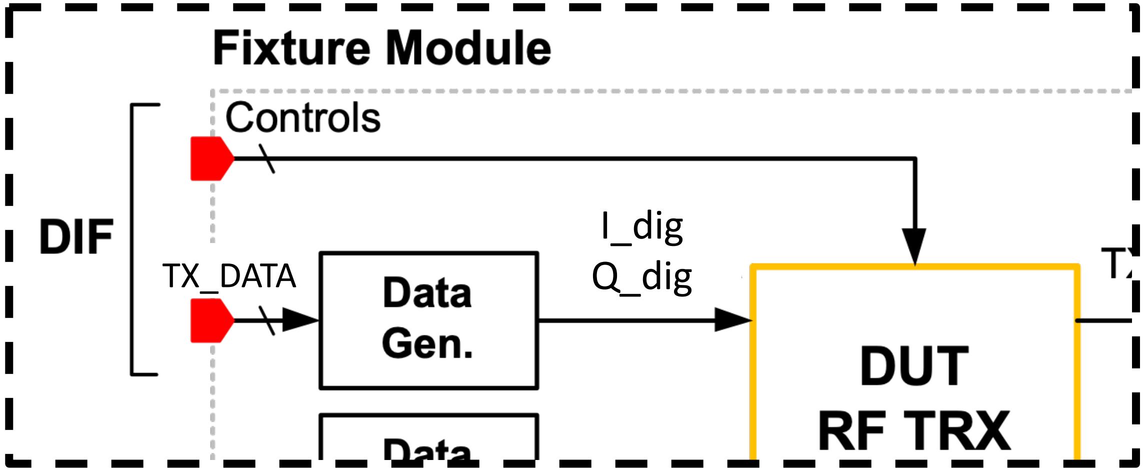
- Contains XMODEL primitives for supplying stimuli and measure responses
- Connects to the UVM driver and monitor via SV virtual interfaces



- Proposed fixture module performs:
  - Data generation
  - Data check
  - Connectivity/control check
  - EVM measurement

Data Generation

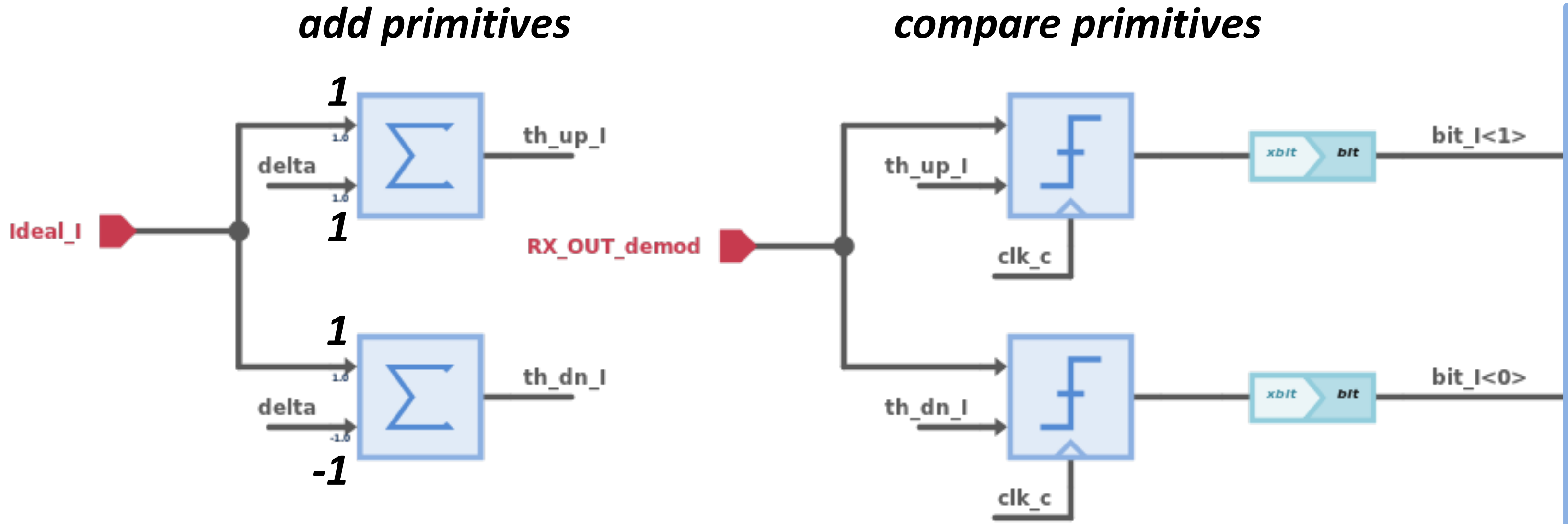
- Emulates the OFDM modulator by converting the digital data entering via DIF to a pair of I/Q digital streams for a single OFDM sub-channel carrier
  - Instead of using a full FFT processor




```
always @(posedge OFDM_clk) begin
  I_bit = TX_PKT.TX_DATA[4:0];
  Q_bit = TX_PKT.TX_DATA[9:5];
  I_mod += I_bit * A_I * cos(2*M_PI*k/N);
  Q_mod += Q_bit * A_Q * sin(2*M_PI*k/N);
  k++;

  if (k == N) begin
    I_dig = int'(I_mod / N / LSB);
    Q_dig = int'(Q_mod / N / LSB);
    k = 0;
  end
end
```

Data Check

- Checks whether the received signal after de-modulation is within the expected range given the transmitted data (ideal value $\pm \delta$)

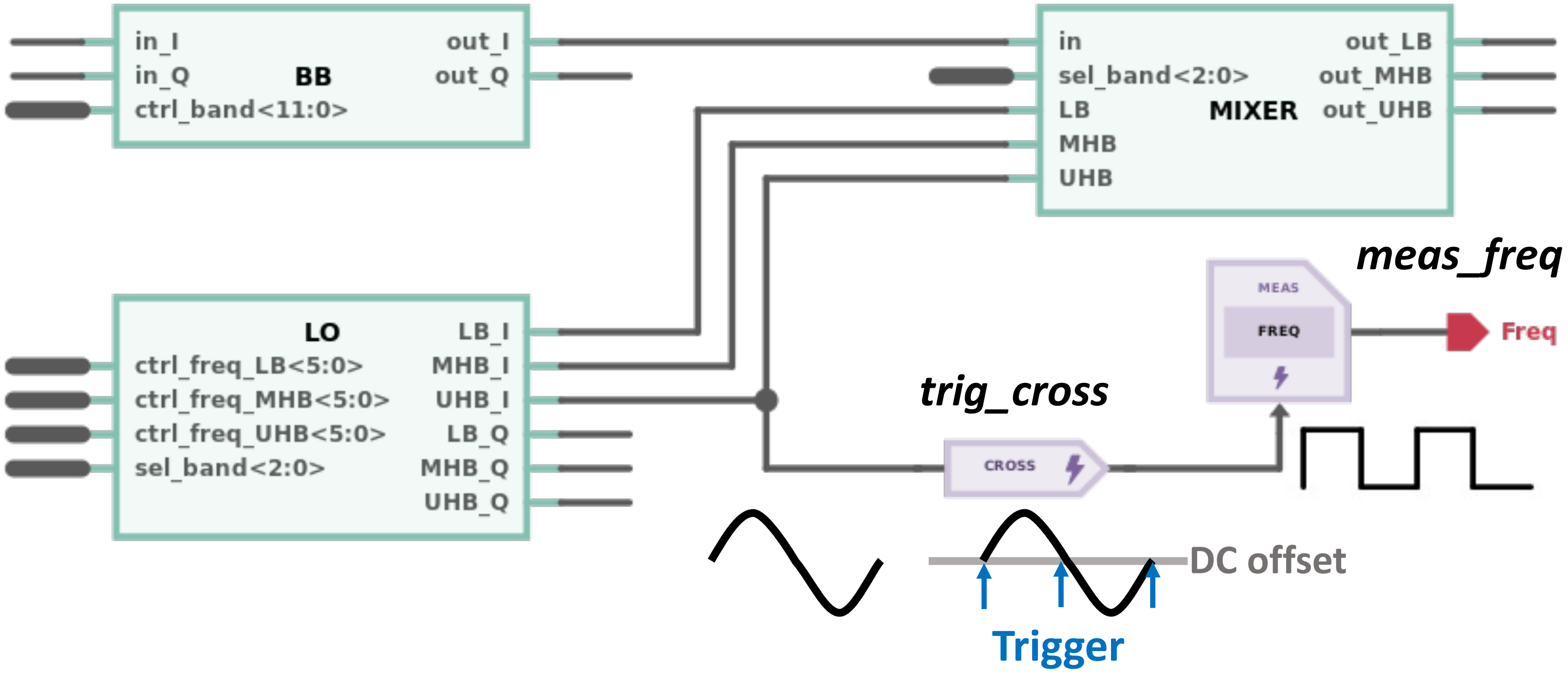


inline primitive

```
// DATA CHECK
module err_check (
     input [1:0] bit_I,
     input clk,
     output err_I
);
bit [1:0] bit_ref;
initial begin
    bit_ref <= 2'b01;
    err_I <= 1'b0;
end
always @(posedge clk) begin
    err_I <= bit_ref - bit_I;
end
endmodule
```

Connectivity Check

- Checks whether the proper LO signal is selected and routed by checking the carrier frequency of the RF signals



Connectivity Check (2)

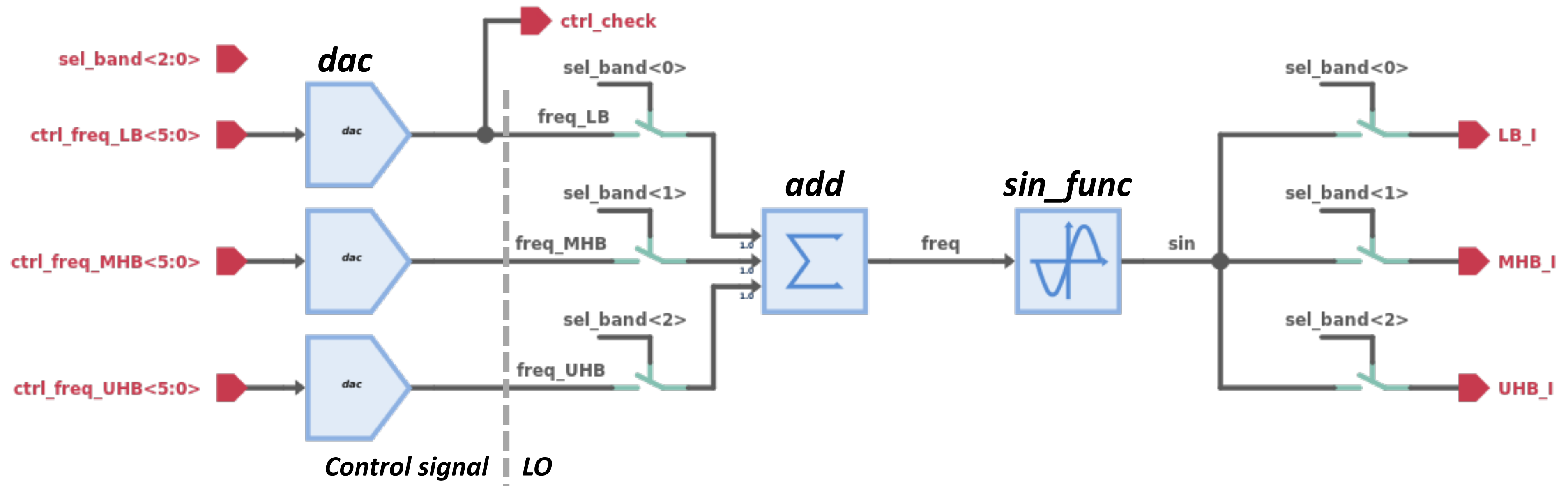
```
property LB_I_connection;
  @(posedge CALIB)
  $rose(CTRL_IN.sel_band_bit == 3'b001) |->
     0.4e9 <= LO_LB_I_freq &&
     LO_LB_I_freq <= 1.0e9;
endproperty: LB_I_connection

...
```

```
always @(posedge CALIB) begin
  if (CTRL_IN.sel_band_bit == 3'b001) begin
    LB_I: cover property (LB_I_connection);
    LB_I_CONN:
      assert property (LB_I_connection)
        uvm_report_info("LB I:PASS", UVM_HIGH);
      else
        uvm_report_info("LB I:FAIL", UVM_LOW);
  end
  ...
end
```

- A set of SV properties and assertion checks perform the connectivity checks by checking if the carrier frequency of each RF signal is within the expected range set by the selected band

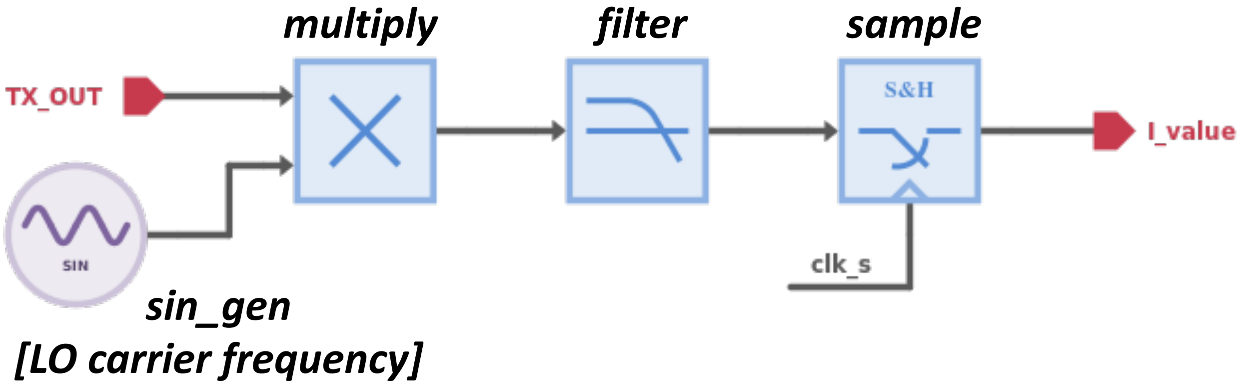
Control Signal Check

- Check whether the digital control signals are being interpreted correctly
  - Each component takes the digital control bits and converts to an analog value expressing LO frequency, amplifier gain, filter bandwidth, …
  - This analog value is tapped and sent to the UVM monitor



Error Vector Magnitude (EVM) Measurement

- Measures the quality of the QAM-modulated, transmitted signal
  - Error vector magnitude (EVM) $= \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} I_{err}[n]^2 + Q_{err}[n]^2 / Ref} \times 100\%$



inline

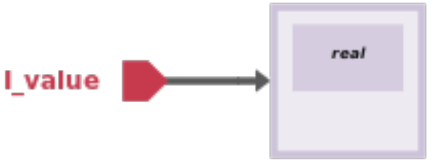
I_value, Q_value, Ideal_I, Ideal_Q, clk → EVM

```
// Calculate EVM
module cal_EVM (
    input real I_value,
    input real Q_value,
    input real Ideal_I,
    input real Ideal_Q,
    input reg clk,
    output real EVM
);

always @(posedge clk) begin
    EVM = ((Ideal_I – I_value)**2 +
           (Ideal_Q – Q_value)**2)**0.5;
end

endmodule
```

QAM Constellation Diagram Measurement

- A Python script plots the QAM constellation diagram after the simulation

probe_real



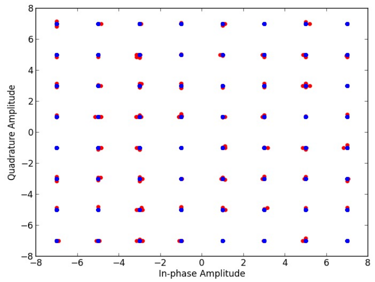
waveform file

EVM_results
.fsdb

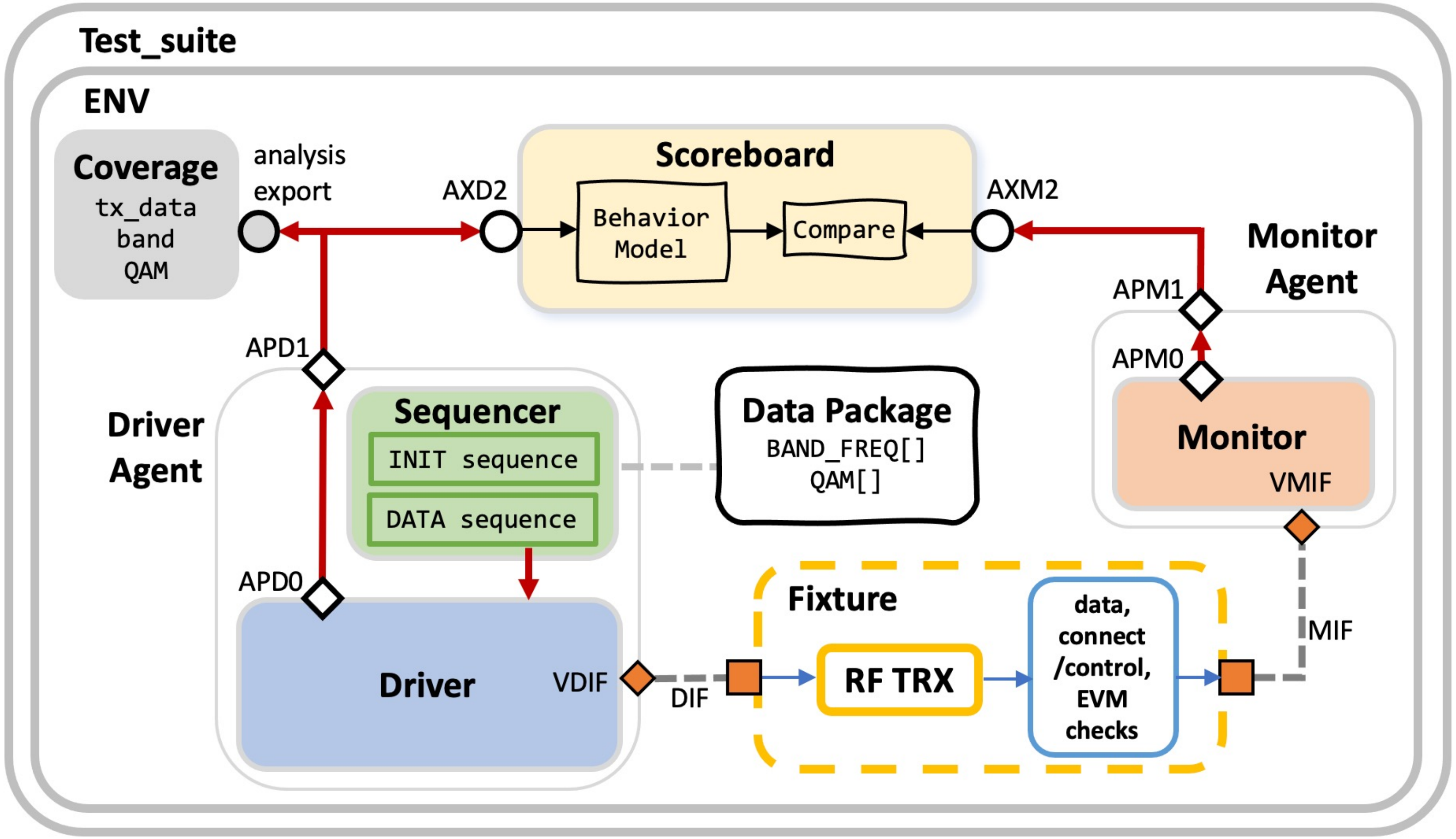
Python script

```
import matplotlib.pyplot as plt
from xmulan import mulan, rowml, waveform

filepath = './EVM_results.fsdb'

# read data from waveform
row = rowml()
row.readmeas(filepath)
wv1 = waveform(row['%s.%s' % (FIXTURE, I_value)])

# compute QAM constellation points
QAM_Imeas, QAM_Qmeas = compute_constellation(wv1)
QAM_Iref, QAM_Qref = compute_ref()

# plot graph
plt.scatter(QAM_Imeas, QAM_Qmeas, color='r')
plt.scatter(QAM_Iref, QAM_Qref, color='b')
plt.xlabel('In-phase Amplitude')
plt.ylabel('Quadrature Amplitude')
plt.show()
```

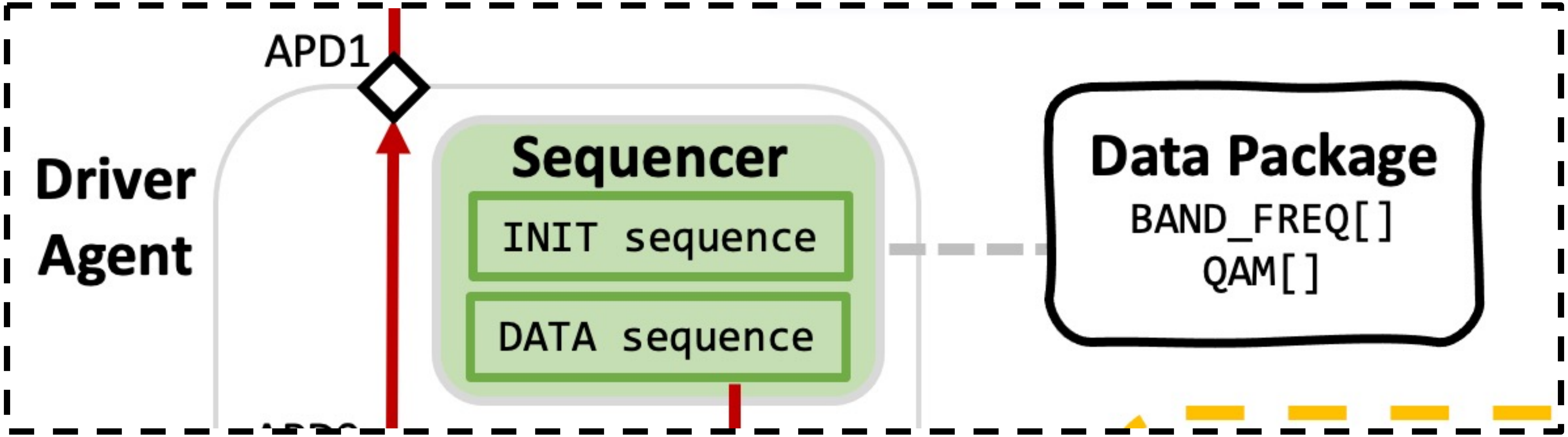
QAM



UVM Components for the Testbench



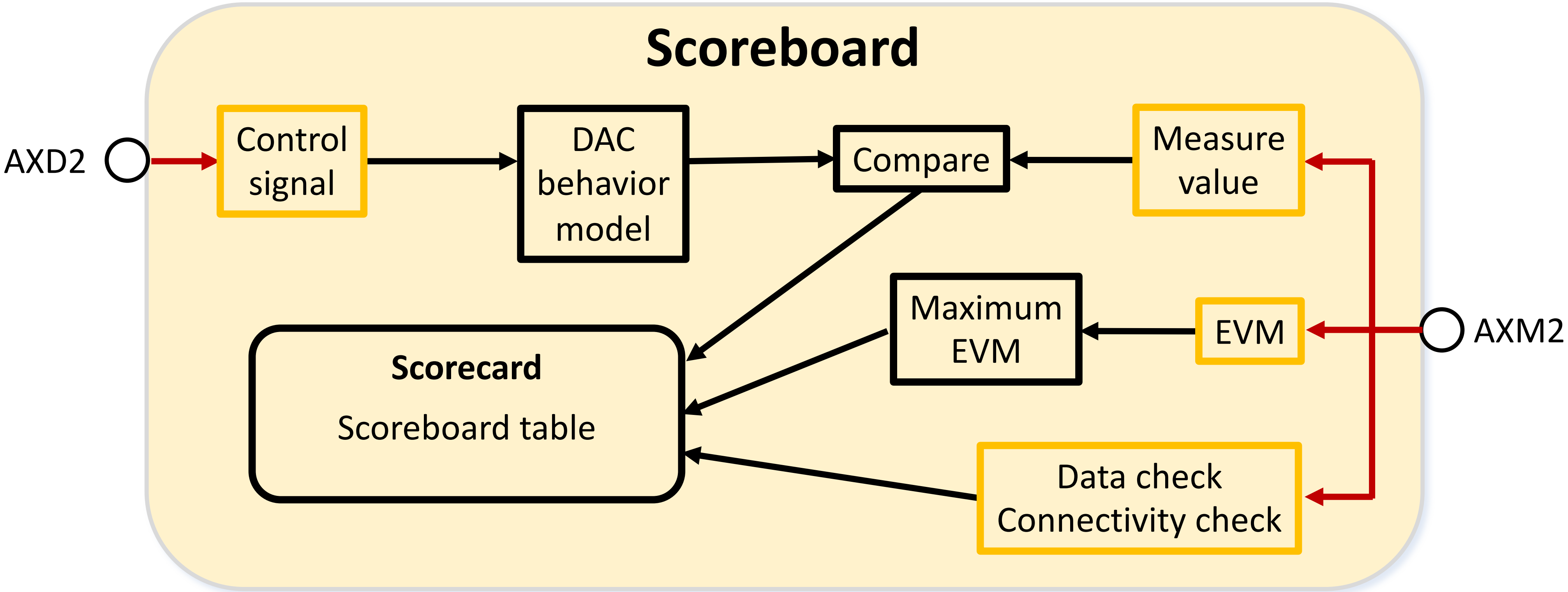
Sequence and Sequencer Components



- Generates the data and control inputs to be fed to the fixture module
  - INIT sequence performs initialization
  - DATA sequence generates random data while exercising multiple operating modes defined in the data package

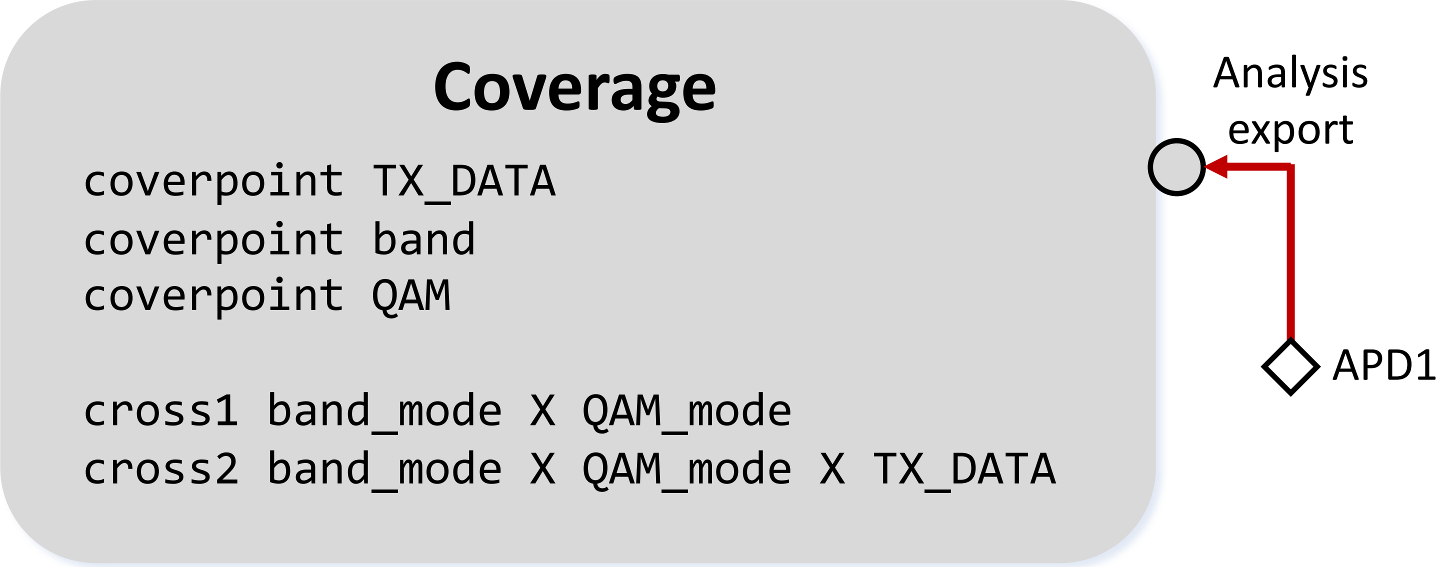
Scoreboard Component

- Collects all the validation/measurement results and prints a report



Subscriber Component for Coverage Check

- Listens to the data/control signals being generated and measures the coverage for data values, band frequencies, and modulation modes
  - Also measures the cross coverages among them (**`cross1`** and **`cross2`**)



Simulation Results: UVM Output Log

BAND NAME	BAND MODE	QAM MODE	I/Q P/F	I O/X	Q O/X	MAX EVM
N46	UHB	QAM_1024	PASS	O	O	0.482522
LTE_BAND2	MHB	QAM_64	PASS	O	O	1.844600
N75	MHB	QAM_256	PASS	O	O	0.989446
LTE_BAND19	LB	QAM_1024	PASS	O	O	0.482271
N24	MHB	QAM_256	PASS	O	O	0.961091
N40	MHB	QAM_64	PASS	O	O	2.151301
N70	MHB	QAM_256	PASS	O	O	1.057623
N25	MHB	QAM_64	PASS	O	O	1.926195
N5	LB	QAM_1024	PASS	O	O	0.470259
N84	MHB	QAM_256	PASS	O	O	0.983086
N76	MHB	QAM_256	PASS	O	O	0.923541
N40	MHB	QAM_64	PASS	O	O	2.060640
N53	MHB	QAM_256	PASS	O	O	0.930971
N92_H	MHB	QAM_256	PASS	O	O	1.107136
LTE_BAND2	MHB	QAM_256	PASS	O	O	0.976312
LTE_BAND30	MHB	QAM_64	PASS	O	O	1.575720
N85	LB	QAM_256	PASS	O	O	0.832701
N86	MHB	QAM_1024	PASS	O	O	0.472347
N47	UHB	QAM_1024	PASS	O	O	0.483332
LTE_BAND5	LB	QAM_64	PASS	O	O	2.462184
N91_L	LB	QAM_256	PASS	O	O	0.907705
N102	UHB	QAM_256	PASS	O	O	1.138304
LTE_BAND69	MHB	QAM_64	PASS	O	O	1.147057
N86	MHB	QAM_256	PASS	O	O	0.845882

```
## [CHECKER] DAC QAM AI : PASS  (351/351)
## [CHECKER] DAC QAM AQ : PASS  (351/351)
## [CHECKER] TX BB I : PASS  (351/351)
## [CHECKER] TX BB Q : PASS  (351/351)
## [CHECKER] LO Band Frequency : PASS  (351/351)
    LO Frequency LB  : PASS
    LO Frequency MHB : PASS
    LO Frequency UHB : PASS
## [CHECKER] UCM I : PASS  (351/351)
## [CHECKER] UCM Q : PASS  (351/351)
## [CHECKER] PA I : PASS  (351/351)
## [CHECKER] PA Q : PASS  (351/351)
## [CHECKER] LNA : PASS  (351/351)
## [CHECKER] DCM I : PASS  (351/351)
## [CHECKER] DCM Q : PASS  (351/351)
## [CHECKER] RX ABB I : PASS  (351/351)
## [CHECKER] RX ABB Q : PASS  (351/351)
## [CHECKER] ADC I : PASS  (351/351)
## [CHECKER] ADC Q : PASS  (351/351)
```

```
   ---COVERAGE STATISTICS----
BAND QAM 64    Coverage: 100.00000%
BAND QAM 256   Coverage: 100.00000%
BAND QAM 1024  Coverage: 100.00000%
DATA QAM 64    Coverage: 94.87180%
DATA QAM 256   Coverage: 94.83841%
DATA QAM 1024  Coverage: 95.03455%
```

Coverage Results

- 100% coverage can be reached more easily by combining the results of multiple simulations with different seed values

```
---COVERAGE STATISTICS----
BAND QAM 64   Coverage: 100.00000%
BAND QAM 256  Coverage: 100.00000%
BAND QAM 1024 Coverage: 100.00000%
DATA QAM 64   Coverage: 94.87180%
DATA QAM 256  Coverage: 94.83841%
DATA QAM 1024 Coverage: 95.03455%
```

Seed value = 418

```
---COVERAGE STATISTICS----
BAND QAM 64   Coverage: 100.00000%
BAND QAM 256  Coverage: 100.00000%
BAND QAM 1024 Coverage: 100.00000%
DATA QAM 64   Coverage: 94.61806%
DATA QAM 256  Coverage: 95.01870%
DATA QAM 1024 Coverage: 95.03575%
```

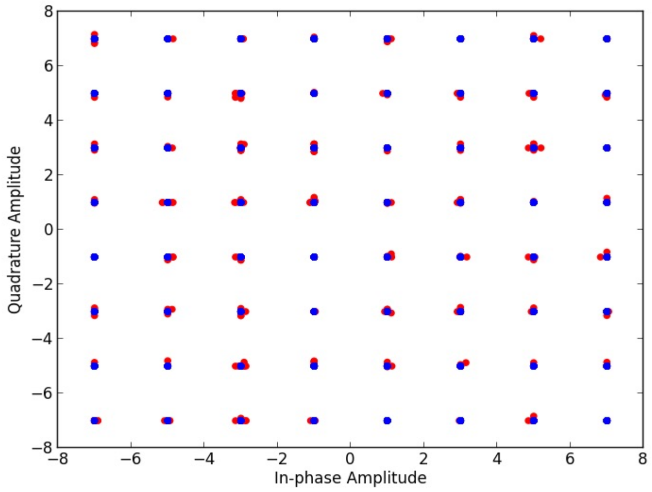
Seed value = 911

Merge

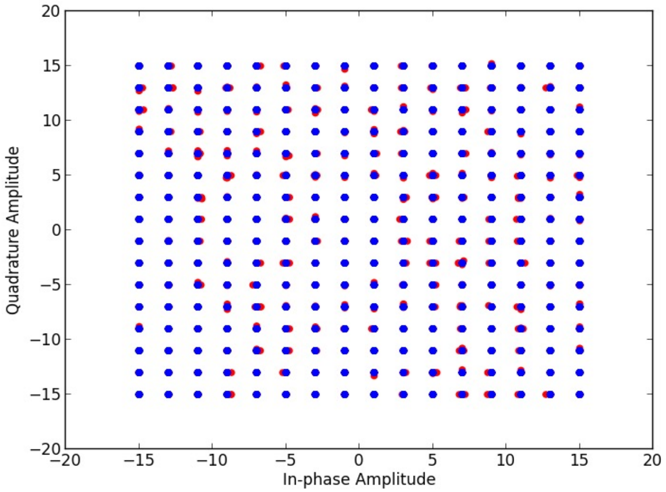
Crosses for Group RF_COV_PKG::COVERAGE::CVG

CROSS	EXPECTED	UNCOVERED	COVERED	PERCENT	GOAL
cross_64	117	0	117	100.00	100
cross_256	117	0	117	100.00	100
cross_1024	117	0	117	100.00	100
cross_64_tx	7488	0	7488	100.00	100
cross_256_tx	29952	0	29952	100.00	100
cross_1024_tx	119808	0	119808	100.00	100

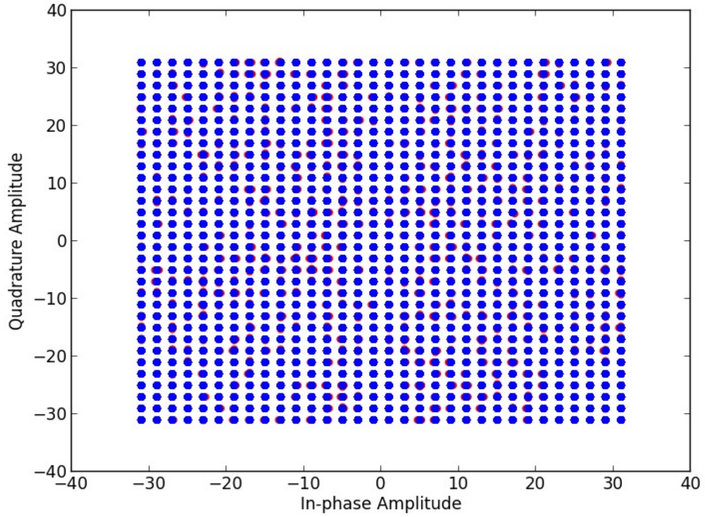
QAM Constellation Diagrams



64QAM in LTE band 11



256QAM in LTE band 25



1024QAM in LTE band 53

- Worst-case EVM: 2.3% (64-QAM), 1.4% (256-QAM), and 0.7% (1024-QAM)
- Satisfying the 3GPP standard at all bands

Summary

- This work presented a UVM testbench for verifying a multi-standard RF transceiver across all of its operating modes
  - The RF transceiver is modeled in SystemVerilog using XMODEL primitives
  - With the fixture module enclosing all the analog specifics, the UVM components built for digital verification can be extended to AMS verification as well

- The presented UVM testbench successfully completed the data checks, connectivity/control checks, and EVM measurements on the 5G/LTE RF transceiver model over 351 operating modes in 3.3 hours

Thank You