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

Byeong Kyu Kim and Jaeha Kim
Seoul National University, Seoul, Korea
bkkim@mics.snu.ac.kr, jaeha@snu.ac.kr

Abstract - This work presents a UVM SystemVerilog testbench for verifying the functionalities of a multi-standard RF transceiver (TRX) with 351 operating modes. To thoroughly verify a highly-reconfigurable analog/mixed-signal design using the standard UVM components, the proposed testbench encapsulates all the analog specifics of the DUT and its instrumentations in a fixture module described with XMODEL primitives. The testbench generates a test sequence that randomly enumerates the operating modes and performs data check, connectivity check, and control signal check for each of the operating modes. 100% coverage is achieved by combining four independent simulations with different seed values.

I. INTRODUCTION

The key challenge in verifying a multi-standard RF transceiver (TRX) like the one described in [1] is that its correct operation must be thoroughly checked for a large number of operating modes, resulting from the combination of multiple frequency bands, RF modulation schemes, local oscillator (LO) frequencies, power amplifier (PA) and low-noise amplifier (LNA) gain controls, etc., required to support all the specifications defined in the standards. A trivial mistake in selecting the active circuit blocks, routing the signals, or decoding the control bits can render the whole transceiver IC not functional [2].

This paper aims to achieve the full verification coverage of such a highly-reconfigurable analog/mixed-signal (AMS) system controlled by a large number of digital modes, utilizing the standardized components from the universal verification methodology (UVM) library. Specifically, a scalable and reusable UVM SystemVerilog testbench that can verify the functionalities of a 5G/LTE RF TRX model, including the error-free data transmission, propagation of the analog RF signals, and decoding the digital controls, over 117 5G/LTE bands and 3 RF modulation modes is presented.

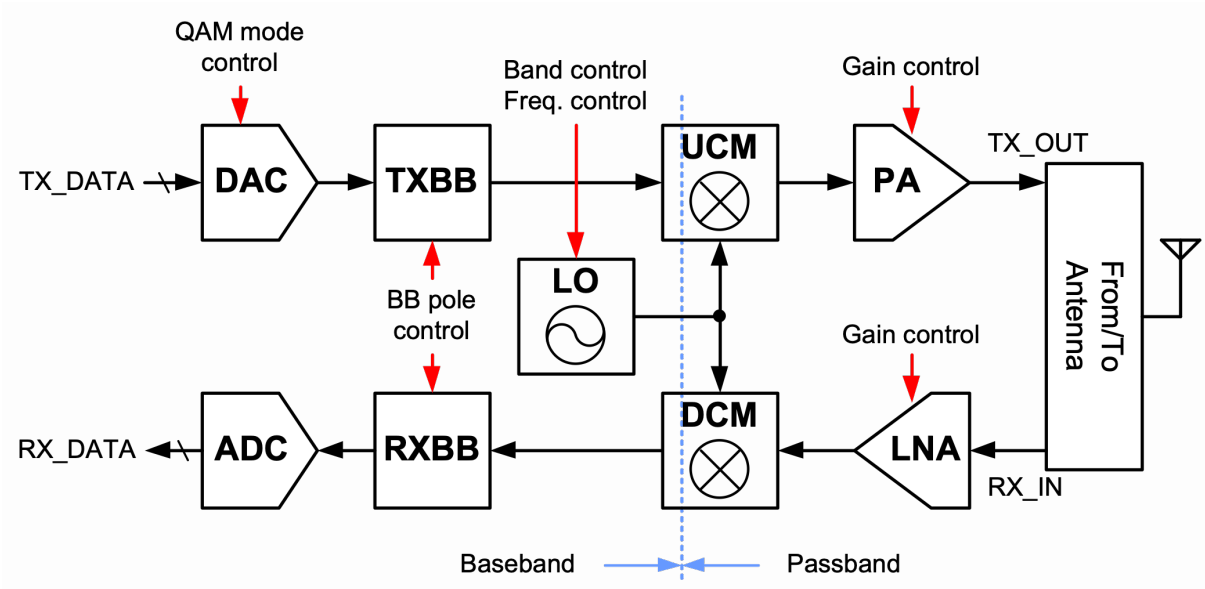


Figure 1. A block diagram of a 5G/LTE multi-standard RF transceiver [2].

II. Multi-Standard RF Transceiver Model

To carry out a verification with UVM, the device-under-test (DUT), in this case, the RF TRX circuit must be first modeled in SystemVerilog. Fig. 1 illustrates the overall block diagram of the RF TRX model presented in [3]. Its transmitter (TX) path consists of a digital-to-analog converter (DAC), a transmitter baseband circuit (TX BB), an up-conversion mixer block (UCM), and a power amplifier (PA) driving the off-chip antenna. On the other hand, the receiver (RX) path consists of a low-noise amplifier (LNA), a down-conversion mixer block (DCM), a receiver baseband circuit (RX BB), and an analog-to-digital converter (ADC) recovering the received data. Both the UCM and DCM blocks are driven by the carrier signals produced by the local oscillator (LO) block. Each block must support multiple operating modes, such as multiple ranges of LO frequencies, multiple modulation schemes, multiple amplifier gains, multiple filter cut-off frequencies, etc., depending on the choice of the RF standards (5G or LTE), bands (LB, MHB, or UHB), or modulations (64, 256, or 1024 QAMs), made at the system level. To do so, each block may contain multiple instances of the same circuit, each optimized for a specific operating condition, instead of having one covering all operating conditions. As a result, a modern RF TRX system can contain multiple local oscillators, multiple mixers, and multiple low-noise amplifiers, and only one of each set is selected active for a given operation mode. In addition, each block may contain digital calibration loops to compensate various non-idealities in the circuit, such as signal leakage, distortion, and gain or phase mismatches, which can contribute to the DC offsets, harmonics, and I/Q phase mismatch, respectively. Except for the initial input signal to the TX path (*TX_DATA*) and the final output signal of the RX path (*RX_DATA*), most of the signals propagating through the TRX paths are baseband or passband analog signals. On the other hand, the control signals selecting the operating mode of each block are all digital. The analog/mixed-signal modeling of this highly-reconfigurable RF TRX in SystemVerilog has been successfully demonstrated in [3], using a set of XMODEL primitives offered by Scientific Analog [4].

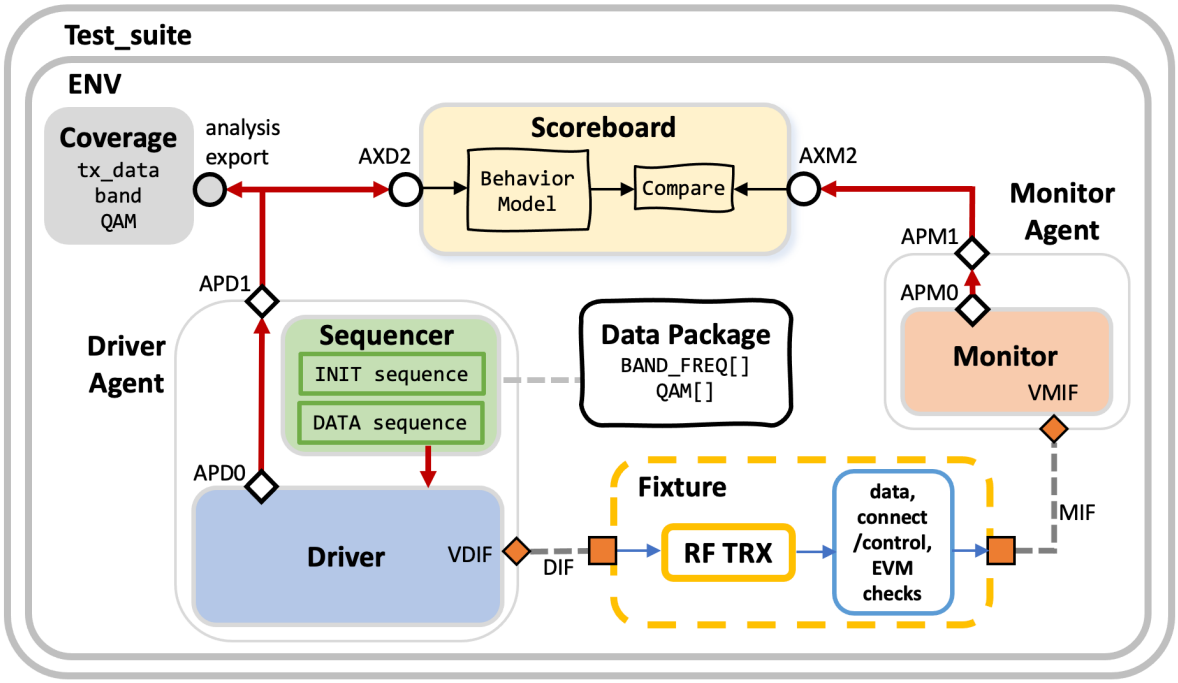


Figure 2. Organization of the UVM SystemVerilog testbench for a multi-standard RF transceiver.

III. UVM SystemVerilog Testbench

Fig. 2 illustrates the organization of the proposed UVM testbench to verify the described RF TRX model. Following the approach described in [5], all the analog-specific details are encapsulated within a *fixture* module, so that the rest of the testbench can be built using the standard UVM components only, including the driver agent, monitor agent, scoreboard, and coverage.

A. UVM Sequence

A UVM sequence component defines a series of test cases to be fed to the DUT. In the presented testbench, it generates a randomly-ordered sequence of operating modes and for each operating mode, it generates a set of randomized data bits to be transmitted.

To define a set of operating modes to be tested in a scalable way, a separate package file defines a set of 117 band/frequency modes and set of 3 modulation modes using the arrays named *BAND_FREQ*[] and *QAM*[], respectively, as listed in Fig. 3. The arrays also define the associated data values for each mode, such as the frequency control bits, band name, and number of modulation symbols supported.

Fig. 4 illustrates how the UVM sequence component is organized to generate a sequence of test cases. First, an instance of a *uvm_sequence_item*-derivative class named *TX_PKT* containing the indices for the band/frequency mode and modulation mode is created and randomized. The index values are defined as a *packed struct* type (*operation_bit*), so that the randomized value range of each index can be set by the size of the *BAND_FREQ*[] or *QAM*[] arrays. Second, using the randomized indices, the control bits selecting the band (*sel_band_bit*), the LO frequency (*ctrl_LO_bit*), and the modulation scheme (*sel_qam_bit*) are read from the *BAND_FREQ*[] and *QAM*[] arrays. Third, for each operation mode selected, a set of data bits to be transmitted are randomized. The sequence component transmits a total of *N* symbols per operation mode, where *N* is the product of the number of modulation symbols (*qam_mode*) and a user-defined constant *TRIALS*.

B. Driver Agent and Monitor Agent

The driver agent, including the sequencer and driver components, drives a sequence of data packets (*TX_PKT*) as triggered by a packet clock via the virtual interface named *VDIF*, connected to the interface named *DIF* of the fixture module. The sequencer alternates between two sequences: the initialization sequence and the data sequence explained in the previous subsection. During the initialization sequence, indicating that the band/frequency or modulation mode is about to change, the driver switches the control codes to the DUT and waits for its calibration loops to settle. During the data sequence, the driver starts driving the data bits to the DUT.

The monitor agent, on the other hand, receives a sequence of data packets via the virtual interface named *VMIF*, connected to the interface named *MIF* of the fixture module. The monitor component forwards the received data packet (*RX_PKT*) to the scoreboard component.

```
typedef struct{
  int band;
  bit [5:0] freq;
  string band_name;
} BAND_FREQ_MODE;

BAND_FREQ_MODE BAND_FREQ[] = '{
  '{2, 6'b100010, "LTE_BAND1"},
  '{2, 6'b011000, "LTE_BAND2"},
  '{2, 6'b010011, "LTE_BAND3"},
  '{2, 6'b001110, "LTE_BAND4"},
  ...
};
```

```
typedef struct {
  int qam_bit;
  int qam_mode;
} QAM_MODE;

QAM_MODE QAM[] = '{
  '{1, 64},
  '{2, 256},
  '{4, 1024}
};
```

Figure 3. The arrays defining the sets of band/frequency modes and QAM modes to be tested.

```
typedef struct packed {
  bit [9:0] band_bit;
  bit [9:0] qam_bit;
} operation_bit;

class PACKET extends uvm_sequence_item;
  randc operation_bit OP;
  rand bit [9:0] TX_DATA;
  ...
endclass: PACKET
```

(a)

```
task body();
  TX_PKT = PACKET::type_id::create("TX_PKT");
  TX_PKT.randomize() with {
    OP.band_bit inside {0:BAND_FREQ.size-1};
    OP.qam_bit inside {0:QAM.size-1};
  };

  TX_PKT.sel_band_bit = BAND_FREQ[TX_PKT.OP.band_bit].band;
  TX_PKT.ctrl_LO_bit = BAND_FREQ[TX_PKT.OP.band_bit].freq;
  TX_PKT.sel_qam_bit = QAM[TX_PKT.OP.qam_bit].qam_bit;
  TX_PKT.qam_mode = QAM[TX_PKT.OP.qam_bit].qam_mode;

  count = 0;
  while (count < TX_PKT.qam_mode*TRIALS) begin
    start_item(TX_PKT);
    TX_PKT.randomize(TX_DATA) with {
       TX_DATA inside {[0:QAM[TX_PKT.OP.qam_bit].qam_mode]};
    };
    finish_item(TX_PKT);
    count++;
  end
endtask: body
```

(b)

Figure 4. The definition of (a) the sequence item (i.e. packet) and (b) the body task of the sequence component.

C. Scoreboard

The scoreboard component collects the results contained in the packets from the driver and monitor agents. The packets from the driver (*TX_PKT*) contain the stimuli and the packets from the monitor (*RX_PKT*) contain the corresponding responses, such as the results of the data checks, connectivity checks, control signal checks, and EVM measurements. As for the control signal checks, the packets from the monitor contain the analog values that the DUT generated from the control bit values. The scoreboard compares them against the expected values to check whether the control bits are being interpreted correctly by the DUT. As for the EVM measurement, the scoreboard keeps the record of the worst-case EVM value for each operating mode.

The scoreboard component stores the results in a scorecard object, which not only contains an array named *DATA* to store the check results, but also defines a method to print the results in a table format.

D. Coverage

The coverage component collects the coverage metrics and measures whether all the operation modes and all the modulation symbols have been exercised during the simulation. Its class is derived from the *uvm_subscriber* class, which has a built-in analysis port to receive the packets from the driver agent. The coverage component also measures two cross coverage metrics: the cross coverage between the band and QAM modes, and the cross coverage among the data symbol, band, and QAM modes. The first one checks whether all the operating modes have been enumerated and the second one checks whether all the data symbols have been exercised for each operating mode. The total number of cases for the second cross coverage is 157,248, which is very large.

IV. FIXTURE MODULE

The fixture module illustrated in Fig. 5 contains the RF TRX model and the testbench instrumentations to supply its stimuli and measure its responses. As stated earlier, this fixture module is the only module that contains analog/mixed-signal model and testbench instrumentations described using the XMODEL primitives, so that the rest of the testbench can be built using the standard UVM components. The RF TRX DUT is operating in a loop-back mode. Via the interface *DIF*, the fixture module receives the control bits and randomized data bits from the driver agent. The fixture module then feeds the control bits directly to the DUT to select its operating mode. It also performs QAM modulation to drive the data input (*TX_DATA*) and QAM demodulation to check the data output (*RX_DATA*) of the DUT. In addition to this data check, the fixture module also performs the connectivity/control check, measures the error vector magnitude (EVM) of the RF output (*TX_OUT*), and sends all the results to the monitor agent via the interface *MIF*.

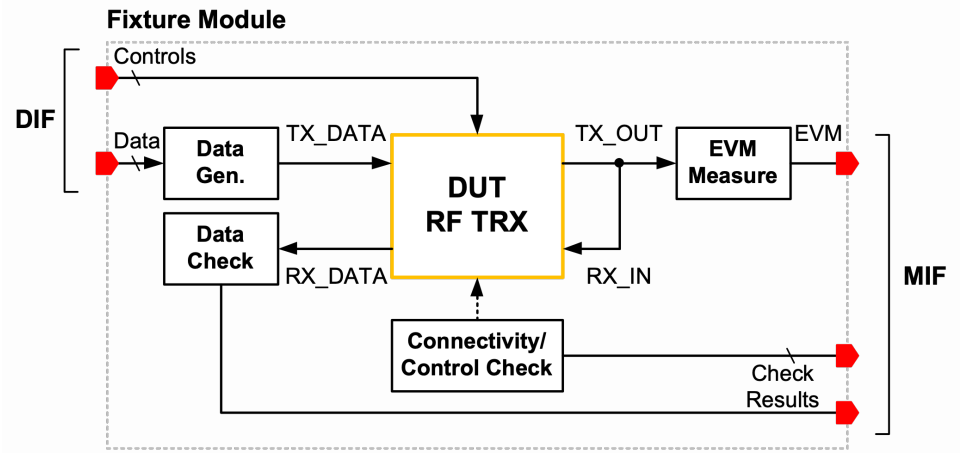


Figure 5. The fixture module encapsulating the device-under-test (DUT) and its instrumentations.

As for the data check, since the DUT model encompasses only the analog front-end (AFE) of the RF TRX, the fixture module must emulate some of the functionalities of the digital back-end processor, performing the orthogonal frequency-division multiplexing (OFDM) modulation and demodulation. For the purpose of verifying the AFE, it is deemed sufficient to emulate the data transmission via a single OFDM sub-channel only, transmitting and receiving a pair of I/Q data via quadrature amplitude modulation (QAM). The fixture module receives 10-bit randomized data from the driver agent (*TX_PKT.TX_DATA*), of which lower 5 bits indicate the I data value and upper 5 bits indicate Q data value. These I and Q values are modulated with the sub-channel I/Q carrier signals via analog multiplications, and their results are converted to digital to emulate the outputs of a OFDM baseband processor, and fed to the *TX_DATA* input of the RF TRX DUT. Fig. 6 lists the codes of the QAM data generator.

```
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
```

Figure 6. The QAM data generator emulating a single OFDM sub-channel.

On the other hand, when the RF TRX model returns the received data output RX_DATA, the fixture module performs QAM demodulation and extracts the received I/Q values. It compares these values against the transmitted I/Q values and determines if any transmission errors have occurred. The data check results are sent to the monitor agent.

The fixture module adds various assertion checks inside the blocks of the RF TRX model to check their connectivity based on the selected operating mode. These assertion checks are useful in catching trivial errors when selecting the active instance among multiple circuit instances within a block and when routing connections among the blocks, without having to examine the correctness of the data bits transmitted and received. Recognizing that most of the signals propagating between the blocks are amplitude-modulated RF signals of which carrier frequencies change with the selected band (*LB*, *MHB*, and *UHB*), the connectivity assertion checks measure the carrier frequency of each RF signal and check if it is within the correct range. Two XMODEL primitives are used to measure the carrier frequency of each RF signal: *trig_cross* and *meas_freq*. The *trig_cross* primitive generates a trigger signal indicating when the RF input signal crosses its DC offset value, and the *meas_freq* primitive measures the frequency of the trigger signal. Using the result, the assertion checks can examine if the frequency is within the correct range for the selected band.

Fig. 7 lists three assertion properties defined for a given RF signal, each checking the carrier frequency for LB, MHB, and UHB modes, respectively. Fig. 8(a) shows an excerpt of the report generated by the assertion checks during the SystemVerilog simulation. At the *UVM_HIGH* verbosity mode, the report displays *"PASS"* for each correct connection and *"FAIL"* for each incorrect connection. In addition, the cover properties are defined to measure whether these connection checks have been triggered during the simulation. An excerpt of the report in Fig. 8(b) generated after the simulation is completed displays the number of matches for each assertion check.

```
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

property MHB_I_connection;
  @(posedge CALIB)
  $rose(CTRL_IN.sel_band_bit == 3'b010) |->
     1.4e9 <= LO_MHB_I_freq &&
     LO_MHB_I_freq <= 2.7e9;
endproperty: MHB_I_connection

property UHB_I_connection;
  @(posedge CALIB)
  $rose(CTRL_IN.sel_band_bit == 3'b100) |->
     3.3e9 <= LO_UHB_I_freq &&
     LO_UHB_I_freq <= 6e9;
endproperty: UHB_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
  if (CTRL_IN.sel_band_bit == 3'b010) begin
    MHB_I: cover property (MHB_I_connection);
    MHB_I_CONN:
      assert property (MHB_I_connection)
        uvm_report_info("MHB I:PASS", UVM_HIGH);
      else
        uvm_report_info("MHB I:FAIL", UVM_LOW);
  end
  if (CTRL_IN.sel_band_bit == 3'b100) begin
    UHB_I: cover property (UHB_I_connection);
    UHB_I_CONN:
      assert property (UHB_I_connection)
        uvm_report_info("UHB I:PASS", UVM_HIGH);
      else
        uvm_report_info("UHB I:FAIL", UVM_LOW);
  end
  ...
end
```

Figure 7. The assertion checks for checking the connectivity of RF signals.

```
INITIAL
BAND = LTE_BAND38
UVM_INFO @ 40000: reporter [MHB I PASS]
UVM_INFO @ 40000: reporter [MHB Q PASS]
INITIAL
BAND = N85
UVM_INFO @ 865000: reporter [LB I PASS]
UVM_INFO @ 865000: reporter [LB Q PASS]
```

(a)

```
RF_UVM_TB.FIX.LB_I, 4307 attempts, 2 match
RF_UVM_TB.FIX.LB_Q, 4307 attempts, 2 match
RF_UVM_TB.FIX.MHB_I, 4307 attempts, 7 match
RF_UVM_TB.FIX.MHB_Q, 4307 attempts, 7 match
RF_UVM_TB.FIX.UHB_I, 4307 attempts, 1 match
RF_UVM_TB.FIX.UHB_Q, 4307 attempts, 1 match
```

(b)

Figure 8. (a) The assert property report and (b) cover property report generated by the simulation.

The fixture module also performs control signal checks. In other words, it verifies whether various digital control bits adjusting the analog properties of the circuit are correctly encoded to carry the information. These control checks are precautionary measures to prevent trivial mistakes such as bit-order mismatch, encoding mismatch (e.g. two's complement vs. sign-magnitude), polarity inversion, etc. Assuming that each block within the RF TRX model uses a kind of digital-to-analog converter (DAC) to convert the digital control code to an equivalent analog value, the fixture module captures the outputs of these DACs and send them to the monitor agent. The scoreboard then compares the values against the expected values and determines if the check passes or fails.

Finally, the fixture module measures the EVM and collects the traces for the QAM constellation diagram as the quality of the transmitted signal. To do so, the fixture module performs an ideal QAM demodulation on the transmitted output (*TX_OUT*) using an effective carrier signal, of which frequency is equal to the sum of the LO carrier frequency and OFDM sub-channel carrier frequency. The demodulated I/Q values are compared against the intended I/Q values. Computing the root-mean-squared value of these differences yields the EVM. Also, the traces of the demodulated I/Q values are recorded into a file, which is then post-processed after the simulation to plot the QAM constellation diagram.

V. RESULTS

The described UVM testbench was used to verify the RF TRX model operating over 117 5G/LTE bands and 3 QAM modulation modes, resulting in a total of 351 operating modes. Fig. 9 shows the excerpts of the simulation log and scoreboard report generated by the simulation, reporting the results of the I/Q data checks, connectivity checks, and control signal checks. The worst-case EVM values for each modulation mode over all the frequency bands are 2.3% for 64-QAM, 1.4% for 256-QAM, and 0.7% for 1024-QAM, all satisfying the 3GPP standard.

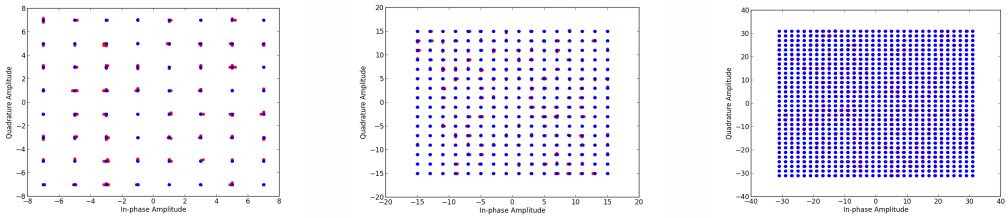
Fig. 10 shows the QAM constellation diagrams plotted for 64-QAM (LTE band 11), 256-QAM (LTE band 25), and 1024-QAM (LTE band 53) modes. For each constellation diagram, the blue and red dots represent the reference and transmitted I/Q symbols, respectively. All the red dots are located sufficiently near the blue dots, confirming that the EVMs are within the satisfactory range.

To reach 100% coverage sooner, it is common to run multiple simulations with different random seeds and merge their results. As an example, Fig. 11 shows two coverage reports each with the seed value of 418 and 911, respectively (*TRIALS*=3). In the report, the BAND-QAM coverage means the cross coverage between the band and QAM modes and the DATA-QAM coverage means the cross coverage among the data symbol, band, and QAM modes. Fig. 12 shows the final coverage report merging the results from four simulations with different seed values. In the table, the *cross_64*, *cross_256*, and *cross_1024* entries report the BAND-QAM coverages and *cross_64_tx*, *cross_256_tx*, *cross_1024_tx* entries report the DATA-QAM coverages.

```
## [CHECKER] DAC QAM AI : PASS  (351/351)
    DAC AI QAM 64   : PASS
    DAC AI QAM 256  : PASS
    DAC AI QAM 1024 : PASS
## [CHECKER] DAC QAM AQ : PASS  (351/351)
    DAC AQ QAM 64   : PASS
    DAC AQ QAM 256  : PASS
    DAC AQ QAM 1024 : PASS
## [CHECKER] TX BB I : PASS  (351/351)
    TX BB I DAC  : PASS
## [CHECKER] TX BB Q : PASS  (351/351)
    TX BB Q DAC  : PASS
## [CHECKER] LO Band Frequeny : PASS  (351/351)
    LO Frequeny LB  : PASS
    LO Frequeny MHB : PASS
    LO Frequeny UHB : PASS
## [CHECKER] UCM I : PASS  (351/351)
    UCM I Connect : PASS
## [CHECKER] UCM Q : PASS  (351/351)
    UCM Q Connect : PASS
## [CHECKER] PA I : PASS  (351/351)
    PA I Connect : PASS
    PA I LB Gain3  : PASS
    PA I MHB Gain3 : PASS
    PA I UHB Gain3 : PASS
    PA I LB Gain1  : PASS
    PA I MHB Gain1 : PASS
    PA I UHB Gain1 : PASS
```

BAND NAME	BAND MODE	QAM MODE	I/Q P/F	I O/X	Q O/X	MAX EVM
N46	UHB	QAM_1024	PASS	0	0	0.482522
LTE_BAND2	MHB	QAM_64	PASS	0	0	1.844600
N75	MHB	QAM_256	PASS	0	0	0.989446
LTE_BAND19	LB	QAM_1024	PASS	0	0	0.482271
N24	MHB	QAM_256	PASS	0	0	0.961091
N40	MHB	QAM_64	PASS	0	0	2.151301
N70	MHB	QAM_256	PASS	0	0	1.057623
N25	MHB	QAM_64	PASS	0	0	1.926195
N5	LB	QAM_1024	PASS	0	0	0.470259
N84	MHB	QAM_256	PASS	0	0	0.983086
N76	MHB	QAM_256	PASS	0	0	0.923541
N40	MHB	QAM_64	PASS	0	0	2.060640
N53	MHB	QAM_256	PASS	0	0	0.930971
N92_H	MHB	QAM_256	PASS	0	0	1.107136
LTE_BAND2	MHB	QAM_256	PASS	0	0	0.976312
LTE_BAND30	MHB	QAM_64	PASS	0	0	2.575720
N85	LB	QAM_256	PASS	0	0	0.832701
N86	MHB	QAM_1024	PASS	0	0	0.472347
N47	UHB	QAM_1024	PASS	0	0	0.483332
LTE_BAND5	LB	QAM_64	PASS	0	0	2.462184
N91_L	LB	QAM_256	PASS	0	0	0.907705
N102	UHB	QAM_256	PASS	0	0	1.138304
LTE_BAND69	MHB	QAM_64	PASS	0	0	2.147057

Figure 9. Excerpts of the simulation log (left) and scoreboard report (right).



(a) (b) (c)

Figure 10. The simulated QAM constellation diagrams for (a) 64 QAM, (b) 256 QAM and (c) 1024 QAM.

```
    ---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%
```

(a)

```
    ---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: 95.05375%
```

(b)

Figure 11. Two coverage reports with TRIALS =3: (a) seed value = 418 and (b) seed value = 911.

Crosses for Group RF_COV_PKG::COVERAGE::CVG

CROSS	EXPECTED	UNCOVERED	COVERED	PERCENT	GOAL	WEIGHT	AT LEAST	PRINT MISSING	COMMENT
cross_64	117	0	117	100.00	100	1	1	0	
cross_256	117	0	117	100.00	100	1	1	0	
cross_1024	117	0	117	100.00	100	1	1	0	
cross_64_tx	7488	0	7488	100.00	100	1	1	0	
cross_256_tx	29952	0	29952	100.00	100	1	1	0	
cross_1024_tx	119808	0	119808	100.00	100	1	1	0	

Figure 12. The combined coverage report showing that 100% coverage is achieved.

VI. CONCLUSION

This work demonstrated how to harness the strengths of UVM and SystemVerilog to verify a highly-reconfigurable, analog/mixed-signal RF transceiver IC. The proposed testbench encapsulates the analog specifics of the DUT within a well-defined fixture module, enabling the use of standard UVM components for the rest of the testbench and performing extensive analog/mixed-signal verification thoroughly checking each and every operating mode. Also, the sequences enumerating the operating modes can be easily scaled and reused. The presented testbench performed three functionality checks over a total of 351 operating modes: data check, connectivity check, and control signal check. The SystemVerilog simulation produces a log containing the scoreboard table, cover property report, and coverage report. 100% coverage is achieved by combining 4 simulation results. The total simulation time is 197 minutes.

ACKNOWLEDGMENT

The EDA tools were supported by the IC Design Education Center and Scientific Analog, Inc.

VII. REFERENCES

[1] 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," *In'tl Solid-State Circuits Conf.* (*ISSCC*), Feb. 2019.

[2] K. Muhammad, et al., "Verification of Digital RF Processors: RF, Analog, Baseband, and Software," IEEE J. Solid-State Circuits, April 2007.

[3] Chan Young Park and Jaeha Kim, "Event-Driven Modeling and Simulation of 5G NR-Band RF Transceiver in SystemVerilog," *Int'l Conf. on Synthesis, Modeling, Analysis and Simulation Methods, and Appl. to Circuit Design* (*SMACD*), July 2021.

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

[5] Charles 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.