VHDL-AMS Behavioral Modeling and Simulation of a $\pi/4$ DQPSK Transceiver System

Erik Normark, Lei Yang, Cherry Wakayama, Pavel Nikitin, and Richard Shi

Department of Electrical Engineering
University of Washington
Seattle, WA 98195-2500, USA.

Email: {ecn1, yanglei, wakayama, nikitin, cjshi}@ee.washington.edu

Abstract— **This paper describes a methodology for top-down design, modeling, and simulation of complete $\pi/4$ DQPSK system using hardware description language VHDL-AMS. Two system implementations are considered: with and without Viterbi encoder/decoder.**

VHDL-AMS implementations of various RF blocks (e.g. a realistic channel model) are developed, the system is simulated, and bit error rate is evaluated in the presence of noise. We show that the results of VHDL-AMS simulations for basic $\pi/4$ DQPSK system in Mentor Graphics ADVance-MS match both Agilent ADS results and theoretical calculations. Adding a simple Viterbi encoder and decoder in VHDL to the basic system results in an approximate 1.4 dB SNR improvement.

This paper together is targeted towards engineers who work on behavioral modeling and simulation of complete RF systems using hardware description languages.

I. Introduction

As the demand for system-on-chip (SoC) implementations increases, the need to accurately model mixed-signal designs becomes more important. Digital designs have been highly automated, and the prevalence of top-down design is very strong in this area. In contrast, traditional analog RF designs are normally bottom-up, starting at the transistor level. Mixed-signal designers must then take a combination of hierarchical design approaches, and effort is being made to automate this design flow in a similar manner as seen for current digital systems. The overall goal is to provide designers tools to allow the combination of digital and RF models at the netlist level, creating a physical SoC model from which masks can be made for quick prototyping and fabrication. The ability to model and co-simulate digital and RF components together was made possible by the creation of hardware description languages (HDLs) such as VHDL-AMS [1] and Verilog-A. That requires the development of high-level behavioral models for mixed-signal systems blocks. Later, the abstraction levels of these models can be reduced to more accurately model physical circuit implementations.

Many of the recently documented system-level behavioral models in VHDL-AMS [2], [3] have been basic functionality tests that use highly ideal behavioral descriptions and do not include simulation of the high-frequency RF blocks. The next step in these simulations is to provide a benchmark for the transceiver system performance by implementing a realistic RF transmission channel. Although noise such as jitter has been modeled in VHDL-AMS [4], additive white Gaussian noise (AWGN) has not been implemented. This work has not been performed in part because there is no inherent VHDL-AMS command to create various noise distributions, a recognized problem for RF behavioral modeling in VHDL-AMS [5].

In this paper, an AWGN channel model is implemented in VHDL-AMS [6], allowing a more complete model of system performance measured in terms of bit error rate (BER). Two VHDL-AMS implementations of a $\pi/4$ DQPSK transceiver system are developed to verify the functionality of the system illustrate a top-down design approach to mixed-signal behavioral modeling. The first system closely matches the architecture of the theoretical model, while the second system is an iteration of the basic model, implementing a Viterbi encoding algorithm.

II. Basic $\pi/4$ DQPSK System

In order to validate the ability of VHDL-AMS to successfully describe the system-level performance of a full transceiver design, a common and complex mixed-signal transceiver model is needed to compare to implementations in other engineering design automation companies' development environments. Starting the design at the system-level allows us to compare the overall design concept with a theoretical implementation and iteratively decrease the level of abstraction. The architecture of each block can be refined and re-simulated to continuously monitor overall system performance as the design progresses. Using VHDL-AMS, these architectures can be recycled for future system implementations.

The $\pi/4$ DQPSK transceiver is common in many consumer electronics and is the IS-54 TDMA standard for US and Japanese cell phones. It contains mixed-signal components, many of which are normally implemented in a DSP [7]. For our implementation, we used highly abstracted circuit models and architectures contain linear models in order to best compare to theoretical.

There are many advantages to using π/4 DQPSK modulation [8], the two most important being that the signal can be differentially detected and symbol transitions in the constellation diagram do not pass through the origin. This allows for a less complex receiver implementation, a less variant envelope, and better spectral characteristics than other QPSK techniques.

This modulation technique can be viewed as two QPSK constellations offset 45 degrees from each other, where phase transitions are determined by alternating between the two offset constellations. Figure 1 shows the constellation diagram for π/4 DQPSK and illustrates the QPSK and OQPSK constellations.

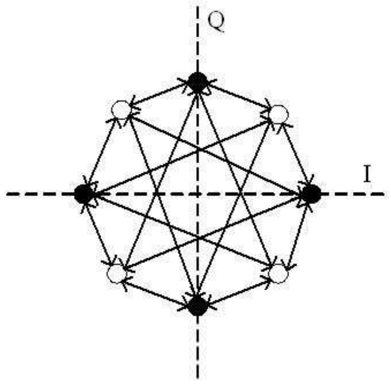

Fig. 1. Constellation diagram of π/4 DQPSK modulation. QPSK and OQPSK points are indicated as black and white circles respectively

This constellation is formed by taking and input symbol pair (A_k, B_k) and mapping them to in-phase and quadrature-phase symbols (I_k, Q_k):

$$I_k = (I_{k-1}A_k - Q_{k-1}B_k)\cos(\pi/4)\,, \tag{1}$$
$$Q_k = (Q_{k-1}A_k + I_{k-1}B_k)\cos(\pi/4)\,. \tag{2}$$

A. VHDL-AMS Implementation

A highly ideal system, modeled after the theoretical $\pi/4$ DQPSK system, is implemented in VHDL-AMS. To better manage the number of components involved in this design, the code hierarchy in Figure 2 is implemented, with each block containing the components needed to perform its intended function. The DQPSK system wrapper is used to initialize variables concerning only the RF system: transmitter, propagation channel and receiver. The BER wrapper encapsulates the entire design and attaches purely digital components and BER testing components to the mixed-signal system under test.

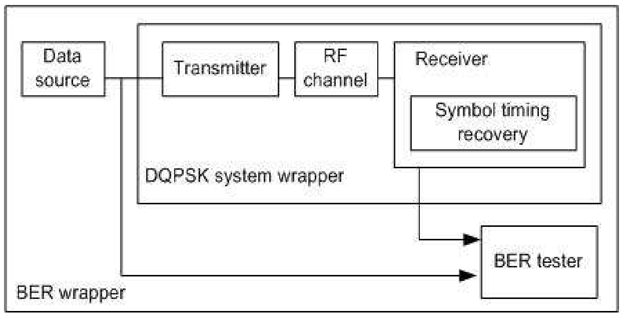

Fig. 2. Basic $\pi/4$ DQPSK code hierarchy of VHDL-AMS model.

A.1 Transmitter

For the transmitter shown in Figure 3, the incoming random serial data stream is first converted to two-bit parallel symbols (A,B) and converted to bi-polar analog. These symbols are then passed to the symbol mapper, which performs the calculations in Equations 1 and 2. A state machine architecture is used to simplify the symbol mapping process since previous values of I_k and Q_k are needed.

Once the I and Q channels are created, the signals are passed through identical pulse shaping filters to decrease their out of band spectrum. Most implementations use root-raised cosine filters with a roll-off factor (α) equal to 0.35, but this requires the use of a DSP or other discrete implementation. Although z-domain modeling is available in VHDL-AMS, a low-pass filter implementation was chosen for simplicity. The pulse shaping filters are three-pole low-pass filters with cutoff frequency at the symbol rate. Later implementations can use root-raised cosine filters.

The pulse shaped signals are then up-converted by mixing the I channel with $\cos(\omega_c t)$ and Q channel with $\sin(\omega_c t)$. The resulting signals are then summed and input to an ideal power amplifier. Normally a band-pass filter is placed after amplification to improve output spectral characteristics, but in order to best match the theoretical model, this filter is left unimplemented.

A.2 Propagation Channel

There are two calculations performed simultaneously in the AWGN channel component shown in Figure 4. First, the input signal is attenuated and delayed. Second, Gaussian distributed noise with scalable power setting is added to the delayed, attenuated signal at each simulation time step. Settings for the delay, attenuation, and noise power are set by values in the generic port.

Although simple in concept, the design of the AWGN RF channel is complicated by the lack of support for VHDL-AMS simultaneous procedural statements, or procedurals, by the ADMS tools. This functionality in

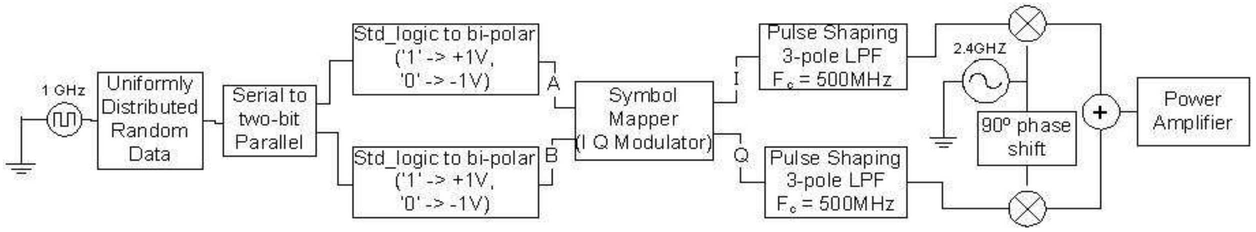
1 GHz
Uniformly Distributed Random Data
Serial to two-bit Parallel
Std_logic to bi-polar ('1' -> +1V, '0' -> -1V)
A
Std_logic to bi-polar ('1' -> +1V, '0' -> -1V)
B
Symbol Mapper (I Q Modulator)
I
Q
Pulse Shaping 3-pole LPF F_c = 500MHz
Pulse Shaping 3-pole LPF F_c = 500MHz
2.4GHZ
90º phase shift
Power Amplifier

Fig. 3. Block diagram of VHDL-AMS $\pi/4$ DQPSK transmitter.

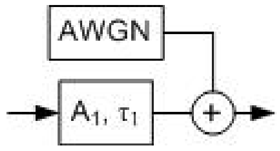

Fig. 4. Block diagram of AWGN propagation channel

VHDL-AMS provides a designer with the ability to define an analog calculation as a sequence of steps. Procedural statements contain an architecture block and a declarative section for creating local items, such as variables, types, subtypes, etc. Quantities are treated like variables, where new values are calculated based on the current value according to the simultaneous statements in the procedural block. Examples and a summary of the use of procedural statements can be found in [9].

In order to get around not being able to use procedural statements, Gaussian noise is created in a process, shown in Figure 5. The signal `awgn` dictates the next instance when the process will be called while `hmin` sets the period at which new noise values are generated. When the process is triggered, the noise signal is calculated using the Box-Muller [10] method:

$$X = \sqrt{-2\ln(U_1)}\cos(2\pi U_2)\,. \tag{3}$$

Two uniform random variables (U_1, U_2) are created, the Gaussian transformation is computed, and then the noise value is asserted after the prescribed delay time, which causes the process to be triggered again at a later time step. After the process is completed, the noise signal is converted to a quantity.

With this implementation of noise generation, it is very important to set the minimum and maximum simulation time steps equal to each other and equal to the triggering delay of the Gaussian noise generation process. If a new noise value is not added at every simulation time step, repeated AWGN values will be very common in analog waveforms.

Testing of the Gaussian noise generator in Figure 6 shows that the distribution is within 2% of expected mean and variance. Tests performed at different power levels show that the variance changes as expected, and different seed values produce similar distributions to each other.

```
-- repeatedly creates a noise value
-- after (hmin) time.
noise_calc : process (awgn)
  -- seeds for UNIFORM function call
  variable s1 : integer := seed1;
  variable s2 : integer := seed2;
  -- Uniform random variables
  variable x1,x2 : real;
begin
  -- create two uniform variables
  UNIFORM(s1,s2,x1);
  UNIFORM(s1,s2,x2);
  -- create Gaussian variable using
  -- Box-Muller method
  awgn<=SQRT(-2.0*LOG(x1))*COS(MATH_2_PI*x2)
          after hmin;
end process noise_calc;
```

Fig. 5. Process for generating white Gaussian noise

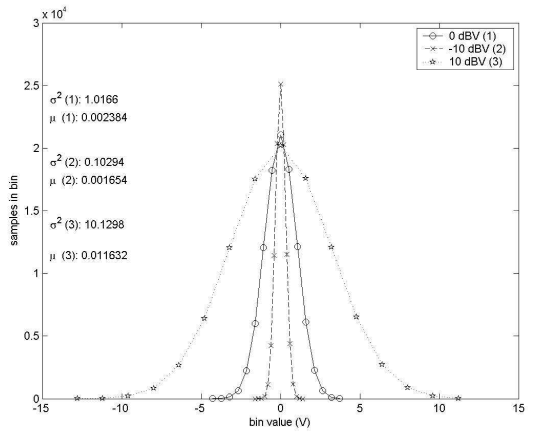

Fig. 6. Distributions of generated white Gaussian noise.

A.3 Receiver

After the signal is transmitted through the simulated channel, it is passed to the receiver and demodulator depicted in Figure 7. The signal is first amplified using an ideal LNA and then down converted by mixing it with the sine and cosine of the carrier frequency. It is not necessary for the down-converting oscillator to be in phase with the original carrier oscillator since this signal is differentially detected. Normally the LNA is proceeded by a bandpass filter to attenuate the out of band noise, but is left unimplemented to more closely match the theoretical example.

Once down converted, the signals are passed through low-pass filters identical to the filters in the transmitter, creating the reconstructed I and Q channels. In the theoretical model, these are also root-raised cosine filters.

The reconstructed I and Q channels are routed to the I-Q demodulator, as well as to the symbol timing recovery circuit. In order to demodulate the incoming symbols, phase information from the previous and current symbols is required. It is shown in [11] that only the sign of $\cos\theta_k$ and $\sin\theta_k$ are required to completely determine the bits transmitted in each symbol. Estimates for the received A_k and B_k parallel data channels are then found using:

$$A_k = \mathrm{sign}(\cos\theta_\mathrm{k}) = \mathrm{sign}(\mathrm{Q_k Q_{k-1} + I_k I_{k-1}})\,, \quad (4)$$

$$B_k = \mathrm{sign}(\sin\theta_\mathrm{k}) = \mathrm{sign}(\mathrm{I_{k-1} Q_k - I_k Q_{k-1}})\,. \quad (5)$$

In the simulated demodulator's VHDL-AMS architecture, shown in Figure 8, a delayed version the incoming signal is created in internal quantities to store the previously received symbol. Then, Equations 4 and 5 are executed. The demodulator also performs threshold detection and outputs the estimated sign of the received bits.

```
begin  -- behav
  -- set up input port impedances
  Ik == Ii*Zo;
  Qk == Qi*Zo;
  -- Perform A, B recovery
  Ip == Ik'delayed(2.0e-9);
  Qp == Qk'delayed(2.0e-9);
  Atemp == Qk*Qp+Ik*Ip;
  Btemp == Ip*Qk-Ik*Qp;
  -- Threshold detection
  if Atemp > 0.0 use Ak == 1.0;
  else Ak == -1.0;
  end use;
  if Btemp > 0.0 use Bk == 1.0;
  else Bk == -1.0;
  end use;
end behav;
```

Fig. 8. VHDL-AMS behavioral description of ideal $\pi/4$ DQPSK demodulator

To properly sample each bit in the middle of its period, a data clock signal is reconstructed from the input I and Q channels. When the channels are squared and added, tones are created at multiples of the symboling frequency, The tone at the symboling frequency is isolated using a narrow band, high-Q filter, creating a sinusoid that is in phase with the input symbols. Passing this signal through a threshold detector creates a clock where the rising edge occurs approximately in the middle of each symbol period.

The A and B channels output from the demodulator are then digitized and passed to the parallel to serial converter, where they are sampled using the recovered symbol clock, hard limited, and output serially.

B. Agilent ADS Implementation

For comparison, the BER simulation of ideal $\pi/4$ DQPSK system was also performed using Agilent's Advanced Design System (ADS) simulation software. The system is made up of the following ADS components: Data, DQPSK_Pi4Mod, DQPSK_Pi4Demod, berMC4, summerRF, noise, QAM_Mod, and Delay. The data rate and the modulation frequency are configured in the same way as in VHDL-AMS implementation.

For the system noise implementation, the receiver is considered noiseless and a known noise signal is injected between the transmitter and receiver. In our system, the data is quadrature-phase modulated onto the carrier, and then transmitted. The transmitted signal is represented as a complex envelope about the carrier, and the Gaussian noise sources are injected at RF as a complex modulation envelope using QAM. These signals are then passed through the demodulator and fed to the BER calculation block along with the delayed original data stream.

III. $\pi/4$ DQPSK Implementation with Viterbi Encoding

In a practical digital system, channel coding is employed to minimize the BER performance of the system, subject to constraints on transmitted energy and channel bandwidth. As an illustration of a simple design iteration, a rate $3/10$ Viterbi encoder and decoder are added to the $\pi/4$ DQPSK transceiver. This encoding method is described and thoroughly analyzed in many general communications texts, and many Viterbi algorithms have been previously implemented in VHDL. Figure 9 shows the modified code hierarchy of the system implemented with Viterbi encoding and decoding.

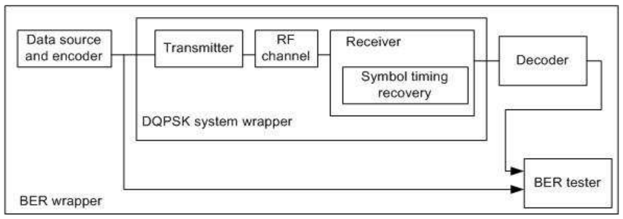

Fig. 9. Code hierarchy of $\pi/4$ DQPSK system with Viterbi encoding.

A. VHDL-AMS Viterbi Encoder

The implemented Viterbi encoder is a simple rate $\frac{1}{2}$ and constraint length three (K=3) structure. Since two bits are output for every one input bit, this allows the transmitter to run at half the clock frequency of the previous design and eliminates the need for a serial to paral-

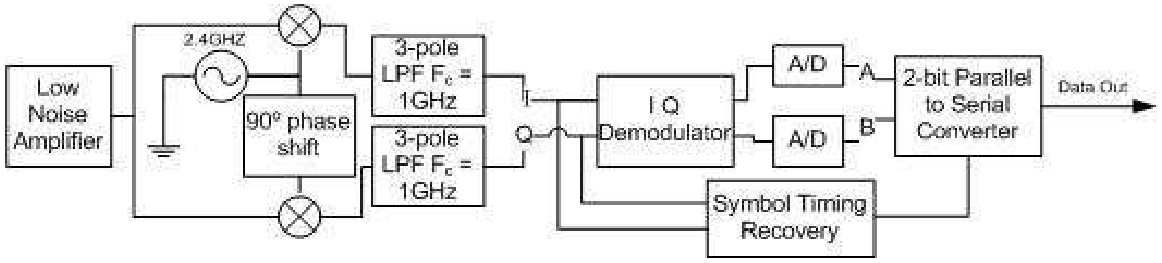

Fig. 7. Block diagram of VHDL-AMS $\pi/4$ DQPSK receiver.

lel converter. To further simplify the design, three memory bits are used (M=3). For every three-bit input to the encoder, two bits from the cleared shift register are appended, creating a five-bit vector. This vector is passed through the encoder, resulting ten output bits. The registers are then cleared and the next set of three input bits is requested, creating a continuous data stream for transmission.

B. VHDL-AMS Viterbi Decoder

The receiver continuously streams demodulated data to the Viterbi decoder. To catch up with the demodulation speed, a two-register-array "Ping-Pong" structure is implemented. The decoding process is partitioned into two phases: loading and decoding. When one register-array is performing the loading process, the other register-array is executing the decoding process, eliminating the decoding delay while adding to initial latency.

IV. Results

The theoretical, closed-form representation of BER [11] for $\pi/4$ DQPSK modulation is

$$BER = \left[1 - Q\left(\sqrt{\frac{E_b(2+\sqrt{2})}{N_o}}, \sqrt{\frac{E_b(2-\sqrt{2})}{N_o}}\right) + Q\left(\sqrt{\frac{E_b(2-\sqrt{2})}{N_o}}, \sqrt{\frac{E_b(2+\sqrt{2})}{N_o}}\right)\right] \times \frac{1}{2}, \quad (6$$

where $Q(a,b)$ is the Marcum Q-function. This expression is also closely estimated as

$$BER = Q\left(\sqrt{1.1716\frac{E_b}{N_o}}\right), \qquad (7)$$

where $Q(x)$ is

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2}\, dt\,. \qquad (8)$$

For this implementation, an important difference exists between how the theoretical models and the simulated models calculate E_b/N_o. The the theoretical model in [11] and the Agilent ADS simulation use methods that add noise directly to each transmitted bit in the I and Q channels separately before up-conversion. In the simulated model, noise is added in the RF channel after up-conversion and amplification, distributing the noise power over each symbol rather than each bit. This produces exactly half (a factor of 3 dB) the noise power density per bit than theoretical. Thus, the simulated BER curve has been shifted by 3 dB to the right.

BER measurement is performed using Monte Carlo methods described in [10].

A. Basic VHDL-AMS System

Simulations of the ideal VHDL-AMS implementation of the π/4 DQPSK produced BER results very close to those of the theoretical model and Agilent ADS implementation, all of which are plotted in Figure 10. Confidence intervals for the VHDL-AMS data are also given, and shows that the BER curve slightly deviates from theoretical as SNR increases. This is expected since the pulse shaping filters are implemented differently than those in the theoretical model, producing different spectral characteristics.

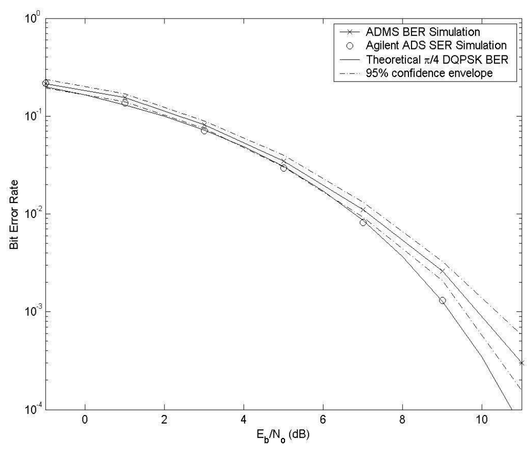

Fig. 10. Bit error rate for basic $\pi/4$ DQPSK system: comparison between theoretical, Agilent ADS, and VHDL-AMS (ADMS) models.

B. VHDL-AMS System with Viterbi Encoding

BER results from simulating the $\pi/4$ DQPSK system with Viterbi encoding and decoding, shown in Figure 11, indicate that BER performance is improved over standard theoretical implementation by approximately 1.4 dB because of its inherent error correction ability. Since the rate of the Viterbi implementation is 30% of the unencoded implementation, simulation times tended to be rather lengthy. As a result, fewer bits are simulated, giving rise to much wider confidence intervals than those seen in the basic $\pi/4$ DQPSK BER graph.

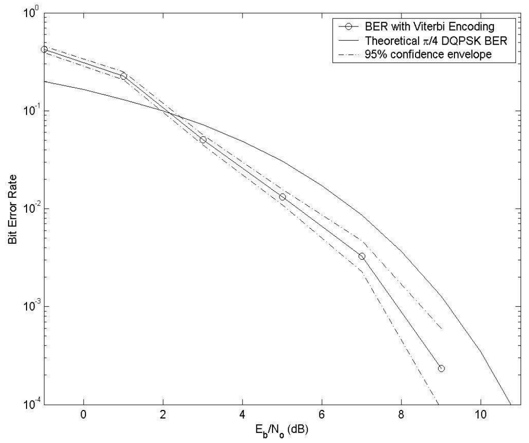

Fig. 11. Bit error rate for Viterbi encoded $\pi/4$ DQPSK system: comparison between theoretical and VHDL-AMS (ADMS) models.

V. Conclusions and Future Work

In this paper, we described a methodology for top-down modeling and simulation of complete RF/mixed-signal system using VHDL-AMS. As a demonstration example, we considered a $\pi/4$ DQPSK system and developed a library of behavioral level blocks for it. We simulated our system in a noisy environment and using Mentor Graphics Advance-MS and Agilent ADS. BER analysis showed that, although some minor differences existed between the VHDL-AMS and Agilent ADS models, the performance of the system very closely matched that of the theoretical model.

Following top-down design methodologies, the data source and receiver were made more complex by implementing a simple Viterbi encoder and decoder in VHDL, resulting in an approximate 1.4dB improvement over theoretical.

The next step in the design process for the systems presented here would be to further increase their design complexity by adding non-linear effects and sub-circuit implementations to the architectures of each block. Another extension of this research would be to implement channel effects such as fading and multipath to the RF channel. This would serve to better measure system performance in a realistic RF environment.

Each architectural block in this design can be further refined until the desired level of abstraction is reached before physical implementation is performed. This implementation a $\pi/4$ DQPSK transceiver verifies that complete system-level modeling and performance measurement can be easily executed using VHDL-AMS.

We target this paper toward the audience of VHDL-AMS users and hope that this paper will help VHDL-AMS designers to better understand the process of HDL modeling and simulation for RF/mixed-signal systems.

References

[1] "VHDL Analog and Mixed-Signal Extensions: IEEE standard 1076.1-1999,"

[2] M. Sida, R. Ahola, and D. Wallner, "Bluetooth transceiver design and simulation with VHDL-AMS," *IEEE Circuits and Devices Magazine*, vol. 19, pp. 11–14, March 2003.

[3] J. Ravatin, J. Oudinot, S. Scotti, A. Le-Clercq, and J. Lebrun., "Full transceiver circuit simulation using VHDL-AMS," *Journal of Microwave Engineering*, pp. 29–33, May 2002.

[4] A. Fakhfakh, N. Milet-Lewis, J. Tomas, and H. Levi, "Behavioral modeling of phase noise and jitter in voltage-controlled oscillators with VHDL-AMS," *Proceedings of IEEE International Conference on Circuits and Systems for Communications*, pp. 370–373, 2002.

[5] U. Knochel, J. Hartung, and R. Kakerow, "Verification of the RF subsystem within wireless LAN system level simulation," *Design, Automation and Test in Europe Conference*, pp. 286–291, 2003.

[6] P. Nikitin, E. Normark, C. Wakayama, and R. Shi, "VHDL-AMS modeling and simulation of a BPSK transceiver system," *Proceedings of IEEE International Conference on Circuits and Systems for Communications*, 2004.

[7] J. Webber and N. Dahnoun, "Implementing a $\pi/4$ shift DQPSK baseband modem using the TMS320C50," *University of Bristol, UK*, Sep. 1996.

[8] Y. Akaiwa and Y. Nagata, "Highly efficient digital mobile communications with a linear modulation method," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, pp. 890–895, June 1987.

[9] P. J. Ashenden, G. Peterson, and D. Teegarden, *The System Designer's Guide to VHDL-AMS*. Amsterdam: Morgan Kaufmann Publishers, 2nd ed., 2003.

[10] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems*. New York: Plenum Press, 1992.

[11] L. Miller and J. Lee, "BER expressions for differentially detected $\pi/4$ DQPSK modulation," *IEEE Transactions on Communications*, vol. 46, pp. 71–81, Jan 1998.