A Compact Single-FPGA Fading Channel Simulator

Amirhossein Alimohammad, *Member, IEEE*, Saeed Fouladi Fard, *Student Member, IEEE*,
Bruce F. Cockburn, *Member, IEEE*, Christian Schlegel, *Senior Member, IEEE*

***Abstract*— This article presents a novel computationally-efficient design and implementation of a Rayleigh flat fading channel simulator. To generate complex Gaussian variates with the required U-shaped power spectrum, the simulator utilizes an infinite impulse response (IIR) spectrum shaping filter followed by multistage interpolators and low-pass IIR filters. The new simulator significantly simplifies the characterization of wireless systems by providing a fast and area-efficient field-programmable gate array (FPGA) implementation of the fading channel. Our fixed-point Rayleigh fading channel simulator utilizes only 4% of the configurable slices, 20% of the dedicated multipliers, and 2% of the available memories on a Xilinx Virtex2P XC2VP100-6 FPGA, while generating 25 million fading variates per second. The parameterized fading channel simulator can be readily reconfigured to accurately simulate a wide variety of different channel characteristics.**

I. Motivation and Background

Two methods are commonly used to evaluate the performance of communication systems: experimental measurements and theoretical analysis. Due to the random nature of the wireless propagation environment, it is difficult to generate repeatable field test results. A multipath fading channel simulator allows rapid performance evaluation of mobile communication systems under a variety of conditions without the need for expensive and non-repeatable field testing. The primary goal of a fading channel simulation model is to faithfully reproduce the statistical properties of the real world channel.

It is common to model the behavior of a multipath fading channel as a complex Gaussian wide-sense stationary (WSS) process $c(t) = c_i(t) + jc_q(t)$ [1]. In a two-dimensional isotropic scattering environment with an omnidirectional antenna at the receiver [2], both c_i and c_q are uncorrelated, and the autocorrelation function (ACF) associated with either $c_i(t)$ or $c_q(t)$ is expressed as $\mathrm{R}_{c_i,c_i}(\tau) = \mathrm{R}_{c_q,c_q}(\tau) = \mathcal{J}_0(2\pi f_D \tau)$ where f_D is the maximum Doppler frequency and $\mathcal{J}_0(\cdot)$ is the zeroth-order Bessel function of the first kind [2], [3]. The power spectral density functions (PSDs) of $c_i(t)$ and $c_q(t)$, denoted by $\mathrm{S}_{c_i}(f)$ and $\mathrm{S}_{c_q}(f)$ respectively, also known as the Jakes' PSD, can be written as

$$\mathrm{S}_{c_i}(f) = \mathrm{S}_{c_q}(f) = \begin{cases} \frac{1}{2\pi f_D \sqrt{1-(f/f_D)^2}} & |f| < f_D \\ 0 & |f| \geq f_D \end{cases} \quad (1)$$

Two major approaches have been used in the literature for generating a random complex Gaussian process with the statistics described above. The first approach, also known as the sum-of-sinusoids method, was first proposed by Clarke [4]. In this method, the Rayleigh fading channel is modeled as the superposition of a plurality of sinusoids. Over the past three decades, numerous modified sum-of-sinusoids models have been proposed. Unfortunately, it has been found that many of the proposed algorithms for generation of correlated Rayleigh random variates do not produce statistically accurate fading variates [5]–[7]. Therefore, a careful study of these models is required before selecting them as the basis for fading channel simulators. Different hardware-based implementations of this scheme can be found in [8]–[10].

In the second approach, which is used in this paper and is called the filter-based scheme henceforth, independent complex Gaussian random variables are passed through a filter that has a frequency response equal to the square root of the Jakes Doppler PSD in (1). Since the input to the filter has a flat PSD (i.e., uncorrelated samples), the PSD of the output samples will match the expression (1). This shaping filter determines the PSD shape as well as the temporal correlation function of the fading process. This technique has also been used in some fading channel simulators [11]–[13].

Hardware-based simulators can greatly reduce the simulation time compared to software-based simulators [14]. Commercially available simulators are costly [15], [16], and recently proposed fading channel simulators require a complex heterogeneous architecture (usually consisting of general-purpose processors, DSPs, FPGAs, etc.) [11], [17]. A much more cost-effective approach is to implement the entire simulator on a single FPGA. Recent reductions in the cost of high-density FPGAs combined with advances in design tools have made FPGAs a practical platform for implementing hardware simulators. Reconfigurability, fixed-point arithmetic units with parameterized precisions, variable-length registers, and numerous dedicated signal processing cores provide a flexible target for the efficient realization of digital signal processing algorithms that were formerly implemented on digital signal processors (DSPs). The key contributions of our work include: (i) a novel filter design scheme for fading channel simulators; (ii) an efficient hardware architecture for implementing filter-based simulators; (iii) the only reported channel simulator realizable on a portion of a FPGA that accurately produces standard fading channel characteristics.

The rest of this article is organized as follows. Section II considers the design and hardware implementation of a filter-based Rayleigh fading channel simulator. Our proposed datapath for implementing a fading channel simulator and implementation results are presented in section III. The statistical properties of the generated fading variates are also verified. Finally, Section IV makes some concluding remarks.

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Alberta Informatics Circle of Research Excellence (iCORE), and the Alberta Ingenuity Fund (AIF).

A. Alimohammad, S. F. Fard, B. F. Cockburn and C. Schlegel are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2V4, Canada.

II. Filter Design

In wireless communication channels, the Doppler frequency is typically much smaller than the sampling rate. Thus the normalized bandwidth of the spectral shaping filter is relatively small. For example, consider the digital cellular system DSC1800 (GSM1800) which operates at $f_c = 1.8$ GHz. If the mobile receiver has a maximum speed of $v = 300$ km/hour, the maximum Doppler frequency would be $f_D = f_c \times (v/c) = 500$ Hz, where c is the speed of light. If the signal is sampled at $R = T_s^{-1} = 10$ MHz, the normalized Doppler frequency would be $f_D T_s = 0.00005$. However, such a narrow-band filter can easily become unstable. Instead, greater stability and computational efficiency is obtained by designing the shaping filter at a lower sampling frequency and then later increasing the sampling frequency using interpolation.

We introduce our new general design method by means of an example. Assume that the maximum Doppler frequency is $f_D = 4$ kHz and the target sampling rate is $R = T_s^{-1} = 10$ MHz. To decrease the filter orders, we will design the shaping filter at the sampling rate $R_1 = T_1^{-1} = 20$ kHz, which is 2.5 times the Nyquist frequency. Later, the sampling frequency will be increased to $R_2 = T_2^{-1} = 100$ kHz and $R = 10$ MHz with $I_1 = 5$ and $I_2 = 100$ times interpolation, respectively.

The filter is designed in two steps. In the first step, the required filters are designed to meet the desired characteristics. In a second step, the filters are optimized to reduce their hardware complexity. Since only signal amplitudes impact the correlation properties, no limitation is imposed on the phase of the target signal. Considering the greater computational efficiency of infinite impulse response (IIR) filters, we thus decided to approximate the PSD in (1) with a rational filter using the least pth-norm approximation [18]. This was achieved by utilizing the MATLAB function iirlpnorm [19]. This function allows one to weight the design error over different frequency ranges. Since the signal has a narrow bandwidth, we chose to utilize so-called low-order IIR interpolation low-pass filters (ILPFs) in the interpolation stages. Specifically, we use inverse-Chebychev lowpass filters for interpolation.

In the next step, the filters are changed to be more efficiently mapped onto the hardware. Since the shaping and first stage interpolation lowpass filters operate at relatively low sampling rates, it is advantageous to reuse hardware to implement both filters. Employing the second-order section (sos) form of the IIR filters simplifies hardware sharing. In fact a shared biquad can perform the processing of different sos parts in the shaping and the first ILFP. However, to simplify the control unit, as discussed in Section III, it is crucial for the two filters to have the same number of sos's.

To minimize the total number of sos's in the shaping filter and the first-stage ILPF, they should be optimized *together* to meet the design requirements. In a "discrete" design, the shaping filter should ideally have a frequency response matched to (1) in the range $|f| < 4$ kHz and a zero response elsewhere. Also, the first ILPF should ideally pass the signal within the range $|f| < 10$ kHz (recall that $R_1 = 20$ kHz). However, with finite out-of-band attenuation, the desired statistics of the target Rayleigh fading samples might not be met. In particular, the level crossing rate (LCR) of the envelope of the fading samples is sensitive to the out-of-band attenuation. We decided to design these two filters together to minimize the error in the pass band $|f|$ <4 kHz while maximizing the attenuation in the stopband 4<$|f|$<50 kHz. To find optimal filters, we employed a genetic search algorithm [20]. We used the weights computed by iirlpnorm and the lowpass filter parameters as the search variables. Fig. 1 shows the frequency response of the resulting shaping filter with $K = 6$ second-order sections. As this figure shows, the shaping filter provides a frequency response that closely matches the desired response over the passband and up to frequencies just inside the stop-band. However, at higher frequencies the attenuation is somewhat less. Fig. 2 shows that the first-stage ILPF provides at least 70 dB attenuation over frequencies where the shaping filter might not provide adequate attenuation. Therefore, the ILPF, not only attenuates the out-of-band signals, it can also help the shaping filter to provide more accurate samples.

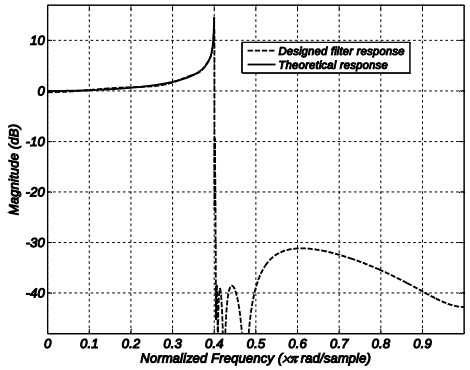

Fig. 1. Magnitude response of the elliptic shaping filter.

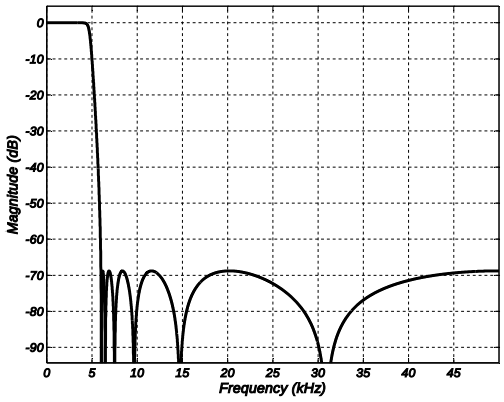

Fig. 2. Magnitude response of the first stage IIR low-pass interpolation filter.

The second-stage ILPF is designed using a similar technique. However, there is no constraint on the length of this filter. This allows us to use this filter to further increase the attenuation over the stop-band where the shaping filter and the first-stage ILPF could not provide enough attenuation. Since the second-stage ILPF runs at a higher frequency, minimizing the filter order significantly decreases the required processing.

The main differences between our proposed scheme and the previous fading channel simulators (e.g., in [11]–[13]) are

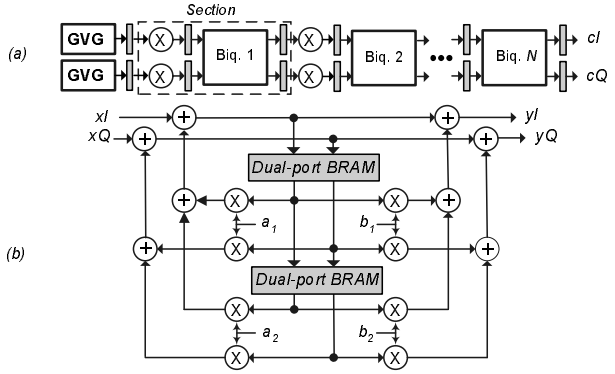

Fig. 3. (a) The cascaded structure of a Gaussian variate generator with $N = K + P$ second-order sections. (b) Biquad datapath.

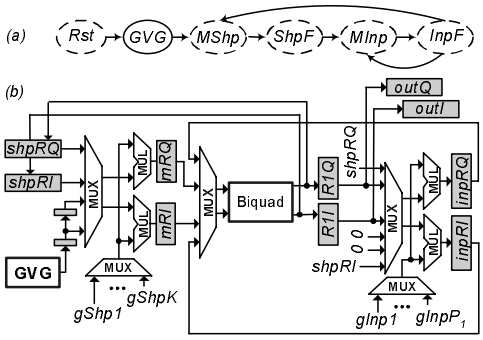

Fig. 4. (a) Control flow graph and (b) the datapath of the shaping filter and the first-stage low-pass interpolation filter.

two fold: (a) instead of using variable polyphase filters for interpolation, we used IIR filters, hence, the shaping filter and ILPFs can be used in a time-multiplexed fashion. As will be discussed in the next section, this technique leads to a significantly smaller design; (b) to accurately approximate the Jakes' PSD, we designed and optimized the shaping filter and the first-stage ILPF "together" and in two steps. In the first step, the shaping filter is designed to accurately approximate the Jakes' PSD in the in-band region. However, the shaping filter might not provide enough attenuation in the out-of-band region. In the second step, the interpolation filters are designed to not only eliminate the image signals caused by up-sampling, but also to provide additional attenuation in the out-of-band region. This will results in a much more accurate statistical properties as will be presented in the next section.

III. Implementation of Fading Channel Emulator

When implementing the fading channel emulator in fixed-point arithmetic, the stability of the designed IIR filters after quantization is ensured by keeping all poles within the unit circle in the z-plane. Also, to maintain accuracy, the fixed-point format of the intermediate signals is chosen based on numerical studies of how different precisions impact the statistical properties of the generated fading variates. A 32-bit fixed-point format was found to give sufficient accuracy.

The input to the filter chain is generated using Gaussian variate generators (GVGs). Realizing the fading channel simulator on an FPGA requires two GVGs (for the real and imaginary components), K cascaded biquads and $10K$ multipliers to implement the shaping filter, and P cascaded biquads and $10P$ multipliers to implement the ILPFs, as shown in Fig. 3(a), where $N = K+P$. The biquad datapath is shown in Fig. 3(b), where four registers are stored in two on-chip dual-port BlockRAM (BRAM) memories.

When the order of the filters is increased for greater accuracy, it becomes more challenging to realize the structure in Fig. 3(a) with high-precision arithmetic components on resource-constrained FPGAs. For example, for 32-bit realization of a biquad, the enclosed section in Fig. 3(a) requires 855 configurable slices and utilizes 9% of the dedicated 18×18-bit multipliers available on a Xilinx Virtex2P XC2VP100-6 FPGA. These results illustrate that the maximum number of second-order sections that can be implemented on a large FPGA is only around 11 due to the relatively large number of high-precision arithmetic units required by each biquad. Consequently, the direct instantiation of cascaded sections might be impractical for higher order filters (i.e., for smaller values of $f_D T_s$) or might not be efficient and flexible enough to implement variable sampling rates (e.g., for larger interpolation factors).

One widely-used solution is to utilize a heterogeneous architecture (e.g., consisting of general-purpose processors, DSPs, FPGAs, etc.) [11], [17]. The two main reasons are (I) direct implementation of a parameterizable Rayleigh fading channel simulator may not fit into one FPGA; and (II) the fading channel process usually varies much more slowly than the transmitted signal. This implies that the GVGs and the shaping filter can be updated at a much lower rate than the signal sampling rate. Thus, one can implement the GVGs and the shaping filter on a DSP, and the interpolator (or at least the last stage of a multi-stage interpolator) on a FPGA.

Our design exploits a timing-driven resource sharing approach to obtain the maximum performance with a minimum amount of FPGA resources. In this technique, portions of the spectral shaping filter are time-shared with the ILPFs. Resource sharing of independent operations of the spectral shaping filter and ILPFs offers significant resource savings. The throughput of a fading simulator depends on the binding of the second-order sections to the shared resources. The sampling rate of the hardware-based digital filters can be high enough that the throughput reduction of the time-multiplexed scheme, compared to the direct instantiation approach, does not impact the maximum target sample rate.

Due to the slow variation of samples at the shaping filter compared to the much faster possible operating rate of biquads implemented on an FPGA, we implemented the shaping filter and the first ILPF using one shared biquad. The second ILPF is mapped onto a separate set of configurable resources to achieve the target sampling rate. Fig. 4(a) shows the control flow graph that generates an appropriate control sequence for the datapath (shaping filter and first ILPF) shown in Fig. 4(a). The state machine controller starts in reset state *"Rst"*, where the *"gShp1"* to *"gShpK"* registers are initialized with the K scaling factors of the spectral shaping filter, and registers *"gInp1"* to *"gInpP"* are initialized with the P_1 scaling factors

of the first ILPF. Intermediate registers, such as *"shpRI"* and *"shpRQ"*, are cleared to zero. In the next state, *"GVG"*, two Gaussian variates are generated. State *"MShp"* denotes the multiplier state of the shaping filter and state *"MInp"* denotes the multiplier state of the first-stage interpolator. At the *"ShpF"* (*"InpF"*) state, one of the K (P_1) sections of the shaping filter (first-stage ILPF) is executed. For every execution of the *"ShpF"* state, states *"MInp"* and *"inpF"* will be executed I_1 times. After K executions of state *"ShpF"* (and thus KI_1 executions of the *"mInp"* and *"InpF"* states), *"InpF"* goes back to state *"MShp"* to multiply the previously generated Gaussian variates with one of the shaping filter scaling factors.

To implement the datapath shown in Fig. 4(b), the GVG block uses the Gaussian variate generator described in [21]. The GVG core operates at 269 MHz, utilizes 1.3% of the configurable slices, uses only three dedicated 18×18-bit multipliers, and use two BlockRAMs in the Xilinx Virtex2P XC2VP100-6 FPGA. Note that due to the slower operating rate of the shaping filter compared to that of the interpolator, only one GVG core is required. The registers of the datapath and the $K + P_1$ scaling factors of biquads are implemented using configurable slices while $4(K + P_1)$ registers and $4(K + P_1)$ coefficients of $K + P_1$ sos's are implemented using four dual-port BlockRAMs. The datapath in Fig. 4(b) utilizes 3% of the configurable slices, 1% of the dedicated multipliers and 6 on-chip BlockRAMs. The second ILPF was designed using a fourth-order low-pass inverse Chebyshev IIR filter and was implemented using one shared biquad, which receives its inputs either from the outputs of the first ILPF (i.e., the *"outI"* and *"outQ"* registers) or from "0" values inserted for zero padding. A complete fading channel simulator, for $f_D = 4$ kHz and $R = 10$ MHz with $I_1 = 5$, $I_2 = 100$, $K = P_1 = 6$ and $P_2 = 2$, utilizes 4% of the configurable slices, 20% of the dedicated 18×18-bit multipliers, and 10 BlockRAMs, and operates at 50 MHz.

An important property of the proposed scheme is that the complexity of the filter implementation will only impact the rate of fading variate generation and has almost no effect on hardware resource utilization. Even though the shaping and the low-pass interpolator filters are designed to emulate fading channels sampled at 10 MHz, since the fading channel simulator runs at a high clock frequency of 50 MHz, for the example system that the second ILPF generates one fading sample every four clock cycles, the simulator operates 1.25 times faster than the target sample rate. The simulator can be slowed down to emulate a wide variety of different channel characteristics over bandwidths of up to 12.5 MHz. To further speed up the fading sample generation rate, we utilized the commutative properties of the cascaded sos's and re-ordered the operations in such a way that the multiplication state *"MShp"* (*"MInp"*) can be performed simultaneously with the *"ShpF"* (*"InpF"*) state. Assume R_i is the output register of the i-th multiplier and BR_i is the register at the output of the i-th biquad in the cascade structure of Fig. 3(a). As given in Table I, we utilized an out-of-order scheduling scheme to reduce the required number of clock cycles to execute N cascaded second-order sections from $2N$ in the sequential scheme down to $\lceil N/2 \rceil$. A clock frequency of 20 MHz is required to generate 10 Msamples/sec. Since the designed fading channel simulator operates at 50 MHz, this approach provides 2.5 times speed up to generate fading variates at up to 25 Msamples/sec. Increasing the sample generation throughput in this way requires $N = K + P_1 + P_2$ times the number of registers compared to the sequential approach, as given in Table I. An important point to note in Table I is that allocating a physical register for every instance can lead to an inefficient register allocation while a register can be re-used after its lifetime (when its value is no longer needed). Note that R_i can be re-used after Biq_i reads its content, and BR_i can be re-used after multiplication by g_{i+1} where g_{i+1} is the scaling factor of the Biq_{i+1}. The scheduling of operations after re-using the registers is given in Table II, where only one register is utilized for the output of multiplier operations.

TABLE I

OUT-OF-ORDER EXECUTION SCHEME FOR CASCADE STRUCTURE

Cycle	*State*	*Operations*
$6i+0$	6	$R_5 = \text{MUL}(g_5, BR_4);\ \ BR_6 = \text{Biq}(sos_6, R_6);$
$6i+1$	5	$R_4 = \text{MUL}(g_4, BR_3);\ \ BR_5 = \text{Biq}(sos_5, R_5);$
$6i+2$	4	$R_3 = \text{MUL}(g_3, BR_2);\ \ BR_4 = \text{Biq}(sos_4, R_4);$
$6i+3$	3	$R_2 = \text{MUL}(g_2, BR_1);\ \ BR_3 = \text{Biq}(sos_3, R_3);$
$6i+4$	2	$R_1 = \text{MUL}(g_1, BR_0);\ \ BR_2 = \text{Biq}(sos_2, R_2);$
$6i+5$	1	$R_6 = \text{MUL}(g_6, BR_5);\ \ BR_1 = \text{Biq}(sos_1, R_1);$

TABLE II

OUT-OF-ORDER SCHEDULING AND REGISTER RENAMING OF CASCADE STRUCTURE

Clock no.	*State*	*Operation*
$6i+0$	6	$R = \text{MUL}(g_5, BR_4);\ \ BR_6 = \text{Biq}(sos_6, R);$
$6i+1$	5	$R = \text{MUL}(g_4, BR_3);\ \ BR_5 = \text{Biq}(sos_5, R);$
$6i+2$	4	$R = \text{MUL}(g_3, BR_2);\ \ BR_4 = \text{Biq}(sos_4, R);$
$6i+3$	3	$R = \text{MUL}(g_2, BR_1);\ \ BR_3 = \text{Biq}(sos_3, R);$
$6i+4$	2	$R = \text{MUL}(g_1, BR_0);\ \ BR_2 = \text{Biq}(sos_2, R);$
$6i+5$	1	$R = \text{MUL}(g_6, BR_5);\ \ BR_1 = \text{Biq}(sos_1, R);$

The HDL model of the proposed datapath was simulated to verify the accuracy of the results against the fixed-point software simulation results. A block of $50,000$ Gaussian variate samples was generated and passed through the designed filters, then the statistical properties of the 2.5×10^7 generated fading variates were measured. Fig. 5 plots the desired ACF along with the ACF of the samples generated with the new model. As Fig. 5 shows, the generated ACF accurately matches the theoretical ACF. The LCR [3] of the envelope of the generated fading variates and the desired LCR are plotted in Fig. 6. Here again a close match is observed. Finally, Fig. 7 plots the probability density function (PDF) of the generated fading variates against the ideal PDF. These plots show that the new hardware fading channel simulator faithfully reproduces the properties of the reference model.

We also simulated a communication system with 4-PSK and 16-QAM modulations and zero-forcing equalization at the receiver to show the performance of the new fading simulator with fixed point arithmetic (see Fig. 8). Here, for each signal to noise ratio (SNR), we transmitted 10^{10} symbols and measured the average symbol error rate (SER). As Fig. 8 illustrates, the SER plot generated by our new fading simulator again matches the expected theoretical target.

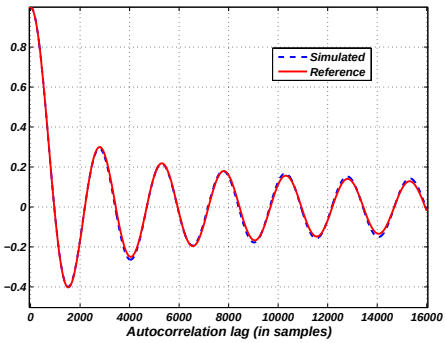

Fig. 5. ACF and CCF of 2.5×10^7 generated fading variates.

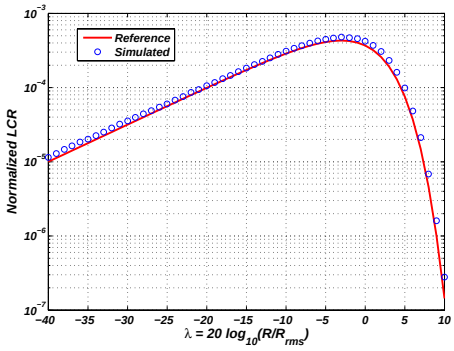

Fig. 6. LCR of 2.5×10^7 generated fading variates.

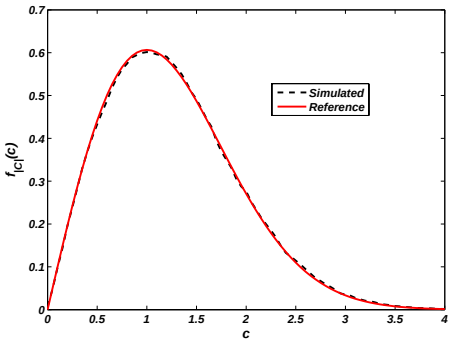

Fig. 7. PDF of magnitude of 2.5×10^7 generated fading variates.

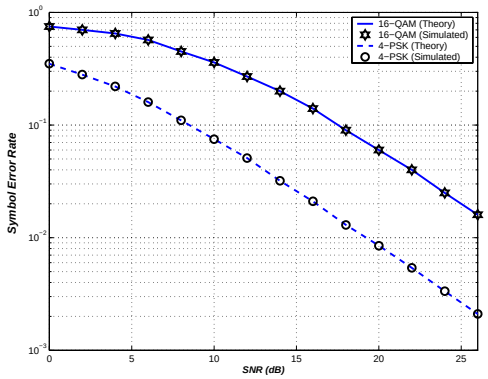

Fig. 8. Symbol error rate for simulated 4-PSK and 16-QAM system.

IV. Conclusions

A computationally-efficient and compact implementation of a Rayleigh fading channel simulator was presented. The filters designed to shape the power spectrum of samples, provide accurate fading samples, and yet minimize hardware complexity. Our fixed-point implementation on a readily available FPGA utilizes only 4% of the configurable slices, 20% of the dedicated multipliers, and $10\,(2\%)$ of the BlockRAMs, and can generate up to 25 million accurate fading variates per second. The compact simulator should be especially convenient for assessing the performance of communication systems over multiple-input multiple-output channels and diversity combining scenarios.

References

[1] P. Bello, "Characterization of randomly time-variant linear channels," *IEEE Trans. Commun.*, vol. 11, no. 4, pp. 360–393, 1963.
[2] W. C. Jakes, *Microwave Mobile Communications*. Piscataway, NJ: Wiley-IEEE Press, 1994.
[3] G. L. Stüber, *Principles of Mobile Communication*. New York: Kluwer Academic Publishers, 2001.
[4] R. H. Clarke, "A statistical theory of mobile-radio reception," *Bell System Technical Journal*, vol. 47, pp. 957–1000, 1968.
[5] M. F. Pop and N. C. Beaulieu, "Limitations of sum-of-sinusoids fading channel simulators," *IEEE Trans. Commun.*, vol. 49, pp. 699–708, 2001.
[6] C. S. Patel, G. L. Stüber, and T. G. Pratt, "Comparative analysis of statistical models for the simulation of Rayleigh faded cellular channels," *IEEE Trans. Commun.*, vol. 53, pp. 1017–1026, 2005.
[7] A. Alimohammad, S. F. Fard, B. F. Cockburn, and C. Schlegel, "An improved SOS-based fading channel emulator," in *IEEE Veh. Tech. Conf.*, 2007.
[8] M. Pätzold, R. Garcia, and F. Laue, "Design of high-speed simulation models for mobile fading channels by using table look-up techniques," *IEEE Trans. Veh. Technol.*, vol. 49, no. 4, pp. 1178–1190, July 2000.
[9] B. Surendra, B. Srinivas, C. Rambabu, and A. Gogoi, "A prototype architecture for the accurate modeling of Rayleigh fading waveforms," in *Intl. Conference on Information, Communications and Signal Processing*, vol. 2, 2003, pp. 1101–1105.
[10] A. Alimohammad and B. F. Cockburn, "Compact implementation of a sum-of-sinusoids Rayleigh fading channel simulator," in *Proceedings of the IEEE ISSPIT*, 2006, pp. 253–257.
[11] A. K. Salkintzis, "Implementation of a digital wide-band mobile channel simulator," *IEEE Trans. Broadcast.*, vol. 45, no. 1, pp. 122 – 128, 1999.
[12] C. Komninakis, "A fast and accurate Rayleigh fading simulator," in *Proceedings of the IEEE Global Telecommunications Conference*, 2003, pp. 3306–3310.
[13] M. Kahrs and C. Zimmer, "Digital signal processing in a real-time propagation simulator," *IEEE Trans. Instrum. Meas.*, vol. 55, no. 1, pp. 197–205, 2006.
[14] T. Jämsä, T. Poutanen, and J. Meinila, "Implementation techniques of broadband radio channel simulators," in *IEEE Veh. Tech. Conf.*, 2001, pp. 433 – 437.
[15] *Baseband Studio for Fading*, Agilent Technologies Inc., 2005.
[16] *Baseband Fading Simulator ABFS, Reduced costs through baseband simulation*, Rohde & Schwarz, 1999.
[17] M. A. Wickert and J. Papenfuss, "Implementation of a real-time frequency-selective RF channel simulator using a hybrid DSP-FPGA architecture," *IEEE Trans. Microw. Theory Tech.*, vol. 49, no. 8, pp. 1390 – 1397, 2001.
[18] A. Antoniou, *Digital Filters, Analysis, Design, and Applications*. McGraw-Hill Book Co., 1993.
[19] *Filter Design Toolbox For Use with Matlab, User's Guide*, The Mathworks, 2005.
[20] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.
[21] A. Alimohammad, S. F. Fard, B. F. Cockburn, and C. Schlegel, "A compact and accurate Gaussian variate generator," *to appear in IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*