A Flexible Layered Architecture for Accurate Digital Baseband Algorithm Development and Verification

Amirhossein Alimohammad, Saeed F. Fard, Bruce F. Cockburn
Department of Electrical and Computer Engineering, University of Alberta
Edmonton, T6G 2V4, AB, Canada
Email: {amir,saeed,cockburn}@ece.ualberta.ca

***Abstract*—Many emerging communication technologies significantly increase the complexity of the physical layer and have dramatically increased the number of operating configurations. To ensure maximum performance, designers have to optimize their algorithm implementations, which requires for comprehensive performance testing in all possible operating modes various channel conditions. This paper presents a flexible and affordable framework for baseband algorithm development and performance verification for digital communication systems with an arbitrary number of modules, each operating at a possibly different sampling rate with various latencies. The proposed architecture is scalable to support complex scenarios, such as multiple antenna systems, and is compact enough to be implemented within a single field-programmable gate array.**

I. Introduction

A fundamental measure of reliability for digital communication systems is the bit error rate (BER). While different stand-alone BER testers (BERTs) are available [1], [2], in general they have several key limitations: (I) BERTs are used after the communication device has been fabricated and at the RF level; (II) testing wireless devices using the BERTs at the non-repeatable air interface does not allow comparative analyses unless the measured channel samples are stored in a very large memory; (III) available BERTs tend to have limited flexibility and may not be capable of evaluating emerging technologies and standards; (IV) since commercially-available BERTs support different interfaces, engineers often have to develop custom interfaces in order to connect test equipment to their devices-under-test; (V) BERTs typically require additional equipment such as vector analyzers, signal generators, A/D converters, etc, so the overall cost can be high; (VI) and finally, it is challenging to verify the BER performance of RF hardware without accessing the baseband functionalities [3].

Hardware-based prototyping of communication systems at the baseband level has several key complementary advantages over RF testing using commercial BERTs including:

• It allows verification of error rate performance at a much earlier design stage, hence reducing the probability that a component or a subsystem design will need to be revisited after RF testing, when redesign would be more costly.

• It allows designers to study fundamental system trade-offs. For example, in IEEE $802.11n$ standard, there are more than 300 modulation-coding schemes. Choosing the optimal combination of baseband signal processing algorithms that meet a system's performance requirements is a challenging task and typically requires time-consuming software simulations.

• It allows exploration of the architectural design space and study of the practical limitations of baseband algorithms. This makes it easier to find an efficient fixed-point and/or floating-point solution with affordable computational complexity that meets given design constraints (such as area, performance, and power). It allows verification of receiver compliance prior to expensive ASIC implementation of baseband components.

We propose an integrated and flexible prototyping platform for both baseband algorithm development and bit/symbol error rate performance verification of wireless communication systems. Our approach minimizes the need for custom hardware and test systems by integrating the baseband functionalities on one (or more) field-programmable gate arrays (FPGAs). FPGAs provide a mix of high-performance dedicated signal processing modules and a large reconfigurable logic fabric, which make them a suitable platform for prototyping communication systems. Our parameterizable framework enables designers to develop, optimize and verify the transmitter and receiver baseband algorithms under a wide variety of system parameters such as channel conditions, modulations and coding schemes.

Three key differences between our proposed framework and previously published work [4]–[6] are:

(1) Some designs (e.g., in [4]) use model-based system development tools such as Simulink [7] and System Generator [8]. While these tools provide various block sets that contain useful functions, such as digital filters and error correction IP cores, they might not include signal processing modules for emerging technologies. Unfortunately, some of the available modules have relatively poor accuracy. For example, the Gaussian noise generator (GNG) supplied by Xilinx [9] degrades at the tails of the PDF for $|n| \geq 4.8\sigma_n$ and the PDF accuracy of the GNG in [4] is also limited to the interval $[0.2\sigma_n\%, 4.8\sigma_n\%]$. For a reliable low BER measurement, it is especially important for the GNG to accurately generate samples at the tails of the PDF.

(2) We utilize more accurate models for the source generator, the Gaussian noise model, and the wireless fading channel. Instead of a linear Gaussian channel model we use a more realistic parameterizable fading channel model.

(3) We propose a flexible and scalable interface structure that allows the cascading of an arbitrary number of baseband

978-3-9810801-5-5/DATE09 © 2009 EDAA

modules, each operating at a possibly different clock frequency and having various latencies. Even though for our proof-of-concept implementation we designed only one set of basic signal processing modules, the proposed interface structure makes this framework extremely flexible for implementing a wide variety of desired baseband functions.

The rest of this paper is organized as follows. Section II discusses a flexible interface subsystem. Implementation of a pseudo-random digital source is presented in Section III. Section IV gives implementations of Hamming encoder and decoder modules. A memory-based pseudo-random block interleaver/deinterleaver is discussed in Section V. Accurate implementations of a fading channel and Gaussian noise models are presented in Sections VI and VII, respectively. The architecture of a pipelined equalizer is presented in Section VIII. Section IX presents the proposed BERT implementation characteristics and also simulation results for an example communication system over a fading channel. Section X makes concluding remarks.

II. A Flexible Interface for Elastic Implementation of Communication Systems

The cascaded architecture of most digital communication systems allows communication functions to be partitioned into a sequence of independent modules. Each module K receives its input from the preceding module $K-1$ and sends the processed information to the next module $K+1$. A system model with the cascaded architecture of a digital baseband BERT is shown in Fig. 1. Designers may need to include other baseband signal processing algorithms/modules in the layered architecture of Fig. 1, such as an encryptor/decryptor pair and an FFT/IFFT [10]. Since different signal processing modules can be implemented independently of each other (e.g., modulation and interleaving), a suitable BERT architecture provides the interface flexibility to add/remove different components without requiring module interface redesign and without complicating the synchronization between modules.

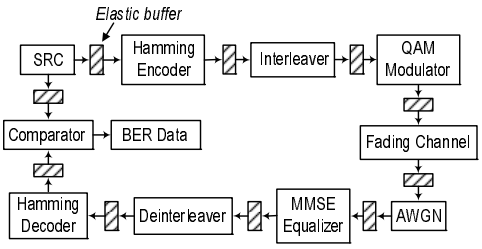

Fig. 1. Layered architecture of a digital baseband BER system.

As shown in Fig. 1, the information sequence generated by the digital source SRC will pass through various blocks such as a channel encoder, interleaver and modulator. The fading channel model attenuates and distorts the transmitted message symbols. The noise at the receiver, commonly modeled as Gaussian [10], is superimposed on the channel symbols. The receiver performs the reverse operation of the transmitter. Finally, the decoded bit stream will be compared with the reference sequence to calculate the BER.

In a layered baseband architecture, the rate at which samples are processed is often different for each module. An elastic buffer (EB) can accommodate differing sampling rates among cascaded modules. As shown in Fig. 2(a), the EB utilizes a circular Block RAM (BRAM) buffer that operates as a FIFO with two asynchronous ports: one port for writing in synchrony with *CLK1*, the enable signal *WriteCE* for write operations, with write address port *WriteAddr*; and a second port for reading in synchrony with *CLK2*, the enable signal *ReadCE* for read operations, with read address port *ReadAddr*. Each EB controls the operation of neighboring blocks, for example K and $K+1$ in Fig. 2(b), using the active low control signals *Full* and *Empty*. When *Full* is low, block K is disabled (i.e., it cannot write into EB since the BRAM is full) and when *Empty* is low block $K+1$ is disabled (i.e., it cannot read from EB since the BRAM is empty). Hence, the EBs control the flow of data by disabling the faster module in the chain. The average data rate is therefore dictated by the slowest module in the chain. In addition, each block is controlled by a global clock enable *CE* signal and also enable signals from other neighboring EBs.

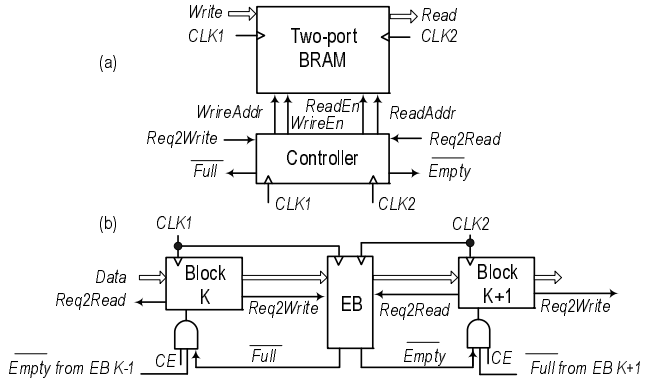

Fig. 2. (a) Elastic buffer component and its signals. (b) Interaction of the elastic buffer with neighboring modules.

The EB controller receives *Req2Write* from module K and *Req2Read* from module $K+1$ exactly half a clock cycle before every write into and read from EB, respectively. To avoid timing hazards, modules use positive edges of the clock signal to assert *Req2Write* and *Req2Read* signals while the EBs use negative edges of the clock to write and read to the BRAM.

III. Digital Information Source

It is common to utilize a pseudo-random number generator (PRNG) as the information source. Single-output linear feedback shift registers (LFSRs) are compact and fast sources that are often used in hardware-based simulations. However, there are cases where multiple pseudo-random bits are required per clock cycle. For example, PRNGs with parallel outputs are required for generating various initial parameters in the proposed channel simulator and also in the utilized GNG model, as will be discussed below. Moreover, the randomness

quality of the PRNG can also be an important factor. It is shown in [11] that sequences using LFSRs with parallel outputs typically have undesirable correlations. We thus used combined Tausworthe generators (CTGs) [12] that have multiple-bit outputs and substantially better randomness and correlation properties compared to conventional LFSRs.

A PRNG must have a long period to ensure that the test conditions do not repeat during simulation runs. To produce statistically meaningful error rate results, the period of the PRNG should be greater than N_e/BER, where N_e is the minimum number of errors required to reliably evaluate the system error rate at a given BER. We used a 32-bit CTG with four components and period $\rho \approx 2^{113}$ (`CTG2To113`) [13], as shown in Fig. 3, where $s_j, j = 1, \cdots, 4$ are four 32-bit variables initialized with different seeds.

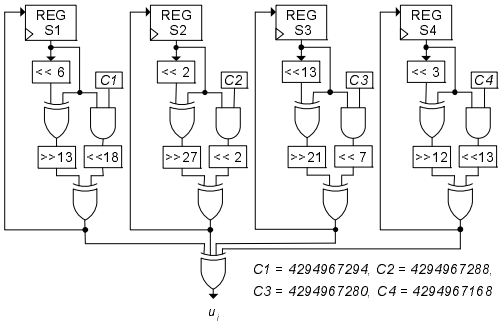

Fig. 3. Logic diagram of a CTG2To113.

IV. Channel Encoder and Decoder

For reliable communication at a target data rate, channel coding can reduce the probability of error or reduce the required signal-to-noise ratio (SNR) at the expense of increased decoder complexity. While different channel coding schemes are used in most modern communication systems [10], without loss of generality, we utilize simple Hamming encoder and decoder modules in the proposed example architecture. Specifically, we use a Hamming $(8, 4)$ encoder, as shown in Fig. 4(a), which encodes 4 data bits into 7-bit blocks by adding three parity bits in addition to an extra even parity bit [10]. At the receiver, the soft decoder calculates the distance of the received symbol $\mathbf{r} \in \mathbb{C}$ from all valid codewords $\mathbf{d} \in D$ as

$$\min_{\mathbf{d}\in D} |\mathbf{r} - \mathbf{d}|^2 = \min_{\mathbf{d}\in D} \sum_{i=1}^{8} \big(r(i) - d(i)\big)^2$$

Figure 4(b) shows the datapath of a Hamming soft decoder for QAM-modulated symbols. The Hamming decoder calculates the distance of the received signal (first the in-phase component and then the quadrature part) from $+1$ and -1 and then selects the best choice by searching over all valid codewords.

V. A Memory-based Interleaver/Deinterleaver

Interleaving is a simple and yet powerful technique for enhancing the error correcting capabilities of channel codes.

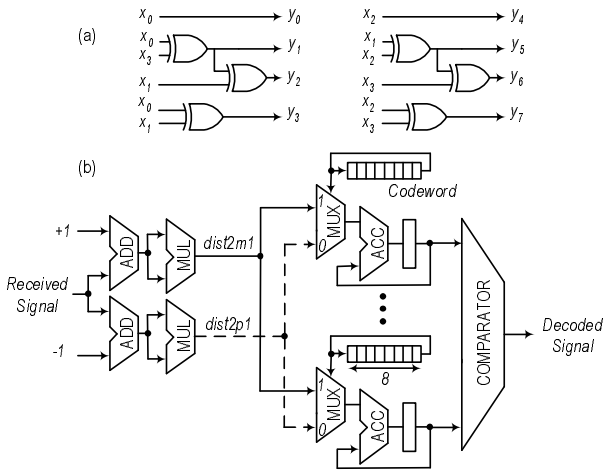

Fig. 4. Datapath of the (a) Hamming encoder and (b) decoder.

We utilize a pseudo-random block interleaver that randomizes the input sequence from the Hamming encoder. As shown in Fig. 5, we used a maximum-length LFSR with the characteristic polynomial $p(x) = x^{10} + x^3 + 1$ to randomly generate addresses between 1 to 1023. The interleaver writes the incoming bit stream into one 1024×1-bit BRAM addressed by an incrementor (`INC`) and then reads out the stored sequence in a pseudo-random order defined by the LFSR. The de-interleaver performs the reverse operation of the interleaver, using the same LFSR with the same initial seed. The soft output samples of the channel equalizer will thus be written into a 1024×16-bit BRAM in the same random order as the interleaver. Note that the output of the channel equalizer is a 16-bit complex variable. A parallel-to-serial module passes the in-phase and quadrature components to the deinterleaver to be written onto the BRAM, in two clock cycles. The 16-bit quadrature components are read sequentially from the BRAM using an `INC` and then passed to the Hamming decoder.

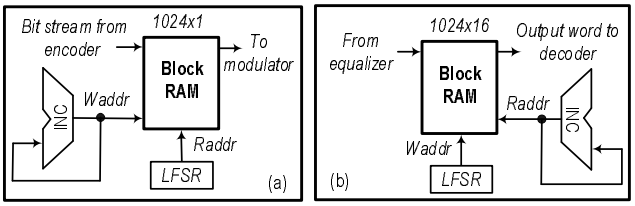

Fig. 5. Structure of the implemented (a) interleaver and (b) de-interleaver.

VI. An Efficient Wireless Channel

A well-known technique for modeling a fading process is the sum-of-sinusoids (SOS) method in which the Rayleigh fading channel is modeled as the superposition of a sufficient number of sinusoidal waveforms [14]. For Rayleigh fading channels, where the fading process is a complex Gaussian process $c(t) = c_i(t) + jc_q(t)$, the in-phase and quadrature components can be written in discrete time as follows:

$$c_i[m] = \sqrt{\frac{1}{N}} \sum_{n=1}^{N} \cos\left(2\pi f_D T_s m \cos(\alpha_n[m]) + \varphi_n[m]\right)$$

$$c_q[m] = \sqrt{\frac{1}{N}} \sum_{n=1}^{N} \cos\left(2\pi f_D T_s m \sin(\alpha_n[m]) + \psi_n[m]\right)$$

where $n = 1, \cdots, N$, $\alpha_n[m] = (2\pi n - \pi + \theta[m])/4N$, m is the discrete-time index, $f_D T_s$ is the normalized maximum Doppler frequency, and T_s is the symbol period. The above model is an improved version of our fading channel model from [15]. In the new model θ, φ_n, and ψ_n are independent stationary random processes (RPs). Specifically, we use random walk processes to update θ, φ_n, and ψ_n as given in Algorithm 1, where χ denotes one of the above RPs and u_χ is a random variable with independent, uniformly-distributed samples over $(0, 1)$. The coefficient ξ should be small enough

Algorithm 1 The proposed updating procedure for RP $\chi[m]$.

1: Initialize $0 < \xi \ll 1$; $s_\chi = 1$; $\chi[0] \in U(-\pi, \pi)$; $u_\chi \in U(0, 1)$;
2: **for** $m > 0$ **do**
3: $\quad \delta_\chi = s_\chi \cdot (\xi \times u_\chi[m])$; $\quad \chi[m] = \chi[m-1] + \delta_\chi$;
4: $\quad$ **if** $\chi[m] > +\pi$ **then** $\chi[m] = +\pi$; $s_\chi = -s_\chi$; **end if**
5: $\quad$ **if** $\chi[m] < -\pi$ **then** $\chi[m] = -\pi$; $s_\chi = -s_\chi$; **end if**
6: **end for**

to produce highly correlated (i.e., slowly changing) angles of arrival and phases for the required range of Doppler rates. Some suitable values for ξ were suggested in [15].

For a compact implementation of the channel simulator, we use a parallelized scheme in which updating the random processes executes at the same time as generating the fading coefficients. Using this approach, the $2N+1$ values of RPs θ, φ_n, and ψ_n are updated sequentially using shared arithmetic resources but distinct registers. To calculate $\cos\alpha_n[m]$ and $\sin\alpha_n[m]$ we can write $\alpha_n[m]$ as $2\pi(n - 0.5 + \tilde{\theta}[m])/(4N)$, where $\tilde{\theta}[m] \in [-0.5, 0.5)$ can be generated using a PRNG. For $n = 1, \cdots, 8$, the lower and upper bounds of α_n are $[0, \pi/16)$ and $[7/16\pi, \pi/2)$, respectively. Therefore, we only need to calculate the cosine values (for $c_i[m]$) and the sine values (for $c_q[m]$) of $\alpha_n[m]$ within the interval $[0, \pi/2)$. We uniformly quantize the cosine and sine values within $[0, \pi/2)$ into 512 segments and store their values in one BRAM. Only one BRAM and one multiplier are sufficient to multiply $f_D T_s$ in turn by $\cos\alpha_n[m]$ and $\sin\alpha_n[m]$. Then the $m(f_D T_s \cos\alpha_n[m])$ are calculated using an accumulator instead of a multiplier. The N values of $\cos\Big(2\pi\big((f_D T_s \cos\alpha_n[m]) \cdot m + \tilde{\varphi}_n[m]\big)\Big)$, where $\tilde{\varphi}_n \in [-0.5, 0.5)$, are obtained using N adders and $N/2$ dual-port BRAMs that store the pre-calculated values of $\cos(\cdot)$ within $[0, 2\pi)$. Figure 6 confirms the match between the Rayleigh PDF and the PDF of the generated fading samples.

VII. An Accurate Gaussian Noise Generator

The Box-Muller (BM) algorithm [16] has been widely used to transform pairs of uniformly-distributed pseudo-random

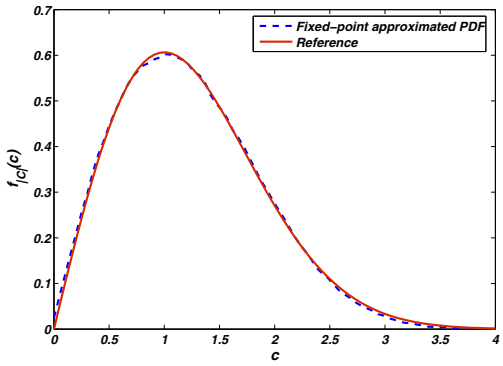

Fig. 6. PDF of 2×10^6 generated fading samples with $f_D T_s = 0.01$ and $N = 8$ and the reference Rayleigh distribution.

numbers into samples from a two-dimensional bivariate normal distribution. The inputs to the BM algorithm are two independent uniformly-distributed PNs, $u_1, u_2 \in (0, 1)$. The corresponding outputs are two independent samples, n_1 and n_2, from a zero-mean, unit-variance Gaussian distribution $\mathcal{N}(0, 1)$. The transformation involves multiplying $f(u_1) = \sqrt{-2\ln(u_1)}$ by $g_1(u_2) = \sin(2\pi u_2)$ and $g_2(u_2) = \cos(2\pi u_2)$ to yield n_1 and n_2, respectively. We adopted a polynomial curve fitting approach [17] to approximate $f(u_1)$ between $(0, 1)$. To accommodate the nonlinear shape of $f(\cdot)$, as shown in Fig. 7(a), the $(0, 1)$ interval is divided logarithmically into 62 segments. Each segment is then divided uniformly into 2^3 sub-segments. The value of $f(\cdot)$ within each sub-segment can be approximated using a linear function. The linear function coefficients for each segment can be calculated using the orthogonal least squares fit method [18] to minimize the residual error. The coefficients of $f(u_1)$ for all $62 \times 2^3 = 496$ segments can be scaled to maintain accuracy and then stored in one on-chip memory. To determine the segment number of a given PN input u_1, the addressing unit generates indices to address the coefficient memory. For the 32-bit u_1, we stored 496×2 scaled coefficients in one BRAM in a 16-bit fixed-point format. To approximate $g_1(u_2)$ and $g_2(u_2)$, we used a table look-up scheme in which the sine and cosine functions over $[0, 2\pi)$ are partitioned into 1024 uniform segments and the values of the functions are stored in two block memories.

Fig. 7(b) shows a dataflow diagram and the corresponding hardware datapath illustrating the evaluation of $f(\tilde{u}_1)$, where $\tilde{u}_1$ is a linearly scaled value of u_1. The core of the GNG contains three pipelined multipliers, one for calculating the $f(\cdot)$ and two for multiplying $f(\cdot)$ by $g_1(\cdot)$ and $g_2(\cdot)$, one adder, registers and associated routing resources. The operations are fully pipelined and scheduled to maximize the output rate. Figure 8 superimposes the PDF of 10^{11} generated GVs on top of a PDF plot of the ideal normal distribution using (a) a linear scale and (b) a logarithmic scale (since the tail accuracy of the Gaussian PDF is hard to judge on the linear scale). The two plots are indistinguishable over $\pm 6.6\sigma_n$.

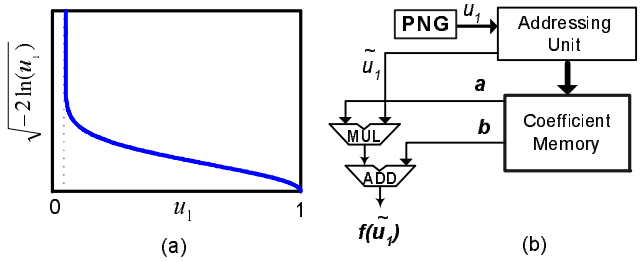

Fig. 7. (a) Plot of $\sqrt{-2\ln(u_1)}$. (b) The datapath for computing $f(\tilde{u}_1)$.

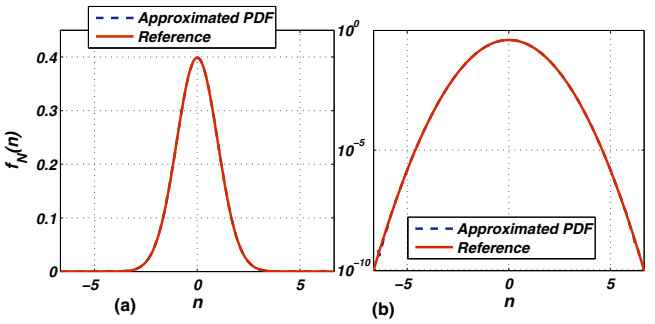

Fig. 8. Gaussian PDF compared with the PDF of 10^{11} generated GVs on (a) a linear scale and (b) a logarithmic scale.

VIII. A Compact Pipelined MMSE Equalizer

The data input sequence $\{x_k\}$ to a single path wireless fading channel can be estimated from the channel output sequence $\{y_k\}$ using an equalizer, where $y_k = c_k x_k + n_k$ and where c_k denotes the channel response and n_k denotes the channel noise. A minimum mean squared error (MMSE) equalizer calculates w_k such that $w_k \times y_k$ is the closest estimate of x_k. The MMSE equalizer calculates $w_k = c_k^*/(c_k^* c_k + \sigma_n^2)$, where c_k^* denotes the complex conjugate of channel response c_k. Since $c_k c_k^* + \sigma_n^2$ is a real number, to calculate w_k we use a pipelined real divider [19] and a complex multiplier to multiply $1/(c_k c_k^* + \sigma_n^2)$ with c_k^*. To perform $1/z$, where z is a normalized real number between $[1, 2)$, we can write $z = z_h + z_l$, where $1 \le z_h < 2 - 2^{-(\ell_h - 1)}$, $0 \le z_l < 2^{-(\ell_l - 1)}$, where ℓ_h and ℓ_l denote the number of bits dedicated to parts z_h and z_l, respectively. The reciprocal operation $1/z$ can be written using Taylor-series expansion as follows:

$$\frac{1}{z} = \frac{1}{z_h + z_l} = \frac{1}{z_h}\left(1 - \frac{z_l}{z_h} + \frac{z_l^2}{z_h^2} - \cdots\right)$$

Since $z_h \gg z_l$, we need only consider the first two terms and approximate $1/z$ with $(z_h - z_l)/z_h^2$. The datapath for calculating $1/z$ is shown in the dashed box in Fig. 9, where 2^{ℓ_h} different values of $1/z_h^2$ are precomputed and stored in a $2^{\ell_h} \times 16$-bit BRAM. Hence the reciprocal of z can be approximated using only a subtractor, a multiplier, and an on-chip memory. The left-hand side of the dashed-box in Fig. 9(a) calculates the divisor of w_k and the right-hand side of the box multiplies the reciprocal result by the in-phase and quadrature components of c_k^*. Fig. 9(b) shows the complex multiplication of w_k and y_k using only three real multipliers.

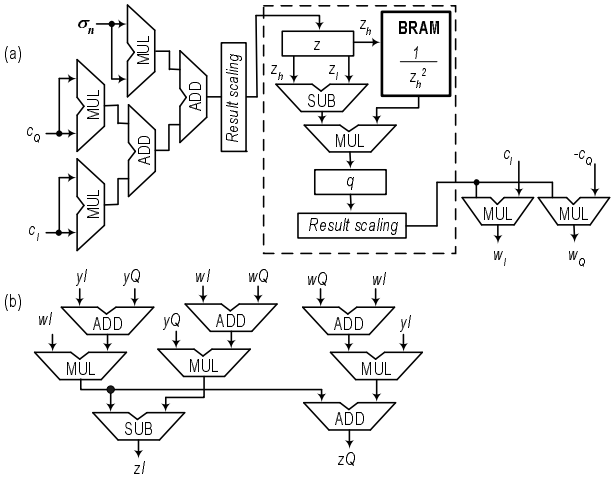

Fig. 9. The structure of the proposed pipelined MMSE equalizer. (a) Calculating w_k and (b) Computing $w_k \times y_k$ using only three multipliers.

IX. BERT System Architecture and Results

The block diagram of the example BERT system is shown in Fig. 10. For clarity we did not show the EB blocks between different modules. The PRNG data source, shown in Fig. 10(a), generates 4-bit symbols and the Hamming encoder generates 8-bit codewords. Then the interleaver receives the output of the Hamming encoder bit-by-bit through a parallel-to-serial (P2S) converter and generates a pseudo-random interleaved bit sequence. The interleaver output is then passed to a QAM modulator by grouping the bits into 2-bit words using a serial-to-parallel (S2P) converter. The modulator then generates 16-bit complex-valued soft symbols. Note that complex-valued signals appears as bold lines in Fig. 10. Note also that EBs at interfaces with complex-valued signals must use two copies of a BRAM: one each for the in-phase and quadrature parts.

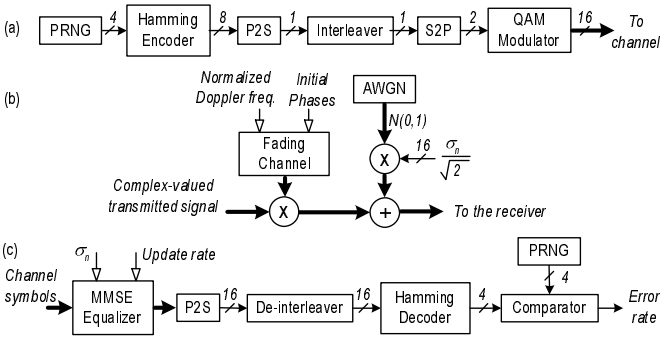

Fig. 10. The architecture of the proposed system. (a) The transmitter structure. (b) The baseband channel model. (c) The receiver structure.

The multiplicative effect of the channel and the additive impact of Gaussian noise on the transmitted symbols is reproduced using the baseband channel model shown in Fig. 10(b). Since we have already normalized the average symbol energy to unity, to evaluate the error rate over various SNR values,

only the noise power needs to be adjusted. For different SNR values, white Gaussian samples generated by the GNG core (with zero mean and unit variance) are scaled to normalize the noise power to σ_n^2. In the receiver chain, see Fig. 10(c), the MMSE equalizer, de-interleaver, and Hamming decoder are cascaded to estimate 4-bit transmitted symbols. To compare the reference transmitted bit stream with the estimated sequence, one option is to store the transmitted sequence in a FIFO [6]. Since the length of the required FIFO could be very long, we instead use the same PRNG with the same initial seed at the receiver to recreate the transmitted bit sequence after a suitable latency. Then the decoded bit stream is compared with the regenerated sequence to calculate the BER.

Table I gives the characteristics of the implemented modules on a Xilinx Spartan3 XC3S1500-5FG320. This FPGA includes 13312 configurable slices, 32 BRAMs, and 32 dedicated multipliers. Since the depth and width of the EBs' BRAMs are relatively small (16×16-bit), we implemented them using distributed memories (i.e., LUTs in shift register mode) [20]. The width and depth of the EB BRAM is parameterizable to support various interface settings. Also, the modulator module supports different modulation schemes such as BPSK, 16-QAM, and 64-QAM. Figure 11 plots the BER versus SNR with QAM modulation and a single-tap MMSE equalizer for uncoded and coded systems. The interleaver length is set to $2^{10} - 1$, the block length is $10 \times (2^{10} - 1)$, the number of transmitted blocks is 500, the Doppler frequency $f_D = 5$ Hz, the number N of sinusoids for modeling the fading channel is set to 8, $T_s^{-1} = 10^5$, and the minimum number of errors to be found is set to 1000. For channel estimation, one pilot symbol is inserted in every block of 16 symbols, which is also the latency of the reciprocal circuit used for equalization. The channel is estimated every 16 symbols during equalization. The interleaver length is set to almost 10 seconds, which is much longer than the average fade duration. Figure 11 shows that the proposed fixed-point BERT can indeed accurately evaluate the performance of baseband system models.

TABLE I
CHARACTERISTICS OF DIFFERENT MODULES ON A XILINX SPARTAN3

	Slices	*BRAMs*	*Mults.*	*Freq.*
CTG2To113	92	–	–	282.35
Hamming encoder	3	–	–	424.54
Interleaver	6	1(3%)	–	254.39
Modulator	107	–	–	331.81
MMSE equalizer	343(2.5%)	1(3%)	9(27%)	183.09
Fading channel	1125(8.4%)	9(28%)	1(3%)	175.07
GNG	565(4.2%)	2	3	202.51
De-interleaver	6	2(6%)	–	252.75
Hamming decoder	1025(7.6%)	–	2(6%)	149.94
Elastic buffer	55	–	–	142.56

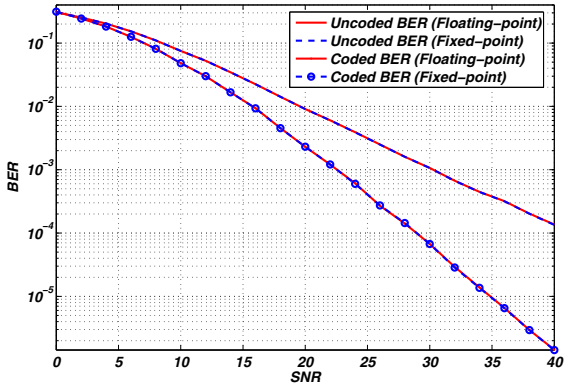

Fig. 11. BER versus SNR with a 4-QAM modulation and MMSE equalizer for an uncoded and coded system.

X. CONCLUSIONS

A configurable baseband platform is proposed for the efficient prototyping and performance verification of physical layer algorithms. Accurate modules for digital source generation, a fading channel, and a Gaussian noise model are proposed and implemented on a single FPGA. A compact MMSE equalizer utilizing a pipelined divider was also proposed. The platform's design is parameterizable to support various coding/decoding, modulation/demodulation, interleaving/deinterleaving modules, independent of their computational complexity and operating rates.

REFERENCES

[1] *Measuring bit error rate using the ESG-D series RF signal generators*, Product Note, Agilent E4430B 1GHz, Agilent Technologies Inc., 2000.

[2] *Fastbit FB100A BER Test System*, Application Note, Aeroflex Inc., 2007.

[3] *RF/IF-to-Digital Connected Solutions Bit Error Rate using the Advanced Design System*, Applic. Note, No. 1471, Agilent Technologies Inc., 2004.

[4] V. Singh, A. Root, E. Hemphill, N. Shirazi, and J. Hwang, "Accelerating bit error rate testing using a system level design tool," in *IEEE Symposium on FCCM*, 2003, pp. 62– 68.

[5] L. G. Barbero and J. S. Thompson, "Rapid prototyping system for the evaluation of MIMO receive algorithms," in *IEEE EUROCON '05*, 2005, pp. 1779–1782.

[6] *Hardware acceleration of 3GPP turbo encoder/decoder BER measurements using system*, Xilinx, XAPP948, December, 2006.

[7] *Using Simulink 7*, The Mathworks Inc., 2007.

[8] *System Generator for Simulink v2.1*, Xilinx Inc., San Jose, CA, 2003.

[9] *Additive White Gaussian Noise (AWGN) Core v1.0*, Xilinx Inc., San Jose, CA, 2002.

[10] B. Sklar, *Digital Communications: Fundamentals and Applications*. Prentice Hall, 2001.

[11] I. Vattulainen, K. Kankaala, J. Saarinen, and T. Ala-Nissila, "A comparative study of some pseudorandom number generators," *Comp. Phys. Comm.*, vol. 86, p. 209, 1995.

[12] P. L'Ecuyer, "Maximally equidistributed combined Tausworthe generators," *Math. of Comp.*, vol. 65, no. 213, pp. 203–213, 1996.

[13] ——, "Tables of maximally equidistributed combined LFSR generators," *Math. of Comp.*, vol. 68, no. 225, pp. 261–269, 1999.

[14] R. H. Clarke, "A statistical theory of mobile-radio reception," *Bell System Technical Journal*, vol. 47, pp. 957–1000, 1968.

[15] A. Alimohammad, S. F. Fard, B. F. Cockburn, and C. Schlegel, "An improved SOS-based fading channel emulator," in *IEEE Fall Veh. Tech. Conf.*, 2007, pp. 931–935.

[16] G. E. P. Box and M. E. Muller, "A note on the generation of random normal deviates," *Ann. of Math. Stat.*, vol. 29, pp. 610–611, 1958.

[17] *MATLAB Curve Fitting Toolbox, User's Guide*, The Mathworks, 2007.

[18] N. Chernov, C. Lesort, and N. Simanyi, "On the complexity of curve fitting algorithms," *Journal of Complexity*, vol. 20, no. 4, pp. 484–492, August 2004.

[19] J.-C. Jeong *et al.*, "A cost-effective pipelined divider with a small lookup table," *IEEE Trans. on Comput.*, vol. 53, no. 4, pp. 489–495, 2004.

[20] *Using the Virtex Look-Up Tables, The Quarterly Journal for Xilinx Programmable Logic Users*, Xilinx, Xcell Journal, 2000.