Towards Scalable Network Emulation: Channel Accuracy Versus Implementation Resources

Pengda Huang, Matthew Jordan Tonnemacher, Yongjiu Du, Dinesh Rajan, and Joseph Camp
Electrical Engineering, Southern Methodist University
{phuang, mtonnemach, ydu, rajand, camp}@smu.edu

***Abstract*—Channel emulators are valuable tools for controllable and repeatable wireless experimentation. Often, however, the high cost of such emulators preclude their widespread usage, especially in large-scale wireless networks. Moreover, existing channel emulators offer either very realistic channels for simplistic topologies or complex topologies with highly-abstracted, low-fidelity channels. To bridge the gap in offering a low-cost channel emulation solution which can scale to a large network size, in this paper, we study the tradeoff in channel emulation fidelity versus the hardware resources consumed using both analytical modeling and FPGA-based implementation. To reduce the memory footprint of our design, we optimize our channel emulation using an iterative structure to generate the Rayleigh fading channel. In addition, the channel update rate and word length selection are also evaluated in the paper which greatly improve the efficiency of implementation. We then extend our analysis of a single channel to understand how the implementation scales for the emulation of a large-scale wireless network, showing that up to 24 vehicular channels can be emulated in real-time on a single Virtex-4 FPGA.**

I. Introduction

Today's channel emulators allow extensive experimentation on complex wireless channels that can be controlled and repeated to evaluate protocols and hardware platforms directly in the same instance of an environment. Unfortunately, the emulators with the greatest channel fidelity are often limited to a single link and have a high cost [1], precluding larger-scale network emulation and widespread use by the research community. Conversely, many have used software-based simulators which scale to thousands of nodes but have very basic wireless channels [2]. There have been some efforts to build emulators that scale to a network, often using FPGAs and/or DSPs (e.g., up to 15 nodes [3]). However, to enable large-scale emulation, the degree to which the channel becomes more accurate at the cost of hardware resources must be fully understood.

In this paper, we study the tradeoff in channel emulation accuracy versus the implementation resources consumed, using both analytical modeling and FPGA-based implementation. We optimize our channel emulation using an iterative structure to generate the Rayleigh fading channel with a reduced memory footprint. The channel update rate and word-length selection are also analyzed to improve implementation efficiency and experimentally show the efficacy of our iteration-based scheme. We then leverage the tradeoffs in emulating a single wireless path as a fundamental building block to scale to a large wireless network emulator, designing a metric for implementation complexity.

This work was supported by the U.S. National Science Foundation under Grants CNS-0958436, CNS-1040429, and CNS-1150215

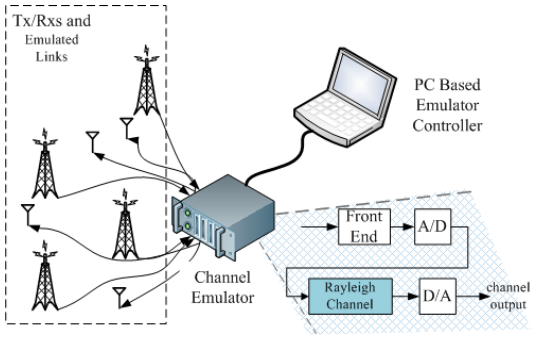


Fig. 1. Scalable wireless network emulation by exploiting knowledge of channel fidelity versus implementation resource costs.

To envision how the emulation would scale, Fig. 1 presents a general framework for our system design. A large number of wireless transmitters and receivers (i.e., units under test) are physically connected to the bank of hardware necessary to implement the wireless network emulator, and many different connectivity matrices, topologies, and mobility patterns can be emulated. A control PC allows the design and control of the experiment. On the back end, FPGAs implement the channel emulation from the RF front end to the channel output. To enable these FPGAs to generate a large number of the necessary fading wireless channels and know the amount of hardware required, the implementation resource consumption must be understood and optimized according to the size of the network. To lay the foundation for realizing such a vision, we present the following contributions in this work:

1) A novel Autoregressive (AR) model based Rayleigh fading channel generator is proposed. Prior channel generation solutions employing Sum-Of-Sinusoids (SOS) for Rayleigh fading channels demand large volumes of memory to store quantized cosine values. Since the memory resources available on FPGAs is limited, we implement our AR-based fading channel generator using a second order iterative structure which greatly reduces the amount of memory used.
2) Update rate selection and word length are jointly analyzed. Word-length and channel status update rate selection also affect hardware resource optimization on an FPGA. A tradeoff exists between the fading channel accuracy and word length selected. A high channel status update rate will increase the computational burden on the FPGA. Therefore, we design an optimization algorithm for jointly considering word length and update rate to reduce hardware resource consumption.
3) The scalability of wireless the network emulator is

studied. Based on our analysis and experimental evaluation of a single path, we show the achievable network scale according to a desired level of channel emulation accuracy.

The remainder of this paper is organized as follows. In Section II, the principle idea of SOS-based fading channel generation is introduced. In Section III, the novel generation scheme based on a second-order AR model is explored. Section IV discusses the word-length and channel status update rate optimization. Section V provides numerical results. Section VI discusses the expansion of our work to a scalable wireless network emulator. Finally, Section VII discusses related work followed by the conclusion in Section VIII.

II. Background: Rayleigh Fading Channel

A wireless fading channel $h(t,\tau)$, is widely described by its Power Delay Profile (PDP) as:

$$h(t,\tau) = \sum_{n=1}^{N} c_n(t)\delta(t-\tau_n), \tag{1}$$

where N denotes the number of taps in the fading channel, τ_n is the time delay of the *n*-th tap, and $c_n(t)$ represents the channel fading on the *n*-th tap. The fading on each tap is commonly described as Rayleigh and is characterized in the seminal work by Clarke [4] and Jakes [5]. In a Rayleigh fading channel, the transmitted waves on multiple paths are assumed to arrive at the receiver with a uniform incidence angle α. Based on this assumption, the Doppler shift, f_l, on the l-th multipath component can be expressed as $f_l = f_M \cos\alpha_l$, where f_M is the maximum Doppler shift and $\alpha_l = \frac{l}{L}\frac{\pi}{2}(l = 0, 1, \ldots, L-1)$.

Thus, each of the fading coefficients $c_n(t)$ in (1) is of the form (subscript n is omitted for simplicity):

$$c(t) = \frac{1}{L}\sum_{l=0}^{L-1} cos(2\pi f_M(cos(\alpha_l))t + \phi_l), \tag{2}$$

where ϕ_l is the carrier phase on the l-th multipath component which is modeled as a uniform variable within $[0,\pi]$.

This fading channel generation method is effective and often used for designing fast-fading channel emulators and simulators. However, this method for generating fading channels has a major disadvantage especially when applied to large-scale emulation systems. When multiple independent Rayleigh fading channels are desired, the independence of the channels generated using this approach cannot be guaranteed, *i.e.*, the cross-correlation between the generated Rayleigh fading channels does not approach zero.

To ensure independence of the multiple channels that are generated, Xiao [6] improved the SOS-based Rayleigh fading channel generation method (described in (2)) by using a modification factor (ξ_l) on the arrival angle of each component. This factor ξ_l is uniformly distributed in the range $\xi_l \sim U[\frac{l\pi}{2L} - \frac{\pi}{2L}, \frac{l\pi}{2L} + \frac{\pi}{2L}]$. This improved SOS-based Rayleigh fading channel generation model is given by:

$$c(t) = \frac{1}{L}\sum_{l=0}^{L-1} cos(2\pi f_M(cos(\frac{l}{L}\pi + \xi_l))t + \phi_l) \tag{3}$$

This model reduces the cross-correlation of any two simulated fading channels. Our proposed channel emulation method combines this improved model with an AR structure to guarantee independent channels and consume less implementation resources.

III. Iterative Structure in Cosine Waveform Generation

In Section II, we briefly introduced the SOS Rayleigh fading generation method. Traditionally, this method is implemented using a lookup table for the cosine values at every channel update time instance for every multipath component. This implementation strategy encounters severe problems due primarily to a memory bottleneck which induces large delays. In this section, we propose an AR model based on a second-order iterative structure to generate the Rayleigh fading channel. An FPGA based implementation of this scheme is also discussed.

A. AR Model Based Rayleigh Fading Channel Generation

1) Disadvantage of Conventional SOS-Based Rayleigh Fading Channel Simulation: Since there is no cosine generation function in the code set of an FPGA, the most popular method of cosine implementation is based on a lookup table for quantized cosine values stored in the RAM. Since a Rayleigh channel model is used to describe a fast-fading channel, the lookup operations are frequent during channel emulation. There are two main drawbacks in using the solution described above. First, the RAM resources are occupied throughout the entire procedure of channel generation. Unfortunately, the FPGA chip, though powerful in calculation, is often limited in memory resources [7], [8]. Second, the lookup operation has a non-zero delay. For each channel tap, the number of multipath components is usually selected in the range of 8 to 12 [9], to ensure desirable statistical characteristics of the resulting channels. In most cases, there are several channel taps between a single transmitter-receiver pair. Thus, the simultaneous lookup operations would cause excessive memory-lookup delay for FPGA-based channel generation. Furthermore, we ultimately seek to emulate multiple wireless links of a network. Thus, memory consumption and latency with a large amount of lookup operations precludes the use of SOS-based generation for large networks.

2) Iterative Solution to Sinusoid Generation: We now propose a simple, second-order AR model to generate the sinusoids needed for channel emulation. This AR model is based on the following identities:

$$\cos(i\omega - \omega) = \cos(i\omega)\cos(\omega) + \sin(i\omega)\sin(\omega) \tag{4}$$
$$\cos(i\omega - 2\omega) = \cos(i\omega)\cos(2\omega) + \sin(i\omega)\sin(2\omega) \tag{5}$$

Using (4) and (5), a simple and useful iterative structure of the fading channel generation can be obtained as:

$$\cos i\omega = 2\cos\omega\cos(i\omega - \omega) - \cos(i\omega - 2\omega) \tag{6}$$
$$\sin i\omega = 2\cos\omega\sin(i\omega - \omega) - \sin(i\omega - 2\omega) \tag{7}$$

As a general form, the AR sinusoid generation function can be described as:

$$y(i) = 2\cos\omega y(i-1) - y(i-2) \tag{8}$$

With the generalized AR-model based sinusoid generator, and by combining (3) and (8), the Rayleigh fading channel generation is described by:

$$\begin{cases} c(i) = \frac{1}{L}\sqrt{\left(\sum_{l=0}^{L-1} y_{Il}(i)\right)^2 + \left(\sum_{l=0}^{L-1} y_{Ql}(i)\right)^2} \\ y_{Il}(i) = 2cos(\omega_l) y_{Il}(i-1) - y_{Il}(i-2) \\ y_{Ql}(i) = 2cos(\omega_l) y_{Ql}(i-1) - y_{Ql}(i-2) \end{cases} \quad (9)$$

For convenience, Algorithm 1 presents the AR-model based iteration structure for generating the Rayleigh fading channel.

Algorithm 1 AR-model Based Rayleigh Channel Generation

1: Initialize

- generate $2L$ uniform random variable ϕ_l and γ_l,, $(l = 1, 2, \ldots, L)$;
- look up $5L$ cosine values $y_{Il}(0) = \cos(\phi_l)$, $y_{Il}(0) = \cos(\phi_l)$,

$$y_{Il}(1) = \cos\left(2\pi f_M \cos\left(\frac{l+\gamma_l}{L}\pi\right) + \phi_l\right),$$
$$y_{Ql}(1) = \sin\left(2\pi f_M \cos\left(\frac{l+\gamma_l}{L}\pi\right) + \phi_l\right),$$
$$B_l = 2\cos\left(2\pi f_M \cos\left(\frac{l+\gamma_l}{L}\pi\right)\right);$$

2: **for** $i < \lfloor T_{sim}/T_s \rfloor$ **do** ; ▷ T_{sim} and T_s are simulation duration and sampling interval, respectively

3:
$$y_{Il}(i) = B_l y_{Il}(i-1) - y_{Il}(i-2),$$
$$y_{Ql}(i) = B_l y_{Ql}(i-1) - y_{Ql}(i-2),$$

4:
$$c(i) = \frac{1}{L}\sqrt{\left(\sum_{l=0}^{L-1} y_{Il}(i)\right)^2 + \left(\sum_{l=0}^{L-1} y_{Ql}(i)\right)^2}$$

5: **end for**

From Algorithm 1, uniform random variables need to be generated first, which are used for the initial phase ϕ_l and the arrival angle modification factor γ_l. After that, the initial two iteration values are obtained by using a cosine value lookup table; then, the resulting sinusoid values $y_l(i), i \geq 2$, are generated iteratively; the summation of the sinusoids on each of the components describes the Rayleigh fading channel.

Based on Algorithm 1, only in the initialization step is the cosine lookup table used. Thus, only one cosine table is needed in the iterative Rayleigh fading channel generation. Subsequent computations require only one addition and one multiplication at each iteration. Since an FPGA is powerful in terms of computation, the addition and multiplication can be easily processed with minimal latency.

B. Proposed Rayleigh Fading Channel Generation Method

A block diagram of the proposed channel generation algorithm based on Xiao's method is given in Fig. 2. This algorithm has two stages: initialization and iteration. In the initialization stage, there are two main components: the cosine initial value generation and uniform random variable generation. In order to generate the uniform variable component, the

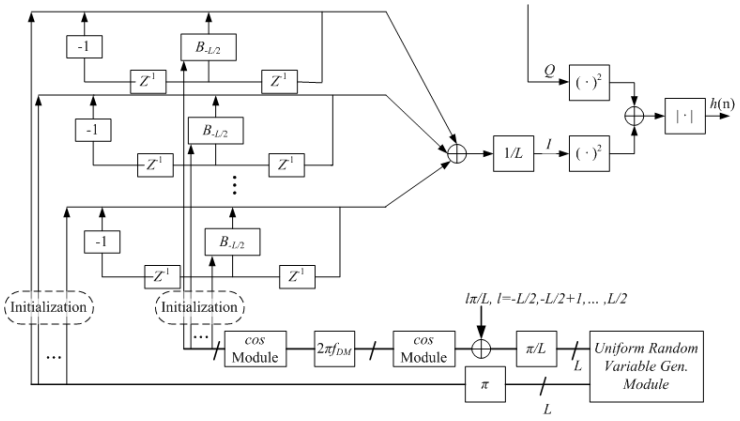


Fig. 2. Scheme of proposed Rayleigh fading channel generation simulator

Mersenne Twister algorithm [10] is employed which ensures a long period in the resulting pseudo-random sequence. The cosine value initialization is based on a lookup table, which is straightforward to implement. Fig. 3 illustrates the architecture of the iterative structure of the proposed channel generation.

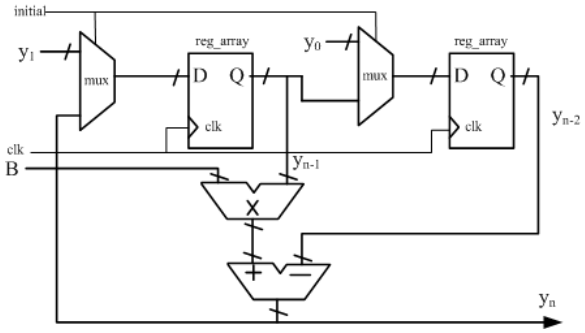


Fig. 3. Iteration architecture of one multipath component

From Fig. 3, only two registers are needed for one component. As introduced before, let L denote the number of multipath components in each tap, N be the number of taps in each fading channel, and T represent the desired number of fading channels to be generated. Assume the volume of RAM used for building the *sinusoid* lookup table is V_{LUT}. Thus, the AR-based channel generation scheme can save RAM resources up to V_D, which is given in (10),

$$V_D = T\left(2LN(V_{LUT} - 2) - V_{LUT}\right) \quad (10)$$

For instance, consider one ($T = 1$) ITU-Vehicle B channel containing six taps ($N = 6$). With ten components for a tap ($L = 10$), and the *sinusoid* lookup table is built by 1024 RAM word of two bytes. The amount of RAM savings can be calculated as $V \cong 243$ kB.

IV. Update Rate and Word Length Optimization

In this section, we study the effect of varying the rate and word length resolution at which channel coefficients are regenerated. The word-length selection is important in balancing the generated channel accuracy and the hardware cost. In this section, an analytical method and a numerical method will jointly be employed to study the optimal parameter selection. The MSE of the channel is used as the metric to determine the accuracy of channel emulation. The results from this study will be used in subsequent sections to enable temporal sharing of the multipliers (which are limited) on the FPGA.

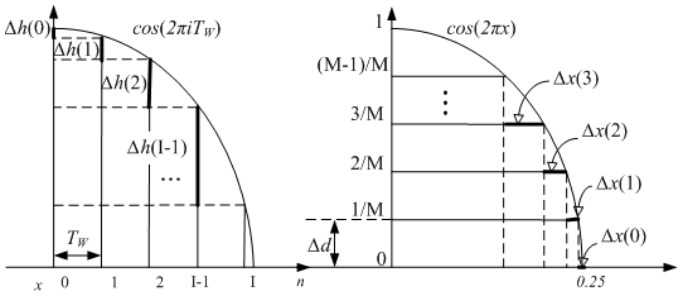


Fig. 4. Channel update rate effect and word length optimization illustration

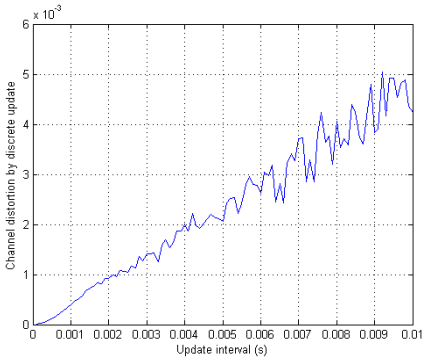


Fig. 5. Error of channel caused by discrete update

A. Channel Update Rate Emulation with Analytical Method

Let T_W denote the time duration between updates, which should be smaller than the coherence time of the channel. Hence, we select the channel update interval T_W to be smaller than the inverse of the maximum Doppler shift $1/f_M$. This assumption is reasonable because the channel output loss will become extremely large when T_W approaches $1/f_M$.

We first model one multipath component, which we later leverage to determine the scalability of a network emulator. For simplicity, consider a quarter of the sinusoid period for analysis, since a quarter period contains all the curvature information for the entire sinusoid. Let I denote the number of the update points in 1/4 the period of a sinusoid in one component (Fig. 4). From this figure, I can be determined as $I = \lfloor 1/4 f_M T_W \rfloor$.

The sinusoid at the *i-th* time instant is $c(i) = \cos 2\pi i f_M T_W, i = 0, 1, 2, \ldots, I$. The difference between two adjacent points can be calculated by:

$$\begin{aligned}\Delta c(i) &= c(i) - c(i+1) \\ &= 2\sin(2\pi i f_M T_W + \pi f_M T_W)\sin(\pi f_M T_W)\end{aligned} \tag{11}$$

The channel error, e, caused by discretely updating over the 1/4 period range can be calculated by summing up all the differences $(\Delta c(i), i = 0, 1, \ldots, I-1)$ and is given by

$$\begin{aligned} e &= \sum_{i=0}^{I-1} \frac{1}{2\Delta c(i) T_W} \\ &= \sin(\pi f_M T_W) \sum_{i=0}^{I-1} \sin(2\pi f_M T_W (i + \frac{1}{2})) T_W \\ &\cong \frac{1}{2\pi f_M} \sin(\pi f_M T_W) \cong \frac{T_W}{8\pi I} \sin(4\pi I) \end{aligned} \tag{12}$$

The longer update interval T_W induces a larger error caused by discrete channel updates. Further, for a fixed Doppler shift, a smaller T_W allows a larger I, decreasing the error.

B. Channel Update Rate Discussion with Numerical Method

To generate a tap of a fading channel, multiple components will be added to constitute the in-phase and quadrature components of a fading channel. Then, the square root of the in-phase and quadrature components will be calculated as the final fading channel output. In the process of calculating the square root, the phase information will be lost, in which case, the analytical method above does not work. Thus, the numerical method is employed which considers the square root of in-phase and quadrature components. During the observation time Δt, the discretely-updated Rayleigh fading channel is given as:

$$\begin{aligned} c'(t) = A \sum_{p=0}^{P-1} \sum_{l=0}^{L-1} &\cos(2\pi f_M (\cos\alpha_l) p T_W + \pi_l) \\ &\cdot [U(t - pT_W) - U(t - (p+1)T_W)], \end{aligned} \tag{13}$$

where P accounts for the fading channel updates within the time duration of Δt, $P = \lfloor \Delta t / T_W \rfloor$.

To measure the approximation error of the discretely updated fading channel, the mean square of error (MSE) $e^2_{\Delta t}$ is calculated as,

$$\begin{aligned} e^2_{\Delta t} =& \frac{1}{\Delta t} \int_0^{\Delta t} |c(t) - c'(t)|^2 dt \\ =& \frac{1}{\Delta t} \sum_{p=0}^{P-1} \int_{pT_W}^{(p-1)T_W} | \sum_{l=0}^{L-1} \cos(2\pi f_M (\cos\alpha_l) p T_W + \pi_l) \\ &- \sum_{l=0}^{L-1} \cos(2\pi f_M (\cos\alpha_l) t + \pi_l)|^2 dt \end{aligned} \tag{14}$$

Fig. 5 presents the channel distortion caused by the discrete update interval. The maximum Doppler frequency for the generated Rayleigh fading channel is 500 Hz. The minimum channel status update interval is 0.1 ms and the maximum interval is 10 ms. From Fig. 5, the channel errors increase in a nearly linear manner as the update interval increases, which agrees with the earlier analytical result.

C. Word-Length Effect on SOS Based Channel Generation

Besides the channel update rate consideration, the word length of each point in the fading channel generation will affect the hardware resources consumed. In this section, we first discuss the approximation error for a quantized cosine waveform at a single carrier. We assume the cosine wave is quantized by M levels, $M = 2^P$, where P is a positive integer. Thus, the distance between two quantization levels is $\Delta d = 1/M$, which is illustrated in Fig. 4. From Fig. 4, within the range [0, 0.25], the quantized cosine waveform at the *m-th* level is given by

$$x(m) = 1/2\pi \arccos(m\Delta t), m = 0, 1, 2, \ldots, M \tag{15}$$

After the uniform quantization, the corresponding error caused by the limited resolution can be calculated. The difference

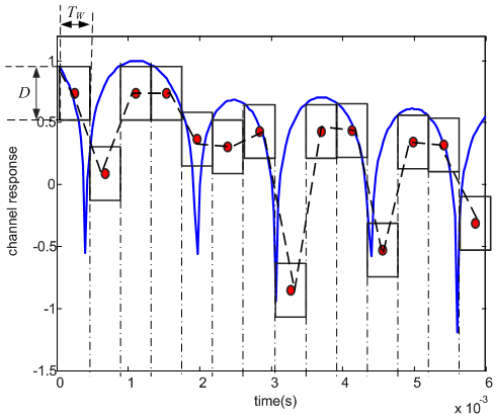


Fig. 6. Illustration of joint consideration on channel status update and word-length

between two adjacent quantizer levels is:

$$\Delta x(m) = -\Delta d \frac{d \arccos(2\pi m \Delta d)}{dm\Delta d} = \Delta d \frac{2\pi}{\sqrt{1-(2\pi m \Delta d)^2}} \tag{16}$$

The fading channel error caused by finite word length can be calculated by the area $e_{WL}(P)$ between the quantized and continuous curve. Thus, the fading channel error in $1/4$ period by P-bit quantization is given as:

$$\begin{aligned} e_{WL}(P) &= \sum_{m=0}^{M-1} \frac{\Delta d \Delta x(m)}{2} = \frac{\Delta d}{2} \sum_{m=0}^{M-1} \frac{1}{\sqrt{1-(2\pi m \Delta d)^2}} \Delta d \\ &\cong \frac{1}{2}\Delta d \int_0^1 \frac{1}{\sqrt{1-x^2}} dx = \frac{1}{4}\Delta d \end{aligned} \tag{17}$$

Hence, within a quarter period of the cosine waveform the quantization error by different word lengths can be simplified as,

$$e_{WL}(P) = \frac{1}{4}\frac{1}{2^P}, \tag{18}$$

which clearly shows that the error decreases exponentially with the number of bits.

D. Joint Optimization on Updated Rate and Word Length

With a P-bit word length, there exists a resolution window $D = 1/2^P$. After considering the channel status update interval T_W, the resolution rectangle S_R for measuring the Rayleigh fading channel is proposed. In general, a smaller rectangle corresponds to increased channel simulation accuracy. In the limit as the area of the rectangle approaches zero, there will be no quantization error. However, in practice, the area of the rectangle is always positive. Intuitively, a smaller area requires more hardware resources, a longer word length, and a higher update frequency than a larger area. We assume the area of the resolution rectangle is a constant A, which means the hardware resources consumed for a channel status update rate and word length are fixed.

The resolution rectangle is used to approximately sample the Rayleigh fading channel illustrated in Fig. 7. The centers of the rectangles are connected to represent the simulated channel with a generation error. Linear interpolation is performed between every two adjacent centers. After the interpolation, the approximate fading channel is constructed.

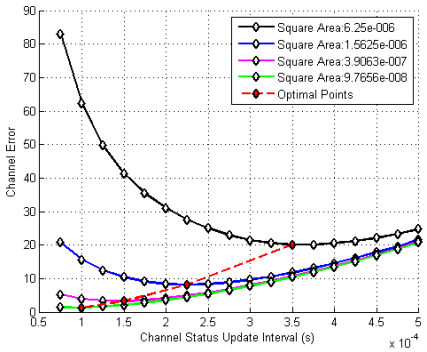


Fig. 7. Word-length and channel status update rate joint optimization

The accuracy of the channel is evaluated by computing the MSE, $f(D, T_W)$, with respect to a baseline channel. The joint optimization problem aims to minimize $f(D, T_W)$ over both D and T_W using standard numerical optimization methods.

Fig. 7 presents the MSE obtained by jointly considering word length and the channel update rate. According to Fig. 7, several values of rectangle areas are selected which are constant in the optimization process and equal the product of the fading channel update interval T_W and the limited word-length resolution D. Subsequently, we gradually change the value of the update interval with a renewed word-length resolution to calculate the MSE with a constant product. When the minimum MSE is found, the optimization converges. In Fig. 7, all the curves are concave, which implies that a minima exists. From Fig. 7, it is also observed that reducing the rectangle area can effectively reduce the minimum MSE at the optimal points.

V. Experimental Evaluation

In this section, we evaluate the performance of the proposed channel generation using analytical evaluation, numerical simulations, and experimentation with an FPGA platform. For comparison, a channel generation method using the conventional structure is also evaluated using an FPGA. Further, an offline channel generation method using a general-purpose computer is also considered in the experiments.

A. Iterative-Based Rayleigh Fading Channel Generation

1) Verification of Iterative Structure: The Wireless Open-Access Research Platform (WARP), shown in Fig. 8, is used as the experimental platform to study channel emulation performance. WARP uses a Virtex-4 VFX100FFG1517 FPGA chip. On the FPGA, both the iterative and the table lookup based Rayleigh fading channel generation are implemented. An oscilloscope is used to monitor the waveform of the generated channels.

Fig. 9(a) is a snapshot of the channel as seen using an oscilloscope. Fig. 9(b) is the histogram of the generated channel on the FPGA chip whose envelope matches the probability density function curve of a Rayleigh distribution.

To investigate the accuracy of channel generation using the iterative structure, we calculate two commonly-used metrics: the level crossing rate (LCR) and average duration of fading (ADF). As a reference, the LCR and ADF of the conventional SOS-based channel generation are also evaluated.

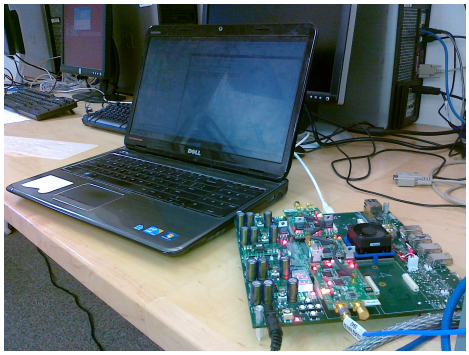
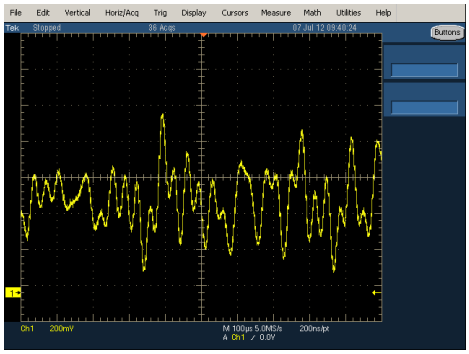
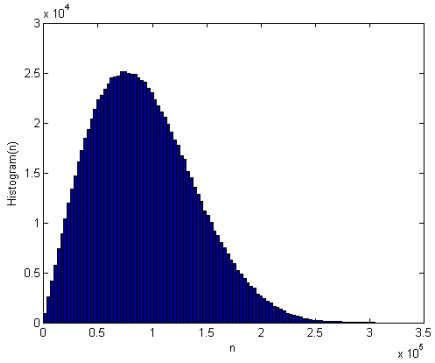


Fig. 8. Experiment platform



(a) Fading channel waveform on oscilloscope



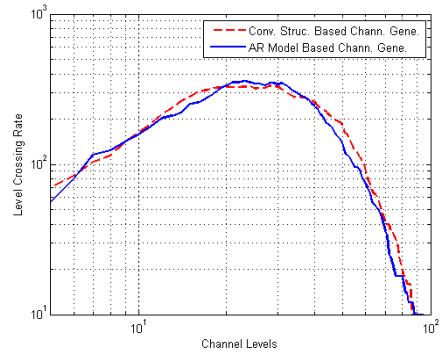
(b) Histogram of generated fading channel

Fig. 9. FPGA based fading channel generation output

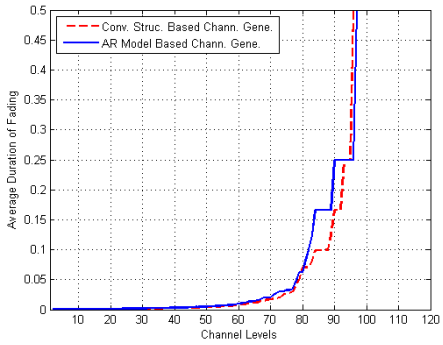
Fig. 10(a) and Fig. 10(b) illustrate the LCR and ADF for both the proposed method and conventional method. These figures demonstrate the similarity of the channel generated using these two methods. In subsequent sections, the hardware resource consumption of the proposed scheme is quantified.

2) Complexity Analysis: We now perform a complexity analysis for the proposed iterative method and the lookup-based method. The complexity is quantified by the number of real-number additions, real-number multiplications, and memory used to generate the output for a 3 ms time duration. In the evaluation, the Doppler frequency is 300 Hz which corresponds to a relative velocity of 56 km/h between the sender and receiver using a 5.8 GHz carrier frequency. For simplicity, we generate a channel with 1-tap ($N = 1$) and 10-components ($L = 10$) for this tap.

Table II presents a summary of the resources used as reported by the Xilinx ISE for the two fading channel generation methods. The percentage denotes the ratio of the used resources over the entire resources available on the FPGA.



(a) LCR



(b) ADF

Fig. 10. Statistical probability of generated fading channel

TABLE I
RAYLEIGH CHANNEL GENERATOR HARDWARE CONSUMPTION COMPARISON TO CONVENTIONAL STRUCTURE

	Flip-Flops	IOB	RAM16S	DSP48S
Iterative Str.	896(1%)	11(1%)	1(1%)	10(6%)
Table Lookup	589(1%)	46(5%)	8(2%)	0(0%)
Total Resources	84352	768	376	160

From Table I, the advantages in terms of RAM and IOB are evident due to the fewer *cosine* value tables. As expected, the proposed iterative method requires additional flip-flops and multipliers due to more complex logic than the conventional method. However, there is an abundance of flip-flop resources in most FPGAs and the number of multipliers will likely become the bottleneck to scalability. We further address this issue by developing a novel temporal re-use of the multipliers for generating different channels. As discussed in Section IV, the channel update rate is related to the Doppler frequency which is much smaller than the FPGA clock frequency. Thus, the same multipliers can be utilized by different emulated channels in allocated time slices.

B. Channel Generation Data Throughput Analysis

Another solution to channel emulation is to use an offline general-purpose computer to generate the fading channel coefficients and periodically transfer these to the FPGA to implement the emulation. The FPGA uses the stored channel data to modify the signal passing through the emulator. We call such a solution an offline generation scheme, as opposed to online channel generation on the FPGA.

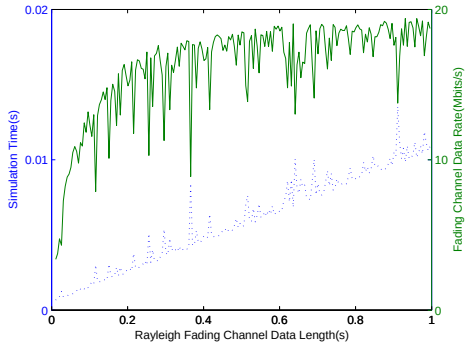


Fig. 11. Rayleigh fading channel offline generation rate

In the experiment, the PC used for generating the channel is equipped with an Intel i3 CPU at 2.26 GHz with 4 GB of RAM. Fig. 11 demonstrates the experimental results. The dashed curve represents the time consumed for generating a certain duration of the fading channel. The green solid curve denotes the corresponding channel parameter data rate, which represents the effective rate at which data needs to be transferred between the PC and the FPGA to enable emulation. From Fig. 11, we see that this rate is on the order of 20 Mbits/s for each tap.

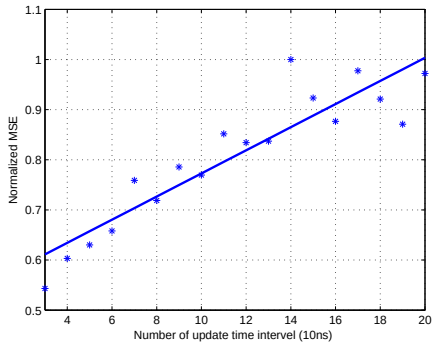
From the result of experiments on the Virtex-4 VFX100, the FPGA can generate up to 3.3 *Gbits/s* of channel coefficients for emulation. Clearly, the proposed channel generation scheme based on an FPGA outperforms the PC-based solution in terms of channel data throughput. Further, the offline PC-based generation requires additional RAM and buffers, especially for large-scale networks. On the other hand, the PC-based solution would reduce the computation burden on the FPGA, which could be useful in certain scenarios.

C. Performance Across Update Rates and Word Lengths

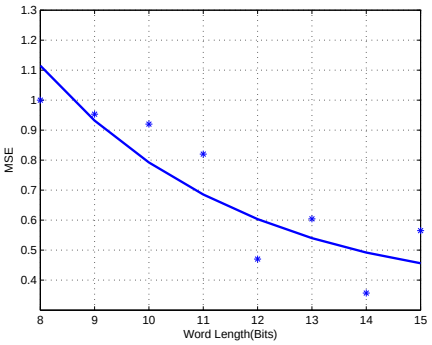
In Section IV, we analytically derived the expression for the MSE caused by a finite word length and update rate. In this section, the accuracy of the two expressions will be verified. The x-axis values in Fig. 12(a) denote the update interval which is measured in terms of 10 times the slice of the FPGA chip. The stars in Fig. 12(a) are the MSE values at different update intervals. The bold line represents the optimal linear fit to the data. We can see that the bold line agrees with MSE prediction in (12). To analyze the effect of word length on the MSE of the generated channel, we change the bit width in the process of generating fading channel from 8 to 16 bits. Fig. 12(b) presents the normalized MSE (normalized to the performance with 16 bits) for the different bit widths (8 to 16 bits). From Fig. 12(b), the MSE decreases when the word size increases, which agrees with our intuition that a larger word size results in a higher accuracy. The stars mark the MSE at every word size, and the bold line represents the best exponential fit, matching the result in (18).

D. Accuracy of Generated Fading Channel

In this subsection, the accuracy of the generated channel is measured using the resulting BER for a particular modulation scheme. In the experiments, a simulated channel generated in MATLAB (using real-valued coefficients with double precision) is taken as one reference. Furthermore, the performance



(a) MSE versus update interval



(b) MSE versus word length

Fig. 12. MSE of generated fading channel on FPGA

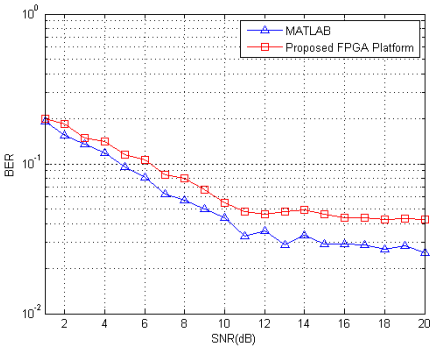


Fig. 13. 16QAM BER curves over generated fading channel

using a commercial fading channel emulator, the Azimuth 400MX, is also shown.

Using the same iterative algorithm implemented on the FPGA, we employ MATLAB to generate the channel. The same parameter settings are used in the two implementation platforms. Specifically, ITU-Vehicle B fading channels are generated which each contain six taps. The relative delay between the taps are [0 100 200 300 500 700] in *ns* and the relative energy ratios are [0 -3.6 -7.2 -10.8 - 18 -25.2] in *dB*. Signals using 16-QAM modulation are passed through the two fading channels. For simplicity, we do not consider channel estimation and equalization at the receiver. Fig. 13 shows that the BER over the fading channel from the two platforms behave in a similar manner.

Then, we compare the packet error rate (PER) of the

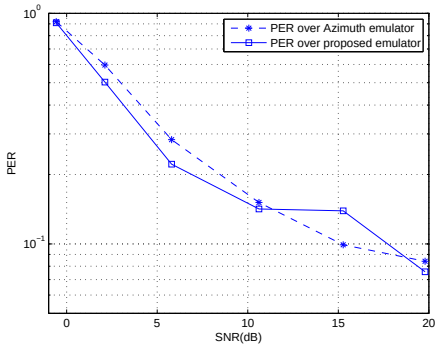


Fig. 14. QPSK PER curves on WARP and Azimuth emulator

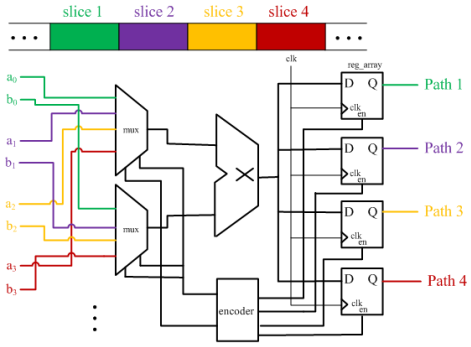


Fig. 15. Architecture of multiplier time division multiplexing

proposed emulator with the existing commercial channel emulator-Azimuth 400MX. In the experiments, two WARP boards are used as the transmitter (TX) and receiver (RX).

In the experiment, two kinds of trials are performed. In the first trial, the WARP-TX generates QPSK signals which are passed through the fading channel and also generated on the WARP-TX board. The resulting output signals are sent to the WARP-RX board. In the second trial, the generated QPSK signals from the WARP-TX board is connected to the Azimuth emulator via cable. The output signal from the emulator is sent to the WARP-RX board. In the two trials, the packet size is 1.5 *kB* and uses an 802.11a physical layer. Fig. 14 presents the variation of the PER curves versus SNR. From this figure, it is clear that the the PER of the two emulators, Azimuth and the proposed emulator, closely agree with one another.

VI. Network Emulation Scalability

Large-scale network emulation using off-the-shelf approaches requires significantly large hardware resources and is consequently very expensive. In this section, we discuss the scalability of the proposed network emulation framework.

A. Factors Allowing Greater Scalability

Two key novelties in the proposed emulation approach allows for better scalability of the networks. First, the proposed iterative structure results in significant memory savings for the FPGA. Second, update rate optimization is performed and coupled with a time-sharing architecture for multipliers to ensure better scaling.

Contribution: Scalability by Saving Memory. According to the experimental results, one tap of a Rayleigh fading channel consumes one RAMB16. The total memory resources of the aforementioned FPGA chip equals 376 units of RAMB16s. Therefore, using the proposed iterative structure, we can support the generation of 60 ITU-Vehicle B channels.

Contribution: Scalability by Update Rate Optimization. From the experimental results, we recognize that the hardware bottleneck for the emulator, in general, is the number of multipliers available. As discussed in Section IV, the fading channel coefficients need not be updated at the clock rate. Assume that there are 20 updates during each time duration of one over the maximum Doppler frequency $(1/2\pi f_M)$. From Fig. 5, it can be seen that using 10 update points per period can guarantee an MSE of less than 10^{-4}. With high levels of mobility (1000 Hz Doppler), the corresponding update rate is $20,000$ points/second. According to our measurements, the highest clock rate can reach 9600 ps which corresponds to the frequency of $104,170,000$ Hz. Therefore, the number of multipliers is no longer the bottleneck to increasing the network emulation scale.

Fig. 15 presents the architecture of multiplier temporal reuse for the in-phase component. Each color bar represents different time slices allocated to generate the fading channel on each multipath component. From the architecture, to implement the temporal reuse on one multiplier, two multiplexers and four registers are needed. Also, one encoder is taken as the control circuit built with basic logic gates. In the condition of 16 bits per variable, the implementation of the in-phase components for four components require 32 of the 4-input LUTs and 64 flip-flops. Indeed, more than four components can be implemented by multiplexing one multiplier. To guarantee low latency, one multiplier is used for 20 components.

B. Network Scale Analysis

We now estimate the achievable scale on the selected FPGA chip: Virtex-4 VFX100. With the same update rate, the top of Table II shows the consumption of RAM, flip-flops, and LUTs versus word lengths for one tap. The bottom of Table II presents the expenditure versus taps with a fixed word length. From the two figures, we can intuitively see the required hardware resources are small compared with the total.

TABLE II
Hardware Resource Consumption

Word Length	8 Bits	10 Bits	12 Bits	14 Bits	16 Bits	Total Resource
RAMB16s	1	1	1	1	1	376
Flip-Flops	602	700	792	896	994	84352
LUTs	618	712	807	901	995	84352
Taps	1	2	3	4	5	Total Resource
RAMB16s	1	2	3	4	5	376
Flip-Flops	994	1988	2982	3976	4970	84352
LUTs	995	1990	2985	3980	4975	84352

We now analytically compute the number of channels that can be emulated. Let variables W, L, N, and T denote, respectively, the word length, the number of multipath components per tap, number of taps per channel, and the number of channels. Using simple enumeration, it can be shown that the total RAM required equals TN. We also calculate the number of flip-flops and LUTs required for several different values of the parameters. Based on these values, the number of flip-flops required is of the form $TN(2LW - 13L + 324)$. Including the number of flip-flops required for the controller

circuit to enable temporal multiplexing of the multipliers, the total number of flip-flops required is of the form $TN(2LW - 13L + 324) + \lceil \frac{TNL}{20} \rceil 20W$. Similarly, it can be shown that the total number of LUTs needed to support this network equals $TN(2LW + 4L + 161) + \lceil \frac{TNL}{20} \rceil 40W$. The selected Virtex-4 FPGA has 376 RAMB16s, 84352 flip-flops, and 84352 LUTs. Thus, the selected network and channel parameters can be emulated if the following conditions are satisfied:

$$\begin{cases} TN \leq 376 \\ TN(2LW - 13L + 324) + \left\lceil \frac{TNL}{20} \right\rceil 20W \leq 84352 \\ TN(2LW + 4L + 161) + \left\lceil \frac{TNL}{20} \right\rceil 40W \leq 84352 \end{cases} \tag{19}$$

Based on the derived constraint equations, the maximum wireless network scale can be estimated. Assume the word length is 16 bits (W=16), and each tap consists of 10 components (L=10). According to (19), the maximum product of T and N is 66. Thus, for instance, 11 links of an ITU-Vehicle B channel can be emulated. If the word length is reduced to 12 bits and we use 8 components per tap, the achievable networks scale is 24 links for a single FPGA.

VII. Related Work

Due to their extensive use and low price, FPGAs are often selected as the hardware platform for fast-fading channel emulation [11]–[15]. Rao et al. [12] developed an architecture for VLSI implementation of mobile wireless communication channels based on Xiao's improvement [6] to the Jakes' model. Fard et al. [14] and Nasr [16] designed channel emulators based on the principle of filtering white Gaussian random variables. All these works consider the emulation of a single link and do not focus on the scalability of their solutions nor on the optimization of hardware resources.

Eslami [17] presented a design and implementation of a frequency-based, scalable channel emulator which is based on an FFT/IFFT fading channel generation method. The FFT was implemented using the CORDIC algorithm. Such a fading channel emulation solution has several drawbacks. First, the generated channel data length is fixed, which is determined by the size of the FFT. Thus, the scalability of the emulator is limited in terms of data generation flexibility. Second, the CORDIC algorithm can save hardware resources, but the computational speed is relatively slow. Third, even though Eslami saved hardware resources compared to the other FIR filter based emulator, the number of multipliers required is still large. Buscemi and Sass [18] described the feasibility of a scalable wireless channel emulator based on a 64-FPGA cluster. Clearly, the price of the system would preclude widespread usability. Scalable network emulator design is also addressed in Koizumi et al. [19] and Zheng and Ni [20]. However, the focus is on the general structure of the emulator and no details on the link-level channels are provided.

VIII. Conclusion

This paper discussed the design of a scalable wireless network emulator by analytically and experimentally studying the tradeoff in channel accuracy and implementation resource consumption. To do so, we presented and analyzed two key techniques: an iterative-based method for generating Rayleigh fading channels and a joint optimization of the update rate and word length. Extensive experimentation was performed and compared against both idealized models generated by a simulator and a high-fidelity commercial emulator. We leveraged our single channel analysis to understand the implementation resources necessary to build a large-scale network emulator. Using our methods which also included a multiplexing method for reducing the multiplier usage, a single FPGA could emulate up to 24 vehicular channels in real-time.

References

[1] "Azimuth ACE - MIMO Channel Emulator for Broadband Wireless Testing," http://www.azimuthsystems.com/platforms_channel_mx.htm, Jul 2012.

[2] "The Network Simulator NS-2," http://www.isi.edu/nsnam/ns/.

[3] K. C. B. et al., "FPGA-based channel simulator for a wireless network emulator," in *Proc. of IEEE VTC*, Barcelona, Spain, Apr 2009.

[4] R. Clarke, "A statistical theory of mobile-radio perception," *Bell Syst. Tech. J.*, vol. 47, pp. 957 –1000, 1968.

[5] W. Jake, *Microwave Mobile Communication*. Piscataway, NJ: Wiley-IEEE Press, 1974.

[6] C. X. et al., "Novel sum-of-sinusoids simulation models for rayleigh and rician fading channels," *Wireless Communications, IEEE Tran. on*, vol. 5, no. 12, pp. 3667 –3679, Dec 2006.

[7] J. X. et al., "Fpga-accelerated real-time volume rendering for 3d medical image," in *Biomedical Engineering and Informatics (BMEI), 2010 3rd Int. Conf. on*, vol. 1, Oct 2010, pp. 273 –276.

[8] G. W. et al., "A high performance and memory efficient lu decomposer on fpgas," *Computers, IEEE Tran. on*, vol. 61, no. 3, pp. 366 –378, Mar 2012.

[9] A. Alimohammad and B. Cockburn, "Modeling and hardware implementation aspects of fading channel simulators," *Vehicular Technology, IEEE Tran. on*, vol. 57, no. 4, pp. 2055 –2069, Jul 2008.

[10] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model Comput. Simul.*, vol. 8, no. 1, pp. 3 – 30, May 1998.

[11] A. e. a. Alimohammad, "An improved sos-based fading channel emulator," in *Vehicular Technology Conf., 2007. VTC-2007 Fall. 2007 IEEE 66th*, 30 2007-Oct. 3 2007, pp. 931 –935.

[12] G. Rao, R. Bhattacharjee, and S. Nandi, "Vlsi architecture for rayleigh and rician fading generators," in *TENCON 2004. 2004 IEEE Region 10 Conf.*, vol. C, Nov 2004, pp. 121 – 124 Vol. 3.

[13] K. e. a. Borries, "FPGA-based channel simulator for a wireless network emulator," in *Vehicular Technology Conf., 2009. VTC Spring 2009. IEEE 69th*, Apr 2009, pp. 1 –5.

[14] S. e. a. Fard, "A single fpga filter-based multipath fading emulator," in *Global Telecommunications Conf., 2009. GLOBECOM 2009. IEEE*, 30 2009-Dec. 4 2009, pp. 1 –5.

[15] J.-K. H. et al., "Fast fpga prototyping of a multipath fading channel emulator via high-level design," in *Communications and Information Technologies, 2007. ISCIT '07. Int. Symposium on*, Oct 2007, pp. 168 –171.

[16] O. Nasr and B. Daneshrad, "Design and fpga implementation an accurate real time 3x4 mimo channel emulator," in *Signals, Systems and Computers, 2009 Conf. Record of the Forty-Third Asilomar Conf. on*, Nov 2009, pp. 764 –768.

[17] H. e. a. Eslami, "Design and implementation of a scalable channel emulator for wideband mimo systems," *Vehicular Technology, IEEE Tran. on*, vol. 58, no. 9, pp. 4698 –4709, Nov 2009.

[18] S. Buscemi and R. Sass, "Design of a scalable digital wireless channel emulator for networking radios," in *Military Communications Conf., 2011 - MILCOM 2011*, Nov 2011, pp. 1858 –1863.

[19] M. e. a. Koizumi, "Design and implementation of scalable distributed wireless network emulator for high-speed mobility," in *Information Networking (ICOIN), 2012 Int. Conf. on*, Feb 2012, pp. 302 –307.

[20] P. Zheng and L. Ni, "Empower: a network emulator for wireline and wireless networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conf. of the IEEE Computer and Communications. IEEE Societies*, vol. 3, Mar.-3 Apr 2003, pp. 1933 – 1942 vol.3.