Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 3517489, 9 pages
https://doi.org/10.1155/2018/3517489

*Research Article*

# A Triply Selective MIMO Channel Simulator Using GPUs

**R. Carrasco-Alvarez,[1] R. Carreón-Villal,[1] J. Vázquez Castillo,[2] J. Ortegón Aguilar,[2] O. Longoria-Gandara,[3] and A. Castillo Atoche[4]**

[1]*Department of Electronic Engineering, UDG-CUCEI, 44430 Guadalajara, JAL, Mexico*
[2]*Department of Engineering, Universidad de Quintana Roo, 77019 Chetumal, QROO, Mexico*
[3]*Department of Electronics, Systems, and IT, ITESO, 45604 Tlaquepaque, JAL, Mexico*
[4]*Department of Mechatronics, Universidad Autónoma de Yucatán, 97000 Mérida, YUC, Mexico*

Correspondence should be addressed to R. Carrasco-Alvarez; r.carrasco@academicos.udg.mx

Received 29 September 2017; Revised 23 January 2018; Accepted 6 February 2018; Published 5 March 2018

Academic Editor: Neji Youssef



A methodology for implementing a triply selective multiple-input multiple-output (MIMO) simulator based on graphics processing units (GPUs) is presented. The resulting simulator is based on the implementation of multiple double-selective single-input single-output (SISO) channel generators, where the multiple inputs and the multiple received signals have been transformed in order to supply the corresponding space correlation of the channel under consideration. A direct consequence of this approach is the flexibility provided, which allows different propagation statistics to each SISO channel to be specified and thus more complex environments to be replicated. It is shown that under some specific constraints, the statistics of the triply selective MIMO simulator are the same as those reported in the state of art. Simulation results show the computational time improvement achieved, up to 650-fold for an $8 \times 8$ MIMO channel simulator when compared with sequential implementations. In addition to the computational improvement, the proposed simulator offers flexibility for testing a variety of scenarios in vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) systems.

## 1. Introduction

With the growing demand by users of rapid transfer of high amounts of data, it has been necessary to develop new transmissions strategies and, consequently, to verify their performance in order to accomplish the established goals. In this sense, multiple-input multiple-output (MIMO) simulator has been considered in recent years a fundamental part of new communication standards embraced by long-term evolution-vehicle (LTE-V) and vehicle-to-vehicle (V2V) technologies. This is due to the fact that MIMO can take advantage of space diversity and multipath scattering for increasing the data transmission rate considerably in comparison with single-input single-output (SISO) communication systems [1]. Because of its relevance, several mathematical channel models and hence diverse channel simulators/emulators have been proposed in order to prove the performance of MIMO-based communication systems.

In this sense, as summarized by [2], MIMO channel models can be classified in diverse ways; the broadest classification considers physical models and analytical models. Physical models take into account the electromagnetic propagation and the environment under study for obtaining a channel model. Moreover, such models can be categorized as deterministic models [3], geometric-based stochastic models [4], or stochastic models [5, 6]. Analytical models, on the other hand, abstract the complex electromagnetic propagation mechanisms into tractable channel impulse responses for modeling the MIMO channels. Moreover, analytical models can be classified as propagation-based models and correlation-based models, where the distinguished Kronecker model [7] and the Weichselberger model [8] fit in the latter classification. These models are the most commonly used models for simulating MIMO channels due to their simplicity and the conceptualization of the propagation environment.

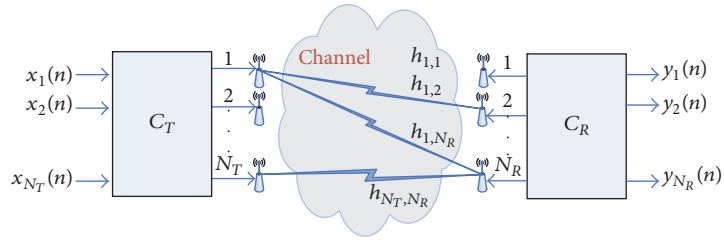


Figure 1: MIMO system of $N_T \times N_R$ antennas.

This paper considers an analytical model based on correlation, which assumes a triply selectivity; that is, the propagation channel for each transmitter and receiver antenna pair presents both time and frequency selectivity and a spatial correlation. This model was chosen to capture the nature of the propagation channel with greater precision. As expected, the implementation of this triply selective channel simulator is highly complex; for example, if a MIMO system made up of $N_T = 8$ transmitter antennas and $N_R = 8$ receiver antennas is considered, then it is necessary to implement $N_T \times N_R = 64$ independent SISO channel simulators in parallel. This number of SISO channels increases as well as the number of antennas is increased. Therefore, graphics processing units and GPU-accelerated computing techniques can help to manage the computational complexity in the MIMO channel simulator.

The GPU-accelerated computing techniques use the GPU together with CPUs, which are available in affordable computers or servers. The purpose is to accelerate scientific computation and engineering calculations, among others. As a result, several works related to wireless channel simulators have been presented in order to handle the computational complexity in the implementation of channel simulators [9–12] and high demanding digital signal processing algorithms [13, 14].

In [9], the authors present a GPU-based implementation, which uses the filtering method for developing a doubly selective SISO channel simulator (frequency and time selective). In [10], a full 3-D GPU-based beam-tracing method is presented for propagation modeling in complex indoor environments. Likewise, the study in [11] presents an improved path loss simulation incorporating a three-dimensional terrain model using parallel coprocessors or GPUs. In addition, in [12] and references therein, a selection of GPU-based implementations is presented, which improves the computational processing time and reports GPU-based implementations with significant speedups. However, even though many GPU-based implementations have been reported in the state of the art, a complete triply selective fading channel simulator has not been reported in the open literature. Examples of complete MIMO fading channel simulators and emulators can be found in [15–17], but these architectures do not exploit the triple selectivity at the same time, or they configure the hardware with a low number of antennas.

This paper presents a triply selective MIMO channel simulation methodology using GPU techniques; the methodology will be based on SISO channel generators presented in [9]. This simulator includes the phenomenology of the propagation environment (time, frequency and space selectivities), which has been left out of the state-of-art simulators due to the computational complexity. Likewise, the introduced methodology exhibits enough flexibility for implementing channel simulators with different MIMO channel configurations.

*1.1. Notation.* Bold upper (lower) case letters are used for denoting matrices (vectors); $(\cdot)^T$, $(\cdot)^H$, $\lceil\cdot\rceil$, and $E(\cdot)$ denote transpose, complex transpose (Hermitian), the ceil function, and the expectation operator, respectively. $[\mathbf{A}]_{i,j}$ denotes the element in the $i$th row and $j$th column of $\mathbf{A}$. vec($\mathbf{A}$) is the reordering of the columns of $\mathbf{A}$ into a single column vector. diag($\mathbf{a}$) is a diagonal matrix whose elements are those from vector $\mathbf{a}$. $\mathbf{I}_j$ denotes an identity matrix of length $j \times j$. Finally, $\otimes$ stands for the Kronecker product of two matrices.

*1.2. Organization.* The paper is organized as follows: in Section 2, the mathematical model of the triply selective channel is analyzed. The methodology for implementing this mathematical model using GPUs is described in Section 3. Implementation results assuming diverse scenarios are presented in Section 4. Finally, some concluding remarks in Section 5 close this paper.

## 2. Triply Selective Channel Model

Assume a baseband discrete time MIMO communication channel that presents time, frequency, and spatial selectivity (triple-selective channel). Moreover, consider that this communication system is conformed of $N_T$ transmit antennas and $N_R$ receive antennas as depicted in Figure 1. This system can be envisaged as an array of $N_T \times N_R$ SISO channels where a transmitter and a receiver correlation stage are included in order to provide the corresponding spatial correlation statistics. Without loss of generality, if it is stated that all the SISO channels are modeled as FIR filters of $L$ coefficients, then the MIMO system at time index $n$ can be expressed mathematically as follows:

$$\mathbf{y}(n) = \mathbf{C}_R \mathbf{H}(n) \widehat{\mathbf{C}}_T \mathbf{x}(n), \quad (1)$$

where $\mathbf{y}(n) \in \mathbb{C}^{N_R}$ with $\mathbf{y}(n) = [y_1(n), y_2(n), \ldots, y_{N_R}(n)]^T$ is a vector containing the samples received from each antenna, $\mathbf{C}_R \in \mathbb{C}^{N_R \times N_R}$ is the matrix that provides the spatial correlation due to the receive antennas, and $\widehat{\mathbf{C}}_T = \mathbf{C}_T \otimes \mathbf{I}_L$,

$\mathbf{C}_T \in \mathbb{C}^{N_T \times N_T}$ correlates the transmitted samples according to the correlation of the transmit antennas.

Additionally, $\mathbf{x}(n) \in \mathbb{C}^{N_T L}$, where $\mathbf{x}(n) = [\mathbf{x}_1^T(n), \mathbf{x}_2^T(n), \ldots, \mathbf{x}_{N_T}^T(n)]^T$ is a vector composed of the actual and past samples of each transmitter, where $\mathbf{x}_i(n) = [x_i(n), x_i(n-1), \ldots, x_i(n-L)]^T$. The matrix $\mathbf{H}(n) \in \mathbb{C}^{N_R \times N_T L}$ is defined as follows:

$$\mathbf{H}(n) = \begin{bmatrix} \mathbf{h}_{1,1}^T(n) & \mathbf{h}_{2,1}^T(n) & \cdots & \mathbf{h}_{N_T,1}^T(n) \\ \mathbf{h}_{1,2}^T(n) & \mathbf{h}_{2,2}^T(n) & \cdots & \mathbf{h}_{N_T,2}^T(n) \\ \vdots & & \ddots & \vdots \\ \mathbf{h}_{1,N_R}^T(n) & \cdots & \cdots & \mathbf{h}_{N_T,N_R}^T(n) \end{bmatrix}, \quad (2)$$

where $\mathbf{h}_{i,j}(n) = [h_{i,j}(n,1), h_{i,j}(n,2), \ldots, h_{i,j}(n,L)]^T$ is the discrete time-varying channel impulse response from the transmitter $i$ to the receiver $j$. The value $L$ is selected as $\max(\lceil(\tau_{\max_{i,j}} + T_B)/T_s\rceil)\ \forall i = 1, \ldots, N_T;\ \forall j = 1, \ldots, N_R$, where $\tau_{\max_{i,j}}$ is the maximum delay corresponding to the subchannel from transmitter $i$ to receiver $j$, $T_B$ is the impulse response duration of the combined transmitter and receiver filters, and $T_s$ is the symbol period.

Thus, it is possible to consider the MIMO channel as

$$\overline{\mathbf{H}}(n) = \mathbf{C}_R \mathbf{H}(n) \widehat{\mathbf{C}}_T, \quad (3)$$

whose autocorrelation function can be stated as follows:

$$\mathbf{R}_{\overline{\mathbf{H}}}(n_1, n_2) = E\left(\operatorname{vec}\left(\overline{\mathbf{H}}(n_1)\right) \operatorname{vec}\left(\overline{\mathbf{H}}(n_2)\right)^H\right). \quad (4)$$

This channel model can be interpreted as a form of the Kronecker model reviewed in [7, 18], where time and frequency selectivity have also been considered. In order to simplify the calculations, autocorrelation of $\overline{\mathbf{H}}^T(n)$ is analyzed instead, which does not affect the statistics of the channel under analysis. Thus, the autocorrelation of $\overline{\mathbf{H}}^T(n)$ is

$$\begin{aligned} \mathbf{R}_{\overline{\mathbf{H}}^T}(n_1, n_2) &= E\left(\operatorname{vec}\left(\overline{\mathbf{H}}^T(n_1)\right) \operatorname{vec}\left(\overline{\mathbf{H}}^T(n_2)\right)^H\right) \\ &= \left(\mathbf{C}_R \otimes \widehat{\mathbf{C}}_T^T\right) E\left(\operatorname{vec}\left(\mathbf{H}^T(n_1)\right) \operatorname{vec}\left(\mathbf{H}^T(n_2)\right)^H\right) \\ &\cdot \left(\mathbf{C}_R \otimes \widehat{\mathbf{C}}_T^T\right)^H = \left(\mathbf{C}_R \otimes \widehat{\mathbf{C}}_T^T\right) \mathbf{R}_{\mathbf{H}^T}(n_1, n_2) \\ &\cdot \left(\mathbf{C}_R \otimes \widehat{\mathbf{C}}_T^T\right)^H, \end{aligned} \quad (5)$$

where it has already made use of $\overline{\mathbf{H}}^T(n) = \widehat{\mathbf{C}}_T^T \mathbf{H}^T(n) \mathbf{C}_R^T$ and the fact that $\operatorname{vec}(\widehat{\mathbf{C}}_T^T \mathbf{H}^{\mathrm{T}}(n) \mathbf{C}_R^T) = (\mathbf{C}_R \otimes \widehat{\mathbf{C}}_T^T) \operatorname{vec}(\mathbf{H}(n))$; in addition, $\mathbf{R}_{\mathbf{H}^T}(n_1, n_2)$ is the autocorrelation function of $\mathbf{H}^T(n)$. If it is assumed that all the subchannels in $\mathbf{H}^T(n)$ are

(i) uncorrelated among themselves,

(ii) wide-sense stationary uncorrelated scattering (WSSUS),

(iii) Rayleigh,

then, $\operatorname{vec}(\mathbf{H}^T(n))$ can be expressed as

$$\begin{aligned} \operatorname{vec}\left(\mathbf{H}^T(n)\right) &= \begin{bmatrix} \mathbf{h}_{1,1}(n) \\ \mathbf{h}_{2,1}(n) \\ \vdots \\ \mathbf{h}_{N_T,1}(n) \\ \mathbf{h}_{1,2}(n) \\ \vdots \\ \mathbf{h}_{N_T,N_R}(n) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{G}_{1,1} & & & & & & & \\ & \mathbf{G}_{2,1} & & & & & & \\ & & \ddots & & & & & \\ & & & \mathbf{G}_{N_T,1} & & & & \\ & & & & \mathbf{G}_{1,2} & & & \\ & & & & & \ddots & & \\ & & & & & & \mathbf{G}_{N_T,N_R} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{1,1}(n) \\ \mathbf{w}_{2,1}(n) \\ \vdots \\ \mathbf{w}_{N_T,1}(n) \\ \mathbf{w}_{1,2}(n) \\ \vdots \\ \mathbf{w}_{N_T,N_R}(n) \end{bmatrix} \\ &= \boldsymbol{\mathscr{G}} \boldsymbol{\omega}(n). \end{aligned} \quad (6)$$

From (6), $\mathbf{w}_{i,j}(n) \in \mathbb{C}^{P_{i,j} \times 1}$ is a zero-mean circularly symmetric complex Gaussian random vector with an autocorrelation function:

$$E\left(\mathbf{w}_{i,j}(n_1) \mathbf{w}_{i,j}^H(n_2)\right) = \boldsymbol{\Sigma}_{i,j}^2 \boldsymbol{\Lambda}_{i,j}(n_1 - n_2), \quad (7)$$

where $\boldsymbol{\Sigma}_{i,j}^2$ and $\boldsymbol{\Lambda}_{i,j}(n_1 - n_2)$ are both diagonal matrices, such that $[\boldsymbol{\Sigma}_{i,j}^2]_{m,m} = \sigma_m^2$ is the power of the $m$th path of the subchannel from antenna $i$ to antenna $j$ with delay $\tau_m^{i,j}$ and autocorrelation function $[\boldsymbol{\Lambda}_{i,j}(n_1 - n_2)]_{m,m}$ evaluated at the instant difference $n_1 - n_2$ for $m = 1, \ldots, P_{i,j}$, where $P_{i,j}$ is the number of paths of the subchannel from $i$ to $j$.

Let $\boldsymbol{\mathscr{G}}$ be a block diagonal matrix, where the elements of each block $\mathbf{G}_{i,j} \in \mathbb{C}^{L \times P_{i,j}}$ are given by

$$\left[\mathbf{G}_{i,j}\right]_{l,m} = g\left((l-1)T_s - \tau_m^{i,j}\right), \quad (8)$$

where $g(t)$ is the impulse response of the band limiting filter, for example, sinc or raised cosine filter with duration $T_B$, $l = 1, \ldots, L$, and $m = 1, \ldots, P_{i,j}$.

From (6), $\mathbf{R}_{\mathbf{H}^T}(n_1, n_2)$ is given by

$$\begin{aligned} \mathbf{R}_{\mathbf{H}^T}(n_1, n_2) &= E\left(\boldsymbol{\mathscr{G}} \boldsymbol{\omega}(n_1) \boldsymbol{\omega}(n_2)^H \boldsymbol{\mathscr{G}}^H\right) \\ &= \boldsymbol{\mathscr{G}} \left(\widetilde{\boldsymbol{\Sigma}}^2 \widetilde{\boldsymbol{\Lambda}}(n_1 - n_2)\right) \boldsymbol{\mathscr{G}}^H, \end{aligned} \quad (9)$$

where

$$\begin{aligned} \widetilde{\boldsymbol{\Sigma}}^2 &= \operatorname{diag}\left(\left[\boldsymbol{\Sigma}_{1,1}^2, \boldsymbol{\Sigma}_{2,1}^2, \ldots, \boldsymbol{\Sigma}_{N_T,1}^2, \boldsymbol{\Sigma}_{1,2}^2, \ldots, \boldsymbol{\Sigma}_{N_T,N_R}^2\right]\right), \\ \widetilde{\boldsymbol{\Lambda}}(n_1 - n_2) &= \operatorname{diag}\left(\left[\boldsymbol{\Lambda}_{1,1}(n_1 - n_2), \boldsymbol{\Lambda}_{2,1}(n_1 - n_2), \ldots,\right.\right. \\ &\quad \boldsymbol{\Lambda}_{N_T,1}(n_1 - n_2), \boldsymbol{\Lambda}_{1,2}(n_1 - n_2), \ldots, \\ &\quad \left.\left.\boldsymbol{\Lambda}_{N_T,N_R}(n_1 - n_2)\right]\right) \end{aligned} \quad (10)$$

are both block diagonal matrices.

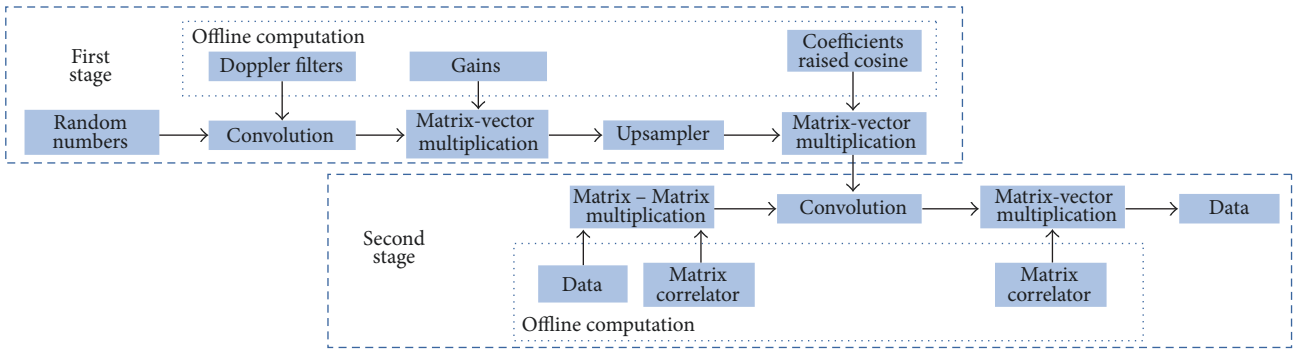


Figure 2: Triply selective MIMO channel simulator scheme.

Substituting (9) into (5), $\mathbf{R}_{\overline{\mathbf{H}}^T}(n_1, n_2)$ is

$$\mathbf{R}_{\overline{\mathbf{H}}^T}(n_1, n_2) = \left(\mathbf{C}_R \otimes \widehat{\mathbf{C}}_T^T\right) \boldsymbol{\mathcal{G}} \left(\widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{\Lambda}}(n_1 - n_2)\right) \boldsymbol{\mathcal{G}}^H \left(\mathbf{C}_R \otimes \widehat{\mathbf{C}}_T^T\right)^H. \tag{11}$$

Function (11) is the autocorrelation function of the proposed model (3). It is possible to observe that this model offers great flexibility by allowing the subchannels to have different statistics, for example, different power delay profile (PDP) and power Doppler spectrum (PDS). As a special case, if all the paths of all subchannels are forced to have the same PDS, then $\widetilde{\boldsymbol{\Lambda}}(n_1 - n_2) = \mathbf{I} f(n_1 - n_2)$, where $f(n_1 - n_2)$ is the autocorrelation function of all the paths. As a particular case, if Jakes' model is considered, where the different scatters propagate in the azimuth plane and arrive at the receivers with an angle of arrival distributed uniformly between $[0, 2\pi]$, then $f(n_1 - n_2) = J_0(2\pi f_{\max}(n_1 - n_2)T_s)$, where $J_0(\cdot)$ is the Bessel function of order 0 and $f_{\max}$ is the maximum Doppler frequency. If it is also considered that all the subchannels have the same PDP composed of $P$ paths, then $\widetilde{\boldsymbol{\Sigma}}^2 = \mathbf{I}_{N_T N_R} \otimes \boldsymbol{\Sigma}_e^2$ and $\boldsymbol{\mathcal{G}} = \mathbf{I}_{N_T N_R} \otimes \mathbf{G}_e$, where $\boldsymbol{\Sigma}_e^2 \in \mathbb{R}^{P \times P}$ is a diagonal matrix containing the gains of the paths and $\mathbf{G}_e \in \mathbb{R}^{L \times P}$ is calculated as in (8). However, assuming that the paths of all the subchannels have the same delay, then (11) transforms into (12), which coincides with the model proposed in [16]:

$$\begin{aligned}
\mathbf{R}_{\overline{\mathbf{H}}^T}(n_1, n_2) &= \Big(\left(\mathbf{C}_R \otimes (\mathbf{C}_T \otimes \mathbf{I}_L)^T\right)\left(\mathbf{I}_{N_T N_R} \otimes \mathbf{G}_e\right) \\
&\cdot \left(\mathbf{I}_{N_T N_R} \otimes \boldsymbol{\Sigma}_e^2\right)\left(\mathbf{I}_{N_T N_R} \otimes \mathbf{G}_e\right)^H \\
&\cdot \left(\mathbf{C}_R \otimes (\mathbf{C}_T \otimes \mathbf{I}_L)^T\right)^H\Big) J_0\left(2\pi f_{\max}(n_1 - n_2)T_s\right) \\
&= \Big(\left(\left(\mathbf{C}_R \otimes \mathbf{C}_T^T\right) \otimes \mathbf{I}_L^T\right)\left(\mathbf{I}_{N_T N_R} \otimes \mathbf{G}_e \boldsymbol{\Sigma}_e^2 \mathbf{G}_e^H\right) \\
&\cdot \left(\left(\mathbf{C}_R \otimes \mathbf{C}_T^T\right) \otimes \mathbf{I}_L^T\right)^H\Big) J_0\left(2\pi f_{\max}(n_1 - n_2)T_s\right) \\
&= \Big(\left(\left(\mathbf{C}_R \otimes \mathbf{C}_T^T\right)\left(\mathbf{C}_R \otimes \mathbf{C}_T^T\right)^H\right) \otimes \left(\mathbf{G}_e \boldsymbol{\Sigma}_e^2 \mathbf{G}_e^H\right)\Big) \\
&\cdot J_0\left(2\pi f_{\max}(n_1 - n_2)T_s\right) = \Big(\mathbf{C}_R \mathbf{C}_R^H \otimes \mathbf{C}_T^T \left(\mathbf{C}_T^T\right)^H \\
&\otimes \left(\mathbf{G}_e \boldsymbol{\Sigma}_e^2 \mathbf{G}_e^H\right)\Big) J_0\left(2\pi f_{\max}(n_1 - n_2)T_s\right).
\end{aligned} \tag{12}$$

## 3. GPU Implementation

The simulator implementation takes advantage of the parallel capabilities of General-Purpose Computing on GPU (GPGPU) and the use of the VexCL library [19]. VexCL is an OpenCL/CUDA library developed in C++ by Denis Demidov. It supports multidevice, multiplatform computations and provides functions for floating-point vector/matrix operations. VexCL uses vector expressions, which are automatically processed in parallel across all devices.

The simulation process comprises two stages: channel coefficient generation and data frame processing. These stages are depicted in Figure 2. The first stage corresponds to the generation of the elements of matrix $\mathbf{H}$ of (2); the second stage represents (1).

*3.1. Complex Operations.* GPU frameworks, like OpenCL and CUDA, do not provide complex data types; the closest data type is the 2-vector type `cl_double2` of OpenCL. Hence, custom vex functions were implemented to perform addition and multiplication of complex numbers. These functions are used in vector expressions. The codes implemented for performing the multiplication and addition of complex numbers are presented in Listings 1 and 2, respectively. The previous functions receive two 2-vectors used as complex numbers.

*3.2. Channel Coefficient Generation.* The channel coefficients require random numbers, which are filtered for generating the channel taps with specific PDS and correlation.

*3.2.1. Random Number Generation.* VexCL provides templates to generate random numbers; they are based on Random123 [20, 21]. The templates support the Philox generator family (based on integer multiplication) or the Threefry family (based on Threefish encryption). It is possible to generate uniformly distributed random numbers with the Random

```
VEX_FUNCTION(cl_double2, cmul,
   (cl_double2, a) (cl_double2, b),
   double2 c = {
        a.x * b.x - a.y * b.y,
        a.x * b.y + a.y * b.x
   };
   return c;
);
```

Listing 1: Multiplication code.

```
VEX_FUNCTION(cl_double2, csum,
   (cl_double2, a) (cl_double2, b),
   double2 c = {
        a.x + b.x, a.y + b.y
   };
   return c;
);
```

Listing 2: Sum code.

template. Additionally, there are RandomNormal templates that use the Box-Muller transform to generate normally distributed random numbers. In this implementation,

```
vex::RandomNormal<cl_double2,
vex::random::threefry > random_numbers
```

are used as a C++ functor. A double random number vector is generated with

```
noise=random_numbers(vex::element_index(),
123),
```

where `noise` is a vector of $nSamples \times \sum_{i=1}^{N_T} \sum_{j=1}^{N_R} P_{i,j}$ of pairs of double precision floating-point numbers, `vex::element_index()` is the function to get the $n$th random number, and 123 is the seed for the generator. Instead of using doubles, the vector noise is composed of `cl_double2` numbers; for example, a pair of double precision numbers is used as complex numbers.

*3.2.2. Doppler Filter.* The generated complex random numbers are passed through a filter which provides the corresponding time domain statistics. The transfer function of this filter is the square root of the autocorrelation function in the time domain for each path. As a particular case where all the paths in all the subchannels follow the Jakes model, the impulse response of this filter is $\Gamma(3/4)(f_{\max}/(\pi|(k - tFilter/2)T_s|))^{1/4} J_{1/4}(f_{\max}|(k-tFilter/2)T_s|)$, where $\Gamma(\cdot)$ is the gamma function, $tFilter$ is the length of the filter, and $k = 0, \ldots, tFilter$ is an index that enumerates the coefficients [22]. In order to perform the noise filtering, a custom function, named convolution, was coded; it receives pointers to arrays of 2-vectors corresponding to the noise and Doppler filter

```
VEX_FUNCTION_D(cl_double2, convolution,
   (size_t, i)(cl_double2*, x)
   (double*, y)(int, tF)(int, fS)
   (int, nS)(int, nP),(csum)
   (cmulscalar),
   double2 sum = {0.0,0.0};
   int tmp1 = i+(i/fS) * (nS-fS);
   int tmp2 = tF * (i/(fS*nP)+1) - 1;
   for(size_t j = 0; j < tF; j++)
     sum = csum(sum,
       cmulscalar(x[tmp1+j],y[tmp2-j]));
   return sum;
);
```

Listing 3: Filtering code.

coefficients, respectively. The code used for this convolution is presented in Listing 3, where `i` is the OpenCL's element index, `x` is the noise, `y` is the filter, `tF` is the length of the filter, `fS` is the number of samples to be filtered, `nS` is the number of samples, and `nP` is the number of paths. The convolution function processes in parallel the complete noise vector. It executes the function body for each noise sample.

*3.2.3. Path Gains.* They are implemented as the product of a vector and a scalar. Each path has its own gain.

*3.2.4. Upsampling.* The upsampling is a quadratic interpolation to a desired number of values of the noise vector.

*3.2.5. Tap Generation.* The resulting noise vector, representing the paths, is correlated with a matrix $\mathbf{G}_{i,j}$ (e.g., raised cosine). This correlation is implemented as a matrix-matrix product. A MIMO system has multiple channels; the correlation is done for each channel.

*3.3. Data Frame Processing.* This section describes the implementation of Figure 1 and (1): a transmitter and a receiver correlation stage and the discrete time-varying channel impulse response from the transmitter $i$ to the receiver $j$.

*3.3.1. Transmitter Correlation.* The correlation on the transmitter side is implemented as a product of the data $\mathbf{x}(n)$ and correlation matrices $\mathbf{C}_T$; the latter matrix contains the coefficients that correlate data frames and the transmit antennas.

*3.3.2. FIR Filter.* The transmitted data frames are filtered with the channel coefficients, $\mathbf{H}(n)$, described in Section 3.2. In order to achieve this goal, a convolution is implemented. The code used for this convolution is presented in Listing 4, where the variable `i` is the OpenCL's element index, the variable `x` is a pointer to the data to be processed, the variable `y` is a pointer to the generated channel coefficients, the variable `tRc` is the number of taps of the raised cosine, the variable `nD` is the data frame size, the variable `uS` is the size of the path, the variable

```
VEX_FUNCTION_D(cl_double2, convolution2,
   (size_t, i) (cl_double2*, x)
   (cl_double2*, y)(int, tRc)(int, nD)
   (int, uS)(int, nT)(int, nR),(cmul)
   (csum),
   double2 sum = {0.0,0.0};
   int tmp1 = i%nD;
   int tmp2 = i/nD;
   int ext = uS * tRc;
   int limt = tRc;
   if(tmp1 < tRc)
      limt = tmp1 + 1;
      for(size_t k=0; k<nT; k++)
        for(size_t j=0; j<limt; j++)
          sum = csum(sum, cmul(
            x[tmp1-j+k*nD],
            y[tmp1+j*uS+ext*(k*nR+tmp2)]
          ));
   return sum;
);
```

Listing 4: Convolution code.

Table 1: Power delay profiles for configuring the SISO subchannels of the MIMO 2 × 2 simulator.

| Subchannel | Delay (nsec) |
|---|---|
| $\mathbf{h}_{1,1}$ | [0, 310, 710, 1090, 1730, 2510] |
| $\mathbf{h}_{1,2}$ | [0, 300, 8900, 12900, 17100, 20000] |
| $\mathbf{h}_{2,1}$ | [0, 4450, 8900, 12300] |
| $\mathbf{h}_{2,2}$ | [400, 710, 800, 920, 1200] |
| | Gain (dB) |
| $\mathbf{h}_{1,1}$ | [0, −1, −9, −10, −15, −20] |
| $\mathbf{h}_{1,2}$ | [−2.5, 0, −12.8, −10, −25.2, −16] |
| $\mathbf{h}_{2,1}$ | [0, −1, −5, −8] |
| $\mathbf{h}_{2,2}$ | [−2, 0, −2, −8, −9] |

`nT` is the number of transmitters, and the variable `nR` is the number of receivers.

*3.3.3. Receiver Correlation.* The correlation on the receiver side is implemented as a product of the received data and the correlation matrix $\mathbf{C}_R$; the latter matrix contains the coefficients that correlate data frames and the receive antennas.

## 4. Results

In this section, the time performance of the proposed simulator is evaluated. In order to achieve this goal, a MIMO 2 × 2 channel is considered, where all the subchannels have the same PDS with Jakes shape and $f_{\max} = 2000$ Hz. If a carrier frequency of 5.9 GHz is considered, then the assumed $f_{\max}$ corresponds to a scenario where the speed of the mobile is 360 km/h. Likewise, the PDP of each SISO subchannel is fixed as described in Table 1. The PDPs are selected in order to prove the functionality of the simulator, with the PDPs of $\mathbf{h}_{1,1}$ and $\mathbf{h}_{1,2}$ corresponding to the vehicular test environment channel A and channel B as defined in [23] respectively. The values of matrices $\mathbf{C}_R$ and $\mathbf{C}_T$ are fixed in such a way that they make $[\mathbf{C}_R\mathbf{C}_R^H]_{i,j} = 0.3^{|i-j|}$ and $[\mathbf{C}_T\mathbf{C}_T^H]_{i,j} = 0.9^{|i-j|}$, respectively [24]. It is assumed that the transmitter and the receiver filters are both a square root raised cosine with a rolloff factor of 0.5 and the duration of its convolution is $2T_B = 6T_s$, where the period symbol is fixed at $T_s = 0.1\,\mu$s. According to the maximum delay values presented in Table 1, as well as the values of $T_B$ and $T_s$, the parameter $L$ is equal to 206 taps. Finally, the MIMO simulator generates channels assuming data frames composed of 1024 symbols.

Table 2: Time consumption by each module of the proposed simulator considering an 8 × 8 MIMO simulator.

| Module | Time (%) |
|---|---|
| Multiple matrix-matrix multiplications | 74.23 |
| Simulation | 19.55 |
| Upsampling | 3.62 |
| Gaussian random number generator | 1.25 |
| Doppler filter | .92 |
| Path gain | .43 |

Figure 3 shows a parallel realization of each MIMO subchannel using the proposed GPU-based simulator, where each of the subfigures presents the time-variation of the corresponding filter coefficients, which are associated with the assigned PDS. Moreover, it is possible to observe that each subchannel satisfies the specifications of the PDP assigned. Thus, PDPs with large delays correspond to filters with more coefficients, as observed in the channel realizations $h_{1,2}$ and $h_{2,1}$, respectively.

The time performance evaluation is carried out using a personal computer (PC) with the following specifications:

(i) Fedora 25, 64 bits

(ii) Intel core i7-920 (2.66 GHz)

(iii) 12 GB DDR3 RAM

(iv) Graphics card Asus GTX Titan Black with 6 GB of RAM and 2880 CUDA cores.

In order to evaluate the time performance, 15 different scenarios of triply selective $N_R \times N_T$ MIMO channels are considered: 2 × 2, 3 × 3, 4 × 4, 5 × 5, 6 × 6, 7 × 7, 8 × 8, 10 × 10, 12 × 12, 14 × 14, 16 × 16, 18 × 18, 20 × 20, 22 × 22, and 24 × 24. For all the cases, it is assumed that all the subchannels are configured with the same PDP (vehicular test environment channel B) and PDS (Jakes with $f_{\max} = 2000$ Hz) statistics; moreover, the matrices $\mathbf{C}_T$ and $\mathbf{C}_R$ have identical values which make $[\mathbf{C}_T\mathbf{C}_T^H]_{i,j} = [\mathbf{C}_R\mathbf{C}_R^H]_{i,j} = 0.5^{|i-j|}$ for $i = 1, \ldots, N_R$ and $i = 1, \ldots, N_T$. The rest of the simulation parameters have the same values as described previously.

The simulations are carried out over 100 frames and the elapsed processing time per frame is obtained by averaging all the frames. Table 2 shows the time required to execute each of the simulator blocks.

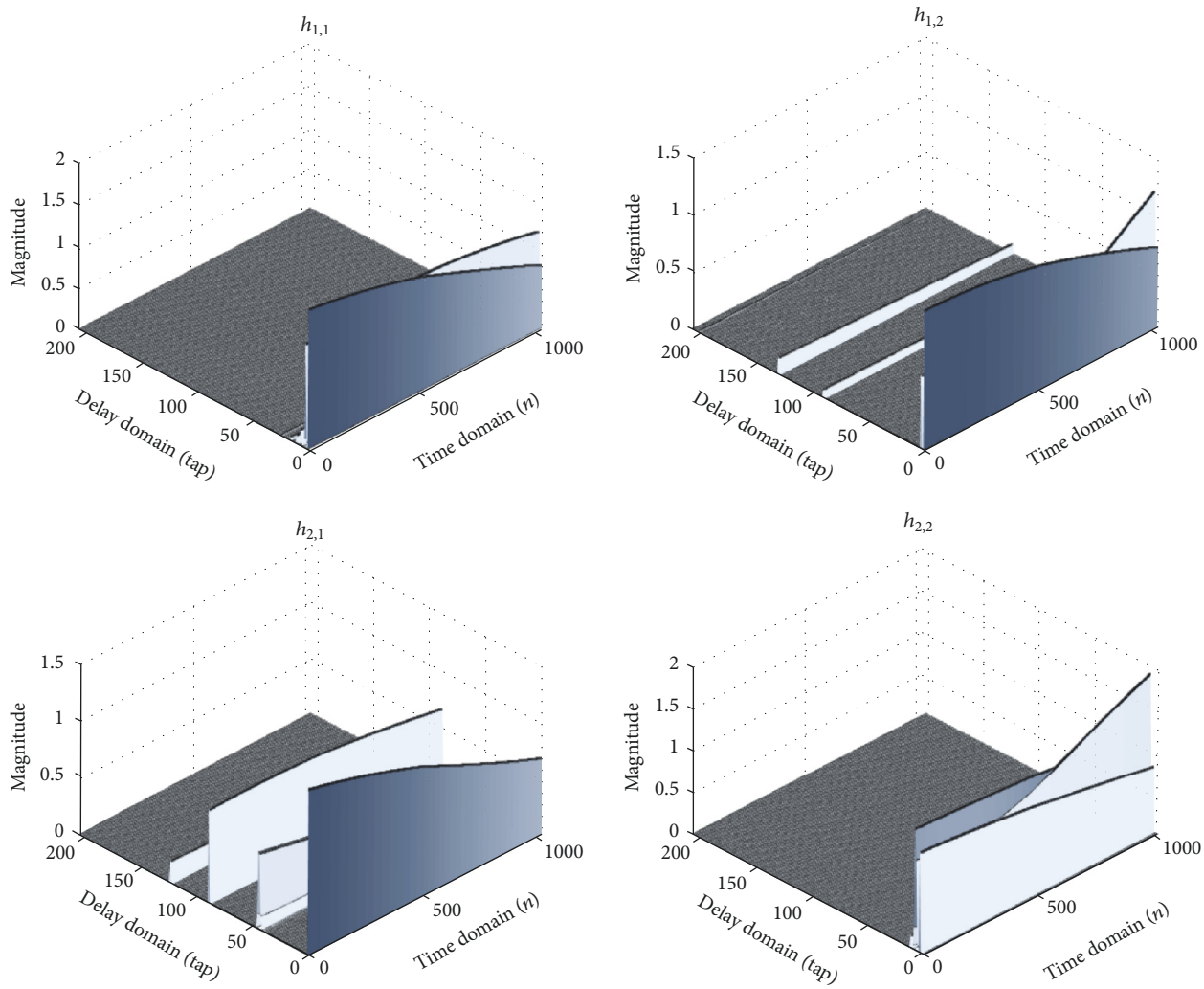


Figure 3: MIMO channel realizations.

Figure 4 presents the comparison of the time required to execute all the considered cases when the GPU-based simulator is used, as well as the sequential implementation of this simulator using C language. For a better appreciation, the results are plotted in logarithmic scale.

Table 3 summarizes the time consumption and also presents the $x$-fold time gain. This gain is calculated as the quotient of the time required by the sequential implementation divided by the GPU implementation. It is possible to observe that for the cases from $2 \times 2$ to $7 \times 7$, as the complexity of simulator increments, the gain increments too. This is due to the fact that as more operations are required, better utilization of the parallel resources of the GPU is achieved. In the remaining cases, the gain is around 680; starting at the $7 \times 7$ case, the simulator simultaneously uses the available resources in the device. As a matter of fact, with the GPU considered for testing the simulator, the $24 \times 24$ case is the most complex MIMO channel that can be simulated; nevertheless, if the hardware is upgraded, then this inconvenience can be addressed, opening the possibility for the simulation of more complex scenarios, for example, massive MIMO.

*4.1. Discussion.* Recently, several channel simulator approaches have been introduced in the open literature. However, these implementations are based on field programmable gate array (FPGA) devices, and they are restricted to using a single configuration in the developed simulator; that is, the time- and frequency-selective correlations are fixed. As a result, setting new channel statistics in the simulator (new channel propagation conditions) implies the redesign of the implemented hardware. For example, in the channel simulator introduced by [16], the time selectivity is generated by using a sum of sinusoids, which approximates a Jakes PDS. This simulator uses uniform random number generators for defining the configuration of the parameters of these sinusoids. Thus, if one wishes to generate a channel with a different PDS, then this will imply changing the probability density function of the random number generator block. In contrast, for changing the statistics of the simulator proposed in this paper, it is necessary only to upgrade the coefficients of the filters, which allows for a quick redesign of the simulator.

As regards performance, even though the reported FPGA implementations can operate in real time, they are constrained in terms of the number of operations that they

Table 3: Time consumption for different MIMO channel realizations when the GPU-based simulator and a sequential implementation are used.

| MIMO Size | GPU Implementation (ns) | Sequential Implementation (ns) | $x$-fold Gain |
|---|---|---|---|
| $2 \times 2$ | 1703 | 504223 | 296 |
| $3 \times 3$ | 2456 | 1139590 | 464 |
| $4 \times 4$ | 3719 | 2037134 | 547 |
| $5 \times 5$ | 5318 | 3176655 | 597 |
| $6 \times 6$ | 7049 | 4578364 | 649 |
| $7 \times 7$ | 9096 | 6266204 | 688 |
| $8 \times 8$ | 12542 | 8160310 | 650 |
| $10 \times 10$ | 18854 | 12726565 | 675 |
| $12 \times 12$ | 26093 | 18376942 | 704 |
| $14 \times 14$ | 35766 | 25051177 | 700 |
| $16 \times 16$ | 50766 | 32819589 | 646 |
| $18 \times 18$ | 62645 | 41863632 | 668 |
| $20 \times 20$ | 75303 | 51563394 | 684 |
| $22 \times 22$ | 91884 | 62385816 | 678 |
| $24 \times 24$ | 110943 | 74266271 | 669 |

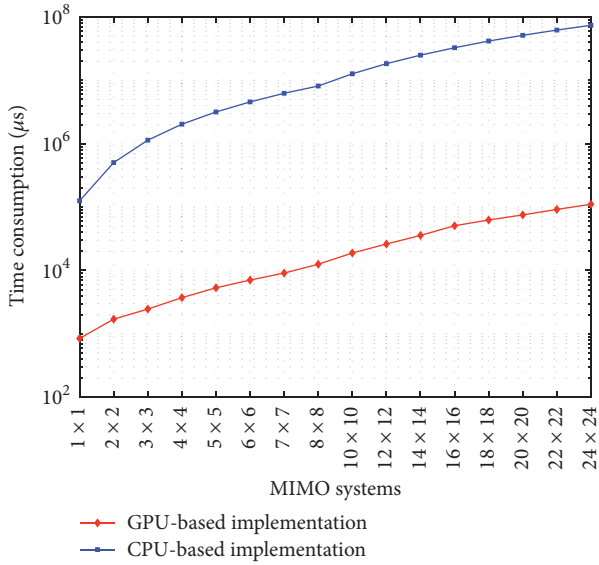


Figure 4: Comparison of execution time for different MIMO channels.

can execute. For example, in [16] it is reported that the MIMO channels which can be simulated should be limited to $N_R \times N_T \times L \leq 160$ total taps. Meanwhile, the tests performed on the proposed simulator with the selected GPU device reveal that the most complex channel that can be simulated contains 118656 total taps ($24 \times 24 \times 206$). Moreover, the implementations presented in the open literature for achieving the reported performance consider that the paths of the PDP occur in multiples of the symbol period and all the subchannels share the same PDP. This contrasts with our GPU-based implementation, which can deal with paths with arbitrary positions and distinct PDPs for each subchannel. This is noteworthy because communication standards recommend PDPs with paths whose allocations are symbol period independent, while the flexibility of setting different PDPs for each subchannel makes it possible to simulate more complex scenarios. If the constrains assumed in the reported simulators were considered, then the proposed simulator could generate more total taps and consequently MIMO channels with larger numbers of antennas.

## 5. Conclusions

This paper presents a methodology for implementing a triply selective MIMO simulator based on GPUs. The proposed simulator considers the use of multiple SISO simulators, which are also implemented with GPUs, together with input and output correlation matrices in order to provide the corresponding space selectivity. This approach allows for great flexibility because each SISO subchannel can be set with different statistics and, therefore, more complex environments can be modeled. It is shown that under some considerations, the scheme fits with the more specific model proposed in the state of the art. Moreover, the implementations with GPUs considerably reduce the execution time compared with sequential implementations. Implementation results show a 688-fold gain for MIMO $7 \times 7$ when compared with sequential implementations. This suggests that this approach will enable more complex MIMO channels to be simulated with a greater number of antennas and a minimum penalty of time execution; in this way, communication systems can be tested in less time. Likewise, the considered model allows for the simulation of different scenarios, which are not forced to have the same statistics in each subchannel. Therefore, it becomes possible to simulate propagation environments that are more adequate for V2I and V2V communication systems.

## Appendix

## Implemented Codes

See Listings 1–4.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

## Acknowledgments

This work was financed by the Mexican Council for Science and Technology (CONACYT) through the SEP-CONACYT Basic Research Program (no. 241272).

## References

[1] A. Goldsmith, *Wireless Communications*, Cambridge University Press, New York, NY, USA, 2005.

[2] G. L. Stüber, *Principles of Mobile Communication*, Springer, New York, NY, USA, 3rd edition, 2012.

[3] M. Pätzold, *Mobile Radio Channels*, Wiley, Chichester, UK, 2nd edition, 2011.

[4] B. Talha and M. Pätzold, "Channel models for mobile-to-mobile cooperative communication systems: a state of the art review," *IEEE Vehicular Technology Magazine*, vol. 6, no. 2, pp. 33–43, 2011.

[5] J. Vázquez Castillo, L. Vela-Garcia, C. Gutiérrez, and R. Parra-Michel, "A reconfigurable hardware architecture for the simulation of Rayleigh fading channels under arbitrary scattering conditions," *AEÜ - International Journal of Electronics and Communications*, vol. 69, no. 1, pp. 1–13, 2015.

[6] L. Vela-Garcia, J. V. Castillo, R. Parra-Michel, and M. Pätzold, "An accurate hardware sum-of-cisoids fading channel simulator for isotropic and non-isotropic mobile radio environments," *Modelling and Simulation in Engineering*, vol. 2012, Article ID 542198, 12 pages, 2012.

[7] J. P. Kermoal, L. Schumacher, K. I. Pedersen, P. E. Mogensen, and F. Frederiksen, "A stochastic MIMO radio channel model with experimental validation," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 6, pp. 1211–1226, 2002.

[8] W. Weichselberger, M. Herdin, H. Özcelik, and E. Bonek, "A stochastic MIMO channel model with joint correlation of both link ends," *IEEE Transactions on Wireless Communications*, vol. 5, no. 1, pp. 90–99, 2006.

[9] R. Carrasco-Alvarez, J. Vázquez Castillo, A. Castillo Atoche, and J. Ortegón Aguilar, "A fading channel simulator implementation based on GPU computing techniques," *Mathematical Problems in Engineering*, vol. 2015, Article ID 237061, 8 pages, 2015.

[10] J. Tan, Z. Su, and Y. Long, "A full 3-D GPU-based beam-tracing method for complex indoor environments propagation modeling," *IEEE Transactions on Antennas and Propagation*, vol. 63, no. 6, pp. 2705–2718, 2015.

[11] Z. B. Loo, P. K. Chong, K. Y. Lee, and W.-S. Yap, "Improved path loss simulation incorporating three-dimensional terrain model using parallel coprocessors," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 5492691, 11 pages, 2017.

[12] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015.

[13] S. Roger, C. Ramiro, A. Gonzalez, V. Almenar, and A. M. Vidal, "Fully parallel GPU implementation of a fixed-complexity soft-output MIMO detector," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 8, pp. 3796–3800, 2012.

[14] C. Zhang, L. Liu, D. Marković, and V. Öwall, "A heterogeneous reconfigurable cell array for MIMO signal processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 3, pp. 733–742, 2015.

[15] Y. Karasawa, K. Nakada, G. Sun, and R. Kotani, "MIMO Fading Emulator Development with FPGA and Its Application to Performance Evaluation of Mobile Radio Systems," *International Journal of Antennas and Propagation*, vol. 2017, Article ID 4194921, 15 pages, 2017.

[16] F. Ren and Y. R. Zheng, "A novel emulator for discrete-time MIMO triply selective fading channels," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 9, pp. 2542–2551, 2010.

[17] A. Alimohammad and S. F. Fard, "A compact architecture for simulation of spatio-temporally correlated MIMO fading channels," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1280–1288, 2014.

[18] K. Yu and B. Ottersten, "Models for MIMO propagation channels: a review," *Wireless Communications and Mobile Computing*, vol. 2, no. 7, pp. 653–666, 2002.

[19] D. Demidov, *VexCL – a vector expression template library for OpenCL/CUDA*, 2017, https://github.com/ddemidov/vexcl.

[20] D. E. S. Research, *Random123*, 2017, https://www.deshawresearch.com/resources_random123.html.

[21] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw, "Parallel random numbers: As easy as 1, 2, 3," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 16:1–16:12, ACM, New York, NY, USA, 2011.

[22] C.-D. Iskander, "A MATLAB-based objectoriented approach to multipath fading channel simulation," Hi-Tek Multisystems, 2008.

[23] "TR 101 112 V3.1.0," Tech. Rep, 2008, http://www.etsi.org/deliver/etsi_tr/101100_101199/101112/03.01.00_60/tr_101112v030100p.pdf.

[24] "3GPP TR 36.817 V10.0.0," Tech. Rep, 2011, http://www.qtc.jp/3GPP/Specs/36817-a00.pdf.