Implementation of a Maximum Likelihood Doppler Spread Estimator on a Model-Based Design Platform

Adel Ati, Faouzi Bellili, Haithem Haggui, Abdelaziz Samet, and Sofiène Affes
INRS-EMT, 800, de la Gauchetière Ouest, Bureau 6900, Montreal, H5A 1K6, Canada.
Emails: {adel.el.ati, bellili, haggui, samet, affes}@emt.inrs.ca

***Abstract*—A new maximum likelihood (ML) Doppler spread estimator, recently shown to outperform most representative state-of-the-art solutions both in accuracy and complexity, is implemented on a FPGA-based platform. Rapid prototyping of the entire design is built as a high-level Simulink model using Xilinx System Generator IP blocks. The RF front-end of the design is implemented using the Nutaq's model-based design kit (MBDK). The ML Doppler spread estimator is assessed at a sampling rate of 80 Msps over a realistic RF channel generated by the EB Propsim FS8 channel emulator. Comparisons with the original floating-point MATLAB version suggest negligible performance losses, thereby validating and confirming the efficiency of the new real-time overt-the-air hardware design and implementation.**

I. Introduction

Wireless channel parameters estimation is a key factor for transceiver optimization in mobile communication systems. The wireless propagation environment is typically characterized by a multi-path time-varying channel whose fading rate is essentially related to the Doppler spread. The *a priori* knowledge/estimation of the Doppler spread is, therefore, a requirement for many applications such as adaptive channel estimation [1], power control, and handoff schemes [2], [3]; just to name a few. Many Doppler spread estimators have been proposed in the open literature (see [4], [5] and references therein), each having its own merits and drawbacks. However, it is the estimator that shows the most attractive trade-off between computational complexity and estimation accuracy that is usually deemed as the best candidate for hardware implementation in practice. Motivated by this fact, we have chosen to implement the very recent Doppler spread estimator derived in [6] from maximum likelihood (ML) theory and as such exhibits good performance over a wide range of practical Doppler values. Most interestingly, it does not involve any matrix inversion unlike all the existing ML-type estimators. On the top of offering one of the best accuracy/complexity trade-offs, it is also the first of its kind in terms of robustness to any mismatch in the channel mobility model, i.e., it does not require its *a priori* knowledge, which is a quite precious degree of freedom in practice.

Recently, software defined radios (SDRs) have been widely adopted for rapid prototyping of wireless communication applications. In fact the huge technological progress in the fields of reconfigurable devices and digital signal processing in general has opened up new opportunities in wireless communication systems design and implementation [7], [8]. Field Programmable Gate Array (FPGA) is considered as the most efficient device for SDR applications due to its performance, low-power consumption and re-configurability [9]. Yet, conventional FPGA-based design implementation approaches rely on low-level VHDL and Verilog programming, which suffer from poor flexibility and low productivity. Recently, a new approach of top-down design flow has been proposed in order to overcome this problem. This approach relies on a high-level Simulink model-based environment and exploits an existing block set of digital signal processing (DSP) cores. Hence, increasingly sophisticated hardware implementation designs can now be easily devised by a "drag-and-drop" procedure using these existing building blocks. The corresponding hardware descriptive langage (HDL) codes can be then automatically generated by powerful tools such as Xilinx System Generator. In this paper, we adopt this new model-based design approach to implement the selected Doppler spread estimator on a FPGA-Based platform. To the best of our knowledge, it is the very first hardware implementation of a Doppler spread estimator in the open literature. Furthermore, the estimator is showcased using a channel emulator which mimics real-world radio channels and, hence, allows testing in near over-the-air (OTA) real conditions. Comparisons with the original floating-point MATLAB version suggest negligible performance losses, thereby validating and confirming the efficiency of the new real-time overt-the-air hardware design and implementation.

This paper is organized as follows. In Section 2, the ML Doppler spread estimator of [6] is briefly summarized. In Section 3, the implementation and test workstation are described. In Section 4, the proposed hardware implementation of the estimator is detailed. In Section 5, we gauge its performance against its floating-point MATLAB counterpart. In Section 6, we draw out some concluding remarks.

II. Brief Summary of the selected ML Doppler Spread Estimator

In this section, we briefly introduce the ML Doppler spread estimator of [6] whose optimized hardware implementation is conceived and tested in the subsequent sections. Our motivations behind choosing this specific estimator are fuelled by its appealing complexity/accuracy tradeoff. Being derived from ML theory in [6], this estimator outperforms by far main state-of-the-art techniques in terms of estimation accuracy. More-

This work is supported by the NSERC CREATE Research Training Programme PERSWADE, the Engage program of NSERC, Nutaq, and SYTACom.

978-1-4673-6555-0/15/$31.00 ©2015 Crown

over, it is built upon a very simple frequency-domain two-ray approximation model for the actual channel covariance matrix. Therefore, the required inverse for the covariance matrix of the received signal is found analytically thereby avoiding its online computation. Most interestingly, the underlying approximation model is valid for various types of the channel's power spectral density (PSD) whose *a priori* knowledge becomes unnecessary for the estimator, that is in total contrast to all the existing techniques. This makes it even more suitable for practical implementation since the channel's PSD is usually unknown in practice and often changes from one environment to another. From [6], the local ML Doppler spread estimate, $\widehat{\sigma}_D(m)$, obtained over the m^{th} local approximation window, is obtained as follows:

$$\widehat{\sigma}_{D(m)} = \underset{\sigma_D}{\operatorname{argmax}}\ \mathcal{L}\left(\widehat{\mathbf{h}}(m), \sigma_D\right) \tag{1}$$

where $\mathcal{L}\left(\widehat{h}(m), \sigma_D\right)$ is the log-likelihood function (LLF) of $\widehat{h}(m)$ parametrized by σ_D. All the details related to the formulation of the estimator leading to the final expression of the LLF are given in [6]:

$$\begin{aligned}\mathcal{L}\left(\widehat{\mathbf{h}}(m), \sigma_D\right) &= \log([2+\widehat{\rho}\sigma_1][2+\widehat{\rho}\sigma_2]) \\ &\quad + \tfrac{1}{\widehat{\sigma}_n^2}\sum_{i=1}^{2}\sqrt{\tfrac{\widehat{\rho}\sigma_i}{2+\widehat{\rho}\sigma_i}}\left|\mathbf{u}_i^H\widehat{\mathbf{h}}(m)\right|^2\end{aligned} \tag{2}$$

where $\{.\}^H$, is the conjugate transpose operator. Moreover, $\widehat{\sigma}_n^2$ and $\widehat{\rho}$ are the ML estimates of the noise variance and the SNR, respectively, that are obtained using the estimator proposed in [10]. Using approximation windows of equal size, N, the expressions of the two eigen-values σ_1 and σ_2 are given as follows:

$$\sigma_1 = N + \frac{\sin(N\sigma_D T_s)}{\sin(\sigma_D T_s)} \quad \text{and} \quad \sigma_2 = N - \frac{\sin(N\sigma_D T_s)}{\sin(\sigma_D T_s)} \tag{3}$$

where T_s is the sampling period. Their associated eigen-vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ are given by the following expressions:

$$\left\{\begin{array}{lcl}\mathbf{u}_1 &=& \frac{\sqrt{2}}{\sqrt{\sigma_1}}e^{-j\frac{N-1}{2}\sigma_D T_s}\tilde{\mathbf{u}}_1 \\ \mathbf{u}_2 &=& j\frac{\sqrt{2}}{\sqrt{\sigma_2}}e^{-j\frac{N-1}{2}\sigma_D T_s}\tilde{\mathbf{u}}_2\end{array}\right. \tag{4}$$

where:

$$\tilde{\mathbf{u}}_1 = \begin{bmatrix}\cos(\frac{N-1}{2}\sigma_D T_s)\\ \cos(\frac{N-3}{2}\sigma_D T_s)\\ \vdots \\ \cos(\frac{N-(2k+1)}{2}\sigma_D T_s)\\ \vdots \\ \cos(\frac{N-(2k+1)}{2}\sigma_D T_s)\\ \vdots \\ \cos(\frac{N-3}{2}\sigma_D T_s)\\ \cos(\frac{N-1}{2}\sigma_D T_s)\end{bmatrix} \quad \tilde{\mathbf{u}}_2 = \begin{bmatrix}\sin(\frac{N-1}{2}\sigma_D T_s)\\ \sin(\frac{N-3}{2}\sigma_D T_s)\\ \vdots \\ \sin(\frac{N-(2k+1)}{2}\sigma_D T_s)\\ \vdots \\ -\sin(\frac{N-(2k+1)}{2}\sigma_D T_s)\\ \vdots \\ -\sin(\frac{N-3}{2}\sigma_D T_s)\\ -\sin(\frac{N-1}{2}\sigma_D T_s)\end{bmatrix}. \tag{5}$$

In order to enhance the estimation accuracy, we average the obtained ML Doppler spread estimates over M local approximation windows (of size N) as follows:

$$\bar{\widehat{\sigma}}_D = \tfrac{1}{M}\sum_{m=1}^{M}\widehat{\sigma}_D(m) \tag{6}$$

where MN is the size of the entire observation window. In the next section, we will detail the framework of the FPGA-based design adopted for the hardware implementation and testing of the aforementioned Doppler spread estimator.

III. Hardware Implementation Design and Setup

The main elements of our experimental setup are:

- the Nutaq's PicoSDR platform [11];
- the Nutaq's MBDK sofware [12]; and
- the EB Propsim FS8 channel emulator [13].

A. The Nutaq's PicoSDR platform:

The Nutaq PicoSDR platform is mainly composed of a Perseus card built around a Virtex-6 Xilinx FPGA and a FPGA mezzanine card (FMC) [14] to which are connected two RF transceiver cards. All applications targeting this platform can be implemented using the high-level Simulink environment which facilitates both the design and validation flows. The reconfigurability of the FMC card allows this platform to cover a wide range of wireless applications.

B. The Nutaq's MBDK sofware:

The Nutaq model-based design kit (MBDK) enables rapid prototyping of DSP algorithms, especially in our field of interest, (i.e., radio channel parameter estimation) [6]. The MBDK tool is an element of Nutaq's ADP software and works in the graphical environment of MATLAB-Simulink. It allows the design of different applications and their soft validation without using the VHDL or C languages. The FPGA implementation of well-developed and tested algorithms is ultimately achieved using the Xilinx System Generator tool.

C. The EB Propsim FS8 channel emulator:

The use of a channel emulator in our experimental setup has enabled us to emulate accurately several types of quite realistic radio channel models. We integrated in the RF chain the channel emulator "EB Propsim FS8" [13], a very compact equipment that has a user-friendly interface. With 8 RF channels, it allows generation of a bidirectional 4x4 MIMO channel for testing the wireless performance of mobile devices and chipsets and to perform studies over various radio access technologies such as 2G, 3G, and 4G long-term evolution (LTE).

IV. Hardware Design and Implementation

In this section, we describe the entire architecture and present the detailed of all blocks and their connections. Fig. 1 depicts a high-level block diagram for the designed hardware implementation of ML Doppler estimator. It shows a module for the control path and three modules that constitute the data path.

- The "Control" module represents the control path of our design which consists of three finite state machines (FSMs) used to control the data path modules and to generate their required command signals. Fig. 2 shows the state diagrams of the FSMs which describe their sequential logic. A Moore's machine concept is used to implement the three FSMs. It is robust and ensures a correct operation of the design since the outputs change

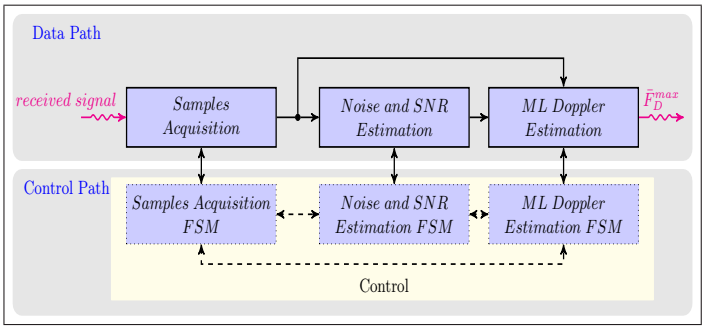


Fig. 1: Block diagram of the ML Doppler spread estimator.

at clock edge. This is in contrast with the Mealy machines where an input change can cause output change as soon as logic is done thereby generating an asynchronous feedback problem when two or more FSMs are connected as in our case. The outputs described in Fig. 2 represent the command signals for different elementary blocks used to build the data path modules such as counters enable signals, memories and accumulators resets, arithmetic operations enable signals, etc. The three FSMs are implemented using the MCode block in the Xilinx blockset library which allows the use of MATLAB functions to build a hardware block instead of coding in VHDL or Verilog.

- The "Samples Acquisition" module stores N samples of the received signal in shared memories. The sampling period, T_s, is specified by the end user and can be modified depending on the underlying system's signalling standard. These samples are used in different stages of our design and, therefore, they are copied in different shared memories to enable their simultaneous use and processing.
- The "Noise and SNR Estimation" module is used to estimate the SNR and the noise power which are necessary to the evaluation of the LLF in (2). To estimate these two parameters, we used the method described in [10].
- The "ML Doppler Estimation" module represents the core of our work. It allows the estimation of the Doppler spread using the algorithm described in Section 2 and we are now ready to highlight its main features.

In fact, the m^{th} local ML estimate, $\widehat{\sigma}_D(m)$, of the Doppler spread is obtained numerically by finding the maximum of the LLF given in (2). The evaluation of the local LLF at each point, σ_D, of the line grid requires computation of the following variables in the given order: σ_1 and σ_2, vectors $\mathbf{u}_1$ and $\mathbf{u}_2$, and finally $\mathcal{L}(\widehat{\mathbf{h}}(m), \sigma_D)$.

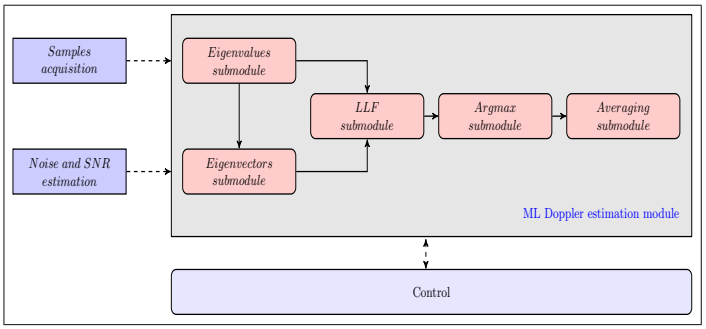


Fig. 3: Block diagram of the "ML Doppler Estimation" module.

Fig. 3 shows that the "ML Doppler Estimation" module is composed of the following five submodules:

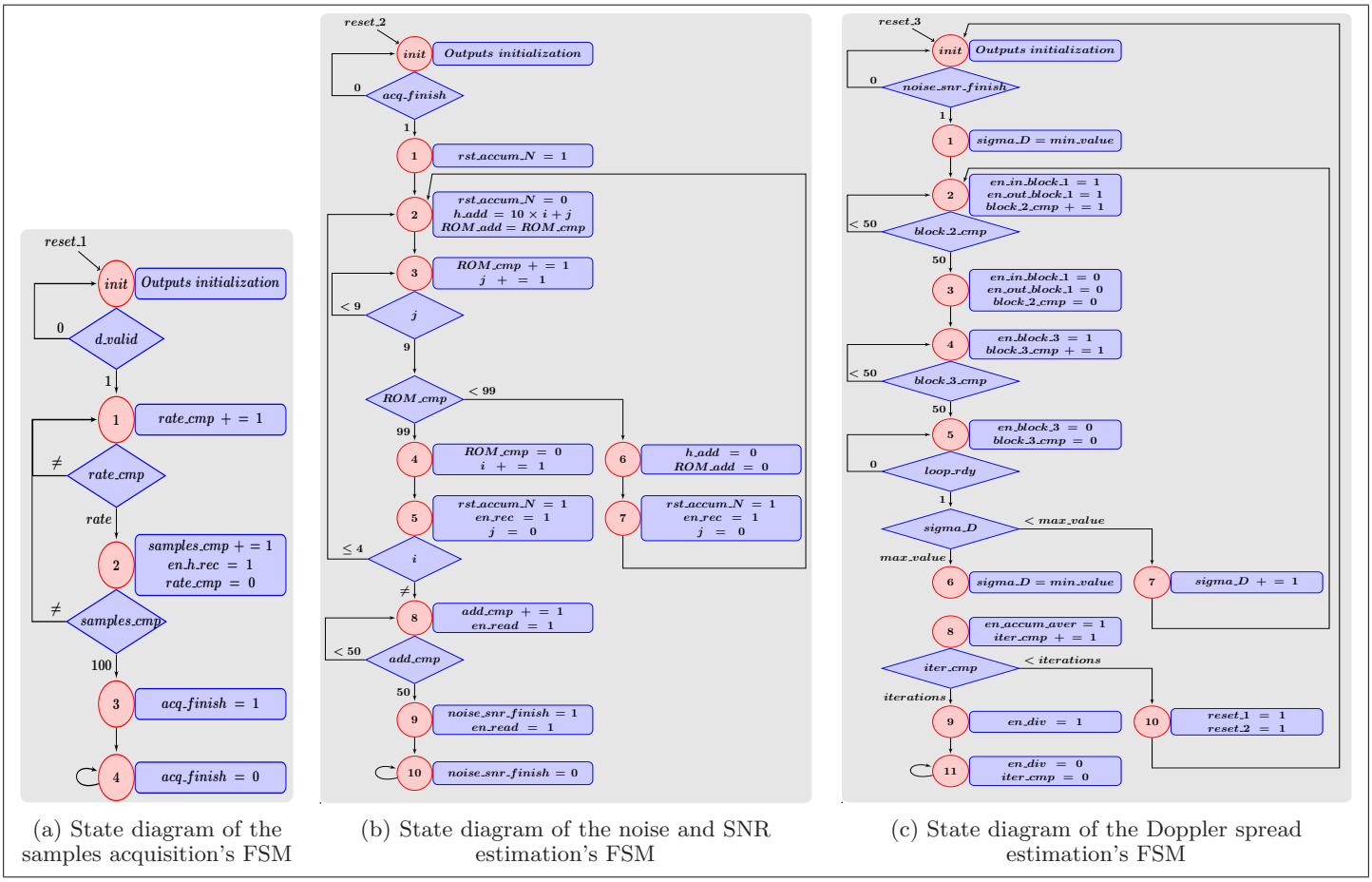
- EIGENVALUES submodule.
- EIGENVECTORS submodule.
- LLF submodule.
- ARGMAX submodule.
- AVERAGING submodule.

As suggested by its name, the EIGENVALUES submodule computes σ_1 and σ_2 using the two expressions given in (3). It takes N, T_s and σ_D as inputs and outputs σ_1 and σ_2. Once a valid outputs is obtained, a ready (rdy) signal activates the EIGENVECTORS submodule. The latter, takes as inputs N, T_s, σ_1 and σ_2 and computes the two vectors $\mathbf{u_1}^H$ and $\mathbf{u_2}^H$ using (4) and (5) and then records their real and imaginary parts into memory. Once computation is finished, another rdy signal activates the LLF submodule which takes as inputs N, T_s,$\widehat{\rho}$, σ_n^2, σ_1,σ_2, $\mathbf{u}_1^H$,$\mathbf{u}_2^H$ and $\widehat{\mathbf{h}}$ and evaluates the LLF in (2) at all the candidate values σ_D of the unknown Doppler spread. The LLF submodule outputs two signals which are L_valid and L. The latter contains the value of the LLF in the underlying value for σ_D and the former instructs calculation over the next block, i.e., the ARGMAX submodule, which takes as inputs L and σ_D. After being activated, this submodule computes the argmax of the LLF by comparing the current calculated L to the registered one and keeping the largest of the two with the corresponding σ_D. This block brings out two outputs which are Fd_max and rdy. The latter allows the system to loop over a new σ_D value. Finally, the AVERAGING submodule computes the average of the different Doppler spread estimates obtained from the multiple local approximation windows in order to reduce the estimation bias. The number of local approximation windows is specified by the user offline. Instead of spending much time coding testbenchs in VHDL or Verilog langages in order to test the different blocks of the design, Simulink signals can be used as stimulus which makes the test and validation tasks easier for the designer.

V. CO-SIMULATION AND RESULTS

To measure the Doppler spread estimate provided by our estimator, we also devised and developed a host test model in Simulink. Custom registers were used to define the following test configuration parameters:

- *Rate*: defines the rate used in the sample acquisition process and which is related to both the system clock and the sample period defined by the LTE standard.
- *Sigma_d_step*: represents the discretization step of the line grid that we used to find the suitable σ_D which maximizes the cost function.
- *Reset*: is the signal in charge of the initialization of the whole design. It is used in the test to execute a Monte-Carlo loop.
- T_s: represents the sampling period.
- *Iteration*: represents the number of local approximation windows.
- *Min_val_sigma*, and *Max_val_sigma*: define the interval in which our grid search is carried out.



(a) State diagram of the samples acquisition's FSM

(b) State diagram of the noise and SNR estimation's FSM

(c) State diagram of the Doppler spread estimation's FSM

Fig. 2: State diagrams of the FSMs.

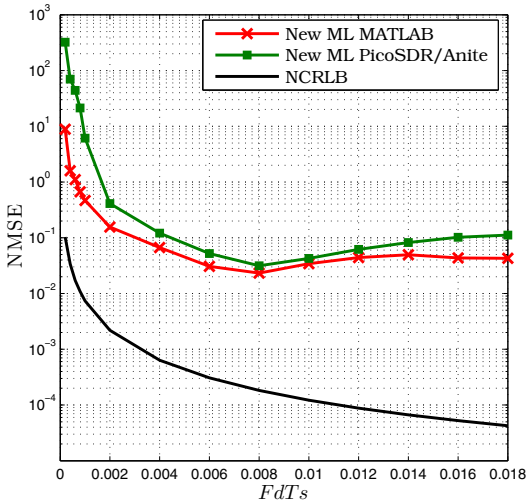


Fig. 4: NMSE vs. F_dT_s with $T_s = 10\ \mu s$, $SNR = 0\ dB$, and $N = 1000$.

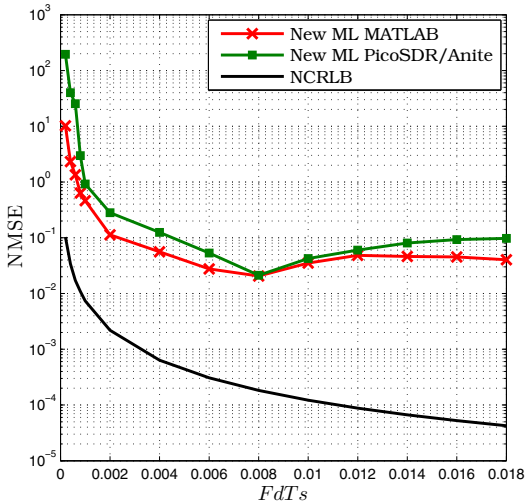


Fig. 5: NMSE vs. F_dT_s with $T_s = 10\ \mu s$, an $SNR = 5\ dB$, and $N = 1000$.

The Doppler estimate is given by the *Fd_max_aver* custom register. In order to asses the practical performance of the estimator, we compute the normalized mean square error (NMSE) from 10000 Monte-Carlo runs. In Figs. 4 and 5, we compare the FPGA-Based NMSE to its counterpart obtained via MATLAB simulations in terms of the maximum Doppler Frequency which is related to the Doppler spread for a uniform Jakes' model by $\sigma_D = 2\pi F_d/\sqrt{2}$ [6] for SNR levels of 0 dB and 5 dB, respectively.

As seen from both figures, the two curves are slightly different far from each other and basically follow the same trend. This confirms the validity of the proposed design for the hardware implementation of the underlying Doppler spread estimator. The small discrepancies between the two curves can be explained by the fact that the MATLAB version does not take into consideration high-frequency effects caused by high-pass filters, power amplifiers, and low-noise amplifiers. Furthermore, implementation causes errors due to limitations

of its binary representation in fixed point against floating point MATLAB. It is indeed difficult to predict the inputs and outputs range in order to choose suitable binary representation that minimizes these implementation errors. In Figs. 6 and 7, we compare the FPGA-Based NMSE to its counterpart obtained via MATLAB simulations in terms of SNR for F_dT_s values of 0.006 and 0.01, respectively. We verify again that the MATLAB and PicoSDR/Anite curves are relatively close to each other and follow the same trend. At SNR values above about -5 dB, we start to see a slight gap that can reach a maximum value of 10^{-2} due to the same reasons explained previously.

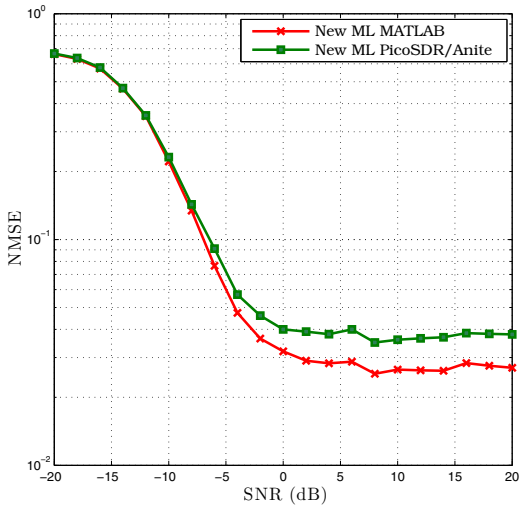


Fig. 6: NMSE vs. SNR with $F_dT_s = 0.006$ and $N = 1000$.

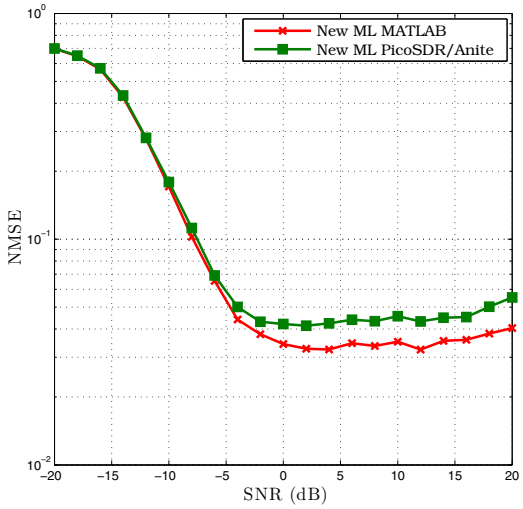


Fig. 7: NMSE vs. SNR with $F_dT_s = 0.01$ and $N = 1000$.

VI. Conclusion

In this paper, we presented a model-based approach for ML Doppler spread estimation in a FPGA-based platform. Simulations and implementation were performed using MATLAB-Simulink library in conjunction with the Xilinx system generator toolbox and the Nutaq's MBDK. The proposed hardware implementation design was validated by computer-based simulations and confirmed that the underlying Doppler estimator has indeed a very low computational load. Finally, this contribution proves the advantage of using the model-based top-down design approach as an alternative to the traditional techniques that require the time-consuming low-level HDL coding.

References

[1] S. Affes and P. Mermelstein, "A new receiver structure for asynchronous CDMA: STAR-the spatio-temporal array-receiver", *IEEE Jour. Selected Areas in Commun.,* vol. 16, no. 8, pp. 1411-1422, Oct. 1998.

[2] C. Tepedelenlioglu, A. Abdi, G. B. Giannakis, and M. Kaveh, "Estimation of Doppler spread and signal strength in mobile communications with applications to handoff and adaptive transmission," *Wireless Communications and Mobile Computing,* vol.1, pp. 221-242, 2001.

[3] G. L. Stüber, *Principles of Mobile Communication,* 2nd ed., Kluwer, 2001.

[4] Y.-R. Tsai and K.-J. Yang, "Approximate ML Doppler spread estimation over flat Rayleigh fading channels," *IEEE Signal. Process. Lett.,* vol. 16, no. 11, pp. 1007-1010, Nov. 2009.

[5] M. Souden , S. Affes , J. Benesty, and R. Bahroun, "Robust Doppler spread estimation in the presence of a residual carrier frequency offset," *IEEE Trans. Signal Process.,* vol. 57, no. 10, pp. 4148- 4153, Oct. 2009.

[6] F. Bellili and S. Affes, "A low-cost and robust maximum likelihood Doppler spread estimator", *Globecom-Wireless Communications Symposium*, Atlanta, 2013, pp. 4430-4435. USA: IEEE.

[7] V. B. Alluri, J. R. Heath, and M. Lhamon, "A new multichannnel , coherent amplitude modulated, gate array technology implementation", *IEEE Transaction on Signal Processing,* vol. 58, pp. 5369-5384, Oct. 2010.

[8] J. Kim, H. Seungheon, and S. Choi; , "Implementation of an SDR system using graphics processing unit", *IEEE Communications Magazine,* vol.48, no.3, pp.156-162, Mar. 2010.

[9] U. Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Array", Springer, 2001.

[10] F. Bellili, R. Meftehi, S. Affes, and A. Stephenne, "Maximum Likelihood SNR Estimation of Linearly-Modulated Signals over Time-Varying Flat-Fading SIMO Channels", *IEEE Transaction on Signal Processing,* vol. 63, pp. 441-456, Oct. 2014

[11] PicoSDR-User's Guide-v1.0. Jun. 2013.

[12] Nutaq MicroTCA BSDK API-Version 1.0. Jun. 2013.

[13] EB Propsim Radio Channel Emulator-Quick Guide-Release 3. Document version 1.5

[14] *Nutaq Radio420X: Multimode SDR FMC RF transceiver*, Nutaq, 2014.