Received April 1, 2019, accepted May 10, 2019, date of publication May 30, 2019, date of current version June 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919978

FPGA-SDR Integration and Experimental Validation of a Joint DA ML SNR and Doppler Spread Estimator for 5G Cognitive Transceivers

HAITHEM HAGGUI[1], **SOFIÈNE AFFES**[1], **(Senior Member, IEEE), AND FAOUZI BELLILI**[2]
[1]INRS-EMT, Montreal, QC H5A 1K6, Canada
[2]Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB R3T 5V6, Canada

Corresponding author: Haithem Haggui (haggui@emt.inrs.ca)

This work was supported in part by the Discovery Grants and the CREATE PERSWADE <www.create-perswade.ca> Programs of the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the Discovery Accelerator Supplement Award from NSERC.

ABSTRACT In a multi-connected, multi-technology, and pervasive mobile infrastructure, such as what is being planned for 5G, artificial intelligence and cognition will play a major role. An important goal of future mobile infrastructures is to self-adapt their characteristics to their operating conditions, at the physical link, as well as at the network and application layers, which gives rise to a new paradigm known as context-aware cognitive radio (CR). CR transceivers (CTRs) mostly incorporate a cognitive engine that relies on various sensorial entities, which attempt to provide sufficient information about the quality of the link through the estimation of various key channel parameters. Two important parameters are required in a wide range of CTR architectures: the signal-to-noise ratio (SNR) and the Doppler spread. Within this context, we tackle the hardware design and integration of a joint data-aided (DA) maximum likelihood (ML) SNR and Doppler spread estimator recently shown to outperform main state-of-the-art solutions both in terms of accuracy and complexity. We propose a deep-pipelined and resource-efficient architecture for the outlined joint DA ML estimator, and we integrate our design on an FPGA-based software-defined radio (SDR) platform. We finally validate and test this prototype in real time under realistic over-the-air propagation conditions reproduced by a highly-scalabile channel emulator. Compared to its MATLAB floating-point version, our hardware prototype suggests negligible losses in performance despite the existence of several hardware impairments, thereby confirming its very strong potential and attractiveness for possible integration in future 5G CTRs.

INDEX TERMS Context awareness, cognitive radio (CR), software-defined radio (SDR), SNR, Doppler spread, maximum likelihood (ML), data-aided (DA), parameter estimation, FPGA.

ABBREVIATIONS AND ACRONYMS

ADC	Analog to Digital Converter
BPS	BEEcube Platform Studio
CPU	Central Processing Unit
CR	Cognitive Radio
CTF	Coarse-To-Fine
CTR	CR Transceivers
DA	Data-Aided
DAC	Digital to Analog Converter
FMC	FPGA Mezzanine Cards
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GPP	General Purpose Processor
HDL	Hardware Description Language
HW/SW	Hardware/Software
LUT	Look-Up Table
MAC	Multiply-And-Accumulate
ML	Maximum Likelihood
NDA	Non-Data-Aided
NMSE	Normalized Mean Square Error
OTA	Over-The-Air

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Fadda.

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

RTL	Register-Transfer-Level
SDR	Software Defined Radio
VHSIC	Very High Speed Integrated Circuit

I. INTRODUCTION

The continuing growth in demand for ubiquitous wireless ultra-broadband communications has pushed service providers and mobile industries to deploy new data processing techniques and network infrastructures in order to enable highly-efficient, secure, ultra-reliable, and ultra-low latency services to everyone and everything [1]–[3]. "Zero latency", gigabit, and fully immersive experiences will be the main drivers for the uptake of new technology components on future 5G mobile networks [3], [4]. Several challenges, however, need to be tackled to meet not only the expected performance in terms of throughput, energy efficiency, service-level latency, battery lifetime, quality of service (QoS), manageability, etc., but also any future requirements.

This calls, indeed, for the integration of highly flexible, scalable, and modular infrastructure, as well as the setting-up of some "intelligence" capabilities in the network [5]. Therefore, cognitive architectures and networking should be an inherent characteristic of future 5G mobile communication systems. This feature will provide automatic and dynamic adaptation policies to the different transmission areas owing to the integration of context-awareness capabilities across all communication levels, from the physical and networking to transport and application layers. In fact, context awareness is considered as a response process to some context information such as activities, network states, user speeds, battery levels, energy consumption, and wireless channel states, etc., obtained from the involved concrete or virtual "sensorial" entities.

Recently, a new context-aware cognitive transceiver (CTR) architecture has been devised in [6] by the Wireless Lab <www.wirelesslab>. The new CTR is able to change its internal configuration automatically by making the best selection of the combination triplet among three different pilot-utilization modes, two different channel identification schemes, and two data detection modes. At low SNRs and high user mobility, the software integration of the new CTR was shown to provide, respectively, up to 700% and 40% of peak link-level and system-level throughput gains, compared to conventional transceiver architectures [6].

The cognitive engine of our CTR requires the *a priori* knowledge of two key channel parameters, which represent its basic sensorial entity: the Doppler spread and the SNR. Therefore, the knowledge/estimation of these two parameters is required to set up the CTR's context awareness, since the Doppler and the SNR reveal, respectively, the channel's time variation rate and the operational conditions that both dictate countless functional mode selectors and criteria such as the adaptation rate or the channel quality information (CQI) [7], [8] to name a few. Motivated by this fact, we set as a main goal of this work the hardware design, implementation, and real-time over-the-air validation of a link-layer CTR sensorial functionality using a robust hardware prototyping environment.

Typically, a suitable parameter estimator can be selected from a plethora of state-of-the-art techniques. For instance, a number of Doppler spread estimation schemes have been reported in the open literature. But most of them are either level-crossing-rate (LCR) [9], [10], power spectrum density (PSD) [11], or covariance-based methods [12]–[14]. These three types of estimators do not extract the Doppler information from the received signal itself, but rather from its statistics. Therefore, they require a large number of received samples to meet the target estimation performance. Moreover, most of them involve some complex mathematical operations such as matrix inversion or multiplication and, therefore, their computational complexity is deemed too high for practical implementation. However, in [15], a new robust and low-complexity maximum likelihood (ML) Doppler spread estimator based on a frequency-domain two-ray approximation of the channel covariance matrix has been proposed. It extracts the Doppler information directly from the received signal itself, regardless of the existing Doppler spectrum, thereby requiring a small number of received samples. Moreover, unlike the previously discussed techniques, it involves no matrix inversion or multiplication. Motivated by these facts, we have chosen to implement this estimator since it not only entails a low computational burden, but also unambiguously outperforms state-of-the-art techniques in terms of estimation accuracy, more so at extremely small normalized Doppler frequencies.[1]

In line with the requirements of the CTR itself, the selected Doppler estimator of [15] also requires the knowledge of the instantaneous SNR and the noise variance, which are also unknown in practice. For this purpose, among many others, we have also recently developed a new low-cost joint ML SNR, channel, and noise variance estimator and data demodulator over time-varying flat-fading channels [16]. Depending on the pilot-use mode, this estimator is referred to as data-aided (DA), non-data-aided (NDA), or hybrid. The DA version, in particular, relies on the sole use of some known pilot symbols and as such is derived in closed form. Owing to its simplicity and the huge performance gains it brings with respect to state-of-art-the techniques, the joint estimator/demodulator of [16] was also integrated in the CTR. Its DA version, in particular, will be used to identify the operating instantaneous SNR required by the CTR's cognitive engine in order to dynamically select the best combination triplet of pilot-use, channel identification, and data-detection mode.

Several works have adopted the key perspective of pushing promising algorithms from a simple software status to a

[1]In future 5G communication systems, the normalized Doppler frequency, $f_D T_s$ is indeed expected to be extremely small since the latter will operate at a very small symbol period, T_s, in order to provide high-data-rate communications.

fully-functional hardware prototype [17]–[19]. More than proving the concept, this perspective significantly reduces time-to-market for new outstanding application-oriented techniques. Indeed, the challenging process of hardware development and integration allows the DSP designers to face the practical realization and standardization constraints from the earliest conception stage of the product. To that end, this multidisiplinary process requires the cooperation of software, hardware, radio-frequency (RF) and printed-circuit-board (PCB) designers in an application-oriented approach. Software defined radio (SDR) is an adapted integration host-platform which is gaining popularity among our peers [20]–[23] due to its high reconfigurability and flexibility. In fact, to enable rapid and cost-effective development of modern wireless communication systems, SDR combines the scalability and the excellent computation capabilities of FPGAs with a user-friendly software design flow. Some works [22], [23] have put forward the SDR capabilities of prototyping communication systems, however, illustrating them with any practical examples. Other works [24], [20] have implemented entire communication systems over SDR thereby showcasing the great potential of these prototyping platforms. However, partitioning of the SDR resources adopted there is very often suboptimal [24] since the main base-band computation burden is processed in the host computer rather than in the resources-abundant FPGA. In general, the SDR's embedded general purpose processor (GPP) implements communication protocols, user applications, and supports powerful real-time test and debugging environments. Whereas, the reconfigurable logic of the SDR's FPGA is more adapted for a parallel execution of various baseband processing blocks including modulation/demodulation, filtering, channel coding/decoding, etc. These tasks were a throughput bottleneck in the earliest SDR platforms which came without FPGA accelerators such as [25]. In [21], a powerful multi-standard FPGA-hosted baseband OFDM transceiver architecture has been presented to explore the outcome of the FPGA dynamic partial reconfiguration [26] in terms of resource utilization and reconfiguration latency in the context of a CTR implementation. However, its cognitive engine used to instruct on a runtime the base-band processing operates based on a pure spectrum efficiency perspective without considering the wireless channel conditions. Moreover, the FPGA dynamic partial reconfiguration, despite promising, requires high-level of FPGA expertise that can slow down the prototyping process. A more fluid prototyping method in [20], adopting a model-based design-flow [27], has implemented a generic communication system with an efficient partitioning of SDR resources. However, its performance has been tested over a nearly-perfect wireless channel rather than in realistic over-the-air transmission scenarios. Besides, utilization of FPGA resources and energy consumption were not investigated there to seek possible enhancements of the generic implementation aspect.

In this paper, we propose a new model-based FPGA design and the SDR integration of the sensorial entity of the new CTR which is composed of the selected ML Doppler spread and SNR estimators. More specifically, we put forward an efficient hardware architecture that ensures an optimized performance/resource usage trade-off to produce a modular, versatile, and reusable hardware prototype. Then, we take advantage of the high reconfigurability of the SDR host platform to integrate, test, and validate our FPGA design in real time. Finally, we showcase this core in realistic over-the-air (OTA) operating conditions using a channel emulator which mimics real-world radio channels. Compared to the original reference MATLAB version, the new hardware core, operating in real time and over-the-air, suggests negligible performance losses, thereby validating and confirming the efficiency of our implementation and its robustness to all hardware imperfections.

We organize the remainder of this paper as follows. In Section II, we introduce the system model and the mathematical formulation of the Doppler spread and SNR estimators. In Section III, we present the hardware setup, the integration platform, and the most important hardware design trade-offs. We will dedicate Section IV to the discussion of the proposed design's architecture, while we assess in Section V its performance and gauge it, in real-time and near OTA conditions, against its MATLAB-based floating-point software version. Finally, we draw out some concluding remarks in Section VI.

We define beforehand the adopted mathematical notations. Vectors and matrices are represented in lower- and upper-case bold fonts, respectively. The Euclidean norm of a vector is denoted as $\|.\|$, and the operators $\{.\}^*$ and $|.|$ return the complex conjugate and amplitude of any complex number, respectively. $\{.\}^H$ represents the Hermitian conjugate operator. Whereas, the operators $L\{.\}$ and $ln\{.\}$ denote the log-likelihood function and the natural logarithm, respectively. For the sake of clarity, the mathematical notations for the most important variables and parameters adopted in this work are listed in Table 1.

II. SYSTEM MODEL AND MATHEMATICAL FORMULATION

In this section, we briefly review the mathematical formulation of the selected DA ML Doppler spread and SNR estimators introduced in [15] and [16], respectively.

We consider a continuous transmission of symbols over a flat-fading[2] Rayleigh channel, $h(t)$, immersed in an additive white Gaussian noise, $w(t)$. Assuming an ideal receiver with perfect time and frequency synchronization, the sampled

[2]The narrowband model in (1) is well justified in practice by its wide adoption in current and next-generation *multicarrier* communication systems, such as long-term-evolution (LTE), LTE-Advanced (LTE-A) and Beyond (LTE-B) systems. In fact, it is well known that OFDM systems transform a multipath frequency-selective channel in the time domain into a frequency-flat (i.e., narrowband) channel over each subcarrier as modeled by (1). Actually, multicarrier technologies were primarily designed to combat the multipath effects in high-data-rate communications by bringing back the per-carrier propagation channel to the simple flat-fading case.

TABLE 1. List of mathematical notations for most important variables and parameters.

Notation	Variable/Parameter
n	time index
T_s	sampling period
x	transmitted signal
h	flat-fading Rayleigh channel
w	additive white Gaussian noise
y	sampled baseband received signal
m	local approximation window index
N	size of local approximation windows
M	total number of local approximation windows
L	channel polynomial order
σ_D	Doppler spread
λ_1, λ_2	eigenvalues of approximated channel covariance matrix
$\widehat{\rho}$	estimate of instantaneous SNR
$\widehat{\sigma}_n^2$	estimate of noise variance
$\widehat{\sigma}_D$	ML estimate of Doppler spread
$\widehat{\mathbf{h}}$	channel estimate
$\widehat{\mathbf{c}}$	channel approximating-polynomial coefficients

baseband received signal can be expressed as:

$$y(nT_s) = h(nT_s)x(nT_s) + w(nT_s), \quad n = 0, 1, 2\ldots \quad (1)$$

where n and T_s denote, respectively, the time index and the sampling period. The equivalent discrete-time observation data sequence is:

$$y[n] = h[n]x[n] + w[n], \quad n = 0, 1, 2\ldots \quad (2)$$

The ML Doppler spread and SNR estimators were both developed for a fully-DA scheme. In other words, only the received samples corresponding to pilot positions are used during the estimation process. Without loss of generality, we assume that the known pilot sequence that is periodically transmitted is an all-ones sequence. Then, the observed baseband signal at pilot positions can be expressed as:

$$y[n] = h[n] + w[n], \quad n = 0, 1, 2\ldots \quad (3)$$

In order to apply locally a Taylor series expansion advocated in [16] (clarifications will follow shortly), the entire observation window is further split into multiple local approximation windows of size N. The main advantage of this approach is its ability to locally capture the unpredictable time variations of the channel using very few approximating-polynomial coefficients. Fig. 1 better illustrates the received data layout. We denote by M the total number of the local approximation windows. Hence, the m^{th} local observation sequence at pilot positions is given by:

$$y^{(m)}[n] = h^{(m)}[n] + w^{(m)}[n], \quad n = 0, 1, \ldots, N-1. \quad (4)$$

A. DA ML DOPPLER SPREAD ESTIMATOR

By considering the system model described in (4), it is worth mentioning that the information about the Doppler spread is hidden in the channel's autocorrelation coefficients. As mentioned previously, the ML Doppler spread estimator derived in [15] is built upon the following very simple second-order Taylor series approximation for the covariance matrix of the channel:

$$\mathbf{R_h}(\sigma_D) = \frac{\sigma_h^2}{2}\mathbf{A}(\sigma_D)\mathbf{A}^H(\sigma_D), \quad (5)$$

where σ_D denotes the Doppler spread, and $\mathbf{A}(\omega)$ is a $(N \times 2)$ matrix explicitly given by:

$$\mathbf{A}(\omega) = \left[\mathbf{a}(-\omega) \quad \mathbf{a}(\omega)\right], \quad (6)$$

in which the vector $\mathbf{a}(\omega)$ is defined as:

$$\mathbf{a}(\omega) \triangleq \left[1 \quad e^{j\omega T_s} \quad e^{j2\omega T_s} \quad \cdots \quad e^{j(N-1)\omega T_s}\right]^T. \quad (7)$$

Building upon the approximation in (5), and resorting to tedious algebraic manipulations (for more details, cf. [15]), the log-likelihood function (LLF) of the m^{th} local observation sequence $\mathbf{y}^{(m)} \triangleq \left[y^{(m)}[1], y^{(m)}[2], \ldots, y^{(m)}[N]\right]^T$, parametrized by σ_D is obtained as:

$$L^{(m)}(\sigma_D) = -\text{Ln}(\psi(\sigma_D)) + \frac{1}{\widehat{\sigma}_n^{2(m)}}\sum_{i=1}^{2}\gamma_i(\sigma_D)^2\left|\mathbf{u}_i(\sigma_D)^H\mathbf{y}^{(m)}\right|^2, \quad (8)$$

where:

$$\psi(\sigma_D) = \left[2 + \widehat{\rho}^{(m)}\lambda_1(\sigma_D)\right]\left[2 + \widehat{\rho}^{(m)}\lambda_2(\sigma_D)\right], \quad (9)$$

$$\gamma_i(\sigma_D) = \sqrt{\frac{\widehat{\rho}^{(m)}\lambda_i(\sigma_D)}{2 + \widehat{\rho}^{(m)}\lambda_i(\sigma_D)}}, \quad i = 1, 2. \quad (10)$$

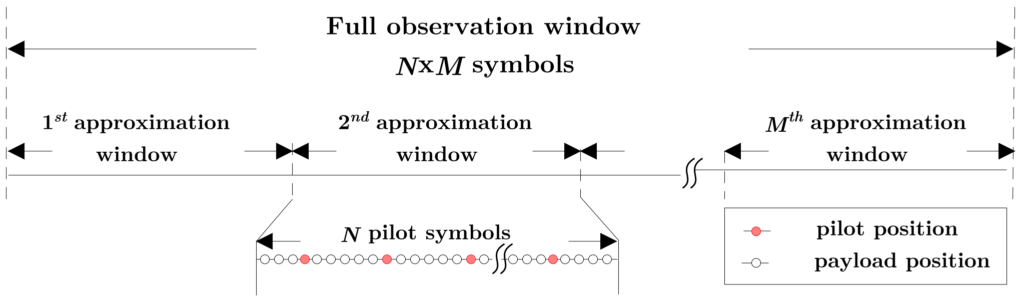

FIGURE 1. Payload and pilot symbol layout with M local approximation windows.

Here, $\widehat{\rho}^{(m)}$ and $\widehat{\sigma}_n^{2(m)}$ are, respectively, the values of the estimated instantaneous SNR and noise variance over the m^{th} local approximation window. Moreover, λ_1 and λ_2 are the two eigenvalues of the approximated covariance matrix in (5), which are expressed as follows [15]:

$$\lambda_1(\sigma_D) = N + \left|\frac{\sin(N\sigma_D T_s)}{\sin(\sigma_D T_s)}\right|, \tag{11}$$

$$\lambda_2(\sigma_D) = N - \left|\frac{\sin(N\sigma_D T_s)}{\sin(\sigma_D T_s)}\right|. \tag{12}$$

Hence, their associated eigenvectors, $\mathbf{u}_1$ and $\mathbf{u}_2$, are given as:

$$\mathbf{u}_1(\sigma_D) = \frac{1}{\sqrt{2\lambda_1(\sigma_D)}}\left(\mathbf{a}(-\sigma_D) + \frac{\varphi(2\sigma_D T_s)^*}{|\varphi(2\sigma_D T_s)|}\mathbf{a}(\sigma_D)\right), \tag{13}$$

$$\mathbf{u}_2(\sigma_D) = \frac{1}{\sqrt{2\lambda_2(\sigma_D)}}\left(\mathbf{a}(-\sigma_D) - \frac{\varphi(2\sigma_D T_s)^*}{|\varphi(2\sigma_D T_s)|}\mathbf{a}(\sigma_D)\right), \tag{14}$$

in which φ is defined as follows:

$$\varphi(x) = \frac{\sin\left(\frac{Nx}{2}\right)}{\sin\left(\frac{x}{2}\right)} \exp\left(j\frac{(N-1)}{2}x\right). \tag{15}$$

Finally, the ML estimate of the Doppler spread is obtained as follows:

$$\widehat{\sigma}_D^{(m)} = \underset{\sigma_D}{\operatorname{argmax}} \left\{L^{(m)}(\sigma_D)\right\}. \tag{16}$$

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

$$\widehat{\sigma}_D = \frac{1}{M}\sum_{m=1}^{M} \widehat{\sigma}_D^{(m)}. \tag{17}$$

B. DA ML SNR ESTIMATOR

The main attractive feature of the DA ML SNR estimator of [16] is that instead of evaluating the LLF expression over a two-dimensional (2D) grid of candidate values for $\boldsymbol{\theta} = [\rho, \sigma_n^2]$, it relies on an analytical closed-form expression of their ML estimates. First of all, the DA ML SNR estimation algorithm finds the optimal coefficients for the polynomial that best approximates the time-varying channel over the m^{th} local window, as follows:

$$\widehat{\mathbf{c}}^{(m)} = \left(\mathbf{B}^H\mathbf{B}\right)^{-1}\mathbf{B}^H\mathbf{y}^{(m)}. \tag{18}$$

where $\mathbf{B} = \mathbf{AT}$ is a $(N \times L)$ matrix. L denotes the order of the approximation polynomial, $\mathbf{A} = \operatorname{diag}\{x(T_s), x(2T_s), \ldots, x(NT_s)\}$, and $\mathbf{T}$ is a known Vandermonde matrix whose entries correspond to the sampling time instants, $\{0, T_s, 2T_s, \ldots, (N-1)T_s\}$.

In a second step, the DA ML algorithm exploits the estimated channel polynomial coefficients in order to find the noise variance as:

$$\widehat{\sigma}_n^{2(m)} = \frac{1}{2N}\left\|\mathbf{y}^{(m)} - \mathbf{B}\widehat{\mathbf{c}}^{(m)}\right\|^2. \tag{19}$$

Finally, the m^{th} local instantaneous SNR estimate is given by:

$$\widehat{\rho}^{(m)} = \frac{\|\widehat{\mathbf{h}}^{(m)}\|^2}{2N\widehat{\sigma}_n^{2(m)}}. \tag{20}$$

The channel estimate, $\widehat{\mathbf{h}}^{(m)}$, is obtained from the estimated channel coefficients over the m^{th} local approximation window, is obtained as follows:

$$\widehat{\mathbf{h}}^{(m)} = \mathbf{T}\,\widehat{\mathbf{c}}^{(m)}. \tag{21}$$

III. HARDWARE SETUP

In order to ensure an effective prototyping process and a fluid integration in a near real-world conditions, our hardware setup will be composed of:

- A BEEcube miniBEE4 SDR platform;
- A BEEcube Platform Studio (BPS) environment;
- An EB Propsim FS8 channel emulator;

A. THE BEECUBE MINIBEE4 SDR PLATFORM

The main part of the outlined hardware setup consists in a BEEcube miniBEE4 SDR platform [28] which is a highly efficient prototyping platform built around a large capacity Virtex-6 Xilinx FPGA and an Intel Core i7 central processing unit (CPU). This complete "R&D in the box" platform is equipped with two integrated FPGA mezzanine cards (FMC111) which are connected to the RF front-end, where the local oscillator frequency, attenuators, and filters are user-defined. Our main motivations for choosing this particular SDR are as follows. First, the miniBEE4 SDR provides the flexibility required for the hardware development process. In fact, it supports the major existing digital design flows such as register-transfer-level (RTL), high-level-synthesis (HLS), and model-based design solutions. Second, the strong test and debugging capabilities of the miniBEE4 are definitely a key prototyping feature that helps meeting the stringent 5G time-to-market constraints [29]. Third, we also emphasize the ability of the selected SDR to off-load heavy processing to the cloud or a centralized processing unit (e.g. C-RAN) owing to its high-data-rate connection interfaces.

B. BEECUBE PLATFORM STUDIO (BPS)

BPS is a high-level hardware/software co-development environment which runs on the top of the Mathworks Simulink®framework. It has been designed to abstract the low-level implementation details in order to accelerate the development and integration process. BPS ensures an automatic generation of all platform-specific hardware interfaces and the corresponding software drivers, thereby, ensuring a fluid CPU-FPGA interaction while enabling high flexibility in design partitioning [28].

C. THE EB PROPSIM FS8 CHANNEL EMULATOR

The EB Propsim FS8 channel emulator supports up to 4×4 bidirectional MIMO topology over various existing radio access technologies such as UMTS, LTE, LTE-A, MANET, and VANET. Furthermore, FS8 enables very accurate signal fading processing in terms of time, phase and

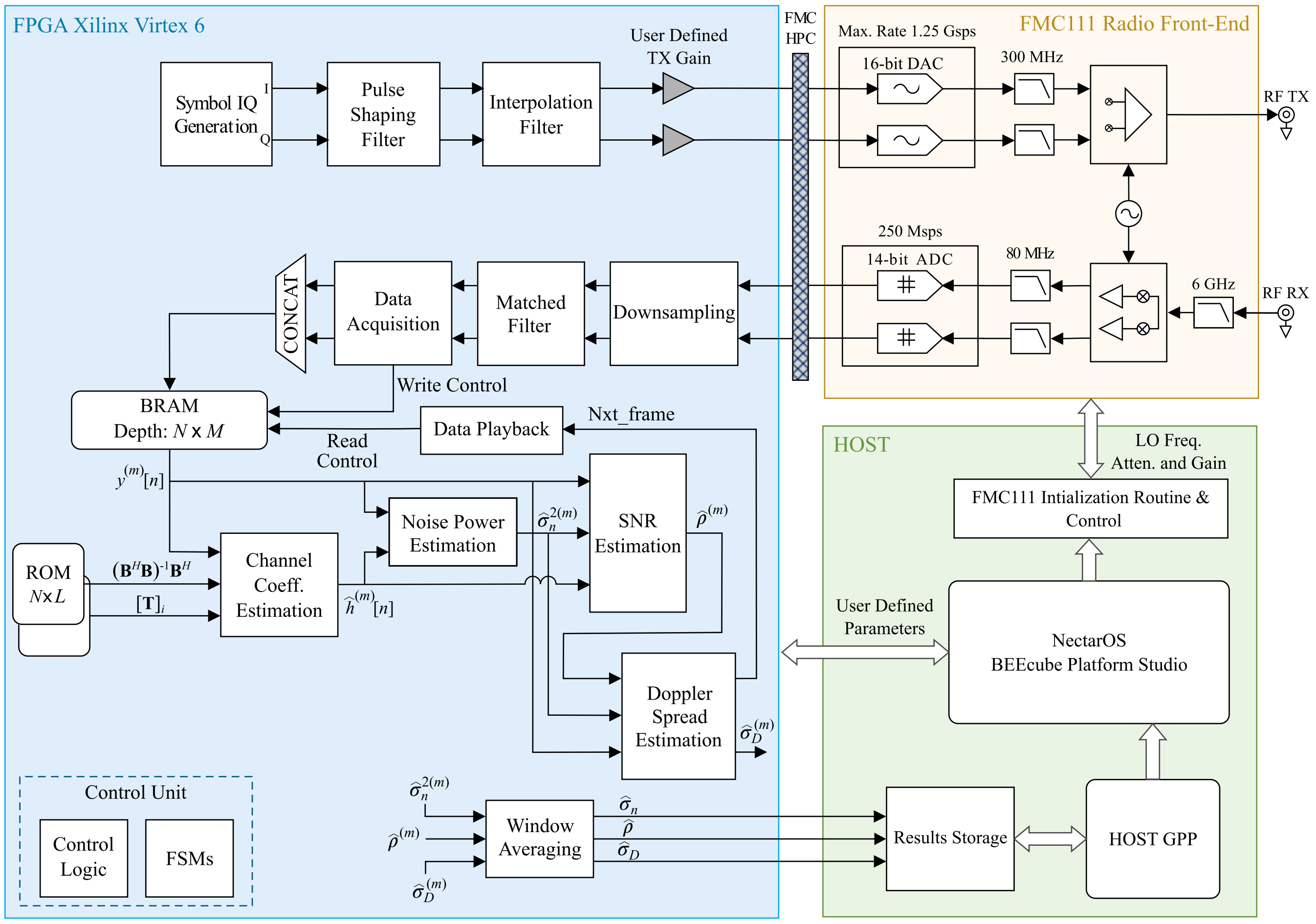

FIGURE 2. High-level FPGA architecture adopted for the joint DA ML SNR and Doppler spread estimator.

amplitude [30]. This feature made it a great option for our in-system over-the-air integration and evaluation.

IV. PROPOSED HARDWARE ARCHITECTURE

A. SYSTEM SPECIFICATIONS

From a system-level point of view, the outlined DA ML SNR and Doppler spread estimator architecture needs to be versatile and resource-efficient for future extensions. Moreover, as a part of our CTR's sensorial entity, this prototype should be reconfigurable in order to self-adapt to different environment conditions. Therefore, a wise CPU/FPGA design partitioning needs to be applied within the SDR platform. To that end, baseband processing should be sufficiently pipelined and entirely carried out by the FPGA resources. The high-level reconfiguration and parameter switching should be executed automatically or manually by the user through the SDR's CPU. Furthermore, the produced prototype must ensure an optimized resources/performance trade-off. As a performance metric, the 1 ms latency constraint for ultra-reliable and low-latency communications (URLLC) in future 5G systems [31] must be met while ensuring a satisfactory estimation accuracy for a wide set of real-world SNR values. The SDR-embedded SNR and Doppler spread estimator should be robust to hardware impairments and to setup imperfections. Therefore, it must preserve in real-time the accuracy of the MATLAB-based software version.

B. SYSTEM-LEVEL ARCHITECTURE

As discussed previously, the proposed FPGA design of our CTR's sensorial unit is embedded in a miniBEE4 SDR platform. Fig. 2 depicts a system-level view of our FPGA core within its hardware/software (HW/SW) development environment which allows us to use simultaneously a general purpose CPU and a customized hardware component, namely the FPGA. The outlined block diagram consists of 3 main parts [28]: the FPGA, the FMC111 board, and a host terminal. While the baseband processing is totally carried out by the FPGA, the host CPU manages the storage and analysis of the FPGA's real-time results for debugging and visualization purposes. Besides, it is up to the CPU to reconfigure the FPGA's architecture on a runtime (whenever required) depending on the desired performance and link conditions. This HW/SW functional partitioning benefits from the high flexibility of the software routines to execute the decision-making processes in the host CPU. Whereas, the FPGA computation capabilities are entirely dedicated to baseband processing.

And, the RF front-end is integrated in the FMC111 board. The latter enables the modification, through the host interface, of various parameters such as the local oscillator frequency, DAC/ADC clocks, and TX/RX gains, etc.

On the top of the functional HW/SW partitioning, a datapath/control partitioning is required in order to handle the algorithm's complexity in the most effective way. Unlike software instruction-based programs, a hardware design needs to be synchronized at the low level of the Register-Transfer-Level (RTL). From this perspective, our design datapath is an unsynchronized description of the baseband processing using a hardware description language (HDL). Meanwhile, our control unit monitors the design and controls the datapath module. To build this key unit, we make use of efficient synchronization tools, namely the finite state machines (FSMs) [32].

As illustrated in Fig. 2, the baseband I/Q symbols are generated, pulse-shaped, then up-sampled within the FPGA at the transmitter side. After applying a user-defined TX gain, the FMC111 RF card up-converts the baseband signal and broadcasts the resulting RF waveform over the communication link (i.e., the channel emulator in III.C). At the receiver side, the collected signals are down-converted then down-sampled prior to matched filtering. At this point, the *"Data Acquisition"* block acquires and stores NM data samples from the received baseband signal (at the pilot positions) in a shared SW/HW block RAM (BRAM) [28]. The parameters N, M, and the sampling period (T_s) are user-defined. They can be reconfigured —through the host CPU— depending on the application requirements and the underlying system's signaling standard. To ensure such flexibility and control the acquisition task, a dedicated FSM has been implemented. It takes these user-defined parameters and generates the appropriate triggers, commands, and other control signals in order to monitor the sampling and recording tasks. The stored samples are used at different estimation stages of our design. In the next subsections, we detail our proposed hardware architecture for the joint DA ML SNR and Doppler spread estimator.

C. DA ML SNR ESTIMATOR

As explained in Section II, the DA ML SNR algorithm estimates the local SNR in three steps:

- Estimate channel approximating-polynomial coefficients at the mt^th sub-block, $\widehat{\mathbf{c}}^{(m)}$, using (18);
- Estimate the noise power, $\widehat{\sigma}_n^{2(m)}$ over the m^{th} local approximation window using (19);
- Estimate the m^{th} local instantaneous SNR, $\widehat{\rho}^{(m)}$ using (20);

The first step is performed by the *"Channel Coeff. Estimation"* module. There, we estimate the local N channel coefficients through the process illustrated by the block diagram in Fig. 3.

In practice, an SDR allows one to modify —through the host CPU on a runtime— several operating parameters [33] such as the sampling period (T_s), the local window size (N),

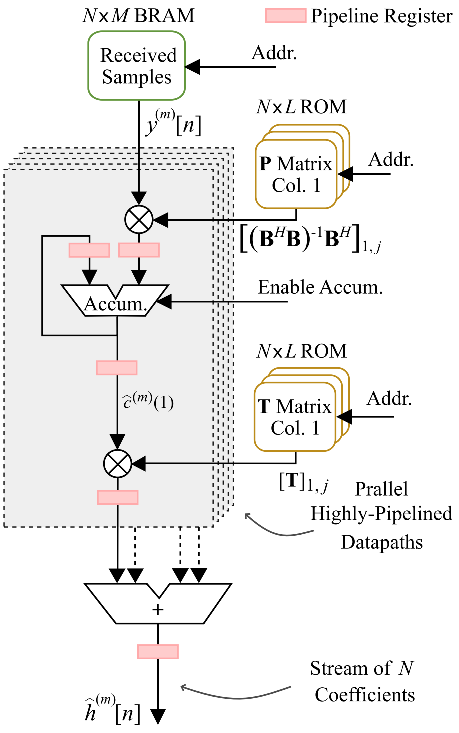

FIGURE 3. Block diagram of the *"Channel Coeff. Estimation"* module.

and the channel polynomial order (L). In our implementation, these parameters cannot be modified, on the fly, during the real-time operation of the hardware versions of the estimators. Consequently, the NL elements of the matrices $\mathbf{B}$ and $\mathbf{T}$, stay all constants and known *a priori*. A very cost-effective way to execute the *"Channel Coeff. Estimation"* task is to store matrix elements of $\mathbf{P}$ in the FPGA's read-only-memories (ROMs) to be ready-to-use on runtime. This avoids us the heavy computational burden of online high-order matrix multiplications and inversions, namely $\mathbf{P} = \left(\mathbf{B}^H\mathbf{B}\right)^{-1}\mathbf{B}^H$ in (18). Specifically, this approach reduces the number of multiply-and-accumulate (MAC) modules in our design by a factor of $N(N-1)(L-1)$.

At RTL, the proposed *"Channel Coeff. Estimation"* design incorporates L parallel high-pipelined datapaths, monitored by an optimized Moore FSM [32]. In the i^{th} datapath, $i = 1, 2, \ldots, L$, the data stream, ($\mathbf{y}^{(m)}$), of the N local received samples, already stored in BRAM, goes through a MAC module before multiplication by the N elements of the i^{th} column of $\mathbf{P} = \left(\mathbf{B}^H\mathbf{B}\right)^{-1}\mathbf{B}^H$. Then, the output results are accumulated to ultimately produce the i^{th} estimated polynomial coefficient, $\widehat{c}^{(m)}(i)$. Once the accumulator output is tested valid, the FSM enables the second multiplication $[\mathbf{T}]_{i,j} \times \widehat{c}^{(m)}(i)$. Finally, the L datapath outputs pass through an adder to obtain an online stream of the N estimated channel coefficients which are the results of (21).

The DA ML SNR algorithm provides us also with the noise power estimation. By considering all-ones pilot sequences, the matrix $\mathbf{A} = \text{diag}\{x(T_s), x(2T_s), \ldots, x(NT_s)\}$ is simply the identity matrix and, consequently, $\mathbf{B} = \mathbf{AT} = \mathbf{T}$.

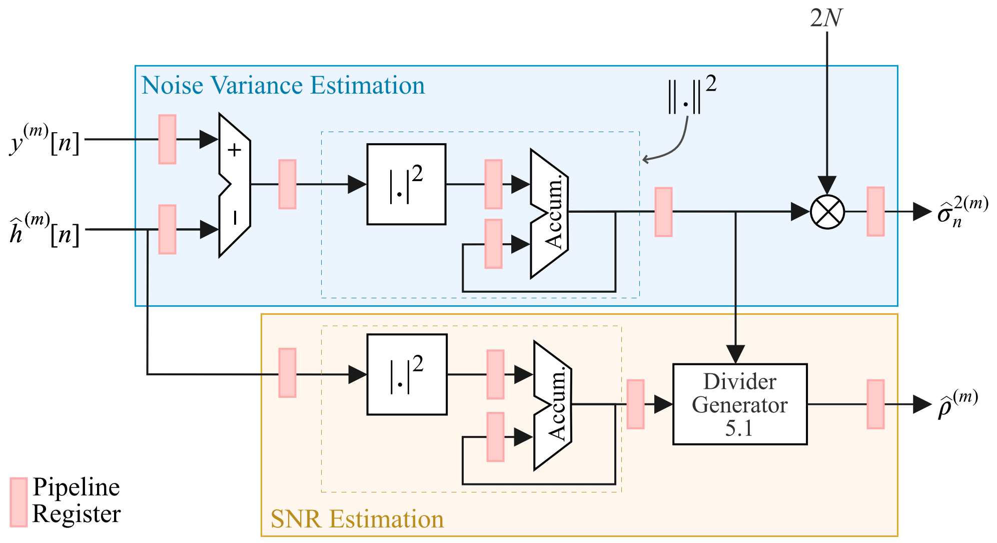

FIGURE 4. Block diagram of the *"Noise Variance Estimation"* and *"SNR Estimation"* modules.

Therefore, the estimate of the noise variance can be simply obtained as follows:

$$\widehat{\sigma}_n^{2(m)} = \frac{1}{2N}\left\|\mathbf{y}^{(m)} - \mathbf{T}\widehat{\mathbf{c}}^{(m)}\right\|^2 = \frac{1}{2N}\left\|\mathbf{y}^{(m)} - \widehat{\mathbf{h}}^{(m)}\right\|^2. \quad (22)$$

The handshake from the *"Channel Coeff. Estimation"* module is ensured by its inner FSM through a flag signal output that triggers the controller of the *"Noise Power Estimation"* block once the first element of the estimated channel coefficients stream becomes valid. An RTL description of the noise power estimation design according to (22) is shown in Fig. 4 wherein the design consists of a dedicated FSM (the operations controller), two adders, three multipliers, and an accumulator. This highly cost-effective self-controlled datapath provides the estimated noise variance to the last stage of the DA ML SNR estimator's architecture, the *"SNR Estimation"* module which estimates the local SNR according to (20). In line with the general principle of intellectual property (IP) reuse in hardware design, we make use of the *"Xilinx Divider Generator 5.1"* [34] to implement the division in (20). The output is a reconfigurable resource efficient and a high-performance solution based on radix-2 approach. The underlying IP division block allows reconfiguration of the degree of parallelism and the latency in order to meet the required trade-off between performance, speed, and resource utilization.

Finally, the estimated noise variance and SNR are fed as inputs to the *"Doppler Spread Estimation"* module, whose inner architecture is described in the next subsection.

D. ML DOPPLER SPREAD ESTIMATOR

In this subsection, we describe the proposed RTL architecture for the ML Doppler spread estimator. From a high-level point of view, the *"Doppler Spread Estimation"* module takes as inputs the estimated values of the noise power and local SNR, and the N received pilot samples (over the same m^{th} approximation window), and outputs the estimated Doppler spread value. Unlike the DA ML SNR estimator, the ML Doppler spread estimator [15] is not in closed form. Therefore, this module evaluates the LLF function at different candidate values for the unknown Doppler spread, then finds the global maximum. Consequently, it is obvious that this module will require more resources and relatively higher latency. An effective way to proceed is to start by further simplifying the mathematical equations of the ML Doppler spread estimator in order to adapt the required processing to hardware implementation. In other words, we rewrite the different mathematical terms involved in the LLF expression in such a way that we eliminate high-cost operations such as the square root and ensure maximal block reuse. Indeed, after some algebraic manipulations, (8) is rewritten as (23).

Beyond mathematical model adaptations, operation scheduling is a critical step in the design of efficient hardware architectures [35]. It identifies parallel operations, the predecessors and successors of each operation, as well as, their inter-dependencies in order to design a robust control unit. Furthermore, owing to operation scheduling, it is possible to prematurely identify critical paths (i.e., those having the highest relative latencies) and then accelerate them by incorporating highly-pipelined datapaths.

$$\begin{aligned} &L^{(m)}(\sigma_D) \\ &= -\ln\left[2+\widehat{\rho}^{(m)}\lambda_1(\sigma_D)\right] - \ln\left[2+\widehat{\rho}^{(m)}\lambda_2(\sigma_D)\right] \\ &\quad + \frac{\frac{\widehat{\rho}^{(m)}}{2\widehat{\sigma}_n^{2(m)}}}{2+\widehat{\rho}^{(m)}\lambda_1(\sigma_D)}\left|\left(\mathbf{a}(-\sigma_D)+\frac{\varphi^*(2\sigma_D T_s)}{|\varphi(2\sigma_D T_s)|}\mathbf{a}(\sigma_D)\right)^H \mathbf{y}^{(m)}\right|^2 \\ &\quad + \frac{\frac{\widehat{\rho}^{(m)}}{2\widehat{\sigma}_n^{2(m)}}}{2+\widehat{\rho}^{(m)}\lambda_2(\sigma_D)}\left|\left(\mathbf{a}(-\sigma_D)-\frac{\varphi^*(2\sigma_D T_s)}{|\varphi(2\sigma_D T_s)|}\mathbf{a}(\sigma_D)\right)^H \mathbf{y}^{(m)}\right|^2. \end{aligned} \quad (23)$$

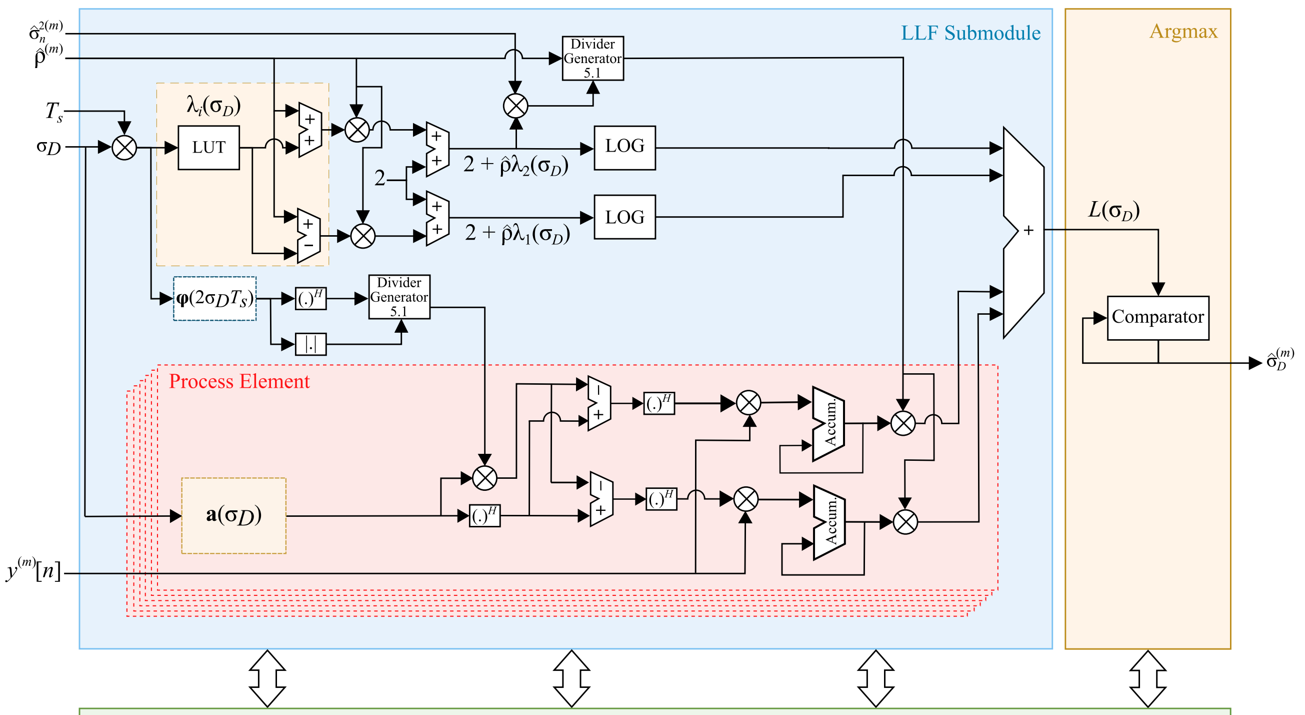

FIGURE 5. System-level view of the proposed hardware design.

Through an intuitive functional partitioning, we structured the RTL architecture of the *"Doppler Spread Estimation"* module into the *"LLF Submodule"*, *"Argmax"*, and *"Control Unit"* blocks as depicted in Fig 5. Each block was designed, verified, and validated separately before its integration and synchronization within the general structure. The *"LLF Submodule"* is the capital block of the Doppler spread estimator. There, the candidate σ_D goes through the so-called $\lambda_i(\sigma_D)$ and $\varphi(\sigma_D)$ sub-blocks. The latter mainly consist of two look-up-tables (LUTs) in which predefined complex trigonometric functions are stored instead of using high-cost techniques such as CORDIC algorithms [36]. The appropriate output address can be found via a bijective conversion which depends on the input data, the input's range, and the desired resolution. Explicitly, this bijection is given by:

$$address = \left\lfloor (input - min) \times \frac{depth - 1}{max - min} \right\rfloor, \qquad (24)$$

where $input \in [min, max]$, $address \in \{0, 1, \ldots, depth - 1\}$, and $depth$ is the size of the LUT, while $\lfloor . \rfloor$ denotes the flooring function of real numbers.

The main advantage of a LUT is its very low latency. In fact, obtaining data from a predefined array is much faster than calculating it using iterative or rotating algorithms. Yet, the design of an efficient LUT is still challenging as it requires the knowledge of the upstream range, which is not *a priori* known in several applications. Furthermore, there is a trade-off between the output resolution and the upstream range, since the larger the range, worse is precision. In order to sidestep this issue, one may be tempted by integrating huge LUTs. Unfortunately, such a simplistic solution is not even practical due to the high-size ROM usage it requires. Hence, we propose a new technique that reduces the size of the required LUTs without necessarily compromising accuracy.

The new solution consists in integrating pre-processing and post-processing stages in order to reduce the upstream range by exploiting some basic mathematical identities. Indeed, in our case, LUTs are only used to perform trigonometric functions. Therefore, we use in a first step an obvious translation of the input data into the range $[-\pi, \pi]$. This pre-processing block was incorporated in order to map the input angle to its main measure. Due to this optimization, we greatly reduce the LUT's depth and consequently decrease memory utilization. Besides, we further reduce hardware usage by encoding only the first quadrant (i.e., the upstream range is limited to $[0, \pi/2]$) in the LUT and then casting inside any angle outside this quadrant from proper adjustments of the input data signs. This simplification reduces four times memory usage. A post-processing block unwraps the angle back into the original four quadrants to return the correct value. Fig. 6 illustrates the inner RTL structure of the proposed sine LUT. There, the input x (with an unknown data range) is translated into its main value (denoted as θ) and assigned conventionally to the first quadrant using two multipliers, an adder, and a truncate circuit that gives the

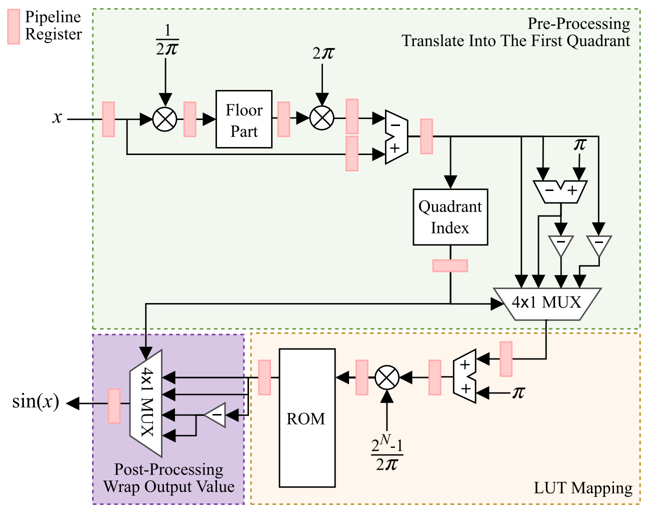

FIGURE 6. Inner structure of the optimized sine function's LUT.

floor value. Then, by relying on some comparators, it is possible to determine the quadrant index of the main angle value. This index is used to control the behavior of the 1×4 *MUX* and select the appropriate value between θ, $(\pi - \theta)$, $(\theta - \pi)$, and $-\theta$. This value will be fed to the ROM in which the pre-calculated sine grid points are stored before being transformed into a memory address using the bijection described in (24).

By using these optimization strategies along with some arithmetic and logic instructions, we succeed in building accurate output values for the complex functions $\frac{\varphi(2\sigma_D T_s)^H}{|\varphi(2\sigma_D T_s)|}$, $[2 + \widehat{\rho}\lambda_1(\sigma_D)]$, and $[2 + \widehat{\rho}\lambda_2(\sigma_D)]$, three basic terms used extensively in a later step to evaluate the LLF expression.

Meanwhile, a systolic-based process is run to compute the vector $\mathbf{a}(\sigma_D)$ and perform the projection on the received samples. The idea behind this approach is to decompose the required inner product operation —between any two real vectors $\mathbf{u}$ and $\mathbf{v}$ — into multiple parallel threads. Each thread runs a process element (PE) which independently computes a partial result as a function of the data received from its upstream. An example of a K-order decomposition, where $q\frac{N}{K}$ is an integer, is given by:

$$\mathbf{u}^T\mathbf{v} = \sum_{k=0}^{K-1}\left\{\sum_{n=1}^{N/K} u\left[n+k\frac{N}{K}\right]v\left[n+k\frac{N}{K}\right]\right\}. \qquad (25)$$

By carefully controlling these parallel PEs, the above decomposition results in a $(\frac{N}{K}\times)$ saving in terms of latency.

As described previously, the ML Doppler spread estimator evaluates the LLF at different candidate values for the Doppler spread σ_D. Their range varies depending on the application. In a high-speed railway, for instance, the maximum mobile velocity can reach $v = 500\ km/h$ [37]. By considering a transmission over a carrier frequency of about $f_c = 2.45$ GHz, the maximum Doppler frequency is $f_D = \frac{v}{c}f_c \simeq 1200$ Hz, where c denotes the speed of light.

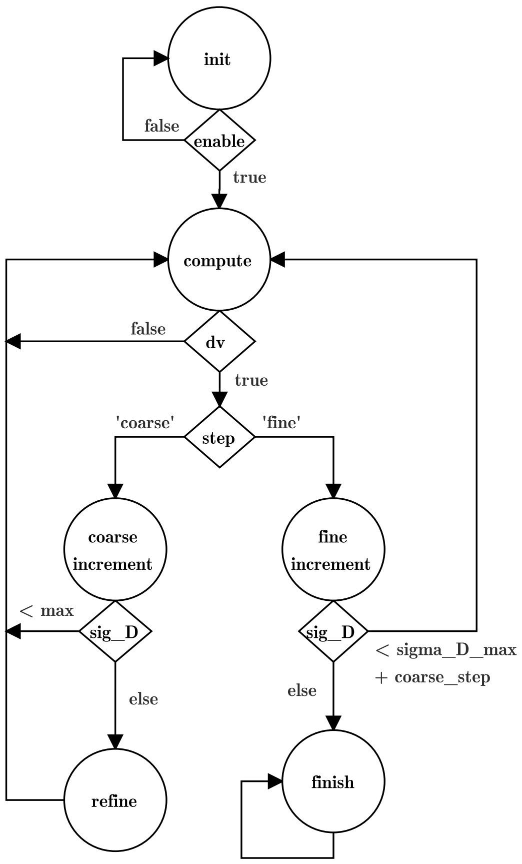

FIGURE 7. State diagram of *"Doppler Spread Estimation FSM"*.

Thus, to cover almost all possible applications at the given carrier, we need to evaluate the LLF function at different candidates in the range[3] $\sigma_D \in [0, 10000]$ rad.s^{-1}. Obviously, the latency of this process is directly related to the search technique. In this work, we propose a coarse-to-fine (CTF) technique [38] for the maximum's detection.

The CTF search strategy, an efficient algorithm for detection, reduces the computation latency and the FPGA resource usage by minimizing the searching range. In fact, we first evaluate the LLF at a number of candidates incremented by a coarse step in order to detect a first coarse maximum. Then, we use a fine search by incrementing the candidate value in a limited range to find the local maximum. We design a dedicated FSM to control and monitor the CTF search task, whose state diagram is illustrated in Fig. 7. Initially, the FSM is in the *INIT* state where the candidate value (σ_D) is initialized to the user-defined *min* parameter. Once the *enable* signal is set to valid (*enable* is a status signal produced by the *"SNR Estimation"* module), the machine transits immediately to the *COMPUTE* state that enables the LLF computation for the current Doppler spread candidate value. Once the computing

[3] Please note here that the maximum Doppler frequency is related to the Doppler spread, σ_D, where the underlying relationship depends on the specific channel's spectrum model. In this work, we consider the widely studied uniform Jakes' model for which we have: $\sigma_D = 2\pi f_D/\sqrt{2}$ [15].

TABLE 2. Hardware-in-the-loop co-simulation parameters.

	Parameter	Description	Value
Baseband Configuration	N	Local window size	128
	M	Number of local windows	8
	T_s	Sampling period	10 μsec
	L	Channel polynomial coefficients order	3
FPGA/FMC111 Front-end Configuration	f_c	Carrier frequency	2.4576 GHz
	f_{IF}	Intermediate frequency	32.720 MHz
	vga	Variable gain amplifier	8 dB
	rx_atten	RX attenuation	8 dB
	DAC/ADC freq	DAC/ADC sampling frequency	61.440 MHz
	FPGA clock	FPGA clock	61.440 MHz

task is finished, indicated by a high-logic-level of a status signal, the candidate value will be incremented by a coarse or a fine step, according to the current search task. A control logic on the candidate value is made in the *COARSE INCREMENT* and the *FINE INCREMENT* states. As long as the candidate value does not exceed the range limit, denoted by the *max* parameter, the FSM moves to the *COMPUTE* state and starts another computation task. Otherwise, it refines the *step* value or transits to the *FINISH* state. Here, the FSM indicates the end of the estimation task with the status signal *finish*. This signal is fed to the *"Windowing"* FSM, i.e., the top-level control unit in charge of data playback and estimation averaging.

E. FUNCTIONAL SIMULATION OF THE DESIGN

The pre-synthesis final simulation of the design verifies whether the overall system's processing matches the behavior of the reference MATLAB-based software version in terms of datapath accuracy and inter-component interactions ensured by the high-level control unit. The simulation process is made much easier with the strong simulation capabilities of the BPS environment [28] wherein several HW/SW shared blocks allow a bidirectional reading and writing access both from the host processor and the FPGA side and, hence, enable an effective MATLAB/hardware interaction. In fact, HDL test-benches are not needed since stimulus signals and test vectors can be generated in the MATLAB workspace then injected dynamically to the HW design via the *"FromWorkspace"* block. Besides, the outputs and the internal signals can be visualized using the *"Scope"* and *"Display"* interfaces or simply stored to the MATLAB workspace for further analysis via the *"ToWorkspace"* block.

The simulation results obtained in BPS environment are cycle- and bit-accurate, and mirror those which should be obtained once the design is implemented on the SDR's FPGA. The resulting accuracy, despite the fixed point representation imperfections, is enough to validate the design's datapath and synchronization before its hardware synthesis and implementation.

V. DESIGN TRADE-OFF

The joint DA ML SNR and Doppler spread estimator, whose top-level block diagram is shown in Fig. 2, was designed and tested within the BPS framework. During the design process, we exploited the Xilinx IP blockset to conceive the proposed architecture over a model-based framework. Nevertheless, we made use of MATLAB and VHSIC HDL (VHDL) particularly for the finite state machine's description. The range analysis of the different inputs/outputs, parameters and internal signals is a key factor in the design trade-off. We took advantage of MATLAB's fixed-point dedicated tool [39] to perform this task. This tool accelerates range analysis by intensively simulating the entire design with the user-specified fixed-point representations, and comparing it with the floating-point version in a way to maximize precision while covering the dynamic range. Based on this comparison, an optimized fixed-point representation of the design signals is produced.

It is worth mentioning that we are able to directly influence the performance/FPGA usage trade-off by specifying several internal parameters within the Xilinx IP blocks of the design such as latency, synthesis, and implementation rules so as to save area or increase processing speed. The design was synthesized, mapped, placed, and routed using Xilinx System Generator tool, and finally implemented on the FPGA of the miniBEE4 SDR platform.

In this section, we will evaluate thoroughly the proposed hardware architecture in terms of hardware design trade-off. Indeed, we will investigate the effect of the proposed hardware optimizations on the overall FPGA resource utilization, energy consumption, and processing latency. Then, we will examine the overall system performance of the optimized architecture in terms of estimation accuracy under several propagation scenarios. To that end, we consider the configuration of the system's setup parameters displayed in Table 2. There, a basic baseband configuration is adopted along with CMC's FMC111 front-end configuration parameters already fine-tuned and set in [28]. We also fix the FPGA's clock rate to the DAC/ADC sampling frequencies in order to comply with CMC's recommendations for best operation of the miniBEE4 SDR platform.

A. FPGA RESOURCE UTILIZATION

Since there are no related works on the hardware implementation of the considered joint estimator, we conceive another "unoptimized" design which employs temporary buffers, CORDIC blocks, and does not implement the CTF search technique. We then compare the resource utilization of the optimized solution to the straightforward unoptimized architecture, called *Unoptimized Architecture*, and evaluate the impact on the overall usage/performance trade-off. The FPGA programming file is generated for the target device *Virtex-6 XC6VSX475T* and its resource utilization is summarized in Table 4.

At a glance, one can easily verify that the introduced optimization techniques offer a large savings in terms of resource utilization by consuming less than 2% of the available slice registers on the target board. Whereas *Unoptimized Architecture* requires up to 10% of the same resource or five times more. Moreover, the optimized design uses 9837 out of 297600 LUTs, which represents only 3.31% of all the LUTs provided on the Virtex-6 FPGA. The last two features presented in Table 4 (i.e., BRAM/FIFO and DSP48) are specific limited resources available on the Virtex-6 device series. There, the optimized design uses 34.86% and 17.61% of the available BRAM/FIFO and DSP48 cores, respectively. Compared to the unoptimized design (i.e., *Unoptimized Architecture*), it saves up to 34.45% of the BRAMs/FIFOs but loses about 11.64% of the DSP48 resources.

To summarize, the proposed optimization strategies bring together up to (5×), (4×), and (1.5×) savings in terms of slice registers, LUTs, and BRAM/FIFO utilization, respectively. This comes at the cost, however, of losing 11% of the DSP48 specific resources. This is due to the fact that the proposed optimized design considers the DSP48 high-performance MAC slices instead of using embedded multipliers or a Fabric approach that takes though longer latency to produce a valid output. These strategies require considerable routing efforts, and are less efficient in terms of hardware costs and latency [40]. Finally, as expected, Table 4 confirms that the optimized design uses a very small area of the target chip, making it an attractive candidate for possible peripheral extension for industry standard compliance and for integration in future 5G cognitive transceivers.

As an overall benchmark, we will gauge the complexity of the proposed hardware design against works on the hardware implementation of Doppler-based techniques for different applications. For instance, [41] developed FPGA implementations of improved least-squares methods for Doppler centroid frequency fitting dedicated to a synthetic-aperture radar (SAR) application. There, the authors compare the proposed techniques against the traditional linear least-squares methods in terms of complexity. The resources utilization provided therein for a Xilinx Virtex6 XC6VLX240T, a chip from the same FPGA family adopted in our hardware setup, are summarized in Table 4. In terms of slice logic usage, our design outperforms the traditional least-squares methods. It is worth mentioning here that the achieved gains are actually much higher because the proposed implementation encompasses a lot more than a Doppler fitting method per se. Rather it literally estimates the Doppler spread jointly with the SNR and the wireless channel at the expense of an additional computational burden and more FPGA resources. Another work [42] implemented a Doppler ultrasound imaging system on a Xilinx Virtex 5 FPGA, a smaller one compared to our Virtex 6 that can still stand as a valid reference. Regardless of its application, the DSP complexity of the outlined technique is comparable to ours. As reported by the authors, this technique uses 3223 out of 69120 slice registers and 3199 out of 69120 slice LUTs, that is 4.67% and 4.62% of the total available slices, respectively. FPGA resources usage in [42] is comparable to ours. And that is without accounting once again for the key fact that our design jointly estimates the Doppler, the SNR, and the wireless channel. These sample comparisons highlight the effectiveness of our optimized design.

TABLE 3. On-chip power estimation and maximum operating frequency results at 25°C.

	Arcitecture I	Proposed Architecture
Dynamic Power (mW) at 61.44 MHz	1191.43	741.96
Static Power (mW) at 61.44 MHz	7489.74	
Energy for 1000 Clock Cycles (μJ)	141.33	133.98
Max. Operating Frequency (MHz)	125.33	149.12

B. ENERGY CONSUMPTION

The saving in terms of resource utilization directly translates into a reduction of about 38% in dynamic power consumption. This result was obtained using the Xilinx Power Estimator (XPE) tool. For the sake of fairness, we estimated the energy and power consumption at the same operating frequency of 61.44 MHz and the results are summarized in Table 3. There, the optimized highly-pipelined architecture saves 6% in energy consumption compared to the unoptimized one. Moreover, the synthesis results show that the optimized architecture achieves a maximum operating frequency significantly higher than that of *Unoptimized Architecture*.

C. PROCESSING LATENCY

In this subsection, we focus on the processing latency which is a very important performance metric in the FPGA design trade-off. To do so, we compute the clock-cycle-accurate latencies for the different processing blocks of the proposed architecture and list the results in Table 5. We observe that the *"Channel Estimation"*, the *"Noise Variance Estimation"*, and the *"SNR Estimation"* modules require only 148, 135, and 36 clock cycles, respectively. These relatively low latencies are due to the highly-parallel and deeply-pipelined

TABLE 4. **Resource utilization results at $N = 128$, $M = 8$, and $L = 3$.**

	Slice Logic		Specific Features	
	Registers	**LUTs**	**BRAM/FIFO**	**DSP48**
Available on XC6VSX475T	595200	297600	1064	2016
Unoptimized Achitecture	58789 (9.88%)	36117 (12.14%)	563 (52.91%)	318 (15.77%)
Proposed Achitecture	11854 (1.99%)	9837 (3.31%)	369 (34.68%)	355 (17.61%)
Gain	79.83%	72.76%	34.45%	-11.64%
Available on XC6VLX240T	301440	150720	416	768
Least Squares [41]	13959 (4.63%)	14962 (9.92%)	Unavailable	
ROM-Based [41]	6666 (2.21%)	7382 (4.89%)	Unavailable	
Time-shared [41]	5972 (1.98%)	7080 (4.69%)	Unavailable	

TABLE 5. **Detailed design latency at 61.44 *MHz* FPGA speed with $N = 128$, $M = 8$, $L = 3$, $min = 0$, $max = 10000$, $coarse\_step = 400$, and $fine\_step = 40$.**

Module	Clock cycles	Latency (μsec)
Channel Estimation	148	2.408
Noise Variance Estimation	135	2.196
SNR Estimation	36	0.586
Doppler Spread Estimation	6643	108.272
LLF module	138	2.246
Total Local Estimation	**6962**	**113.462**
Averaging	9	0.146
Total	**55705**	**906.652**

HDL architectures. The number of required clock cycles depends mainly on the local approximation-window-size (N) since the inner product is the most expensive operation in terms of execution time (i.e., latency). Similarly to the mentioned blocks, the *"LLF-submodule"* needs just 138 clock cycles to calculate a valid output for each Doppler candidate value. However, the whole *"Doppler Spread Estimation"* module takes up to 95% of the total processing time. This is due the LLF function evaluation within this module at Q different candidate values, according to the CTF approach described previously, where:

$$Q = \left\lfloor \frac{max - min}{coarse\_step} + 2 \times \frac{coarse\_step}{fine\_step} \right\rfloor . \qquad (26)$$

In (26), Q represents the latency factor equal to 45 with our setup.

During the estimation process, we considered the case of $M = 8$ local approximation windows. Assuming an online averaging, the global latency of the entire system is approximately M times the latency of the local estimation process. Running the design with an FPGA's operating frequency of 61.44 MHz, the entire proposed design takes less than 1 *msec* to jointly estimate the channel coefficients, the noise variance, the SNR, and the Doppler spread over $M = 8$ local approximation windows of size $N = 128$. Without implementing the CTF search technique, the hardware design with the same configuration would have taken 4.570 *msec* to execute the whole estimation process.

VI. EXPERIMENTAL RESULTS

The ultimate objective of this work is a first hardware realization and integration of the outlined joint SNR and Doppler spread ML estimator. In the previous section, we proved that the proposed design meets the required specifications while ensuring a satisfactory cost/performance trade-off. In this section, we analyze the SDR-integrated prototype's performance in terms of estimation accuracy under realistic propagation scenarios. To do so, we embed the produced FPGA core of the estimator in the baseband chain depicted in Fig. 2. The whole system is implemented in the miniBEE4 SDR platform. Between the *RF TX* and the *RF RX* ports, we install the EB Propsim F8 channel emulator in order to mimic realistic wireless channels within real-time propagation scenarios. This integrated channel emulator enables building, defining, and customizing wireless channel models by configuring several parameters such as mobile velocity, gains and attenuations, path-loss, etc.

To assess the performance of our hardware prototype in terms of estimation accuracy, we perform a comparative study between two setups using the normalized mean square error (NMSE) as a performance metric:

i) MATLAB simulations: the estimation results are obtained using MATLAB floating-point simulations. The wireless channel is generated using MATLAB scripts;
ii) EB Propsim - mBEE4 SDR real-time emulations: this setup represents the full hardware version of our FPGA-embedded estimator. The reconfigurable RF interface is provided by the host mBEE4 SDR platform, and the wireless channel is emulated using the EB Propsim F8 channel emulator.

In all simulations, the NMSE is computed over $M_c = 10000$ Monte-Carlo runs. We start by studying the effect of the maximum Doppler frequency (f_D) on the performance of the channel coefficients estimation in setup ii. There, the channel is emulated by EB Propsim for different maximum Doppler frequencies. Simultaneously, on-the-fly channel estimation is performed by the miniBEE4 SDR. In Fig. 8, we can see that the proposed hardware prototype managed to successfully identify the channel and to track its variations for the entire set of the proposed Doppler frequencies. In high-mobility scenarios, channel estimation becomes a challenging issue due to the channel's fast time variations. Despite this fact, Fig. 8.c confirms the ability of the proposed FPGA-embedded estimator to correctly identify and track

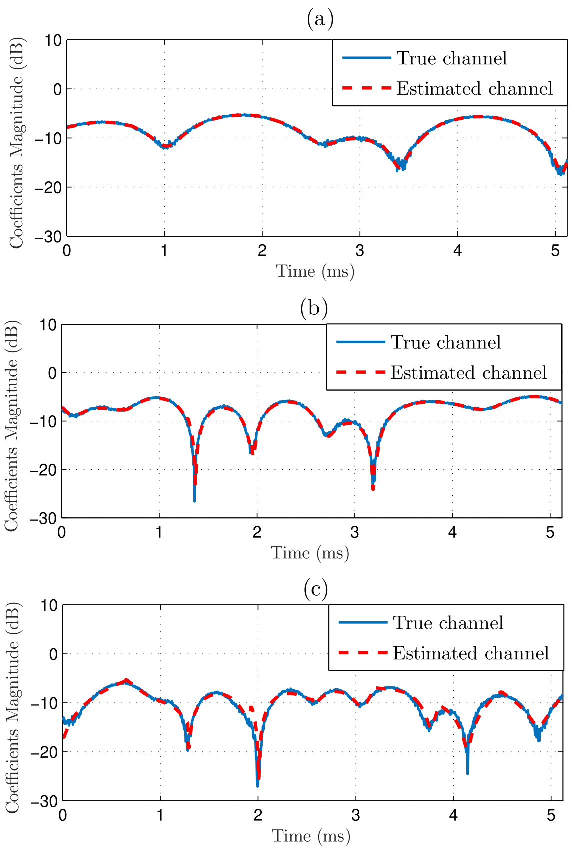

FIGURE 8. Estimated vs. real channel at (a) $f_D = 400$ Hz, (b) $f_D = 800$ Hz, and (c) $f_D = 1200$ Hz.

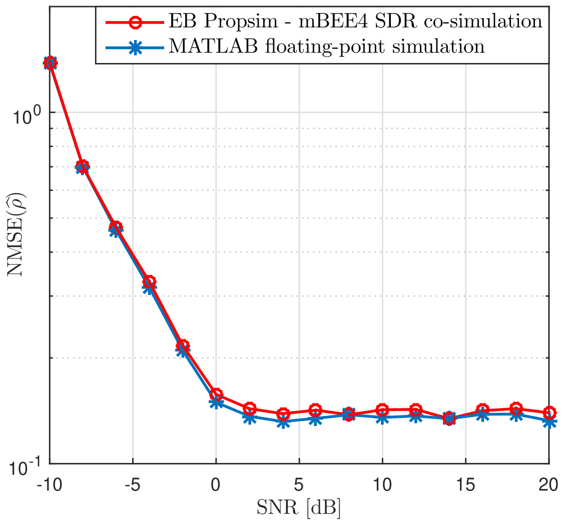

FIGURE 9. SNR NMSE obtained from EB Propsim - mBEE4 SDR real-time emulations against MATLAB-based simulations as function of the average SNR, with $f_D = 200$ Hz and $L = 3$.

highly time-varying channels, i.e., where the user mobility can reach 500 Km/h in Fig. 8(c) as required in future 5G mobile communication systems.

In Figs. 9 and 10, we study the impact of the average (i.e., long-term) SNR on the "instantaneous SNR" and the Doppler spread estimation performance, respectively, with

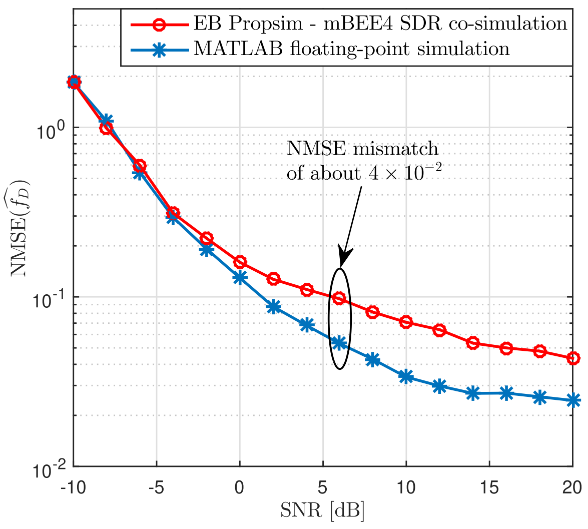

FIGURE 10. Doppler NMSE obtained from EB Propsim - mBEE4 SDR real-time emulations against MATLAB-based simulations as function of the average SNR, with $f_D = 200$ Hz and $L = 3$.

both MATLAB floating-point and EB Propsim-miniBEE4 SDR setups. The average SNR is defined as follows:

$$\mathrm{SNR} = \frac{E\left\{|x(n)|^2\right\}}{2\sigma_n^2}, \tag{27}$$

in which $E\{.\}$ denotes the expectation over all transmitted symbols. But since the constellation energy is assumed to be normalized to one, i.e., $E\left\{|x(n)|^2\right\} = 1$, the average SNR is simply given by:

$$\mathrm{SNR} = \frac{1}{2\sigma_n^2}. \tag{28}$$

Fig. 9 suggests that the NMSE curves for both hardware and software versions coincide and follow the same trend over a wide range of practical average SNRs, i.e., between -10 and 20 dB. This confirms the validity of the proposed SNR estimator hardware design and validates its real-time performance under real-world operating conditions.

The NMSE of the estimated Doppler frequency versus the average SNR for the considered setups is depicted in Fig. 10. There, we can see that both curves basically follow the same trend albeit the presence of a negligible NMSE mismatch of about 4×10^{-2} versus the whole range of long-term SNR. Such performance mismatch is due to the hardware imperfections introduced by the fixed-point representation of the signals within the FPGA, the miniBEE4 SDR DAC/ADC quantization errors and resolution, the EB Propsim channel emulator, and the interconnections between the setup equipments.

In Fig. 11, we compare the hardware-based real-time Doppler NMSE to its MATLAB counterpart versus the true Doppler. The latter is fixed in the considered setups through the EB Propsim channel emulator user interface and in the MATLAB simulation code. The results confirm that the hardware-based real-time results obtained on the SDR

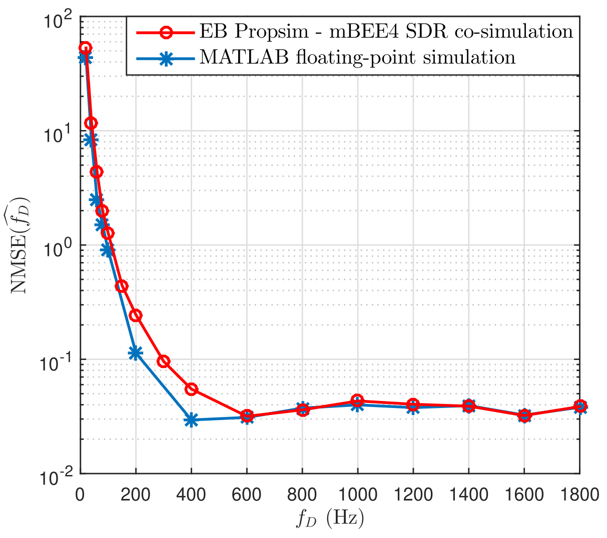

FIGURE 11. NMSE of the Jakes' Doppler frequency estimation using EB Propsim - mBEE4 SDR real-time emulations against MATLAB-based simulations vs. the real Doppler frequency, at SNR = 0 dB.

platform are equivalent to those obtained offline through idealized MATLAB-based simulations even in harsh propagation conditions (i.e., SNR = 0 dB) and, over a wide range of practical Doppler values. Again, the discrepancy between the two curves is mainly due to the hardware imperfections introduced by the FPGA, the RF-front end, and the channel emulator. Indeed, we observe from Fig. 11 that these imperfections have worse effects at small Doppler values. This is in part due to the fact that decreasing the SNR (due for instance to quantization errors and hardware-inherent thermal noise) affects the estimation of small Doppler values more than it does for large ones [15].

To be more specific, the hardware imperfections introduced by the miniBEEE4 platform coupled with the Anite's EB Proposim channel emulator are either of electronic and/or RF nature [43]. In fact, the electronic cluster consists of thermal and flicker noise components resulting from electrons motion and the latter are inevitable sources of signal corruption. Moreover, LUT-mapping and fixed-point data representation introduce quantization errors which are folded in the overall additive noise, thereby resulting in an SNR deterioration. Besides, according to [44], phase noise, front-end components' non-linearities, and IQ-imbalance are considered as the most important phenomena that degrade the performance of wireless communication systems. Based on the system model presented in [44] and [45], the baseband hardware received signal, corrupted by the hardware noise component is given by:

$$\widetilde{y}[n] = K_1 y[n] + K_2 y^*[n] + DC + n_{th} + n_f + n_q, \qquad (29)$$

where $K_1 = (1 + Ge^{-j\Phi})/2$ and $K_2 = (1 - Ge^{-j\Phi})/2$ denote the non-linearity and IQ-imbalance coefficients represented by the gain mismatch G and the phase error Φ, respectively. DC is the complex DC-offset introduced by the entire system on the constellation while n_{th}, n_f, and n_q denote the thermal, flicker, and quantization noises, respectively, assumed to be zero-mean and mutually uncorrelated.

As evidenced by (29), hardware imperfections decrease the effective SNR at the receiver side and induce a modified channel model, i.e., $\widetilde{\mathbf{h}} = K_1\mathbf{h} + K_2\mathbf{h}^*$. Obviously, this channel modification affects many intrinsic channel properties such as its perceived Doppler spread at the receiver side. Ultimately, this may also dramatically increase the bit error rate (BER) of the system, thereby decreasing the intended quality of service. More appropriate methods that better mitigate or account for these hardware imperfections will be developed and then implemented and tested in hardware in a future work.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed and developed the FPGA design, the hardware implementation, the SDR integration, and the experimental validation in real-world operating conditions of a joint DA ML estimator for the SNR and Doppler spread parameters. This joint estimator is most suitable for future 5G wireless communication systems that deploy context-aware cognitive transceivers. We have produced a proof of concept that validates unambiguously the very high accuracy, cost efficiency, and robustness of this very promising joint estimator despite the presence of hardware impairments. Through this work, we have explored the whole FPGA prototyping process from the design to the integration and experimental testing in real-world conditions. A top-down approach was adopted to design an optimized and flexible datapath and a robust control unit. A highly-efficient and deeply-pipelined reconfigurable HDL architecture was built using a model-based framework, then integrated on a SDR platform, and tested in real-world propagation conditions produced by a powerful channel emulator. The proposed architecture requires a relatively very small FPGA area to host the joint estimator, thus allowing new extensions in compliance with industry standards. In future works, we plan to develop efficient hardware prototypes for other modules of the CTR, [6] then integrate them to build and showcase a fully operational version in real-time OTA conditions.

ACKNOWLEDGMENT

The authors would like to acknowledge CMC Microsystems for the provision of products and services that facilitated this research, including CAD tools and prototyping platforms.

REFERENCES

[1] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, 1st Quart., 2015.

[2] M. E. Hoque, *Advanced Applications of Rapid Prototyping Technology in Modern Engineering*. Rijeka, Croatia: InTech, 2011.

[3] Nokia. (2014). *5G Use Case and Requirements—White Paper*. [Online]. Available: resources.alcatel-lucent.com/asset/200010

[4] S. E. Elayoubi, M. Fallgren, P. Spapis, G. Zimmermann, D. Martin-Sacristan, C. Yang, S. Jeux, P. Agyapong, L. Campoy, Y. Qi, and S. Singh, "5G service requirements and operational use cases: Analysis and METIS II vision," in *Proc. IEEE Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2016, pp. 158–162.

[5] W. H. Chin, Z. Fan, and R. Haines, "Emerging technologies and research challenges for 5G wireless networks," *IEEE Wireless Commun.*, vol. 21, no. 2, pp. 106–112, Apr. 2014.

[6] I. Mrissa, F. Bellili, S. Affes, and A. Stéphenne, "A context-aware cognitive SIMO transceiver for enhanced throughput on the downlink of LTE HetNet," *Wireless Commun. Mobile Comput.*, vol. 16, no. 11, pp. 1414–1430, Aug. 2016.

[7] T. Yoo, N. Jindal, and A. Goldsmith, "Multi-antenna downlink channels with limited feedback and user selection," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 7, pp. 1478–1491, Sep. 2007.

[8] J. C. Ikuno, M. Wrulich, and M. Rupp, "System level simulation of LTE networks," in *Proc. IEEE 71st Veh. Technol. Conf. (VTC-Spring)*, Taipei, Taiwan, May 2010, pp. 1–5.

[9] M. D. Austin and G. L. Stuber, "Velocity adaptive handoff algorithms for microcellular systems," *IEEE Trans. Veh. Technol.*, vol. 43, no. 3, pp. 549–561, Aug. 1994.

[10] G. Park, D. Hong, and C. Kang, "Level crossing rate estimation with Doppler adaptive noise suppression technique in frequency domain," in *Proc. IEEE 58th Veh. Technol. Conf. (VTC-Fall)*, vol. 2, Oct. 2003, pp. 1192–1195.

[11] S. Mohanty, "VEPSD: A novel velocity estimation algorithm for next-generation wireless systems," *IEEE Trans. Wireless Commun.*, vol. 4, no. 6, pp. 2655–2660, Nov. 2005.

[12] K. E. Baddour and N. C. Beaulieu, "Robust Doppler spread estimation in nonisotropic fading channels," *IEEE Trans. Wireless Commun.*, vol. 4, no. 6, pp. 2677–2682, Nov. 2005.

[13] C. Tepedelenlioglu and G. B. Giannakis, "On velocity estimation and correlation properties of narrow-band mobile communication channels," *IEEE Trans. Veh. Technol.*, vol. 50, no. 4, pp. 1039–1052, Jul. 2001.

[14] O. Mauritz, "A hybrid method for Doppler spread estimation [mobile radio systems]," in *Proc. IEEE 59th Veh. Technol. Conf. (VTC-Spring)*, vol. 2, May 2004, pp. 962–965.

[15] F. Bellili and S. Affes, "A low-cost and robust maximum likelihood Doppler spread estimator," in *Proc. IEEE GLOBECOM*, Atlanta, GA, USA, Dec. 2013, pp. 4325–4330.

[16] F. Bellili, R. Meftehi, S. Affes, and A. Stéphenne, "Maximum likelihood SNR estimation of linearly-modulated signals over time-varying flat-fading SIMO channels," *IEEE Trans. Signal Process.*, vol. 63, no. 2, pp. 441–456, Jan. 2015.

[17] M. F. Brejza, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "A high-throughput FPGA architecture for joint source and channel decoding," *IEEE Access*, vol. 5, pp. 2921–2944, 2017.

[18] P. Hailes, L. Xu, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "A flexible FPGA-based quasi-cyclic LDPC decoder," *IEEE Access*, vol. 5, pp. 20965–20984, 2017.

[19] A. Li, P. Hailes, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "1.5 Gbit/s FPGA implementation of a fully-parallel turbo decoder designed for mission-critical machine-type communication applications," *IEEE Access*, vol. 4, pp. 5452–5473, 2016.

[20] X. Cai, M. Zhou, and X. Huang, "Model-based design for software defined radio on an FPGA," *IEEE Access*, vol. 5, pp. 8276–8283, 2017.

[21] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "An end-to-end multi-standard OFDM transceiver architecture using FPGA partial reconfiguration," *IEEE Access*, vol. 5, pp. 21002–21015, 2017.

[22] M. Petrova, A. Achtzehn, and P. Mähönen, "System-oriented communications engineering curriculum: Teaching design concepts with SDR platforms," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 202–209, May 2014.

[23] S. G. Bilén, "Software-defined radio: A new paradigm for integrated curriculum delivery," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 184–193, May 2014.

[24] D. Kuswidiastuti, S. Suwadi, T. Suryani, and D. Elvia, "Implementation and performance analysis of convolution codeon WARP (wireless open access research platform)," *JAVA Int. J. Elect. Electron. Eng.*, vol. 13, no. 1, pp. 1–6, 2016.

[25] Free Software Foundation, Inc. (2009). GNU Radio—The GNU Software Radio. [Online]. Available: http://www.gnu.org/software/gnuradio

[26] S. Donthi and R. L. Haggard, "A survey of dynamically reconfigurable FPGA devices," in *Proc. 35th Southeastern Symp. Syst. Theory*, Mar. 2003, pp. 422–426.

[27] M. Ahmadian, Z. J. Nazari, N. Nakhaee, and Z. Kostic, "Model based design and SDR," in *Proc. 2nd IEE/EURASIP Conf. DSPenabledRadio*, Sep. 2005, pp. 1–8.

[28] BEEcube. (2016). *FPGA Based Rapid Prototyping Platforms for Telecommunications*. [Online]. Available: www.beecube.com/uploads/6/3/4/9/63495763/beecube_brochure_web.pdf

[29] (2014). *Challenges and Solutions in Prototyping 5G Radio Access Network*. [Online]. Available: www.usdatavault.com/library/5gwhitepaper.pdf

[30] Anite. (2008). *Scalable Tool for Radio Channel Emulation EB Propsim F8*. [Online]. Available: www.gigacomp.ch/pdfs/EB_Propsim_F8_Datasheet.pdf

[31] *Study on Scenarios and Requirements for Next Generation Access Technologies, Version 14.0.0*, document (TR) 38.913, 3rd Generation Partnership Project (3GPP), 2016.

[32] G. DeMicheli, *Synthesis and Optimization of Digital Circuits*. New York, NY, USA: McGraw-Hill, 1994.

[33] T. Peng, Y. Zhou, and C. Hu, "An efficient design of FPGA-based sample rate converting filter in software defined radio," in *Proc. Int. Conf. Commun. Technol. (ICCT)*, 2013, pp. 634–638.

[34] Xilinx. (2016). *Divider Generator v5.1, LogiCORE IP Product Guide*. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/_div_gen/v5_1/pg151-div-gen.pdf

[35] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for FPGAs: From prototyping to deployment," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 4, pp. 473–491, Apr. 2011.

[36] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, 1959.

[37] E. Dahlman, S. Parkvall, J. Skold, and P. Beming, *3G Evolution: HSPA and LTE for Mobile Broadband*. New York, NY, USA: Academic, 2010.

[38] M. Pedersoli, A. Vedaldi, and J. Gonzàlez, "A coarse-to-fine approach for fast deformable object detection," in *Proc. IEEE CVPR*, Jun. 2011, pp. 1353–1360.

[39] *MATLAB Fixed-Point Tool*, MathWorks, Natick, MA, USA, 2013.

[40] O. A. Pfänder, R. Nopper, H.-J. Pfleiderer, S. Zhou, and A. Bermak, "Configurable blocks for multi-precision multiplication," in *Proc. IEEE DELTA*, Jan. 2008, pp. 478–481.

[41] W. Yan and H. Chen, "Time-shared fitting method of Doppler parameters and the implementation on FPGA," in *Proc. IET Int. Radar Conf.*, Apr. 2013, pp. 1–5.

[42] A. Page and T. Mohsenin, "An efficient & reconfigurable FPGA and ASIC implementation of a spectral Doppler ultrasound imaging system," in *Proc. IEEE 24th Int. Conf. Appl.-Specific Syst., Architectures Processors*, Jun. 2013, pp. 198–202.

[43] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital Communication*. Berlin, Germany: Springer, 2004.

[44] T. Schenk, *RF Imperfections in High-Rate Wireless Systems: Impact and Digital Compensation*. Berlin, Germany: Springer, 2008.

[45] J. Tubbax, A. Fort, L. V. der Perre, S. Donnay, M. Engels, M. Moonen, and H. D. Man, "Joint compensation of IQ imbalance and frequency offset in OFDM systems," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, vol. 4, Dec. 2003, pp. 2365–2369.

HAITHEM HAGGUI received the M.Sc. degree from the École de technologie supérieure (ÉTS), University of Quebec, Montreal, QC, Canada, in 2012, where he is currently pursuing the Ph.D. degree with the Institut National de la Recherche Scientifique (INRS). His current research interests include FPGA and ASIC implementation, software-defined radio (SDR), digital signal processing, and wireless communications.

SOFIÈNE AFFES (S'95–SM'05) received the Diplôme d'Ingénieur in telecommunications and the Ph.D. degree (Hons.) in signal processing from Télécom ParisTech (ENST), Paris, France, in 1992 and 1995, respectively. He was a Research Associate with INRS, Montreal, QC, Canada, until 1997; an Assistant Professor, until 2000; and an Associate Professor, until 2009. He is currently a Full Professor and the Director of PERWADE, a unique M$4 million research-training program on wireless in Canada involving 27 partners from eight universities and ten industrial organizations. He has been twice a recipient of the Discovery Accelerator Supplement Award from NSERC (2008–2011) and (2013–2016). From 2003 to 2013, he was the Canada Research Chair in wireless communications. Since 2017, he has been holding the Cyrille-Duquet Research Chair in telecommunications. In 2006, 2015, and 2017, he has served as the General Co-Chair or Chair of the 64th IEEE VTC'2006-Fall, the 15th IEEE ICUWB'2015, and the 28th IEEE PIMRC'2017 co-located with the 28th IEEE 5G Summit, respectively, all held in Montreal, QC, Canada. He received the IEEE VTC Chair Recognition Award from the IEEE VTS and the IEEE ICUWB Chair Recognition Certificate from the IEEE MTT-S for exemplary contributions to the success of both events, in 2008 and 2015, respectively. He has previously served as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the *Journal of Electrical and Computer Engineering* (Hindawi), and the *Journal of Wireless Communications and Mobile Computing* (Wiley). He currently serves as a member of the Editorial Board of the *MDPI Sensors Journal* and the Advisory Board of the *MDPI Multidisciplinary Journal Sci*.

FAOUZI BELLILI received the B.Eng. degree (Hons.) in electrical engineering from Tunisia Polytechnic School, in 2007, and the M.Sc. and Ph.D. degrees (Hons.) from the National Institute of Scientific Research (INRS), University of Quebec, Montreal, QC, Canada, in 2009 and 2014, respectively. From 2014 to 2016, he was a Research Associate with INRS-EMT, where he coordinated a major multi-institutional NSERC Collaborative R&D (CRD) project on 5th-Generation (5G) Wireless Access Virtualization Enabling Schemes (5G-WAVES). From 2016 to 2018, he was a Postdoctoral Fellow with the University of Toronto, ON, Canada. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB, Canada. His research focuses on statistical and array signal processing for wireless communications and 5G-enabling technologies. He received the very prestigious NSERC PDF Grant (2017–2018). He was also a recipient of another prestigious PDF Scholarship offered over the same period (but declined) from the Fonds de Recherche du Quebec Nature et Technologies (FRQNT). He was also awarded the INRS Innovation Award for the year 2014/2015, the very prestigious Academic Gold Medal of the Governor General of Canada (2009–2010), and the Excellence Grant of the Director General of INRS (2009–2010). He received the Award of the Best M.Sc. Thesis in INRS-EMT (2009–2010), and twice—for both the M.Sc. and Ph.D. programs—the National Grant of Excellence from the Tunisian Government. In 2011, he received the Merit Scholarship for Foreign Students from the Ministere de l'Education, du Loisir et du Sport (MELS) of Quebec, Canada. He serves regularly as a TPC member for the major IEEE conferences and acts as a Reviewer for many international scientific journals and conferences.