IEEE Open Journal of
Circuits and Systems

Received 18 December 2022; revised 16 March 2023; accepted 18 April 2023. Date of publication 26 April 2023; date of current version 9 May 2023.

Digital Object Identifier 10.1109/OJCAS.2023.3270752

A Sub-mW Cortex-M4 Microcontroller Design for IoT Software-Defined Radios

MATHIEU XHONNEUX (Member, IEEE), JÉRÔME LOUVEAUX (Member, IEEE), AND DAVID BOL (Senior Member, IEEE)

ICTEAM Institute, Université catholique de Louvain, 1348 Louvain-la-Neuve, Belgium

This article was recommended by Associate Editor J. McAllister.

CORRESPONDING AUTHOR: D. BOL (e-mail: david.bol@uclouvain.be)

ABSTRACT We present an Internet-of-Things (IoT) software-defined radio platform based on an ultra low-power microcontroller. Whereas conventional wireless IoT radios often implement a single protocol, we demonstrate that general-purpose microcontrollers running software implementations of wireless physical layers are a promising solution to increase interoperability of IoT devices. Yet, since IoT devices are often energy-constrained, the underlying challenge is to implement the digital signal processing of the radio in software while maintaining an overall very low power consumption. To overcome this problem, we propose an ultra low-power microcontroller architecture with an ARM Cortex-M4 processor for the protocol-specific computations and a hardware digital front-end for the generic signal processing. The proposed architecture has been prototyped in 28nm FDSOI and the physical layers of the well-known LoRa and Sigfox protocols have been implemented in software. Thanks to the efficient hardware/software partitioning and an ultra-low power digital implementation, experimental evaluations of the microcontroller prototype show sub-mW power consumptions (32 – 332 μW) for the digital signal processing of the software-defined radios.

INDEX TERMS Internet of Things, wireless communications, microcontrollers, low-power wide area networks, reconfigurable devices, software defined radio.

I. INTRODUCTION

THE LAST decade witnessed the massive surge of Internet of Things (IoT) devices and services, especially in the industry. The Industrial IoT (IIoT) is an emerging paradigm that aims at increasing the knowledge of existing physical systems (e.g., machines or a production environment) by deploying low-cost, resource-constrained microcontrollers with sensing features. These sensors are wirelessly connected to the cloud, and the data they provide is then used to improve the monitoring and efficiency of industrial processes [1].

The wireless connectivity of the microcontrollers is a key functionality of IIoT applications. In particular, the fact that the wireless devices are usually energy-constrained has led to the development of a new generation of low-power wide-area network (LPWAN) technologies [2]. Since sensors are used across a large variety of processes and systems, no single wireless technology matches the broad range of requirements of all IIoT applications [3]. Several LPWAN protocols hence co-exist together, as each favors a different set of characteristics (payload size, communication range, energy efficiency, cost, . . . ,). Nowadays, three main standards share the LPWAN market: NB-IoT, LoRaWAN and Sigfox [4]. NB-IoT is a cellular standard derived from LTE that uses licensed frequency bands. LoRaWAN and Sigfox are two technologies operating in the unlicensed industrial, scientific and medical (ISM) frequency bands, mainly around 868 MHz in Europe and 915 MHz in North America. LoRaWAN is rolled out by both commercial operators and non-profit organizations, whereas the Sigfox network is managed worldwide by a single company.

Nonetheless, new LPWAN technologies are constantly designed and put on the market, even more with the recent advent of satellite IoT communications [5]. The LPWAN market hence evolves at a higher pace than the deployment of industrial processes, which have much longer life

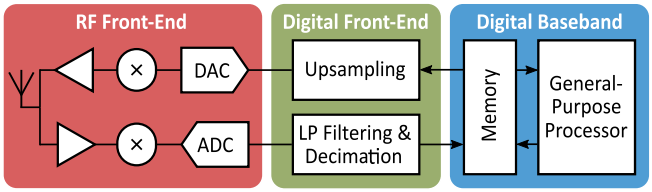

FIGURE 1. Architecture of an IoT software-defined radio with a common sub-GHz RF front-end, a configurable digital front-end (DFE) and a general-purpose processor performing the protocol-specific digital baseband (DBB) computations.

cycles. This raises an issue for the long-term maintenance of IIoT devices and their interoperability with newer wireless technologies and services. As of today, LPWAN radios are often implemented in hardware and tied to a single standard, which limits the reconfigurability of the devices and thus their in-field lifetimes.

Sensors with multiple radio access technologies (multi-RAT) have the potential to enhance the interoperability of IIoT devices. The authors of [3] and [6] suggest combining several hardware LPWAN radios in a single device to benefit from the integration of multiple wireless technologies. This solution however does not allow to support protocols that are yet to come. A more flexible approach consists in the design and deployment of software-defined radios (SDRs) [7], [8]. In an SDR, the RF front-end is designed to be generic while most of the digital baseband (DBB) computations of the physical (PHY) layer are performed in software by a processor (CPU). Such an architecture enables over-the-air updates of the communication protocol, and therefore allows a device to switch from one standard to another during its lifetime, or simply to follow the latest revisions of a protocol.

Yet, the main challenge of SDRs for IIoT sensors is to execute the DBB processing in software with a power consumption that fits in the 1–10 mW budget of LPWAN transceivers [9]. Three different approaches have been identified in the literature to tackle this challenge. In [7], the DBB computations are offloaded to a custom baseband processor with an instruction set specifically designed for LPWAN protocols. In [8], the authors observe that most standards share very similar characteristics (i.e., the use of small bandwidths in sub-GHz bands), and therefore propose an architecture (shown in Fig. 1) with a single common RF front-end, a configurable digital front-end (DFE) that executes the generic baseband operations (e.g., low-pass filtering) in hardware for power savings, and a general-purpose CPU that carries out the protocol-specific computations. They however provide only simulation results of a generic GFSK receiver implemented in software. In a subsequent study using the same architecture [10], the authors extend the RISC-V instruction set architecture (ISA) with custom instructions that implement the complex arithmetic operations used in DBB computations at lower power. Using this ISA extension, they implement in software and simulate parts of the LoRa and Bluetooth Low Energy (BLE) Rx chains. In [11], an SDR platform for experimental IoT communications running on a generic MCU and a reprogrammable field-programmable gate array (FPGA) is presented. The large price (55$), bulky size and important power consumption (around 100 mW in Rx) of the platform are however incompatible with the specifications of cheap, small and low-power IIoT sensors. Although the concept of SDRs for LPWAN protocols originated several years ago, the design and experimental validation of a low-power SDR architecture actually implementing several standards is so far still missing in the literature.

A. CONTRIBUTIONS

In this paper, we demonstrate an SDR transceiver running the LoRaWAN and Sigfox PHY layers on an ultra low-power (ULP) microcontroller unit (MCU), such as encountered in typical IIoT applications. We design an MCU architecture that follows the approach from [8] by offloading the generic DBB computations to a configurable hardware accelerator. The protocol-specific DBB processing is implemented in software on a general-purpose ARM Cortex-M4 CPU. The LPWAN protocol deployed in the MCU can hence be modified by the means of a software update. We prototype our MCU architecture in a 28nm CMOS FDSOI technology and implement in C the PHY layers of LoRaWAN (usually named LoRa) and Sigfox. We validate the functionality of both protocols in an experimental testbed and show that the SDRs attain receiver sensitivities below −120 dBm. Thanks to an efficient hardware/software co-design of the MCU, the signal processing of the SDRs exhibits a sub-mW power consumption. To the best of our knowledge, this work is the first to demonstrate an actual prototype of an ULP SDR with implementations of two commercial IoT protocols.

This work is an extension of [12], which only presented a preliminary version of the LoRaWAN PHY software implementation on the same MCU. In this journal version, we describe in more detail the design of the proposed MCU architecture. In addition to the LoRaWAN SDR, we also present a new implementation of the Sigfox PHY layer. We finally conduct more in-depth experimental evaluations of both SDRs, and we provide a new discussion on the relevance of low-power CPUs with extended ISAs by comparing our results with those of [10].

B. OUTLINE

The remainder of this paper is organized as follows. In Section II, we conduct a hardware/software co-design of an ULP MCU architecture suitable for IIoT SDRs, including the CPU exploration. In Section III, we briefly introduce the PHY layers of LoRa and Sigfox and we then describe how we implement them on our custom MCU architecture. Finally, we perform in Section IV an experimental evaluation of both SDR implementations in a testbed and we discuss their performance.

II. DESIGN OF A MICROCONTROLLER FOR IOT SOFTWARE-DEFINED RADIOS

In this section, we explore several key architectural issues when designing an ULP SDR-capable MCU. We first motivate the choice of the ARM Cortex-M4 as CPU for the digital signal processing (DSP) of LPWAN physical layers. We then explain the hardware implementation of our DFE. The overall architecture of our custom ULP MCU with the different peripherals (memories, DFE, ...) is finally described.

A. LOW-POWER CPU EXPLORATION

In the general architecture presented in Fig. 1, one of the most critical design choices consists in the selection of the general-purpose processor. Indeed, more advanced CPUs require fewer cycles to carry out a given operation, which therefore lowers the minimal frequency at which the CPU needs to run. On the other hand, such processors also require a greater silicon area and consume more energy per cycle compared to low-area and low-power CPUs. The CPU choice hence amounts to a trade-off between the minimum required frequency, the consumed energy per cycle and the occupied silicon area.

For the design of the MCU, we considered three different popular low-power CPUs from ARM: the Cortex-M0, Cortex-M3 and Cortex-M4.[1] These 32-bit processors all share a RISC instruction set and feature a 32-bit three-stage pipeline. The Cortex-M0 features the lowest area by implementing a Von Neumann architecture with a unique memory for both program and data, whereas the Cortex-M3 and M4 occupy more area by relying on a Harvard architecture, with distinct program memory (PMEM) and data memory (DMEM). Moreover, contrary to the other cores, the Cortex-M4 also contains two 16-bit single-instruction multiple-data (SIMD) lanes to accelerate fixed-point DSP computations. These additional SIMD instructions allow the Cortex-M4 to efficiently carry out DSP computations on pairs of Q16 fixed-point numbers. For instance, the `SMUAD` instruction performs two multiplications on two pairs of Q16 numbers and returns their sum in a single cycle. This instruction allows an efficient implementation of the multiplication of two complex 16-bit numbers.

When performing the CPU selection, we conducted benchmarks with two DSP kernels implemented in the ARM CMSIS DSP library: a 256-point complex FFT and a FIR filter with decimation (63 taps, decimation factor of 52). The FFT and FIR filter kernels are used, respectively, in the LoRa and Sigfox SDRs. Both kernels use the Q16 fixed point representation. Fig. 2 shows the number of CPU cycles required for each kernel and the three studied processors. Compared to the Cortex-M0, we observe that the Cortex-M3 requires

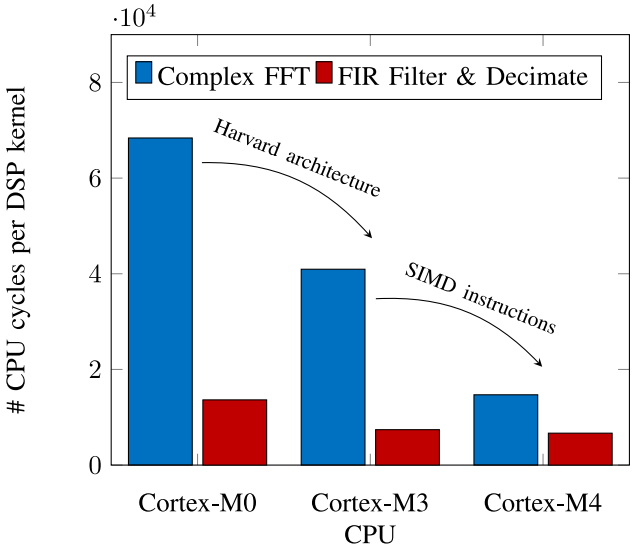

FIGURE 2. CPU cycles required for a Q16 complex 256-point FFT and a FIR filtering and decimation of 260 Q16 complex samples (63 taps, decimation factor of 52) for three low-power ARM Cortex-M CPUs.

40% and 46% less CPU cycles to perform the FFT and the FIR filtering, respectively, thanks to its Harvard architecture. In the Von Neumann architecture, the Cortex-M0 needs to stall for one cycle to retrieve every data word (e.g., a FIR filter coefficient) since the retrievals of the next instructions and the data word cannot be parallelized. The Cortex-M3 and M4 do not suffer from this limitation. Moreover, running the complex FFT and FIR filtering on the Cortex-M4 instead of the M3 further reduces the number of cycles by 64% and 10%, respectively, thanks to SIMD instructions such as `SMUAD`.

This benchmarking illustrates the importance of selecting a CPU with parallel processing features such as SIMD to attain both a real-time and energy-efficient DBB processing. To minimize their power consumption, ULP MCUs often rely on low supply voltages, which also imply low CPU frequencies. As such, ULP MCUs usually exhibit minimum energy points below 50 MHz [13]. Overall, the Harvard architecture and the SIMD DSP instructions of the Cortex-M4 offer a 4.65× speed-up for the complex FFT compared to the M0, at the cost of a relative power consumption only 2.3× higher and a CPU area only 3.5× larger (for a 40LP technology) [14], [15]. For the LoRa Rx baseband chain described in Section III, solely executing the complex 256-point FFT at the Nyquist sampling frequency of 125 kHz already requires a minimum CPU frequency of 33.48 MHz (versus only 7.2 MHz for the M4). Since running the entire LoRa Rx chain on the Cortex-M0 would need high CPU frequencies (> 100 MHz), which would in turn require a higher supply voltage and therefore be energy inefficient, and since the Cortex-M3 and M4 have similar areas while the latter is computationally superior, we opted for the Cortex-M4 in our MCU architecture.

1. There exist more recent Cortex-M CPU architectures. The more recent Cortex-M33 and Cortex-M55 CPUs were not available when we started the design of the MCU. The Cortex-M33 features performance characteristics similar to the Cortex-M4, whereas the Cortex-M55 is a more advanced CPU with the M-Profile Vector Extension. However, these CPUs were not available for academic access when we started the design of the MCU.

TABLE 1. Required ARM Cortex-M4 CPU frequency for the two most intensive DSP operations of the LoRa baseband Rx chain.

Operation	# cycles per sample	Sampling rate	Min. CPU frequency
Q16 FIR Filter & Decimate (16 taps)‡	23.2	2 MHz	46.4 MHz
Q16 256-point FFT	57.4	125 kHz	7.2 MHz

‡ Offloaded to a hardware accelerator in this work.

The authors of [10] draw a similar conclusion by implementing a SIMD DSP extension similar to the one of the Cortex-M4, but for the RISC-V Bk3 core. Thanks to their ISA extension, the number of cycles required to carry out a complex FFT with the Bk3 core is decreased by 55%. This gain is much in line with the one observed between the Cortex-M3 and M4 (64%).

B. HARDWARE OFFLOADING OF RX LOW-PASS FILTERING

As LPWAN protocols use different modulations with various data rates and signal bandwidths, both the sampling frequency of the baseband signal and its low-pass filtering in Rx have to be configurable. In the proposed architecture, we assume that the RF front-end has a single analog low-pass filter with a fixed cut-off frequency. The proper low-pass filtering to the actual bandwidth of the sampled signal must hence by done in the digital domain. A first solution would be to perform the low-pass filtering and decimation in software. Table 1 shows the minimum CPU frequency required to carry out in software each of the two most intensive DSP operations of the Rx LoRa baseband chain implemented in this work: the low-pass FIR filtering with decimation and the complex FFT used to demodulate LoRa symbols. Since the low-pass filtering is performed at a higher frequency than the demodulation (oversampling rate $R = 16$ to allow subsequent timing synchronization), the FIR filter is one order of magnitude more computationally intensive than the FFT and would require a minimum CPU frequency of 46.4 MHz for its sole execution.

To reduce the power consumption of the SDR, we offload instead the generic low-pass filtering to a configurable hardware accelerator. This strategy allows to further scale down the CPU frequency and hence to decrease the power consumption (e.g., the proposed LoRa Rx chain runs at 32 MHz instead of >78 MHz). The design of the hardware accelerator is illustrated in Fig. 3. We implemented a low-area FIR filter with two multipliers and two accumulators, each of these operating on the in-phase and quadrature sample streams in parallel. The number of filter taps is equal to the decimation factor R, which allows carrying out a single multiply-and-accumulate operation per in-phase or quadrature input sample. The weights and the decimation factor of the FIR filter are configurable by the CPU through the

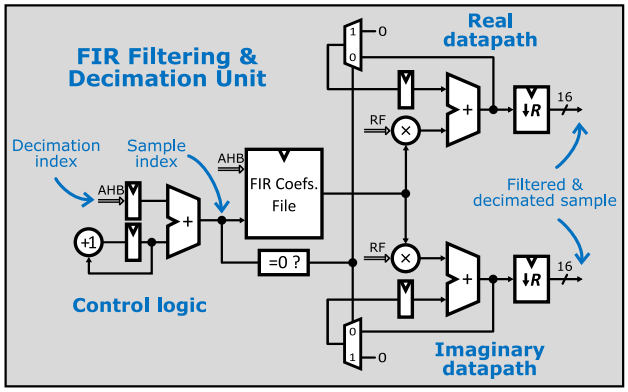

FIGURE 3. Low-area digital implementation of the FIR filtering and decimation unit in the DFE. All registers are clocked with the same clock running at the frequency f_S.

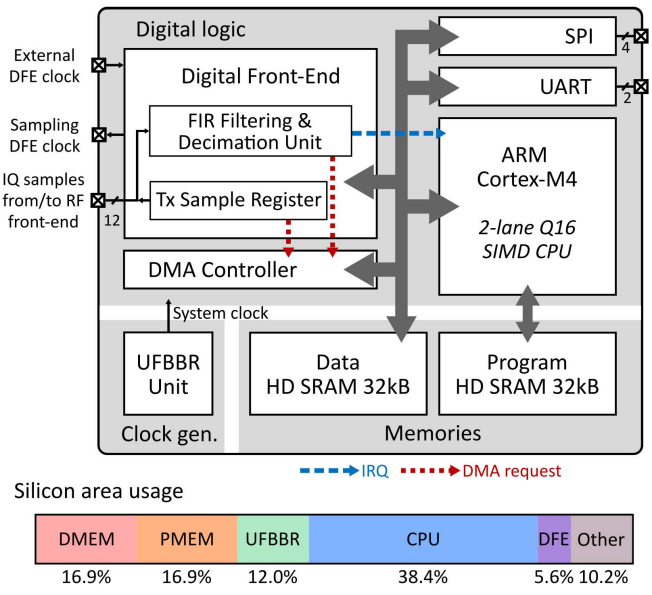

FIGURE 4. Architecture of the proposed SDR-capable SleepRider microcontroller, with the silicon area usage of the underlying blocks.

AHB bus, as well as the decimation index, i.e., the relative input sample offset at which the computation of a new output sample begins. The programming of the decimation index enables a fine-grain time synchronization before the signal is decimated to a lower frequency and demodulated.

C. MICROCONTROLLER ARCHITECTURE WITH DIGITAL FRONT-END

Fig. 4 shows the architecture of the proposed SDR-capable MCU, codenamed SleepRider [13]. The design contains an ARM Cortex-M4 CPU, two 32-kB high-density (HD) SRAM memories, a Direct Memory Access (DMA) controller, a custom digital front-end (DFE), a unified frequency and back-bias regulation (UFBBR) unit and I/O peripherals. Except for the PMEM, the CPU and all other blocks share a single memory bus to exchange data. SleepRider MCU was designed for ULP consumption in 28nm FDSOI with various techniques, including ultra-low supply voltage (0.4V)

for its logic [13]. The UFBBR unit jointly generates the main system clock and the back-bias voltages of the FDSOI transistors to compensate for PVT variations. The frequency of the system and CPU clock is programmable and spans from 12 to 80 MHz.

The ARM Cortex-M4 CPU runs a bare-metal software that implements the protocol-specific DBB processing. In this prototype, the software instructions are stored in the program HD SRAM and are written through JTAG by a debugger at boot time. With the SleepRider MCU prototype, due to the limited size of the SRAM memories, we have to program the MCU with another software to change the protocol. Commercial MCUs with larger non-volatile memories should not suffer from this inconvenience, and should instead allow to switch between protocols on-the-fly. In addition, programmers of IoT end nodes with an LPWAN technology usually seek to communicate as less as possible to save power and avoid interference. During the vast majority of the time, when the MCU is not sending or receiving data, the CPU is free to execute the usual tasks of an end-node (data acquisition from sensors, edge data processing, ...).

To minimize the CPU load and power consumption, the protocol-dependent signal pre-conditioning is offloaded to a configurable DFE. The DFE is a custom hardware accelerator that features several parameters to accommodate IoT protocols with different characteristics (e.g., bandwidth, signaling rates). Our DFE contains the FIR filtering and decimation unit previously described and directly interfaces with the RF front-end and the data memory bus. Its logic uses an external clock as reference sampling clock. The nominal frequency of the DFE clock is 2 MHz,[2] but it can be adapted to match the signaling rate of the targeted PHY layer. Frequency dividers in the DFE further allow to divide the external clock by 2, 4, 8 or 16 to generate the sampling frequency f_S at the RF front-end. Similarly, the weights and the decimation rate R (between 1 and 16) of the FIR filter can also be configured following the targeted waveform specifications.

The DFE contains separate datapaths and control logic for the Tx and Rx chains. In Rx, the block retrieves baseband samples from the RF chain at an oversampled frequency f_S. The oversampled samples are first low-pass filtered using the FIR filter and then decimated to the baseband frequency $\frac{f_S}{R}$. In Tx, the DFE directly transfers samples to the RF front-end without performing upsampling, i.e., the sampling rate is directly equal to f_S. In either Tx or Rx, the digital samples retrieved from or sent to the RF front-end are complex baseband samples coded on 2x12-bit fixed point numbers. Since the CPU contains two 16-bit SIMD lanes, these 12-bit words are extended to (resp. truncated from) 16 bits before entering (resp. leaving) the memory bus.

To further alleviate the CPU load by avoiding costly context switches, the DMA controller is responsible for transferring the 32-bit complex baseband samples between the DMEM and the DFE. The memory transfers are organized as follows. The CPU first defines windows of L complex samples in the DMEM and indicates the length of these windows to the DFE. In Tx, when the CPU has modulated L samples, it configures the DMA to transfer the L 32-bit words to the DFE. The rate of the memory transfers is determined by the baseband frequency $\frac{f_S}{R}$, i.e., the DFE processes one complex sample at a time and only requests a new sample to the DMA controller when it has transmitted the previous one to the RF front-end. Similarly, in Rx, the CPU first configures the DMA controller for L transfers, and the DFE then issues for each new decimated sample a 32-bit transfer request to the controller. When L samples have been transferred, the DFE raises an interrupt request (IRQ) to the CPU, indicating that the window has been transmitted (in Tx) or is ready to be demodulated (in Rx).

2. The RF front-end used in this work has Rx low-pass filters with a minimum cut-off frequency of 0.75 MHz. The sampling frequency of the DFE must hence be above 1.5 MHz to avoid aliasing.

III. SOFTWARE IMPLEMENTATION OF IOT PHYS

Since LoRa and Sigfox use the same sub-GHz ISM spectrum, whereas NB-IoT employs separate licensed frequency bands, both standards are a perfect match for our SDR architecture introduced in Section I. We hence conduct in this section a brief description of the LoRa and Sigfox PHY layers, and we then explain how their associated DBB chains have been implemented on our SDR-capable MCU. Both DBB implementations are coded in C and rely on the ARM CMSIS DSP library, which provides complex DSP routines optimized for the Cortex-M4. All DSP software operations are executed in Q16 fixed-point to benefit from the SIMD instructions of the CPU.

A. LORA: A SPREAD-SPECTRUM SYMMETRIC PHY LAYER

The LoRa PHY layer, contrary to the open LoRaWAN MAC standard, is a patented technology invented in 2010 and now owned by Semtech. The main characteristic of the LoRa PHY is its chirp spread spectrum (CSS) modulation. This spread spectrum technology enables devices that use different *spreading factors* (SFs) to transmit data simultaneously without interfering with each other [16].

1) THE LORA PHY LAYER

LoRa is a symmetric PHY layer, i.e., both the uplink and downlink channels use the same modulation and code. CSS modulated symbols are chirps, i.e., signals whose instantaneous frequency increases linearly and spans the entire bandwidth $B \in \{125, 250, 500\}$ kHz. The SF $\in \{7, \ldots, 12\}$ of the modulation corresponds to the number of bits carried by a symbol and also determines its duration $T = \frac{2^{\mathrm{SF}}}{B}$. Selecting a large spreading factor increases the symbol period, which in turn decreases the data throughput but enhances the communication range.

LoRa symbols are modulated by selecting the initial instantaneous frequency of the chirp, with $N = 2^{\mathrm{SF}}$ possible different initial frequencies. For a symbol

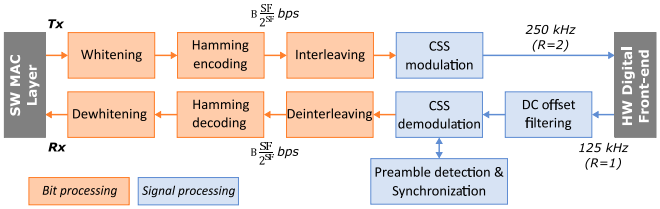

FIGURE 5. Block diagram of the LoRa DBB processing for $B = 125$ kHz.

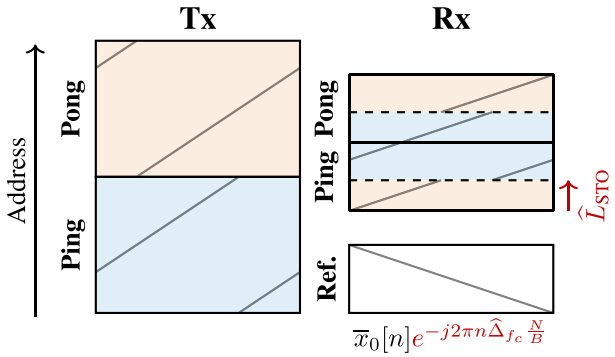

FIGURE 6. Organization of the buffers in DMEM for LoRa Tx ($f_S = 2B$) and Rx (Nyquist rate, after decimation by the DFE) chains. The gray lines in the buffers show the instantaneous frequency of the modulated chirps.

$s \in \{0, \ldots, N-1\}$, the signal starts at an initial frequency of $B(\frac{s}{N} - \frac{1}{2})$. The instantaneous frequency of the modulated signal increases linearly over time, until the moment $t_{fold} = \frac{N-s}{B}$, when the instantaneous frequency is decreased by B (i.e., folded) to keep the signal in the allocated bandwidth [17]

$$x_s(t) = \begin{cases} e^{j2\pi\left(\frac{B}{2T_s}t^2 + B\left(\frac{s}{N}-\frac{1}{2}\right)t\right)} & \text{for } 0 \le t < t_{\text{fold}}, \\ e^{j2\pi\left(\frac{B}{2T_s}t^2 + B\left(\frac{s}{N}-\frac{3}{2}\right)t\right)} & \text{for } t_{\text{fold}} \le t < T_s. \end{cases} \quad (1)$$

The DSP chain of a LoRa transmitter is designed as follows [18]. The input payload bits are first whitened with a predefined pseudo-random sequence. The whitened bits are then coded using a (4, CR) Hamming code (CR $\in \{6, 7, 8\}$). Blocks of SF $\times$ CR coded bits are subsequently interleaved with a diagonal interleaver and modulated following (1) using a Gray mapping. Moreover, to facilitate the synchronization at the receiver, LoRa packets start with a preamble that contains eight repetitions of an *upchirp* (the base waveform $x_0(t)$), followed by two network identifier symbols and 2.25 repetitions of a *downchirp* (the complex conjugate of $x_0(t)$, i.e., a chirp whose frequency decreases over time) [19].

A LoRa receiver implements the following steps to demodulate a signal $x_s[n]$ containing a symbol s sampled at the Nyquist frequency B, where $n \in \{0, \ldots, N-1\}$ [19]. Let $y[n] = x_s[n] + w[n]$ be the received signal when the receiver is perfectly synchronized in time and frequency, where $w[n]$ is additive white Gaussian noise (AWGN). The receiver processes windows of N samples, with each window containing one symbol. For every window, the sampled signal $y[n]$ is first *dechirped*, i.e, multiplied point-wise with an unmodulated downchirp $\overline{x}_0[n] = e^{-j2\pi[\frac{n^2}{2N} - \frac{n}{2}]}$. The point-wise product removes the squared phase component from the chirp but leaves the frequency that carries the modulated symbol s. The dechirped signal $\tilde{y}[n]$ contains a single-tone term of frequency $\frac{s}{N}$ and AWGN such as

$$\tilde{y}[n] = y[n] \cdot \overline{x}_0[n] = e^{j2\pi\frac{sn}{N}} + \tilde{w}[n], \quad (2)$$

where $\tilde{w}[n] = \overline{x}_0[n] \cdot w[n]$. The symbol s is then retrieved by computing the N-point discrete Fourier transform (DFT) of $\tilde{y}[n]$ and selecting the DFT bin of greatest energy, i.e., $\widehat{s} = arg\,max_k |DFT(\tilde{y}[n])|$.

2) LORA SOFTWARE DBB IMPLEMENTATION

We now describe our software implementation of the LoRa PHY layer, which is illustrated in Fig. 5. In the Tx chain, the whitening, Hamming coding, interleaving operations are directly implemented in C using bitwise operations. LoRa symbols are generated using (1) with an oversampling rate $R = 2$ to obtain a spectral representation close to the signal generated by the reference transceiver SX1276 [20]. To transmit symbols in real time, the software relies on two ping-pong buffers. The CPU computes the modulated samples and stores them in one buffer, while the DMA controller transfers the samples from the other buffer to the DFE. After the transmission of an entire buffer, the DFE notifies the CPU with an IRQ and the roles of the ping-pong buffers are exchanged. If the CPU fills one buffer before the transmission of the other buffer, it waits for an IRQ of the DFE. The DFE frequency is set $f_S = 2B$ and the ping-pong buffers hence contain a single LoRa symbol of $2N$ complex samples each, as shown in Fig. 6.

In Rx mode, the DFE frequency is set to $f_S = 16B$ ($R = 16$). The DFE is configured to low-pass filter the received signal from the RF front-end according to the spectral representation of LoRa symbols and to decimate it to the Nyquist frequency B. The DMA transfers the decimated samples from the DFE to the DMEM. The first stage of the Rx software chain is a first-order IIR filter $y[n] = \alpha y[n-1] + \frac{1+\alpha}{2}(x[n] - x[n-1])$ ($\alpha = 0.9$) that removes a residual DC offset from the RF front-end. To perform a real-time demodulation of the received samples, the Rx chain also uses two adjacent ping-pong buffers and a buffer storing $\overline{x}_0[n]$, the conjugate of the base waveform, as shown in Fig. 6. When a ping-pong buffer is full, the DFE notifies the CPU with an IRQ and the processor demodulates it by dechirping the sampled signal with the sequence $\overline{x}_0[n]$ stored in memory and by computing the FFT.

Before demodulating payload symbols, the receiver first needs to detect the preamble of an incoming packet and to synchronize to the transmitter. We showed in a previous work that both the preamble detection and synchronization of a LoRa receiver can be performed using the CSS demodulator previously described [19]. A preamble is detected when the same symbol is repetitively demodulated. The receiver synchronization requires the correction of a carrier frequency

offset (CFO) Δf_c and a sampling time offset (STO) τ. A receiver that is not synchronized in time retrieves windows of N samples containing two consecutive partial chirps instead of a single entire chirp. The STO τ represents the sample-level time offset between the beginning of the window and the first sample of the second chirp in the window. The CFO Δf_c represents the difference of carrier frequency between the transmitter and the receiver.

After the preamble detection, the synchronization algorithm of [19] is applied. The estimates $\hat{\tau}$ and $\widehat{\Delta f_c}$ of the STO and CFO, respectively, are computed on the remainder of the preamble [12], [19]. The CFO is corrected by multiplying the reference waveform $\bar{x}_0[n]$ by $e^{-j2\pi n \widehat{\Delta}_{f_c} \frac{N}{B}}$. The estimated STO is split in integer $\widehat{L}_{\text{STO}}$ and fractional $\widehat{\lambda}_{\text{STO}}$ parts. The integer STO is corrected by treating the two adjacent ping-pong buffers as a single circular buffer of two symbols, as illustrated in Fig. 6. The DSP routines of the CMSIS DSP library required by the demodulation (complex multiplication, FFT and magnitude computation) have been rewritten to properly handle this circular buffer. The processor can hence start the demodulation of a symbol anywhere in the circular buffer, depending on $\widehat{L}_{\text{STO}}$. The fractional STO λ_{STO}, i.e., the intra-sample time offset, is corrected by updating the decimation index in the DFE among the R available offsets. Once the entire packet is demodulated, the received bits are deinterleaved and decoded using a hard-decision Hamming decoder. The decoded payload bits are finally dewhitened and sent to the MAC layer. A more detailed description of the LoRa Rx chain is available in [12].

B. SIGFOX: AN ASYMMETRIC ULTRA-NARROWBAND PHY LAYER

Sigfox is an IoT wireless technology that runs on a single global network licensed by the homonymous company. The protocol achieves low receiver sensitivities thanks to ultra narrowband (UNB) communications, which minimize the noise power at the receiver. Contrary to LoRa, the Sigfox PHY is an asymmetric layer that shifts most of the signal processing complexity (synchronization, decoding,. . .) to the gateway.

1) THE SIGFOX PHY LAYER

To alleviate the load on the sensor, Sigfox relies on different modulation and coding schemes for the uplink and downlink channels. In uplink, the sensor encodes the payload with a convolutional code and transmits data using differential binary phase shift keying (DBPSK) at a fixed bit rate of 100 bps. Up to three repetitions of the same message, with different code polynomials, can be transmitted to bring time diversity at the gateway.

In downlink, the gateway uses a Gaussian frequency shift keying (GFSK) modulation with a bit rate of 600 bps, a frequency deviation of 800 Hz and a bandwidth-time product $BT = 0.5$. A BCH (15, 11) forward error correcting code and a whitening sequence are applied on the payload bits before the modulation stage. Downlink messages are only sent when

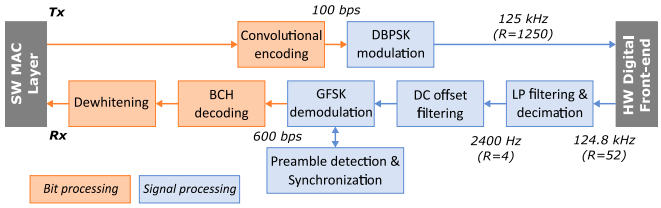

FIGURE 7. Block diagram of the Sigfox DBB processing.

a response is requested by an uplink message. Since the gateway already estimated the CFO of the sensor during the reception of the uplink packet, it also performs a precorrection of this CFO when transmitting the downlink packet. This technique simplifies the synchronization procedure at the sensor, as no CFO correction is needed [21]. GFSK symbols can directly be demodulated using a frequency discriminator, i.e., by determining if the phase of the signal increases or decreases over the symbol period. Let $y[n]$ and $y[n-1]$ be two consecutive samples of a GFSK symbol. A frequency discriminator computes the phase derivative $\phi[n]$ between the samples $y[n]$ and $y[n-1]$

$$\phi[n] = \Re(y[n])\Im(y[n-1]) - \Im(y[n])\Re(y[n-1]). \quad (3)$$

2) SIGFOX SOFTWARE DBB IMPLEMENTATION

Whereas the CSS modulation of LoRa uses large bandwidths, Sigfox relies on UNB signaling with very low data rates. However, due to the limited filtering capabilities of our generic RF front-end, a sampling frequency f_S above 1.5 MHz is required to avoid aliasing. The proposed DFE has hence been designed to operate with a nominal external clock frequency of 2 MHz. In Tx, this clock can further be divided down to $f_S = 125$ kHz. The low signaling rates of the Sigfox PHY compared to the high sampling frequencies f_S of the DFE imply that the Sigfox DBB needs to process samples with a high oversampling rate. Fig. 7 shows our software implementation of this PHY. The Tx chain only comprises two stages: the convolutional encoder and a DBPSK modulator. Using the minimum sampling frequency $f_S = 125$ kHz, the modulation stage generates the DBPSK symbols ($+1$ or -1) with an oversampling rate of 1250 to achieve a baud rate of 100 symbols per second. The modulated samples are transferred to the DFE using the DMA controller, similarly to the LoRa DBB.

In Rx, the baud rate is 600 Hz and the GFSK waveform occupies a narrow bandwidth of approximately 2 kHz. To attain a sampling frequency in the DBB that is a multiple of the 600 Hz symbol rate, the external clock frequency is decreased from the 2 MHz nominal frequency to 1.9968 MHz. The DFE is configured to filter and decimate the baseband signal from the RF front-end to a sampling frequency of 124.8 kHz. In addition to the FIR filter in the DFE, a second low-pass filtering stage in software is finally needed to attain the very low cut-off frequency of 2 kHz. The DMA controller transfers chunks of 1664 complex samples

to the DMEM in ping-pong buffers, which are then again filtered and decimated by the CPU (63 taps, decimation factor of 52). It is however worth noting that filtering the received signal at $f_S = 124.8$ kHz down to 2 kHz using only the CPU would be particularly energy-intensive, as already discussed in Section II. We show in the following section that, thanks to the first filtering and decimation stage in hardware, the second FIR filter can be carried out with a CPU frequency of only 12 MHz, and hence a lower power consumption.

The baseband signal after the second filtering stage has a sampling frequency of 2400 Hz, with four samples per symbol. We therefore adapt the detection rule of (3) to use three consecutive samples per symbol from the four available: $\overline{\phi}[n] = \phi[n+1] + \phi[n]$, which improves the resistance to noise. When a preamble is detected, the STO is estimated by selecting the sample offset $\hat{\tau} \in \{0, 1, 2, 3\}$ that maximizes $|\overline{\phi}[4n + \tau]|$ over successive preamble symbols. The k-th symbol is demodulated by retrieving the sign of the time-aligned frequency discriminator: $\hat{b}_k = sgn\{\overline{\phi}[4k + \hat{\tau}]\}$. Upon the demodulation of all payload symbols, the demodulated bits are finally decoded with a hard BCH decoder and dewhitened.

IV. EXPERIMENTAL PERFORMANCE EVALUATION AND DISCUSSION

We finally demonstrate and evaluate the LoRa and Sigfox SDRs in an experimental testbed. SleepRider MCU [13] is used in the testbed as a prototype of the microcontroller architecture introduced in Section II. In this section, we first describe our testbed. We then present and discuss the energy consumption of the two SDR implementations on SleepRider, as well as the sensitivities in Rx of both SDRs. Based on these results, we finally discuss current limitations of low-power SIMD CPUs for IoT SDRs.

A. EXPERIMENTAL TESTBED WITH SLEEPRIDER MCU

The testbed is shown in Fig. 8. Beside the MCU, the testbed also contains a LimeSDR Myriad-RF1 LMS6002D configurable transceiver board, a low-noise preamplifier, passive attenuators and a NI-USRP 2900. The Myriad-RF1 is used as RF front-end and transfers complex baseband samples from or to the MCU on a 12-bit bus with a sampling clock driven by the DFE [22]. Because the Myriad-RF1 is originally designed for cellular communications with higher signal strengths than those of LoRa and Sigfox, a 30 dB low-noise preamplifier with a 3 dB noise figure is inserted to enhance the receiver gain of the RF front-end. Passive attenuators, with an overall attenuation of 120 dB, are used in Rx to obtain input signals from the USRP with power levels between −130 and −120 dBm.

When testing the LoRa SDR, the USRP is driven by a computer running the LoRa GNU Radio SDR implementation of [18], which is compatible with commercial LoRa radios. The Sigfox SDR has been validated in the testbed by replacing the USRP with the official Sigfox USB test dongle [23]. All experiments use a carrier frequency of 868 MHz.

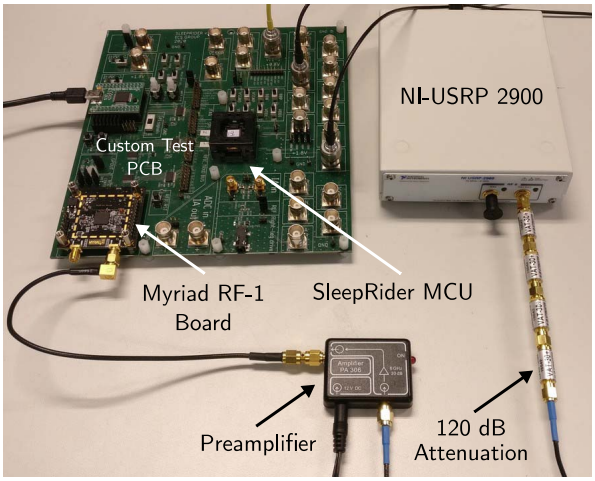

FIGURE 8. Testbed with the SleepRider MCU, the Myriad-RF1 configurable transceiver, a low-noise preamplifier and a NI-USRP 2900.

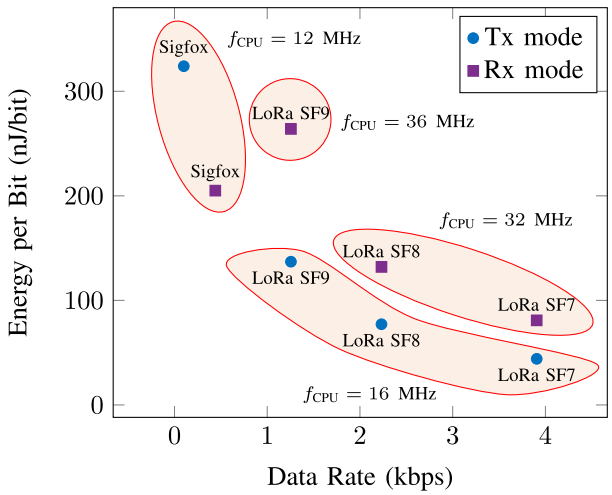

FIGURE 9. Minimum CPU frequency and average energy required by the LoRa and Sigfox SDRs for processing an information bit in the DFE and the DBB.

B. MINIMUM CPU FREQUENCY AND ENERGY CONSUMPTION

We validated the functionality of the Tx and Rx chains for both LoRa and Sigfox in the testbed. For LoRa, we used the bandwidth value $B = 125$ kHz and the coding rate CR = $4/7$ with the SFs 7, 8 and 9. We are unable to run the Rx chain for SFs higher than 9 as the total size of the program and FFT tables exceeds the available space in the PMEM (32 kB). This PMEM size is limited by the die area of our custom chip. Commercial MCUs however usually embed much larger PMEMs (up to 1 MB).

Fig. 9 shows the minimum CPU frequency required to run the SDRs in both Tx and Rx, as well as the average energy per bit transmitted/received by the DFE and DBB. To minimize the energy consumption, we scale down the CPU frequency using the UFBBR (by steps of 4 MHz) for

each mode (Tx or Rx), PHY layer and SF in the case of LoRa. Contrary to commercial transceivers with integrated RF circuits, the RF front-end and the low-noise preamplifier in our setup are off-the-shelf components that are not optimized for low-power operation. The Myriad RF board has a typical power consumption of 725 mW in Rx [22], whereas recent integrated LPWAN transceivers feature power budgets in the 1–10 mW range [9]. Since this work focuses only on the digital signal processing, and not on the RF front-ends that often dominate the power budgets, we ignore the non-optimized low-noise preamplifier and RF front-end from the power measurements and include only the DFE and DBB.

We first discuss the LoRa SDR. The most intensive part of the Tx LoRa DBB is the CSS modulation, which has a linear complexity $\mathcal{O}(N)$ with respect to the symbol period $T = \frac{2^{SF}}{B} = \frac{N}{B}$. Since the number of CPU cycles required to modulate a single sample is independent of the SF, the MCU runs at a minimum CPU frequency of 16 MHz for all SFs, with a power consumption of 172 μW. However, since the data rate of the CSS modulation decreases with the SF, the energy required to encode and modulate one bit of information data increases exponentially with the SF, from 44 nJ/bit for SF 7 to 137 nJ/bit for SF 9. On the contrary, the Rx chain includes the demodulation stage, which has an algorithmic complexity of $\mathcal{O}(N \log N)$ because of the FFT. Incrementing the SF from 7 to 9 hence slightly increases both the CPU frequency (from 32 to 36 MHz) and the power consumption (from 316 to 332 μW). This $\mathcal{O}(N \log N)$ complexity explains why the consumed energy per bit increases with the SF faster in Rx than in Tx.

Regarding the Sigfox SDR, we observe in Fig. 9 that the Tx DBB has a lower energy efficiency with 324 nJ required per information bit, despite the simple signaling DBPSK scheme. Due to practical design limitations of the MCU, the CPU frequency cannot be set under 12 MHz, although the minimum CPU frequency required to modulate DBPSK symbols in real-time would theoretically be 60 kHz. As a consequence, the MCU spends 99.5% of its time idle with an average power consumption of 32 μW, which is only amortized over the low data rate of 100 bps. This power consumption could be reduced if our MCU design was capable of either running at a lower frequency without having its power dominated by static leakage, or by going in deep-sleep mode between two symbols. The same issue applies to the Rx chain to a lesser extent, for which the MCU spends 61% of the time waiting on the next symbol and has a power consumption of 90 μW.

Nevertheless, the power consumption of the DSP in the MCU prototype, between 32 and 332 μW, lie well below the total power usage of LPWAN transceivers. As a comparison, the commercial LoRa SX1276 transceiver consumes 34 mW in Rx [20], and the Sigfox-compatible GFSK receiver from [21] draws 14.5 mW. The power budgets of both transceivers are dominated by the consumption of the RF front-ends. These results indicate that the power needed to run the DSP of our SDRs remains negligible compared to the overall power budget of a LPWAN transceiver.

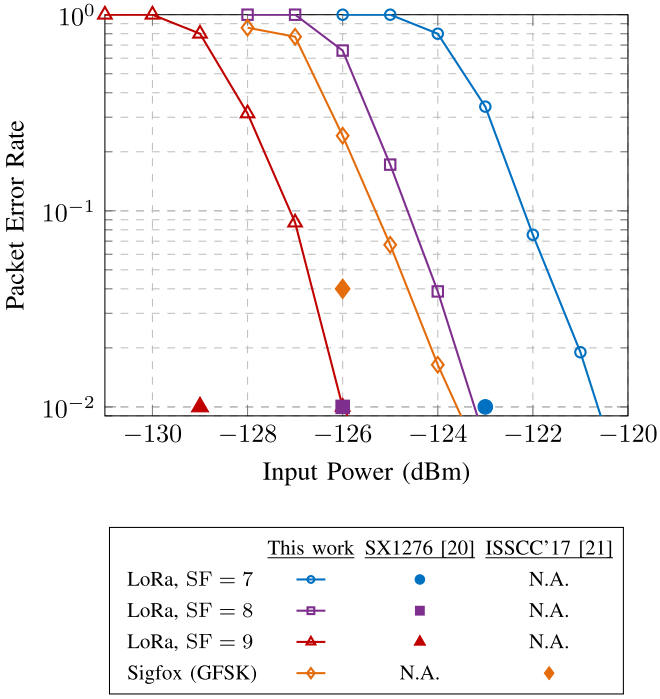

FIGURE 10. Experimental PERs when receiving LoRa (64 payload bytes) and Sigfox (8 payload bytes) packets for different input powers.

C. EVALUATION OF THE SDR RX SENSITIVITIES

We also assess the performance of our SDR Rx chains by measuring the packet error rate (PER) versus the signal power at the input of the preamplifier. In this experiment, we generate different input power values by sweeping the Tx gain of the USRP. To evaluate the sensitivity of our Sigfox radio in Rx, we recorded Sigfox downlink packets from the USB dongle with the USRP, and then replayed these packets with the USRP. 1000 LoRa or downlink Sigfox packets are transmitted per power level. Fig. 10 shows the experimental PERs of both SDRs, as well as the advertised sensitivity levels of the LoRa SX1276 transceiver and the GFSK receiver from [21]. We first observe that our LoRa SDR receiver benefits from a 3 dB spreading gain when the SF is incremented, similarly to the SX1276. Yet, for a target PER of 10^{-2}, our receiver requires about 3 dB higher input power than the SX1276. This 3 dB gap is mainly due to our implementation of the low-pass FIR filter in the DFE which only has 16 taps and hence features non-negligible side lobes outside the signal bandwidth. A hardware implementation of the FIR filter in the DFE with several multipliers could however improve the Rx sensitivities of our SDRs at the cost of a larger area. For the Sigfox SDR, a 4% PER is attained at an input power level of −124.5 dBm, compared to −126 dBm for the GFSK receiver from [21]. Overall, the results presented in Fig. 10 indicate that our MCU prototype

TABLE 2. Power consumptions and minimum CPU frequencies of the SDR implementations of the Sigfox, LoRa and BLE protocols presented in this work and in [10].

	This work *ARM Cortex-M4*△	**Amor *et al.* [10]** *RISC-V Bk3 + SIMD*▲
Included in design	CPU, PMEM, DMEM DFE, UFBBR, periph.	CPU
Implementation characteristics	Post-silicon measurement 28nm FDSOI @ 0.8V	Pre-layout simulation 22nm FDSOI @ 0.8V
Sigfox (2 kHz)	90 µW @ 12 MHz	N.A.
LoRa SF7 (125 kHz)	316 µW @ 32 MHz	110 µW @ 11 MHz
LoRa SF7 (500 kHz)	128 MHz‡	203 µW @ 45 MHz
BLE (1 MHz)	N.A.	380 µW @ 110 MHz

△ The Sigfox and LoRa Rx chains include the synchronization, demodulation and decoding stages. Code compiled with GCC ARM, optimization flag *-O2*.

▲ The LoRa Rx implementation only includes the demodulation stage. Code compiled with LLVM, optimization flag *-O3*.

‡ Theoretical minimum frequency. No power measurement carried out since SleepRider MCU was not tested above 64 MHz.

is capable of receiving packets of two different protocols with near state-of-the-art sensitivity levels.

D. COMPARISON WITH [10] AND LIMITATIONS OF SIMD CPUS

Finally, we compare the previously presented results with the performance of the LoRa and BLE SDRs described in [10]. Based on these results, we discuss current limitations of low-power CPUs with SIMD instructions for IoT SDRs. Table 2 shows the power consumptions and CPU frequencies needed by the different SDRs in Rx. Both studies use the same SDR architecture introduced in Section I, with CPUs featuring 16-bit SIMD instructions for complex arithmetic operations. Contrarily to this work, the results of [10] have been obtained by post-synthesis (pre-layout) simulation with a more advanced 22nm FDSOI technology. Their work covers only on the extension of the RISC-V Bk3 core with SIMD instructions and the software implementations of two demodulation algorithms, i.e., they do not propose a complete MCU implementation and their power consumption results do not take into account the memories, nor the DFE that consume a significant portion of the power.

We first compare the results obtained for the LoRa protocol with SF $= 7$ and $B = 125$ kHz. The extended RISC-V Bk3 core of [10] implements the demodulation of a LoRa symbol with 90 cycles per sample, resulting in a minimum frequency of 11.2 MHz. The same operations on the ARM Cortex-M4 with the CMSIS DSP library require 151 cycles per sample, resulting in a minimum frequency of 18.9 MHz. Yet, the critical path in our LoRa Rx implementation is not the demodulation of a symbol *per se*, but the synchronization stage during the reception of the preamble, during which the estimation of the frequency and time offsets must be carried out in addition to the demodulation of the preamble symbols, as explained in Section III. The synchronization algorithm, which is not implemented in [10], increases the minimum CPU frequency from 18.9 to 32 MHz. Although both implementations are not directly comparable (simulation vs. measurement, different compilers, ...), they still exhibit similar power consumptions and CPU frequencies.

Nonetheless, Table 2 also allows to observe the intuitive relationship between the signal bandwidths and the minimum required CPU frequencies of the SDRs. For moderate bandwidths such as LoRa with $B = 500$ kHz or BLE, both our MCU and the core of [10] need to operate in the 50 – 150 MHz range. This high frequency range is not often possible for commercial low-power MCUs, which typical exhibit minimum energy points below 50 MHz. Moreover, other IoT protocols such as NB-IoT or DECT-2020 NR also rely on moderate bandwidths ($B \geq 200$ kHz), but with more complex digital basebands than those of LoRa and Sigfox [24], [25]. These observations suggest that, despite the effectiveness of the proposed SDR architecture, low-power CPUs featuring only SIMD DSP instructions may not be powerful enough to implement a wide variety of IoT protocols. The recent introductions of ISA vector extensions for low-power CPUs, such as ARM's M-Profile Vector Extension or the RISC-V Vector extension, however open up new possibilities in this regard [26], [27]. For instance, an implementation of a generalized frequency division multiplexing (GFDM) demodulator with the RISC-V Vector extension demonstrated a speedup of up to 60 times compared to the scalar base case [28].

V. CONCLUSION

Long-term maintenance of IoT devices in industrial systems will be an important challenge to keep the operational expenses of IIoT applications low. In particular, the prevailing hardware architecture of IoT radios, often tied to a single protocol, strongly weakens the overall interoperability of the sensors. To mitigate this issue, we designed in this work an ultra low-power microcontroller architecture suitable for a software-defined radio implementation of LPWAN protocols. The proposed architecture features (a) a general-purpose low-power processor with SIMD DSP instructions for the protocol-specific computations (b) a hardware DFE for the generic signal processing and (c) an ultra-low power digital implementation with low supply voltage and frequency scaling. We implemented in software the physical layers of the Sigfox and LoRa standards, and we validated both implementations in a testbed with an MCU prototype in 28nm CMOS FDSOI. Experimental measurements show that both SDRs reach sensitivities below −120 dBm in Rx. Even more so, our prototype performs the digital signal processing of the two protocols with a power consumption between 32 and 332 μW, thus demonstrating that an SDR can fit in the 1–10 mW power budget of a LPWAN transceiver. Future works should investigate the usage of low-power CPUs with vector ISA extensions to enable real-time implementations of more complex IoT protocols than Sigfox and LoRa.

REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[2] W. Mao, Z. Zhao, Z. Chang, G. Min, and W. Gao, "Energy-efficient Industrial Internet of Things: Overview and open issues," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7225–7237, Nov. 2021.

[3] G. Leenders, G. Callebaut, G. Ottoy, L. Van der Perre, and L. De Strycker, "Multi-RAT for IoT: The potential in combining LoRaWAN and NB-IoT," *IEEE Commun. Mag.*, vol. 59, no. 12, pp. 98–104, Dec. 2021.

[4] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Exp.*, vol. 5, no. 1, pp. 1–7, 2019.

[5] M. Centenaro, C. E. Costa, F. Granelli, C. Sacchi, and L. Vangelista, "A survey on technologies, standards and open challenges in satellite IoT," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1693–1720, 3rd Quart., 2021.

[6] K. Mikhaylov et al., "Multi-RAT LPWAN in smart cities: Trial of LoraWAN and NB-IoT integration," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.

[7] Y. Chen, S. Lu, H.-S. Kim, D. Blaauw, R. G. Dreslinski, and T. Mudge, "A low power software-defined-radio baseband processor for the Internet of Things," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2016, pp. 40–51.

[8] H. B. Amor and C. Bernier, "Software-hardware co-design of multi-standard digital baseband processor for IoT," in *Proc. IEEE DATE Conf.*, 2019, pp. 646–649.

[9] D. D. Wentzloff, A. Alghaihab, and J. Im, "Ultra-low power receivers for IoT applications: A review," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, 2020, pp. 1–8.

[10] H. B. Amor, C. Bernier, and Z. Přikryl, "A RISC-V ISA extension for ultra-low power IoT wireless signal processing," *IEEE Trans. Comput.*, vol. 71, no. 4, pp. 766–778, Apr. 2022.

[11] M. Hessar, A. Najafi, V. Iyer, and S. Gollakota, "TinySDR: Low-power SDR platform for over-the-air programmable IoT testbeds," in *Proc. 17th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2020, pp. 1031–1046.

[12] M. Xhonneux, J. Louveaux, and D. Bol, "Implementing a LoRa software-defined radio on a general-purpose ULP microcontroller," in *Proc. IEEE Workshop Signal Process. Syst.*, 2021, pp. 105–110.

[13] R. Dekimpe, M. Schramme, M. Lefebvre, and D. Bol, "SleepRider: A 5.5μW/MHz cortex-M4 MCU in 28nm FD-SOI with ULP SRAM, biomedical AFE and fully-integrated power, clock and back-bias management," in *Proc. IEEE Symp. VLSI Circuits*, 2021, pp. 1–2.

[14] "Cortex-M0 characteristics." ARM. Accessed: Mar. 15, 2023. [Online]. Available: https://developer.arm.com/Processors/Cortex-M0

[15] "cortex-M4 characteristics." ARM. Accessed: Mar. 15, 2023. [Online]. Available: https://developer.arm.com/Processors/Cortex-M4

[16] B. Reynders and S. Pollin, "Chirp spread spectrum as a modulation technique for long range communication," in *Proc. IEEE Symp. Commun. Veh. Technol. (SCVT)*, 2016, pp. 1–5.

[17] M. Chiani and A. Elzanaty, "On the LoRa modulation for IoT: Waveform properties and spectral analysis," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8463–8470, Oct. 2019.

[18] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming, and A. Burg, "An open-source LoRa physical layer prototype on GNU radio," in *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun.*, 2020, pp. 1–5.

[19] M. Xhonneux, O. Afisiadis, D. Bol, and J. Louveaux, "A low-complexity LoRa Synchronization algorithm robust to sampling time offsets," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3756–3769, Mar. 2022.

[20] *SX1276/77/78/79: 137 MHz to 1020 MHz Low Power Long Range Transceiver, Rev. 5*, Semtech, Camarillo, CA, USA, Aug. 2016.

[21] D. Lachartre et al., "7.5 A TCXO-less 100Hz-minimum-bandwidth transceiver for ultra-narrow-band sub-GHz IoT cellular networks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2017, pp. 134–135.

[22] *LMS6002D: Multi-Band Multi-Standard Transceiver With Integrated Dual DACs and ADCs, Rev. 1.1.0*, Lime Microsyst., Guildford, U.K., Mar. 2012.

[23] *Sigfox SDR-USB100M SDR Dongle*, Sigfox, Labege, France, Jun. 2018.

[24] M. Kanj, V. Savaux, and M. Le Guen, "A tutorial on NB-IoT physical layer design," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2408–2446, 4th Quart., 2020.

[25] R. Kovalchukov et al., "DECT-2020 new radio: The next step toward 5G massive machine-type communications," *IEEE Commun. Mag.*, vol. 60, no. 6, pp. 58–64, Jun. 2022.

[26] "RISC-V-SPEC vector extension, version 1.0." RISC-V International. [Online]. Available: https://github.com/riscv/riscv-v-spec

[27] "Whitepaper: Introduction to the Armv8.1-M." ARM. [Online]. Available: https://www.arm.com/resources/white-paper/intro-armv8-1-m-architecture

[28] V. Razilov, E. Matúš, and G. Fettweis, "Communications signal processing using RISC-V vector extension," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, 2022, pp. 690–695.

MATHIEU XHONNEUX (Member, IEEE) received the M.Sc. degree in electrical engineering from the Université catholique de Louvain (UCLouvain), Louvain-la-Neuve, Belgium, in 2018, and the Ph.D. degree in engineering science from UCLouvain in 2022, under the supervision of Prof. J. Louveaux and Prof. D. Bol.

His research interests include hardware/software co-design of signal processing systems and wireless communications for the Internet of Things.

JÉRÔME LOUVEAUX (Member, IEEE) received the Electrical Engineering and Ph.D. degrees from the Université catholique de Louvain (UCLouvain), Louvain-la-Neuve, Belgium, in 1996 and 2000, respectively.

From 2000 to 2001, he was a Visiting Scholar with the Electrical Engineering Department, Stanford University, Stanford, CA, USA. From 2004 to 2005, he was a Postdoctoral Researcher with the Delft University of Technology, The Netherlands. Since 2006, he has been a Professor with the ICTEAM Institute, UCLouvain. His research interests are in signal processing for digital communications, and in particular: multicarrier modulations, xDSL systems, resource allocation, synchronization, and estimation. He was a co-recipient of the "Prix biennal Siemens 2000" for a contribution on filter-bank-based multicarrier transmission and the "Prix Scientifique Alcatel 2005" for a contribution in the field of powerline communications.

DAVID BOL (Senior Member, IEEE) received the Ph.D. degree in engineering science from the Université catholique de Louvain (UCLouvain), Louvain-la-Neuve, Belgium, in 2008.

He is currently a Professor with UCLouvain, where he leads the Electronic Circuits and Systems Group focused on ultra-low-power design of integrated circuits for environmental and biomedical IoT applications, including computing, power management, sensing, and wireless communications with a holistic focus on environmental sustainability. He is actively engaged in a social-ecological transition in the field of ICT research with a post-growth approach. He has authored or coauthored more than 150 technical papers and conference contributions and holds three delivered patents.

Prof. Bol (co-)received four Best Paper/Poster/Design Awards in IEEE Conferences (ICCD 2008, SOI Conference 2008, FTFC 2014, and ISCAS 2020) and was the Ph.D. Supervisor of Charlotte Frenkel who received the 2021 Nokia Bell Scientific Award for her Ph.D.