LOLA SDR: Low power low latency software defined radio for broadcast audio applications

Olivier Etrillard, Robin Gerzaguet, Laurent Feichter, Malo Mabon, Antoine Courtay, Olivier Berder

LOLA SDR: Low power low latency software defined radio for broadcast audio applications

Olivier ETRILLARD*, Robin GERZAGUET†, Laurent FEICHTER*,
Malo MABON†, Antoine COURTAY†, Olivier BERDER†.
* Feichter Electronics, surname.name@feichter-electronics.com
† Univ Rennes, CNRS, IRISA, surname.name@irisa.fr

***Abstract*—This brief proposes a new low power, low latency and low cost reconfigurable architecture for software defined radio. Due to their flexibility and reconfigurability, software defined radios are now massively used as wideband transceivers, channel sounders or network gateways. However, they often struggle to meet the desired requirements in terms of energy consumption and throughput. In this brief, we present a new architecture capable of tackling these challenges, by combining an off-the-shelf generic radio component with a low power microcontroller associated to a Fourier transform coprocessor. To prove the benefit of our approach, after describing the key assets of the architecture, we derive a complete physical layer dedicated to audio broadcast applications. This chain is capable of streaming High Definition audio stream in real time with low power (437 mW) and very low latency (854 μs). We show that our processing chain can be flawlessly run on our architecture paving the way for larger adoption of a new generation of low power low latency software defined radio architectures.**

***Index Terms*—Software Defined Radio, microcontroller, audio processing, low power, low latency**

I. Introduction

Since the first definition of Software Defined Radio (SDR) done by Mitola [1] thirty years ago, many studies carried in various domains show the benefits and the numerous challenges this topic has to tackle. The initial purpose was quite simple: shift as many parts as possible into digital domain and limit the Radio Frequency (RF) analog parts to the antenna and the analog-to-digital converters. If nowadays the myth of a pure digital architecture fizzled out, the adoption of the SDR has been massive and the scope of its use has largely grown.

Many SDR architectures have been proposed: some based on General Purpose Processors (GPP), Digital Signal Processors (DSP) or on specific hardware such as Field Programmable Gate Arrays (FPGA) [2]. More recently, the advances in hardware integration and efficient software methodologies [3] allow the emergence of embedded SDR architectures based on System on Chip (SoC) that combines hardware and software computational resources. This paves the way for high performance SDR applications such as wideband channel sounding, reconfigurable cellular base stations, etc.

SDR are highly appealing for low power scenarios, especially for Internet Of Things (IoT) deployment. Indeed, the low cost and the reconfigurability of these devices are key assets for volume deployment, easy maintenance and extensive reuse. However, as SDR are wideband receivers, use generic components and are often piloted by greedy processing units, it seems illusional to use them as end-devices. Hence, most of SDR integrations for IoT scenarios have used the flexible radios on the gateway side where energy constraints are greatly alleviated [4]. Standalone embedded SDRs are based on SoC [5], [6] or custom multiprocessor architectures based on Network On Chips (NoC) [7]. If it allows portability and flexibility, it also leads to energy consumption several orders of magnitude higher than the requirements for IoT end-devices [8]. On the other side, there are also very cheap SDR such as RTL-SDR [9], but these devices have limited bandwidth, limited targetable carrier frequencies, and require processing to be done on an external computing unit.

We strongly believe that the recent advances in very low power hardware architectures, with limited but efficient embedded computational resources, will change the game. Some preliminary works have tended to prove that SDR can be used as low cost end-device. TinySDR [10] is based on a low power FPGA and is dedicated to low power wide area transmissions. Marmote is an FPGA based SDR [11] focuses on protocol study. The SDR presented in [12] addresses satellite communications and embeds an FPGA. These proposals are promising but however fail in two key aspects: i) the reconfigurability is limited as processing is only done through hardware description (e.g. FPGA) instead of a software approach ii) and the cost is not sufficiently reduced and dominated by the FGPA. Very recently, a similar architecture based on microcontroller has been proposed for LORA communications [13]. The computational complexity of this architecture is however very limited and can only address low datarate applications.

This is the reason we introduce LOLA SDR for Low pOwer Low lAtency reconfigurable Software Defined Radio. Leveraging generic radio components and efficient low power microcontroller (MCU), this architecture is capable of addressing low power and low latency while being able to run flawlessly in real time a complex audio broadcast processing chain that follows High Definition (HD audio) requirements. The core contribution of the brief is the proposal of the architecture and how the different blocks are mapped together. The second contribution of the brief is related to the physical layer chain and how we fit the processing chain with the architecture. The final contribution is a show-off where we demonstrate that our platform can run flawlessly our application in real time, with a very low latency and with a reduced computational power.

The brief is divided into four main parts. In Section II, we introduce the proposed SDR architecture and its key

components and justify why it is suitable for low power flexible applications. In Section III, we describe the proposed processing chain dedicated to audio broadcast and justify the technical choices and parameters. Section IV assesses the key performance indicators that are the latency, the power consumption and demonstrate the viability of our approach through trials and a comparison with a Zynq based implementation. Section V eventually draws some conclusions.

II. Proposed architecture

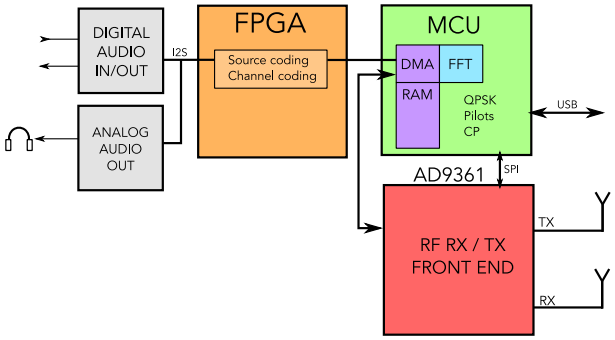

Fig. 1. LOLA SDR architecture

The proposed architecture for the LOLA SDR is depicted on Figure 1 and is composed of several modules described hereafter:

- An agile front-end receiver (depicted in red) is used for transmitting and receiving signals. The stages are composed of wideband configurable analog to digital converters (and digital to analog converters), band-filters, power amplifiers and antennas. Depending on the target applications, large tunable RF boards or specific front-ends can be chosen. For the former, it will increase the flexibility and the agility (more targetable frequency bands) at the price of increased cost, energy consumption and slight degradation of the noise figure. For the latter, the specialization of the SDR will be enhanced, leading to potential better noise figure, reduced cost and reduced energy consumption which is interesting for some IoT scenarios where only a few bands can be targeted.
- A microcontroller unit (MCU), depicted in green is the core processing unit of the LOLA SDR. This is a strong difference with classical processing units proposed in the literature and we strongly believe that using MCU offers several advantages to address low power efficient SDR. Some modern MCU have two cores that can be used as independent processing units. Besides, their energy consumption is magnitude lower than the one of GPP and the numerous interfaces can be used to map external devices, peripherals or coprocessors. Finally, their clock speed is on the same order of magnitude as the one required by modern SDR applications.
- The specific coprocessor is depicted in blue. Some modern MCU are built with specific hardware components such as encryption engines or Fast Fourier Transform (FFT). Using an MCU with coprocessors offers two key advantages: the operations are performed with high efficiency and the MCU can perform other tasks during the coprocessor use. In our LOLA show-off (see Section IV) we have used an MCU with an FFT coprocessor which greatly alleviates the receiver timing constraints. This is also a strong novelty of our architecture.
- The RAM (in purple) is located in the MCU. The Direct Memory Access (DMA) used in LOLA architecture aims to ease the sample acquisition by the radio front end that can be then processed by the MCU or its coprocessors.
- A small FPGA is depicted in orange and has two objectives: the first is to provide glue logic for the different components and interfaces. Depending on the size of the FPGA, some coprocessors can also be synthesized on the FPGA and mapped on the MCU. In our example, we have used a tiny FPGA on which we have synthesized three small blocks of channel coding (see Section III). The use of FPGA is a key enabler to specialize our architecture through specific hardware description. The use of a small FPGA still ensures low cost and low energy consumption.
- Additional analog and digital audio connectivity with I2S interface is also present. A USB link is also used for programming purposes, and to obtain performance feedback (see Section IV).

The proposed architecture is built to have a promising trade-off between simplicity and low cost of the modules with still computational capacity illustrated by the double core MCU and the coprocessors. We detail in the next section an audio broadcast processing chain that uses this architecture and which is capable of meeting the HD audio requirements.

TABLE I
Defined parameters, value and associated type. CS stands for Complex Symbols

Definition	Name	Value
Audio sampling frequency	F_a	96 kHz
Audio word size	n_a	24 bits
Number of audio words per packet	N_a	10
Hamming FEC input size	H_i	4 bits
Hamming FEC output size	H_o	7 bits
Bits per symbol	n	2
FFT size	N_{FFT}	512 CS
CP size	N_{CP}	36 CS

III. Audio broadcast processing chain

A. Physical layer properties and parameters

The processing chain is built in order to ensure low power, low latency and high resilience of the audio processing stack. The proposed physical layer starts with an HD audio stream and uses Orthogonal Frequency Division Multiplexing (OFDM) as it allows to perform the channel equalization in frequency domain.

The proposed chain depends on several parameters that can be split into two categories: the *defined* parameters (see Table I) and the *obtained* parameters. For the former, the choice is either technological (for instance based on audio stack) or practical (implementation tricks). For the latter, the values are derived from the *defined* parameters:

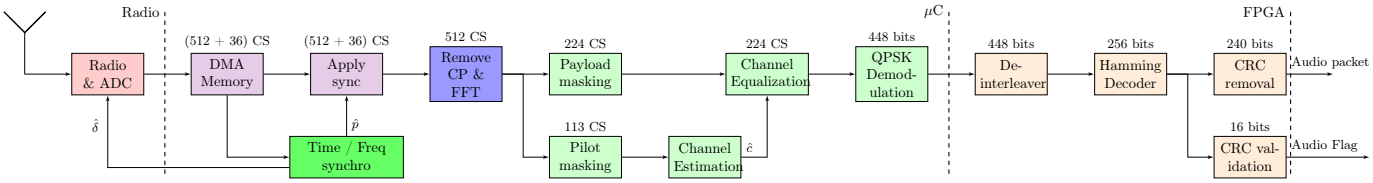

Fig. 2. Proposed receiver chain. CS stands for complex symbols

- The audio rate is 2.304 Mbits/s and is the product of the audio frequency and audio word size ($F_a \times n_a$)
- The packet encapsulation aims to append a Cyclic Redundancy Check (CRC) to ensure the validity at the receiver side of a group of $N_a = 10$ audio words. The audio packet size is 240 bits ($n_a \times N_a$)
- The CRC size is computed to have the output of the CRC as a power of 2 (for efficient memory store). The desired output size is $2^{(1+\lfloor \log_2(N_a \times n_a) \rfloor)}$, the next closest power of 2 based on the audio packet size. It leads to a CRC size of $256 - n_a \times N_a$ = 16. A generator polynomial 0x8005 is used.
- The output size of the CRC encoder is thus 256 bits ($2^{(1+\lfloor \log_2(n_a \times N_a) \rfloor)}$).
- We apply a Hamming (7, 4) encoder to protect the audio data. The output of the encoder is then interleaved to enhance error resistance against burst of errors that can typically happen in case of a strong fading channel. The output size of the Hamming encoder is defined as $2^{(1+\lfloor \log_2(n_a \times N_a) \rfloor)} \times \frac{H_o}{H_i} = 448 = N_o$. The rate of the output of the Forward Error Correcting (FEC) is $= F_a \times \frac{448}{240} = 4.3008$ MBits/s.
- The number of payload subcarriers is $\frac{N_o}{n} = 224$. The number of allocated subcarriers (payload + pilots) is set to have 65% of the spectral allocation (similar ratio as it is done in Long Term Evolution [14]), leading to a total number of 336. Based on this the number of pilot subcarriers is $336 - 224 = 112$. In practice, to be sure that the last subcarrier on the upper corner is a pilot one (and to ease the channel interpolation) one pilot is added at the corner. We thus have 113 pilots and 337 total allocated subcarriers.
- The binary encoded data is then modulated with a Quadrature Phase Shift Keying (QPSK) modulator that is then mapped in the frequency domain. The rate of the output of the QPSK mapper (complex samples) is $F_a \times \frac{224}{240} = 2.1504$ MS/s.
- We apply an IFFT on this buffer using the coprocessor. The input rate of the IFFT is equal to $F_a \times \frac{N_{FFT}}{(n_a \times N_a)} = 4.915$ MS/s. It means that an IFFT should be done for each 240 bits of data, so every $\frac{n_a \times N_a}{Fa} = 104.16$ μs.
- To ensure channel circularity, we insert a Cyclic Prefix (CP) of a duration of $6.9\mu s$. The output rate of the transmitter is then calculated as $F_e = F_a \times \frac{N_{FFT}+N_{CP}}{(n_a \times N_a)} = 5.2608$ MS/s

The processing blocks have been carefully chosen as they offer appealing trade-off between obtained performance (that match the desired indicators) and their relative simplicity: the OFDM waveform uses the coprocessor while the FEC and the CRC can be easily implemented on the FPGA. The choice of QPSK, albeit offering a low spectral efficiency, allows implementation tricks on the MCU side that strongly alleviate the timing constraints and ensures strong noise resistance.

B. Receiver side

To receive and decode the audio stream, the processing is done in two steps: i) an initialization phase with fine time/frequency synchronization and then ii) the decoding phase where the audio stream is decoded in real time with low latency.

The initialization step is computationally intensive as it aims to finely synchronize the receiver in time and frequency domains. This synchronization is done with a correlation between the received buffer in time domain and a pre-calculated sequence. This sequence $s_p[n]$ is defined as the Inverse Fourier Transform of the scattered pilots (set to 1) mapped in the frequency domain.

$$s_p[n] = \sum_{p \in \mathcal{S}_p} e^{\frac{2j\pi p \times n}{N_{FFT}}}, \tag{1}$$

where $\mathcal{S}_p$ denotes the indexes of pilot subcarriers. The synchronisation index $\hat{p}$ is obtained from the maximum of $\Gamma_{d,s_p}[n]$, the correlation between the received sequence $d[n]$ and the pilot sequence $s_p[n]$. The phase of this maximum gives the Carrier Frequency Offset (CFO) $\hat{\delta}$ as depicted in Eq. 2.

$$\hat{p} = \text{Argmax}_n \left(\Gamma_{d,s_p}[n] \right), \hat{\delta} = F_e \times \frac{\langle \Gamma_{d,s_p}[\hat{p}] \rangle}{2\pi N_{FFT}} \tag{2}$$

In a second step the CFO estimation $\hat{\delta}$ is used to tune the receiver local oscillator. The low residual carrier frequency offset is handled in the channel equalization step.

We depict the receiver blocks of the decoding part on Figure 2. Each color corresponds to a different computational domain and follows the same pattern as Figure 1. The purpose of the chain to be able to track the multipath channel variations and to monitor the performance with the CRC check while efficiency uses all the hardware and software resources. For each received symbol, we will have to provide the audio packet (240 bits) and one CRC flag. In purple are depicted the Direct Memory Access (DMA) mechanics: the DMA takes sample from the RF front-end and align the buffer with the estimated delay $\hat{p}$. As the time and frequency localization of the received symbols may slightly vary, the second core of the microcontroller is dedicated to time/frequency lock as depicted in dark green. A correlation is computed on samples at the neighborhood of the previously locked time synchronization index $\hat{p}$. If a mismatch is detected, a retro-action is done on the DMA to

TABLE II
COMPARISON BETWEEN LOLA SDR AND USRP E310

	RF Board	Processing unit	Frequency (MHz)	Operating System	PER @-20dB	Rx Energy consumption
LOLA SDR	AD-9361	LPC55S69	150 MHz	Bare metal	3.9e-3	437 mW
USRP E310	AD-9361	Zynq 7010	866 MHz	Yocto Linux	4e-3	3.25 W

TABLE III
SUMMARY OF USED FPGA RESOURCES

	Tx Resources	Tx %	Rx Resources	Rx %
Registers	488/2352	21%	632/2352	27%
Slices	413/1056	39%	513/1056	49%
LUTs	817/2112	39%	1011 / 2112	48%

keep the received buffer perfectly synchronized. In blue is depicted the OFDM demodulator on the FFT coprocessor. In green, the main processing steps done on the first core of the MCU. First, the data in frequency domain is separated between zeros (dropped), pilots and data. The channel is then estimated directly from the pilots (as their value in frequency domain have been set to 1). The equalization and the QPSK demodulation are performed in a joint manner: for each payload subcarrier, the bits after decision are identical to the most significant bits of the output of the complex channel estimation $\hat{c}$. It also allows us to pack the bits after decision to reduce the memory footprint. The de-interleaver is done on the FPGA as it only corresponds to flip in bits ordering. The hamming74 decoder is also implemented on the FPGA as it corresponds to simple `xor` operations, such as the CRC check, implemented in a parallel approach [15]. These are depicted in orange on Figure 2. The last step corresponds to the audio rendering with I2S interface and the jack output.

IV. BENCHMARKS AND EXPERIMENTS

A. Board synthesis discussion

For our show-off, we have chosen to use a wideband radio front-end, the AD9361 from Analog Devices as it has a wide carrier frequency range. It also allows us to have a very flexible (in both carrier and sampling frequencies) and to be able to configure all the digital stages (automatic gain control policy, digital filter coefficients, etc). We have integrated a Lattice Semiconductor FPGA MachXO, a 26 dBm power amplifier SE5004L from Skyworks and a dual Cortex-M33 microcontroller from NXP : the LPC55S69. This $4 MCU, to be compared to a $50 Zynq (see comparison in Table II), integrates an FFT coprocessor.

The board use the proposed processing chain at a carrier frequency of 5.5 GHz as it is a large licence free band with less traffic. The power supply is provided by an external battery of 37Wh. We depict on Table IV-A a summary of the FPGA resources used in terms of number of registers, slices and Look Up Table (LUT). It is shown that only a part of the FPGA is used with half of the resources still available and with slightly more resources used on the Rx side.

B. Measurements in controlled environment

We validated the technical choices by measurements in an anechoic chamber as depicted in Figure 3. We put the transmitter and the receiver at a distance of 3.8 meters. At the transmitter side, we used a variable attenuator HMC624 from Analog Devices. We consider three scenarios: Line Of Sight (LOS) and two scenarios with a plate to have constructive (Plate-1) and destructive (Plate-2) interference. We compute on Figure 4, the Packet Error Rate versus the attenuation level. It demonstrates the good performance of the reception stage and also proves the capacity of our receiver to synchronize and detect the OFDM symbol even at low signal-to-noise ratio.

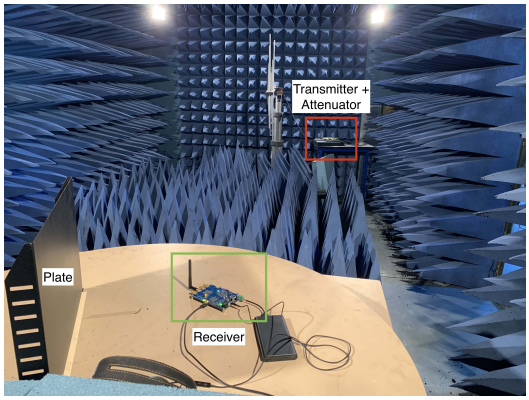

Fig. 3. Setup for validation in the anechoic chamber

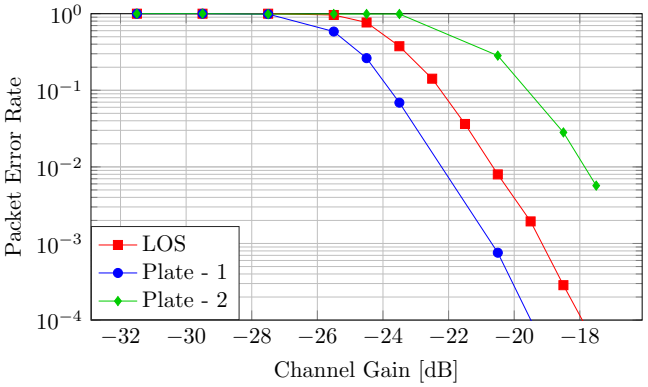

Fig. 4. Packet Error Rate versus attenuation for the three scenarios.

C. Energy consumption and latency

We expose in Table IV the latency of the Tx and Rx stacks. It shows that the complete latency is 854 μs and is equally shared between Tx and Rx. The latencies are function of the audio word duration (10.4 μs) and the audio packet duration (104.1μs).

In table V, we derive the power consumption of the Rx chain as most of the processing is done on the Rx side. The consumption is split among the different processing parts. The overall consumption of the LOLA SDR Rx part is 437 mW which is magnitude lower than the Zynq based architecture.

TABLE IV
LATENCY EVALUATION

Tx part: 427 μs	
I2S Input	10.41 μs
From I2S to FPGA	104.1 μs
OFDM encoding	208.2 μs
Transfer to radio	104.1 μs
Rx part: 427 μs	
Radio acquisition to DMA	104.1 μs
OFDM decoding and FPGA transfer	208.2 μs
Hamming decoding, CRC, audio extraction	104.1 μs
Audio stack	10.41 μs

TABLE V
ENERGY CONSUMPTION OF THE DIFFERENT RX PARTS

Rx part	Intensity	Voltage	Power
Lattice FPGA	23 mA	3.3 V	76.9 mW
LPC MCU	35 mA	1.8V	63 mW
Radio front-end (AD9361)	165 mA	1.8V	297 mW

D. Comparison with Zynq based implementation

In this part, we compare the performance obtained with our board to an implementation done on a Zynq based SDR architecture. We have implemented the same physical layer chain as described in Section III on a USRP E310 from Ettus Research [16]. This dual core Zynq architecture has the same RF board as our LOLA board and the processing was run on the custom Linux provided by Ettus. We depict on Table II some key performance indicators such as the energy consumption, the obtained PER in the LOS case at -20dBm and architecture properties. It shows that the decoding performance is very similar but that we have damatically reduced the power consumption by a factor 7.5.

V. CONCLUSION

In this brief we present the LOLA SDR, a new SDR architecture dedicated to low cost, low power and low latency applications. The proposed architecture leverages new generation of low power microcontrollers that embed calculation capacities as well as coprocessors. By incorporating such component on a board with a generic wideband radio transceiver and low power FPGA, a very flexible yet efficient SDR architecture can be synthesized. We have demonstrated the benefit of our approach by proposing a showcase of an audio broadcast application: the complete processing chain is capable of continuously handling a high definition audio stream in real time with a very low latency and good decoding performance. The energy consumption is of order of magnitude lower than state-of-the-art SDR architectures with equivalent computational capacity. It is also worth mentioning that the cost is also greatly alleviated. LOLA architecture paves the way for an area of flexible, efficient, low cost, low power SDR platform that can be used in various applications.

VI. ACKNOWLEGMENT

The research leading to these results received funding from the French Brittany Region and Lannion Tregor Communauté (APP-PME HAD-OC).

REFERENCES

[1] J. Mitola, "Software radios: Survey, critical evaluation and future directions," *IEEE Aerospace and Electronic Systems Magazine*, vol. 8, no. 4, pp. 25–36, 1993.

[2] M. Dardaillon, K. Marquet, T. Risset, and A. Scherrer, "Software defined radio architecture survey for cognitive testbeds," in *Proc. International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2012, pp. 189–194.

[3] C. Lavaud, R. Gerzaguet, M. Gautier, and O. Berder, "AbstractSDRs: Bring down the two-language barrier with Julia Language for efficient SDR prototyping," *IEEE Embedded Systems Letters*, pp. 1–1, Jan. 2021.

[4] Y. H. Lin, Q. Wang, J. S. Wang, L. Shao, and J. Tang, "Wireless IoT Platform Based on SDR Technology," in *Proc. IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013, pp. 2245–2246.

[5] S. Garg, N. Agrawal, S. J. Darak, and P. Sikka, "Spectral coexistence of candidate waveforms and DME in air-to-ground communications: Analysis via hardware software co-design on Zynq SoC," in *Proc. IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, 2017, pp. 1–6.

[6] N. Kumar, M. Rawat, and K. Rawat, "Software-Defined Radio Transceiver Design Using FPGA-Based System-on-Chip Embedded Platform With Adaptive Digital Predistortion," *IEEE Access*, vol. 8, pp. 214 882–214 893, 2020.

[7] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, and N. Wehn, "MAGALI: A Network-on-Chip based multi-core system-on-chip for MIMO 4G SDR," in *Proc. IEEE International Conference on Integrated Circuit Design and Technology*. IEEE, 2010, pp. 74–77.

[8] R. Akeela and Y. Elziq, "Efficient co-design partitioning of WLANs on SoC-based SDRs," *Microsystem Technologies*, vol. 26, no. 4, pp. 1141–1158, 2020.

[9] R. W. Stewart, L. Crockett, D. Atkinson, K. Barlee, D. Crawford, I. Chalmers, M. McLernon, and E. Sozer, "A low-cost desktop software defined radio design environment using matlab, simulink, and the rtl-sdr," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 64–71, 2015.

[10] 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} Symposium on Networked Systems Design and Implementation*, 2020, pp. 1031–1046.

[11] S. Szilvási, B. Babják, P. Völgyesi, and A. Lédeczi, "Marmote SDR: Experimental platform for low-power wireless protocol stack research," *Journal of Sensor and Actuator Networks*, pp. 631–652, 2013.

[12] X. Cai, M. Zhou, T. Xia, W. H. Fong, W.-T. Lee, and X. Huang, "Low-power SDR design on an FPGA for intersatellite communications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 11, pp. 2419–2430, 2018.

[13] M. Xhonneux, J. Louveaux, and D. Bol, "Implementing a LoRa Software-Defined Radio on a General-Purpose ULP Microcontroller," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS)*, 2021, pp. 105–110.

[14] 3GPP, Technical Specification Group Radio Access Network, "Physical channels and modulation," Dec. 2016.

[15] M. Sprachmann, "Automatic generation of parallel CRC circuits," *IEEE Design Test of Computers*, vol. 18, no. 3, pp. 108–114, 2001.

[16] Ettus Research, "Universal Software radio platform (USRP)," 2017, https://www.ettus.com/product/details/E310-KIT.