Software Defined Radio Platforms for Wireless Technologies

Dereje M. Molla, Hakim Badis, Laurent George, Marion Berbineau

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xx.xxxx/ACCESS.xxxx.xxx

Software Defined Radio Platforms for Wireless Technologies

DEREJE M. MOLLA[1], HAKIM BADIS[1], LAURENT GEORGE[1], AND MARION BERBINEAU[2]

[1]LIGM, ESIEE Paris, Gustave Eiffel University, CNRS, F-77454 Marne-la-Vallée, France (e-mail: dereje.molla@esiee.fr, hakim.badis@univ-eiffel.fr, laurent.george@esiee.fr)
[2]COSYS Department, Gustave Eiffel University, France (e-mail: marion.berbineau@univ-eiffel.fr)

Corresponding author: Dereje M. MOLLA (e-mail: dereje.molla@esiee.fr).

The authors acknowledge funding from I-Site Future in the framework of the Western project. One of the authors acknowledge partial funding by the regional project SMARTIES in the framework of the ELSAT 2020 program co-financed by the European Union with the European Regional development fund, the French state and Hauts de France Regional council.

ABSTRACT Wireless connectivity standards have been developed to meet the requirements of various applications. To support a wireless standard, a wireless transceiver should be equipped with a Radio Frequency (RF) transceiver. Traditional RF transceivers are designed and implemented on a radio chip or an embedded module in a System-on-a-Chip (SoC), ensuring small size, high performance, low power consumption, and cost. However, this traditional implementation design limits directly or indirectly the programmability and flexibility of the RF transceivers. An alternative solution is to implement RF transceivers using Software Defined Radio (SDR) platforms. In the market, SDR platform hardware exists with different configurations, performance, cost, size, etc., making it hard to select the minimum SDR platform necessary to satisfy the wireless standard requirements. This paper aims to provide a list of well-known General Purpose Processor (GPP) based SDR platforms satisfying the minimum specifications of selected wireless standards. To this end, we first review the characteristics of selected wireless technologies. Then, we investigate existing SDR platform architecture and their maximal performance in terms of the frequency range, bandwidth, symbol rate, bitrate, and latency support. Finally, we intersect the wireless standard requirements with the corresponding SDR platform parameters and provide a list of GPP-based SDR platforms for some existing wireless technology implementations. While investigations related to frequency, bandwidth, symbol rate and bitrate are supported by theoretical results, latency is obtained from experiments by benchmarking existing implementations.

INDEX TERMS General Purpose Processor, Software Defined Radio, Transceivers, Wireless Technologies.

I. INTRODUCTION

THE number of wireless devices used by various wireless application domains such as Wireless Sensor Networks (WSNs) [1], Internet of Things (IoT) [2], cellular base stations [3], etc., has increased tremendously in the past decade. Several wireless technologies are standardized to enable the interconnection between the different wireless devices including NFC, RFID, IEEE 802.15x, IEEE 802.11x, LoRa, Sigfox, 3GPP 3G/4G/5G, etc., [3]–[5]. A wireless device can incorporate one or multiple wireless transceivers supporting distinct wireless technologies. Each transceiver performs all the physical (PHY) and a portion of the Media Access Control (MAC) layer operations through integrated analog and digital circuit blocks. Indeed, most of the PHY layer analog operations are implemented on a dedicated and integrated analog hardware such as amplifiers, radio frequency (RF) synthesizers, filters, etc. On the other hand, some PHY layer digital baseband and time critical MAC layer functions are fully implemented on a digital hardware such as Application Specific Integrated Circuits (ASICs), a Programmable Digital Signal Processor (PDSP), Application Specific Instruction Set DSP (DSP ASIP) or a mixed solution using ASIC hardware accelerators with PDSP or with DSP ASIP [6]. This traditional implementation considerably limits directly or indirectly the programmability and flexibility of the transceivers for upgrading or handling multiple wire-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3154364, IEEE Access

less standards. Moreover, processors of wireless transceivers are mostly proprietary which prevent programmers and researchers from access to reprogram the instruction code.

An alternative solution to allow programmers and researchers to easily control the hardware and program the software of wireless transceivers is to use an implementation based on general-purpose processor (GPP) based Software Defined Radio (SDR) platforms, which is a reconfigurable and reprogrammable radio transceiver. In such platforms, the PHY layer digital baseband and MAC layer operations are implemented on a GPP, and the PHY layer analog RF/IF front-end operations are controlled using an analog device board supporting a wide range radio spectrum. This solution have been explored by many researchers to investigate the architecture, challenges and compare the performance of several SDR platforms [7]–[9]. In addition, as the PHY and MAC layers are performed in software by GPP host and due to the reconfigurability and reprogrammability of the radio transceiver, the SDR platform can be used to implement multiple wireless technologies. Such benefit has been exploited in many recent research works such as [10], [11]. These benefits conjugated with the continuous advancement in processing performance (hardware and software) and decreasing price of GPPs made GPP-based SDR platforms gain much attention for implementing and testing wireless technologies [12]–[16]. Moreover, it is also used to build testbeds and/or perform experimentation to study different features of communication systems and suggest performance improvement [17]–[19].

Currently, several SDR platforms are available in the market and research community. To implement a desired wireless technology, an appropriate SDR platform need to be selected. Previous research works such as [7] and [9] have presented the challenges during SDR platform selection process and compared the performance of SDR platforms in the general context. However, they abstain from addressing the specific considerations required by SDR platforms based on the requirement of wireless technologies. This problem is slightly addressed by other researchers and developers in two different perspectives, by designing a custom SDR architecture suited to a specific implementation such as [20]–[22], and/or providing list of recommended requirements to implement a certain wireless technology. Regarding the latter case, the recommendations are usually provided by SDR platform software implementations such as Software Radio Systems (SRS) [14], OpenAirInterface Software (OAI) [23], gr-IEEE-802.15.4 [15], gr-LoRa [24], etc. However, these recommendations are essentially formulated after multiple experimental tests. Nevertheless, as the experimental tests are not exhaustive, the recommended SDR platforms may be over-dimensioned and thus the minimal necessary configuration (carrier frequency and bandwidth, clock rate, communication interface support, GPP cores, GPP processing power, software architecture, etc.), can be exceeded.

The aim of this paper is to provide a list of possible GPP-based SDR platforms in terms of hardware components satisfying the minimum specifications of well-known wireless technologies. This is achieved by analyzing what a wireless technology requires at minimum in terms of frequency range, bandwidth, symbol rate, bitrate and latency, and the performance offered by GPP-based SDR platform components. The contributions of this paper can be summarized as:

- we present a detailed study of the architecture of GPP-based SDR platforms, and analyze their capabilities in terms of the performance metrics;
- we drive the minimum performance requirements of the most relevant wireless technologies. We use these requirements to draw mapping conditions in order to determine which GPP-based SDR platform is appropriate to successfully perform a targeted wireless technology;
- we identify existing wireless technology implementations from the literature that use GPP-based SDR platforms, examine their performance metrics, and suggest a list of other possible SDR platforms to implement the use-cases described in the literature.

Thus, the in-depth analysis of selected wireless technologies and SDR platforms allows researchers from both academia and industry to easily understand required parameters, software and hardware components of SDR platforms. We believe this paper will help researchers (SDR platform users and SDR software developers) looking for the appropriate SDR platform to implement a given wireless technology. To the best of our knowledge, this paper is the first to perform mapping several wireless technologies with GPP-based SDR platforms.

This paper is organized as follows: Section II provides classification and characteristics of well-known wireless connectivity technologies. Section III discusses the architecture of GPP-based SDR platforms and provides general background on the hardware and software components. Section IV provides a detailed study on the performance parameters of GPP-based SDR platforms and presents numerical results using selected GPP-based SDR platforms. Section V presents a mapping between SDR platform performance and wireless technology requirements. Open research challenges and future directions are given in section VI. Finally, section VII provides conclusions to this paper.

II. WIRELESS CONNECTIVITY TECHNOLOGIES

The interconnection between different wireless devices is enabled by a wide range of wireless technologies that can cover from very short distance (in centimeter range) to several kilometers. Thus, wireless connectivity technologies can mainly be classified into three groups based on the range they cover [25]: i) short-range wireless technologies, ii) Wireless Local Area Networks (WLANs), and iii) Wireless Wide Area Networks (WWANs). Within each of these categories, several wireless technologies are standardized, as shown in Fig. 1. The subsequent subsections present the main PHY and MAC layer characteristics of major wireless technologies. We note that, the frequency requirement of wireless technologies stated in the tables is given based on their standard. However,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3154364, IEEE Access

the specific frequency band used by a wireless technology depends on its allocation for global and regional use and also countries regulation [26].

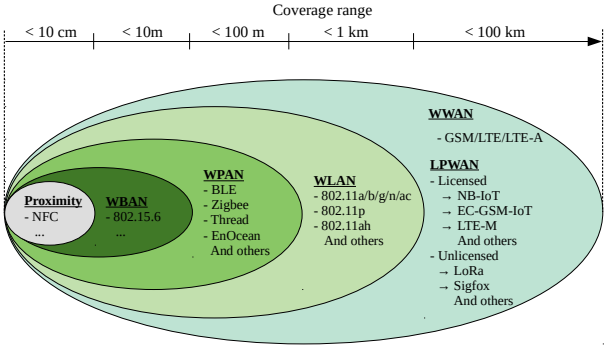

FIGURE 1: Wireless technologies.

A. SHORT-RANGE WIRELESS TECHNOLOGIES

Short-range wireless technologies include proximity communication, Wireless Body Area Networks (WBANs) and Wireless Personal Area Networks (WPANs). They are mainly characterized by their short-range coverage operating under the unlicensed industrial, scientific and medical (ISM) frequency bands. International Organization for Standardization(ISO)/International Electrotechnical Commission (IEC) defines the PHY and MAC layer requirements for proximity technologies, and IEEE defines for WBAN and WPANs. Some of the most prominent wireless technologies are Near Field Communication (NFC), Radio Frequency Identification (RFID), IEEE 802.15.6 (NB PHY, UWB PHY and HBC PHY), IEEE 802.15.1, and IEEE 802.15.4. IEEE also defines the PHY and MAC layer for Low-Rate WPAN to meet the limited resource requirement of IoT and WSN devices. Among these standards are IEEE 802.15.4 and Bluetooth Low Energy (BLE), which are developed for networks with low power consumption, low deployment cost and less complexity. Table 1 gives the PHY and MAC layer characteristics of some of the short-range wireless technologies.

B. WIRELESS LOCAL AREA NETWORK - WLAN

WLAN is mostly designed for wireless connectivity covering less than one kilometer range. Different standards falling under this group include IEEE 802.11b, IEEE 802.11a/g, IEEE 802.11n, IEEE 802.11ac/ax, IEEE 802.11ah and IEEE 802.11p. The first four standards (IEEE 802.11a/b/g/n/ac) are the most popular wireless standards used by WiFi [32], [33]. They are high bandwidth technologies that supports the communication of bandwidth-intensive applications like streaming video, and enable wireless gateways with a high-speed interface to relay traffics requiring large bandwidth and IP connectivity [34]. Other standards promising for IoT and WSN deployment due to their low power and long range wireless communication support are IEEE 802.11ah (WiFi-HaLoW) [35] and IEEE 802.11p [36]. IEEE 802.11ah defines PHY and MAC layer specification for large scale sensor networks and extended range hotspot. It operates in the sub-GHz ISM bands. IEEE 802.11p is an amendment of 802.11 standard that operates in the 5.9 GHz band and offer wireless connectivity between mobile vehicles (and/or vehicles and roadside units) and designed to guarantee low latency [36]. Table 2 enlists their PHY and MAC layer characteristics.

C. WIRELESS WIDE AREA NETWORK - WWAN

WWANs are meant for large area coverage in the order of kilometers. The wireless communication technologies standardized for such wide coverage have mainly two groups: cellular networks such as 2G, 3G, 4G and 5G; and Low Power Wide Area Networks (LP-WANs). The latter also has two groups: licensed and unlicensed. The licensed LP-WAN consists of Narrow-Band IoT (NB-IoT), Enhanced Coverage-GSM IoT (EC-GSM-IoT) and Long Term Evolution for Machine Type Communication (LTE-M). They are upgrades of cellular communication technologies for IoT applications.

TABLE 1: Characteristics of short-range wireless technologies.

Standards	PHY layer						MAC layer	
	Frequency Band (MHz)	Channel size (Bandwidth (MHz))	Max. symbol rate (MSym/s)	Modulation		Max. bitrate (kbit/s)	Latency (ms)	Access scheme
				Bit-to-Sym mapper	Subcarriers			
ISO/IEC 18092:2013 (NFC) [27]	13.56	1	0.106, 0.424, 1.696	ASK (OOK) FSK	1	106, 212, 424	5.340	Electromagnetic coupling
ISO/IEC 18000 (JTC1 SC31) (RFID) [28]	0.125,13.5,433, subGHz, 2.4G, UWB	10, 14, 1.74, 7, 8	–	BPSK	1	640		FDMA/TDMA
NB PHY [29]	0.4 – 2.4	0.3, 0.32, 0.4, 1	0.6	DBPSK, $\pi/4$-DQPSK, $\pi/8$-D8PSK, GMSK	1	0.0759 – 0.9714	pSIFS*= 75	CSMA/CA, slotted ALOHA
UWB PHY (IR-UWB, FM-UWB) [29]	3.493 – 9.984	499.2	15.6	On-OFF modulation, BPSK/QPSK, CP-BFSK, WB-FM, DBPSK/DQPSK	1	15.6	pSIFS = 75	CSMA/CA, slotted ALOHA
HBC PHY [29]	0.021	5.25	0.328	Based on NFC (OOK, DSSS)	1	1.3125	pSIFS = 75	CSMA/CA, slotted ALOHA
IEEE 802.15.1 [30]	2.4G	1, 2	2	FHSS based GFSK, DQPSK	1	2000	IFS = 150	TDMA
IEEE 802.15.4 [31]	sub-GHz and 2.4G	0.4, 0.6, 2, 5	0.0625	DSSS based BPSK, O-QPSK	1	20, 40, 100, 250	*aTurnaroundTime* = 192; *macAckWaitDuration* = 864	Unslotted CSMA/CA
							aTurnaroundTime = 192; *aTurnaroundTime* + *aUnitBackoffPeriod* = 512	Slotted CSMA/CA
							macTsTxAckDelay = 1000	GTS allocation

* SIFS: Short Inter-Frame Space.

TABLE 2: Characteristics of WLAN standards.

Standards	PHY layer							MAC layer	
	Frequency Band (GHz)	Channel size (Bandwidth (MHz))	Symbol duration + GI*(μs)	Max. symbol rate (kSym/s)	Modulation		Max. bitrate (Mbit/s)	Latency (μs)	Access scheme
					Bit-to-Symbol mapper	Subcarriers			
IEEE802.11b [32]	2.4	22	0.7273	1375	CCK,QPSK	1	11	SIFS = 10; ACK timeout = 30	CSMA/CA
IEEE802.11a/g [32]	5/2.4	20	4	250	BPSK,QPSK, (16,64)QAM	52	54	SIFS=16/10; ACK timeout = 25/19	CSMA/CA
IEEE802.11n [32]	2.4/5	20/40	4	250	BPSK,QPSK, (16,64)QAM	56/114	72.2, 150	SIFS = 10; ACK timeout = 19	CSMA/CA
IEEE802.11ac [33]	5	20/40/80/160	4	250	BPSK,QPSK, (16, 64, 256)QAM	56/114/242/ 484	86.7 – 6000	SIFS = 16; ACK timeout = 25	CSMA/CA
IEEE802.11ah [35]	subGHz (0.7 - 0.9)	1/2/4/8/16	40	25	BPSK,QPSK, (16, 64, 256)QAM	32/56/114/ 242/484	0.1 – 347	SIFS = 160	CSMA/CA
IEEE802.11p [36]	5.8/5.9	10	8	125	BPSK,QPSK, (16,64)QAM	52	3 – 27	SIFS = 32	CSMA/CA

* GI is long Guard Interval (8μs for 802.11ah, 1.6μs for 802.11p and 0.8μs for others).

TABLE 3: Characteristics of WWAN standards.

	Standards	PHY layer							MAC layer	
		Frequency Band (MHz)	Channel size (Bandwidth (MHz))	Symbol duration + GI (μs)	Max. symbol rate (kSym/s)	Modulation		Max. bitrate (bit/s)	Latency (ms)	Access scheme
						Bit-to-Sym mapper	Subcarriers**			
Cellular	GSM/	800/900, 1800/1900	0.2	3.69	271	GMSK, QPSK, 8PSK, 16QAM	1	171k (GPRS), 384k(EDGE)	slot time = 0.577	TDMA/FDMA
	3GPP LTE	LTE bands [40]	1.4,3,5,10,15,20	66.66	15	QPSK, 16QAM, 64QAM	73/181/301/ 601/901/1201	50.42M (UL), 100.8M (DL)	4	SC-FDMA, OFDMA
Cellular for IoT	NB-IoT [41]	LTE bands	0.18	66.66	15	BPSK, QPSK, 16QAM	12	250k	20	SC-FDMA/ FDMA, OFDMA
	EC-GSM-IoT	GSM bands	0.20	3.69	271	GMSK, 8PSK	1	240k	slot time = 0.577	TDMA
	LTE-M	LTE bands	1.4	66.66	15	QPSK, 16QAM	73	1M	end-end = 100	SC-FDMA, OFDMA
LP-WAN	LoRa*	433/868(EU), 915(US), 430(Asia)	0.125,0.25,0.5	1024,512,256	3.91	LoRa, FSK	1	50k	24	Proprietary CSS
	Sigfox	868(EU), 902(US)	0.0001(UL), 0.0006(DL)	10,000/60,000	0.1, 0.6	DBPSK(UL), GFSK(DL)	1	100(UL), 600(DL)	$\sim$ 2000	Proprietary UNB/FHSS

* Spreading factor (SF) = 7. ** Subcarriers are shown only for the downlink.

Unlicensed LP-WAN includes Long Range (LoRa), Sigfox, etc., [37]. LP-WAN technologies are designed for Machine-to-Machine (M2M) and IoT applications that need to forward small payload data at low data rate and low power consumption [38], [39]. The cellular technologies (2G, 3G, and 4G), on the other hand, consume a lot of device energy which may cause a negative impact on low-power IoT devices. However, they are useful for IoT gateways or IoT devices running bandwidth-intensive applications. Table 3 quantifies the requirements of cellular and LP-WAN technologies.

III. GPP-BASED SDR PLATFORM ARCHITECTURE

The implementation design of conventional wireless transceivers, in general, lacks reprogrammability, flexibility and scalability. Therefore, upgrading the software, changing the logic of the dedicated hardware or reusing the transceiver to implement a wireless standard other than the one the transceiver was designed for is limited or non-existent. Moreover, conventional wireless transceivers are mostly proprietary which prevent developers and researchers from access to reprogram the assembly instruction set. An alternative solution to mitigate these limitation is using SDR platforms. In addition to the programmability feature, the SDR platform also serve as a multi-technology gateway by performing multiple wireless technologies using a common set of radio transceiver [10], [11]. It also allows to reuse software across multiple radio devices and download software over-the-air to implement new standards and fix bugs [8]. Furthermore, it is recently being used to mitigate cross-technology interference problem faced by conventional technologies [42].

An SDR platform is a class of radio transceivers which controls the analog RF/IF part using an open-source analog device board, named SDR device, and implements all the digital part using programmable host processor. The programmable host processor can be GPP, DSP ASIP or FPGA. The scope of our study is limited to GPP-based SDR platforms due to its easy programmability using a high-level language and its flexibility for reconfiguration and handling complex algorithms [9]. The general architecture of GPP-based SDR platform is illustrated in Fig. 2. It is mainly sectioned into three parts as **SDR device**, **communication interface** and **GPP host**. Each component of the platform has its communication parameters that contribute to the overall performance of the SDR platform. This section investigates the GPP-based SDR platform.

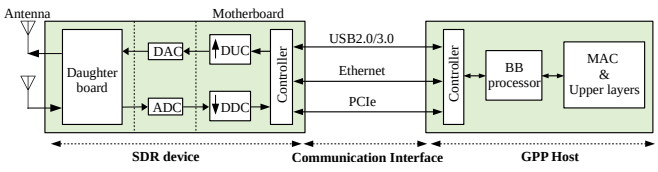

FIGURE 2: GPP-based SDR platform general architecture.

A. SDR DEVICE

An SDR device is a small handheld type of device which is capable of transmitting and receiving signals at different frequencies. It typically consists of software controllable analog RF/IF and digital IF front-ends. The former is called as daughterboard and the latter as motherboard. This subsection describes their respective tasks and characteristics.

1) Daughterboard

The daughterboard is essentially responsible to perform analog RF/IF processing functions such as filtering, amplification, conversion of signals from RF to IF and vice-versa, etc. It is mainly characterized by the frequency band it covers, the analog bandwidth, RF performance, number of channels, channel's capability (only RX or TX/RX), mode of operation as a half (HD) or full (FD) duplex, etc. Mainly, the operating frequency range, width of analog bandwidth and channel's capability determines the scope of the daughterboard to implement a wide range of wireless technologies using SDR device. The daughterboard interfaces the antenna with the motherboard. Indeed, most daughterboards integrate multiple input/output circuits to connect multiple separated antennas enabling simultaneous transmission and reception capability [43], [44]. Also, daughterboards integrate analog inputs/outputs to connect motherboard ADCs and DACs. In the market, daughterboards are either stand-alone component or integrated with a motherboard forming a single board. Table 4 lists few commercially available daughterboards.

TABLE 4: List of selected daughterboards.

Daughterboard	Frequency Band (MHz)	Bandwidth (MHz)	TX/Rx/Mode	Type	SDR family
UBX	10 – 6000	40/160	Tx/Rx/FD	Stand-alone	USRP [43]
CBX	1200 – 6000	40/120	Tx/Rx/FD		
SBX	400 – 4400	40/120	Tx/Rx/FD		
WBX	50 – 2200	40/120	Tx/Rx/FD		
B210	70 – 6000	56	Tx/Rx/FD	Integrated	
HackRF FE [45]	1 – 6000	20	TX/RX/HD	Integrated	HackRF
WARP Radio RF [46]	2400-2500, 4900-5875	40	TX/RX/FD	Stand-alone	Used by Sora
LMS7002M FPRF [47]	0.1 – 3800	60	Tx/Rx/FD	Integrated	LimeSDR
AD-FMCOMMS2-EBZ [44]	70 – 6000	56	Tx/Rx/FD	Integrated	Analog Devices

2) Motherboard

Motherboard is mainly responsible to perform digitization, channelization and sample rate conversion (digital up/down conversion [DUC/DDC]). To achieve these operations, motherboards integrate ADCs/DACs and a DSP processor that can be implemented using ASIC, DSP ASIP or FPGA. It also integrates one or more communication interfaces to connect with GPP host. It is mainly characterized by the maximum ADC and DAC sample rates, ADC and DAC resolution, DSP processor design and the supported communication interface. Table 5 provides characteristics of motherboards of selected SDR devices.

The motherboard, in SDR devices, interfaces daughterboard with a GPP host. It exchanges baseband samples with GPP host and IF analog signals with daughterboards. Actually, a received IF analog signal from a daughterboard is first digitized by ADC to get IF digital samples and then downsampled through DDC to obtain baseband digital samples.

TABLE 5: Characteristics of motherboards of SDR devices.

SDR family	DSP processor	Sample rate (MS/s)		Resolution (bits)		Comm. Interface	Type
		ADC	DAC	ADC	DAC		
HackRF One [45]	NXP LPC43XX ARM Cortex-M4 MCU	20	20	8	8	USB 2.0	Integrated
USRP B210 series [43]	Xilinx Spartan-6 FPGA	61.44	61.44	12	12	USB 3.0	Integrated
USRP X310 series [43]	Xilinx Kintex-7-410T FPGA	200	800	14	16	Ethernet	Separate
Microsoft Sora [48]	Virtex-5 FPGA	44	40	12	12	PCIe	Separate
LimeSDR [47]	Altera Cyclone IV EP4CE40F23 FPGA	160	640	12	12	USB 3.0, PCIe	Integrated

Finally, the baseband samples are transmitted to the GPP host through an integrated communication interface. On the reverse direction, received baseband samples are first upsampled through DUC to get IF digital samples and then converted to IF analog signal by DAC. Finally, the IF analog signal is forwarded to the daughterboard [49]. In the market, motherboards are either separate boards containing slots to plug daughterboards or a board integrating daughterboards.

B. COMMUNICATION INTERFACE

Data is transferred from GPP Host to SDR device and vice-versa using wired communication interfaces. These are based on commonly used data transfer communication interfaces like Universal Serial Bus (USB 2.0, USB 3.0, etc.), Ethernet (standard, fast, gigabit, etc.) and Peripheral Component Interconnect Express (PCIe 1.x, PCIe 2.x, etc). They consist of controllers such as Network Interface Controller (NIC), installed in both GPP host and SDR motherboard, to implement the communication interface standard. The communication interface technology implemented by the controller has specific characteristics defined by standards such as the maximal supported rate, maximal payload size, maximal cable length, etc., [50]–[52]. To allow more flexibility in data transfer speeds, some SDR devices include multiple communication interfaces (see Table 5). The controllers of both SDR device and GPP host should implement the same interface standard but not necessarily the same version. Indeed, different versions of the same standard may create connection between the two ends but they need to be synchronized for efficient data exchange. In such cases, the communication standard with the lowest rate will be agreed by auto-negotiation (as for Ethernet [51]) or backward compatibility (for example between USB 2.0 and USB 3.0) or manually [53].

C. GPP HOST

A GPP host is a programmable device that can perform computational tasks based on instructions given by software programs using either high or lower level programming languages. As such a GPP host combines hardware and software, and is responsible to handle their interaction. The subsequent sections review these components highlighting the parts that impact the processing speed of a GPP host.

1) Hardware

The GPP host components are, mostly, assembled in a single-board. This board contains SoC internal components such as GPP, internal memories, co-processors (GPU, DSP ASIP, etc.), and possible controllers for communication interfaces

(see section III-B) and SoC external components such as external memories, expansion slots, etc.

A GPP, which can be either microprocessor or microcontroller, is responsible for performing the digital PHY, MAC, and upper layer operations. Unlike the conventional transceivers, it has the advantage of either high or low level programmability without modifying the hardware. Although it offers high user flexibility, the high-level programmability usually results in performance degradation of the processor to satisfy the requirements of intensive computation signal processing tasks [54]. Indeed, the performance (processing speed) of GPP is largely determined by a clock, where lower clock speed implies a slow processor and less energy consumption [55].

The GPP may be of single core or multi-core processor. However, most GPPs currently are based on multi-core (Dual-core, Quad-core, etc.,) processors on a single physical Central Processing Unit (CPU) [7]. Each core, in a multi-core single CPU system, represents a single processor or execution unit capable of executing processes concurrently with other processors. This increases the number of instructions to be processed per clock cycle. In addition to clock speed and number of cores, system bus architecture (bus width, its clock frequency, the number of data it can transfer per clock cycle, etc.,) significantly affects the speed of processing [55]. The size and level of cache a CPU has also affects the speed of its processing. Other parameters that could affect overall processing speed of GPP are number of threads, memory size, number of ALUs, hyper-threading support, size of Single Instruction Multiple Data (SIMD) units, etc. The SIMD units allow a processor to perform simultaneously the same instruction (operation) on multiple data units [56]. Recent GPPs support SIMD architecture to improve performance capabilities [57]. Table 6 provides few examples of GPP host hardware. To achieve more computing performance, GPPs are usually complemented with co-processors like GPU, FPGA and DSP ASIP as accelerators [56], [58], [59].

TABLE 6: List of well-known GPP Host hardware.

GPP Characteristics	Desktop computer	Laptop	Smartphones	Embedded computer boards
Processor	Intel x86, AMD, ARM, MIPS	Intel x86, AMD, ARM, MIPS	ARM, Intel, MIPS	ARM, 8051 cores, microMIPS
SIMD support	Yes	Yes	Yes	Yes
Comm.Interface controller	USB,Ethernet,PCIe	USB, Ethernet, PCIe	USB	USB, Ethernet
Computing power	Low to High	Low to High	Low to Medium	Low to Medium
Size/Weight	Large	Medium	Small	Small
Example	Dell, Apple, HP, Lenovo, Acer, etc.	Dell, Apple, HP, Lenovo, Acer, etc.	iPhone, Samsung, Huawei, etc.	Raspberry Pi, Arduino, NXP, etc.

2) Software

The software part of GPP host controls the operation of the processor, input/output traffic of communication controllers and the SDR device. It is generally layered into three on top of the hardware processor as instructions set, kernel space and user space. The instruction set is defined as a group of instructions a processor can execute. Thereby, an instruction code (object code) generated by a compiler or an assembler can only contain instructions from this set. The instruction set is one of two types of instruction set architecture (ISA) designs: Reduced Instruction Set Computers (RISC) or Complex Instruction Set Computers (CISC) [60]. The ISA of GPPs can be based on CISC or RISC. To exploit the advantages of both instruction sets, modern GPPs are more based on hybrid ISA (using CISC instructions externally, but RISC techniques internally) [61]. Moreover, the use of RISC architecture can also be enhanced by adding Very Long Instruction Word (VLIW) extensions, a technique that offer instruction level parallelism [62].

The middle layer of the software system architecture is the kernel. It is the heart of an operating system (OS), linking the user space with the hardware processor [63]. To interact with the hardware, the kernel includes hardware drivers such as processor driver, hard disk driver, network controller driver, etc. To interact with user space, the kernel includes Application Program Interface (API) allowing programs in user space to access system resources (e.g., file systems, GPP time, virtual memory, etc.,) and services (e.g., scheduling, swapping, interrupt request (IRQ) handling, context switching, etc). It is precisely these services that impact the kernel space performances in terms of latency and overhead. To reduce the latency, additional functions such as the IRQ handler, process scheduling, reducing number of context switches, etc., are required. On the other hand, kernel overhead is the time due to managing resources such as GPP time, memory, disk, etc. The increased overhead often results in reducing the GPP time occupation and consequently the GPP throughput. As reducing the kernel latency requires additional functions, the kernel overhead will increase. It is obvious that a trade-off between kernel latency and GPP throughput exists, and a balance should carefully be designed as per user need.

At the top of software system architecture is the user space that consists of a portion of memory in which user applications are executed. Hereby, the user applications are PHY and MAC functions of the wireless technologies. The user applications are mostly written using high-level programming languages like C, C++, Java, Python, Matlab, etc. It is also possible to generate the user applications code via data flow textual/graphical programming languages like G programming, Python, C++, etc. These programming languages (high-level and data-flow) are generally included in software development toolkits such as GNU Radio [64], LabVIEW [65], Matlab [66], etc. The toolkits provide DSP libraries for DSP functions, libraries for runtime and compilation, graphical tools for creating signal flow graphs and generating flow-graph source code, etc.

The user application compiler is an element of most importance in assisting the processor to achieve high performance in speed and execution time. It is responsible for generating the instruction code using the ISA of the target processor. When a large variety of target processors are supported, the compiler is said to be general (like the GNU Compiler Collection (GCC) [67]). The general compilers also implement optimizations to improve the GPP performance by increasing

the parallelism levels through three mechanisms: instruction-level parallelism (ILP) which allows multiple instructions to be executed at the same time, thread-level parallelism (TLP) which allows multiple threads to run simultaneously or pseudo-simultaneously on single/multiple cores, and data-level parallelism (DLP) which enables performing multiple data-elements simultaneously. This entails an optimal source code generation in size and execution time, according to the target processor. Examples of optimizations are automatic vectorization [68], automatic parallelization [68], [69], inter-procedural optimization [70], and SIMD intrinsics (assembly-coded functions [71]). Table 7 lists well-known software implementation toolkits.

TABLE 7: Software development toolkits.

Implementation tools	Data-flow programming language		High Level programming language	Scheduler	Optimization type	License
	Textual	Graphical				
GNU Radio	C++/python	GRC	C++/python	yes	SIMD (VOLK) [72]	open-source
Labview [73]	C/MathScript	G/DFIR	C	yes	Auto (DFIR/LLVM)	Commercial
Matlab/ Simulink	Matlab(.m code)	Simulink	C	yes	Manual or Auto [69]	Commercial
C/C++	X/X	X/X	C/C++	Hard code	Manual or compiler	open-source

IV. SDR PLATFORM PERFORMANCE ANALYSIS

To implement a wireless technology on SDR platforms or use existing implementations, it is necessary that the selected SDR platform (SDR device, communication interface and GPP Host) performance should meet at least the requirements of the target wireless technology. These requirements are mainly given in terms of operating frequency band, bandwidth, symbol rate, bitrate, latency, etc. In this section, a thorough theoretical analysis of these performance parameters in GPP-based SDR platform architecture is presented along with the minimal/maximal values offered by the components.

A. FREQUENCY BAND

The frequency band of SDR platforms is the operating frequency range covered by the SDR device. This is determined at the daughterboards from the local oscillator (LO) signals generated by the frequency synthesizer, such as Phase-Locked Loop (PLL) synthesizer. Large frequency band needs large LO frequency range, and consequently wideband frequency synthesizers. Daughterboard's datasheet usually lists the operating frequency band of SDR devices (see Table 4 for well-known daughterboards). To cover the range of frequency bands supported by daughterboards, SDR device's need to use appropriate type of antenna [74].

B. BANDWIDTH

Any analog or digital signal has a bandwidth defined as the occupied range of frequencies carrying most of its energy. This range varies at each stage of the signal chain. Hence, it can be expressed differently (but related) according to the signal processing stage. Indeed, at the RF front end stage, it is expressed as the analog bandwidth or RF channel width. At the ADC/DAC stage, it is expressed as the DAC/ADC sample rates. When the signal is processed at the digital front end (DFE) stage, its bandwidth is expressed as the DFE sample rate. On the communication link between the DFE and GPP, the bandwidth is limited by the communication interface speed. At the GPP host, the bandwidth is expressed as the symbol rate. Fig. 3 illustrates the main points in the signal chain where the bandwidth of SDR platform is characterized. In this section, we examine the analog bandwidth, ADC/DAC and DFE sample rates. Sections IV-C, IV-D and IV-E will address the interface speed, symbol rate and bitrate.

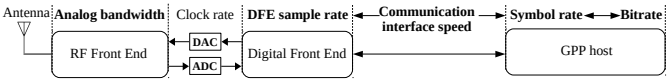

FIGURE 3: Bandwidth at each block of SDR-GPP signal chain.

1) Analog bandwidth

This bandwidth, measured in Hz, is determined by the RF front end (daughterboard) of the SDR device. It is configured mainly by the analog baseband low pass filter (LPF) to vary from 0 Hz to the specified cut-off frequency, f_{cut}. Thus, in a direct-conversion (Zero IF) I/Q modulator [75] both at the transmitter and receiver side, the analog bandwidth at RF front end is determined by the LPF and its f_{cut}. It is equal to twice f_{cut} for Ideal LPFs that completely eliminate (attenuate) all frequencies above the f_{cut}. The excess bandwidth, defined as the transition band in LPF datasheets [76], of an Ideal LPF is null, and hence its Roll-off factor (ratio between passband and transition band) is null. In real-world, practical LPFs are not perfect and have a transition region where some high frequencies above the f_{cut} can pass. Consequently, the real analog bandwidth is greater than twice f_{cut} and can be formulated as follows:

$$\text{Real Analog bandwidth} = (2 \times f_{cut}) \times (1 + \text{Roll-off factor}) \quad (1)$$

where the Roll-off factor is in the range [0,1]. Equation (1) assumes signals spectra that would occur after theoretical cut-off point. Thus, the Roll-off factor shifts the bandwidth towards the transition band so that we can minimize loss of information.

Most of the LPFs are programmable and can take different f_{cut} values where each f_{cut} is assigned a roll-off factor. The maximal real analog bandwidth is achieved by the highest f_{cut} scaled by [1 + Roll-off factor]. Table 8 summarizes the analog bandwidth as theoretically computed from (1) and the maximal ideal analog bandwidth (twice f_{cut}) as indicated on the daughterboard's datasheet.

2) ADC/DAC sample rate

The DAC sample rate, given on Samples per Second (S/s), allows to determine the time interval between two samples applied to the input of a DAC. The ADC sample rate determines the time interval between two samples at the output of an ADC. Both sample rates are related to the input signal spectrum by the Nyquist-Shannon sampling theorem [77].

TABLE 8: Maximal analog bandwidths of SDR devices.

SDR	Max.cutoff frequency	Max. Ideal Analog bandwidth (MHz)	Roll-off factor	Max. real Analog bandwidth (MHz)
HackRF [45]	10	20	0	20
UBX-40, SBX-40, CBX-40, WBX-40 [43]	20	40	1.5	100
SBX-120, CBX-120, WBX-120 [43]	60	120	0.667	200
UBX-160 [43]	80	160	0.25	200
USRP-B2x0 [43]	28	56	0.097	61.44
Microsoft Sora [46]	10	20	1	40
LimeSDR [47]	80	160	0	160

Theoretically, for a given ADC/DAC sample rate, the maximum frequency that can be reproduced is half the sample rate (Nyquist frequency) to avoid aliasing effect. As the maximum frequency of an equivalent complex baseband (a complex valued representation of the real baseband) is [real analog bandwidth / 2], the sample rate needs to be greater than the real analog bandwidth (see (1)). The more sample rate is greater than the real analog bandwidth, the more the band gap increases between the real analog bandwidth copies repeated at multiples of sample rate resulting on zero-loss on bandwidth. This band gap has an amount of [ADC sample rate – real analog bandwidth] Hz.

In motherboard of SDR devices, the integrated ADC/DAC can support one or multiple sample rates where one at a time can be selected. The highest sample rate value determines the largest analog bandwidth. Table 9 shows the supported sample rates by well-known SDR devices. This table also shows whether the SDR device has fixed or selective sample rates. Using selective sample rate is preferable than fixed rate to adapt the real analog bandwidth to the necessary bandwidth asked by applications' rate requirement and expressed by the DFE sample rate (see section IV-B3). When the nearest sample rate is greater than the DFE sample rate, an adjustment through interpolation and decimation process is necessary [49].

TABLE 9: Analog bandwidth and ADC/DAC sample rates.

SDR device	Ideal Analog bandwidth (MHz)	Clock rate (MHz)	Supported ADC sample rates (MS/s)	Supported DAC sample rates (MS/s)	Selective/ Fixed
HackRF [45]	20	20	20	20	Fixed
USRP-X3x0 (UBX-160) [43]	160	200, 184.32	195.31K – 200M; 180.0K – 184.32M	800	Selective
USRP-B2x0 [43]	56	5 – 61.44	61.44	61.44	Selective
Microsoft Sora [48]	20	40, 44	40, 44	40	Selective
LimeSDR [47]	160	640	20 – 160	80 – 640	Selective

3) Digital Front End (DFE) sample rate

This rate, given on Samples per second (S/s), defines the constant speed by which I/Q samples are exchanged between the DUC/DDC (interpolation/decimation) stages and the interface controller. It can be specified either explicitly by the user or implicitly from the real analog bandwidth of the channel. The sample size (in bits) is determined by the DAC/ADC resolution. At the transmitter side of an SDR device, arriving I and Q data samples from the GPP host (in a format configured by the user, e.g., 32-bit float) join their corresponding queue waiting for service by the DFE. The arrival rate at the queue is constant over time and is determined by the bitrate of GPP host. The interpolation stage of the DFE retrieves samples from the queue at DFE sample rate, which is the service rate of the queue system. As the DFE sample rate can take very high values, it is extremely important that the arrival and service rates should be equal after normalization to avoid waiting for the queues to become non-empty (underflow). Then, the interpolation stage increases the DFE sample rate of input samples to higher output rate equal to the DAC sample rate (see Fig. 4). The applied interpolation factor is equivalent to the ratio of the DAC sample rate to the DFE sample rate.

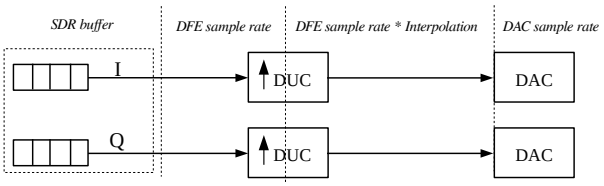

FIGURE 4: DFE sample rate (Digital Up Converter).

At the receiver side of an SDR device, as shown in Fig. 5, the DFE receives samples at a speed of ADC sample rate. It performs decimation to decrease the input ADC sample rate to a lower rate equal to the DFE sample rate. The applied decimation factor is equivalent to the ratio of ADC clock rate to the DFE clock rate. The output samples are, then, inserted into the I/Q queues waiting to be transmitted to the GPP host. A queue overflow occurs when the GPP host cannot retrieve samples as fast enough. As in the transmitter side, it is extremely important that the DFE sample rate be close to the bitrate of GPP host.

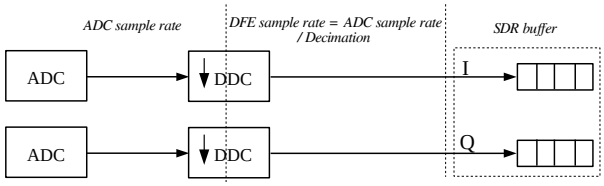

FIGURE 5: DFE sample rate (Digital Down Converter).

From the above discussion, as the DFE sample rate should be close to the bitrate of GPP host, it can be used to define the necessary channel bandwidth required by user application in the GPP host. Since the necessary channel bandwidth is included in the real analog bandwidth (see section IV-B1), the DFE sample rate should be smaller than the real analog bandwidth. Also, the DFE sample rate should be increased or reduced to fulfill the DAC and ADC clock rates. Some SDR devices require a strictly-integer interpolation and decimation factors, and it is strongly desirable for that factors to be even and it's much better if the factors are in power of two [78]. Thus, specifying appropriate DFE sample rate is another requirement to be considered by the user.

4) The maximal GPP-based SDR platform bandwidth

Since GPP-based SDR platform's bandwidth is concave, it takes the lowest value between analog bandwidth, ADC/DAC sample rate and DFE sample rate. The maximal DFE sample rate is achieved when its value reaches to ADC sample rate and DAC sample rate at unity decimation and interpolation factor, respectively. However, as stated above, DFE sample rate cannot be greater than real analog bandwidth. Consequently, the maximal GPP-based SDR platform bandwidth (in Hz) is the minimum from maximal DFE sample rate and analog bandwidth.

C. COMMUNICATION INTERFACE SPEED

This speed, given in bit per second (bit/s), refers to the PHY layer net bitrate of the wired interface between GPP host and SDR device. The PHY layer net bitrate defines the amount of transferred bits, excluding the PHY layer protocol overhead, per second over a communication link. Obviously, its value (specified by the standard) is less than the real transmission rate defined at the PHY layer gross bitrate that includes the PHY layer overhead (channel coding, modulation, physical framing, guard interval, etc). For example, the PHY layer net bitrate of the Gigabit Ethernet is 1 Gbit/s and the real transmission rate is 1.25 Gbit/s due to the 8B/10B encoding [79]. Since I/Q samples, exchanged between GPP host and SDR device, are encapsulated in frames of the used interface, their rate is limited by the communication interface speed. In the upcoming discussion, the I/Q sample rate is the equivalent of DFE sample rate defined in bit/s and related by [DFE sample rate $\times$ I/Q symbol format].

The communication interface speed, despite it excludes the PHY layer protocol overhead, it includes upper layer protocols head such as link layer head, network layer head, etc. Based on this remark, the peak (maximum achievable) rate of encapsulated I/Q samples (useful data rate) is limited by the communication interface speed weighted by a factor of [payload size / data link frame size], known as the link efficiency. Higher the link efficiency, closer the peak I/Q sample rate to the communication interface speed. The peak I/Q sample rate is an instantaneous rate that doesn't consider link occupancy. Consequently, its value will be very large to be taken as an acceptable upper bound, otherwise the real I/Q sample rate will be under-constrained. Thus, it is necessary to include the link occupancy on the upper bound to get the maximal acceptable I/Q sample rate (or maximal acceptable DFE sample rate). To do so, the peak I/Q sample rate weighted by the link occupancy, expressed as [peak I/Q sample rate $\times$ link occupancy], will give us the maximal acceptable I/Q sample rate. Henceforth, we limit the I/Q sample rate or DFE sample rate by the maximal acceptable I/Q sample rate.

The maximal acceptable I/Q sample rate is affected by three factors: the communication interface mode, GPP host application mode and SDR device capability. The communication interface between the GPP host and SDR device motherboard can work in either half or full duplex modes. With full-duplex interface mode, the communication link can simultaneously be fully occupied in both directions (Host$\rightarrow$SDR and SDR$\rightarrow$Host) allowing to benefit from full-speed on each direction. In this case, the transmit and receive links have independent occupancy of up to 100% each at any time. With half-duplex interface mode, the communication link is used to either exclusively transmit or receive. In this case, the link occupancy is shared so that transmitting and receiving occupancies are 100%'s complement. One could think that full-duplex interfaces always achieve highest performance but in reality it depends on the GPP host application and SDR motherboard capability (half or full-duplex transceiver). Indeed, half and full duplex interfaces can have similar performance when:

- The GPP host application is one-way communication. So, using either half or full duplex is the same, as the communication needed is only to transmit or receive. This implies the link occupancy of the used direction can attain 100%, allowing a maximal acceptable TX or RX I/Q sample rate equal to the peak rate (on the used direction) and always null on the other direction. Thus, using only half-duplex interface is sufficient when the GPP host application is one-way communication;
- The GPP host application is non-overlapped two-way communication. Both directions between SDR device and GPP host cannot be used simultaneously since transmission and reception are separated in time. So, the occupied time for transmission doesn't consume the time of reception and vice versa. Consequently, the sum of TX and RX link occupancies can go to 100% allowing a maximal acceptable TX rate of [peak rate $\times$ TX link occupancy] and maximal acceptable RX rate of [peak rate $\times$ (1 $-$ TX link occupancy)]. Thus, using only half-duplex interface is sufficient when GPP host user application is non-overlapped two-way;
- The SDR device is half-duplex. As SDR device can either receive or transmit, only single direction on the link between SDR device and GPP host is solicited. Such SDR device capability is suitable for GPP host user applications using one-way communication or two-way communication without temporal overlap between transmission and reception. Since only single direction is being used, its link occupancy can go to 100% allowing a maximal acceptable rate equal to the peak rate (null for the unused direction). Hence, using only half duplex interface is sufficient for one-way and non-overlapped two-way communications when SDR device is half duplex.

Full-duplex interfaces become necessary when the GPP host user application is temporally overlapped two-way communication. In addition, SDR devices should also be full-duplex. In this scenario, TX and RX link occupancies are independent and can simultaneously go to 100%. Now, the maximal acceptable TX and RX rate can attain the peak I/Q sample rate. Table 10 shows the maximal supported I/Q

TABLE 10: Maximal I/Q sample rates of selected communication interfaces between GPP host and SDR transceivers.

Interface Host – SDR	Speed (Mbit/s)	Interface mode	Link efficiency	Host application	Max TX. rate (Mbit/s)		Max. RX rate (Mbit/s)	
					Half Dup.SDR	Full Dup.SDR	Half Dup.SDR	Full Dup.SDR
USB2.0 [50]	480	Half dup.	0.9970	TX 1-way Comm.	478.59		0	
				RX 1-way Comm.	0		478.59	
				Non-overlap 2-way Comm.	478.59× TX link Occup.		478.59× (1 − TX link Occup.)	
				Overlap 2-way Comm.	not supported	not supported	not supported	not supported
USB3.0 [50]	5120	Full dup.	0.9808	TX 1-way Comm.	5021.91		0	
				RX 1-way Comm.	0		5021.91	
				Non-overlap 2-way Comm.	5021.91× TX link Occup.		5021.91× (1 − TX link Occup.)	
				Overlap 2-way Comm.	not supported	5021.91	not supported	5021.91
Gig.Eth [51]	1000	Full dup.	0.9881	TX 1-way Comm.	988.1		0	
				RX 1-way Comm.	0		988.1	
				Non-overlap 2-way Comm.	988.1× TX link Occup.		988.1× (1 − TX link Occup.)	
				Overlap 2-way Comm.	not supported	988.1	not supported	988.1
10Gig.Eth [51]	10000	Full dup.	0.9881	TX 1-way Comm.	9881.0		0	
				RX 1-way Comm.	0		9881.0	
				Non-overlap 2-way Comm.	9881.0× TX link Occup.		9881.0× (1 − TX link Occup.)	
				Overlap 2-way Comm.	not supported	9881.0	not supported	9881.0
PCIe 4.X (x4 link) [52]	63015.38	Full dup.	0.99369	TX 1-way Comm.	62617.75		0	
				RX 1-way Comm.	0		62617.75	
				Non-overlap 2-way Comm.	62617.75× TX link Occup.		62617.75× (1 − TX link Occup.)	
				Overlap 2-way Comm.	not supported	62617.75	not supported	62617.75

sample rates of Host-SDR interface solutions for both full and half-duplex SDR transceivers.

D. SYMBOL RATE

This rate, given in Symbol per second (Sym/s), refers to the constant rate at which symbols occur. One symbol can carry one or more bits according to the digital modulation format. For example, in a BPSK system, each symbol represents one bit; in a 64-QAM system, each symbol represents 6 bits. Symbol rate is determined from the symbol duration as [1 / symbol duration], where symbol duration is the sum of the useful symbol duration and the potential guard interval expressed as [useful symbol duration + guard interval]. The guard interval is used between two successive symbols to reduce inter-symbol interference that results from multi-path fading or band-limited channels [80]. It is given by the wireless technology specification. The useful symbol duration is the time used to carry the useful data and is related to the number of samples per symbol and the sampling interval time. It can be formulated as [number of samples per symbol × sampling interval]. The sampling interval parameter is the inverse of the DFE sample rate, [1 / DFE sample rate]. As the DFE sample rate for quadrature sampling systems is equal to the occupied baseband bandwidth, the sampling interval will be the inverse of the occupied baseband bandwidth. The number of samples per symbol parameter can be computed from the frequency domain based on the total number of spectral lines [80].

The number of samples per symbol is equal to the number of spectral lines in quadrature sampling systems and twice in direct-sampling systems. The number of spectral lines is related to the number of carrier/sub-carrier frequencies. Considering quadrature sampling system, when a conventional single-carrier modulation is applied (like in IEEE 802.15.4, ...), the number of spectral lines is equal to one and hence the number of samples per symbol will be one. When multiple sub-carrier modulation technique is used (like in IEEE 802.11ac, ...), the number of spectral lines is equal to the total number of used and unused sub-carriers. In general, the total number of sub-carriers is specified by the used FFT size [80]. To summarize, the useful symbol duration is expressed as [number of spectral lines / occupied baseband bandwidth].

In some wireless systems spread spectrum techniques such as Frequency-Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS), Time-Hopping Spread Spectrum (THSS) and Chirp Spread Spectrum (CSS) are used to prevent interference by transmitting symbols at low power density over a wide band. This band is named as spread occupied baseband bandwidth. The spreading process is achieved by multiplying the symbols with a spreading code, known as chip sequence, having a faster rate than the input symbol rate (symbol rate before spreading). Thus, the spread occupied baseband bandwidth is larger than the original baseband bandwidth by a factor of chip sequence size, [original occupied baseband bandwidth × chip sequence size]. The spread occupied baseband bandwidth is always given as the channel size of wireless systems using spread spectrum techniques. The spreading process has no effect on the useful symbol duration. However, as the given channel size is the spread baseband bandwidth and not the original occupied baseband bandwidth, the useful symbol duration needs to be relied on the spread baseband bandwidth and the chip sequence size. Based on the fact that the symbol duration before spreading is [number of spectral lines / original occupied baseband bandwidth] and the original occupied bandwidth is [spread baseband bandwidth / chip sequence size], the output symbol duration can be written as [number of spectral lines × chip sequence size / spread occupied baseband bandwidth]. It is obvious that the input and output useful symbol duration are the same.

In general, the useful time duration with/without spreading process can be formulated as [number of spectral lines × chip sequence size / channel size]. Fig. 6 depicts an example of possible single/multiple carrier and spreading/non-spreading cases of digital baseband transmitter. Table 11 provides equations of the symbol rate for each path-end. By applying these equations, users can generate the symbol rate for their desired wireless technology and can also be verified from the corresponding specifications [31], [33], [35], [39]–[41].

The software programmer at user space should consider both symbol rate and symbol format. The symbol rate can be either explicitly set or implicitly driven from the DFE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3154364, IEEE Access

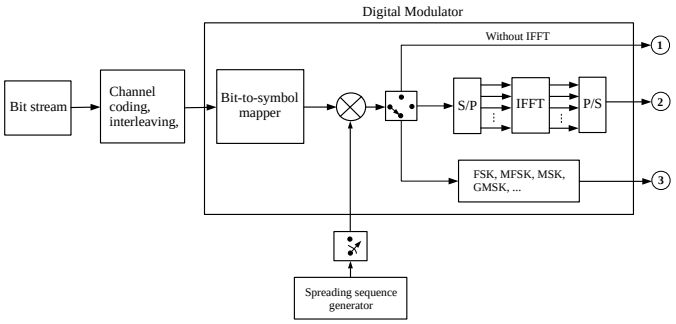

FIGURE 6: Digital baseband transmitter paths.

TABLE 11: Symbol rate of transmitter paths in Fig. 6.

Path-end	Spreading	Symbol rate
①	No	$\frac{1}{\frac{1}{\text{occupied baseband bandwidth}}+\text{guard interval}}$
	Yes	$\frac{1}{\frac{\text{chip sequence size}}{\text{spread occupied baseband bandwidth}}+\text{guard interval}}$
②	No	$\frac{1}{\frac{\text{number of spectral lines}}{\text{occupied baseband bandwidth}}+\text{guard interval}}$
	Yes	$\frac{1}{\frac{\text{number of spectral lines} \times \text{chip sequence size}}{\text{spread occupied baseband bandwidth}}+\text{guard interval}}$
③	No	$\frac{1}{\frac{1}{\text{occupied baseband bandwidth}}+\text{guard interval}}$
	Yes	$\frac{1}{\frac{\text{chip sequence size}}{\text{spread occupied baseband bandwidth}}+\text{guard interval}}$

sample rate and the number of used subcarriers. The symbol format, on the other hand, should be explicitly specified (e.g., complex int16, complex int32, etc). When using a complex int16, the I and Q samples of each symbol are coded each by 16 bits, and so 32 bits per I/Q sample are transmitted to the communication interface. This transmission has a rate of [symbol rate $\times$ symbol format]. The communication interface considers the received data as a data payload and performs its operation related to its technology.

1) Maximal symbol rate at SDR devices

From the equations depicted in Table 11, the theoretical maximal symbol rate at SDR device can be attained when the channel width (bandwidth) is at its maximal value, and the number of spectral line, guard interval and chip sequence size are at their lowest values. The highest channel width can be set to the maximal bandwidth of SDR device; the lowest number of spectral lines can be set to one; the lowest guard interval can be set to zero (without guard interval) and the lowest chip sequence size can be set to one (i.e., without spreading). Consequently, the maximal symbol rate takes the maximal bandwidth of the SDR device. On the other hand, as symbols should traverse the communication interface between the GPP host and the SDR device and vice versa, the maximal symbol rate at the SDR device is bounded by the communication interface speed (after normalization). This latter value is given by [(interface speed $\times$ link efficiency $\times$ link occupancy) / symbol format] in Sym/s. Since, we have two theoretical upper bounds, the maximal symbol rate will take the minimum value between them, i.e., minimum (maximal bandwidth of SDR device, [(interface speed $\times$ link efficiency $\times$ link occupancy) / symbol format]). See Table 13 in subsecion IV-D3 for the maximal theoretical upper bound of the symbol rate at SDR devices.

2) Maximal symbol rate at GPP host

In GPP-based SDR platforms, all symbols are either generated or consumed by GPP host. Therefore, GPP host can impact the maximal symbol rate supported by the platforms. To determine the maximal speed at which symbols are generated or consumed at a GPP host, a continuous data transmission/reception without MAC operations, physical framing, channel coding and spreading is considered. Consequently, the executed physical layer in software should incorporate only bit generator and digital modulator blocks at the transmission side (baseband TX path); or bit sink and digital demodulator blocks at the receiving side (baseband RX path). Hence, only Bit stream and Digital Modulator blocks in Fig. 6 are used to determine the upper bound of symbol rates supported at GPP hosts.

Each symbol generation/consumption requires a time duration in the baseband TX/RX path, when inverted gives the symbol rate. This duration represents the makespan (execution time) of an executable file, created after compiling the TX/RX path blocks, from generating a stream of bits to delivering the corresponding output symbol or from taking a symbol as input and delivering its corresponding bits. Several hardware and software parameters of the GPP host impact the makespan. The hardware parameters include: number of cores, speed of cores, ISA design, pipeline stages, caches, accelerators, hyperthreading support, SIMD support, etc. The software parameters include: user program quality, compiler optimization to enhance the degree of ILP/TLP/DLP, user threads and their degree of TLP, kernel type (non-preemptive and preemptive) and its operations (scheduling, context switching, etc). Multiple parameters can be merged into some big factors such as the minimum number of cycles per instruction (CPI), the number of executed instructions and overheads (due to kernel and memory access operations). The minimum execution time supported by GPP hosts to generate/consume one symbol is achieved when the number of input/output symbols reaches a threshold value (K) to fully benefit from the high degree of parallelism (ILP/TLP/DLP). Above this threshold value, the execution time will increase due to the high CPU load and overhead. Thereby, the minimal makespan can be formulated as in (2):

$$\text{makespan} = \left[\frac{I}{K} \times \text{Inf CPI} \times T \times \frac{1}{\#\text{ of cores} \times 2_{ht}} + \text{Overhead}\right] \quad (2)$$

where I is number of executed instructions to generate/consume K symbols, Inf CPI represents the lowest CPI, T is clock cycle duration given by $\frac{1}{\text{GPP clock speed}}$, and $2_{\text{HT}} = 2$ if hyperthreading or 1 otherwise.

Using (2), numerical results are conducted and illustrated in Table 12. Some parameters such as the threshold number of symbols K and the number of executed instructions are

obtained from experimental data by benchmarking specific functions. To this end, a baseband TX/RX python program is written using GNU Radio software. This program includes source/sink and digital modulator/demodulator. To cover the various types of baseband processing in multiple wireless technologies, different modulation formats are considered such as GFSK, GMSK, BPSK, QPSK, 8-PSK, 16QAM, BPSK OFDM-64 (BPSK with 64-point IFFT), QPSK OFDM-64, and 16QAM OFDM-64. By running the baseband TX/RX python program on different GPP hosts based on Intel core processor, the maximum number of supported symbols (K) that achieves the lowest makespan per symbol is examined. An average maximum value equivalent to fifty six million symbols is obtained. At this threshold value, other parameters such as the total number of instructions, Inf CPI and overhead were also recorded via `perf stat` tool [81].

TABLE 12: Maximal symbol rate at GPP host.

TX path blocks		Total number of executed instructions	Overhead (%)	CPI	GPP cores and speed	Hyperthreading Enabled		Hyperthreading Disabled	
Source	Modulator					Makespan (ns)	Sym. rate (MSym/s)	Makespan (ns)	Sym. rate (MSym/s)
Random	GFSK	39,442,851,301	0.98	0.44	2 core, 2.0 GHz	38.3500	26.0756	76.7000	13.0378
					4 core, 1.5 GHz	25.5726	39.1044	51.1452	19.5522
					4 core, 4.2 GHz	9.1500	109.2896	18.3000	54.6448
					8 core, 3.6 GHz	5.3276	187.7018	10.6552	93.8509
Random	GMSK	38,349,575,894	0.79	0.44	2 core, 2.0 GHz	37.3672	26.7614	74.7344	13.3807
					4 core, 1.5 GHz	24.9114	40.1422	49.8229	20.0711
					4 core, 4.2 GHz	8.8969	112.3987	17.7939	56.1990
					8 core, 3.6 GHz	5.1899	192.6819	10.3798	96.3409
Random	BPSK without FFT	67,051,370,617	1.26	0.44	2 core, 2.0 GHz	65.0243	15.3788	130.0485	7.6894
					4 core, 1.5 GHz	43.3495	23.0683	86.6990	11.5341
					4 core, 4.2 GHz	15.4819	64.5915	30.9639	32.2956
					8 core, 3.6 GHz	9.0311	110.7285	18.0622	55.3642
Random	BPSK OFDM-64	69,664,173,615	1.81	0.44	2 core, 2.0 GHz	67.1817	14.8850	134.3635	7.4425
					4 core, 1.5 GHz	44.7878	22.3275	89.5757	11.1637
					4 core, 4.2 GHz	15.9956	62.5172	31.9913	31.2585
					8 core, 3.6 GHz	9.3308	107.1719	18.6616	53.5859
Random	QPSK without FFT	74,936,632,019	0.82	0.44	2 core, 2.0 GHz	72.9949	13.6996	145.9899	6.8498
					4 core, 1.5 GHz	48.6633	20.5494	97.3266	10.2747
					4 core, 4.2 GHz	17.3797	57.5384	34.7595	28.7691
					8 core, 3.6 GHz	10.1381	98.6378	20.2763	49.3186
Random	QPSK OFDM-64	77,443,318,169	0.43	0.44	2 core, 2.0 GHz	75.7333	13.2042	151.4666	6.6021
					4 core, 1.5 GHz	50.4888	19.8064	100.9777	9.9032
					4 core, 4.2 GHz	18.0317	55.4578	36.0635	27.7288
					8 core, 3.6 GHz	10.5185	95.0706	21.0370	47.5353
Random	8-PSK	82,830,339,448	0.51	0.44	2 core, 2.0 GHz	39.8621	25.0864	79.7242	12.5432
					4 core, 1.5 GHz	26.5747	37.6297	53.1494	18.8148
					4 core, 4.2 GHz	9.4909	105.3640	18.9818	52.6820
					8 core, 3.6 GHz	5.5364	180.6228	11.0728	90.3114
Random	16QAM without FFT	90,791,074,521	1.90	0.44	2 core, 2.0 GHz	87.4756	11.4317	174.9512	5.7159
					4 core, 1.5 GHz	58.3171	17.1476	116.6341	8.5738
					4 core, 4.2 GHz	20.8275	48.0134	41.6550	24.0067
					8 core, 3.6 GHz	12.1494	82.3086	24.2988	41.1543
Random	16QAM OFDM-64	93,191,248,403	0.61	0.44	2 core, 2.0 GHz	90.9688	10.9928	181.9376	5.4964
					4 core, 1.5 GHz	60.6458	16.4892	121.2917	8.2446
					4 core, 4.2 GHz	21.6592	46.1697	43.3184	23.0848
					8 core, 3.6 GHz	12.6346	79.1477	25.2691	39.5740

The results depicted in Table 12, show the minimal makespans and maximal data symbol rates supported by different GPP hosts according to the applied digital modulator. As the consecutive transmit or receive symbols are independent of each other, a high degree of parallelism ILP/TLP/DLP is expected. Thus, GPP hosts having more cores with faster clock speeds and hyperthreading support achieve higher symbol rates by fully exploiting the parallelism. It is important to note that hyperthreading can enhance performance when TLP is very high, otherwise, it might create negative impacts.

3) Maximal symbol rate supported by GPP-based SDR platforms

The maximal symbol rate supported by a given GPP-based SDR platform corresponds to the smallest value between the maximal symbol rate at the used SDR device and at the GPP host. Table 13 gives the maximal symbol rates supported by selected GPP-based SDR platforms according to the type of digital modulation. As the table depicts, for small modulators such as GFSK and GMSK, the maximal symbol rate of the SDR platform is generally limited by the symbol rate of SDR devices, except when X310 is used as SDR device and hyperthreading is enabled for high core and faster clock speed hosts. As the modulator goes higher, the symbol rate starts to be limited by the rate of the GPP host. This is true specially for lower GPP core and slower clock speed. On the other hand, when hyperthreading is disabled, the symbol rate of the SDR platform is mostly limited by the rate of the GPP host for lower core and slower clock speeds. At higher GPP core and clock speed, specially for 8 core and 3.6 GHz host, the SDR platform symbol rate is limited by the symbol rate of the SDR device except for X310 and Lime.

E. BITRATE

This rate, given by bits per second (bit/s), refers to the net bitrate at which data is transferred between the MAC sublayer and the PHY layer of the wireless technology, both implemented in software. It includes the user data and all headers from the application layer to the MAC sublayer. This rate can be expressed based on the wireless PHY layer gross bitrate by excluding from the physical layer frames the error-correction codes and physical layer header. Since the rate of error-correction codes is [code rate] and the rate of physical layer headers is [physical framing], the bitrate can be written as [wireless physical layer gross bitrate $\times$ code rate $\times$ physical framing]. The wireless physical layer gross bitrate is related to the symbol rate, the number of bits per symbol (resulted from bit-to-symbol mapper) and to the number of data subcarriers (if OFDM system is used) to carry data in parallel. It can be expressed by the formula [symbol rate $\times$ # bits per symbol $\times$ # data subcarriers]. All the parameters (code rate, physical framing, symbol rate, number of bits per symbol and number of data subcarriers) are stated in the wireless standard technical specifications. Thus, bitrates and maximal bitrates of wireless technologies can simply be obtained from the bitrate formula. Please refer to section II for the maximal bitrate of selected wireless technologies.

1) Theoretical maximal supported bitrate by GPP-based SDR platforms

The theoretical maximal bitrate that can be achieved by a GPP-based SDR platform depends on the highest values of the wireless physical layer gross bitrate, code rate and physical framing supported by the platform. It can be formulated as $\big[$maximal wireless physical layer gross bitrate $\times$ maximal code rate $\times$ maximal physical framing$\big]$, where the maximal wireless physical layer gross bitrate is computed according to the used digital modulator by $\big[$maximal symbol rate $\times$ involved # of bits per symbol $\times$ involved # of data subcarriers$\big]$. The maximal symbol rate supported by GPP-based SDR platforms is computed in the previous section (see Table 13) using continuous data transmission/reception without MAC operations, physical framing (i.e., maximal physical framing

TABLE 13: Maximal symbol rate of GPP-based SDR platforms.

Modulator	GPP cores and speed	Hyperthreading Enabled						Hyperthreading Disabled					
		Symbol rate (MSym/s)	Hack 14.95	X310 200.00	B210 61.44	Sora 40.00	Lime 160.00	Symbol rate (MSym/s)	Hack 14.95	X310 200.00	B210 61.44	Sora 40.00	Lime 160.00
GFSK	2 core, 2.0 GHz	26.0756	14.95	26.07	26.07	26.07	26.07	13.0378	13.03	13.03	13.03	13.03	13.03
	4 core, 1.5 GHz	39.1044	14.95	39.10	39.10	39.10	39.10	19.5522	14.95	19.55	19.55	19.55	19.55
	4 core, 4.2 GHz	109.2896	14.95	109.28	61.44	40.00	109.28	54.6448	14.95	54.64	54.64	40.00	54.64
	8 core, 3.6 GHz	187.7018	14.95	187.70	61.44	40.00	160.00	93.8509	14.95	93.85	61.44	40.00	93.85
GMSK	2 core, 2.0 GHz	26.7614	14.95	26.76	26.76	26.76	26.76	13.3807	13.38	13.38	13.38	13.38	13.38
	4 core, 1.5 GHz	40.1422	14.95	40.14	40.14	40.00	40.14	20.0711	14.95	20.07	20.07	20.07	20.07
	4 core, 4.2 GHz	112.3987	14.95	112.39	61.44	40.00	112.39	56.1990	14.95	56.19	56.19	40.00	56.19
	8 core, 3.6 GHz	192.6819	14.95	192.68	61.44	40.00	160.00	96.3409	14.95	96.34	61.44	40.00	96.34
BPSK without FFT	2 core, 2.0 GHz	15.3788	14.95	15.37	15.37	15.37	15.37	7.6894	7.68	7.68	7.68	7.68	7.68
	4 core, 1.5 GHz	23.0683	14.95	23.06	23.06	23.06	23.06	11.5341	11.53	11.53	11.53	11.53	11.53
	4 core, 4.2 GHz	64.5915	14.95	64.59	61.44	40.00	64.59	32.2956	14.95	32.29	32.29	32.29	32.29
	8 core, 3.6 GHz	110.7285	14.95	110.72	61.44	40.00	110.72	55.3642	14.95	55.36	55.36	40.00	55.36
BPSK OFDM-64	2 core, 2.0 GHz	14.8850	14.88	14.88	14.88	14.88	14.88	7.4425	7.44	7.44	7.44	7.44	7.44
	4 core, 1.5 GHz	22.3275	14.95	22.32	22.32	22.32	22.32	11.1637	11.16	11.16	11.16	11.16	11.16
	4 core, 4.2 GHz	62.5172	14.95	62.51	61.44	40.00	62.51	31.2585	14.95	31.25	31.25	31.25	31.25
	8 core, 3.6 GHz	107.1719	14.95	107.17	61.44	40.00	107.17	53.5859	14.95	53.58	53.58	40.00	53.58
QPSK without FFT	2 core, 2.0 GHz	13.6996	13.69	13.69	13.69	13.69	13.69	6.8498	6.84	6.84	6.84	6.84	6.84
	4 core, 1.5 GHz	20.5494	14.95	20.54	20.54	20.54	20.54	10.2747	10.27	10.27	10.27	10.27	10.27
	4 core, 4.2 GHz	57.5384	14.95	57.53	57.53	40.00	57.53	28.7691	14.95	28.76	28.76	28.76	28.76
	8 core, 3.6 GHz	98.6378	14.95	98.63	61.44	40.00	98.63	49.3186	14.95	49.31	49.31	40.00	49.31
QPSK OFDM-64	2 core, 2.0 GHz	13.2042	13.20	13.20	13.20	13.20	13.20	6.6021	6.60	6.60	6.60	6.60	6.60
	4 core, 1.5 GHz	19.8064	14.95	19.80	19.80	19.80	19.80	9.9032	9.90	9.90	9.90	9.90	9.90
	4 core, 4.2 GHz	55.4578	14.95	55.45	55.45	40.00	55.45	27.7288	14.95	27.72	27.72	27.72	27.72
	8 core, 3.6 GHz	95.0706	14.95	95.07	61.44	40.00	95.07	47.5353	14.95	47.53	47.53	40.00	47.53
8-PSK	2 core, 2.0 GHz	25.0864	14.95	25.08	25.08	25.08	25.08	12.5432	12.54	12.54	12.54	12.54	12.54
	4 core, 1.5 GHz	37.6297	14.95	37.62	37.62	37.62	37.62	18.8148	14.95	18.81	18.81	18.81	18.81
	4 core, 4.2 GHz	105.3640	14.95	105.36	61.44	40.00	105.36	52.6820	14.95	52.68	52.68	40.00	52.68
	8 core, 3.6 GHz	180.6228	14.95	180.62	61.44	40.00	160.00	90.3114	14.95	90.31	61.44	40.00	90.31
16QAM without FFT	2 core, 2.0 GHz	11.4317	11.43	11.43	11.43	11.43	11.43	5.7159	5.71	5.71	5.71	5.71	5.71
	4 core, 1.5 GHz	17.1476	14.95	17.14	17.14	17.14	17.14	8.5738	8.57	8.57	8.57	8.57	8.57
	4 core, 4.2 GHz	48.0134	14.95	48.01	48.01	40.00	48.01	24.0067	14.95	24.00	24.00	24.00	24.00
	8 core, 3.6 GHz	82.3086	14.95	82.30	61.44	40.00	82.30	41.1543	14.95	41.15	41.15	40.00	41.15
16QAM OFDM-64	2 core, 2.0 GHz	10.9928	10.99	10.99	10.99	10.99	10.99	5.4964	5.49	5.49	5.49	5.49	5.49
	4 core, 1.5 GHz	16.4892	14.95	16.48	16.48	16.48	16.48	8.2446	8.24	8.24	8.24	8.24	8.24
	4 core, 4.2 GHz	46.1697	14.95	46.16	46.16	40.00	46.16	23.0848	14.95	23.08	23.08	23.08	23.08
	8 core, 3.6 GHz	79.1477	14.95	79.14	61.44	40.00	79.14	39.5740	14.95	39.57	39.57	39.57	39.57

TABLE 14: Maximal bitrate (in Mbit/s) of GPP-based SDR platforms.

Modulator	GPP cores and speed	Hyperthreading Enabled					Hyperthreading Disabled				
		Hack	X310	B210	Sora	Lime	Hack	X310	B210	Sora	Lime
GFSK	2 core, 2.0 GHz	14.95	26.07	26.07	26.07	26.07	13.03	13.03	13.03	13.03	13.03
	4 core, 1.5 GHz	14.95	39.10	39.10	39.10	39.10	14.95	19.55	19.55	19.55	19.55
	4 core, 4.2 GHz	14.95	109.28	61.44	40.00	109.28	14.95	54.64	54.64	40.00	54.64
	8 core, 3.6 GHz	14.95	187.70	61.44	40.00	160.00	14.95	93.85	61.44	40.00	93.85
GMSK	2 core, 2.0 GHz	14.95	26.76	26.76	26.76	26.76	13.38	13.38	13.38	13.38	13.38
	4 core, 1.5 GHz	14.95	40.14	40.14	40.00	40.14	14.95	20.07	20.07	20.07	20.07
	4 core, 4.2 GHz	14.95	112.39	61.44	40.00	112.39	14.95	56.19	56.19	40.00	56.19
	8 core, 3.6 GHz	14.95	192.68	61.44	40.00	160.00	14.95	96.34	61.44	40.00	96.34
BPSK without FFT	2 core, 2.0 GHz	14.95	15.37	15.37	15.37	15.37	7.68	7.68	7.68	7.68	7.68
	4 core, 1.5 GHz	14.95	23.06	23.06	23.06	23.06	11.53	11.53	11.53	11.53	11.53
	4 core, 4.2 GHz	14.95	64.59	61.44	40.00	64.59	14.95	32.29	32.29	32.29	32.29
	8 core, 3.6 GHz	14.95	110.72	61.44	40.00	110.72	14.95	55.36	55.36	40.00	55.36
BPSK OFDM-64	2 core, 2.0 GHz	952.32	952.32	952.32	952.32	952.32	476.16	476.16	476.16	476.16	476.16
	4 core, 1.5 GHz	956.80	1428.48	1428.48	1428.48	1428.48	714.24	714.24	714.24	714.24	714.24
	4 core, 4.2 GHz	956.80	4000.64	3932.16	2560.00	4000.64	956.80	2000.00	2000.00	2000.00	2000.00
	8 core, 3.6 GHz	956.80	6858.88	3932.16	2560.00	6858.88	956.80	3429.12	3429.12	2560.00	3429.12
QPSK without FFT	2 core, 2.0 GHz	27.38	27.38	27.38	27.38	27.38	13.68	13.68	13.68	13.68	13.68
	4 core, 1.5 GHz	29.90	41.08	41.08	41.08	41.08	20.54	20.54	20.54	20.54	20.54
	4 core, 4.2 GHz	29.90	115.06	115.06	80.00	115.06	29.90	57.52	57.52	57.52	57.52
	8 core, 3.6 GHz	29.90	197.26	122.88	80.00	197.26	29.90	98.62	98.62	80.00	98.62
QPSK OFDM-64	2 core, 2.0 GHz	1689.60	1689.60	1689.60	1689.60	1689.60	844.80	844.80	844.80	844.80	844.80
	4 core, 1.5 GHz	1913.60	2534.40	2534.40	2534.40	2534.40	1267.20	1267.20	1267.20	1267.20	1267.20
	4 core, 4.2 GHz	1913.60	7097.60	7097.60	7097.60	7097.60	1913.60	3548.16	3548.16	3548.16	3548.16
	8 core, 3.6 GHz	1913.60	12,168.96	7864.32	5120.00	12,168.96	1913.60	6083.84	6083.84	5120.00	6083.84
8-PSK	2 core, 2.0 GHz	44.85	75.24	75.24	75.24	75.24	37.62	37.62	37.62	37.62	37.62
	4 core, 1.5 GHz	44.85	112.86	112.86	112.86	112.86	44.85	56.43	56.43	56.43	56.43
	4 core, 4.2 GHz	44.85	316.08	184.32	120.00	316.08	44.85	158.04	158.04	120.00	158.04
	8 core, 3.6 GHz	44.85	541.86	184.32	120.00	480.00	44.85	270.93	184.32	120.00	270.93
16QAM without FFT	2 core, 2.0 GHz	45.72	45.72	45.72	45.72	45.72	22.84	22.84	22.84	22.84	22.84
	4 core, 1.5 GHz	59.80	68.56	68.56	68.56	68.56	34.28	34.28	34.28	34.28	34.28
	4 core, 4.2 GHz	59.80	192.04	192.04	160.00	192.04	59.80	96.00	96.00	96.00	96.00
	8 core, 3.6 GHz	59.80	329.20	245.76	160.00	329.20	59.80	164.60	164.60	160.00	164.60
16QAM OFDM-64	2 core, 2.0 GHz	2813.44	2813.44	2813.44	2813.44	2813.44	1405.44	1405.44	1405.44	1405.44	1405.44
	4 core, 1.5 GHz	3827.20	4218.88	4218.88	4218.88	4218.88	2109.44	2109.44	2109.44	2109.44	2109.44
	4 core, 4.2 GHz	3827.20	11,816.96	11,816.96	10,240.00	11,816.96	3827.20	5908.48	5908.48	5908.48	5908.48
	8 core, 3.6 GHz	3827.20	20,259.84	15,728.64	10,240.00	20,259.84	3827.20	10,129.92	10,129.92	10,129.92	10,129.92

is set to one), channel coding (i.e., maximal code rate is set to one) and different types of digital modulation techniques. The number of bits per symbol as well as the number of data subcarriers are related to the digital modulation on which the maximal symbol rate is computed. Table 14 illustrates the maximal supported bitrates by SDR platforms.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3154364, IEEE Access

Like the symbol rate, the maximal supported bitrate by a given GPP-based SDR platform increases with GPP core and clock speed, and varies with the used modulation technique and number of data subcarriers. It is also shown that hyperthreading support offers higher maximal bitrate than hyperthreading disabled GPP host. From the SDR device's perspective, those with the highest maximal symbol rate also gives the highest maximal supported bitrate. Thus, for a given wireless technology (i.e., if required modulation type and number of data subcarriers are known), one can easily determine the type of GPP host that can support this requirement. Hence, the candidate SDR platforms with respect to supported maximal symbol rate and bitrate can be determined. For instance, as shown in Table 14, executing GMSK modulator provides a maximal bitrate of (26.76, 40.14, 112.39 and 192.68) Mbit/s, with (2, 4/1.5GHz, 4/3.6GHz and 8) core hyperthreading enabled GPP hosts, respectively. Depending on the SDR device employed, the GPP-based SDR platform will have a maximal bitrate value as illustrated in the table. Therefore, a wireless technology supporting GMSK modulator (such as EC-GSM-IoT having a maximal bitrate of 240 kbit/s), can be implemented using any of the SDR platforms listed in the table with/without hyperthreading support. These findings are used for mapping wireless technologies with SDR platforms in section V.

F. LATENCY

In GPP-based SDR platforms, a latency refers to the time delay spent by a MAC frame (data, control and management) in the transceiver chain between the MAC layer at the GPP host and the antenna connected to the SDR device. As frames can traverse the transceiver chain while being transmitted or received, two types of latencies can be distinguished: GPP-based SDR platform TX latency and RX latency. Both of these latencies contain the same components and results from an accumulation of latencies at each stage of the corresponding path. Since TX/RX path stages are shared between the GPP host, communication interface and SDR device, the GPP-based SDR platform TX/RX latency components can be grouped as: SDR latency, communication interface latency and GPP host latency, as shown in Fig. 7. These latencies are examined below in detail, and we drive the minimal total latency.

1) SDR device latency

This latency consists of three components: DFE latency, ADC/DAC conversion latency and RF front end (RFFE) latency. It may be asymmetrical, providing a varying delay between the case when the SDR device is used for transmitting or for receiving.

At the transmitter side, as shown in Fig. 8, the SDR device latency is the sum of DFE latency, DAC output latency and RFFE latency. The DFE latency is related to both the queuing time of I/Q samples in the SDR buffers (one queue for each type of samples) and the DUCs output latency. The queuing time depends on the arrival rate (i.e., symbol rate), the service rate (i.e., DFE sample rate) and the buffer capacity (limited by dedicated buffer memory space located in the motherboard). The queuing model of the sample buffers can be identified as a D/D/1/buffer_capacity queuing system. Thus, as the I/Q sample rate is always less than or equal to the DFE sample rate, the expected waiting time (in seconds) can be calculated from Little's Law [82] and is given by [1 / DFE sample rate]. The minimal waiting time can be achieved when the DFE sample rate is at its maximum, i.e., the highest DAC sample rate. The DUC output latency is given by [number of cycles × DUC cycle duration]. The values of number of cycles and DUC cycle duration (i.e., [1 / DUC clock rate]) are stated in the motherboard device datasheet [83]. Similarly to the DUC output latency, the DAC conversion latency is given by [number of clock cycles × DAC clock duration] where number of clock cycles depends on the used sample rate and DAC architecture, and the DAC clock duration is [1 / DAC clock rate]. The minimal DAC conversion latency can be formulated as [minimal number of clock cycles / maximal DAC clock rate]. The last part of latency related to the front-end (RFFE latency) is negligible due to high frequency bus. The minimal SDR device latency, therefore, is the sum of the minimum values of DFE and DAC conversion latencies. Table 15 gives theoretical minimal latencies of well-known SDR devices at the transmitting side.

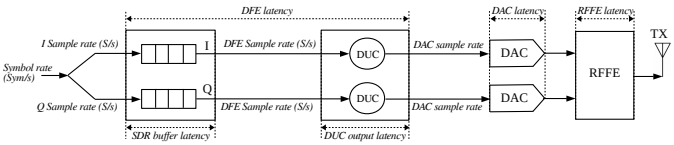

FIGURE 8: Transmitter side SDR device latency.

The SDR latency at the receiver side has the same components as the transmitter side, namely RFFE latency, ADC conversion latency and DFE latency, shown in Fig. 9. The RFFE latency remains negligible for the receiver side due to the high frequency bus. The latency part related to the ADC conversion can be expressed similarly as for DAC conversion latency, by [number of clock cycles × ADC clock duration].

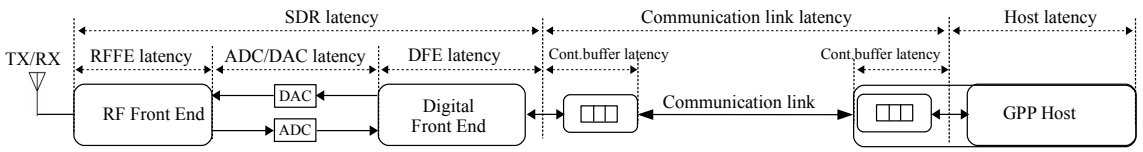

FIGURE 7: GPP-based SDR platform latency.

TABLE 15: Minimal SDR latency for SDR devices.

	SDR device	DFE latency (ns)		DAC latency (ns)	RFFE latency	SDR latency (ns)
		Buffer latency	DUC latency			
TX side	HackRF	50.00	83.33	50.00	–	183.33
	USRP-X3x0	1.25	5.00	1.25	–	7.50
	USRP-B2x0	16.27	16.27	16.27	–	48.81
	Microsoft Sora	25.00	25.00	25.00	–	75.00
	LimeSDR	1.56	6.25	1.56	–	9.37
	SDR device	DFE latency		ADC latency (ns)	RFFE latency	SDR latency (ns)
		Buffer latency (ns)	DDC latency (ns)			
RX side	HackRF	6.55	83.33	50.00	–	6.55
	USRP-X3x0	4.47	5.00	5.00	–	4.47
	USRP-B2x0	3.28	16.27	16.27	–	3.28
	Microsoft Sora	0.95	25.00	25.00	–	0.95
	LimeSDR	0.04	6.25	6.25	–	0.04

The minimal value, therefore, can be formulated as [minimal number of clock cycles / maximal ADC clock rate]. The DFE latency is related to the DFE operations and includes: the DDC output latency and the queuing time at the SDR buffer. The DDC output latency is given by [number of clock cycles $\times$ DDC cycle duration], where the number of clock cycles and DDC cycle duration are found from device motherboard datasheet. The waiting time, on the other hand, represents the time spent by I/Q samples in the SDR buffers before forwarding to the communication interface controller. The waiting time can be given by [(buffer_capacity + 1) (in Sym) / symbol rate (in sym/s)]. The minimal waiting time, therefore, is achieved when the symbol rate attains its maximum, i.e., maximal DFE sample rate, which in turn has its maximum value equivalent to ADC sample rate. Consequently, the minimal waiting time can be approximated to [(buffer_capacity + 1) (in Sym) / ADC sample rate (in Sym/s)]. Table 15 gives theoretical minimal SDR device latency in the receiving side (RX side).

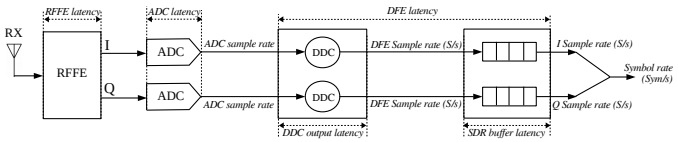

FIGURE 9: Receiver side SDR device latency.

2) Communication interface latency

This latency is related to data exchange between GPP host and SDR device through communication interfaces such as Gigabit Ethernet, USB3.0, PCIe, etc. The data packets are used to carry symbols, where each symbol is pushed in one packet as data payload and followed by a set of header fields. The time spent by a data packet to travel from one communication interface controller, of GPP host or SDR device, to another is defined as the communication interface latency. It includes the waiting time of data packets at both communication interface controllers and the propagation time. The waiting time at each communication interface controller combines the queuing time and the service time. Fig. 10 depicts the components of the communication interface latency.

The queuing system consists of two symmetric queuing networks that can work either simultaneously in case of Full-duplex interfaces, or only one at a time in case of Half-duplex

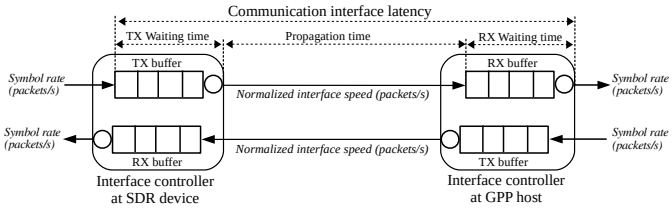

FIGURE 10: Communication interface latency.

interfaces. Each queuing network comprises a TX buffer linked to a RX buffer through the communication interface link. From the TX buffer to RX buffer, the communication interface controller collects I/Q samples (generated by the user application or received from the DFE) and creates data packets according to the used interface standard (data packet size, header fields, etc). These data packets are queued at the TX buffer waiting for transmission to the RX buffer. When a packet is transmitted, it will travel over the communication link with signal propagation speed in a medium. Finally, the communication interface controller intercepts the received packets, extracts the I/Q samples and puts them into the RX buffer for delivery (to the DFE or to the user application). For the queuing network parameters:

- The arrival rate at the TX buffer, expressed in packets/s, takes the same value of the symbol rate as each symbol is carried by a single data packet;
- The service rate of the TX Buffer can be derived from the interface speed and the TX link occupancy, and converted into packets/s using the following formula: [(interface speed $\times$ link efficiency $\times$ TX link occupancy) / payload size] (in packets/s). The resulting speed is called normalized interface speed. For more details about the TX link occupancy computation, see section IV-C;
- The propagation time can be given by [length of the medium / speed of signal propagation] where speed of signal propagation is $3x10^8$m/sec; or can be found from datasheet of the used communication interface [50];
- The arrival rate at the RX buffer, expressed in Sym/s, takes the same arrival rate of the TX buffer (i.e., symbol rate) if this later is less than or equal to TX buffer service rate (normalized interface speed). Otherwise, it takes the normalized interface speed;
- The service rate of the RX buffer, expressed in Sym/s, is symbol rate.

The communication interface latency can, therefore, be regarded as the total waiting time of the queuing network. Thus, it's the result of the sum of the waiting time of data packets at TX/RX buffers and the propagation time. Two cases need to be distinguished: when the symbol rate is less than or equal, and when it's greater than the normalized interface speed (i.e., the interface speed, averaged and converted). In the first case, the expected waiting time in the TX buffer is [1 / normalized interface speed] seconds, and the expected waiting time in the RX buffer is [1 / symbol rate] seconds. Hence,

the total waiting time is given by [(1 / normalized interface speed) + (length of the medium / speed of signal propagation) + (1 / symbol rate)] (seconds). The minimal waiting time is achieved when the symbol rate attains its maximum, i.e., the normalized interface speed, the limit imposed by the condition of the 1st case. Consequently, the total minimum waiting time is [(2 / normalized interface speed) + (length of the medium / speed of signal propagation)] (seconds). In the second case, the expected waiting time in the TX buffer is [(TX buffer capacity (in packets) + 1) / normalized interface speed] seconds, and the expected waiting time in the RX buffer is [1 / symbol rate] seconds. Hence, the total waiting time is given by $\left[\left((\text{TX buffer capacity (in packets)} + 1) / \text{normalized interface speed}\right) + (\text{length of the medium / speed of signal propagation}) + (1 / \text{symbol rate})\right]$ (seconds). The minimal waiting time is achieved when the symbol rate attains its maximum, i.e., the DFE sample rate, and it is given by $\left[\left((\text{TX buffer capacity (in packets)} + 1) / \text{normalized interface speed}\right) + (\text{length of the medium / speed of signal propagation}) + (1 / \text{DFE sample rate})\right]$ (seconds). Table 16 gives theoretical minimal interface latency with 100% TX link occupancy.

TABLE 16: Minimal communication interface latency.

SDR platform	Comm. interface	Propagation time (μs)	Normalized int. speed (kpacket/s)	Max. Sym rate (kpacket/s)	Min.waiting time (μs)		Comm.interface latency (μs)
					TX buffer	RX buffer	
GPP host + HackRF	USB2.0	0.026	116.8432	116.843	8.5584	8.5584	17.1428
				156.250	2148.18	6.4000	2154.60
GPP host + USRP B2x0	USB3.0	0.026	613.026	613.026	1.631	1.631	3.288
				240.000	1.6312	1.6312	3.2884
GPP host + USRP X3x0	10Gig Eth	0.025	813.6528	813.6528	1.2290	1.2290	2.4830
				527.0090	1.2290	1.2290	2.4830
GPP host + Sora	PCIe (x8)	–	1903.6207	1903.62	0.5253	0.5253	1.0506
				153.26	0.5253	0.5253	1.0506
GPP host + LimeSDR	PCIe (x4)	–	951.8103	951.8103	1.0506	1.0506	2.1012
				613.0268	1.0506	1.0506	2.1012

3) GPP host latency

This latency is the time a wireless MAC frame (data, management or control) passes in GPP host between the MAC layer and the interface controller due to TX and RX activities. It mainly includes the processing time due to the MAC layer operations such as access mechanism, MAC framing, generating/transmitting/receiving control and management frames, transmitting/receiving data frames, etc., and the PHY layer operations such as channel coding, digital modulation, PHY framing, etc. The time to forward/retrieve symbols to/from the interface controller should also be added. To perform all these operations require a GPP time and additional extra-time related to the kernel operations and external memory read/write should be considered. Estimating the GPP host latency is a very complex work, due to the high number of hardware and software factors that affect the execution time of the user program (wireless technology implementation). Indeed, the GPP host hardware and software configurations (see section III-C) affects the execution speed and hence impact the response time of all tasks. Consequently, we illustrate the minimum GPP host latency experimentally by benchmarking existing software implementations of selected wireless technologies.

The selected wireless technologies were run on Intel x86_64 microprocessor with different number of cores and speeds (2 GHz Dual-core, 1.5 GHz Quad-core and 3.6GHz Octa-core). Each core has L1/L2/L3 cache sizes of 64KB/256KB/6144KB and access time of 1.2ns/3.6ns/12ns. All the applications listed below are tested on Ubuntu 18.04.2 LTS: NFC based on `gr-nfc` [84], IEEE 802.15.6 based on a prototype proposed in [85] for NB-WBAN, IEEE 802.15.1 based on `scapy-radio` for Bluetooth [86], IEEE 802.15.4 based on `gr-IEEE 802.15.4` [15], IEEE 802.11ac and IEEE 802.11ah are based on GNU Radio implementation of `gr-IEEE 802.11` [16]; EC-GSM-IoT based on `gr-GSM` [87]. LTE is measured using eNodeB and user equipment (UE) PHY downlink (DL) shared channel (PDSCH) and NB-IoT using Narrowband PDSCH (NPDSCH) PHY modules from `srsRAN` [14]; and LoRa based on `gr-lora` [24]. The parameters involved in the experimental latency result like number of executed instructions and CPI are measured using `perf stat` tool. The results are depicted in Table 17.

TABLE 17: GPP host latency for wireless technologies.

Wireless tech.	TX/RX	Hyperthreading Enabled				Hyperthreading Disabled			
		Total # of executed instructions	CPI	GPP cores and speed	latency (ms)	Total #. of executed instructions	CPI	GPP cores and speed	latency (ms)
NFC	TX	558,321,456	0.8620	2 core, 2.0 GHz	240.636	559,515,043	0.8695	2 core, 2.0 GHz	243.249
				4 core, 1.5 GHz	320.848			4 core, 1.5 GHz	324.332
				8 core, 3.6 GHz	133.687			8 core, 3.6 GHz	135.138
	RX	972,461,581	0.8474	2 core, 2.0 GHz	103.008	926,670,202	0.8620	2 core, 2.0 GHz	199.697
				4 core, 1.5 GHz	68.672			4 core, 1.5 GHz	133.132
				8 core, 3.6 GHz	14.306			8 core, 3.6 GHz	27.736
IEEE802.15.6	TX	876,981,272	0.8772	2 core, 2.0 GHz	384.643	878,403,793	0.8928	2 core, 2.0 GHz	392.119
				4 core, 1.5 GHz	512.858			4 core, 1.5 GHz	522.826
				8 core, 3.6 GHz	213.691			8 core, 3.6 GHz	217.844
	RX	796,701,770	0.8695	2 core, 2.0 GHz	346.366	796,549,347	0.9009	2 core, 2.0 GHz	358.806
				4 core, 1.5 GHz	461.821			4 core, 1.5 GHz	478.408
				8 core, 3.6 GHz	192.426			8 core, 3.6 GHz	199.336
IEEE802.15.1	TX	795,768,645	0.8849	2 core, 2.0 GHz	88.022	795,694,115	0.9009	2 core, 2.0 GHz	179.210
				4 core, 1.5 GHz	58.681			4 core, 1.5 GHz	119.473
				8 core, 3.6 GHz	12.225			8 core, 3.6 GHz	24.890
	RX	797,964,859	0.8772	2 core, 2.0 GHz	349.987	799,065,806	0.9009	2 core, 2.0 GHz	359.939
				4 core, 1.5 GHz	466.649			4 core, 1.5 GHz	479.919
				8 core, 3.6 GHz	194.437			8 core, 3.6 GHz	199.966
IEEE802.15.4	TX	861,073,678	0.8772	2 core, 2.0 GHz	94.417	856,394,067	0.8928	2 core, 2.0 GHz	191.147
				4 core, 1.5 GHz	62.944			4 core, 1.5 GHz	127.431
				8 core, 3.6 GHz	13.113			8 core, 3.6 GHz	26.548
	RX	858,617,170	0.8772	2 core, 2.0 GHz	94.147	855,396,059	0.8928	2 core, 2.0 GHz	190.924
				4 core, 1.5 GHz	62.765			4 core, 1.5 GHz	127.283
				8 core, 3.6 GHz	13.076			8 core, 3.6 GHz	26.517
IEEE802.11a	TX	1,515,777,533	0.8064	2 core, 2.0 GHz	152.790	1,543,797,133	0.8130	2 core, 2.0 GHz	313.776
				4 core, 1.5 GHz	203.721			4 core, 1.5 GHz	209.184
				8 core, 3.6 GHz	84.884			8 core, 3.6 GHz	87.16
	RX	1,349,807,636	0.8403	2 core, 2.0 GHz	189.041	1,377,853,186	0.8547	2 core, 2.0 GHz	294.413
				4 core, 1.5 GHz	252.054			4 core, 1.5 GHz	261.700
				8 core, 3.6 GHz	105.023			8 core, 3.6 GHz	109.042
IEEE802.11ah	TX	1,391,308,356	0.8264	2 core, 2.0 GHz	143.722	1,421,664,038	0.8474	2 core, 2.0 GHz	301.179
				4 core, 1.5 GHz	191.630			4 core, 1.5 GHz	200.786
				8 core, 3.6 GHz	79.846			8 core, 3.6 GHz	83.661
	RX	1,346,778,585	0.8403	2 core, 2.0 GHz	188.616	1,378,183,531	0.8547	2 core, 2.0 GHz	294.483
				4 core, 1.5 GHz	251.488			4 core, 1.5 GHz	261.763
				8 core, 3.6 GHz	104.787			8 core, 3.6 GHz	109.068
LTE	TX	1,767,690,400	0.4464	2 core, 2.0 GHz	131.516	1,773,284,720	0.4424	2 core, 2.0 GHz	196.125
				4 core, 1.5 GHz	87.677			4 core, 1.5 GHz	174.334
				8 core, 3.6 GHz	36.532			8 core, 3.6 GHz	72.639
	RX	2,045,481,270	0.4098	2 core, 2.0 GHz	119.748	2,054,979,907	0.4081	2 core, 2.0 GHz	239.611
				4 core, 1.5 GHz	79.832			4 core, 1.5 GHz	159.74
				8 core, 3.6 GHz	33.263			8 core, 3.6 GHz	33.279
NB-IoT	TX	132,609,522	0.7194	2 core, 2.0 GHz	13.628	123,277,890	0.7936	2 core, 2.0 GHz	27.952
				4 core, 1.5 GHz	9.086			4 core, 1.5 GHz	18.635
				8 core, 3.6 GHz	3.786			8 core, 3.6 GHz	3.882
	RX	162,404,628	0.7092	2 core, 2.0 GHz	16.454	161,738,446	0.7246	2 core, 2.0 GHz	33.484
				4 core, 1.5 GHz	10.969			4 core, 1.5 GHz	22.323
				8 core, 3.6 GHz	4.571			8 core, 3.6 GHz	4.651
EC-GSM-IoT	TX	893,349,079	0.8695	2 core, 2.0 GHz	97.096	893,315,010	0.8772	2 core, 2.0 GHz	195.904
				4 core, 1.5 GHz	64.731			4 core, 1.5 GHz	130.603
				8 core, 3.6 GHz	13.486			8 core, 3.6 GHz	27.209
	RX	1,317,231,292	0.7633	2 core, 2.0 GHz	125.680	1,222,540,786	0.7874	2 core, 2.0 GHz	240.657
				4 core, 1.5 GHz	83.787			4 core, 1.5 GHz	160.438
				8 core, 3.6 GHz	32.858			8 core, 3.6 GHz	33.425
LoRa	TX	946,374,830	0.7692	2 core, 2.0 GHz	103.993	1,184,932,351	0.7246	2 core, 2.0 GHz	195.492
				4 core, 1.5 GHz	69.329			4 core, 1.5 GHz	130.328
				8 core, 3.6 GHz	28.887			8 core, 3.6 GHz	31.03
	RX	572,913,761	0.8403	2 core, 2.0 GHz	240.709	571,508,495	0.8695	2 core, 2.0 GHz	248.463
				4 core, 1.5 GHz	320.946			4 core, 1.5 GHz	331.284
				8 core, 3.6 GHz	133.728			8 core, 3.6 GHz	138.035

From Table 17, we can see that under the same hardware and kernel settings at the GPP host, wireless technologies provide different latencies according to the total number of executed instructions and the inherent parallelism in the instruction code. This parallelism defines on one hand the level of TLP exploited by the scheduler to split execution of threads between physical/virtual cores, and on the other hand by the level of TLP exploited by the processor to perform multiple instructions simultaneously within the same core which reduces (on average)

the number of required CPI. Based on the total number of executed instructions and parallelism, the latency increases when the number of instructions increases and parallelism degree decreases and vice versa. Thereby, since the total number of executed instructions of IEEE 802.15.4 is 861,073,678/858,617,170 for TX/RX and the CPI is 0.8772 for both TX and RX, the latencies under hyperthreading for Dual/Quad/Octa-core are 94.417ms/62.944ms/13.113ms for TX and 94.147ms/62.765ms/13.076ms for RX, respectively. Also, we note that the latency is lower with hyperthreading enabled than hyperthreading disabled for all tested technologies, from 0.009% upto 54.31% improvements. The major contributor of the high percentage improvements provided by some tested technologies is due to the large number of threads in the program; and with high-TLP, theoretically, we might expect to have upto 50% improvement due to hyperthreading. However, this depends on the resource contention between threads on hyperthreaded cores. High resource contention often leads to latency degradation.

4) Minimal theoretical GPP-based SDR platform latency

The latency of GPP-based SDR platforms related to TX/RX operations is the result of cumulative latencies over three stages: SDR device, communication interface and GPP host. The minimal latency at both SDR device and communication interface stages is theoretically expressed by modeling the internal architecture using queuing theory. While the minimal latency at the SDR device varies from several nanoseconds to few milliseconds depending on the buffer size, it varies from microseconds to few milliseconds at the communication interface. At the GPP host stage, the minimal latency is investigated experimentally based on multiple parameters such as number and speed of cores, hyperthreading, number of executed instructions, number of threads, degree of ILP/TLP/DLP, kernel scheduler, I/O management (e.g., cache/memory/disk access), etc. Table 18 gives the total minimal latency of GPP-based SDR platforms related to TX/RX operations of some wireless technologies. For the communication interface a link occupancy of 100% and symbol rate less than or equal to the normalized interface speed is used.

Table 18 demonstrates that large part of the total latency is due to the GPP host. Comparing the minimal GPP-based SDR platform latency for TX/RX paths with latencies at the three components (SDR device, communication interface and GPP host), we see that for some wireless technologies the GPP host contributes upto 99% of the total (such as in NFC, IEEE802.15.6, IEEE802.15.1, etc). It is shown that the latency at the GPP host stage could be minimized by using high TLP, high clock rate, hyperthreading (with lower resource contention), etc. One can also see that, TX and RX latencies are different. This is due to the specific operations on each path. Moreover, there's significant difference in TX and RX latencies of SDR devices (see section IV-F1) that also contributes to the total latency. The values in Table 18 indicate the capability of the SDR platforms in executing wireless technologies. For instance, to perform a TX operation of NFC using Hack SDR device and USB2.0 communication interface on a (2 core, 2 GHz, hyperthreading enabled) GPP host, it takes 122.855ms (shaded cell). However, to determine whether a given SDR platform meets the latency required by a wireless technology, one needs to carefully map the two (see section V for the mapping). Note that the latency analysis illustrated in this paper can also be exploited to investigate other SDR platforms and wireless technologies.

V. MAPPING PARAMETERS OF WIRELESS TECHNOLOGY WITH GPP-BASED SDR PLATFORM

In the previous sections, we have carried out investigations to determine the requirements of well-known wireless technologies and the minimum performance of GPP-SDR platform in terms of frequency, bandwidth, symbol rate, bitrate and latency. In this section, we intersect the wireless standard requirements with the SDR platform performance to build a list of possible GPP-based SDR platforms that can successfully implement a given wireless technology.

A. MATCHING CONDITIONS

A successful matching between a wireless technology requirements and a GPP-based SDR platform performance can occur when matching conditions are satisfied. These conditions are applied to perform a simple comparison between similar metrics of the two sets. Further details on the matching conditions are discussed below.

1) Frequency band matching

Wireless technologies are defined to operate in a single or multiple frequency bands of the radio spectrum. On the other hand, SDR device daughterboards are defined to operate in a wide contiguous frequency band. Given a wireless technology, frequency band matching consists of ensuring that the SDR device daughterboard frequency range covers the frequency bands of the wireless technology.

2) Bandwidth matching

Wireless technologies divide their operating frequency bands to single/multiple overlapping/non-overlapping channels of predefined widths. However, the really occupied bandwidth can be less than the channel width depending on the type of modulation. Given a wireless technology, bandwidth matching consists of ensuring that the maximal SDR platform bandwidth is at least equal to the real occupied bandwidth in the channel width defined by the standard.

3) Symbol rate matching

The specifications of wireless technologies provide the symbol duration, and hence the symbol rate, either explicitly or implicitly through related PHY parameters (see equations in Table 11). Given a wireless technology and a GPP-based SDR platform, symbol rate matching ensures that the maximal supported symbol rate by the GPP-based SDR platform

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3154364, IEEE Access

TABLE 18: Theoretical minimal latency (ms) on GPP-based SDR platforms.

Wireless tech.	TX/RX	GPP-based SDR platform	Hyperthreading Enabled					Hyperthreading Disabled				
		SDR device	Hack	X310	B210	Sora	Lime	Hack	X310	B210	Sora	Lime
		Comm. int.	USB2.0	10Gig.	USB3.0	PCIe-x8	PCIe-x4	USB2.0	10Gig.	USB3.0	PCIe-x8	PCIe-x4
NFC	TX	2 core, 2.0 GHz	**122.855**	122.840	122.841	122.839	122.840	123.122	123.107	123.108	123.106	123.107
		4 core, 1.5 GHz	163.782	163.784	163.785	163.783	163.784	164.153	164.138	164.139	164.137	164.138
		8 core, 3.6 GHz	68.264	68.264	68.250	68.248	68.249	68.414	68.399	68.400	68.398	68.399
	RX	2 core, 2.0 GHz	77.903	75.808	74.619	72.287	71.378	119.882	117.787	116.598	114.266	113.357
		4 core, 1.5 GHz	69.972	67.877	66.688	64.356	63.447	97.208	95.113	93.924	91.592	90.683
		8 core, 3.6 GHz	19.781	17.686	16.497	14.165	13.256	31.758	29.663	28.474	26.142	25.233
IEEE802.15.6	TX	2 core, 2.0 GHz	192.961	192.946	192.947	192.945	192.946	193.277	193.262	193.263	193.261	193.262
		4 core, 1.5 GHz	257.273	257.258	257.2593	257.257	257.258	257.694	257.679	257.680	257.678	257.679
		8 core, 3.6 GHz	107.211	107.196	107.197	107.195	107.196	107.389	107.374	107.375	107.373	107.374
	RX	2 core, 2.0 GHz	181.849	179.754	178.565	176.233	175.324	181.819	179.724	178.535	176.203	175.294
		4 core, 1.5 GHz	240.274	238.179	236.990	234.658	233.749	240.233	238.138	236.949	234.617	233.708
		8 core, 3.6 GHz	103.949	101.854	100.665	98.333	97.424	103.935	101.840	100.651	98.319	97.410
IEEE802.15.1	TX	2 core, 2.0 GHz	46.731	46.716	46.717	46.715	46.716	93.467	93.452	93.453	93.451	93.452
		4 core, 1.5 GHz	31.155	31.140	31.141	31.139	31.140	62.302	62.287	62.288	62.286	62.287
		8 core, 3.6 GHz	6.509	6.494	6.495	6.493	6.494	13.006	12.991	12.992	12.990	12.991
	RX	2 core, 2.0 GHz	182.127	180.032	178.843	176.511	175.602	182.373	180.278	179.089	176.757	175.848
		4 core, 1.5 GHz	240.644	238.549	237.360	235.028	234.119	240.971	238.876	237.687	235.355	234.446
		8 core, 3.6 GHz	104.104	102.009	100.820	98.488	97.579	104.242	102.147	100.958	98.626	97.717
IEEE802.15.4	TX	2 core, 2.0 GHz	47.420	47.405	47.406	47.404	47.405	94.353	94.338	94.339	94.337	94.338
		4 core, 1.5 GHz	31.612	31.597	31.598	31.596	31.597	62.886	62.871	62.872	62.870	62.871
		8 core, 3.6 GHz	8.802	8.787	8.788	8.786	8.787	13.134	13.119	13.120	13.118	13.119
	RX	2 core, 2.0 GHz	53.835	51.740	50.551	48.219	47.310	100.793	98.698	97.509	95.177	94.268
		4 core, 1.5 GHz	38.072	35.977	34.788	32.456	31.547	69.363	67.268	66.079	63.747	62.838
		8 core, 3.6 GHz	15.327	13.232	12.043	9.711	8.802	19.669	17.574	16.385	14.053	13.144
IEEE802.11a	TX	2 core, 2.0 GHz	83.392	83.377	83.378	83.376	83.377	169.857	169.842	169.843	169.841	169.842
		4 core, 1.5 GHz	111.182	111.167	111.168	111.166	111.167	113.240	113.225	113.226	113.224	113.225
		8 core, 3.6 GHz	46.340	46.325	46.326	46.324	46.325	47.200	47.185	47.186	47.184	47.185
	RX	2 core, 2.0 GHz	105.561	103.466	102.277	99.945	99.036	208.675	206.580	205.391	203.059	202.150
		4 core, 1.5 GHz	138.556	136.461	135.272	132.940	132.031	141.302	139.207	138.018	135.686	134.777
		8 core, 3.6 GHz	61.567	59.472	58.283	55.951	55.042	62.713	60.618	59.429	57.097	56.188
IEEE802.11ah	TX	2 core, 2.0 GHz	76.547	76.532	76.533	76.531	76.532	156.423	156.408	156.409	156.407	156.408
		4 core, 1.5 GHz	102.054	102.039	102.040	102.038	102.039	104.284	104.269	104.270	104.268	104.269
		8 core, 3.6 GHz	42.537	42.522	42.523	42.521	42.522	43.468	43.453	43.454	43.452	43.453
	RX	2 core, 2.0 GHz	105.338	103.243	102.054	99.722	98.813	208.723	206.628	205.439	203.107	202.198
		4 core, 1.5 GHz	138.260	136.165	134.976	132.644	131.735	141.334	139.239	138.050	135.718	134.809
		8 core, 3.6 GHz	61.443	59.348	58.159	55.827	54.918	62.727	60.632	59.443	57.111	56.202
LTE	TX	2 core, 2.0 GHz	129.663	129.648	129.649	129.647	129.648	195.112	195.097	195.098	195.096	195.097
		4 core, 1.5 GHz	86.445	86.430	86.431	86.429	86.430	173.427	173.412	173.413	173.411	173.412
		8 core, 3.6 GHz	36.033	36.018	36.019	36.017	36.018	36.151	36.136	36.137	36.135	36.136
	RX	2 core, 2.0 GHz	135.155	133.060	131.871	129.539	128.630	264.952	262.857	261.668	259.336	258.427
		4 core, 1.5 GHz	92.290	90.195	89.006	86.674	85.765	178.816	176.721	175.532	173.200	172.291
		8 core, 3.6 GHz	42.289	40.194	39.005	36.673	35.764	42.459	40.364	39.175	36.843	35.934
NB-IoT	TX	2 core, 2.0 GHz	8.367	8.352	8.353	8.351	8.352	15.559	15.544	15.545	15.543	15.544
		4 core, 1.5 GHz	5.582	5.567	5.568	5.566	5.567	10.371	10.356	10.357	10.355	10.356
		8 core, 3.6 GHz	2.340	2.325	2.326	2.324	2.325	2.181	2.166	2.167	2.165	2.166
	RX	2 core, 2.0 GHz	16.790	14.695	13.506	11.174	10.265	26.944	24.849	23.66	21.328	20.419
		4 core, 1.5 GHz	13.380	11.285	10.096	7.764	6.855	20.145	18.05	16.861	14.529	13.62
		8 core, 3.6 GHz	9.410	7.315	6.126	3.794	2.885	9.403	7.308	6.119	3.787	2.878
EC-GSM-IoT	TX	2 core, 2.0 GHz	52.456	52.441	52.442	52.44	52.441	52.469	52.454	52.455	52.453	52.454
		4 core, 1.5 GHz	34.972	34.957	34.958	34.956	34.957 8	34.978	34.963	34.964	34.962	34.963
		8 core, 3.6 GHz	7.304	7.289	7.290	7.288	7.289	7.307	7.292	7.293	7.291	7.292
	RX	2 core, 2.0 GHz	91.836	89.741	88.552	86.22	85.311	85.728	83.633	82.444	80.112	79.203
		4 core, 1.5 GHz	74.775	72.68	71.491	69.159	68.25	69.885	67.79	66.601	64.269	63.36
		8 core, 3.6 GHz	25.523	23.428	22.239	19.907	18.998	24.169	22.074	20.885	18.553	17.644
LoRa	TX	2 core, 2.0 GHz	59.518	59.503	59.504	59.502	59.503	122.942	122.927	122.928	122.926	122.927
		4 core, 1.5 GHz	39.682	39.667	39.668	39.666	39.667	81.96	81.945	81.946	81.944	81.945
		8 core, 3.6 GHz	16.549	16.534	16.535	16.533	16.534	17.095	17.08	17.081	17.079	17.08
	RX	2 core, 2.0 GHz	132.616	130.521	129.332	127.000	126.091	132.31	130.215	129.026	126.694	125.785
		4 core, 1.5 GHz	174.629	172.534	171.345	169.013	168.104	174.221	172.126	170.937	168.605	167.696
		8 core, 3.6 GHz	76.598	74.503	73.314	70.982	70.073	76.43	74.335	73.146	70.814	69.905

is greater than or equal to that defined by the wireless standard.

4) Bitrate matching

Based on PHY parameters (spectral lines, code rate, PHY framing, digital modulation) and symbol rate of wireless technologies, specifications provide a list of supported bitrates. On the other hand, GPP-based SDR platforms support a maximum bitrate according to the used SDR device, communication interface and GPP host capabilities. Given a wireless technology and a GPP-based SDR platform, full bitrate matching ascertains that the maximal supported bitrate by the GPP-based SDR platforms is greater than or equal to the highest bitrate of the wireless technology. However, partial matching can also occur when the maximal supported bitrate by the SDR platform is between the highest and lowest bitrates of the wireless technology. In this case, the SDR platform can still perform the implemented wireless technology but in lower bitrates.

5) Latency matching

Wireless devices operate in a shared wireless medium, and hence, require a MAC protocol to organize access to a channel. In general, MAC protocols can be classified as either contention-based or contention-free protocols [88]. Despite their significant differences in terms of coordination, they define inter-frame times to handle waiting periods between transmission of frames (data, control and management). For example, IEEE 802.15.4 uses slotted CSMA/CA as a contention based protocol and Guaranteed Time Slot (GTS) allocation mechanism as a contention-free protocol. Both protocols define inter-frame times to cover the period from receiving a data frame and transmitting an explicit acknowledgment frame. The smallest inter-frame time among all inter-frame times is defined as the latency. Wireless devices should respect the latency to ensure optimized network performance. Examples of latency for the well-known wireless technologies are given in Tables 1, 2 and 3.

As GPP-based SDR platforms will play the role of wireless device, they should fulfill the latency constraint by completing execution of all operations within the dedicated time period as defined by the specific MAC protocol. The concerned operations are often related to transmission and/or reception of frames. Thus, during the latency period, frames should traverse the transceiver chain in one or both directions based on the objective of MAC operations. Consequently, the latency of GPP-based SDR platforms related to the implemented wireless technology can include latency values of only in TX or RX path, or both TX and RX paths depending on the covered operations desired by the implemented technology.

Given a wireless technology and a GPP-based SDR platform, full latency matching occurs when the minimal latency supported by the GPP-based SDR platform is below the latency of the wireless technology. Such matching allows the GPP-based SDR platform to perform the TX/RX operations efficiently and ensure normal communication with legacy transceivers. However, when the minimal latency supported by the GPP-based SDR platform exceeds the latency of wireless technology and/or some other (perhaps all) inter-frame times, frame exchange with legacy transceivers will be affected. In other words, as TX/RX operations at the GPP-based SDR platform are delayed, the overall network (in presence of one or more legacy transceivers) performance will degrade. To illustrate this, consider the following two cases:

- Case 1: Assume that a GPP-based SDR platform has won the channel access. A delayed frame transmission may cause collisions at legacy receivers in the presence of concurrent transmissions;
- Case 2: Assume that a GPP-based SDR platform is receiving a unicast data frame from a legacy transceiver. A delayed processing of the received frame and transmitting a response (e.g., ACK) may cause the data frame retransmission after response timeout, or starting concurrent transmissions by other devices;

It should be noted that with or without matching, GPP-based SDR platforms can still perform some tasks successfully such as: a) receiving broadcast/multicast frames as they are not acknowledged, and b) acting as a wireless sniffer.

B. MAPPING PERFORMANCE OF GPP-BASED SDR PLATFORMS WITH WIRELESS TECHNOLOGY REQUIREMENTS

Based on the theoretical performance of GPP-based SDR platforms calculated in section IV and the requirements of wireless technologies listed in section II, the matching conditions can be performed. For this purpose, a list of selected wireless technologies and GPP-based SDR platforms are considered. List of wireless technologies include: NFC, IEEE802.15.6, IEEE802.15.1, IEEE802.15.4, IEEE802.11ac, IEEE802.11ah, LTE, NB-IoT, EC-GSM-IoT, and LoRa. The list of GPP-based SDR platform contains several SDR devices connected to different GPP hosts via their fastest supported communication interface. The SDR devices and their fastest supported communication interface considered are: HackRF with USB2.0, USRP-X310 with 10Gigabit Ethernet, USRP-B210 with USB3.0, Microsoft Sora with PCIe (x8), and LimeSDR with PCIe (x4). The GPP hosts are: 2 GHz Dual-core processor, 1.5 GHz Quad-core processor and 8 GHz Octa-core processor.

To match the two lists, according to the matching conditions, a mapping table is created (see Table 19). In this Table, as the maximal frequency band and maximal bandwidth of SDR devices are determined irrespective of the wireless technology and GPP host type, they are defined only once. Whereas, the other three parameters (maximal symbol rate, maximal bitrate and minimal latency), as they are determined for each technology on the three GPP host types, their values are indicated separately as per the technology. Moreover, the minimal latency supported by GPP-based SDR platform for each wireless technology is set based on TX path and RX path latencies computed for each technology. For example, the minimal latency supported by a GPP-based SDR platform when performing IEEE802.15.4 based slotted CSMA/CA protocol should take the sum of TX path and RX path latencies. This is due to the fact that slotted CSMA/CA protocol latency represents the Turn around Time (TT) which covers the delay for a receiver device to receive a data frame on RX path and if successfully decoded, transmit an ACK on TX path.

From Table 19, we see that SDR platforms that have wider range of operating frequency (HackRF, USRP-X310 and USRP-B210) satisfies the frequency requirements of all wireless technologies except for IEEE 802.15.6, which is only partially matched. Another exception is that of USRP-B210, which doesn't cover the frequency range of NFC technology. Whereas, most of the technologies are not fully supported by Sora and LimeSDR, although LimeSDR can fully/partially match with more technologies than Sora. In terms of bandwidth, while all wireless technologies are fully matched with the SDR platforms, the exceptions are IEEE

TABLE 19: Mapping SDR platform with wireless technologies (F : Full match; P : Partial match; N : No match).

wireless technology		GPP-based SDR platform															
		SDR device	Hack			X310			B210			Sora			Lime		
		No. of GPP cores	2 core	4 core	8 core	2 core	4 core	8 core	2 cor	4 cor	8 cor	2 cor	4 cor	8 cor	2 cor	4 cor	8 cor
		Freq.range	0.001 – 6 GHz			0.01 – 6 GHz			0.07 – 6 GHz			2.4-2.5; 4.9-5.875GHz			0.1 – 3.8 GHz		
Name	Metrics	Bandwidth	20 MHz			40/120/160 MHz			56 MHz			40 MHz			60 MHz		
NFC [27]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	26.07	39.10	187.70	26.07	39.10	61.44	26.07	39.10	40.00	26.07	39.10	160.00
		Max.Bitrate (Mbit/s)	14.95	14.95	14.95	26.07	39.10	187.70	26.07	39.10	61.44	26.07	39.10	40.00	26.07	39.10	160.00
		Latency (ms)	200.758	233.771	88.045	198.649	231.662	85.936	197.461	230.474	84.748	195.126	228.139	82.413	194.218	227.231	81.505
	Freq. range (13.56 MHz)		F			F			N			N			F		
	Bandwidth (1 MHz)		F			F			F			F			F		
	Max.Sym. rate (1.69MSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (424 kbit/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Latency (5.34 ms)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
IEEE802.15.6 [29]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	15.37	23.06	110.72	15.37	23.06	61.44	15.37	23.06	40.00	15.37	23.06	110.72
		Max.Bitrate (Mbit/s)	27.38	29.90	29.90	27.38	41.08	197.26	27.38	41.08	122.88	27.38	41.08	80.00	27.08	41.08	197.26
		Latency (ms)	374.810	497.547	211.160	372.701	495.438	209.051	371.513	494.250	207.863	369.178	491.915	205.528	368.270	491.007	204.620
	Freq. range (0.021 - 9.98GHz)		P			P			P			P			P		
	Bandwidth (0.3 – 499.2 MHz)		P			P			P			P			P		
	Max.Sym. rate (15.6 MSym/s)		P	P	P	P	F	F	P	F	F	P	F	F	P	F	F
	Max.Bitrate (15.6 Mbit/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Latency (75 μs)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
IEEE802.15.1 [30]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	26.07	39.10	187.70	26.07	39.10	61.44	26.07	39.10	40.00	26.07	39.10	160.00
		Max.Bitrate (Mbit/s)	27.38	29.90	29.90	27.38	41.08	197.26	27.38	41.08	122.88	27.38	41.08	80.00	27.08	41.08	197.26
		Latency (ms)	228.858	271.799	110.613	226.749	269.69	108.504	225.561	268.502	107.316	223.226	266.167	104.981	222.318	265.259	104.073
	Freq. range (2.4 GHz)		F			F			F			P			F		
	Bandwidth (1, 2 MHz)		F			F			F			F			F		
	Max.Sym. rate (2 MSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (2 Mbit/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Latency (150 μs)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
IEEE802.15.4 [31]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	15.37	23.06	110.72	15.37	23.06	61.44	15.37	23.06	40.00	15.37	23.06	110.72
		Max.Bitrate (Mbit/s)	27.38	29.90	29.90	27.38	41.08	197.26	27.38	41.08	122.88	27.38	41.08	80.00	27.08	41.08	197.26
		Latency (ms)	101.255	69.684	24.129	99.146	67.575	22.020	97.958	66.387	20.832	95.623	64.052	18.497	94.715	63.144	17.589
	Freq. range (subGHz,2.4GHz)		F			F			F			P			F		
	Bandwidth (0.4 - 5MHz)		F			F			F			F			F		
	Max.Sym. rate (62.5kSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (250 kbit/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Latency (Unslotted CSMA/CA: 192 μs)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
IEEE802.11ac [33]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	18.48	27.72	133.06	18.48	27.72	61.44	18.48	27.72	62.11	18.48	27.72	133.06
		Max.Bitrate (Gbit/s)	244.94	244.94	244.94	264.60	396.98	1906.1	264.60	396.98	1006.6	264.60	396.98	655.36	264.60	396.98	1906.1
		Latency (ms)	188.953	249.738	107.907	186.844	247.629	105.798	185.656	246.441	104.610	183.321	244.106	102.275	182.413	243.198	101.367
	Freq. range (5 GHz)		F			F			F			F			N		
	Bandwidth (20 - 160 MHz)		N			F			P			P			F		
	Max.Sym. rate (250 kSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (upto 6 Gbit/s)		P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
	Latency (16 μs)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
IEEE802.11ah [35]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	18.48	27.72	133.06	18.48	27.72	61.44	18.48	27.72	40.00	18.48	27.72	133.06
		Max.Bitrate (Gbit/s)	45.92	45.92	45.92	56.77	85.15	408.76	56.77	85.15	188.74	56.77	85.15	122.88	56.77	85.15	408.76
		Latency (ms)	181.885	240.314	103.980	179.776	238.205	101.871	178.588	237.017	100.683	176.253	234.682	98.348	175.345	233.774	97.440
	Freq. range (subGHz)		F			F			F			N			F		
	Bandwidth (1 - 16 MHz)		F			F			F			F			F		
	Max.Sym. rate (25 kSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (upto 347 Mbit/s)		P	P	P	F	F	F	F	F	F	F	F	F	F	F	F
	Latency (160 μs)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LTE [40]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	18.48	27.72	133.06	18.48	27.72	61.44	18.48	27.72	40.00	18.48	27.72	133.06
		Max.Bitrate (Gbit/s)	183.7	183.7	183.7	249.4	374.1	1796.0	249.4	374.1	754.9	249.4	374.1	491.5	249.4	374.1	1796.0
		Latency (ms)	264.818	178.735	78.322	262.709	176.626	76.213	261.521	175.438	75.025	259.186	173.103	72.690	258.278	172.195	71.782
	Freq. range (0.41 - 5.9 GHz)		F			F			F			P			P		
	Bandwidth (1.4 - 20 MHz)		F			F			F			F			F		
	Max.Sym. rate (15 kSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (100.8 Mbit/s)		F	F	F	F	F	F	F	F	F	F	F	F F	F	F	F
	Latency (4 ms)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
NB-IoT [41]		Max.Sym. rate (MSym/s)	14.88	14.95	14.95	14.88	22.32	107.17	14.88	22.32	61.44	14.88	22.32	40.00 1	14.88	22.32	107.17
		Max.Bitrate (Mbit/s)	2813.44	3827.20	3827.20	2813.44	4218.88	20,259.84	2813.44	4218.88	15,728.64	2813.44	4218.88	10,240.00	2813.44	4218.88	20,259.84
		Latency (ms)	25.157	18.962	11.750	23.048	16.853	9.641	21.860	15.665	8.453	19.525	13.330	6.118	18.617	12.422	5.210
	Freq. range (0.41 - 5.9 GHz)		F			F			F			P			P		
	Bandwidth (180 kHz)		F			F			F			F			F		
	Max.Sym. rate (15 kSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (250 kbit/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Latency (20 ms)		N	F	F	N	F	F	N	F	F	F	F	F	F	F	F
EC-GSM-IoT [38]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	26.76	40.14	192.68	26.76	40.14	61.44	26.76	40.00	40.00	26.76	40.14	160.00
		Max.Bitrate (Mbit/s)	44.85	44.85	44.85	75.24	112.86	541.86	75.24	112.86	184.32	75.24	112.86	120.00	75.24	112.86	480.00
		Latency (ms)	144.292	109.747	32.827	142.182	107.637	30.717	140.994	106.449	29.529	138.66	104.115	27.195	137.752	103.207	26.287
	Freq. range (0.8/0.9, 1.8/1.9GHz)		F			F			F			N			F		
	Bandwidth (200 kHz)		F			F			F			F			F		
	Max.Sym. rate (271 kSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (240 kbit/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Latency (577 μs)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LoRa [39]		Max.Sym. rate (MSym/s)	14.95	14.95	14.95	26.07	39.10	187.70	26.07	39.10	61.44	26.07	39.10	40.00	26.07	39.10	160.00
		Max.Bitrate (Mbit/s)	14.95	14.95	14.95	26.07	39.10	187.70	26.07	39.10	61.44	26.07	39.10	40.00	26.07	39.10	160.00
		Latency (ms)	192.134	214.311	93.147	190.025	212.202	91.038	188.837	211.014	89.850	186.502	208.679	87.515	185.594	207.771	86.607
	Freq. range (433/868 MHz)		F			F			F			N			F		
	Bandwidth 125 - 500 kHz)		F			F			F			F			F		
	Max.Sym. rate (3.91 kSym/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Max.Bitrate (50 kbit/s)		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Latency (24 ms)		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

802.15.6 and IEEE 802.11ac. IEEE 802.15.6 is partially matched by all SDR platforms; and IEEE 802.11ac is fully supported only by USRP-X310 and LimeSDR, partially by USRP-B210 and Sora.

The maximal symbol rate requirement of the wireless technologies, except IEEE 802.15.6, fully matches with that offered by the GPP-based SDR platforms. The maximal symbol rate of IEEE 802.15.6, however, fully matches with all SDR devices, except HackRF, connected to Quad and Octa-core GPP hosts and partially when connected to a Dual-core GPP host. The mapping of bitrate has a similar behaviour to that of symbol rate mapping. Thus, the maximal bitrates of

all wireless technologies, except IEEE 802.11ah and IEEE 802.11ac, fully match with that supported by all GPP-based SDR platforms. For the maximal bitrates of IEEE 802.11ah and IEEE 802.11ac, while the first partially matches with HackRF and fully with other SDR platforms, the second partially matches with all SDR devices connected to any GPP host type. For the latency matching, the requirement of most of the wireless technologies is several orders of magnitude less than what is offered by any of the SDR platforms. Thus, except NB-IoT, all the other technologies have no match. NB-IoT have full match with all SDR platforms interfaced with Quad-core and Octa-core GPP hosts.

As illustrated by the mapping table and discussion given, most GPP-based SDR platforms fully or partially satisfy the frequency, bandwidth, symbol rate and bitrate requirements of most wireless technologies. However, meeting the latency condition was a bit of a challenge, and thus, both software and hardware improvements should be made. For the software part, compiler and kernel should be optimized to increase the degree of parallelism according to the used processor capabilities. For the hardware part, increasing the number of cores can be efficient in case of applications having high degree of parallelism. Otherwise (with low degree of parallelism), processor with a higher clock speed or using hardware accelerators such as GPUs, DSPs and FPGAs are necessary. Several software projects exist to enable hardware accelerators usage like RAPIDS cuSignal [89] and Compute Unified Device Architecture (CUDA) [90] for GPU, RF Network-on-Chip (RFNoC) [91] and Nutaq's Real-Time Data Exchange (RTDEx) [92] for FPGA, meta-sdr OpenEmbedded layer [93] and liquidsdr [94] for DSP.

C. WHAT OTHER GPP-BASED SDR PLATFORMS FOR EXISTING WIRELESS SDR IMPLEMENTATIONS

Implementations of wireless technologies using SDR platforms were considered by several researchers. These implementations are accompanied by a limited, if not a single, recommended GPP-based SDR platforms. Moreover, the implementations doesn't show/demonstrate how the SDR platforms were selected nor there are studies to map several wireless standards with commercial SDR platforms. Table 20 presents examples of recommended GPP-based SDR platforms for selected wireless technologies. Now, based on the mapping table given in previous section, new opportunities appear to perform the existing implementations of wireless technologies. Thus, new possible GPP-based SDR platforms, listed in Table 20, become candidates and may be more convenient for some users in terms of cost, hardware availability, etc. Of course the list of the proposed GPP-based SDR platforms are not exhaustive, but it can be easily extended for other SDR devices and GPP hosts based on the theoretical analysis of our work. For example, a user having a GPP host with Quad-core 3.4 GHz processor can follow our theoretical analysis to determine its eligibility to perform a desired wireless technology through computing the supported bitrate, symbol rate and latency metrics. The following paragraphs demonstrate how to determine the candidate SDR platforms for few existing wireless SDR implementations.

IEEE 802.15.6: the proposed NB-WBAN evaluation platform by [85] uses USRP-N210 and GNU Radio to test different modulation techniques (DBPSK, DQPSK, D8PSK, GMSK) at operating frequency of 950MHz and bandwidth of 0.4MHz. The frequency band and bandwidth help determine the candidate SDR devices, and hence, from our matching conditions and mapping table, HackRF, USRP-X310, USRP-B210, Sora, and LimeSDR are possible candidates. The maximal symbol rate and bitrate for this implementation is 0.6MSym/s and 0.971Mbit/s, respectively, allowing us to choose any communication interface from (USB2.0, USB3.0, Gig.Eth, 10Gig.Eth and PCIe). The listed SDR devices also matches with the required symbol rate. For the target modulation techniques, symbol rate and bitrate, a GPP host having 2 core, 2.0 GHz processor without hyperthreading support is sufficient (see Table 12). Hence, from the list of SDR platforms considered in this paper, HackRF with USB2.0, USRP-X310 with Gig.Eth or 10Gig.Eth, USRP-B210 with USB3.0, Sora with PCIe, and LimeSDR with USB3.0 or PCIe can be used along with Dual-core, Quad-core or Octa-core GPP hosts.

LTE: srsRAN is one of the most popular open-source SDR implementation of LTE that has been tested with USRP, LimeSDR and BladeRF SDR devices; USB3.0, Gig.Eth and 10Gig.Eth as communication interface; and ARM based processors as GPP host [14]. It operates for LTE frequency bands and all current LTE bandwidths. As per the modulation, it supports LTE modulation coding scheme upto QAM256 in DL direction. The maximal bitrate achievable by the current srsRAN release is 75Mbit/s DL and 50Mbit/s uplink in 20MHz bandwidth and single-input single-output configuration. To determine other possible SDR platforms, we check for each parameter. The operating frequency for this implementation (LTE bands) ranges from 0.41 to 5.9GHz, which is supported partially by Sora SDR devices as discussed in the mapping table. For the bandwidth, Sora is capable to handle the implementation. The symbol rate and bitrate are also supported by the PCIe communication interface of Sora. As per the GPP host, the maximal symbol rate and maximal bitrate demanded by srsRAN are attained by Dual/Quad/Octa-core processors in both hyperthreading enabled and disable mode as demonstrated by our result in Table 12. Thus, from the list of SDR platforms considered in this paper and using the mapping table, Sora with PCIe interfaced with any of the three GPP hosts are suggested as possible SDR platform to test srsRAN in addition to those recommended by srsRAN.

D. EXPLOITATION OF THE MAPPING BETWEEN SDR AND WIRELESS TECHNOLOGIES

The mapping Table 19 can be exploited either by SDR software developers or regular users. An SDR software developer integrates two categories: who are programming the embedded software (custom SDR functions, FPGA design, etc.,) for SDR devices, and who are programming open-

TABLE 20: Possible GPP-based SDR platforms for existing implementations.

wireless technology	Description	SDR platform recommended by developers			New SDR platforms suggested by our work		
		SDR device	Comm. interface	GPP host	SDR device	Comm. interface	GPP host *
IEEE 802.15.6	A prototype for NB-WBAN transmit-receive system with high flexibility for healthcare applications (GNU Radio) [85]	USRP-N210	Gig.Eth	–	HackRF	USB2.0	D, Q, Octa
					USRP	USB3.0, Gig.Eth, 10Gig.Eth	
					Sora	PCIe	
					LimeSDR	USB3.0, PCIe	
IEEE 802.15.4	Implemented 802.15.4 encoding and decoding blocks (using GNU Radio) [95]	USRP1	USB2.0	Dual Pentium IV, 2.8GHz CPU	HackRF	USB2.0	D, Q, Octa
					USRP	USB3.0, Gig.Eth, 10Gig.Eth	
					Sora	PCIe	
					LimeSDR	USB3.0, PCIe	
	gr-IEEE-802.15.4 – transceiver testbed for GNU Radio (using GNU Radio) [15]	USRP-N210	Gig.Eth	Intel i5 CPU (2.6 GHz) laptop	HackRF	USB2.0	D, Q, Octa
					USRP	USB3.0, Gig.Eth, 10Gig.Eth	
					Sora	PCIe	
					LimeSDR	USB3.0, PCIe	
	GalioT-802.15.4 [42]	RTL-SDR	USB2.0	Raspberry Pi	HackRF	USB2.0	D, Q, Octa
					USRP	USB3.0, Gig.Eth, 10Gig.Eth	
					LimeSDR	USB3.0, PCIe	D, Q, Octa
IEEE 802.15.1	• Implemented GFSK modulation [96]	Tx: PC with DAC board, Agilent E4438C generator Rx: Wideband SDR analog front-end and PC with ADC board			HackRF	USB2.0	D, Q, Octa
					USRP	USB3.0, Gig.Eth, 10Gig.Eth	
					LimeSDR	USB3.0, PCIe	
IEEE 802.11a/b/g	SoftWiFi: Implementation of IEEE 802.11a/b/g [48]	Sora	PCIe	Dell XPS PCs (Intel Core 2 Quad 2.66GHz CPU)	USRP	PCIe	D, Q, Octa
					LimeSDR	PCIe	
IEEE 802.11a/g/n/ac	Developed open Wi-Fi platform for IEEE 802.11a/g/n/ac [97]	USRP-X310	10Gig.Eth	HP ML350 Gen9, Intel® Xeon® E5-2620, 3.2GHz, 12 cores, 24 Threads	Sora	PCIe	Octa
					LimeSDR	PCIe	
IEEE 802.11a/g/p	Developed a prototype for GNU radio based OFDM receiver (GNU Radio) [16]	USRP-N210	Gig.Eth	Intel Core i7-2600 CPU 3.40GHz	Sora	PCIe	D, Q, Octa
					LimeSDR	PCIe	
3GPP LTE	SoftLTE: Implementation of 3GPP LTE Uplink based on SoftWiFi [98]	Sora	PCIe	Dell XPS small form-factor PCs	USRP	USB3.0, Gig.Eth, 10Gig.Eth	Octa
					LimeSDR	PCIe	
	Implementation of LTE PHY layer based on multi-core GPP parallel processing [99]	USRP-N210	Gig.Eth	Intel-core i7 series CPU, 4 core, 8 Threads	Sora	PCIe	Octa
					LimeSDR	PCIe	
	srsRAN [14]	USRP, BladeRF, LimeSDR	USB3.0, Gig. Eth, 10Gig.Eth	ARM based processors	Sora	PCIe	D, Q, Octa
	OAI [23]	LimeSDR, BladeRF, USRP-B210, EURECOM EXPRESSMIMO2 RF	USB3.0, PCIe	Generation 3/4/5/6 Intel Core i5, i7 (4 core, 6 core)	Sora	PCIe	Octa
LoRa	• Implemented modulation and encoding of LoRa PHY (*gr-lora*) [24] • GNU Radio blocks for receiving and decoding LoRa modulated radio messages (*gr-lora*) [100]	HackRF, USRP, RTL-SDR	USB2.0, USB3.0	–	LimeSDR	USB3.0, PCIe	D, Q, Octa
	Modulation/demodulation and encoding/decoding using LimeSDR (LoRa-SDR) [101]	LimeSDR	USB3.0	–	HackRF	USB2.0	D, Q, Octa
					USRP	USB3.0, Gig.Eth, 10Gig.Eth	
	GalioT-LoRa [42]	RTL-SDR	USB2.0	Raspberry Pi	HackRF	USB2.0	D, Q, Octa
					USRP	USB3.0, Gig.Eth, 10Gig.Eth	
					LimeSDR	PCIe	

* D for Dual-core, 2 GHz GPP; Q for Quad-core, 1.5 GHz GPP; Octa for Octa-core, 3.6 GHz GPP type.

source software (wireless technology libraries, digital signal processing blocks, etc.,) on GPP hosts.

The mapping table cells with values N (no match) or P (partial match) give important information to encourage SDR software developer to propose solutions. As most of the table cells with value N are mainly due to the high latency of GPP hosts that cannot fulfill the wireless standard requirement, both embedded software and open-source software developers can provide improvements and optimizations in their software to reduce latency. Thus, embedded software developers can move some intensive computation or time critical MAC layer functions of wireless standards to be performed by the SDR device, which is more fast to save GPP resources and reduce latency. For example, from the mapping table, the wireless technology IEEE 802.15.4 cannot be performed (in terms of latency) by any SDR platforms due to the MAC layer latency. In this case, embedded software developers can decide to provide APIs to allow open-source software developers to directly implement and perform the critical MAC layer functions such as inter-frame spacing periods manager on the SDR device. Recently, some SDR devices have integrated this option such as all USRP devices from the third-generation integrated RFNoC [91], an API developed by Ettus Research to use USRP's FPGA processing power, with the UHD software. On their side, open-source software developers can adapt their wireless standard source code by integrating custom processing blocks of the critical MAC layer functions. These blocks should be implemented based on the APIs offered by the used SDR device. To continue with the RFNoC example, several RFNoC blocks have been developed and ready to use [102]. Also, on GPP hosts, open-source software developers should make an effort to provide software optimization (vectorization, automatic parallelization, inter-procedural optimization, etc.,) according to the used GPP host characteristics (clock speed, number of cores, memory and cache size, external hardware accelerators). These software optimizations, in context of SDR platforms, remain an open research (for further details, see

section VI-A). Based on the level of optimizations and our mapping table, open-source software developers can test and suggest more SDR platform possibilities to regular users.

Unlike SDR software developers, regular users employ GPP-based SDR platforms as a testbed to experiment their applications (upper layers) without relying on expert knowledge on how the wireless standards (lower layers) are implemented or performed. Indeed, they only need to know how to select the appropriate SDR platform and how to run the selected wireless technology. For a given application, they first determine its characteristics in terms of data type (sensory data, voice, video, etc.), traffic model (event-driven, continuous or query-driven), number of sensors, geographical area, etc., [4], [5]. Then, they express the application characteristics as a set of specific requirements such as range, data rate, energy consumption, etc. After, they select a set of candidate wireless technologies that can theoretically meet the application's requirement. To experiment the application with the selected wireless technologies, regular users can use a single SDR platform due to the reconfigurable and reprogrammable hardware. In order to determine the suitable SDR platform that can support the selected wireless technologies, regular users can refer to our mapping table. They can also refine their selection by choosing an SDR platform capable of satisfying the requirement of most of their selected wireless technologies.

VI. OPEN CHALLENGES AND FUTURE DIRECTIONS

Using GPP-based SDR platforms for wireless prototyping have been extensively accepted by many researchers and companies. Despite its continuing expansion, several existing research challenges have not yet been addressed. This section presents some open challenges and trends related to the performance enhancement of GPP-based SDR platforms to successfully perform wireless transceivers.

A. HARDWARE AND SOFTWARE OPTIMIZATIONS

To improve the performance of GPP-based SDR platforms in dealing with the computational requirements of wireless standards, hardware and software optimizations should be addressed. From the hardware side, GPP-based SDR platforms performance can be enhanced by different solutions such as : a) increasing the processing speed of GPP host by using more number of cores (as reported by our work) with higher clock speed, memory and cache, and b) using external hardware accelerators such as GPU, FPGA and DSPs. In literature, these solutions were more or less studied theoretically (using GPU [58], [59], [103], FPGA [56], [91], [104], [105], DSPs [93], [94]) but it's still an open challenge to fully investigate experimentally. In addition, the latency due to the interface between the GPP and external accelerator should be considered.

From the software side, various optimizations can be applied to enhance the execution time of the wireless transceiver code at the GPP host. Few examples include using increased number of threads (multi-threaded GPP), vectorization, automatic parallelization, inter-procedural optimization, SIMD and look-up tables (LUTs). These solutions are well investigated theoretically and experimentally in a general context. With respect to the SDR platform, there are few studies conducted by applying these solutions such as [20], [48], however, satisfying the full requirements of wireless transceivers still remains an open challenge. Another important point that seeks research attention is satisfying the real-time requirements (e.g., respecting response time) of wireless transceivers, which has been slightly addressed by researchers using Xenomai Real-Time OSs (RTOSs) [106]. However, the real-time performance of GPP-based SDR platforms need to be investigated more based on the real-time demands of different wireless standards using other RTOSs such as Real-time Linux [107], Real-Time Application Interface (RTAI) [108], and ChronOS [109].

B. VIRTUALIZATION IN SDR PLATFORMS

GPP-based SDR platform can use different type of SDR devices to implement one or more wireless transceivers. Two problems could arise: on one hand the PHY layer software portability, and on the other hand running multiple parallel wireless transceivers. For the first problem, as different types of SDR devices are considered, transceiver software portability can be insured by virtualization to abstract the hardware resources. In literature, the common solution to this problem consists of including a dedicated virtual machine (VM), named as radio virtual machine (RVM), on each SDR device image [11], [110], [111]. However, this solution introduces an extra cost in terms of overhead and latency, and limiting their effect remains an open challenge. The second problem concerns gateways (e.g., IoT gateways in home automation box) that require multiple transceivers to communicate with the deployed end-devices through heterogeneous wireless technologies. To enable this role on the GPP-based SDR platform, multiple VMs should be run in parallel at the GPP host where each VM is dedicated to one wireless transceiver [112]. However, in spite of the benefits gained, the performance of each implemented wireless transceiver maybe seriously degraded due to the competition of GPP host and SDR device resources between the co-located wireless transceivers and additional costs of VMs. These problems are not yet addressed and needs to be explored.

C. MOBILITY AND ENERGY CONSUMPTION

The majority of commercially available GPP-based SDR platforms (Desktop PC or Laptop based) consume high energy to achieve high performance, hence rely on main power supply. However, this is unsuitable for mobile application where SDR platforms can be used as mobile wireless transceivers. Indeed, many projects such as [113], [114] are developed addressing mobile applications where sensors are deployed on vehicles (e.g., car, drone, train) with an on-board SDR platform powered by the vehicle's battery like any mobile wireless device. All these projects report the negative impact of GPP-based SDR platforms on the life-time of

vehicle's battery. It becomes necessary to build mobile SDR platforms consuming less energy and at the same time offer expected performance. One discussed solution in literature [42], [115], [116] is to use low power embedded computer boards such as Raspberry Pi, ARM Cortex processor, etc., as GPP hosts. Nevertheless, prototypes based on this solution lack offering adequate performance to support the requirement (specially, the frequency, bandwidth, latency) of most wireless technologies. Thus, building energy efficient SDR platform using embedded computer boards with performance objective is still an open issue. Another alternative solution discussed in literature is to remove the GPP host by migrating its capabilities to the SDR device to form a standalone SDR platform. Examples of such SDR platforms are USRP-E3xx [117], BladeRF [118], μSDR [119]. They are designed to offer high performance, be energy efficient and suitable for mobile applications. However, they require more experimental investigations on their performance in terms of frequency, bandwidth, symbol rate, bitrate and latency.

VII. CONCLUSIONS

Selecting SDR platform to implement and perform a wireless technology is challenging as it comprises, on one hand, to satisfy design requirements both at the hardware and software level. On the other hand, previous recommendations by researchers/developers of wireless technologies suggest to fulfill the proposed hardware and software list to successfully perform their open-source implementations. However, the proposed list is often restrictive in terms of hardware (SDR devices and GPP hosts) and doesn't take into account the use-case desired by users. This paper has reviewed and presented a large list of GPP-based SDR platforms that satisfy the minimum requirement of wireless technologies.

We believe that the study conducted in this paper will help users to determine, for a given wireless technology, which GPP-based SDR platform configuration is necessary to fully or partially perform the MAC and PHY functions. Additionally, through this study, users who already possess a GPP-based SDR platform can identify possible applications that could be implemented. To determine the candidate SDR platform, the paper first evaluated the performance of selected GPP-based SDR platforms through theoretical and experimental analysis. Then, we proposed matching conditions and created a mapping table between the minimum requirements of well-known wireless technologies and performance of GPP-based SDR platforms. Thereby, a list of candidate GPP-based SDR platforms is established for each wireless technology. This list indicates if the matching is complete, incomplete or negative. The two latter are mainly due to the high latency of GPP hosts. A summary of some of the existing implementations were discussed and using the mapping table we suggested other possible GPP-based SDR platforms to be used. Finally, we highlighted some of the research challenges and future directions to be considered by the research community.

REFERENCES

[1] Research and Markets. "Wireless Sensor Network Markets," 2019, Accessed September 2021 [Online]. https://www.researchandmarkets.com/reports/4844854/wireless-sensor-network-markets?w=5&utm_source=CI&utm_medium=PressRelease&utm_code=g87tcl

[2] IoT Analytics. Accessed January 2021 [Online]. https://iot-analytics.com/iot-2020-in-review/

[3] A. Gupta and R. K. Jha, "A Survey of 5G Network: Architecture and Emerging Technologies," in IEEE Access, vol. 3, pp. 1206-1232, 2015, doi: 10.1109/ACCESS.2015.2461602.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," in IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015.

[5] Yaqoob, Ibrar, Ejaz Ahmed, Ibrahim Abaker Targio Hashem, Abdelmuttlib Ibrahim Abdalla Ahmed, Abdullah Gani, Muhammad Imran, and Mohsen Guizani. "Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges." IEEE wireless communications 24, no. 3 (2017): 10-16.

[6] Tessier, Russell, and Wayne Burleson. "Reconfigurable computing for digital signal processing: A survey." Journal of VLSI signal processing systems for signal, image and video technology 28.1-2 (2001): 7-27.

[7] Ulversoy, Tore. "Software defined radio: Challenges and opportunities." IEEE Communications Surveys & Tutorials 12, no. 4 (2010): 531-550.

[8] D. F. Macedo, D. Guedes, L. F. M. Vieira, M. A. M. Vieira and M. Nogueira, "Programmable Networks—From Software-Defined Radio to Software-Defined Networking," in IEEE Communications Surveys & Tutorials, vol. 17, no. 2, pp. 1102-1125, Secondquarter 2015.

[9] Akeela, R., & Dezfouli, B. (2018). Software-defined Radios: Architecture, state-of-the-art, and challenges. Computer Communications, 128(July), 106–125.

[10] Gavrilă, C., Popescu, V., Alexandru, M., Murroni, M., and Sacchi, C. "An SDR-Based Satellite Gateway for Internet of Remote Things (IoRT) Applications." IEEE Access 8 (2020): 115423-115436.

[11] M. Kist, J. Rochol, L. A. DaSilva and C. B. Both, "SDR Virtualization in Future Mobile Networks: Enabling Multi-Programmable Air-Interfaces," 2018 IEEE International Conf. on Communications (ICC), 2018, pp. 1-6.

[12] M. Schadhauser, J. Robert, and A. Heuberger, "Design of autonomous basestations for low power wide area (LPWA) communication," inProc. SmartSysTech; Eur. Conf. Smart Objects, Syst. Technol., Jun. 2017, pp. 1–8.

[13] Wei, Xingguang, Haitao Liu, Zhiming Geng, Kan Zheng, Rongtao Xu, Yang Liu, and Peng Chen. "Software defined radio implementation of a non-orthogonal multiple access system towards 5G." IEEE Access 4 (2016): 9604-9613.

[14] Gomez-Miguelez, Ismael, Andres Garcia-Saavedra, Paul D. Sutton, Pablo Serrano, Cristina Cano, and Doug J. Leith. "srsLTE: An open-source platform for LTE evolution and experimentation." In Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization, pp. 25-32. 2016. https://github.com/srsran/srsRAN

[15] Bastian Bloessl, Christoph Leitner, Falko Dressler and Christoph Sommer, "A GNU Radio-based IEEE 802.15.4 Testbed," Proceedings of 12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013), Cottbus, Germany, September 2013, pp. 37-40.

[16] Bloessl, Bastian, Michele Segata, Christoph Sommer, and Falko Dressler. "An IEEE 802.11 a/g/p OFDM Receiver for GNU Radio." In Proceedings of the second workshop on Software radio implementation forum, pp. 9-16. 2013.

[17] Khan, Muhammad Bilal, Xiaodong Yang, Aifeng Ren, Mohammed Ali Mohammed Al-Hababi, Nan Zhao, Lei Guan, Dou Fan, and Syed Aziz Shah. "Design of software defined radios based platform for activity recognition." IEEE Access 7 (2019): 31083-31088.

[18] Politis, Christos, Sina Maleki, Juan Merlano Duncan, Jevgenij Krivochiza, Symeon Chatzinotas, and Björn Ottesten. "SDR implementation of a testbed for real-time interference detection with signal cancellation." IEEE Access 6 (2018): 20807-20821.

[19] Handagala, Suranga, and Miriam Leeser. "Real time receiver baseband processing platform for sub 6 GHz PHY layer experiments." IEEE Access 8 (2020): 105571-105586.

[20] Chen, Y., Lu, S., Kim, H. S., Blaauw, D., Dreslinski, R. G., & Mudge, T. (2016). A low power software-defined-radio baseband processor for

the Internet of Things. Proceedings - International Symposium on High-Performance Computer Architecture, 2016-April, 40–51.

[21] S. Wu, S. Kang, C. Chakrabarti and H. Lee, "Low power baseband processor for IoT terminals with long range wireless communications," 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Washington, DC, 2016, pp. 728-732.

[22] Subramanian, Ramanathan, Benjamin Drozdenko, Eric Doyle, Rameez Ahmed, Miriam Leeser, and Kaushik Roy Chowdhury. "High-level system design of IEEE 802.11 b Standard-Compliant Link Layer for MATLAB-Based SDR." IEEE Access 4 (2016): 1494-1509.

[23] OpenAirInterface (OAI) Project, 2021, https://gitlab.eurecom.fr/oai/openairinterface5g/-/wikis/home.

[24] KNIGHT, Matthew; SEEBER, Balint. Decoding LoRa: Realizing a Modern LPWAN with SDR. Proceedings of the GNU Radio Conference, [S.l.], v. 1, n. 1, sep. 2016. https://github.com/matt-knight/gr-lora.

[25] Field, N. (n.d.). Internet of Things (IoT) Smart Connected. November 2020. http://literature.cdn.keysight.com/litweb/pdf/5992-1217EN.pdf.

[26] ITU-R (RR Nos. 5.138 and 5.150). August 2021. Online: https://www.itu.int/net/ITU-R/terrestrial/faq/#g013.

[27] International Organization for Standardization and International Electrotechnical Commission. ISO/IEC 18092:2013/Cor 1:2015. [Online]. https://standards.iso.org/ittf/PubliclyAvailableStandards/index.html.

[28] Roy, B. S., Ieee, F., Jandhyala, V., Smith, J. R., Ieee, M., Wetherall, D. J., … Ieee, S. M. (2010). RFID : From Supply Chains to Sensor Nets. Proceedings of the IEEE, 98(9), 1583–1592.

[29] IEEE standard for local and metropolitan area networks: Part 15.6: Wireless body area networks, IEEE submission, 2012

[30] IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN)," in IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002) , vol., no., pp.1-700, 14 June 2005

[31] IEEE Standard for Low-Rate Wireless Networks, in IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011), vol., no., pp.1-709, 22 April 2016

[32] IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Computer Society, (7 December 2016), IEEE Std 802.11™-2016.

[33] IEEE Std 802.11ac™-2013 IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz.

[34] S. Tozlu, M. Senel, W. Mao and A. Keshavarzian, "Wi-Fi enabled sensors for internet of things: A practical approach," in IEEE Communications Magazine, vol. 50, no. 6, pp. 134-143, June 2012.

[35] IEEE Standard for Information technology - Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 2: Sub 1 GHz License Exempt Operation. IEEE Computer Society, (7 December 2016), IEEE Std 802.11ah™-2016.

[36] IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments, in IEEE Std 802.11p-2010, pp.1-51, 15 July 2010.

[37] Mekki, Kais, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. "A comparative study of LPWAN technologies for large-scale IoT deployment." ICT express 5, no. 1 (2019): 1-7.

[38] 3GPP Technical Specification Group GSM/EDGE Radio Access Network, "Cellular system support for ultra low complexity and low throughput internet of things," 3GPP, Tech. Rep. 45.820 V13.1.0, Nov. 2015.

[39] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, "Lorawan specification," LoRa Alliance, San Ramon, CA, USA,Tech. Rep., 2015.

[40] 3GPP TS 36.213: LTE; "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures. https://www.etsi.org/deliver/etsi_ts/136200_136299/136213/12.03.00_60/ts_136213v120300p.pdf

[41] European Telecommunications Standards Institute, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) conformance testing (3GPP TS 36.141 version 13.6.0 Release 13)" European Telecom. Standards Institute, ETSI TS 136 141, V13.6.0 (2017-01).

[42] Narayanan, Revathy, and Swarun Kumar. "Revisiting software defined radios in the iot era." In Proceedings of the 17th ACM Workshop on Hot Topics in Networks, pp. 43-49. 2018.

[43] USRP SDR. June 2020, https://www.ettus.com/products/

[44] AD-FMCOMMS2-EBZ, Analog module. https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/EVAL-AD-FMCOMMS2.html#eb-overview. June 2020.

[45] HackRF One, June 2020. https://github.com/mossmann/hackrf/wiki/HackRF-One.

[46] WARP Radio Board, June 2020. http://warpproject.org/trac/wiki/ HardwareUsers Guides/RadioBoard_v1.4.

[47] LimeSDR Lime Microsystems – Software Defined Radio. July 2020. https://limemicro.com/products/boards/limesdr/

[48] Tan, K., Liu, H., Zhang, J., Zhang, Y., Fang, J., & Voelker, G. M. (2011). Sora: high-performance software radio using general-purpose multi-core processors, Communications of the ACM, 54(1), 99.

[49] T. Hentschel, M. Henker, G. Fettweis, The digital front-end of software radio terminals, IEEE Pers. Commun. 6 (4) (1999) 40–46.

[50] USB specifications. June 2020. https://www.usb.org/documents

[51] IEEE Standard for Ethernet, in IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015) , vol., no., pp.1-5600, 31 Aug. 2018 doi: 10.1109/IEEESTD.2018.8457469

[52] Lawley, J. (2014). Understanding Performance of PCI Express Systems White Paper (WP350). Wp350, 350(2), 1–16. Retrieved from www.xilinx.com

[53] eXtensible Host Controller Interface for USB (xHCI). June 2020. https://www.intel.com/content/www/us/en/products/docs/io/universal-serial-bus/extensible-host-controler-interface-usb-xhci.html

[54] M. Véstias and H. Neto, "Trends of CPU, GPU and FPGA for high-performance computing," 2014 24th International Conference on Field Programmable Logic and Applications (FPL), Munich, 2014, pp. 1-6.

[55] Noergaard, Tammy. Embedded systems architecture: a comprehensive guide for engineers and programmers. Newnes, 2012.

[56] Cardoso, J. M. P., Coutinho, J. G. F., Diniz, P. C., Cardoso, J. M. P., Coutinho, J. G. F., & Diniz, P. C. (2017). High-performance embedded computing. Embedded Computing for High Performance, 17–56.

[57] Amiri, Hossein, and Asadollah Shahbahrami. "SIMD programming using Intel vector extensions." Journal of Parallel and Distributed Computing 135 (2020): 83-100.

[58] K. Li, M. Wu, G. Wang and J. R. Cavallaro, "A high performance GPU-based software-defined basestation," 2014 48th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2014, pp. 2060-2064.

[59] Bernier, Steve, François Lévesque, Martin Phisel, Dmitry Zvernik, and David Hagood. "Using OpenCL to Increase SCA Application Portability." Journal of Signal Processing Systems 89, no. 1 (October 1, 2017): 107–17.

[60] Dubois, Michel, Murali Annavaram, and Per Stenström. Parallel computer organization and design. Cambridge university press, 2012.

[61] Mishra, Sanjeeb, Neeraj Kumar Singh, and Vijayakrishnan Rousseau. System on chip interfaces for low power design. Morgan Kaufmann, 2015.

[62] Kaeli, David R., Perhaad Mistry, Dana Schaa, and Dong Ping Zhang. Heterogeneous computing with OpenCL 2.0. Morgan Kaufmann, 2015.

[63] Silberschatz, Abraham, Peter Baer Galvin, and Greg Gagne. Operating system concepts essentials. John Wiley & Sons, Inc., 2014.

[64] GNU Radio. April 2020. https://www.gnuradio.org/

[65] LabVIEW - National Instruments. April 2020. https://www.ni.com/en-us/shop/labview.html

[66] MathWorks for MATLAB and Simulink, April 2020. https://www.mathworks.com/

[67] GCC, the GNU Compiler Collection, May 2020. https://gcc.gnu.org/

[68] Intel® C++ Compiler Classic Developer Guide and Reference. May 2020. https://software.intel.com/content/www/us/en/develop/documentation/cpp-compiler-developer-guide-and-reference

[69] Altman, Yair M., Accelerating MATLAB Performance: 1001 tips to speed up MATLAB programs. Chapman and Hall/CRC Press, Inc., 2014.

[70] Lattner, C., and Vikram A., "LLVM: A compilation framework for lifelong program analysis & transformation." In Int. Symposium on Code Generation and Optimization, 2004. CGO 2004., pp. 75-86. IEEE, 2004.

[71] Intel Intrinsics Guide, Intel intrinsic instructions. May 2020. https://software.intel.com/sites/landingpage/IntrinsicsGuide/#.

[72] VOLK, GNU Radio. May 2020. https://wiki.gnuradio.org/index.php/Volk

[73] NI LabVIEW Compiler. May 2020. https://www.ni.com/en-us/support/documentation/supplemental/10/ni-labview-compiler–under-the-hood.html

[74] U.L. Rohde, T.T.N. Bucher, Communications Receivers: Principles and Design, 4, McGraw-Hill Education, 1988.

[75] Bensky, Alan. Short-range wireless communication. Newnes, 2019.

[76] Xie, Steven. Practical Filter Design Challenges and Considerations for Precision ADCs, Analog Dialogue, Vol 50, 2016.

[77] H. J. Landau, "Sampling, data transmission, and the Nyquist rate," in Proceedings of the IEEE, vol. 55, no. 10, pp. 1701-1706, Oct. 1967.

[78] Tan, L., Jiang, J., Tan, L., & Jiang, J. (2019). Multirate Digital Signal Processing, Oversampling of Analog-to-Digital Conversion, and Undersampling of Bandpass Signals. Digital Signal Processing, 529–590.

[79] Widmer, Albert X. "8B/10B encoding and decoding for high speed applications." U.S. Patent 6,977,599, issued December 20, 2005.

[80] Perahia, Eldad, and Robert Stacey. Next generation wireless LANs: 802.11 n and 802.11 ac. Cambridge university press, 2013.

[81] Linux kernel profiling, perf. June 2020. https://perf.wiki.kernel.org/index.php/Tutorial

[82] Little J.D.C., Graves S.C. (2008) Little's Law. In: Chhajed D., Lowe T.J. (eds) Building Intuition. International Series in Operations Research & Management Science, vol 115. Springer, Boston, MA

[83] ADS62Px9/x8, "Dual channel ADC." March 2020. http://www.ti.com/lit/ds/slas635b/slas635b.pdf

[84] NFC. July 2020. https://github.com/jcrona/gr-nfc.

[85] Tianchan Guan, Jun Han and Xiaoyang Zeng, "Highly flexible WBAN transmit-receive system based on USRP," 2013 IEEE 10th International Conference on ASIC, Shenzhen, 2013, pp. 1-4.

[86] Scapy radio with GNU Radio for Bleutooth and Sigfox. July 2020. https://bitbucket.org/cybertools/scapy-radio/src/default/

[87] Open Source Mobile Communications, gr-GSM. July 2020. https://osmocom.org/

[88] Dargie, Waltenegus and Christian Poellabauer. "Fundamentals of Wireless Sensor Networks: Theory and Practice." (2010).

[89] Adam Thompson (2019). GPU-Accelerated Signal Processing with cuSignal [Online]. [Accessed: July 2020]. https://medium.com/rapids-ai/gpu-accelerated-signal-processing-with-cusignal-689062a6af8

[90] CUDA toolkit documentation, July 2020. http://docs.nvidia.com/cuda/

[91] Braun, Martin, Jonathan Pendlum, and Matt Ettus. "RFNoC: RF network-on-chip." In Proc. of the GNU Radio Conf., vol. 1, no. 1. 2016.

[92] Nutaq RTDEx: accelerating GNU Radio development with Xilinx FPGAs. [Accessed: July 2020]. https://www.nutaq.com/blog/accelerating-gnu-radio-development-xilinx-fpgas

[93] Ma, Shenghou, Vuk Marojevic, Philip Balister, and Jeffrey H. Reed. "Porting GNU Radio to multicore DSP+ ARM system-on-chip–a purely open-source approach." In Karlsruhe Workshop on Soft. Radios. 2014.

[94] Joseph D. Gaeddert, liquid: open-source DSP library, July 2020. https://liquidsdr.org/

[95] T. Schmid, "GNU Radio 802.15. 4 En-and Decoding," Networked & Embedded Systems Laboratory, UCLA, Technical Report TR-UCLA-NESL-200609-06, June 2006

[96] Schiphorst, R., Hoeksema, F. W., Arkesteijn, V. J., Slump, C. H., Klumperink, E. A. M., & Nauta, B. (2004). A GPP-based Software-Defined Radio Front-end for WLAN Standards. In Proceedings of the Fourth IEEE Benelux Signal Processing Symposium (pp. 203-206). Hilvarenbeek: IEEE Benelux Signal Processing Chapter.

[97] K. Kang, Z. Zhu, D. Liu, W. Zhang and H. Qian, "A software defined open Wi-Fi platform," in China Communications, vol. 14, no. 7, pp. 1-15, July 2017.

[98] Li, Y., Fang, J., Tan, K., Zhang, J., Cui, Q., & Tao, X. (2009). Soft-LTE : A Software Radio Implementation of 3GPP Long Term Evolution Based on Sora Platform. Demo 2009 ACM International Conference on Mobile Computing and Networking (MobiCom), (January), 1–2.

[99] Z. Chen and J. Wu, "LTE physical layer implementation based on GPP multi-core parallel processing and USRP platform," 9th International Conference on Communications and Networking in China, Maoming, 2014, pp. 197-201.

[100] Pieter Robyns, Peter Quax, Wim Lamotte, William Thenaers. (2017). gr-lora: An efficient LoRa decoder for GNU Radio. Zenodo. 10.5281/zenodo.853201

[101] Blum, Josh. Lora-sdr. Source Code on Github, 2016. https://github.com/myriadrf/LoRa-SDR.

[102] Ettus Knowledge Base contributors, "RFNoC," Ettus Knowledge Base, https://kb.ettus.com/index.php?title=RFNoC&oldid=4228 (accessed August 20, 2021).

[103] J. Kim, S. Hyeon and S. Choi, "Implementation of an SDR system using graphics processing unit," in IEEE Communications Magazine, vol. 48, no. 3, pp. 156-162, March 2010.

[104] Cai, Xin, Mingda Zhou, and Xinming Huang. "Model-based design for software defined radio on an FPGA." IEEE Access 5 (2017): 8276-8283.

[105] Kumar, Nishant, Meenakshi Rawat, and Karun Rawat. "Software-Defined Radio Transceiver Design Using FPGA-Based System-on-Chip Embedded Platform With Adaptive Digital Predistortion." IEEE Access 8 (2020): 214882-214893.

[106] Peng Guo, Xin Qi, Limin Xiao and Shidong Zhou, "A novel GPP-based Software-Defined Radio architecture," 7th International Conference on Communications and Networking in China, Kunming, China, 2012, pp. 838-842.

[107] Real-Time Linux, [Online]. https://wiki.linuxfoundation.org/realtime/start

[108] RTAI, the RealTime Application Interface for Linux. [Online]. https://www.rtai.org/

[109] ChronOS Real-time Linux, [Online]. http://www.chronoslinux.org/wiki/Main_Page

[110] R. Hossain, M. Wesseling, and C. Leopold, "Application descriptionconcept with system level hardware abstraction," inSignal ProcessingSystems Design and Implementation, 2005. IEEE Workshop on, Nov.2005, pp. 36–41.

[111] R. B. Abdallah, T. Risset, A. Fraboulet, and Y. Durand, "The radiovirtual machine: A solution for sdr portability and platform reconfigurability,"Parallel and Distributed Processing Symposium, International,vol. 0, pp. 1–4, 2009

[112] Ahn, Heungseop, Seungwon Choi, Markus Mueck, and Vladimir Ivanov. "Data plane framework for software-defined radio access network based on ETSI-standard mobile device architecture." IEEE Access 7 (2019): 163421-163436.

[113] D. M. Molla, H. Badis, A. A. Desta, L. George and M. Berbineau, "SDR-Based Reliable and Resilient Wireless Network for Disaster Rescue Operations," 2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), Paris, France, 2019, pp. 1-7.

[114] Powell, Keith, Aly Sabri Abdalla, Daniel Brennan, Vuk Marojevic, R. Michael Barts, Ashwin Panicker, Ozgur Ozdemir, and Ismail Guvenc. "Software Radios for Unmanned Aerial Systems." In Proceedings of the 1st International Workshop on Open Software Defined Wireless Networks, pp. 14-20. 2020.

[115] Hessar, Mehrdad, Ali Najafi, Vikram Iyer, and Shyamnath Gollakota. "TinySDR: Low-Power SDR Platform for Over-the-Air Programmable IoT Testbeds." In 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), pp. 1031-1046. 2020.

[116] Utrilla, Ramiro, Roberto Rodriguez-Zurrunero, Jose Martin, Alba Rozas, and Alvaro Araujo. "MIGOU: A low-power experimental platform with programmable logic resources and software-defined radio capabilities." Sensors 19, no. 22 (2019): 4983.

[117] USRP E3xx: USRP Embedded Series, December 2020. https://www.ettus.com/product-categories/usrp-embedded-series/

[118] bladeRF 2.0 micro, December 2020. https://www.nuand.com/bladerf-2-0-micro

[119] Kuo, Ye-Sheng, Pat Pannuto, Thomas Schmid, and Prabal Dutta. "Reconfiguring the software radio to improve power, price, and portability." In Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, pp. 267-280. 2012.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3154364, IEEE Access

DEREJE M. MOLLA received the B.S. degree in electrical engineering from Hawassa University, Hawassa, Ethiopia in 2008; and the M.S. degree in Telematics from Politecnico di Torino, Turin, Italy in 2013. He is currently pursuing the Ph.D. degree in computer science at Gustave Eiffel University, Marne la Vallée, France. From 2014 to 2018, he was a lecturer at the communication engineering stream, Hawassa University Institute of Technology, Hawassa, Ethiopia. His research interest includes to study the performance of SDR platforms as a wireless transceiver for WSNs and IoT, integration of SDR and software defined networking for wireless networks.

HAKIM BADIS received the M.S. degree in Distributed Computing from Paris-sud University, Orsay, France in 2002, and the Ph.D. degree in Mobile Networks from LRI, Orsay, France in 2005. He is currently associate Professor in computer science at Gustave Eiffel University, researcher at LIGM lab specialized in next generation wireless networks, multi-hop and IoT sensor networks, smart antennas (MIMO, etc.), software defined radio, software defined networking, discrete mathematics (graph theory, information theory, etc.), distributed algorithms and complexity.

PLACE PHOTO HERE

LAURENT GEORGE received the PhD degree in computer science from University of Versailles-St Quentin, France in 1998, and the HDR (Habilitation to Direct Research) on "Temporal robustness of real-time embedded and distributed systems," from University of Nantes, France in 2008. He is currently Professor and Head of the Computer Science department at ESIEE Paris; Head of Software, Networks and Real-Time research group; member of LIGM lab at University of Paris-Est; and associate researcher at INRIA Paris-Rocquencourt in the AOSTE team. His research activities concern real time embedded systems, software defined network and network functions virtualization, and IoT.

MARION BERBINEAU received the Engineering degree in electrical engineering from Polytech' Lille, France, in 1986, and the Ph.D. degree in electrical engineering from the University of Lille in 1989. She is currently a full-time Research Director with IFSTTAR, the French institute of science and technology for transport, development and networks. She is also an Expert in the field of radio wave propagation in transport environments (tunnels), electromagnetic modeling, channel characterization and modeling, MIMO, wireless systems for telecommunications, cognitive radio for railways, and GNSS localization-based for ITS, particularly for the rail and public transport domains. She is also active as an Expert for the GSM-R and future systems such as LTE-A and 5G. She is involved in several National and European research projects. She has authored and co-authored several publications and patents.