Radio Hardware Virtualization for Coping with Dynamic Heterogeneous Wireless Environments

Xianjun Jiao(✉), Ingrid Moerman, Wei Liu, and Felipe Augusto Pereira de Figueiredo

IDLab, Department of Information Technology, Ghent University - imec, Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium
{xianjun.jiao,ingrid.moerman,wei.liu,felipe.pereira}@ugent.be

Abstract. Diverse wireless standards, designed for diverse traffic types, operate in the same wireless environment without coordination, often leading to interference and inefficient spectrum usage. Although C-RAN (Cloud/centralized RAN) is a promising architecture to achieve intra-operator network coordination, the architecture encounters challenge when low latency services and diverse access technologies are expected over non-fiber fronthaul. So, multi-standard multi-channel access point with low processing latency is preferred to be at the edge of network instead of central cloud. But, developing this kind of equipment is difficult as multiple radio chips and drivers have to be integrated and coordinated. In ORCA (Orchestration and Reconfiguration Control Architecture) project, a SDR architecture is developed on a single chip radio platform including hardware accelerators wrapped by unified software APIs, which offer the following capabilities: (1) concurrent data transmission over multiple virtual radios; (2) runtime composition and parametric control of radios; and (3) radio resource slicing, supporting independent operation of multiple standards in different bands, time slots or beams. Such an architecture offers a fast development cycle, as only software programming is required for creating and manipulating multiple radios. The architecture further achieves an efficient utilization of hardware resources, as accelerators can be shared by multiple virtual radios.

Keywords: Coexistence · SDR · Resource slicing · FPGA · C-RAN Virtualization · CPRI

1 Introduction

Our world is increasingly defined by software. Even sectors that used to rely mainly on hardware are evolving rapidly. From use cases like self-driving cars to the inspection of factory plants, digital assets make the difference.

Software Defined Radio (SDR) [1] is a typical case of "softwarized" hardware: transceiver components (such as mixers, demodulators or decoders) that are typically implemented on hardware are now possible to be implemented by means of software.

P. Marques et al. (Eds.): CROWNCOM 2017, LNICST 228, pp. 287–297, 2018.
https://doi.org/10.1007/978-3-319-76207-4_24

C-RAN (Cloud/centralized Radio Access Network) [2] is a typical application of SDR in wireless industry. In C-RAN architecture, RRH (Remote Radio Head, including antennas) is connected to software BBU (Base Band Unit) in cloud center via digitalized IQ sample fiber link (CRPI or OBSAI) [3]. Each RRH covers a sector or a cell. Because all cells' BBUs are resident in the same cloud center, it brings benefits [4] to operator, such as allocating processing and frequency resources to different cell dynamically according to traffic variation (day VS night; street VS stadium; etc.); coordinate interference among different cells in a more efficient way; multi-cell cooperated transmitting and receiving for UE (User Equipment) and etc. A drawback of C-RAN for operator is that high quality fronthaul is needed to connect RRH at antenna side and BBU in the cloud. The link needs to be low latency and low jitter [5]. When there isn't fiber available, fronthaul will be challenging.

Advanced technologies in computer science, such as virtualization [4], can be easily applied to C-RAN, since all layers of RAN protocol, including L1 – physical layer, are implemented in software running in cloud center. By virtualization, it is no longer necessary to map each BBU software instance to one physical server. Multiple BBU instances can share the same physical server, or multiple physical servers can serve as a super computer to run an ultra-high bandwidth BBU. BBU instance can be created, destroyed or migrated according to dynamic requirement.

In parallel with operators' RAN, there are many other types of wireless networks/standard, such as Wi-Fi [6] for internet or intranet access, 802.15.4/Zigbee [7] and 802.11ah [8] for short-to-middle range IoT (Internet of Things) applications, as well as LoRa [9] and SigFox [10] for long range IoT, remain competitive. In addition, new applications are emerging, serving as driving force of network technologies, which include robot or UAV (Unmanned Aerial Vehicle) control, vehicle-to-vehicle communication for autopilot, Virtual Reality (VR) and real-time gaming. So, new standards or new features are needed over existing standards to interact with diverse physical world.

Furthermore, spectrum sharing is a common issue for many technologies mentioned above, especially for those operating in the ISM (Industrial, Scientific and Medical) bands. In practice, the coordination among these technologies is either very hard to achieve or not present at all. Unlike the case of C-RAN, an operator can run multiple standards and base stations at one location – cloud center, the application here run in different physical access-point/gateways, which are owned by different enterprises or even private home. This situation may lead to inefficient usage of spectrum, and severe QoS (Quality of Service) degradation for specific application due to interference from heterogeneous technologies.

An equipment supporting parallel operation of multiple standards and multiple channels is a promising approach to deliver services with diverse QoS, in terms of optimized wireless access and spectrum utilization efficiency. For low latency application, hardware solution, which could be ASIC (Application-Specific Integrated Circuit) or FPGA (Field-Programmable Gate Array), is more appropriate than software. A problem is that, one chip always is implemented for one or few standard, and operates on one frequency channel at a time. One option is to construct an equipment with multiple chips to support multiple standards and channels, though this will be not only costly but also inconvenient to program and coordinate from the perspective of the developer—ASICs with dedicated configurations, drivers and inter-chip communication

link become necessary prerequisites. For a multi-chip design, features commonly supported in C-RAN, such as dynamic computing resource sharing and migrating among cells/channels, are very hard to realize (if not impossible). This is because multi-chip hardware design implies fixed hardware resources allocation. An integrated new ASIC could be designed to merge multiple chips to single chip, but simply chip merging won't give dynamic resource manipulation among intra ASIC blocks. In addition, long design cycle and high development cost of ASIC would be a big obstacle on the road.

In this paper, an architecture supporting low latency processing via hardware (FPGA) accelerators, which is still flexible enough to support multi-standard multi-channel virtualization, is proposed to meet the requirement of diverse wireless access at the network edge. An initial demo is also implemented based on SDR platform composed by Xilinx Zynq SoC and Analog devices RF frontend.

2 Latency Analysis

Latency is one of key factors for different types of standards and application. System architecture of wireless technology is influenced by the latency requirements and implementation feasibility of specific latency target. In this section, we first investigate the latency requirements of mainstream wireless standards (Sect. 2.1), and then present the latency measurements of two types of SDR platforms, namely the USRP (Sect. 2.2), and Xilinx Zynq SoC based SDR (Sect. 2.3).

2.1 Latency Requirement of Wi-Fi and LTE

According to Wi-Fi standard [6], the most critical latency requirement is from SIFS (Short Interframe Space). The concept of interframe space is shown in Fig. 1, which is from "Fig. 10-4—Some IFS relationships" of 802.11-2016 standard.

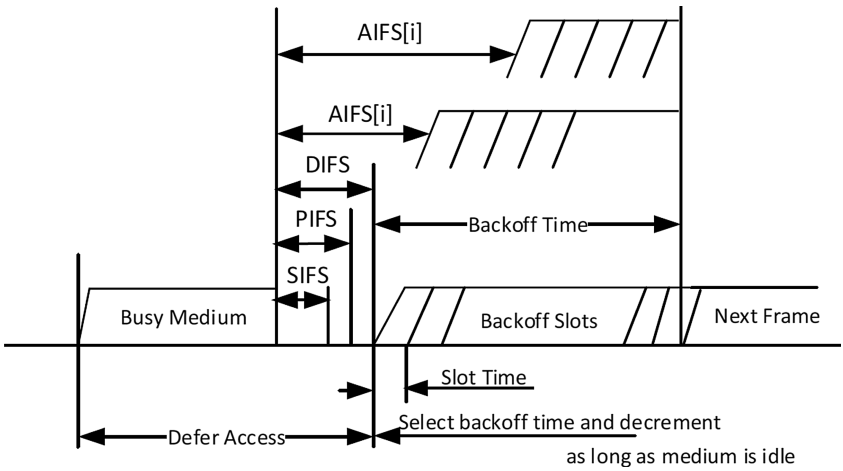

Fig. 1. 802.11 interframe space relationships

As explained in the standard: "The SIFS is the time from the end of the last symbol, or signal extension if present, of the previous frame to the beginning of the first symbol of the preamble of the subsequent frame as seen on the WM. The SIFS shall be used prior to transmission of an Ack frame…". The SIFS specified in the standard is 10 μs or 16 μs, depending on band (2.4 GHz or 5 GHz) and PHY type (DSSS, OFDM, etc.). For the 60 GHz band system, which has directional multi-gigabit ability, SIFS is only 3 μs.

Unlike Wi-Fi's CSMA/CA MAC mechanism, in LTE system everything is scheduled in advance, including acknowledgement of HARQ (Hybrid Automatic Repeat Request) process. Figure 2 shows LTE uplink HARQ procedure (Fig. 10.14 of [11]).

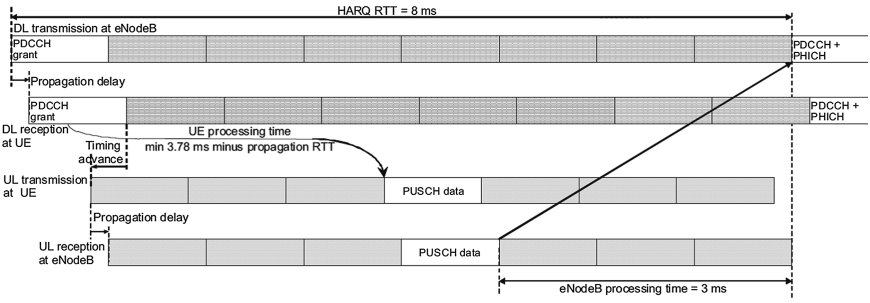

Fig. 2. Timing diagram of the uplink HARQ (SAW – Stop And Wait) protocol

In uplink HARQ, base station has 3 ms to make acknowledgement in downlink PHICH (Physical Hybrid ARQ Indicator Channel) after it receives uplink IQ samples of PUSCH (Physical Uplink Shared Channel). According to the C-RAN white paper [2], base station needs 800–900 μs after receiving IQ samples of each 1 ms. So the round trip latency budget between RRH and BBU is around 100–200 μs.

According to CPRI over fiber specification [3], maximum one-way latency of CPRI link is 5 μs. Besides CPRI latency, other latency overheads between RRH and BBU include physical distance, buffers, switches, cloud computer interface (Ethernet/PCIe/OS), etc. All together should less than round trip latency budget of C-RAN fronthaul. CPRI over Ethernet [12] is a hot topic in C-RAN fronthaul area, because not every places/areas have fiber coverage.

2.2 Latency Measurement Results of USRP + Host Computer

USRP (Universal Software Radio Peripheral) [13] SDR platform is the most widely used platform in research community. In most cases, it is used jointly with host computer, which can be (to some extent) regarded as a minimum version of C-RAN. The latency test is provided by the UHD (USRP Hardware Driver) [14] native example: latency_test [15]. Table 1 shows the measurement results of different USRPs combined with different computer communication links. Each latency result is measured and averaged over 10 round tests of 5 M, 10 M and 25 M sampling rates.

Table 1. Round trip latency between RF frontend and host computer software.

USRP type	Link type	Latency (us)	Host computer configuration
X310	PCIe	79	Intel i7-6700 3.4 GHz, NI PCIe ×4 card
X310	10 Gbps Ethernet	106	Intel E5-2650 v4 2.2GH, Qlogic 57810 Eth
X310	1 Gbps Ethernet	101	Intel i7-6700 3.4 GHz, Intel i219-v Eth
B210/200mini	USB 3.0	66	Intel i7-6700 3.4 GHz, Intel controller
N210	1 Gbps Ethernet	103	Intel i7-6700 3.4 GHz, Intel i219-v Eth

The latency measurement results show that the host computer based SDR architecture can't meet SIFS requirement of Wi-Fi system, but it is good enough to be used for current LTE system, because latency of $\sim$100 μs only consumes tiny portion of LTE processing time needed by HARQ process. That is why there are already several host-computer based LTE systems, such as srsLTE [16], OAI (Open Air Interface) [17] and amarisoft [18], which work quite well. On the contrary, almost all Wi-Fi SDR implementations, such as NI 802.11 Application Framework [19] and WARP (Wireless Open-Access Research Platform) [20], use FPGA to do processing instead of host computer.

2.3 Latency Measurement Results of Xilinx Zynq-7000 SoC

Xilinx Zynq-7000 All Programmable SoC (System on Chip) [21] includes two parts: PL (Programmable Logic) and PS (Processing System). PL actually is a traditional FPGA, which can do computation intensive operation and latency critical tasks. PS is a multi-cores ARM cortex AP (Application Processor), which is suitable to run control program, higher layer protocol and oeprating system. PL and PS are connected by multiple AXI (Advanced eXtensible Interface) high speed buses, which have low latency and high throughput performance. We evaluate its latency using a dummy FPGA block interacting with an ARM testing program in bare metal mode.

The testing system was constructed as in Fig. 3. The dummy FPGA acceleration block was designed by Xilinx HLS (High Level Synthesis) tool. The block receives data from PS in streaming manner via Xilinx DMA (Direct Memory Access) module, and stream the processed result (same amount of data as input) back to PS via DMA. It takes 654 clock cycles for this FPGA block to process 128 input samples, each sample is represented as a 32-bit word on the 32-bit AXI stream bus. The configuration interface between PS, PL and DMA controller is AXI_LITE, which is connected to M_AXI_GP port of ARM. Data link is AXI stream, which is connected to S_AXI_HP port of ARM. A DMA controller was used to convert AXI Memory Mapped interface (needed by PS) to AXI Stream interface (needed by streaming mode FPGA accelerator).

ZC706 Evaluation Board [22] for the Zynq-7000 XC7Z045 SoC is used to do the evaluation. Clock speeds of our design are as follows: AXI buses and PL run at 200 MHz, and ARM cortex-A9 processor run at 800 MHz. Xilinx ILA (Integrated Logic Analyzer) is inserted to the design for event recording with 5 ns resolution. The ARM software event is recorded by writing special value to PL register and detecting this value via ILA. The latency profiling result is shown in Fig. 4.

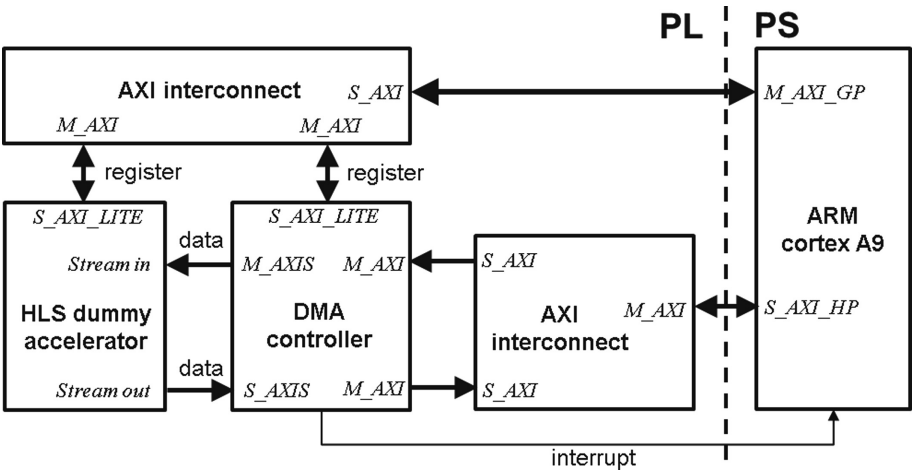

Fig. 3. Diagram of Latency test for Zynq-7000 Platform. M_* – master AXI interface; S_* – Slave AXI interface; AXIS – AXI streaming interface; *_GP – General Purpose (for register read/write); *_HP – High Performance (for data transfer)

Figure 4 shows the event log of the entire round trip delay test captured by Xilinx ILA. First, the software start DMA transmission at −0.135 μs; then DMA controller receives the instruction at 0 μs via AXI LITE register interface; After some internal preparation and buffering operation, the actual DMA transmission on the AXIS bus starts at 0.3 μs; After 3.295 μs, which is caused by the accelerator processing latency of 654 clocks at 200 MHz, the accelerator completes the data transfer of processing results back to DMA controller in streaming manner; Then DMA controller raises interrupt to PS at 4.48 μs; Finally, software becomes aware of this event at 4.595 μs. So the round trip latency between the FPGA accelerator and software in PS is 4.595 + 0.135 − (3.595 − 0.3) = 1.435 μs. Note that this is superior performance comparing to the latency of USRP variants, and it is even not a significant overhead compared to the Wi-Fi SIFS requirement (16 μs or 10 μs).

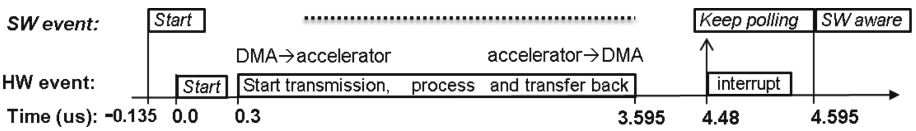

Fig. 4. Latency test result of Zynq-7000 SoC Platform

3 Architecture Design

Hardware/FPGA implementation or hardware/FPGA accelerated software is necessary to achieve realtime operation of certain wireless standards, such as Wi-Fi. Even for mobile operator, in the era of beyond LTE, 5G is also considering much lower latency

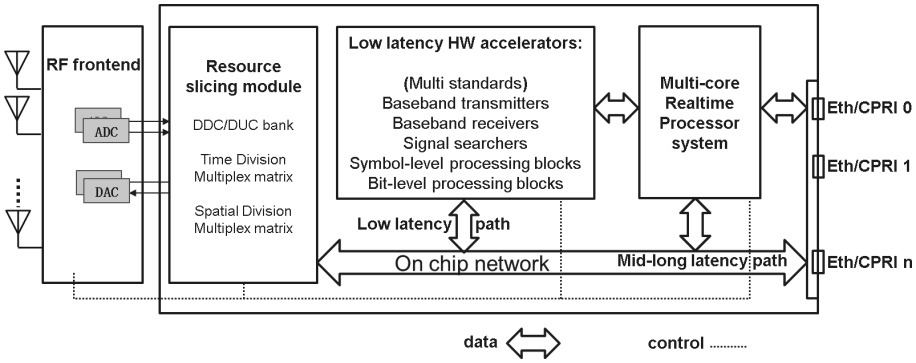

Fig. 5. ORCA Architecture supports both hardware-like low latency performance and software-like flexibility

(at sub millisecond) than LTE. All these trends imply that pure centralized/cloud based software solution won't be the silver bullet. However, simply using pure ASCI/FPGA design to meet performance/latency requirement will lose software-like flexibility, such as virtualization and effective coordination among channels and standards. Exploring an architecture, which has both hardware/FPGA performance and software flexibility, would be a more attractive option for next generation wireless network.

Figure 5 shows an architecture design to meet diverse requirements in a flexible way. First, the wideband multi-antenna RF frontend is connected to a resource slicing module. Then, resource slicing module handles IQ samples to/from specific slice, such as a channel, time slot, antenna/beam, under the software control of PS. Next, slice specific IQ samples are routed via on chip network to/from different destinations according to latency requirements. If the IQ samples belong to a low latency service (UAV/self-driving-car control) or standard (Wi-Fi), their destination/source will be likely other on-chip hardware accelerators. For IQ samples of radio slices which carry mid-long latency service (watching TV) or standard (LTE), they can be routed to pure software domain: either on-chip processor system or off-chip network interface (Ethernet, CPRI, etc.) until a processing unit in a host computer is reached.

To support pure-software-like virtualization and flexibility characteristic for the on-chip hardware accelerator, the accelerator design and on-chip real-time scheduling are the two key enablers. Besides the fact that the accelerator itself needs to meet latency/performance requirement, but also be general enough so that it can be shared by multiple standards. Thanks to that OFDM has been widely adopted by many standards, different standards do share some common processing units. Furthermore, a special issue needs to be addressed when sharing accelerators among multiple slices, this is context saving and restoring of the accelerators. Because when an accelerator is used among multiple slices in TDM (Time Division Multiplex) mode, it has to maintain context internally for each slice it serves. This is necessary to resume accelerator execution for a specific slice from its previous state before interruption. Needless to say, the software design to schedule diverse accelerators for different slices is also a challenging task on its own. This software is essential to achieve effective spectrum utilization with multiple standards/channels coordinated.

4 Demonstration

A demo is setup as shown in Fig. 6 to proof the concept of radio hardware virtualization. The SDR platform is composed by Xilinx zc706 evaluation board together with Analog Devices fmcomms2 RF front end [23]. A host computer is connected to the SDR platform for visualization and processing mid-long latency task. Three standards are covered in the demo: Wi-Fi/802.11 20 MHz mode, Zigbee/802.15.4 2.4 GHz version, and BLE.

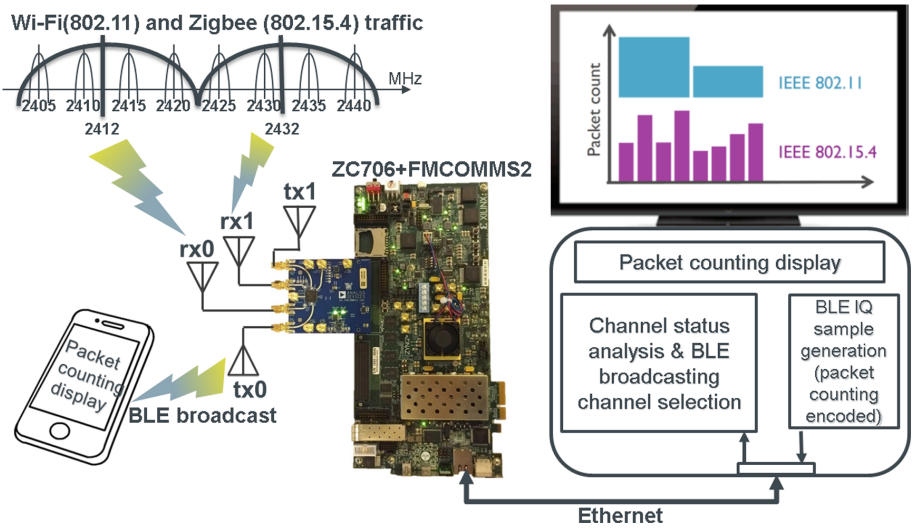

Fig. 6. Radio hardware virtualization demo setup

Ten virtualized preamble detection instances and one BLE (Bluetooth Low Energy) [24] broadcaster instance are created from resource slicing module and a dual-mode preamble detector (as accelerator) in FPGA under the scheduling software in ARM processor. RF front end has two receiving antennas and two transmitting antennas(rx0, rx1 and tx0, tx1 in Fig. 6), and works in 40 MHz bandwidth mode. For each rx antenna, resource slicing module creates one 20 MHz Wi-Fi channel and four 5 MHz Zigbee channels (overlapped with Wi-Fi) by five DDC (Digital Down Converter). Central frequency of each rx antenna's DDC can be tuned independently inside the range of 40 MHz bandwidth. In the demo, rx0 is tuned to cover upper 20 MHz of 40 MHz; rx1 is tuned to cover lower 20 MHz of 40 MHz. Ten (five per rx antenna) slices' IQ sample are routed to ARM processor, then fed one by one to the preamble detector by control software for preamble searching. 9.6Gbps AXI bus is used to construct on chip network. Dedicated AXI links are setup between resource slicing, preamble detector and ARM processor.

ARM reports Wi-Fi and Zigbee packet counting results of each slice to host computer via Ethernet. A program in computer analyzes channel status according to

packet counting report, then it will: (1) select a good BLE broadcasting channel from three options: 2402 MHz, 2426 MHz, 2480 MHz; (2) generate IQ samples for BLE broadcasting packet [25] with Wi-Fi and Zigbee packet counting information encoded; (3) Send BLE IQ samples to ARM processor in the SDR platform via Ethernet. Resource slicing module broadcasts the BLE packet into the air via the right tx slice (frequency, timing and antenna) under ARM software control. Finally, a user can read the BLE message, which contains Wi-Fi and Zigbee packet counting information, on their equipment (phone, pad) screen by general purpose BLE scanner App, for instance, LightBlue and BLE Scanner. In the demo, BLE broadcasting is an example of mid-long latency service which is handled by host computer instead of FPGA.

A dual-mode preamble detector (16-sample auto-correlation algorithm) is designed based on similar structure of Wi-Fi and Zigbee's preamble. According to the standards, at Wi-Fi baseband rate 20 Msps and Zigbee half baseband rate 1 Msps, both preambles have the form of repeating 16-sample random signal. The only difference is that: Wi-Fi' preamble has 160 samples which is generated by repeating 16-sample signal 10 times; Zigbee repeats 8 times, which results in 128-sample preamble.

C language is used to develop both FPGA and ARM software. By Xilinx HLS C, resource slicing and preamble detector blocks achieve 200 MHz clock speed. AXI bus run at 64-bit 200 MHz mode. Each AXI 64-bit word contains two IQ samples (16-bit I and 16-bit Q sample for each antenna). To process 512 samples, which means 25.6 μs under Wi-Fi baseband rate, 512 μs under half Zigbee baseband rate, preamble detector needs only 1094 clocks, i.e. 5.47 μs, according to FPGA synthesis report. So, it is fast enough to handle two Wi-Fi slices and ten Zigbee slices. Because the speed of accelerator consuming IQ sample is higher than speed of ten rx slices generating IQ sample. The whole processing system won't lose any IQ-sample/signal, as if there are ten preamble detector running in fully parallel (logically). Different from technology in [26], multiple virtual AP (Access Point) share one Wi-Fi channel by time division, our implementation allows running multiple APs in multiple channels concurrently without implementing standalone FPGA block for each channel.

For the control software, a basic control slot length is 25.6 μs (512 samples of Wi-Fi), which is also the basic time slot length of resource slicing. 20 slots compose a control period. Different tasks, such as ten preamble detection tasks, one BLE broadcasting task, host computer communication task, tx/rx frequency tuning task, are scheduled in different time slots according to upstream producing rate and latency requirement, just like a computer running multiple programs. By carefully arranging the schedule, there are ten rx instances and one tx instances running in fully parallel from the end user point of view. In this way, a set of radio hardware resources (RF front end, accelerator, etc.) get virtualized to multiple instances.

5 Conclusion and Future Work

ORCA project tries to push the virtualization in cloud/host-computer domain to radio hardware level to solve the spectrum sharing issue under diverse wireless access scenario. By building a platform with a software-hardware co-design philosophy, developer can use the platform to create multiple concurrent virtualized instances from the

low latency high performance hardware/FPGA accelerator. A hierarchical latency handling methodology is proposed to utilize different characters of hardware/FPGA and host computer. With this architecture, multiple wireless accesses run fully in parallel, just like running multiple programs on the same CPU, to maximize utilization of FPGA accelerators. A demo showcase is made to proof the concept by running concurrent two Wi-Fi, eight Zigbee and one BLE instances in 40 MHz bandwidth from the same set of FPGA resource. During the demo development, high level synthesis and processor controlled on chip network are used to shorten the development cycle significantly compared with traditional HDL (Hardware Description Language) based development method. Demonstration results show that this high level design methodology can also generate high performance FPGA blocks.

In the future, software framework and API will be refined in a more formatted way to abstract on chip resources, such as accelerator and on-chip high-speed network. By doing so, FPGA developer and software developer can design platform compatible accelerator and virtualized wireless access service in the more efficient and coordinated way.

Acknowledgment. The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732174 (ORCA project).

References

1. Wyglinski, A.M., et al.: Revolutionizing software defined radio: case studies in hardware, software, and education. IEEE Commun. Mag. **54**(1), 68–75 (2016)
2. C-RAN: The Road Toward Green RAN. http://labs.chinamobile.com/cran/wp-content/uploads/2014/06/20140613-C-RAN-WP-3.0.pdf
3. Common Public Radio Interface (CPRI); Interface Specification. http://www.cpri.info/downloads/CPRI_v_6_0_2013-08-30.pdf
4. Checko, A., Christiansen, H.L., Yan, Y., Scolari, L., Kardaras, G., Berger, M.S., Dittmann, L.: Cloud RAN for mobile networks a technology overview. IEEE Commun. Surv. Tutorials **17**(1), 405–426 (2015)
5. de la Oliva, A., Hernandez, J.A., Larrabeiti, D., Azcorra, A.: An overview of the CPRI specification and its application to C-RAN-based LTE scenarios. IEEE Commun. Mag. **54** (2), 152–159 (2016)
6. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. http://standards.ieee.org/getieee802/download/802.11-2016.pdf
7. IEEE Standard for Low-Rate Wireless Networks. http://standards.ieee.org/getieee802/download/802.15.4-2015.pdf
8. Taneja, M.: 802.11ah—LPWA interworking. In: Proceedings of IEEE NetSoft Conference on Workshops (NetSoft), Seoul, South Korea, pp. 441–446, June 2016
9. Lora Alliance. https://www.lora-alliance.org/
10. Sigfox. http://www.sigfox.com/
11. Sesia, S., Toufik, I., Baker, M.: LTE - The UMTS Long Term Evolution From Theory to Practice, 2nd edn. Wiley, Hoboken (2011)

12. Wan, T., Ashwood-Smith, P.: A performance study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv enhancements. In: 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, pp. 1–6 (2015)
13. USRP. https://www.ettus.com/product
14. UHD. https://www.ettus.com/sdr-software/detail/usrp-hardware-driver
15. Latency_test.cpp in UHD. https://github.com/EttusResearch/uhd/blob/maint/host/examples/latency_test.cpp
16. srsLTE. https://github.com/srsLTE
17. Open Air Interface. http://www.openairinterface.org/
18. Amrisoft. https://www.amarisoft.com/
19. LabVIEW Communications 802.11 Application Framework. http://sine.ni.com/nips/cds/view/p/lang/en/nid/213084
20. WARP: Wireless Open Access Research Platform. http://warpproject.org/trac/
21. Xilinx Zynq-7000 All Programmable SoC. https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html
22. Xilinx Zynq-7000 All Programmable SoC ZC706 Evaluation Kit. https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html
23. FMCOMMS2. https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz/quickstart/zynq
24. Bluetooth Low Energy. https://www.bluetooth.com/specifications/adopted-specifications
25. BLE (Bluetooth Low Energy) transmitter and sniffer. https://github.com/JiaoXianjun/BTLE
26. Bhanage, G., Vete, D., and Seskar, I.: SplitAP: leveraging wireless network virtualization for flexible sharing of WLANs. In: Global Telecommunications Conference. IEEE (2010)