Int. J. Electron. Commun. (AEÜ) 120 (2020) 153230

Contents lists available at ScienceDirect

International Journal of Electronics and Communications (AEÜ)

journal homepage: www.elsevier.com/locate/aeue

Regular paper

CMCVT: A Concurrent Multi-Channel Virtual Transceiver

Muhammad Aslam *, Xianjun Jiao, Wei Liu, Ingrid Moerman

Ghent University – imec, IDLab, Department of Information Technology, Belgium

ARTICLE INFO

Article history:
Received 2 March 2020
Accepted 21 April 2020

Keywords:
Multi-channel transceiver
SDR
Radio virtualization
IoT
FPGA

ABSTRACT

State-of-the-art wireless Gateways (GW) used in Internet of Things (IoT) offer a single channel radio link, which limits the capabilities of the IoT network controlled by the GW, as the GW can only use a single channel at a time to communicate with the end-device(s). The quality of service (e.g., aggregate throughput, latency) offered by a single channel GW could be substantially improved by employing a multi-channel transceiver, which is capable of transmitting/receiving data on different radio channels simultaneously, particularly for larger wireless networks. However, current solutions available in both research and commercial communities only offer multi-channel receiver capabilities, and do not incorporate the multi-channel transmitter part. In addition, in terms of implementation, these multi-channel receivers duplicate single-channel hardware functionality. In this paper, for the first time, a novel concurrent multi-channel virtual transceiver is introduced. The virtual transceiver offers multi-channel capabilities and uses the same single-hardware hardware implementation for the Physical (PHY) layer by employing the virtualization technique. This new virtual transceiver concept is demonstrated for an IEEE 802.15.4 based 8 × 8 channel transceiver, implemented on an Field Programmable Gate Array (FPGA) of a modern Software Defined Radio and is compared with the existing duplication approach. The duplication approach consumes 9008 LUTs, and 12120 FFs, whereas the proposed approach occupies only 2959 LUTs and 2105 FFs, saving 67.15% LUTs and 82.63% FFs in comparison with the duplication approach. The experimental results reveal that the virtual transceiver provides the same performance (e.g., receiver sensitivity of −98.5dBm) as the transceiver achieved by duplicating the PHY layers but consumes much less hardware resources.

1. Introduction

Internet of Things (IoT) is a technology that interconnects things (e.g., objects, machines, etc.) and allows them to exchange information with each other with or without human intervention. No matter what type of IoT applications are considered, the IoT infrastructure in general consists of: (1) smart end-devices capable of processing, sensing, and actuating the environment, which are connected to the cloud via a network device with gateway capabilities (further referred to as IoT Gateway or IoT GW), (2) a IoT GW which manages the bidirectional traffic between smart devices and the cloud (or internet), (3) an data server infrastructure in the cloud responsible for storing, analysing, and processing the huge amount of data in real time, and (4) a user interface which is tangible, visible and accessible by users [1].

While the number of end-devices connected with the Internet in IoT network has proliferated significantly, currently, an IoT GW is the sole way of connecting them with the internet. It is mostly equipped with a single channel Transceiver (TRX), which is only capable of communicating with end-device(s) over a single channel at a time, which often appears to be a bottleneck. The single-channel TRX is adequate for point-to-point communication or for the scenario in which the number of smart devices directly connected with the IoT GW is limited. A single channel TRX based GW adversely affects the performance (in particular in terms of latency and throughput) of an IoT network when multiple smart devices try to access the GW at the same time. One way to improve the performance of the GW is to use concurrent multi-channel TRX. The advantage of the multi-channel TRX is that it has the capacity to transmit/receive data on more than one channel simultaneously. Such a multi-channel TRX will certainly outperform a single-channel TRX when a large number of devices are connected. It is important to note that the multi-channel TRX is different from a multi-band TRX. A concurrent multi-band TRX operates simultaneously in different frequency bands (e.g., 2.4 GHz or 5 GHz band), but utilizes a single channel in each band at a time [2]. While, the multi-channel TRX utilizes multiple frequency channels within a single frequency band.

* Corresponding author.
E-mail address: muhammad.aslam@ugent.be (M. Aslam).

https://doi.org/10.1016/j.aeue.2020.153230
1434-8411/

There are already many multi-channel radio sniffers available in the IoT networks, which can receive data on multiple channels simultaneously. These sniffers are achieved either by using many single channel commercial off-the-shelf chips or by duplicating the same hardware on a Software Defined Radio (SDR) [3,4]. An SDR is a radio communication system on which the radio's components are implemented in software on a host-computer or on a programmable hardware device (e.g., Field Programmable Gate Array (FPGA), etc.). Since the sole purpose of these multi-channel solutions is sniffing packets on the channels, they do not incorporate a multi-channel transmitter. The solutions do not only lack multi-channel transmitter part, but also underutilize the potential processing capabilities of a modern SDR or Application Specific Integrated Chip (ASIC). For example, the parallel processing feature of the FPGA in a modern SDR enables it to process the data at a rate far higher (e.g., the FPGA in Zynq 7000 can provide a maximum DSP performance of 3,634 Giga Multiply-Accumulates per Second (GMACS) [5]) than that required by the wireless protocols in IoT (e.g., IEEE 802.15.4 standard in the 2.4 GHz band demands about 2.1 GMACS). It is worth noting that the value of 2.1 GMACS for IEEE 802.15.4 is calculated by recompiling the works implemented in [6,7].

In this paper, to our best knowledge, we introduce for the first time a complete multi-channel TRX capable of both transmitting and receiving the data on multiple channels concurrently. We have applied Hardware Virtualization (HV) in our multi-channel TRX prototyping by fully exploiting the high processing capability of a modern SDR platform. HV is a technology that allows users to create multiple logical instances from a single physical instance implemented on hardware. It enables us to use the same hardware of single channel TRX for multiple channels with only a limited amount of extra logic added to manage the virtualization overhead.

To this end, we have prototyped the 8×8 multi-channel TRX on an SDR in two approaches:

- **The Hardware Duplication (HD) approach**, in which we have simply duplicated the hardware of a single channel IEEE 802.15.4 compliant TRX for multiple channels on the SDR. The HD approach is implemented as a benchmark to evaluate the proposed approach.
- **Our proposed Hardware Virtualization (HV) approach**, in which we have used the same hardware of single channel TRX for multiple channels, with some extra logic added to manage the virtualization overhead.

By prototyping and comparing these two approaches, it is validated that the HV approach is not only efficient in terms of hardware utilization but also provides the same performance as the HD approach.

2. Related work

In the first section, SDR based multi-channel solutions presented in the literature are investigated. Next, commercially available state-of-the-art chipset based solutions are discussed.

2.1. SDR based solutions

An IEEE 802.15.4 multi-channel Receiver is described in [3]. This solution uses USRP2 [8] to capture the packets of 5 contiguous channels located in the 2.4 GHz ISM band. Further, a dedicated Digital Down Converter (DDC) and demodulator is implemented for each channel in GNU Radio [9]. The major downside of the solution is that it allocates a dedicated DDC and demodulator for each channel. The duplication of these processing modules increases the load on a Central Processing Unit (CPU), which can cause samples of USRP to overflow and packets not being decoded. Yohe and co-authors [10] have presented Multi-Protocol Access Point (MPAP), HV Architecture for Heterogeneous Wireless APs. In this work, an 802.11 g and two 802.15.4 radio receivers are integrated on the Sora platform [11], where IQ samples received by wide-band Radio Frequency (RF) front-end are fed into multi-core Personal Computer (PC). The PHY and upper layers of the respective standards are running on the PC. The incorporation of the PC not only makes the platform inappropriate for the solutions acquiring ultra-low latency (e.g., factory automation) but also causes it to be costly and bulky. Jiao et al. [12] has applied radio HV on System on Chip (SoC), where DDC banks, related PHY layers are implemented in FPGA, and corresponding scheduling software is running on an embedded processing unit. The authors claim that the design can decode 2 Wi-Fi and 8 IEEE 802.15.4 channels concurrently, but the implementation is limited to preamble detection. In other words, it does not decode the complete packets of the protocols.

2.2. Commercial radio transceiver based solutions

A Multi-channel packet sniffer is developed in [4], where multiple IEEE 802.15.4 based commercial off-the-shelf radio are adopted to capture the packet. The architecture inherits two disadvantages: (1) it employs a dedicated radio device for a channel and all the devices have their own clock source, leading to the addition of a time synchronizer to overcome clock drifting, and (2) non-compact, because the developer has to use a separate commercial radio for each channel. Similarly, a special probe has been introduced for multi-channel packet sniffer in [13]. The probe is composed of an FPGA to act as a supervisor, and 15 IEEE 802.15.4 compliant commercial radios. In order to maintain the synchronization among the radio radios, a common clock source is implemented in FPGA. The probe has fixed the clock drifting issue, but it still have the above-mentioned non-compact issue. A multichannel Wi-Fi sniffer able to decode multiple consecutive channels in 2.4 GHz and 5 GHz is designed by Pradeep and his colleagues [14]. They have also employed a dedicated commercial chip for each channel.

In summary, there exists two categories of related work for our multi-channel transceiver, i.e. the solutions based on SDR and the solutions based on commercial chipsets. In both categories, there is no candidate capable of multi-channel transmission, all solutions are receiver only. The receivers on multiple channels are achieved by duplicating software/hardware modules, leading to (i) the consumption of extra resources, (ii) the increase of cost and form factor (in the case of commercial chipset based solution), and (iii) possibly extra complications such as the need to overcome clock drifting across multiple hardware platforms. The novelties of our solution are following:

- Our solution CMCVT consists of a single PHY implemented on FPGA of an SDR. It is capable of both transmit and reciver.
- The concurrent multi-channel operation is achieved by HV, taking full advantage of the high processing capabilities of the modern SDR.
- Lastly, because the CMCVT is working on a single device, it does not have the clock drifting/synchronization issue.

3. SDR based Concurrent Multi-Channel Virtual Transceiver

A multi-channel TRX obtained by using HD approach is referred to as Conventional Multi-Channel Transceiver (CMC-TRX) and is used as a reference for comparison; while our proposed Concurrent Multi-Channel Virtual Transceiver (CMCVT) is achieved by using the HV approach. Fig. 1 highlights a general comparison between

the reference HD approach and our HV approach. The common components shared in both approaches are wideband RF front-end, upper layers running on a processor, Digital Up Converter (DUC) which converts the baseband signal to some Intermediate Frequency (IF) signal, and DDC which converts IF signal back to the baseband signal. The main differences between the HD and the HV based architectures is the implementation of Baseband Processing Unit (BBPU) of the PHY layer. As depicted in Fig. 1(a), the HD approach assigns a separate BBPU to each channel, while in HV approach, the same BBPU is timely shared among multiple channels. We apply the overclocking concept in the HV approach: the PHY layer is running at $N \times CLK_{bb}$, where N and CLK_{bb} represent the number of channels and baseband data rate of a wireless standard (e.g., 250kbps is the CLK_{bb} in IEEE 802.15.4 standard), respectively.

Fig. 1(b) elaborates the internal block diagram of HV approach for CMCVT (see yellow box in Fig. 1). We propose three simple but effective steps to convert any single channel transceiver into HV based multi-channel transceiver: (1) obtain a single channel PHY layer of any wireless standard. For this paper, we have recompiled and modified the existing single channel implementation [6,7] written in Hardware Descriptive Language (HDL) of IEEE 802.15.4 standard [15]. Without the loss of generality, among the three different PHY modes (i.e., 20 kbps, 40 kbps, 250 kbps) specified in the standard, we have chosen the PHY layer of data rate 250kbps functioning in 2.4 GHz band for the implementation. It is worth mentioning that the chosen PHY layer uses Orthogonal Quadrature Phase Shift Keying (O-QPSK) as a modulation scheme, (2) Identify and expose the context saving and restoring signals involved during context switching of the single channel PHY layer. Context switching is a commonly used technique in CPU domain in which context or states of a process are stored so that they can be restored at a time when CPU resumes the execution of the process. The context switching is a feature of multitasking, which enables a CPU to be shared by multiple processes. (3) Implement a Context Switching Finite State Machine (CS-FSM) for the context switching. The FSM is the most important part of our HV approach. It is responsible for the correct functioning of the CMCVT. These three simple steps enable us to convert CMCVT from a conceptual idea into a real-life solution. It is obvious that each CMCVT's PHY layer can be further decoupled into transmitter's PHY and receiver's PHY layers, both are detailed in the following sections.

3.1. Physical layer of the Multi-Channel Virtual Transmitter

A detailed diagram of the Multi-Channel Virtual Transmitter (MCVT) is elaborated in Fig. 2. The PHY layer of MCVT implemented in FPGA is composed of DUC bank, and BBPU.

3.1.1. The working flow of the Multi-Channel Virtual Transmitter's PHY layer

The working flow of the MCVT is as follow:

- The Medium Access Control (MAC) layer running on the embedded Processor System (PS) configures the BBPU and the RF front-end (see control path in Fig. 2). For instance, the sampling

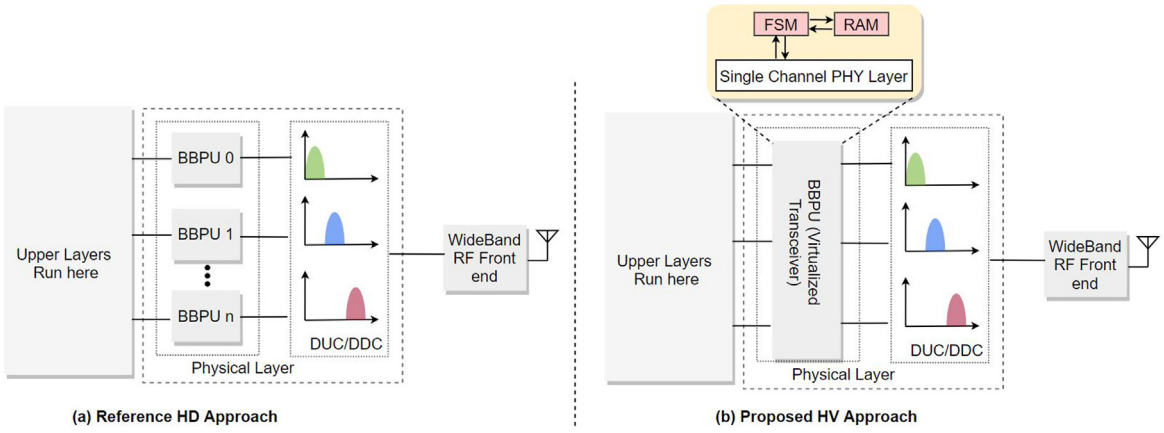

Fig. 1. A block diagram showing a general comparison between (a) the duplication approach to implement the multi-channel transceiver and (b) the proposed HV approach to implement the multi-channel transceiver.

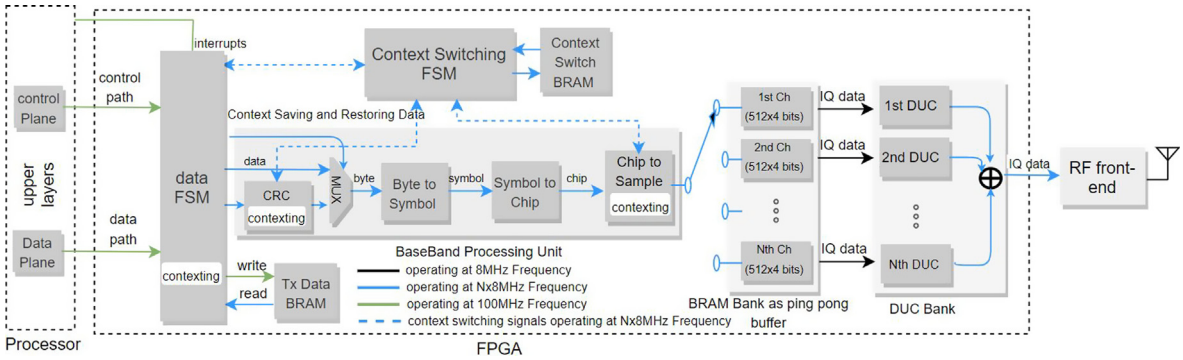

Fig. 2. A detailed diagram showing all the modules involved in realization of the MCVT.

frequency, bandwidth of the RF front-end, and the number of potential channels on which data is to be transmitted in BBPU are amongst the configurable parameters.
- Then, the MAC layer starts sending data to BBPU through DMA (see data path in Fig. 2). The BBPU keeps storing the data into Random Access Memory (RAM), until it receives the corresponding data of all the channel(s) defined in step (1).
- Lastly, the BBPU initiates the transmission, and by generating interrupts, it updates the MAC layer about the status of transmission.

3.1.2. Implementation of the Multi-Channel Virtual Transmitter's PHY layer

The HV is applied for the BBPU. Instead of allocating a separate BBPU for each channel, the HV allows to utilize the same BBPU for multiple channels. We leverage expertise in multitasking, pipelining, and multi-clock domains to achieve virtualization of the BBPU. The technologies are detailed in the following sub-sections.

3.1.2.1. Multitasking. Similar to multitasking in CPU, our design also inherits the context storing-restoring problem. In principle, any functional block/program that has delay, memory, pipeline, and internal states in multitasking requires context saving and restoring operation. To this end, we introduce a CS-FSM and Block RAM (BRAM) in our design (as shown in Fig. 2). In the BBPU, we have identified that only three modules require context switching, namely: data FSM (due to internal states), Cyclic Redundancy Check (CRC) (due to memory), and chip to sample (due to delay). After each processing time unit, hereafter referred to as a tick, the context of these modules needs to be saved and the context for next channel needs to be restored. The CS-FSM is responsible for all these operations. Fig. 3 shows the state diagram of the FSM. At the beginning, the FSM enters into the first state (indicated by state 0 in the Fig. 3) when an 'Enable' signal is high. Then, the FSM instructs the BBPU to start processing the data, meanwhile it enables corresponding BRAM in the BRAM bank (see Fig. 2)) to read/write the IQ samples. After a tick, the FSM enters into second state (indicated by state 1 in Fig. 3) and during the transition, it stores states of the current channel and restores states of the next channel. The FSM keeps the record of the current and next channel number in an 8 bit register (indicated by 'ch' in the Fig. 3). When channel number reaches maximum supported channels (indicated by 'Max_ch' in the Fig. 3), FSM is wrapped around to the first channel. Instead of using a dedicated state for each channel, the FSM always has three states, irrespective to the number of channels that the MCVT supports. This FSM design ensures the correct operation of the MCVT and at the same time offers high efficiency in terms of hardware utilization.

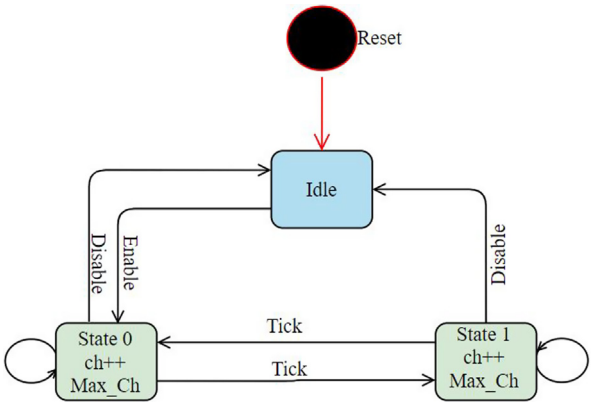

Fig. 3. The state diagram of context switching finite state machine.

3.1.2.2. Pipelining. Taking advantage from the modular structure, we have further included pipelining in the BBPU. The pipelining helps to reduce the clock overhead caused by context saving and restoring operations. An example of pipelining for n channels is illustrated in Fig. 4, where horizontal axis represents the time and vertical axis reflects the modules of the BBPU. At t0 time, *CRC and multiplexer* module (indicated by M0 in Fig. 4) begins processing the data of First Channel (CH1). After a tick (i.e., the time equivalent of processing 256 IQ samples), M0 switch to CH2, meanwhile the *byte to symbol* module (indicated by M1 in Fig. 4) is enabled for CH1. During the third tick (i.e., after t3), all the modules are busy in processing the data of consecutive channels. The output of the *chip to sample*(indicated by M3 in Fig. 4) module is stored into BRAM, which decouples the BBPU from the DUC bank. M3 generates 256 symbols after each tick. There is a separate BRAM for each channel with each BRAM has the capacity to store 512 symbols. The BRAM behaves as a ping pong buffer, which means the RAM can accept new symbols in the 257–512 locations while sending out symbols in the first 256 locations, and vice versa. This configuration prevents the DUC of a channel from processing the wrong IQ symbols.

3.1.2.3. Multi-clock domains. To meet the critical constraints of the specific standard, the MCVT uses three different clocks as indicated in Fig. 2: (1) the clock specified for control and data paths, which is 100 MHz in our design (2) the clock at which BBPU is operation, running at $N \times CLK_{bb}$ (CLK_{bb} is 8 MHz in our design), and (3) the clock at which each DUC in DUC bank reads the data from ping pong BRAM at CLK_{bb} rate. Due to multi-clock domains, the MCVT is prone to metastability. To mitigate the metastability situation, we have introduced 2-flop synchronizer and dual port BRAM for single-bit, and multi-bit data signals, respectively.

3.2. Physical layer of the Multi-Channel Virtual Receiver

Unlike MCVT, turning the Multi-Channel Virtual Receiver (MCVR) from conceptual idea into a working solution is way more

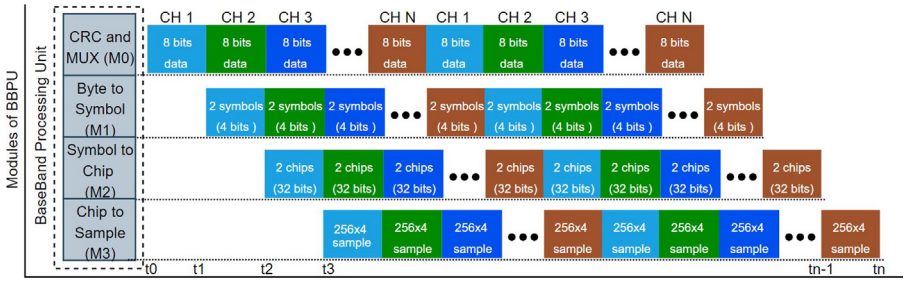

Fig. 4. Applying pipelining in the Multi-Channel Virtual Transmitter.

challenging. Almost all the modules constituting the BBPU of MCVR needs context saving and restoring, adding extra complexity in the architecture. A detailed diagram of the MCVR is presented in Fig. 5, where the corresponding PHY layer incorporates BBPU and DDC bank is realized in FPGA.

3.2.1. The working flow of the Multi-Channel Virtual Receiver's PHY layer

The working flow of the MCVR is as follows:

- After the configuration of PHY layer and RF front-end by the upper layer (indicated by control plane in Fig. 5), the DDCs of corresponding channels in DDC bank first shift the center frequency of IQ samples, down sample and then write the incoming samples to the BRAMs.
- The BBPU begins to decode the samples read from the first BRAM ping pong buffer.
- After a tick (i.e., the time equivalent of reading 8 IQ samples), the "sample reading multiplexer" switches to the second BRAM, meanwhile CS-FSM concurrently performs context saving for the current channel and restoring for the subsequent channel.
- In order for the upper layer to recognize to which channel an incoming packet belongs to, the BBPU appends one extra byte to the decoded data representing the channel number.

3.2.2. Implementation of the Multi-Channel Virtual Receiver's PHY layer

Similar to MCVT, the HV is only applied on BBPU of MCVR. Likewise, the MCVR has benefitted from multi-clock domain, pipelining and multitasking, which together alleviate the clocks overhead added during context saving and restoring. All these technologies are already thoroughly explained in MCVT's implementation section. In addition to the implementation of MCVR, it requires an Automatic Gain Controller (AGC) for each channel that tunes the signal strength to compensate for channel-specific losses and fading required for proper decoding. Since the analog AGC (see RF front-end in Fig. 5) of the used RF front-end works on the wideband RF signal, it is observed that it fails to provide enough signal strength for each channel required to accurately decode it. The descried issue is mitigated by implementing a dedicated digital AGC for each channel (see DDC bank in Fig. 5), which together with analog AGC enables the MCVR to correctly decode the data in all possible situations.

4. Results and discussions

We have combined the MCVT and MCVR into the CMCVT, in which the corresponding BBPUs can independently transmit/receive data on multiple channels. The SDR chosen in the particular setup is composed of ZedBoard [16] and FMCOMMS2 board [17]. ZedBoard is a low-cost development board for the Xilinx Zynq 7000 SoC [18] and the SoC is further comprised of Programmable Logic (PL) (an alternative term for FPGA) and PS. The PHY layers of CMCVT are implemented in the PL part, while FMCOMMS2 board is used as analog RF front-end. A single channel TRX proposed in [6] is used as a building block to implement a multi-channel TRX. Since the sampling rate of the TRX in [6] is 8 MHz and maximum sampling rate supported by FMCOMMS2 is 64 MHz, the implementation of CMCVT is restricted to 8 parallel channels. Similarly, the implementation uses the same Digital to Analog Converter (DAC) in MCVT multi-channels decreasing the transmission power on each channel.

The CMC-TRX, used as a benchmark for comparison, is realized by duplicating the single channel TRX in [6]. Like every design in FPGA, our BBPUs have logic and memory parts. The logic part of a design is mapped on Look-up Table (LUT) and Flip-Flop (FF), whereas the memory part is placed on RAM. The comparison between hardware utilization efficiencies of CMCVT and CMC-TRX is performed in terms logic and memory consumption. Instead of directly comparing the hardware utilization of CMCVT with CMC-TRX, MCVT and MCVR are separately compared against their corresponding conventional counterparts. The efficiency shown is calculated by using Eq. (1); it represents the improvement in hardware resource utilization of CMCVT relative to CMC-TRX.

$$Improved\ Efficiency = \left(\frac{HW_{conv} - HW_{our}}{HW_{conv}}\right) \times 100 \quad (1)$$

where HW_{conv} and HW_{our} represents the hardware consumed by the duplication and our approaches, respectively, and HW can be LUT, FF or RAM.

4.1. Evaluation of logic consumption

Table 1 depicts the comparison of hardware utilization efficiency of our MCVR against Conventional Multi-Channel Receiver (CMCR) obtained by using HD approach, while Table 2 illustrates the comparison of our MCVT against Conventional Multi-Channel Transmitter (CMCT). It is noteworthy that the tables only contain LUTs and FFs (i.e., they only include logic parts of the respective designs). It can be seen from Tables 1 and 2 that the benefit of the proposed HV approach is more pronounced for higher number of parallel channels. Under the setting of 2 channels, our approach interestingly provides either negative (in MCVR case) or slightly improved (in MCVT case) efficiency. The degraded efficiency for the particular case is expected, because our approach involves context storing-restoring, which requires extra logic. However, most part of the extra logic do not change as the number of parallel channels increases, resulting in more improvement in the hard-

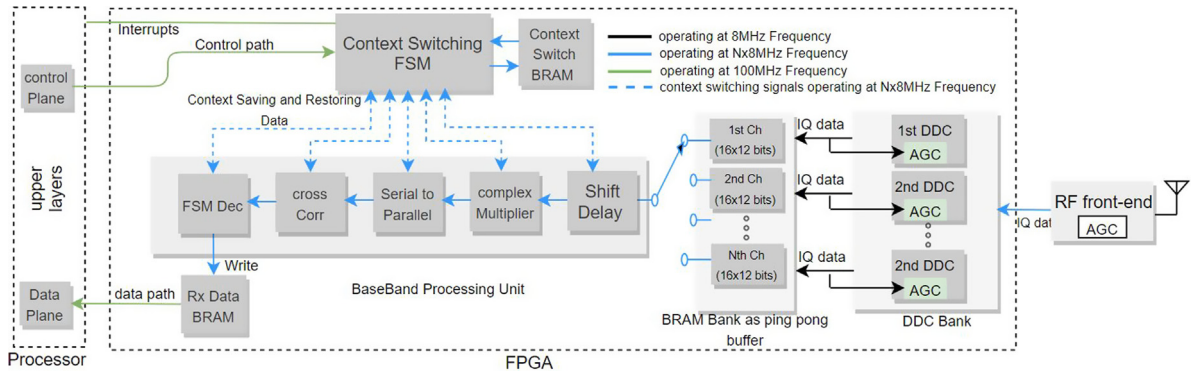

Fig. 5. A detailed diagram showing all the modules involved in realization of the MCVR.

Table 1
A comparison of hardware utilization efficiency of CMCR and our MCVR in term of logic consumption.

Channels	Resource	CMCR	MCVR	Improved Efficiency
1	LUTs	854	854	0
	FFs	1073	1073	0
2	LUTs	1708	2128	−24.59
	FFs	2146	1412	34.20
4	LUTs	3416	2168	36.53
	FFs	4292	1432	66.64
8	LUTs	6832	2220	67.51
	FFs	8584	1464	82.95

Table 2
A comparison of hardware utilization efficiency of CMCT and our MCVT in term of logic consumption.

Channels	Resource	CMCT	MCVT	Improved Efficiency
1	LUTs	272	272	0
	FFs	442	442	0
2	LUTs	544	532	2.21
	FFs	884	565	36.09
4	LUTs	1088	619	43.11
	FFs	1768	593	66.46
8	LUTs	2176	739	66.04
	FFs	3536	641	81.87

Table 3
A comparison of BRAM utilization of CMCR, MCVR, CMCT, and MCVT.

Channels	Resource	CMCR	MCVR	CMCT	MCVT
1	BRAMs	0.5	0.5	0.5	0.5
2	BRAMs	1	2	1	2
4	BRAMs	2	3	2	3
8	BRAMs	4	5	4	5

ware utilization efficiency for higher number of channels. It is logical that the best case is achieved for 8 channels, where MCVR saves 82.95% FFs and 67.51% LUTs as compared to CMCR (shown in Table 1), and MCVT reduces 81.87% FFs and 66.04% LUTs as compared to CMCT (shown in Table 2).

4.2. Evaluation of memory consumption

The PL part incorporates two types of RAMs; Distributed RAM (DRAM) and BRAM. LUTs can be configured as either logic or DRAM. In contrast to DRAM, BRAMs are dedicated RAMs and are located at fixed positions in FPGA. It is generally recommended to map memory part of the design on BRAM, so that more LUTs and FFs would be available for logic implementations. We therefore have mapped the memory parts of the designs onto BRAM. The BRAMs consumed in the design for different number of channels are summarized in Table 3. There are 140 BRAMs of size 36 Kb in ZedBoard FPGA, and the numbers in Table 3 represent the numbers of utilized 36 Kb BRAMs. In contrast to CMC-TRX that employs a single memory (as shown the data BRAM in Figs. 2 and 5) to read/store the packets, the CMCVT includes two extra memories; a context switching BRAM to store and restore the states of corresponding modules of the BBPU every time the BBPU switches to a new channel, and a Ping Pong BRAM to store the incoming/outgoing samples for further processing. Although these extra BRAMs potentially deteriorate the memory utilization efficiency of our approach, the increment is not significant. Our approach uses at most one extra BRAM than the duplication approach, as shown in Table 3.

4.3. Multi-channel receiver sensitivity measurement and analysis

Fig. 6 shows the magnitude response of Finite Impulse Response (FIR) used in the DDC of the CMCVT. It can be seen from the Fig. 6 that the FIR provides an attenuation for more than −33 dB after 5 MHz. The IEEE 802.15.4 standard defines a channel spacing of a 5 MHz among the adjacent channels when operating in 2.4 GHz band. Further, the values of adjacent and alternate channel interference rejection defined in the standard are 0 dB and 30 dB, respectively. Thus, the FIR provides the required adjacent channel interference rejection to our receiver during decoding the data from multiple channels simultaneously. Moreover, we have measured the receiver's sensitivity according to the requirements defined in IEEE 802.15.4 standard [15]. The input power at which the Packet Error Rate (PER) drops to 1% is termed as the sensitivity of a receiver. We have transmitted 50,000 packets with each containing 20 bytes in the air. The original implementation [6] has a receiver sensitivity of −98.5dBm and we have implemented 8

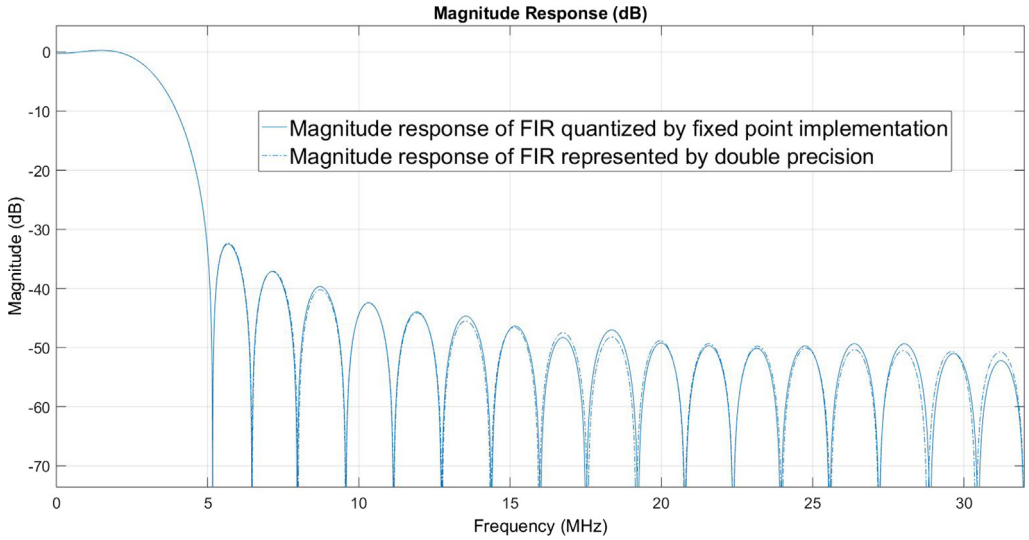

Fig. 6. Magnitude response of FIR used in DDC of the CMCVT. Where, double precision and fixed point magnitude responses are used as a reference and in the implementation of CMCVT, respectively.

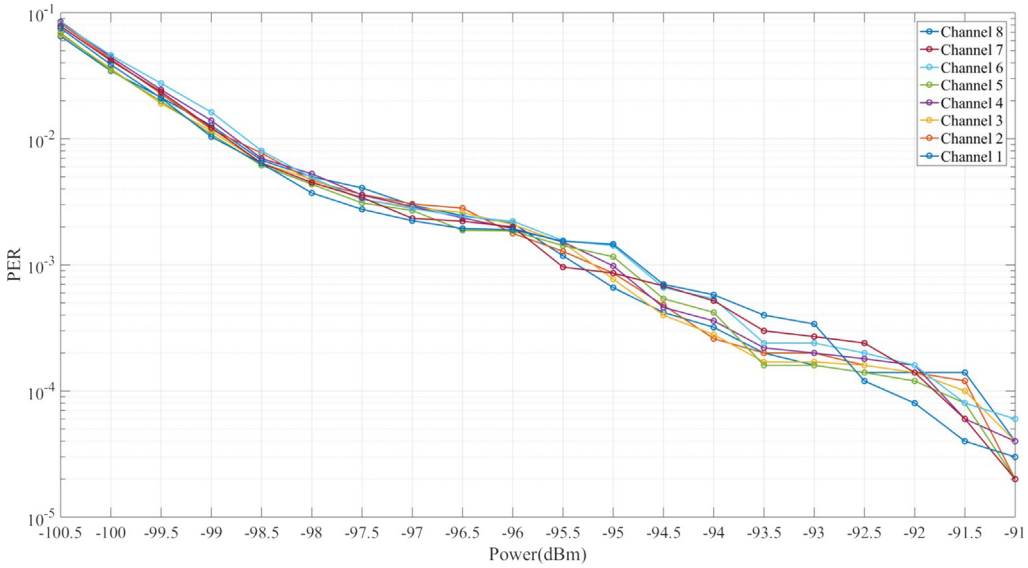

Fig. 7. Sensitivity measurements on all 8 channels of the multi-channel virtual receiver.

concurrent channels in MCVR by modifying the existing single channel receiver. We thus expect the same sensitivity values on all the 8 channels of the MCVR. To this end, we have measured the PER for each channel under a range of received power (in dBm). It can be seen in Fig. 7 that all 8 channels of the MCVR have the same sensitivity value which is −98.5 dBm. Fig. 7 however shows a minor difference of PERs among different channels. It is because the measurements are done on a real-life setup, and it is hard to achieve the same PERs even for the same channel.

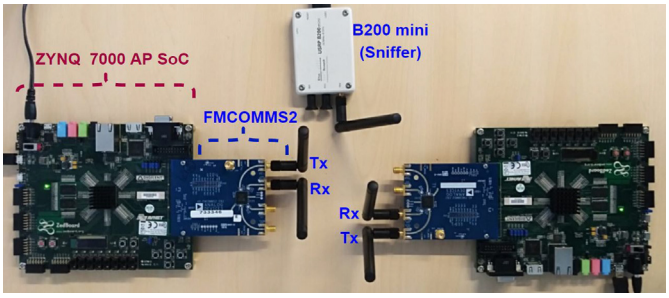

Fig. 8. An experimental setup for the real time validation of CMCVT.

4.4. Proof of concept experiment

After discussing the hardware utilization and sensitivity results of the CMCVT, an experiment is performed to validate the virtual TRX. To this end, two SDRs are used, each consisting of a ZedBoard and an FMCOMMS2 (see Fig. 8). A USRP B200 mini [8] is used to visualize the wideband power spectrum when CMCVT is transmitting concurrently on 8 channels. The power spectrum view is shown in Fig. 9, where the envelope of 8 signals are clearly visible. It shows that CMCVT is capable of transmitting data on multiple channels concurrently. Fig. 10 has displayed the packets decoded by the other SDR. As explained before, BBPU appends one extra

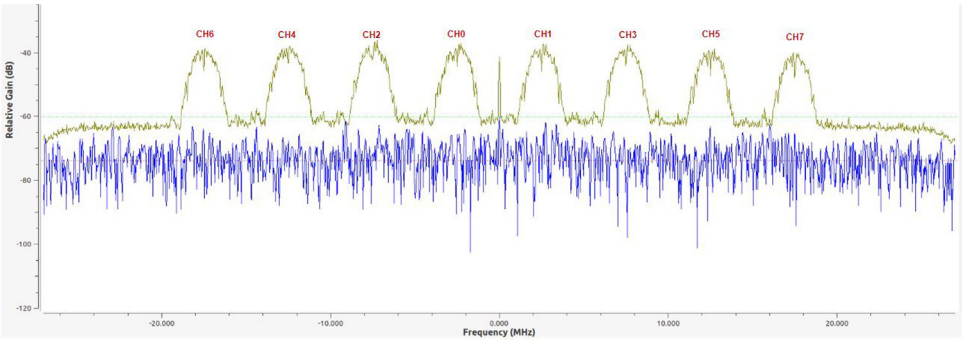

Fig. 9. Power spectrum view captured by a USRP B200 mini when the SDR (i.e., in MCVT mode) is sending packets on all 8 channels concurrently.

```
0  8  0 1 2 3 4 5 9b ed
1  c  1 2 3 4 5 6 7 8 9 a c5 94
2  10  2 3 4 5 6 7 8 9 a b c d e f 6a f4
3  14  3 4 5 6 7 8 9 a b c d e f 10 11 12 13 14 74 74
4  18  4 5 6 7 8 9 a b c d e f 10 11 12 13 14 15 16 17 18 19 ae 47
5  1c  5 6 7 8 9 a b c d e f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e a7 8c
6  20  6 7 8 9 a b c d e f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 87 a2
7  24  7 8 9 a b c d e f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 42 b5
```

Fig. 10. Packets decoded by the SDR when it is receiving the packets from 8 consecutive channels.

byte to the decoded packets to indicate the channel number (red box in Fig. 10). The second byte indicates the packet length as is defined in the standard (green box in Fig. 10), and the rest are the payload of PHY layer.

5. Conclusions

This paper introduces a multi-channel virtual TRX architecture, leveraging the concept of HV and the high processing capability of FPGAs or ASICs in modern communication devices. Instead of allocating a dedicated transceiver for each radio channel, our TRX creates multiple logical transceivers based on a single physical transceiver. Such a design and implementation is highly efficient in terms of hardware utilization, hence effectively reducing the silicon footprint in ASIC design. To validate this new concept, we have prototyped an IEEE 802.15.4 complaint multi-channel virtual TRX on an SDR. The virtual TRX behaves as multiple dedicated transceivers, which can independently transmit/receive data on up to 8 channels. Experimental evaluation of our virtual TRX reveals that it holds the same performance as the multiple physical TRXs on different channels, but 82.63% FFs and 67.15% LUTs hardware resources can be saved for the particular case of 8 channels.

Our proposed method is generic, easy to implement and can therefore be applied on any existing or future wireless standards in IoT domain. Any single channel TRX can be easily virtualized by performing the following 2 steps: (1) identify the context saving and restoring signals from the original design, (2) add an extra FSM and RAM for context storing and restoring operations. We have verified our HV concept in a single antenna scenario, but also other emerging wireless technologies can benefit from hardware virtualization, such as non-orthogonal multiple access and massive multiple-input multiple-output. As virtualization is entirely applied on the digital side of a transceiver, it fully is transparent to the RF part. In the future, we are planning to extend the proposed method for virtualization towards other wireless standards, and also the upper layers of the communication stack.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

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

References

[1] Desai P, Sheth A, Anantharam P. Semantic gateway as a service architecture for iot interoperability. In: IEEE International Conference on Mobile Services, New York, NY; Jun. 2015. p. 313–9. doi:https://doi.org/10.1109/MobServ.2015.51.
[2] De Graauw A, Van Schie M. Concurrent multiband transceiver; 2016. US Patent 9,329,259.
[3] Multi-channel Choong L. IEEE 802.15. 4 packet capture using software defined radio. UCLA Netw Embed Sens Lab 2009;3:1–20.
[4] Se Yoo, Chong PK, Bae J, Kim TS, Kim H, Yoo J. Multi-channel packet-analysis system based on IEEE 802.15. 4 packet-capturing modules. Int J Distrib Sens Netw 2014;10:216504. https://doi.org/10.1155/2014/216504.
[5] XILINX. A user guide for 7 Series FPGAs Configurable Logic Block; 2016. https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf [Accessed 19 February 2020].
[6] Aslam M, Jiao X, Liu W, Moerman I. An approach to achieve zero turnaround time in TDD operation on SDR front-end. IEEE Access 2018;6:75461–70. https://doi.org/10.1109/ACCESS.2018.2883253.
[7] Kazaz T, Jiao X, Kulin M, Moerman I. Demo: WiSCoP: wireless sensor communication prototyping platform. In: The International Conference on Embedded Wireless Systems and Networks, Uppsala, Sweden; Feb. 2017. p. 1–2. doi:http://hdl.handle.net/1854/LU-8508242.
[8] Ettus Research. The Universal Software Radio Peripheral (USRP); 2020. https://www.ettus.com/ [accessed 19 February 2020].
[9] GNU Radio. The free and open-source software development toolkit; 2020. https://www.gnuradio.org/ [accessed 19 February 2020].
[10] He Y, Fang J, Zhang J, Shen H, Tan K, Zhang Y. MPAP: virtualization architecture for heterogenous wireless APs. ACM SIGCOMM Comput Commun Rev 2011;41:133–4. https://doi.org/10.1145/1851275.1851271.
[11] Microsoft. Microsoft Research Software Radio (Sora); 2008. https://www.microsoft.com/en-us/research/project/microsoft-research-software-radio-sora/ [accessed 19 February 2020].
[12] Jiao X, Moerman I, Liu W, de Figueiredo FAP. Radio hardware virtualization for coping with dynamic heterogeneous wireless environments. In: International conference on cognitive radio oriented wireless networks, Lisbon, Portugal; Sep. 2017. p. 287–97. doi:https://doi.org/10.1007/978-3-319-76207-4_24.
[13] Ferrari P, Flammini A, Marioli D, Rinaldi S, Sisinni E. On the implementation and performance assessment of a wirelessHART distributed packet analyzer. IEEE Trans Instrum Meas 2010;59:1342–52. https://doi.org/10.1109/TIM.2010.2040907.
[14] Reddy P, Sharma H, Paulraj D. Multi channel Wi-Fi sniffer. In: 2008 4th international conference on wireless communications, networking and mobile computing, Dalian, China; Oct. 2008. p. 1–6. doi:https://doi.org/10.1109/WiCom.2008.727.
[15] IEEE Standard for Low-Rate Wireless Networks, IEEE Standard 802.15.4- 2015; 2015. https://standards.ieee.org/project/802_15_4.html [accessed 19 February 2020].
[16] Xilinx. ZedBoard, A development board for Zynq-7000 SoC; 2020. https://www.xilinx.com/products/boards-and-kits/1-8dyf-11.html [accessed 19 February 2020].
[17] Analog Devices. User Guide of AD-FMCOMMS2-EBZ an FMC Board for the AD9361; 2020. https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz [accessed 19 February 2020].
[18] Xilinx. An overview of Zynq-7000 SoC Data Sheet; 2020. https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf [accessed 19 February 2020].