GRANT AGREEMENT NO.: 732174

Call: H2020-ICT-2016-2017
Topic: ICT-13-2016
Type of action: RIA

Orchestration and Reconfiguration Control Architecture

D5.1: First operational SDR platforms with advanced Reconfigurability/reprogrammability capabilities

Revision: v.1.0

Work package	WP5
Task	Task 5.1, 5.2
Due date	31/12/2018
Submission date	21.12.2018
Deliverable lead	KUL
Version	1.0

Authors	Sofie Pollin (KUL), Seyed Ali Hassani (KUL), Wei Liu (IMEC), Peter Ruckebusch (IMEC), Ingrid Moerman (IMEC), Prasanthi Maddala (Rutgers)
Reviewers	Roberto (TUD), Walter Nitzold (NI)
Abstract	This deliverable includes the progress and implementation results obtained in Year 2 on the available SDR platforms with advanced reconfigurability and reprogrammability capabilities. Each research group describes the reconfiguration they introduced and integrated into the ORCA SDRs. This deliverable will also explain a plan for functionalities to be implemented in the third year.
Keywords	Reconfiguration and reprogramming, SDR, control plane

** R: Document, report (excluding the periodic and final reports)*

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	R*	
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CI	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to ORCA project and Commission Services	

OTHER: Software, technical diagram, etc

EXECUTIVE SUMMARY

The main aim of the ORCA project is to accelerate flexible end-to-end networking experimentation. Using the ORCA facilities, network designers are given a wide variety of physical layer and control plane solutions to develop their innovative ideas. To this end, ORCA offers open source and modular hardware and software components in the form of real-time software defined radio (SDR) platforms which enable advanced reconfiguration and reprogramming. These capabilities reduce design time and planning time of experiments via runtime updating of the hardware and software code of RF, transceivers chains, network protocols, microcontroller firmware, and FPGA-based functional blocks. To obtain run-time reconfiguration, ORCA does not sacrifice the energy and latency performance which are crucial in real-life networking applications.

This deliverable focuses on the solutions developed by ORCA partners during Year 2 which allow SDRs with advanced run-time reconfiguration. We subdivide these capabilities into the categories of modular design time reconfigurability and run-time live programmability. The former allows design time configuration of pre-implemented modules which deliver various functionalities, e.g. collision detection and clear channel assessment, whereas the latter enables on-the-fly programming and modification. In the next sections, ORCA partners describe their Year 2 achievements and what they have planned for Year 3 to provide such an advanced capability for SDRs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AnSIC	Analog Self-Interference Cancelation
AGC	Automatic Gain Control
AP	Access Point
AXI	Advanced eXtensible Interface
BPSK	Binary Phase Shift Keying
BBU	Base Band Unit
CNN	Convolutional Neural Network
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DDR	Double Data Rate
DFT	Discrete Fourier Transform
DiSIC	Digital Self-interference Cancelation
DMA	Direct Memory Access
DSA	Dynamic Spectrum Access
DSP	Digital Signal Processing
GUI	Graphical User Interface
eNB	eNodeB
ELF	Executable and Linkable Format
FD	Full-Duplex
FO	Frequency Offset
GFDM	Generalized Frequency Division Multiplexing
HALO	Hardware in the loop
HLS	High Level Synthesis
IBFD	In-Band Full-Duplex
IP	Internet Protocol
KVM	Kernel Virtual Machine
LTE	Long Term Evolution
LBT	Listen-Before-Talk
MAC	Medium Access Control
ML	Machine Learning
OCM	On-Chip Memory
PHY	Physical Layer
PL	Programmable Logic
PS	Programmable System
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation

RAM	Random Access Memory
RFNoC	RF Network on a Chip
ROM	Read Only Memory
RTT	Round Trip Time
SDR	Software Defined Radio
SRH	Shared Radio Head
TCP	Transmission Control Protocol
TTI	Transmission Time Interval
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
UD	User Device

1 INTRODUCTION

The recent growth of mobile wireless devices such as smartphones and smart personal assistances is thoroughly grateful to the flexibility of wireless communication solutions. For example, the next fifth generation of communication solutions (5G) and the Internet-of-Things will require a radical rethinking of the wireless communication landscape to keep enhancing the spectral and energy efficiency at acceptable cost [1]. The only way to meet extreme performance requirements in a single network, is allowing extreme reconfiguration, which means that we can configure a given network for ultra-low power operation, or alternatively, for ultra-high reliability, simply by reconfiguring the network.

ORCA explores the potential of such extreme reconfiguration using a **modular design time approach**, which means that at design time we take a modular approach to the design of the radio stack and allow certain functionality on the fly. For instance, for full duplex, it is known that it is possible to boost spectral efficiency of a network, but only when the traffic flows are symmetric. Also, full duplex collision detection makes it possible to decrease the energy cost per bit of a network, but only when the number of nodes in the network is large enough [2]. But taking a modular full duplex design approach, we can only enable the full duplex functions when really needed to optimize the network performance.

At the same time, there is a lot of wireless hardware already installed in the field, and we need to come up with solutions and methods that allow to upgrade those devices in the field. Existing hardware can sometimes be reused for novel networking concepts, if we could do a **live reprogramming** of devices in the field. One example that we will explore is the upgrade of a full duplex SDR facility with radar capabilities. We will show that a full duplex hardware can be used as a mono-static Doppler radio by only adding some extra functional blocks. This will illustrate the benefits of live reprogramming as a method to add diverse and unexplored wireless functionality to existing SDR hardware.

The main goal of this deliverable is the illustration of three ORCA modular design approaches, and how these can be used to allow advanced run-time reconfiguration of an SDR network. In addition, we give examples or where live reprogramming can be used to enrich the SDR functionality at run-time. This is to give an overview of the ongoing work of Tasks 5.1 and 5.2 in WP5.

1.1 Organization of the Deliverable

The remaining sections of the document are organized as follows:

- Section 2 describes two examples of **ORCA modular design approaches**, enabling the reconfiguration of an SDR network at run-time to meet diverse performance requirements.
  - The first example is the RFNoC modular design approach. Different NoC blocks can be assembled by users to make customized FPGA image on the 3rd generation of USRP. Customizing NoC block itself is also supported.
  - The second example is a modular Full Duplex/Half Duplex SDR design, that has a fully flexible PHY and MAC that can be tuned to enable half duplex, full duplex, or full duplex collision detection. This framework is developed in the LabVIEW communications framework.
- Section 3 explores scenarios and approaches for **Live Reprogramming**. Again, two approaches and scenarios are explored:
  - A first scenario focuses on upgrading a full duplex SDR towards a Doppler radar. First the feasibility of this upgrade is illustrated, and a performance analysis is done of a joint full duplex and radar operation. Then, an overview is given of the functional block diagram, including new functional blocks that should be added to enable this Doppler radar operation, while simultaneously maintaining the full duplex radio operation.
  - A second scenario focuses on the partial reconfiguration of FPGA on SDR device. This

is being developed in an ORCA open call project, which allows partial bitstream implementing a dedicated function (eg, DSP filters or modulating blocks) to be replaced at runtime of an SDR radio. It is also foreseen to support live firmware update, which goes hand in hand with the bitstream configuration. And eventually live programming over the air is targeted in the next year's implementation.

- Section 4 concludes this deliverable and lists some steps for Year 3 of the ORCA project.

2 MODULAR DESIGN FOR ADVANCED RECONFIGURATION

2.1 Modular design with RFNoC

RFNoC (RF Network on Chip) is a unique processing and routing architecture developed by Ettus Research, for third generation USRP devices [3]. This framework adopts a modular approach and significantly reduces FPGA development time.

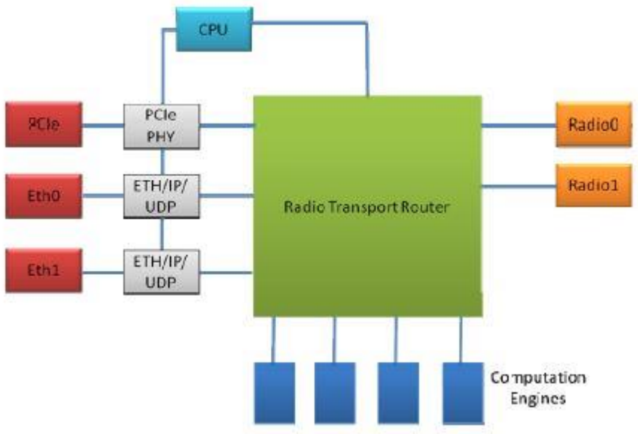


Figure 1: RFNoC Architecture

As shown in Figure 1, various FPGA modules are connected to radio transport router, which acts as a programmable cross switch for routing data/control packets between these modules. Custom user logic can be wrapped into computation engines, and integrated with the design.

2.1.1 Testbed Integration and Support

- Devices – ORBIT grid has 36 USRP X310s deployed. 4 X310s are connected to ceiling nodes (18-1, 18-20, 3-1, 3-20) via PCIe. There are 4 Massive MIMO racks with 8 X310s each. These X310s can be accessed from any node on the grid. Node image with UHD (with RFNoC support) and tools to program the FPGA is provided.
- Computation Engines – following custom modules were built with publicly accessible examples and source code
    - Spectrum sensing – performs FFT on the incoming samples and averages the FFT magnitude over time.
    - Spreader – generates a DSSS signal by multiplying incoming data symbols with locally generated PN sequence.
    - Correlator – correlates receive samples with a real PN sequence.
    - Averaging – computes average of N incoming vectors. This module can be used in conjunction with a different CE such as the correlator.
- Tutorials – tutorial showing how to use a spectrum sensing CE to monitor RF spectrum from 4 USRP X310s in ORBIT grid is provided at http://www.orbit-lab.org/wiki/Tutorials/k0SDR/Tutorial24#RFNoCSpectrumSensing

2.1.2 Advanced reconfiguration

The connections between blocks in RFNoC framework are “programmed” by configuring the Radio Transport Router. This enables runtime configuration of the RFNoC flowgraph or transceiver chain. After building and loading an FPGA image with all the required modules, the RFNoC flowgraph can be built, cleared and rebuilt at run time, from the host, by using RFNoC UHD methods.

This task was also proposed as an extension in Open Call 2: Dynamic runtime composition of mmWave transceiver chains. The aim of this extension is to provide a framework for runtime composition of transceiver chains on widely used SDR platforms, with focus on mmWave systems.

2.1.3 Relation to showcase

By pushing computationally intensive tasks into FPGA, modular design with RFNoC definitely allows for low latency systems. One such example is the channel sounding experiment conducted in ORBIT [4], where power delay profiles for 16 100MHz channels were observed real time, by using RFNoC correlator modules in USRP X310s. Wideband signal processing can be a relevant showcase as well.

2.1.4 Risk analysis

- FPGA space can be seen as a limitation, as all the required modules have to be pre-built into the FPGA image.
- Runtime reconfigurable systems must be carefully designed so that the reconfiguration time (to clear and build a new RFNoC flowgraph) still lies within the system's latency requirements.

2.1.5 Future work

New devices with RFNoC support, such as USRP N310 will be deployed in the future. Future plans also include development of powerful computation engines for massive MIMO signal processing, channel estimation etc..

2.2 Modular design for full duplex radios [KUL]

The KUL (In Band Full Duplex) IBFD platform is based on the modular and parameterized cross-layer adaptable wireless system (CLAWS) [1] architecture, that is extended with a modular MAC protocol for full and half duplex operation and collision detection. Deploying MicroBlaze softcores in this platform enables CPU-based modular design of the MAC protocol where various functionalities can be implemented in the form of C++ objects and functions. Since the softcore is run-time programmable, one can instantiate these modules without interrupting other hardware-realized components on the FPGA.

In addition to the CPU-based modules, there are several already implemented components running on FPGA itself. These elements are developed by LabVIEW FPGA and can be configured and composed in run-time to build up a custom scheme.

Furthermore, the system is controlled by a host PC which not only can accomplish the upper layers tasks but also provides debug and development tools to facilitate network level experimentation.

Figure 2 illustrates the modular architecture of the IBFD SDR platform. Deployed on a NI USRP this architecture can deliver fully flexible MAC and PHY layers to establish a half/full duplex communication node. The FPGA-based elements in this figure are depicted by green color while the blue components represent the CPU-based ones.

As shown in this figure, the device is equipped with an electrical balance duplexer (EBD) to provide the essential analog self-interference cancelation (AnSIC). One of the MicroBlazes hence is dedicated to tune the EBD dynamically to maintain Tx-Rx isolation.

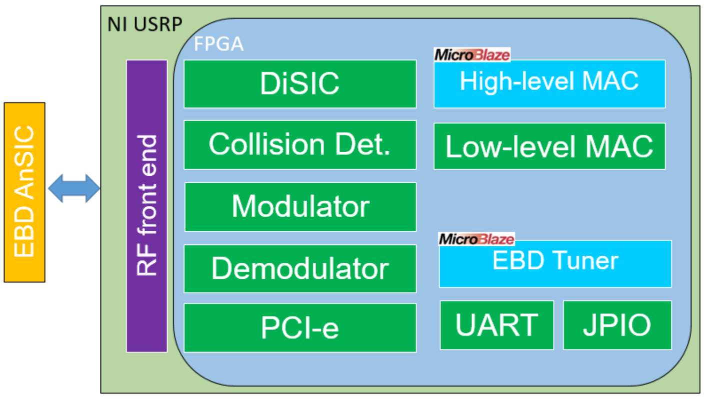


Figure 2. Modular architecture of the KUL IBFD platform.

The modularity in this platform enables various applications. For instance, the collision detector module can be used in CSMA/CD [5] protocol, or the DiSIC and the EBD tuner are needed to perform a full-duplex radio link. Similarly, the software modules can deliver various functionality to satisfy different applications. For example, the whole MAC protocol can be established by C++ functions on the MicroBlaze or partially on hardware to improve MAC's real-time operation. This way the experimenters can trade-off between flexibility and latency performance via choosing appropriate blocks form hardware and software realized modules. The PHY-MAC architecture is more described in D4.3.

2.2.1 PHY layer modular design

The PHY layer in this design comprises four main elements:

- Tunable Modulator
- Parametrizable Demodulator
- Digital self-interference cancellation (DiSIC) module that is only needed in a full duplex operation mode
- Collision detector that is only needed when doing in-band full duplex collision detection

To perform real-time operation, these components are implemented by LabVIEW FPGA. Depending on the scenario, one can configure either of the PHY modules to achieve half/full duplex with/without collision detection functionalities.

2.2.2 MAC layer modular design

To achieve maximum level of reliability and robustness, the MAC layer is implemented in a bi-layer structure. While the high-level MAC is running on the MicroBlaze to accomplish CPU-friendly tasks, the low-level MAC performs somewhat non-sequential straightforward jobs of the MAC protocol and reacts to interrupts instantly. Table 1 lists the implemented hardware modules and C++ functions in both of the MAC layers. As listed in this table, hardware and software modules are implemented with the same functionality to enable experimentation of various MAC realizations. For example, to achieve a low-latency operation one can mostly allocate the hardware modules. Whereas, in scenarios where a deep control level and intensive log recording is needed, the C++ implemented functions can offer a better performance.

Table 1. List of FPGA-based and CPU-based modules in the bi-layer MAC.

	Module	Realization

1	Clear channel assessment (CCA)	Low-level MAC (LabVIEW FPGA)
2	PHY packet parser	
3	PHY packet generator	
3	Imm-ACK packet generator	
4	Frame check sequence generator	
5	Frame check sequence parser	
6	Interframe time watch	
7	PHY packet parser	C++ functions
8	PHY packet generator	
9	Backoff time checker	
10	Frame check sequence generator	
11	Frame check sequence parser	
12	CSMA MAC	
13	CMSMA with collision detection MAC	
14	MAC packet generator	
15	PHY reconfigure	
16	Low-level MAC reconfigure	
17	Particle swarm EBD tuner	
18	Dithered linear search EBD tuner	

2.2.3 Advanced reconfiguration

As explained above, providing several FPGA-based and CPU-realized modules enables ORCA users to experiment their custom networks with emphasis on one or multiple specific features. The hardware-level reconfiguration can be established in FPGA design time when different hardware modules are set together to form a multi-mode SDR. Employing a routing switch, for instance, one can used DiSIC and Collision detector to integrate full duplex functionality and half-duplex with collision detection in one SDR and enable them once they are needed.

The software modules also can be added to the MicroBlaze program. Generating a new BIN file, the experimenter can apply a new functionality by loading the file on the MicroBlaze in run time and without interrupting the hardware realized modules.

2.2.4 Relation to showcase

This capability can be used in SC2 and SC3 where ORCA targets low-latency and/or high-throughput industrial communication. For instance, we can demonstrate how the modular architecture of the IBFD SDR enables run-time reconfigurability to establish a half duplex or full duplex network, where a central processing unit controls multiple robots.

2.2.5 Testbed integration

The implemented modules in Y2 are already integrated to the testbed to enable ORCA experimenter employ advance reconfigurable IBFD SDR offered by ORCA.

2.2.6 Risk analysis

- Optimum interfacing between the modules is a potential challenge. The interfacing should not affect the latency and energy performance negatively.
- The use of multiple hardware pre-implemented modules is limited to the FPAG resources. Besides, the resource consumption can be compromised by the performance via reducing the precision of computation. In this regard, the collision detector and the DiSIC modules are two potential blocks which can be optimized.

2.2.7 Future work

Future work is planned to test and improve the run-time advanced configuration capability. For ORCA Y3, we are also planning over-the-air reprogramming. This way one can reprogram MicroBlaze of several devices by wireless broadcasting a new firmware.

3 LIVE REPROGRAMMING

3.1 Live Reprogramming for full duplex radios

3.1.1 Motivation for environmental sensing by communication devices

Wireless communication is used in almost every electronic device today and offers a rich functionality when communication is combined with device-level sensors. A smartphone, for example, offers several appealing user experiences as it typically is equipped with a number of integrated sensors, such as a high-quality camera, barometer, three-axis gyro, accelerometer or GPS. Recently, researchers have made progress in developing opportunistic and passive remote radio frequency (RF) sensing techniques which enable remote sensing of the environment without adding extra hardware to the device that is often already equipped with a radio. Employing the ambient electromagnetic signals in the form of passive Doppler radar for gesture detection and localization is a remarkable example in this domain.

A passive Doppler radar relies on an existing electromagnetic signal arriving from two paths: the reference and the surveillance channels. The reference signal consists ideally of the line-of-sight signal and the surveillance signal consists of the reflections from static and moving objects in the vicinity. Correlating the reference and surveillance signals captured by two/multiple receivers gives the Doppler frequency shift created by the moving object.

As shown in Figure 2, if one can integrate the surveillance receiver and the radar illuminator in one platform, no extra device is needed to collect the reference signal as it can be provided by the transmitter directly. Thus, the impact of the reference channel such as the additive channel noise, undesirable hardware features and lack of channel isolation, would not degrade the radar performance. Furthermore, since both parties are in a single framework, explicit and computationally light synchronization is achievable to mitigate technical difficulties such as long silence intervals. A radar, integrated into a single communication device is also more interesting from a cost perspective, as only one set of hardware is required to enable estimation of the Doppler information.

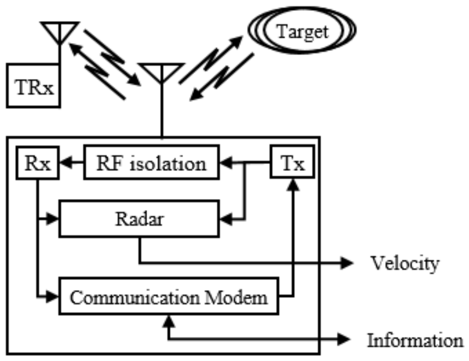


Figure 3. The communication device can be reprogrammed to support radar-like functionality.

3.1.2 Feasibility analysis

In this deliverable, we investigate the feasibility of using the in-band full-duplex (IBFD) technology to combine a Doppler radar and a communication radio in the form of a single transceiver device (e.g., a Wi-Fi AP). Figure 4 shows the overview of the proposed system which brings radar-like functionality into an IBFD device merely by employing its already-existing hardware.

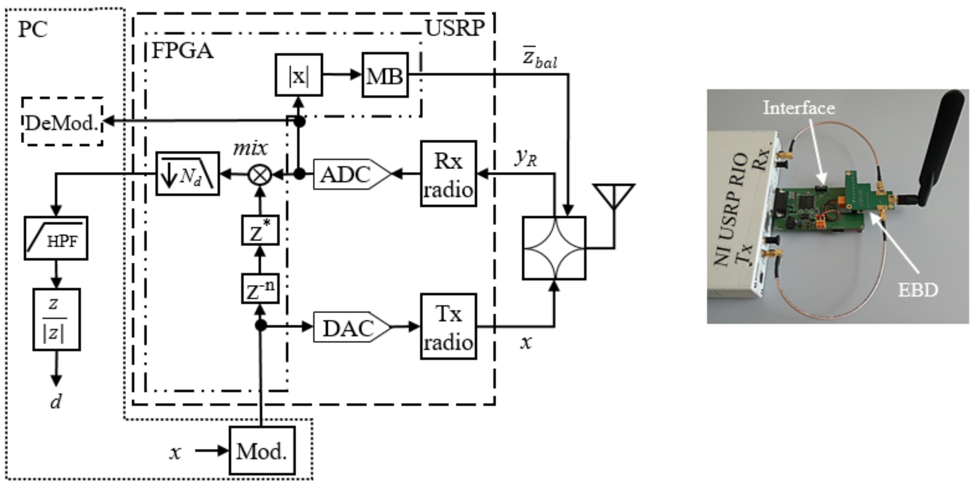


Figure 4. Prototyped IBFD communication device, which also functions as a radar.

The prototyped model consists of one SDR, one EBD-based analog self-interference canceller and one PC. The SDR is a NI USRP RIO with a Kintex-7 FPGA onboard. The whole setup can be controlled/monitored via a PCI-e connection and a LabVIEW user interface. The test is carried out using an IEEE 802.11p OFDM waveform. We also employed an omni-directional antenna as such the device can radiate/receive to/from all directions and maintain its communication functionality. The USRP produces 120 MHz complex baseband signal which has to be mixed with the transmitted signal. The resulting signal is then decimated to achieve a narrow-band (457 Hz) complex signal. The real-time processing on the FPGA is in 16-bit and 32-bit fixed-point precision and, as shown in the figure, the rest of the computation is accomplished on a computer to obtain the micro-Doppler profile which enables detecting up to 9.64 m/s motions with 10.6 mm/s resolution. To evaluate the radar capability of the prototyped device, a remotely/automatically controlled XY table is utilized to move a cone reflector with radar cross section (RCS) 15 dBsm at different speeds ranging from 100 to 800 mm/s, as shown in Figure 5.

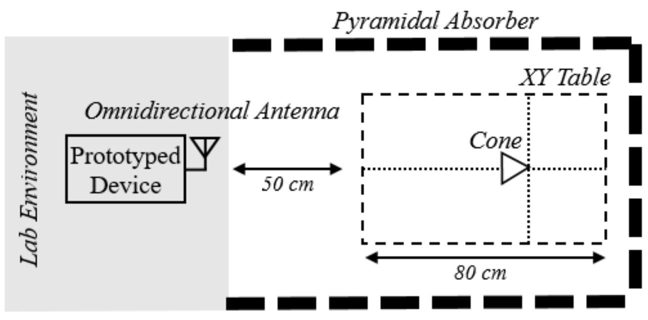


Figure 5. Test setup to evaluate radar performance.

Figure 6 shows the Doppler profile obtained by short-time Fourier transform (STFT), when the reflector moves forward and backward at different velocities. There is a visible artifact in the spectrogram when the XY table changes the direction of the target. It is due to inertia in the mechanical setup.

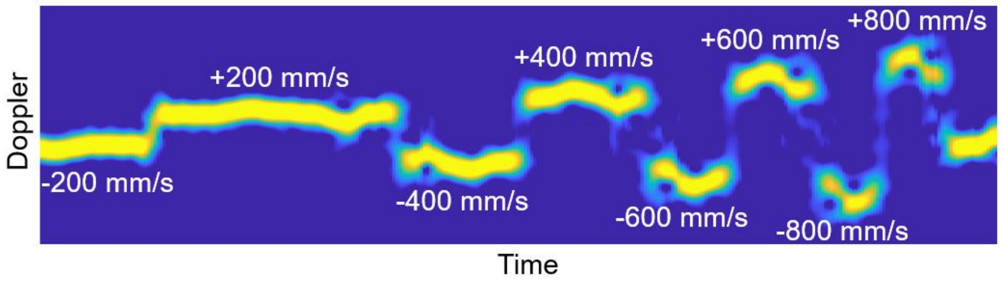


Figure 6. Estimated Doppler information for a moving reflector.

This test proves that IBFD technology can also offer practical solution for simultaneous communication and environmental sensing.

3.1.3 Functionality upgrade

Doppler detection in the form of a run-time programmable module is the main functionality that has to be integrated. As shown in Figure 4, an IQ mixer, a complex signal decimator and a high-pass filter are needed to extract the Doppler signal. In the next step, the estimated Doppler signal can be analysed by fast Fourier transform to obtain the velocity pattern of the dynamics in the environment.

3.1.4 Relation to showcase

This functionality can be employed in SC3 where low-latency and high-throughput communication is needed. Sensing the Doppler signal helps to recognize the environment and tune the communication device accordingly. The EBD tuning rate, for instance, is highly environmental dependent. A context-aware EBD tuning technique hence can improve the performance of the full-duplex system, especially in an industrial scenario where various environmental dynamics exist.

3.1.5 Testbed integration

In ORCA Y3, the KUL IBFD testbed can perform Doppler estimation via live-programming.

3.1.6 Risk analysis

The functional blocks have to be realized in the form of software functions in a way that can be programmed on the host or a softcore on run-time. Efficient development of run-time reprogrammable software modules is a challenging task.

3.1.7 Future work

Future work is needed to enable optimal realization of such a context-aware system in a way that can be reprogrammed on several communication nodes within a network. Besides, the radar-like functionality can be used to recognize the type of the environment, by a neural network classifier for example, and employ the environmental information to retune the PHY and MAC layers aiming maximum performance.

3.2 Live Reprogramming using partial reconfiguration of FPGA on ZYNQ SDR and USRP X310

3.2.1 Motivation

This task is defined as a topic for Open Call 2 for extension. It aims to extend the ORCA facility with capabilities to partially reprogram the FPGA of an SDR platform during runtime. A typical wireless transceiver chain consists of multiple processing units that are performing signal processing functions such as filtering, digital signal up or down conversion, channel estimation, symbol (de)mapping, (de)coding or Fast Fourier Transformation (FFT). Today, deploying a new functionality for such a processing unit requires the generation of a new bitstream that can then be flashed on the FPGA chip, which is a time consuming process and also leads to a long disruption of device operation. This extension seeks a solution to achieve partial reconfiguration of FPGA on SDR during runtime, for the following motivations: (i) to reduce or eliminate the SDR device's downtime during FPGA configuration; (ii) to minimize the compilation delay when generating the bitstream for a complex transceiver chain taking up an entire FPGA; (iii) to achieve remote reconfiguration by distributing a partial bitstream over a backbone network or eventually over the air via framework such as GITAR [7]

3.2.2 Feasibility analysis

As the extension is still ongoing, the feasibility analysis is also a work in progress. The 3rd party Open Call Partner working on this extension is CTTC [6]. Based on the current status, partial reconfiguration for live programming is feasible for ZYNQ SDR. For USRP X310 there are more difficulties, this is partially because of the RFNoC framework used for USRP X310 is a powerful but complicated open source project on itself, some learning curve is required to use it to generate customized FPGA image on the USRP. The RFNoC framework already has taken quite some FPGA resources on the USRP, the partial reconfiguration also requires some management IP cores and possibly a dedicated microcontroller included in the FPGA design, hence there is a risk that not much FPGA resources will be left for meaningful functionalities to be implemented in the partial bitstream. However, this requires further investigation and confirmation.

3.2.3 Functionality upgrade

An overview of the functionality that is being developed in this extension is summarized in the figure below.

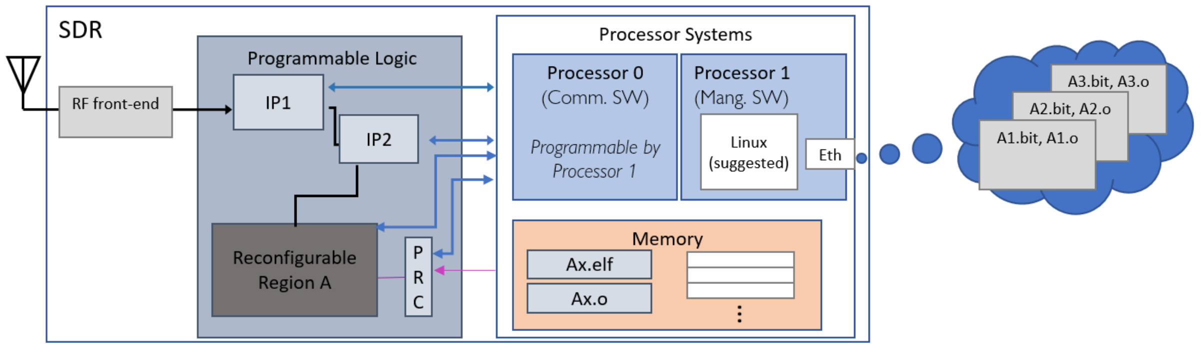
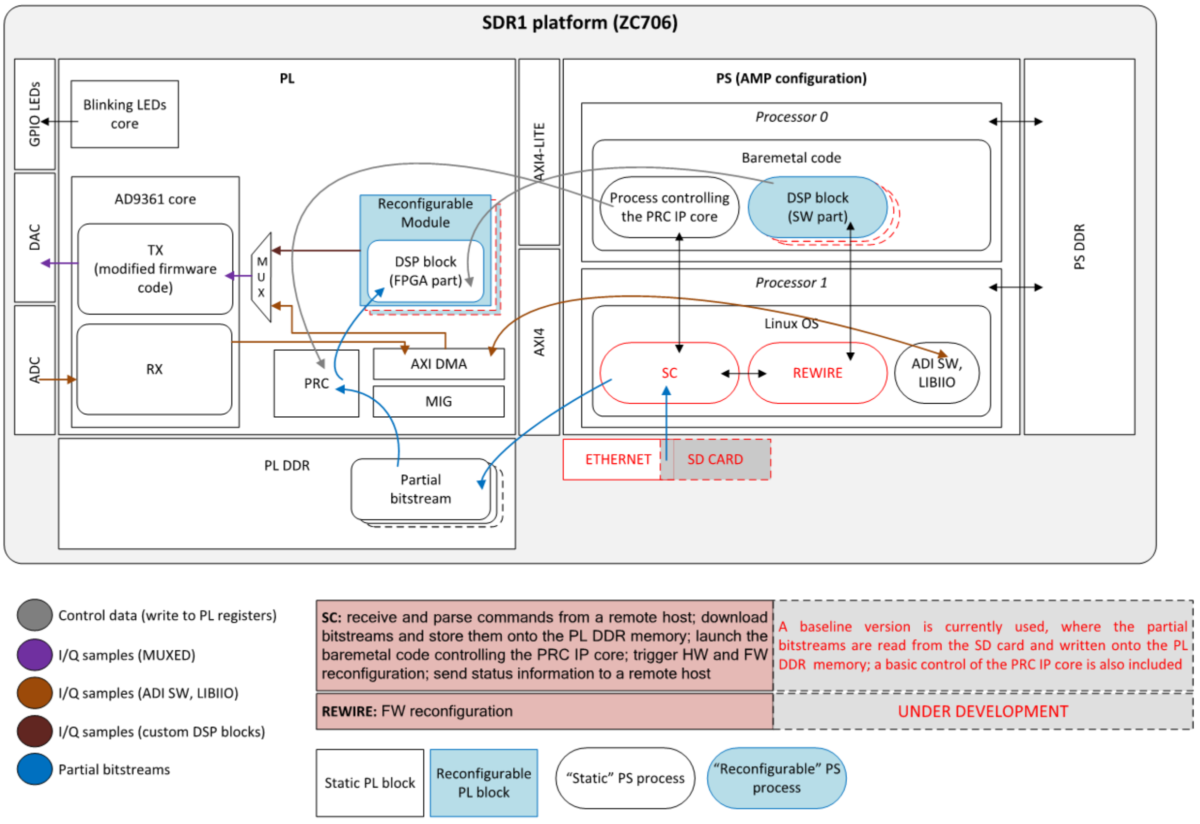


Figure 7 Functional overview of partial FPGA reprogramming on SDR

We foresee two embedded processors present on an SDR device. Software running on processor 0 is responsible for the communication functionalities, such as configuring several IP cores in the signal processing chain or transporting data in between IP cores; processor 1 is responsible for management functionalities, such as fetching partial bitstreams from the cloud, loading the bitstream to the dedicated reconfigurable region on FPGA, and configuring the software running on processor 0 to use the new hardware. Firmware configuration needs to go hand-in-hand with the hardware update, because the usage of new hardware usually requires corresponding updates in the drivers. Ideally, the new firmware should be fetched as an object file from the cloud together with the partial bitstream file, processor 1 stores the object file into appropriate memory region of processor 0 to make the new functions available, without interrupting the normal operation of processor 0. The functional block that can be loaded into the reconfigurable region should be a Baseband Processing Unit (BPU), such as the digital up/down converter, or a carrier sensing module.

Following these functional requirements of the open call, CTTC has made an initial design on the ZYNQ SDR. This design is adapted from the reference design provided by Analog Device. First the partial bitstream is stored in an SD card. Then it is loaded into the DDR memory that is directly connected to the Programmable Logic (PL), which is then accessed by the Xilinx Partial Reconfiguration Controller (PRC) in order to fetch the partial bitstream and use it to program a predefined reconfigurable region on the FPGA device. The static part of the FPGA contains a GPIO IP core to blink the LED. During the partial configuration, the LED does not stop blinking, which means the static part of the FPGA functionality is not affected. The partial bitstream contains a DSP block that generates a sine wave signal. The frequency of the signal can be configured by software. The future work in this design is also highlighted in the graph, among which one of the main updates will be about obtaining the partial bitstream over a backbone network (Ethernet interface), rather than from the SD card.



SC: receive and parse commands from a remote host; download bitstreams and store them onto the PL DDR memory; launch the baremetal code controlling the PRC IP core; trigger HW and FW reconfiguration; send status information to a remote host

REWIRE: FW reconfiguration

A baseline version is currently used, where the partial bitstreams are read from the SD card and written onto the PL DDR memory; a basic control of the PRC IP core is also included

Figure 8 Overview of the architecture for partial FPGA reconfiguration on ZYNQ SDR

3.2.4 Relation to showcase

This task is still ongoing, it is in the planning of ORCA Y3 to integrate partial reconfiguration for live programming of SDR devices into show case demos. At this moment we believe it is most closely related to showcase 3 and showcase 4, where diverse traffic types or multiple RATs are considered.

3.2.5 Testbed integration

As this task is ongoing, there is no testbed integration activity to be reported at this moment.

3.2.6 Risk analysis

As explained in the feasibility analysis, the risk for this task is can be subject to the amount of FPGA resources on SDR platform. For X310 USRP, there is a fear that the leftover resources on FPGA may not suffice for a complicated DSP function to be implemented in the partial bitstream.

3.2.7 Future work

In order to enable advanced configuration such as over-the-air software updates on ZYNQ SDR platform, the previous work, that enables partial reprogramming of embedded devices [**7**], is foreseen to be integrated on the ZYNQ SDR. Luckily, the Zynq 7000 SoC includes an ARM-A9 CPU, which should allow integrating the reprogramming capabilities provided by the GITAR framework. On the other hand, the specificities of the ZYNQ SDR (i.e., Zedboard or Xilinx zc706 board), especially in terms of memory, make the integration effort a non-straightforward task.

There are two general approaches to enable reprogramming embedded devices: (1) Firmware based approaches replace the entire firmware of the embedded devices; and (2) Modular approaches allow replacing specific parts of the firmware (e.g. single software modules or packages). While the former is easier to implement, the latter reduces the overhead in terms of energy and latency. In the remainder of this section, the modular approach will be discussed. Within the modular approaches, there are two options based on where and when the new or updated modules are linked to the already present firmware (as depicted in Figure 9):

a) Runtime linking, requires a linker on the constrained device to install or update software modules in an active system. The linker relocates code (data) section to the allocated ROM (RAM) memory regions and resolves undefined references to already present modules.

b) Offline linking, offloads the task of the dynamic linker to a more powerful server. An offline linker relocates and resolves undefined references before the modules are disseminated to the devices. This approach requires complete knowledge of the code and data memory location of each module installed on each device.

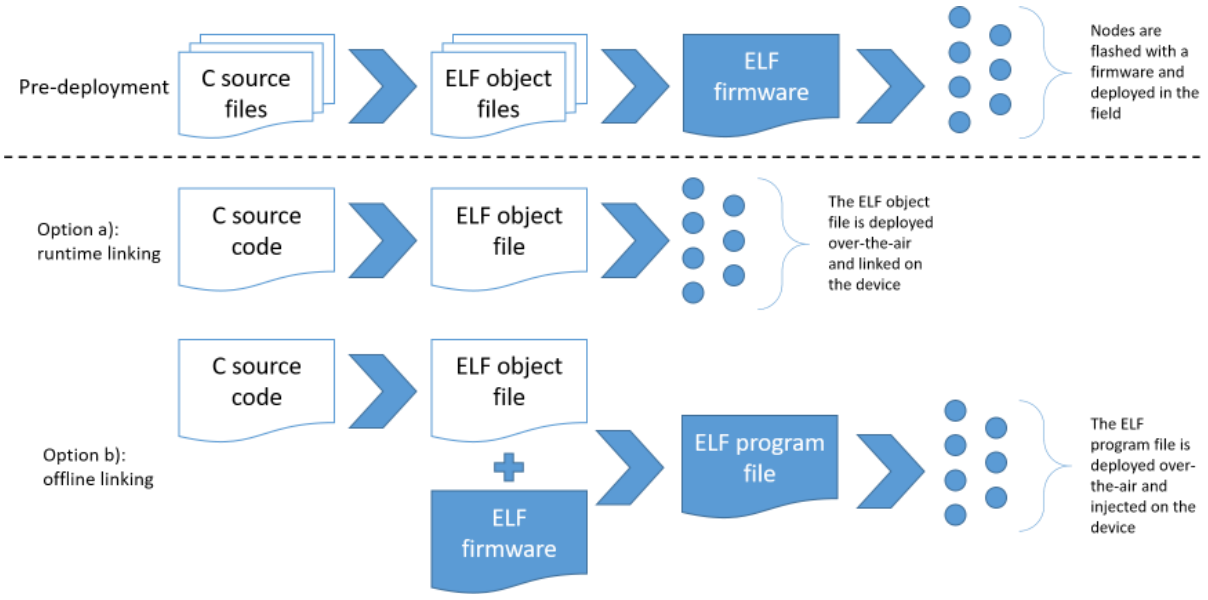

Pre-deployment
C source files
ELF object files
ELF firmware
Nodes are flashed with a firmware and deployed in the field
Option a): runtime linking
C source code
ELF object file
The ELF object file is deployed over-the-air and linked on the device
Option b): offline linking
C source code
ELF object file
ELF firmware
ELF program file
The ELF program file is deployed over-the-air and injected on the device

Figure 9 Over-the-air software updates using a run-time vs. offline linker

Because the Zynq 7000 SoC uses a different type of memory (i.e. DDR RAM) to execute program logic, different type of CPU instructions are required to execute control logic. Therefore, the existing ELF (Executable and Linkable Format) runtime linkers cannot be used directly and should be altered. The modifications required to use an offline linker require less implementation effort. Now only the linker script needs to be altered. Given the benefits of a pre-linker (i.e. less bytes need to be transferred to the devices) combined with the lower implementation effort, this approach was chosen. The different memory type (RAM instead of ROM) also has implications on dynamic memory allocation. The existing component that allows allocating and writing ROM sections should be replaced.

Moreover, the reprogramming capabilities must also be exposed to a network controller running on a remote machine. For this purpose, the ZYNQ SDR should also be integrated in the WiSHFUL framework [8]. This will allow reusing the over-the-air software dissemination protocol designed and implemented in WiSHFUL. The protocol, depicted in Figure 10, includes a robust mechanism to allow transactions, making sure that all bytes are correctly received. Moreover, the algorithm also reduces the control bytes that need to be transferred by adopting a block-wise acknowledgement and multicast data transfers.

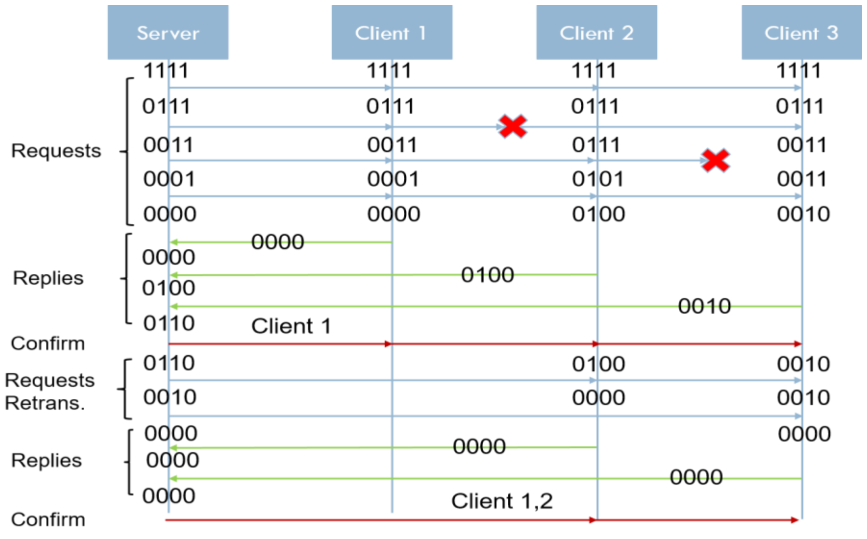


Figure 10 Robust transaction algorithm

4 CONCLUSIONS

ORCA aims to facilitate end-to-end networking by improving the state-of-the-art SDR technology in a way that it provides a high level of flexibility in various network layers. ORCA tackles this challenge by developing strategies for runtime composition of transceiver chains, allowing the instantiation of pre-implemented FPGA-based and CPU-based functional blocks, without the need for a lengthy procedure of regenerating FPGA code. This approach accelerates network development as it enables immediate validation of custom schemes.

This document described the main contributions made to provide advanced reconfigurability and reprogrammability for ORCA SDR solutions during Year 2. Partners provided an overview of the architecture and functionalities of their SDR implementations, allowing advanced reconfiguration. The implemented and targeted functionalities include:

- Rutgers – modular radio design on USRP with RFNoC framework
- KUL – modular MAC-PHY architecture with advanced design time reconfiguration and run-time reprogrammable radar-like functionality
- IMEC – partial reconfiguration of FPGA on ZYNQ SDR and USRP X310, and the approach to integrate of partial bitstream and firmware update with the live reconfiguration functionality in GITAR framework.

Additionally, a brief description was given on how the implementations above were integrated into the second year ORCA showcases and in the partners' testbeds. Lastly, each partner outlined their plans for the development of ORCA functionalities in the third year.

5 REFERENCES

[1] Van den Bergh, Bertold, Tom Vermeulen, Marian Verhelst, and Sofie Pollin. "CLAWS: Cross-Layer Adaptable Wireless System enabling full cross-layer experimentation on real-time software-defined 802.15. 4." EURASIP Journal on Wireless Communications and Networking 2014, no. 1 (2014): 187.

[2] Vermeulen, Tom, Fernando Rosas, Marian Verhelst, and Sofie Pollin. "Performance analysis of in-band full duplex collision and interference detection in dense networks." In Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual, pp. 595-601. IEEE, 2016.

[3] J Malsbury, M Ettus, Simplifying FPGA Design with A Novel Network-on-Chip Architecture, in Proc. Of ACM SIGCOMM'13, 2013

[4] Bhargav Gokalgandhi, Prasanthi Maddala, Ivan Seskar, "Implementation of FPGA based Channel Sounder for Large scale antenna systems using RFNoc on USRP Platform". Proceedings of the 1 st GNU Radio Conference, 2017

[5] Vermeulen, Tom, Fernando Rosas, Marian Verhelst, and Sofie Pollin. "Performance analysis of in-band full duplex collision and interference detection in dense networks." In Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual, pp. 595-601. IEEE, 2016.

[6] CTTC https://www.cttc.es/

[7] Peter Ruckebusch, Eli De Poorter, Carolina Fortuna, Ingrid Moerman, "GITAR: Generic extension for Internet-of-Things ARchitectures enabling dynamic updates of network and application modules", Ad Hoc Networks, Volume 36, Part 1, 2016, pp. 127-151, ISSN 1570-8705, https://doi.org/10.1016/j.adhoc.2015.05.017. (http://www.sciencedirect.com/science/article/pii/S1570870515001225)

[8] P. Ruckebusch et al., "WiSHFUL: Enabling Coordination Solutions for Managing Heterogeneous Wireless Networks," in IEEE Communications Magazine, vol. 55, no. 9, pp. 118-125, Sept. 2017. doi: 10.1109/MCOM.2017.1700073, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8030497&isnumber=8030366