GRANT **AGREEMENT NO.: 732174**

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

Orchestration and Reconfiguration Control Architecture

D5.3: Final Operational SDR Platforms with Advanced Reconfigurability/Reprogra-mmability Capabilities

Revision: v.1.0

Work package	WP5
Task	Task 5.1, 5.2, 5.3
Due date	31/1/2020
Submission date	31/1/2020
Deliverable lead	KUL
Version	1.0
Authors	Sofie Polin(KUL), Seyed Ali Hassani (KUL), Yi Zhang (TCD), Muhammad Aslam (IMEC), Jan Bauwens (IMEC), Wei Liu

	(IMEC), Ingrid Moerman (IMEC).
Reviewers	Roberto Bomfin (TUD), Joao Santos (TCD)

Abstract	The deliverable D5.3 includes the progress and implementation results obtained in ORCA Year 3 on the available SDR platforms with special focus on advanced reconfigurability and reprogrammability. The research groups explain the developed capabilities and functionalities they have integrated into the ORCA control plane.
Keywords	Reconfigurable SDR, advanced reprogrammability

Acknowledgment

This report has received funding from the EC under the grant agreement 731274.

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	

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

EXECUTIVE SUMMARY

One of the major targets of ORCA is providing advanced reconfigurable and reprogrammable software-defined radio (SDR) solutions. This will not only enable easier and larger experiments, it is also an essential step to take SDR technology from laboratory to deployment in the field. In Y1 and Y2, ORCA introduced SDRs with physical (PHY), medium access control (MAC) layer, and eventually end-to-end network functionalities, which are powerful enough to serve as a connectivity solution in the field.

The PHY and MAC solutions started from run-time programmable MicroBlazes (MBs) offering low-latency and end-to-end SDR connectivity, to reconfigurable PHY-MAC architectures with functionality that could be upgraded at run-time to certain extent, however upgrading hardware running on the FPGA is not jet possible. And the configurations are triggered by local intelligence, without high-level coordination at end-to-end connection level. When considering live reprogramming of ORCA functionality, the key questions are: how to do live reprogramming with FPGA to do fundamental hardware update, and how to do live reprogramming for end-to-end SDR bridging to SDN? This deliverable first details how the FPGA can be partitioned, and how a given reconfigurable region can be updated while other parts remain functional. It is shown how the software of the embedded ARM (for FPGA's with a such core) can be updated dynamically at run-time. A reinforcement learning approach is then presented to dynamically update end-to-end resource usage of the network. Here, abstraction is made of the processing, i.e., FPGA or ARM or GPP or any processing is assumed to be reconfigurable and can be moved on the fly.

Secondly, this deliverable answers the question of why such live reprogramming is necessary and what are the advanced functionalities that are enabled by such live reprogramming. Three examples are given of "dynamic deployment of intelligent control" that emphasize the importance of functional reconfiguration for future complex wireless communication systems. The first example shows how full duplex benefits from advanced and dynamic deployment of intelligent control, as this is essential to enable parameter tuning for the self-interference cancellation. The second example shows how such a full duplex system can dynamically be augmented by adding new radar functionality. The third example illustrates how we can scale and allocate computational resources to realize dynamic end-to-end services using virtualization on a shared physical infrastructure, which may comprise hybrid SDR-SDN functionality.

SDR can be reprogrammed, from FPGA, to embedded processor, and to the host computer. The control of such reprogramming is however complex, special tools and design flow are required for FPGA live reconfiguration, deep learning and reinforcement learning are important enablers. These are the key aspects discussed in this deliverable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AnSIC	Analog Self-Interference Cancellation
DDC	Digital Down Converter
DDS	Direct Digital Synthesizer
DevC	Device configuration Controller
DiSIC	Digital Self-Interference Cancellation
DUC	Digital Up Converter
DLS	Dithered Linear Search
EBD	Electrical Balance Duplexer
FD	Full-Duplex
HD	Half-Duplex
ICAP	Internal Configuration Access Port
IBFD	In-Band Full-Duplex
InP	Infrastructure Provider
LTE	Long Term Evolution
MB	MicroBlaze
MDP	Markov Decision Process
MVNO	Mobile Virtual Network Operator
NP	Network Provider
PCAP	Processor Configuration Access Port
PRB	Physical Resource Blocks
PRC	Partial Reconfiguration Controller
PS	Particle Swarm
Rad-com	Radar-communication
RAN	Radio Access Network
RF	Radio Frequency
RR	Reconfigurable Region
SDN	Software-Defined Networking
SDR	Software-Defined Radio
SI	Self-Interference
SP	Service Provider
STFT	Short-time Fourier Transform

1 INTRODUCTION

This deliverable intends to provide an overview of the ORCA SDR platforms, which facilitates an advanced level of flexibility. This is aligned with ORCA control plane development enabling live configuration of SDR devices and run-time re-programming in such a manner that there is no need to reboot/restart the hardware. Important is also that this reconfiguration happens wirelessly, over the air, which means that the hardware can stay where is it deployed for its operation and no access by an operator is needed. As such, an SDR facility that is deployed in a Smart Factory or other private deployment can be orchestrated and reconfigured easily remotely. The remaining chapters of this document focus on live reprogrammability and dynamic deployment of intelligent control. Both are crucial enables for remote orchestration.

1.1 Organization of the Deliverable

As illustrated in Figure 1, the organization of this document and the contribution in the ORCA Y3 showcase are as follows.

- Section 2: Live reprogrammability
  - Subsection 2.1 describes the integration of partial FPGA reconfiguration with EXT1 OC2 where IMEC provides ZYNQ SDR platform with run-time configurable FPGA [SC2].
  - Subsection 2.12.2 explains the software update required for live reprogramming on IMEC's ZYNQ SDR platform [SC2].
  - Subsection 2.3 includes TCD's effort to offer dynamic scaling and allocation of computational resources for end-to-end services [SC3].
- Section 3: Dynamic deployment of intelligent control
  - Subsection 3.1 explains run-time programmable control for Analog Self-interference Cancelation (AnSIC), where this capability allows simultaneous communication and radar functionality by KUL's In-Band Full-Duplex (IBFD) capable SDR [SC2].
  - Subsection 3.2 describes the Optimal allocation of radio resources for heterogeneous virtual wireless networks.
  - Subsection 3.3 represents KUL's enhancement over the IBFD-capable SDR which can deliver a run-time reconfigurable opportunistic environmental sensing system [SC2].

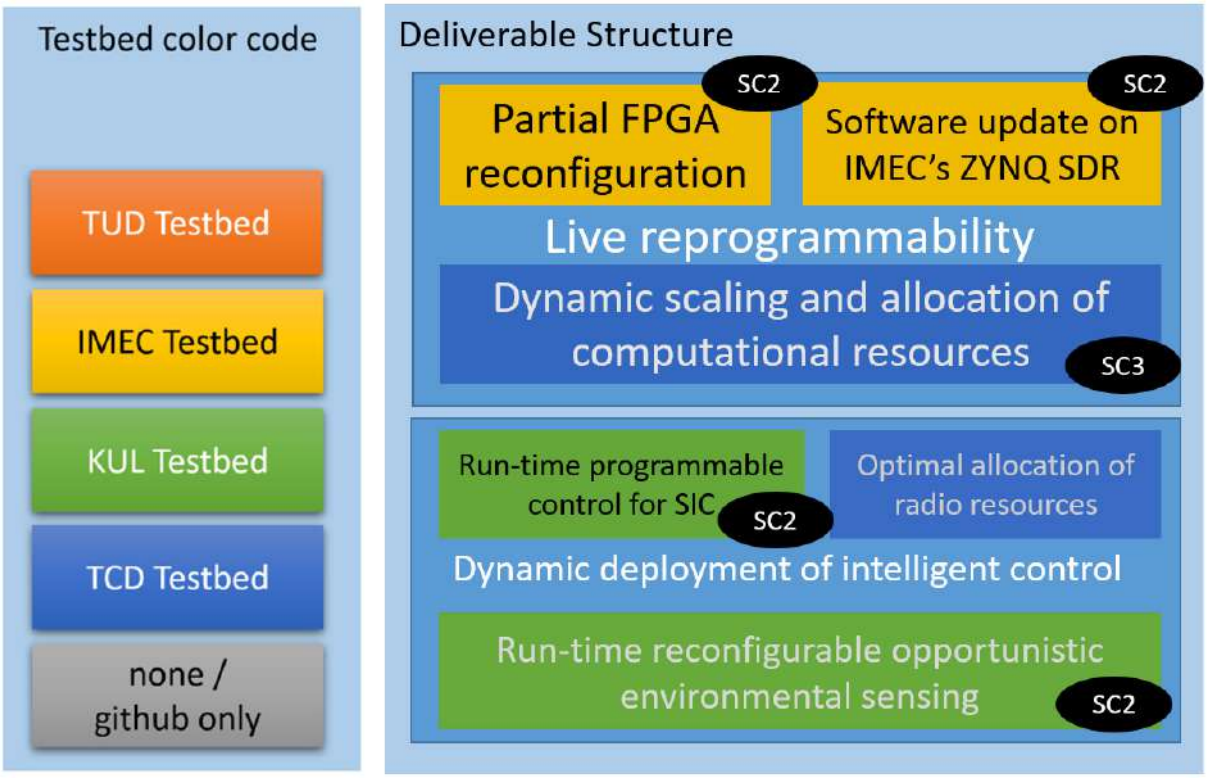

Figure 1. The overall structure of the deliverable.

The subsections mentioned above are further extended to explain the goals, challenges and implementation results in each item.

2 LIVE REPROGRAMMING

2.1 Partial FPGA reconfiguration integration with EXT1 OC2

2.1.1 Motivation

In the ORCA OC2 extension, a solution is developed to use partial reconfiguration of FPGA on two SDR platforms in real time. The reconfiguration happens without interruption of the FPGA logic in static region. The solution is validated with a simple design, the partial reconfigurable region can be either a DDS (Direct Digital Synthesizer) or an Long Term Evolution (LTE) waveform generator.

In the multi-channel virtual Zigbee transceiver implementation, IMEC's Digital Down Converter (DDC) and Digital Up Converter (DUC) Filter banks are used to provide narrow band IQ samples for multiple channels. In the single channel Zigbee transceiver implementation, the channel's bandwidth is configurable by changing the sampling rate of the radio frontend. While this was originally the case, now the configuration is achieved by tuning the interpolation/decimation ratio in the signal processing modules, the sampling rate of the frontend is left untouched, which makes the reaction speed of bandwidth adjustment faster.

Until recently, these two implementations (i.e., multi-channel virtual Zigbee transceiver and single channel bandwidth adjustable Zigbee transceiver) are two independent FPGA designs. Switching between one design to another requires the board to be configured by a host computer, which replaces the entire FPGA design, and the software running on the ARM processor. The partial reconfiguration of FPGA solution can avoid such kind of intervention by the host PC, and hence make both designs available in real-life applications.

2.1.2 Implementation results

2.1.2.1 Identify Blackbox

At the first place, we need to identify what is different in the two designs. It is rather intuitive to think that the DDC/DUC filter banks used in multi-channel design is not needed in the single channel design. However, these filter banks are not directly wiped out, as digital interpolation/decimation filters are required for changing the bandwidth of the single channel, and the filters in the banks are reused to a certain extent. Very fortunately, the DDC/DUC filter banks are indeed where the differences are located, therefore we replace the instance of these filter banks (i.e., *ddc_inst, duc_inst*) by black boxes respectively, and synthesis these modules separately from the global design.

2.1.2.2 Remove parameters in the connection ports

Originally the instances of these modules contain parameters. These parameters are used at the moment of instantiation to configure the number of channels in the multi-channel design, amongst other things. However, a module in Reconfigurable Region (RR) should have fixed connections to the static region, therefore such kind of configuration is not allowed. As a result, we must remove these parameters and adjust the Verilog files where the modules are instantiated. Another consideration is that the multi-channel design was originally with 8 channels, and it takes more than 90% resources. One can expect to have issues to compartment such a highly congested FPGA design into partial and static regions. Therefore instead of having 8 channels, we made modification to support 4 channels.

2.1.2.3 Partial reconfiguration workflow

After the modification, we synthesis the reconfigurable modules, and top design separately. Resulting in three .dcp (design checkpoint files). It is noted that the DDC and DUC filter banks in the top design

are black boxes, which can be resolved by filling the cell with corresponding synthesized design in .dcp file.

As first step, we make one completely synthesized design by stitching the top design with the multi-channel modules, then we specify the DDC, DUC filter banks to be reconfigurable, and assign them to two pblocks (i.e., Xilinx's Vivado defines RR by pblock). ‘

The pblocks should contain sufficient resources to implement the filter banks. This is particularly challenging as the DDC filter bank consumes a large amount of DSP48, which are not concentrated at one place of the FPGA. As a result, some additional regions are added to the pblock containing DDC inst.

The placed design of the multi-channel configuration is shown in the figure below.

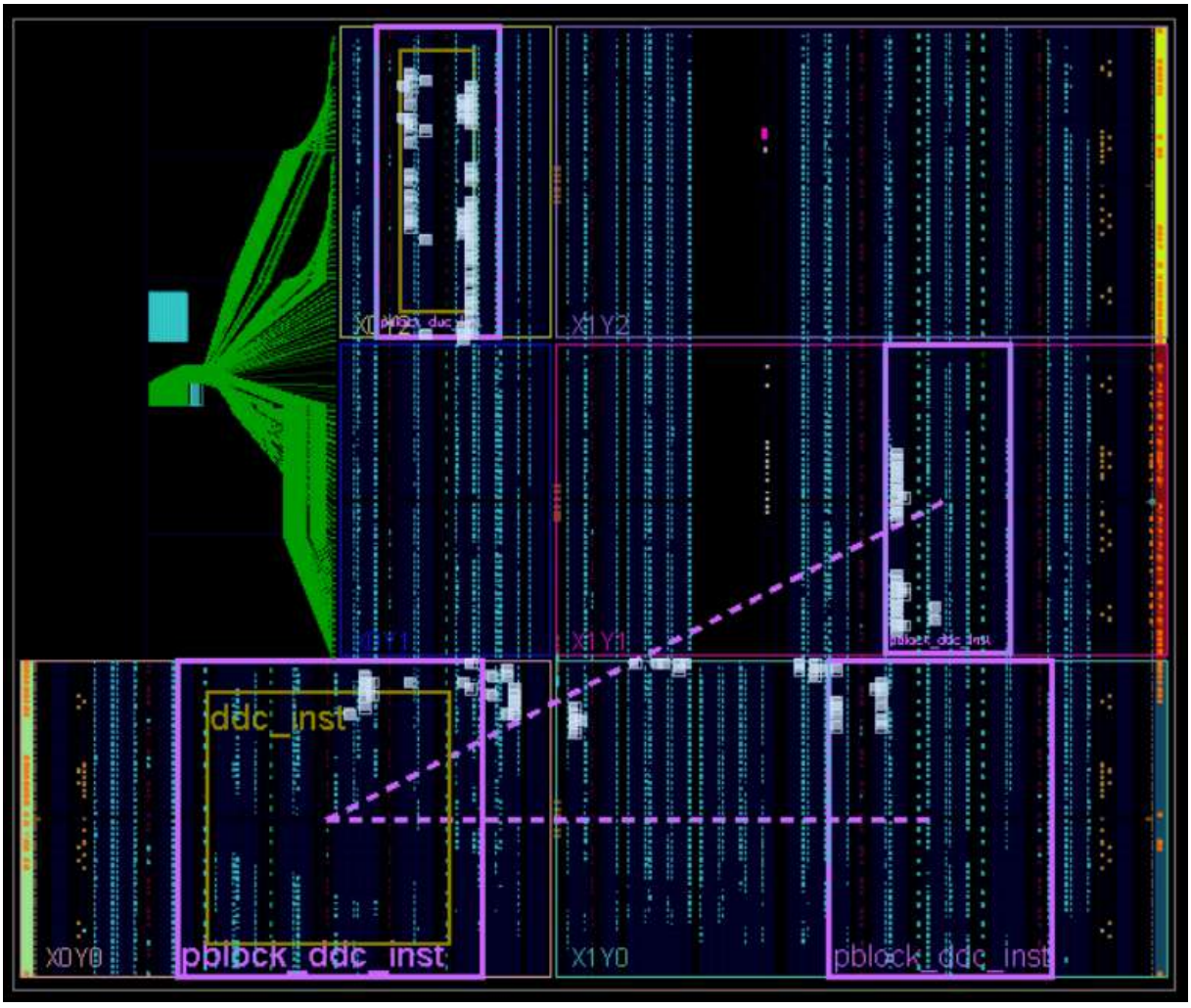

Figure 2: Floor plan of placed design in multi-channel configuration.

As you can see, DUC inst pblock is a simple rectangular shape, whereas DDC inst pblock contains three parts. The partition pins are highlighted in light grey, these pins are the connection pins between the static design and the logic in the pblock.

After the placement of multi-channel configuration, we clear the pblocks and lock the static design at routing level. At this stage, the floor planning is shown in the figure below. As you can see, the partition pins remain at the same location, but the logic inside the pblock is wiped out from the design. After placement, we issue the *route_design* command. It is very important to check if all nets are

routed. After a few iterations, we obtained a fully placed and routed design, and save the result as a checkpoint.

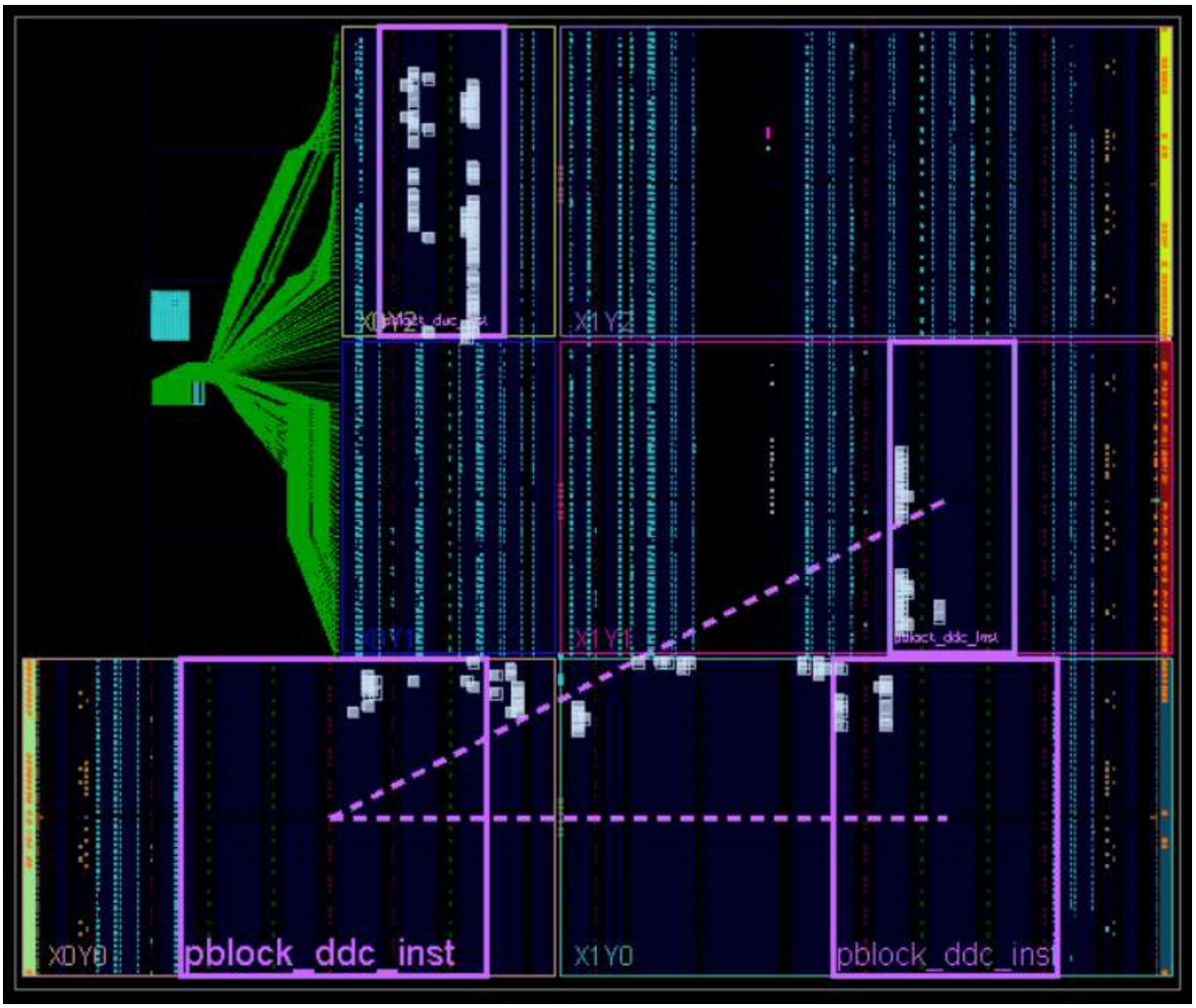

Figure 3: Floor plan of placed design with black boxes.

After that, we update the black boxes with synthesized single channel modules, and implement the design, this results in another design checkpoint file. The placed design of single channel configuration is shown in figure below. As expected, it consumes less resources than the multi-channel configuration. The static part of the design is in the locked status, therefore appears as orange.

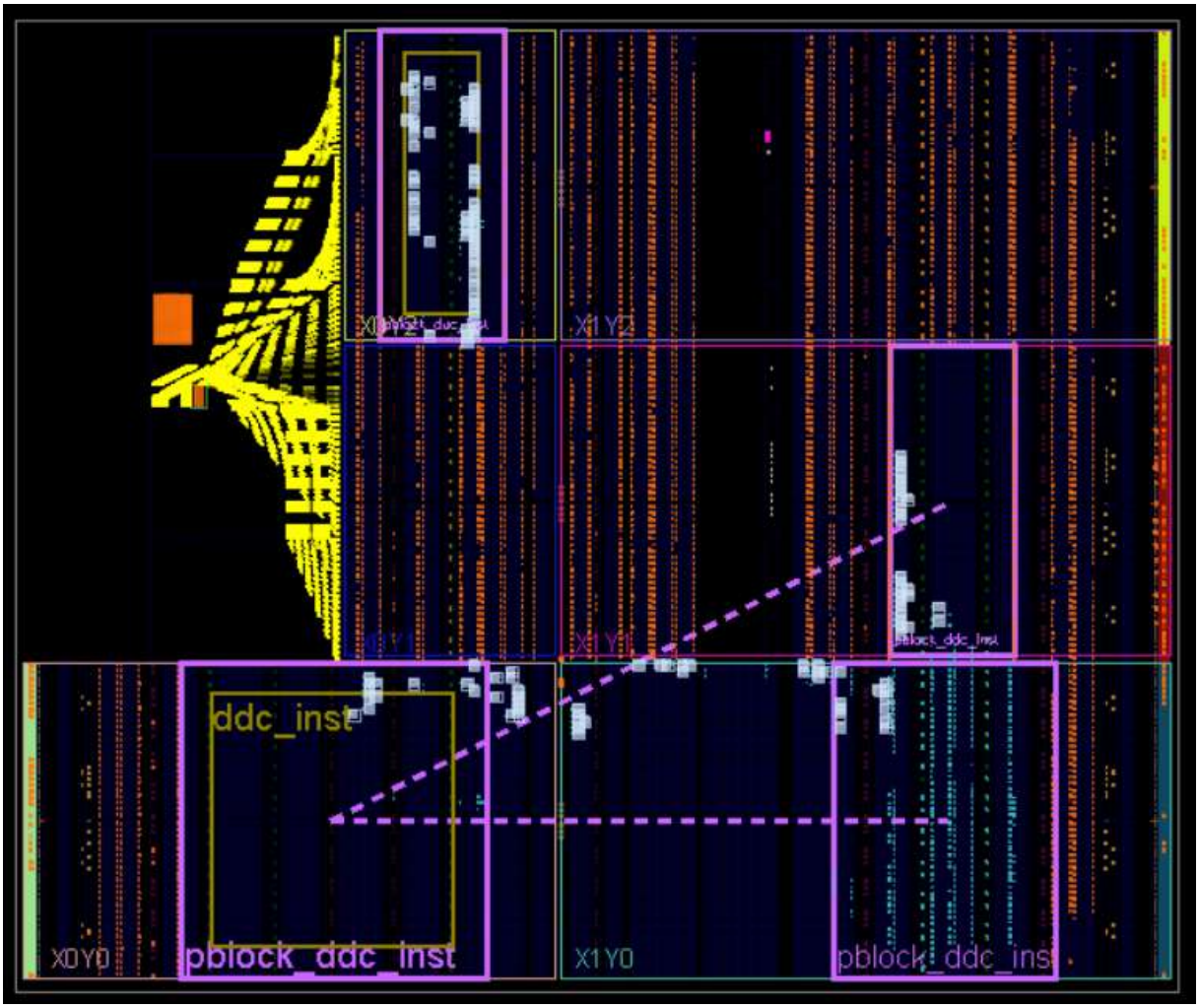

Figure 4: Floor plan of placed design of single-channel configuration.

We verify both checkpoint files match with each other, and generate bitstreams. This results in two top level bitstreams (i.e. one complete bitstream for single channel configuration and one for multi-channel configuration), and four partial bitstreams corresponding with : i) multi-channel ddc inst; ii) multi-channel duc_inst; iii) single-channel ddc_inst; iv) single-channel duc_inst

Finally we also generate bitstreams for the blackbox configuration, in order to avoid floating pins, the partition pins are tied to buffer ports. This results in another 3 bitstreams, one top level bitstream with blank implementations of the ddc_inst and duc_inst, and 2 blank partial bitstreams.

2.1.3 Testbed integration

The partial reconfiguration implementation is available on git repository. Users of the testbed may use the configuration if their application requires switching between single channel (adjustable channel bandwidth) and multi-channel implementation in real time.

2.2 Software updates on ZYNQ SDR exploiting EXT1 OC2 and WiSHFUL Controller

2.2.1 Motivation

In the previous section, we discuss how partial bitstreams are generated for reconfigurable regions. However only having bitstreams are not sufficient to switch between single and multi-channel implementation at runtime, the loading of partial bitstreams needs to be handled by the software running on the Processing System (i.e. ARM) of the SDR.

In order to perform easy and quick experimentation on SDR devices, it was chosen to use the output of the Wishful H2020 project. Wishful offered open, flexible & adaptive software and hardware platforms for radio control and network protocol development allowing rapid prototyping of innovative end-to-end wireless solutions and systems in different vertical markets. The single channel transceiver supports real-time bandwidth adjustment, this has been supported by WiSHFUL controller before, the new implementation makes use of this feature.

2.2.2 Implementation results

Xilinx 7-Series FPGAs can either make use of the Internal Configuration Access Port (ICAP) or the Processor Configuration Access Port (PCAP) to apply partial reconfiguration. PCAP is only available in Zynq devices equipped with hardwired embedded ARM processors, this is fortunately the case of our platform. One option is to use the Device configuration Controller (DevC) and PCAP to implement a software-controlled reconfiguration. The second option is to use ICAP port and the AXI_HWICAP or the Partial Reconfiguration Controller (PRC) IPs of Xilinx. The ORCA OC2 extension implements the second option, as ICAP interface offers much better throughput and therefore shorter configuration time. Due to lack of resources, placing the PRC is avoided in this implementation, we have adopted the same principle but the configuration is achieved via the PCAP interface.

The generated partial bitstreams are stored on an SD card. When required, the files are stored into a given memory location of the ARM processor. The device configuration controller is initialized, and many other operations (such as enable PCAP clock, disable level shifters between the PS and PL) are completed before partial bitstreams can be loaded.

In the last year of Wishful project, the Xilinx Zynq-based SDR was added as a supported platform for Wishful configuration. On top of the supported configuration options for off-the-shelf 802.15.4 devices, following configuration was available through the Wishful framework:

- Run-time configurable data rate
- Run-time configurable channel spacing
- Run-time configuration of channel bandwidth

By consequence it enabled experimentation beyond the IEEE802.15.4 standard. Details can be found in D3.6 of the H2020 Wishful project.

Beyond the Wishful project, in the context of Orca, Wishful was extended with new configuration options in order to make use of the novel multi-channel transceiver.

Table 1: Parameters supported by WiSHFUL controller.

Name	Description
SDR_NUM_CHANNELS	Set the number of channels for the multi-channel transceiver (1,2,4,8)
SDR_CH_RECEIVER	Enable transmissions to a specific receiver on channel
SDR_CH_TXPOWER	Set TX power on a given channel

2.2.3 Testbed integration

The software solution is coupled with hardware implementation, which is offered in the same way, users of testbed may request access to git repositories to use the partial reconfiguration software solution. The extension to Wishful controller can be used under the Wishful licensing condition, more information is available at Wishful repository[1].

[1] Wishful repository https://github.com/wishful-project

2.3 Dynamic scaling and allocation of computational resources for end-to-end services

2.3.1 Motivation

Artificial intelligence and machine learning have been recently utilised in telecommunication and networking industries to solve complicated research problems in an automatic fashion and to save the human efforts. Reinforcement learning, as one of the most promising machine learning methods, has been mostly used for solving optimisation problems intelligently, especially in a highly dynamic and stochastic environment. Virtualized end-to-end services involve centralized control of the resources where cloud-based computing is necessary to be used. Multiple slices of the end-to-end services running on the same cloud-based system share the same physical resources, e.g. CPUs, memories, etc. How to optimise the computational resources is an interesting research topic that has been investigated by researchers in recent years. We focus on the dynamic CPU allocation problem for end-to-end services, to optimise the energy consumption of the cloud-based system as well as balancing the cloud performance.

For a cloud-based end-to-end service network, a reinforcement learning agent can be designed and implemented to learn the features and characteristics of the CPU utilization of the existing running processes. When the new similar process demands coming in to the system, the agent is capable to capture the feature of the new coming service demand and provide the recommendation to the system to dynamically re-allocate the computational resources, i.e., number of assigned CPU cores. The overall performance and energy consumption of the cloud system can be abstracted into "reward" values as the parameters of the reinforcement learning agent to help it make correct decisions.

Modelling the cloud resource allocation environment is the key point for the reinforcement agent to make correct decisions. We use one of the most efficient ways, namely Markov Decision Process (MDP), to model the cloud system CPU utilization status, and solve the MDP with reinforcement learning to get optimal policies to save power consumption and maintain the performance of the CPUs. We use the Software-Defined Networking (SDN) and SDR programs running in TCD's testbed to evaluate the results.

2.3.2 Implementation results

We implement the reinforcement learning agent with the following steps.

1) Model the system by a MDP.
2) Solve the MDP by value iteration and get the optimal policies from existing observation of the system.
3) Apply the optimal policies to the future CPU allocation actions.

Figure shows our MDP transition diagram with 3 CPU states, i.e., "s0", "s1", "s2". Corresponding to different levels of CPU utilization, low, medium and high. For each state we have 3 kinds of actions, "a0", "a1", "a2", corresponding to 1) keep the CPU numbers; 2) add one CPU; 3) reduce one CPU. The transition probability "P (s' | s, a)" means the probability of reaching state "s'" after the action "a" is taken from state "s". The reward "r (s, a, s')" means the reward we define for each transition from "s" to "s'" using action "a".

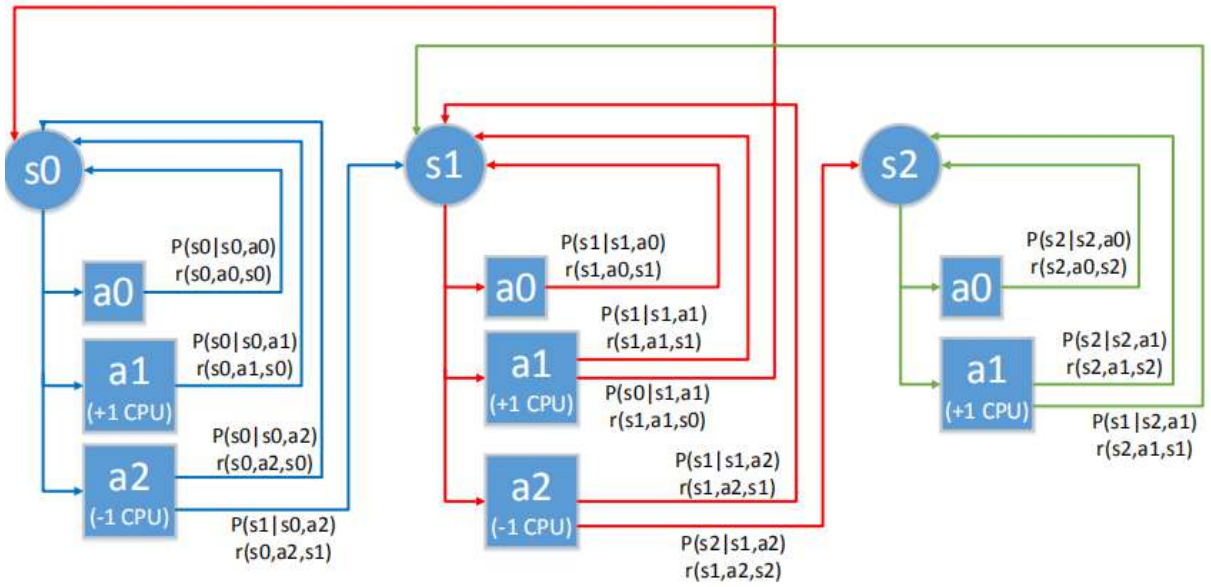

Figure 5: State transitions of the MDP for CPU allocation.

We use the conventional value iteration algorithm for the reinforcement learning agent to learn the optimal policy. We pre-define "P (s' | s, a)" and "r (s, a, s')" for each state "s" and action "a" for the SDN and SDR programs, and then calculate "V(s)" as the value function of the state "s" according to the value iteration algorithm. Figure 6 shows the value iteration results for SDN and Figure 7 shows the value iteration results for SDR. It means that for SDN, the reinforcement learning agent prefers to push the CPU utilisation to higher percentage, while for SDR, the reinforcement learning agent prefers to push the CPU utilisation to lower percentage. Correspondingly, we get the optimal policies after the value iterations for each state "s" for SDN and SDR respectively. The reason why SDN and SDR processes have different results is that SDN process is easier to be parallelized and load-balanced, thus it is safer for SDN to stay on the high CPU load state without getting the CPU overloaded.

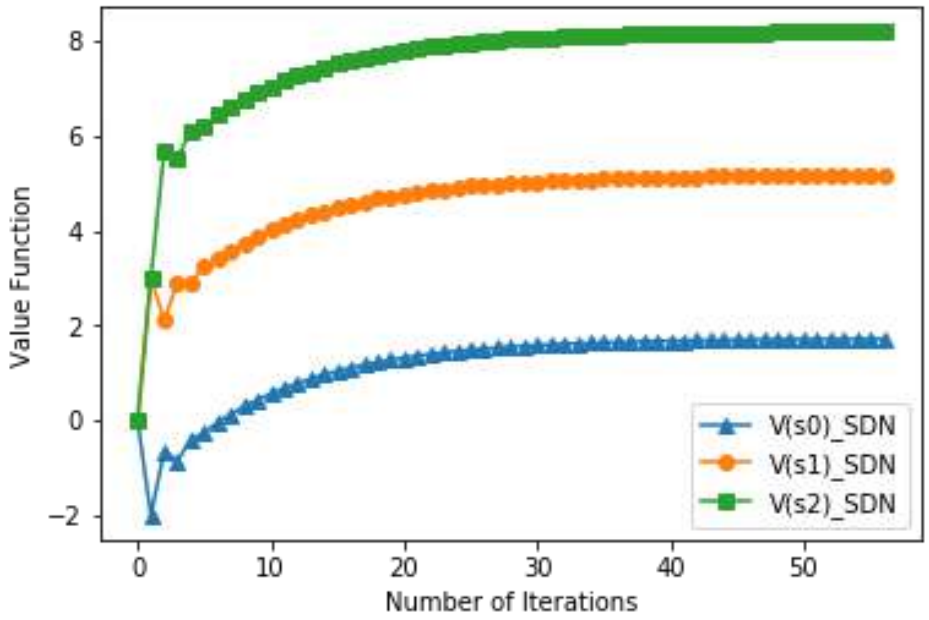

Figure 6: Value iteration for SDN.

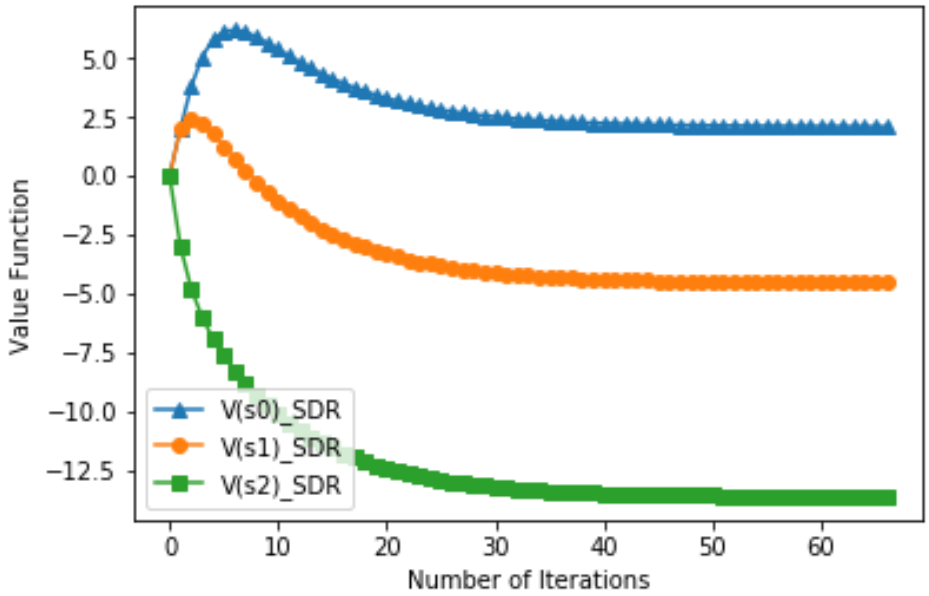

Figure 7: Value iteration for SDR.

2.3.3 Testbed integration

After using the value iteration to get the optimal policies, the reinforcement learning agent is able to apply the optimal policies to future CPU allocation requests. The optimal policy can be embedded in the controller of the CPU allocation engine of the virtualized services for testbed evaluations, e.g. in Trinity College Dublin's Iris testbed (http://iristestbed.eu/). The end-to-end services involve SDR and SDN processes in the cloud system thus the aforementioned analysis can be used to dynamically adjust the number of assigned CPU cores (using the LXD CPU scaling method mentioned in Deliverable D3.5). The utilization percentage of each CPU core is monitored in real time, and categorized to different states mentioned above. Optimal policy, including whether to keep the CPU number or to add/reduce one CPU is applied periodically (typically every one minutes, which can be changed to other values as per requests), to save energy as well as keeping the running CPUs in reasonable utilization percentages to maintain their performance.

3 DYNAMIC DEPLOYMENT OF INTELLIGENT CONTROL

3.1 Dynamic EBD control for full duplex communication and radar

3.1.1 Motivation

Typically, an IBFD transceiver benefits from two stages of self-interference cancelation; one at the analog Radio Frequency (RF) and one at the sampled digital baseband. The former plays a vital role as it prevents the Rx from saturation. KUL's AnSIC makes use of an Electrically Balanced Duplexer (EBD) [1], which reveals high linearity as well as low insertion loss. Although its impressive Self-Interference (SI) cancelation features privileges it rather than its counterparts, the EBD is sensitive to the near field environmental changes [2] and has to be *retuned dynamically as function of changes in the environment*. Since the signal reception is not possible during EBD adaptation the tuning algorithm, hence, has to execute as fast as possible in such a manner that it maximizes the Tx-Rx isolation. One the other hand, various adaptation techniques tend to reveal different performances depending on the EBD status.

In Y1 and Y2, we demonstrated Particle Swarm (PS) [3] and Dithered Linear Search (DLS) [4] as two potential approaches to perform EBD adaptations. The experiments show that the PS-based EBD tuner always achieves better performance when there is a significant change in the near field. As a cost, this natural-inspired algorithm requires long convergence time and is computationally expensive and resource-demanding. Whereas, the DLS tuner is noticeably faster and is suitable to track slight environmental variations, once the PS optimizer already has initialized the EBD. Therefore, *a hybrid mechanize is necessary to choose the optimum tuning method actively*.

3.1.2 Implementation results

The KUL's IBFD SDR utilizes a MB to run the BED tuning algorithm. Table 2 lists the required hardware resources for PS and DLS. The high amount of resources demanded by the PS tuner implies that only one of the optimizers can be programmed on the MB. Therefore, run-time reprogrammability capability is needed to enable a hybrid EBD tuning approach.

Table 2: EBD tuning algorithm, hardware allocation.

	PS	DLS
64-bit floating-point	>3224	<24
32-bit integer	64	4

Besides, the metric for sensing the Tx-Rx isolation is the amount of the leaked/reflected self-transmit signal. While in a practical IBFD scenario, there is an intended signal transmitted from a distant device which may mislead the tuner. Thus, the level of the desirable signal has to be estimated and subtracted from the power at the EBD's receiver port.

Figure 8 shows the hybrid EBD tuning concept where PS and DLS are exploited to perform dynamic EBD control. As shown, two thresholds are defined to decide which adaptation be performed by the MB. Once the tuning program is determined, the PC reprogram it on the MB deployed on the FPGA and makes it operational.

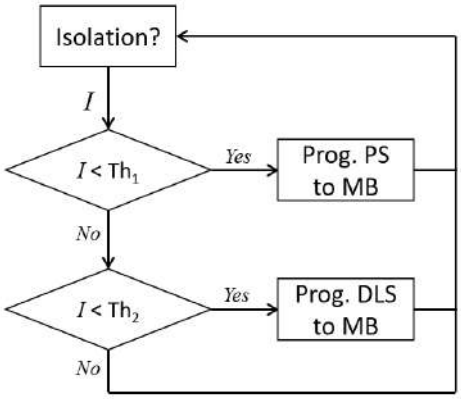

Figure 8: Hybrid dynamic EBD control.

Figure 9 also depicts the block diagram of the realization improved on KUL's IBFD-capable SDR. This architecture allows the tuner to be aware of a second party transmission produced by a remote device, which is a typical case in IBFD communication. As demonstrated in this diagram, the tuner is feed by the received signal and the estimated desired signal by the digital self-interference cancelation module, shown by the Digital Self-Interference Cancellation (DiSIC) block in the figure. This mechanism significantly reduces the chance of an unneeded EBD tuning trigger which increases the probability of packet loss.

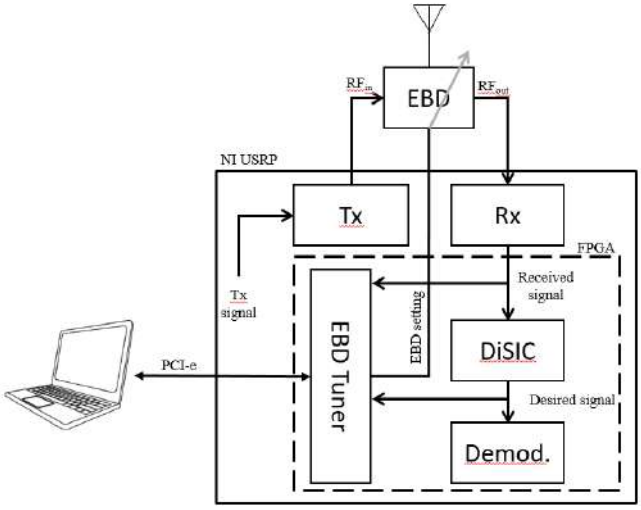

Figure 9: Block diagram of KUL's IBFD-capable SDR.

Figure 10 depicts the measured isolation resulted by the hybrid dynamic EBD control technique. In this test, two thresholds of 20 dB and 45 dB are set to perform PS and DLS adaptation, respectively. At the beginning of the experiment, the MB is programmed by PS optimizer to reach > 45 dB isolation within 300 ms. Once the EBD is initialized, the DLS algorithm is reprogrammed to track the slight environmental changes in 124 μs, maintain the SI rejection level better than 45 dB.

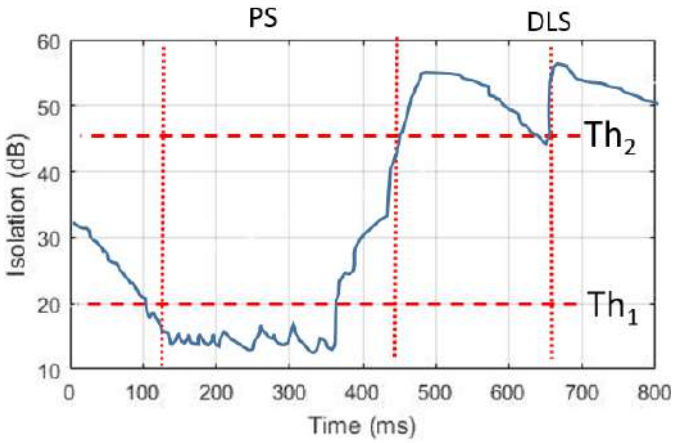

Figure 10: Measure performance of the dynamic EBD control algorithm.

3.1.3 Testbed Integration

The dynamic EBD tuner is ported to KUL IBFD[2] testbed as well as ORCA SC2, and it delivers the required Tx-Rx isolation, that is necessary for concurrent IBFD communication as well as radar functionality, which is discussed later in this document.

[2] https://www.esat.kuleuven.be/telemic/research/NetworkedSystems/infrastructure/full-duplex-testbed

3.2 Optimal allocation of radio resources for heterogeneous virtual wireless networks

3.2.1 Motivation

Radio hypervisors enable shared Radio Access Network (RAN) deployments, where multiple tenants, e.g., Mobile Virtual Network Operators (MVNOs), Service Providers (SPs) and verticals, share a common physical network infrastructure, decreasing the deployment costs and the amount of underutilised radio resources. RAN sharing allows operators to lease their spare capacity to MVNOs, cover areas they have not deployed infrastructure by leasing resources from MNOs in these regions, and share their spectrum resources among themselves to support a larger number of users [5]. Figure 11 illustrates two scenarios in which a radio hypervisor manages the real radio resources for creating different virtual wireless networks.

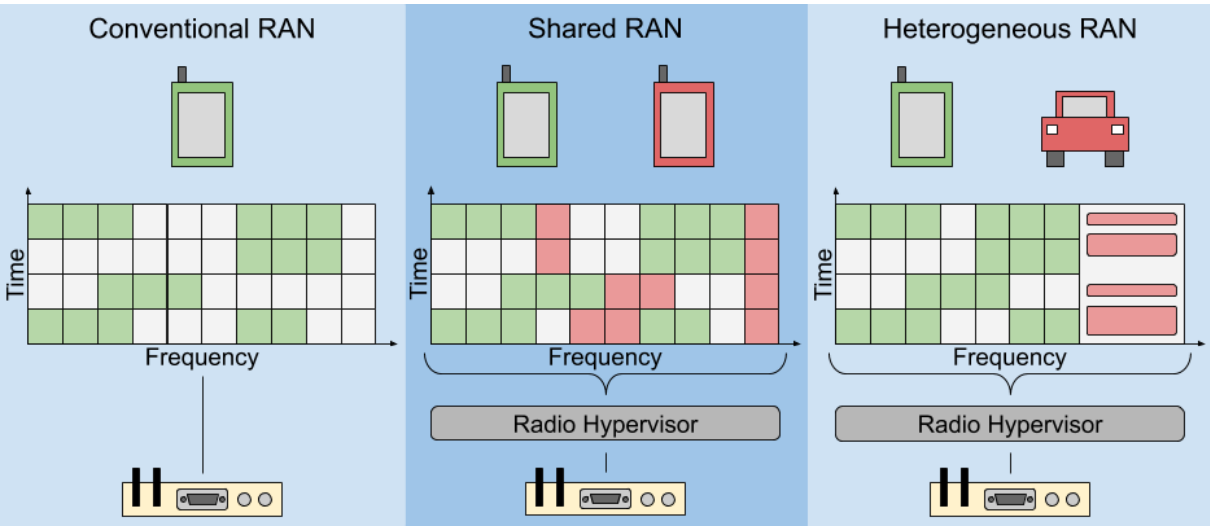

Figure 11: Radio hypervisors enable the sharing of the physical network infrastructure by multiple tenants and/or the use of a single hardware platform to realise heterogeneous virtual wireless networks.

The provision and instantiation of virtual wireless networks, can be offered as a service by MNOs and Infrastructure Providers (InPs) (which do not provide services to end-users), breaking the monolithic nature of legacy mobile networks into value-chain networks. The InPs and MNOs must cope with a diverse set of performance requirements from their tenants, and as a consequence, they need to efficiently dimension and allocate their physical resources for instantiating virtual wireless networks. However, the optimal resource granularity and resource allocations mechanism for InPs and MNOs to deploy heterogeneous virtual wireless network remains an open challenge.

3.2.2 Implementation results

General-purpose radio hypervisors employ technology-agnostic radio virtualisation techniques that multiplex the physical radios at the PHY layer, using low-level radio resources, e.g., time and frequency, for creating heterogeneous virtual wireless networks. In this deliverable, we show our preliminary results where we modelled the allocation of low-level radio resources for heterogeneous virtual wireless networks as a mixed-integer linear programming problem. In ORCA's final report, we will further extend our mechanism to consider the subcarrier spacing of different heterogeneous virtual

wireless networks, as well as include a heuristic for reducing the decision time of the resource allocation.

In our formulation, we consider multiple spectrum bands, each possessing a regular grid of radio resources, and we adopt a modular set of constraints that not only encompasses time and frequency resources but also can be easily extended to support new degrees of freedom, e.g., space and antennas. We assessed the trade-offs and performance of our resource allocation mechanism for maximising the radio resource utilisation with respect to the number of received requests for heterogeneous virtual wireless networks and the granularity of the resource allocation, i.e., the minimum amount of resources that a radio hypervisor can allocate to each of them. **Figure 12** shows the results of our analysis. In addition, we also evaluated the trade-offs for the decision time with respect to the resource granularity and amount of requests, as shown in **Figure 13**. Our results show that the number of heterogeneous virtual wireless network requests is the most discriminant factor in both the resource utilisation and the decision time. In contrast, the granularity of the resource allocation does not affect the results considerably.

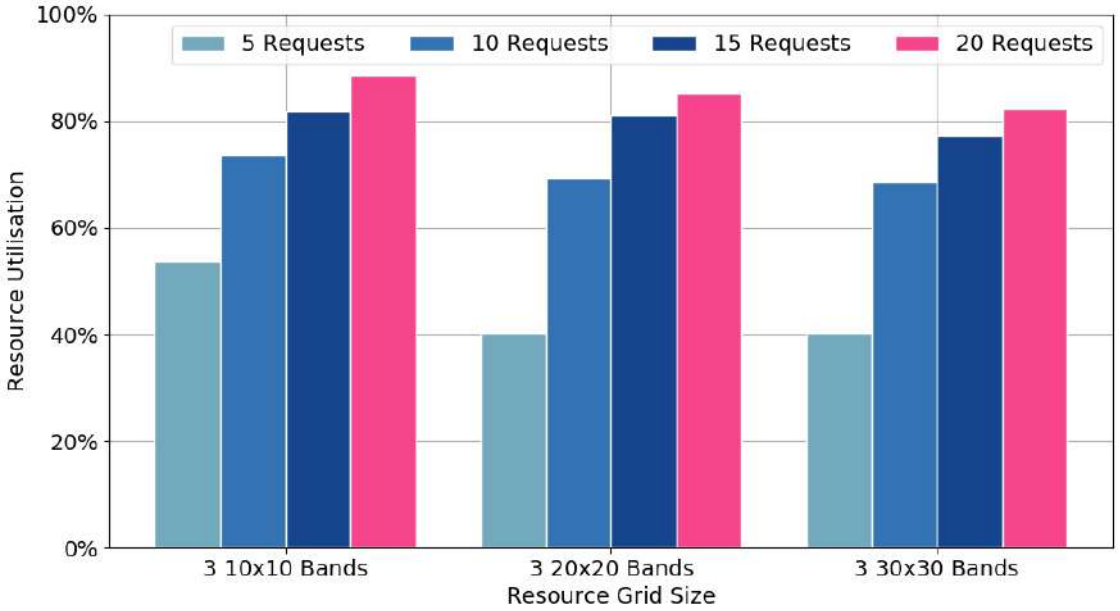

Figure 12: Resource utilisation achieved on different grid sizes in relation to the number of virtual wireless network requests.

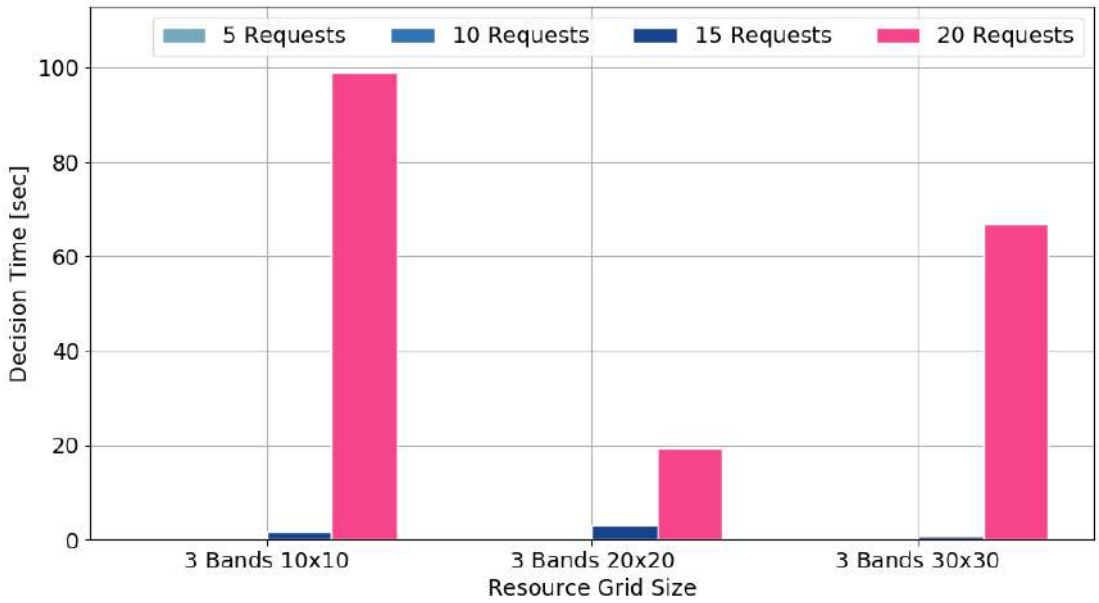

Figure 13: Decision time for different grid sizes in relation to the number of virtual wireless network requests.

3.2.3 Testbed integration

Our solution for optimal resource allocation of heterogeneous virtual wireless networks can be integrated with different types of general-purpose radio hypervisors who allocate resources in the time or frequency domains, or even a combination of both in a grid-like approach similar to Physical Resource Blocks (PRBs). On future works, we plan on integrating our resource allocation mechanism with HyDRA, a general-purpose radio hypervisor used and extended in ORCA's 2nd year, for evaluating the resource allocation performance and overhead in real deployments of heterogeneous virtual wireless networks.

3.3 Novel application to an SDR platform, reconfigurable radar-communication system

3.3.1 Motivation

While the IBFD opens a new horizon to boost the throughput of wireless communication, it potentially enables applications that require constant channel awareness, even while the device is emitting a high power signal. For instance, the environmental reflections of the transmit signal contain the Doppler state of the channel and can be utilized by the device itself to render radar-like functionality.

In Y3, KUL represents a Radar-communication (Rad-com) system [4, 6] that opportunistically employs the Tx signal to enable environment sensing. The KUL's Rad-com system allows context-aware networking where each node of the network can be seen as a mono-static radar.

Although the reflection of the Tx signal for a moving object carries its Doppler state, some distortions have to be removed from the received signal until the radar implements accurate detection.

In the radar point of view, the direct Tx leakage, the static reflections of the self-transmit signal, and the message emitted by a distant communication node are the three sources of interference that affect the radar. Besides, the proposed system has to utilize the already-existing hardware and does not demand a specific waveform or occupy an extra spectrum.

3.3.2 Implementation results

Figure 14 depicts the block diagram and the picture of the developed Rad-com system, equipped with both analog and digital self-interference cancelation modules (shown by AnSIC and DiSIC, respectively). In this architecture, the direct Tx leakage is suppressed through an EBD-based AnSIC block, enhancing the received signal for further self-interference rejection at the digital domain. The radar processing is partially deployed on the FPGA and is reconfigurable in such a manner that it can achieve various velocity detection ranges, regarding the application.

Although employing joint analog and digital SIC improves the quality of the detected Doppler signal, the adaptation of the DiSIC block can suppress the Doppler reflections. Hence, the radar can be reconfigured to operate with and without the DiSIC module. This mechanism enables a kind of application-dependent trade-off, where the system can be reconfigured based on the priority of functionality.

When a high level of Tx-Rx isolation is needed to reconstruct a low-SNR communication message, the system increases the adaptation rate of the DiSIC module. Meanwhile, it reconfigures the radar data flow to use AnSIC merely.

Whereas, in showcases that the radar functionality is required to sense slight motions, the system improves the radar configuration by employing the DiSIC module, with a low-rate adaptation, to cancel the residual SI from the radar signal.

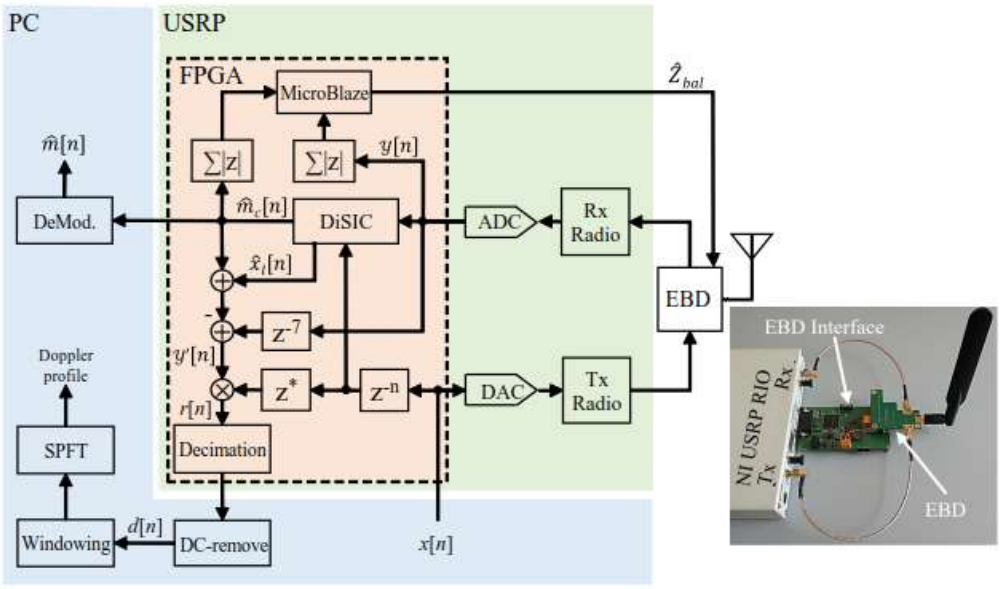

Figure 14: Block diagram of the KUL's IBFD Rad-com system (Left). The picture of the implemented system (Right).

Figure 15 shows the Doppler profile obtained by Short-time Fourier Transform (STFT), when a reflector moves forward and backward at different velocities. Each segment in this spectrogram is 800 ms including 85% overlap with the adjacent frame. There is a visible artifact in the spectrogram when the XY table changes the direction of the target. It due to inertia in the mechanical setup.

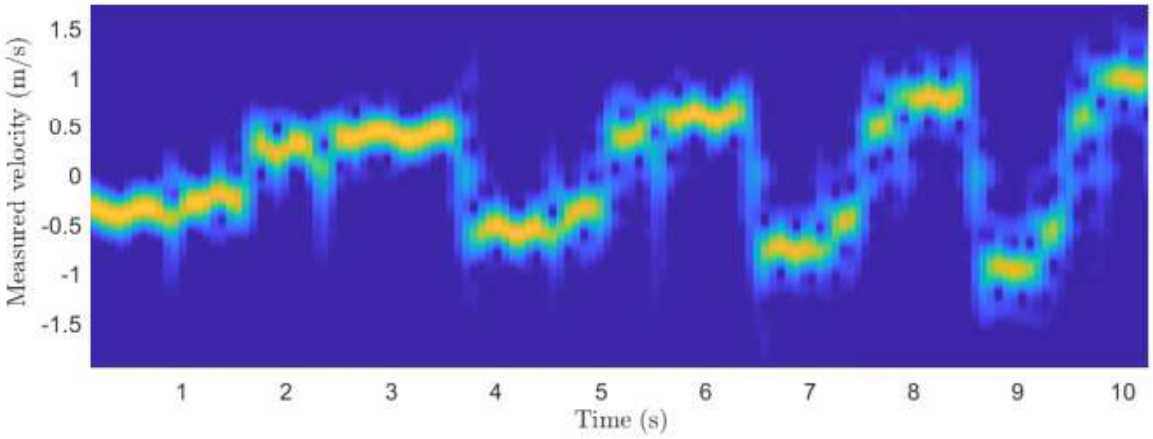

Figure 15: Measured Doppler-profile.

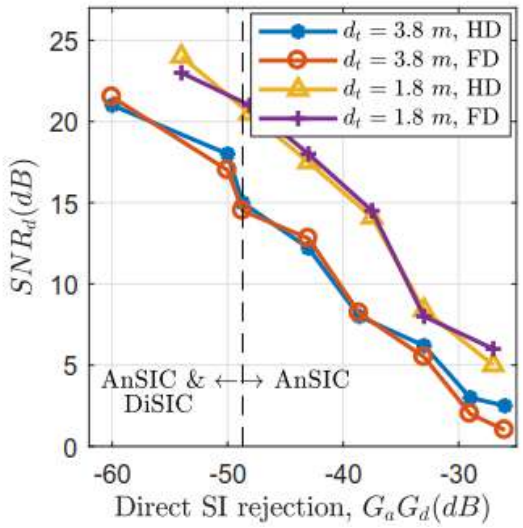

Figure 16: Measured SNR versus the joint analog and digital direct SI rejection (G_aG_d) as a function of the target distance from the radar d_t, in half/full duplex communication mode.

Figure 16 illustrates the performance of the radar in terms of the SNR of the detected Doppler signal. For this test, a moving reflector at 1.8 m and 3.8 m moves at 0.3 m/s, and the radar transmits a −8 dBm IEEE 802.11p-like waveform at 1.74 GHz. The measurement has been done with and without a second party transmitter, shown by Half-Duplex (HD) and Full-Duplex (FD) in the graph. As shown, the system produces a better Doppler when the joint analog and digital SIC are operational. However, for the cases that the modem demands higher Tx-Rx isolation, the Rad-com system reconfigures the radar, so it only makes use of the AnSIC module and increases the DiSIC's adaptation rate for communication.

3.3.3 Testbed integration

The KUL's rad-com system is integrated to KUL IBFD testbed[3] and ORCA SC2, where a control unit is enabled to not only control a robot but also sense its reaction and operational status.

[3] https://www.esat.kuleuven.be/telemic/research/NetworkedSystems/infrastructure/full-duplex-testbed

4 CONCLUSIONS

This document describes the main contributions made to the ORCA SDRs during Y3, focusing on reconfigurable and reprogrammable SDRs that facilities advanced network configuration in real-time. The achievements are organized in two main sections. The first section explains live reprogrammability and the second details how ORCA SDRs can dynamically employ intelligent control. In each section, the ORCA partners provided an overview of the flexibility of their SDR implementations, more particularly the following contributions are introduced:

- IMEC - Partial FPGA reconfiguration integration with EXT1 OC2, and software updates on ZYNQ SDR platform exploiting EXT1 OC2 and WiSHFUL Controller
- KUL - Dynamic EBD control for full duplex communication and radar, and reconfigurable radar-communication system which enables novel applications via a communication node.
- TCD - Optimal allocation of radio resources for heterogeneous virtual wireless networks, and Dynamic scaling and allocation of computational resources for end-to-end services

Additionally, a brief description was given on how the aforementioned implementations were integrated into the partners' testbeds and related ORCA showcases. Through these efforts, we hope the research community can use the reconfiburability/reprogrammability on SDR achieved in this work package for wireless network experimentations.

5 REFERENCES

[1] van Liempd, Barend, Benjamin Hershberg, Saneaki Ariumi, Kuba Raczkowski, Karl-Frederik Bink, Udo Karthaus, Ewout Martens, Piet Wambacq, and Jan Craninckx. "A+ 70-dBm IIP3 electrical-balance duplexer for highly integrated tunable front-ends." IEEE Transactions on Microwave Theory and Techniques 64, no. 12 (2016): 4274-4286.

[2] Laughlin, Leo, Chunqing Zhang, Mark A. Beach, Kevin A. Morris, and John L. Haine. "Dynamic performance of electrical balance duplexing in a vehicular scenario." IEEE Antennas and Wireless Propagation Letters 16 (2016): 1855-1858.

[3] Vermeulen, Tom, Barend van Liempd, Benjamin Hershberg, and Sofie Pollin. "Real-time RF self-interference cancellation for in-band full duplex." In 2015 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), pp. 275-276. IEEE, 2015.

[4] S. A. Hassani, K. Parashar, A. Bourdoux, B. van Liempd and S. Pollin, "Doppler Radar with In-Band Full Duplex Radios," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France, 2019, pp. 1945-1953.

[5] 3rd Generation Partnership Project, “3GPP TR 38.801: Study on New Radio Access Technology: Radio Access Architecture and Interfaces,” 3rd Generation Partnership Project, Tech. Rep., Mar. 2017.

[6] S. A. Hassani, A. Guevara, K. Parashar, A. Bourdoux, B. van Liempd and S. Pollin, "An In-Band Full-Duplex Transceiver for Simultaneous Communication and Environmental Sensing," 2018 52nd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 2018, pp. 1389-1394.