Deliverable Reference	:	D5.3
Title	:	Architectures related EGSE detailed design
Confidentiality Level	:	PU
Lead Partner	:	SPACEAPPS
Abstract	:	This deliverable presents the EGSEs required for porting the CDFF to distributed space-qualified and multi-robot architectures, for verification and validation.
EC Grant N°	:	730014
Project Officer EC	:	Christos Ampatzis (REA)

InFuse is co-funded by the Horizon 2020
Framework Programme of the European Union

DOCUMENT APPROVAL SHEET			
	Name	Organization	Date
Prepared and cross-reviewed by:	Jeremi Gancet	SPACEAPPS	30/01/2018
	Shshank Govindaraj		
	Shashank Govindaraj		23/01/2018
	Jeremi Gancet		
	Fancesco Nuzzolo		
	Mark Post	USTRAT	
	Simon Lacroix	LAAS-CNRS	

DOCUMENT CHANGE RECORD				
Version	Date	Author	Changed Sections / Pages	Reason for Change / RID No
0.1.0	02/10/2017	Shashank Govindaraj	All	Document structure, template and partner assignments
0.2.0	27/10/2017	Fancesco Nuzzolo Mark Post Simon Lacroix	All	Document structure reviewed and updated
0.3.0	20/12/2017	Shashank Govindaraj Fancesco Nuzzolo Mark Post	Chapter 2 and 3	CDFF Deployment approach and basic EGSE descriptions
0.4.0	10/01/2018	Fancesco Nuzzolo Mark Post	Chapter 3	Porting CDFF to RTEMS on Leon and porting it on FPGA
0.5.0	15/01/2018	Simon Lacroix	Chapter 3	Multi-robot EGSE description
1.0.0	23/01/2018	Shashank Govindaraj	All	Final review and finalization
2.0.0	30/03/2018	Jeremi Gancet Shashank Govindaraj	Final check	Addressing PSA RIDs

Executive Summary

This document identifies and addresses the adaptations required for CDFF sub-components required for deploying a set of DFNs or a complete DFPC on a space grade computer (not space qualified) i.e a LEON architecture board with RTEMS as the operating system. A subset of CDFF functions will be selected for being ported to an avionics testbed that is representative of space-grade computing platforms.

The second case addressed is the deployment in a decentralized space grade architecture that is representative of a SPARC Leon3/4 board running RTEMS with an FPGA coprocessor such as the Zync Ultrascale SoC. The aim is to perform a portability check and profiling of the selected functions. The use of an FPGA coprocessor will be used to verify that some appropriate DFNs can be accelerated by use of an FPGA in connection with the rest of the DFPC running on a conventional microprocessor.

The planetary track is the show that the developed DFPCs can be exploited in real time and in realistic conditions. Among the scenarios that have been defined in RD5 (D4.1: Technical trade-offs analysis) in order to define an specify the required DFPCs for the planetary track, the ones which experimental achievement is targeted are - long traverse, Rendez-vous and getting back to base or lander.

Table of Contents

List of of Figures

1 Introduction

1.1 Purpose

This document describes in detail the design of the implementation of the orbital and planetary reference scenarios. It includes the description of the EGSEs and the detailed design of the software, the latter being based on the design of the CDFF as described in D4.2.

The document addresses two reference implementations (i.e. integration and validation tracks): the first one at the consortium level, RI-INFUSE, and the second one at the SRC Space Robotics level, RI-SRC.SR. The objective of RI-INFUSE is to demonstrate and evaluate the full capabilities of the CDFF: from space compliance to state-of-the-art algorithms, from traditional to innovative sensors, and even possibly control in the loop. The objective of RI-SRC.SR is to demonstrate that the CDFF is ready to be integrated with OG1, OG2, OG4 and OG6.

1.2 Structure

This document is structured as follows:

Section 1: This is introductory material.

Section 2: Deployment methods of CDFF on RTEMS, FPGA’s and Multi-robot EGSEs

Section 3: Description of the 3 EGSEs and approach to port CDFF to these architectures

1.3 Applicable documents

AD1	InFuse Grant Agreement
AD2	InFuse Consortium Agreement
AD3	InFuse internal management manual for project partners

1.4 Reference documents

RD1	Description of Action document
RD2	D3.1 Technological Review
RD3	D3.2 System requirements
RD4	D3.3 Early CDFF architecture and ICD

RD5	D4.1 Technical Trade-off Analysis
RD6	D4.2 Advanced CDFF architecture and ICD
RD7	D4.3 CDFF Unitary and integrated test plans
RD8	D4.4 Preliminary design document

1.5 Acronyms

DF: Data Fusion

CDFF: Common Data Fusion Framework

API: Application Program Interface

OOS: On-Orbit Servicing

RCOS: Robot Control Operating System

DFN: Data Fusion Node

DFPC: Data Fusion Process Chain/Compound

DFNCI: Data Fusion Node Common Interface

DPM: Data Product Manager

FPGA: Field-Programmable Gate Array

HDL: Hardware Description Language

HLS: High-Level Synthesis

MW: Middleware

LOS: Line of Sight

Fps: Frames per second

OOS-sim: On Orbit Servicing simulator

OG: Operational Grant

IMU: Intertial Measurement Unit

OT: Orbital Track

PT: Planetary track

OBC: On Board Computer

DEM: Digital Elevation Model

FPGA: Field Programmable Gate Array

2 CDFF deployment requirements

This section describes the required EGSEs for deploying a set of DFNs or a complete DFPC on a space grade computer (not space qualified) i.e a LEON architecture board with RTEMS as the operating system. The objective of this activity is to demonstrate the capability of porting DFNs that are middleware independent onto space grade computation platforms that can enable future space applications on space flight hardware. Utilizing distributed processing (locally) with FPGAs supporting as coprocessors can provide an acceleration of critical and computationally heavy functions. To demonstrate this, one or more DFNs will be selected by doing a trade-off on the complexity of the algorithms and porting effort to port via high level synthesis or VHDL and deploy it on an FPGA coprocessor, supporting the main Leon 4 processor.

2.1 Leon architecture with RTEMS

A subset of CDFF functions will be selected for being ported to an avionics testbed that is representative of space-grade computing platforms. The objectives of this porting experiment are two:

- Perform a portability check, i.e. assess the compatibility of the CDFF software with the limited resources of space processors (computational power, memory size) and with the specific constraints in terms of development environment, operating system, programming languages, available libraries
- Perform profiling of the selected functions, i.e. measure the performance of the selected CDFF modules on a space processor in terms of execution speed and memory consumption.

The target platform for the porting of a subset of the CDFF software is a LEON4 processor. This choice has been based on the fact that one of the most promising high-performance, radiation-hard processors that will be available in Europe in the near future is the GR740, a quad-core LEON4 processor that has been designed as the ESA Next Generation MicroProcessor (NGMP). For spacecraft applications, LEON processors are normally used with the RTEMS operating system, therefore the InFuse architecture EGSE will be a LEON4 + RTEMS platform.

2.2 Decentralized computing architecture

This section describes how CDFF will be deployed in a decentralized space grade architecture that is representative of a SPARC Leon3/4 board running RTEMS with an FPGA co-processor such as the Zync Ultrascale SoC.

An FPGA can be used to accelerate data processing operations by implementing dedicated algorithms in digital logic. However, FPGA acceleration is challenging and intensive in both programming implementation and power consumption per logic cell compared to implementation on a microprocessor. For this reason, only selected algorithms (represented by specific DFNs) will be chosen for coprocessor acceleration.

The use of an FPGA co-processor will be used to verify that some appropriate DFNs can be accelerated by use of an FPGA in connection with the rest of the DFPC running on a conventional microprocessor. As full implementation of DFNs in Verilog/VHDL would require more time and resources than is available for such a functional test, we will instead use high-level synthesis (HLS) tools to convert a subset of C++ code from a DFN representing the function to be accelerated into VHDL, which will then be implemented on an FPGA. To facilitate connection of a microprocessor to the FPGA, we plan to perform this test on a Xilinx Zynq system-on-a-chip device using the Xilinx SDSoC toolchain for HLS and code integration. The degree of compatibility with RTEMS needs further investigation, but an RTEMS board support package is available for the Zynq platform and it is likely that an implementation of RTEMS with HLS support is feasible.

2.3 Integrated planetary track scenarios

The aim of the integrated scenarios for the planetary track is the show that the developed DFPCs can be exploited in real time and in realistic conditions. Among the scenarios that have been defined in RD5 (D4.1: Technical trade-offs analysis) in order to define an specify the required DFPCs for the planetary track, the ones which experimental achievement is targeted are:

- Long traverse
- Rendez-vous
- Getting back

2.4 Multi-robot planetary track scenarios

No actual multi-robot scenario has been defined in RD5, as their achievement would require a decisional layer that yields cooperative decision making and execution coordination. Yet, some DFPCs implement functions that can be exploited for multi-robot collaboration. These are:

- DEM fusion: one robot receives the DEM built by another robot, and fuses it into a larger DEM. This naturally implies the *DEM Building* DFPC
- Inter-robot localisation, which can be achieved by two different means:
    - Direct localisation of one robot with respect to the other, resorting to the *Point Cloud Model-Based Localisation* DFPC
    - Indirect localisation, by registering point-cloud data, DEMs, or point-cloud data with a DEM, resorting to the *LIDAR Map-based Localisation* DFPC.

3 Description of the EGSEs

3.1 LEON architecture with RTEMS

3.1.1 LEON4 processing board

The single board computer used is a GR-LEON4-ITX evaluation board from Gaisler. The board is equipped with a LEON4 SoC implemented in a Structured ASIC from eASIC technologies. The LEON4 SoC has the following features:

- Dual core SPARC V8 integer unit with 7-stage pipeline, 8 register windows, 8 KiB instruction and 4 KiB data caches, hardware multiplier and divider, power-down mode, hardware watchpoints
- non-blocking double precision IEEE-754 floating point unit
- Memory Management Unit
- Multi-processor interrupt controller

The processor core frequency is 200 MHz. As mentioned above, the system has level-1 data and instruction caches but does not feature any level-2 cache.

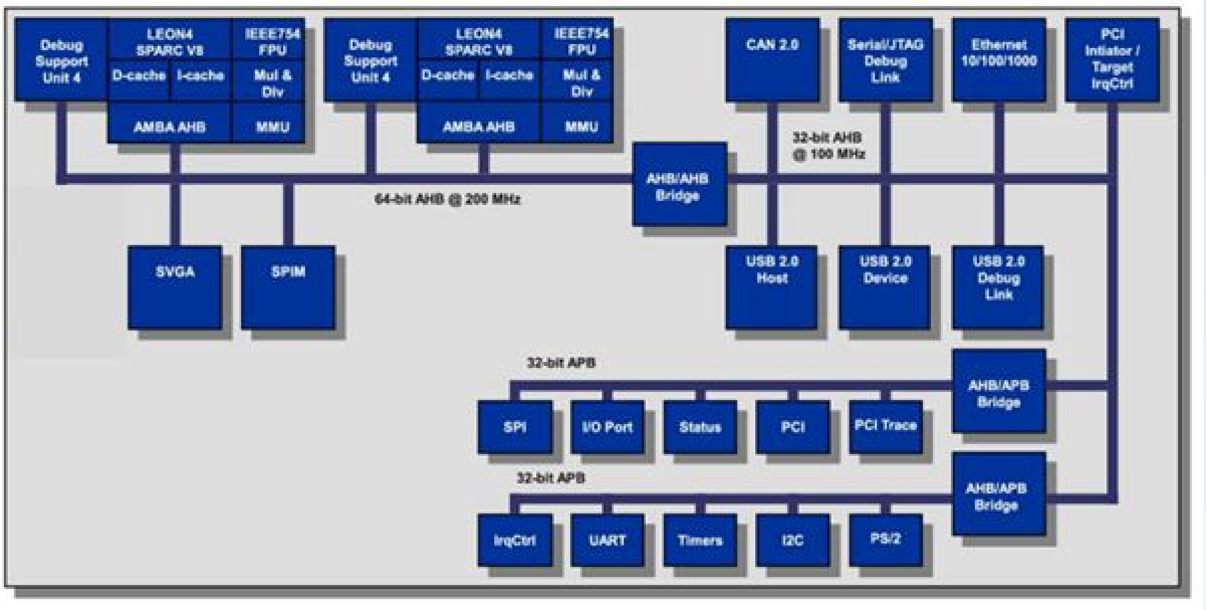

Figure 1: Block diagram of the dual core LEON4 SoC used in the GR-LEON4-ITX board.

The SoC has two AHB buses: the one with the processors and the memory controllers runs at 200 MHz, while the one with most of the peripherals (e.g. USB, Ethernet, PCI) runs at 100 MHz. Low speed peripherals are connected to two APB buses.

On board memories are 256 Mbyte of DDR2 400 SDRAM and 64 Mbit of SPI Serial Flash PROM. The board provides plenty of data interfaces, most notably 2 10/100 Mbit Ethernet interfaces, but also USB, CAN bus, PCI, SPI, I2C, UART.

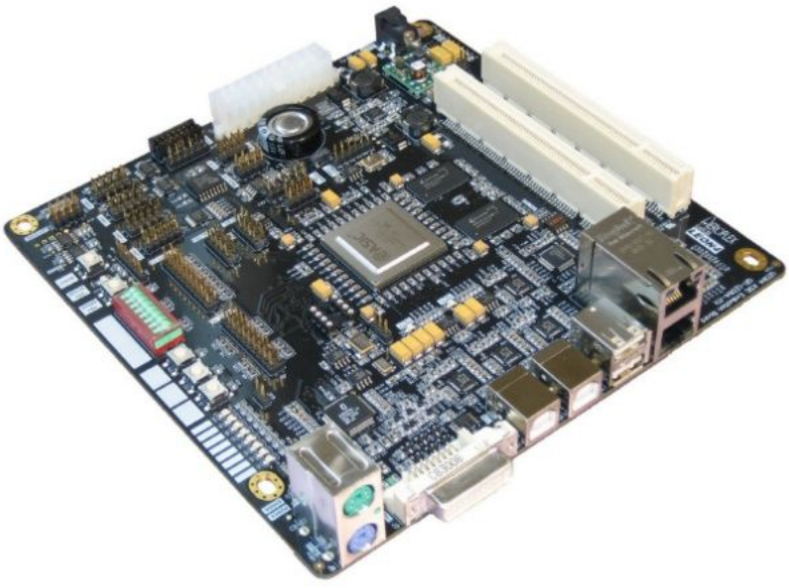

Figure 2: GR-LEON4-ITX evaluation board.

3.1.2 FPGA co-processor with Leon board

The challenge facing embedded processing algorithm implementations in FPGA logic is that procedural image processing code is very difficult to convert into VHDL or Verilog code for implementation into pure logic. Essentially, each operation performed on data must be converted into a logical operation or hard-wired arithmetic logic unit in sequence. This makes even simple floating point vector algorithms very complex in logic implementation and is the main reason that vector processing engines on accelerated graphics cards are most commonly used for computer vision. To overcome this challenge, we make use of High-Level Synthesis (HLS) methods that automate the conversion of procedural code into logical constructs. The toolchain we use for this is the recently released Xilinx SDSoC environment, an Eclipse-based software suite designed to write complete software systems, then move specific algorithms into the Programmable Logic (PL) area of a hybrid System-on-a-Chip (SoC) device with FPGA built in such as the Zynq-7000 series processors, which combine an FPGA with a dual-core ARM Cortex-A9 hard microcontroller. We currently use the AVNet MicroZed Z-7020 board for prototyping and testing algorithms at present due to it's small size, low cost, and accessibility for code development having been in production for several years. This board is shown in Figure 1, and the specifications are as follows:

- Arm Cortex-A9 @667MHz, 1GB SDRAM, 128Mb Flash, 100 I/O

- Artix-7 FPGA (AXI bus), 74K logic cells, 53.2k LUTs, 3.3Mb RAM

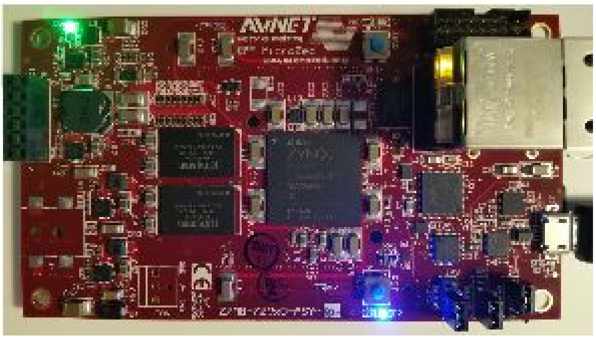

Figure 3: MicroZed board with Zynq-7020

The Xilinx SDSoC environment is used to build boot images on SD card that contain a first-stage bootloader, a Linux kernel, a complete Linux filesystem, ELF-format binaries that implement the software side (un-accelerated) of the application, and a bitstream that represents the hardware (accelerated through HLS) side of the application and is uploaded to the PL automatically on boot-up. After the HLS process, the resulting logic design in a Hardware Description Language (HDL) is synthesized, placed, routed, optimized, and connected to the internal AXI bus for communication by the Vivado software suite. The Linux kernel and filesystem are derived from Xilinx' PetaLinux distribution, which can be easily customized for use on SoC and FPGA based processors. This process is illustrated in Figure 2. The Xilinx reVISION stack of library functions specifically designed for the HLS process in SDSoC is also used to expand the functions that can be moved to the FPGA. The reVISION stack includes a small set of ported OpenCV functions through an API called xfOpenCV, and a set of accelerated neural network implementation functions based on the Caffe framework.

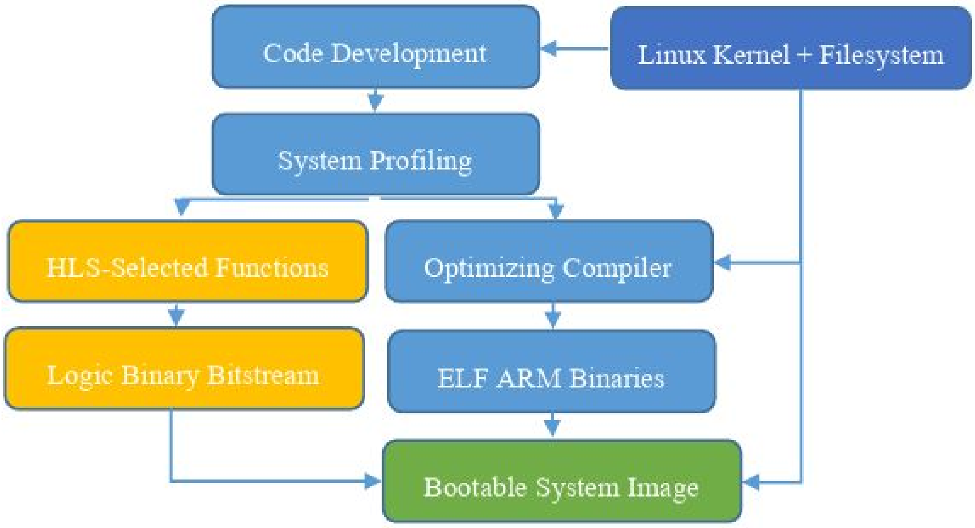

Figure 4: Process for Creating System Image on SD Card

3.1.3 RTEMS and BSP overview

RTEMS is a free open-source real-time operating system for embedded systems. It is the typical choice for the LEON space processor and it has already been used in several ESA missions.

RTEMS applications can be developed in C/C++ using different supported APIs: POSIX 1003.1b, uITRON 3.0 and the so called classic RTEMS API (based on the RTEID/ORKID standard). RTEMS does not provide any form of memory management or processes: in POSIX terminology, it implements a single process, multi-threaded environment. RTEMS' multitasking capabilities are based on priority-based, preemptive scheduling.

A complete SW toolset is available to allow the development of RTEMS C/C++ applications for the LEON4. In the following we briefly introduce the three main tools that are going to be used in the frame of the InFuse project : RCC, GRMON and Mkprom.

RCC, which stands for RTEMS Cross Compiler, is an RTEMS LEON GNU cross compilation system that brings together the following components:

- GCC C/C++ compiler, version 4.4.6
- GNU binary utilities
- RTEMS real-time kernel with network support, version 4.10
- BSP and drivers
- Newlib standalone C-library
- GDB cross-debugger for SPARC

GRMON is a debug monitor for the LEON processor, providing a non-intrusive debug environment on the target hardware. It allows to perform all typical operations for running and debugging an application on the target processor (e.g. read/write access to all LEON registers and memory, downloading and execution of applications, built-in disassembler and trace buffer management, breakpoint and watchpoint management, remote connection to GDB). Typically GRMON will be connected to the LEON4 board using USB or Ethernet ports.

Mkprom is a utility used to create a boot image of an application, which can be stored into the target's flash PROM. Mkprom creates a compressed boot image that will load the application into RAM, initialize various processor registers and finally start the application. GRMON can be used to program the board's flash PROM with the boot image generated by Mkprom.

3.2 Porting CDFF components to RTEMS and FPGAs

3.2.1 Adaptations of DFNs to RTEMS

One of the main difficulties that we envisage in the porting of DFNs and DFPCs to the LEON4 + RTEMS platform is represented by the large dependency of the InFuse SW on third-party image and data processing libraries like OpenCV, PCL, OctoMap, etc. These libraries do not support RTEMS, therefore their source code would have to be compiled in the RTEMS/LEON environment after performing the required adaptations. Considering the size and complexity of these libraries, porting all of them to RTEMS/LEON is not to feasible in the time frame of the InFuse project.

The proposed approach is to limit the porting effort to the OpenCV library, in particular to OpenCV version 1.0. This older OpenCV library version is smaller and simpler than later versions but provides many functions with which a few full DFNs can be built (for example the Harris Feature Extraction). An InFuse DFPC will be identified that contains DFNs that can be implemented using only OpenCV 1.0 function calls or, if this is not possible, that minimizes the dependency on other libraries. Some DFNs that are considered for this porting to LEON are the following:

- Harris Feature Extraction
- Stereo Depth Mapping based on Block Matching
- Image Resize and Filtering

The approach for addressing the case of a required library function that is missing in OpenCV 1.0 will be to extract that function's source code from later OpenCV version (or from other libraries) and include it in the RTEMS/LEON program. The C++ parts that form the DFPC controller and interface code will be taken as they are and, if issues arise with them in the RTEMS/LEON environment, they will be modified appropriately.

The DFPC running on the LEON processor will use a TCP socket to receive input data (e.g. stereo images) and to send output data (e.g. point cloud).

3.2.2 Adaptations of DFNs to FPGA

The process of high-level synthesis of C++ code to HDL is very limited in terms of adaptability; many standard C++ functions and constructs cannot be converted to HDL due to the underlying complexity required. Therefore, the DFNs ported to FPGA fabric are selected by being either 1) composed of very simple and low-level C functions that are found to be portable (e.g. those in math.h), or 2) part of the xfOpenCV API provided with Xilinx' reVISION software stack that is designed for HLS use, detailed in [1].

Due to the limitations above, it is not feasible within the time frame of InFuse to port entire DFNs and run them in FPGA fabric alone without the support of a microcontroller, mainly due to the need to communicate between DFNs. The ARM AMBA AXI Protocol used to communicate between FPGA components is not easily compatible with ASN.1 serialization or generated interface functions, and communication with the DFPC manager and/or an RCOS would require a complex interface to be developed for the FPGA. Instead, the approach we will take is to build the core data fusion algorithms for selected DFNs into discrete functions, move these core functions into FPGA logic with SDSoC while retaining the DFPC controller and interface logic in C++, and compile the entire DFN with SDSoC. Potentially entire DFPCs could be compiled this way as long as functions to be moved to FPGA logic can be partitioned out from the rest of the DFN functionality.

The Data Fusion Nodes that have been selected for full porting of core functionality based on the above criteria are as follows:

- Harris Feature Extraction (implemented using xfOpenCV)
- Stereo Depth Mapping based on Block Matching (implemented using xfOpenCV)
- Image Resize and Filtering (implemented using xfOpenCV)

Additionally, experimental partial porting of some Data Fusion Nodes will be performed to the extent that experimental use of HLS ultimately allows. DFNs that will be test-ported to determine the extent of acceleration possible are as follows:

- Triangulation of features using basic math functions (partial support in HLS and xfOpenCV)
- Feature Matching using basic logic operations (partial support in HLS and xfOpenCV)
- Essential Matrix estimation using RANSAC (previously tested on FPGA but not HLS)

Additional DFN functionality may be ported as the limitations of the current HLS implementation are better established throughout this work, and particular attention will be paid to the acceleration of functions with high processing load. At the moment, it is assumed that all remaining DFNs will operate within the Zynq ARM microcontroller.

Porting custom developed software to RTEMS deployed on a space representative hardware with an FPGA co-processing unit seems to be the path taken by the space industry for upcoming/future space applications. While an RTEMS BSP is available for the Zynq-7020 platform, accelerating functions using HLS from within RTEMS has never been done and constitutes both a high risk of extending development time, and a highly beneficial capability for future development of InFuse. In the interest of ultimately enabling a full embedded link between RTEMS and an FPGA, experiments will be performed to compile RTEMS binaries using SDSoC and moving core functions of DFNs into FPGA logic, in the manner detailed below in section 3.2.2. If success is achieved in compiling and running RTEMS within the time frame of InFuse, a proof-of-concept test of accelerating a simple DFPC while running RTEMS on the Zynq will be conducted. Regardless of whether this test will be conducted, some useful information on the feasibility of using HLS within RTEMS will be obtained.

3.3 Integrated planetary track scenario EGSE

Mana and Minnie from LAAS-CNRS are the robots that will support these scenarios: they are described in detail in the D5.2 EGSE description.

3.4 Planetary track multi-robot EGSE

As stated in section 2.4, no actual multi-robot scenario will be built, but functionalities required by multi-robot scenarios will be demonstrated.

The multi-robot EGSE is composed of the robots Mana and Minnie as they are deployed for the integrated scenarios, and each of the three multi-robot functionalities will be manually triggered by an operator, from the ground control computer.

Triggering this functions is made through requests sent to the DFPCs over the deployed Wifi network. Regarding data exchanges between the two robots, two options are currently considered:

- No direct communication between the robots is exploited. In this case the data to be exchanged transfer through the ground station
- Direct communication between the robots is exploited. In this case the ground control station only send the requests to the involved DFPCs.

At the time of writing this deliverable, the choice between these two options remains pending.

For each of the considered multi-robot functionalities, the data to be exchanged are the following:

- DEM fusion: a part of a DEM built by one robot is transferred to the other
- Direct inter-robot localisation: the relative position between the two robots is send by the robot A, which applied the *Point Cloud Model-Based Localisation* DFPC to the robot B, that has been localized by robot A, and both robots transmit their position estimate to the other.
- Indirect inter-robot localisation: a part of a DEM built by robot A, or a sequence of point clouds acquired by this robot, is transferred to the other robot B, which applies the *LIDAR Map-based Localisation* DFPC. Subsequently, robots A and B exchange their positions.

4 Conclusion

This deliverable so far describes the 3 target EGSEs that are complementary to the EGSEs which will be used for testing and validation of Orbital and Planetary RI of the CDFF. The requirements for deploying a sub-set of DFPCs and DFNs from InFuse CDFF are described for each target computational architecture (ARM/Leon4/FPGA) with the supporting operating system (Linux/RTEMS). The deployment of DFNs on FPGAs for accelerating processing - either entire DFN or a specific computationally expensive function within a DFN has been illustrated. After doing a trade off on the availability of supporting libraries between the Leon4 (RTEMS) and the ARM-FPGA coprocessing board, a set of DFNs that is likely to be ported to both the Zync and Leon boards are listed. The multi-robot EGSE consists of the 2 rovers from LAAS that will deploy DFPCs for collaborative mapping (DEM fusion) and localization (direct and indirect). The details of the rovers are well described in D5.2 and not repeated in this deliverable to avoid duplication of content.

5 References

[1]https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug1233-xilinx-opencv-user-guide.pdf