Deliverable D2.4

Second Updated Application Specifications
2022-10-31

Work package:	WP2 UPDATED SPECIFICATIONS AND EVALUATION CRITERIA	
Editor:	Enginsoft	Gino Perna
Author(s):	M3E	Giovanni Isotton
	Cyberbotics	Olivier Michel
	ICCS	Panagiotis Miliadis
	ICCS	Chloi Alverti
	ICCS	Dimitris Theodoropoulos
	ICCS	Dionisios Pnvematikatos
	Fraunhofer ITWM	Jens Bender
	Fraunhofer ITWM	Sebastian Fett
	Fraunhofer ITWM	Elisa Thiel
Reviewer #1	TSI	Pavlos Malakonakis
Reviewer #2	M3E	Giovanni Isotton
Dissemination Level	Public	
Nature	Report	

Confidentiality

This project has received funding from the European High-Performance Computing Joint Undertaking Joint Undertaking (JU) under grant agreement No 955739. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Greece, Germany, Italy, Netherlands,Spain,Switzerland.

Date	Author(s)	Comments	Version	Status
22-10-19	Gino Perna	Initial version	0.1	
22-11-02	Gino Perna	Pre-release version	0.9	
22-11-07	Pavlos Malakonakis	Review		
22-11-07	Giovanni Isotton	Review		
22-11-07	Gino Perna	Final Version	1.0	

Executive Summary

This document is the updated version of D2.1 and the corresponding Matching OPTIMA platforms of D2.2. The document presents the advanced status of applications in the project, together with some evaluations on how those applications will benefit from the accelerated environment proposed and updated in OPTIMA, increasing the efficiency, and how open source libraries porting, also described here, will give a baseline for application optimization, lesson learned considerations and review of the OPTIMA platform due in WP5.

Applications are playing a central role in the OPTIMA project as they have been hardly used to experiment and implement different hardware accelerators and environments in WP3. In some cases there have been criticalities in order to tightly fit the corresponding libraries/toolchains with the infrastructure, solved during WP3 and have been described in D3.X and D6.1.

The applications presented here are state of the art of HPC computing in different disciplines: Robotics, Underground simulations, CFD simulations and Continuum Mechanic processes.
The porting strategy is herein defined by specifying which part of the application itself will be further optimized in order to take the most of the OPTIMA platforms.

All the applications presented here have been re-engineered and ported to the corresponding platforms, with updated requirements, distinctive elements and specifications described.
The first porting, difficulties and evaluation, identified bottlenecks resulting from D3.X and this deliverable have been presented in order to describe the optimization to be performed in WP5.

Table of Contents

List of Abbreviations

HPC	High Performance Computing
FPGA	Field Programmable Gate Array
CUDA	Compute Unified Device Architecture
OpenGL	Open Graphics Library
GPU	Graphics Processing Units
MPI	Message Passing Interface
OpenMP	Open Multi-Processing.
CP	CheckPoint
NUMA	Non-Uniform Memory Access
NVM	Non-Volatile Memory
OS	Operating System
LBM	Lattice-Boltzmann method
DNN	Deep Neural Network
SP	Single Precision
DP	Double Precision

1. Introduction

As the result of WP2 and WP3 (1st Application Evaluation), this deliverable will describe the second updated specifications of the applications and the open-source libraries in the OPTIMA project.

1.1 Motivation

The exact specifications of the applications and the open-source libraries that will be ported to the different OPTIMA systems are specified herein.
In order to concentrate all the efforts in the right way, special care has been taken so as to adequately give comprehensive visibility of the different OPTIMA systems and describe here the critical parts of the applications so as to make sure that they will take full advantage of the FPGA-based HPC systems.
Specifications described in this deliverable have been updated based on the result of the first evaluation done in WP3. ES will be the leading partner while all the application partners will work on their specific applications.

2. Technical Applications

2.1 Robot Simulation

[Contributed by Cyberbotics Sàrl]

2.1.1 Description

The robotics simulation will rely on the Webots open-source robot simulation software developed by Cyberbotics [1.2][1.3]. These simulations include the generation of simulated sensor data, the actuators dynamics and the environment dynamics, in order to close the control loop. The same robotics simulation scenarios may be run millions of times together with Machine Learning algorithms (for example evolutionary algorithms) to train intelligent robot controllers.

The demonstrators developed by Cyberbotics will include a series of scenarios based on autonomous car driving using machine learning on camera images (multiple layer perceptron) to achieve lane following in a simulated road network environment. Preliminary experiments with a HPC system without GPU demonstrated that the bottleneck was the simulation of the sensors. The new version should benefit from both the GPU acceleration for the sensor simulation and the FPGA acceleration for the machine learning robot controller. To achieve this goal, the simulation system will be split on two machines. On one hand, a HPC machine equipped with GPUs will run the simulation physics, including sensors, motors and interactions with the environment. On the other hand a HPC machine equipped with FPGAs will run the robot controller algorithm for processing image data with a multiple layer perceptron and send motor commands to the simulated model of the robot. The communication between the two HPC machines will go through a specially developed communication mechanism involving TCP and possibly UDP network communications to send the image flow between the two systems, as fast as possible. The performance bottlenecks will be identified, analyzed and if possible further optimized to reach a maximum performance.

2.1.2 Technical details

The Webots software is developed in C++ and makes use of OpenGL hardware acceleration [1.1]. The simulation process is itself compute intensive: the rigid body physics and various sensors and actuators simulations typically run on multiple CPU cores while the sensor rendering (cameras, lidars, range-finders, etc.) runs on GPUs. Usually, several thousands of simulation instances are running in parallel on compute clusters and are combined together by an evolutionary algorithm that gathers results and sets up the next evolutionary iterations. The Artificial Intelligence (AI) of the simulated robotics systems is also usually implemented using computer intensive algorithms, involving sensor data processing, image processing, deep learning, neural networks, and other compute intensive algorithms. Such robotics control algorithms are generally very well suited to run on large FPGA systems, because of their parallel nature.

The simulation is split in several processes: on one hand, the simulation (webots) process and on the other hand the controller processes. Each controller process controls one robot which can be a simulation supervisor. The controller processes make extensive use of machine learning. The supervisor process is responsible for the orchestration of the machine learning algorithm and gathering of results. The controllers may be written in C, C++, Python, Java or MATLAB. However, within the OPTIMA project, we will focus mainly on C/C++ and Python controllers as they are likely to provide the best interfaces with FPGA libraries.

2.1.3 Libraries used

Webots relies mainly on the following C shared libraries: OpenGL 3.3, Qt 5, ODE (open dynamics engine), OpenAL.

The Webots controllers can make use of various deep learning libraries such as: Tensor Flow, Pytorch, Keras, Theano, OpenCV, etc. This doesn't exclude custom-made libraries running on the top of CUDA/OpenCL. The Webots controllers could also be interfaced with CPU, GPU or FPGA libraries. This will allow us to compare the performance of various implementations.

2.1.4 License

Webots is released under the open-source Apache 2.0 license. All the simulations, including robot models, environment models, robot controllers and supervisor controllers will be released under the same open-source Apache 2.0 license, unless they rely on proprietary material owned by customers of Cyberbotics.

All the machine libraries used together with Webots are open-source libraries with various licenses. Any new deep learning library for robotics developed during the OPTIMA project by Cyberbotics will be released as open source software under the terms of the Apache 2.0 license.

2.1.5 Interest for the scientific community

Developing an open-source toolset of libraries for deploying and running deep learning algorithms for robotics simulations will certainly draw the interest of the robotics research community which is very active in the deep learning area. Researchers will be able to adapt their research tools to leverage on these libraries to achieve better results by exploring deeper their algorithms thanks to the available HPC power.

2.1.6 Arithmetic considerations

The Webots physics engine relies on IEEE double precision arithmetics while the rendering library (WREN), running on the top of OpenGL relies mainly on IEEE single precision arithmetics. Controller programs rely on single, double or mixed precision depending on the libraries in use. Custom GPU and FPGA implementations may be developed to take advantage of performance boost using minimal precision.

2.2 Underground analysis

[contributed by M3E]

2.2.1 Description

The Underground analysis mainly consists of simulations of the withdrawal/injection of subsurface resources (water, CO2, oil, gas, etc.) to understand and evaluate the impact on the environment. The main applications are: (1) the evaluation of the induced and/or triggered seismicity due to fault reactivation and/or fracture generation caused by reservoir activities; (2) the prediction of the flow field in variably-saturated or multi-aquifer systems both at the well and at the basin scale; (3) the evaluation of the aquifer artificial recharge by superficial basins and wells.
M3E has developed property state-of-the-art Finite Element (FE) software simulators for underground analysis [2.1][Janna et al., 2020]: ATLAS and GWS, for geomechanics and groundwater, respectively.

When it comes to the solution of extreme-size FE models, it is well known that, for large models, the most time-consuming task (up to 95% of the total computational time) consists of the solution of the arising linear systems of equations. Both ATLAS and GWS take full advantage of Chronos, a proprietary collection of sparse linear algebra kernels designed for HPC [2.2][Janna et al., 2020]. The library implements best-in-class preconditioners to incredibly accelerate the convergence, and can solve systems with hundreds of millions (or even billions) of unknowns.

Within the OPTIMA project, we will focus on Chronos. In particular, the main kernels for the preconditioner computation and the linear solver application will be ported and optimized to the FPGA-based HPC systems.

2.2.2 Technical details

Chronos is specifically designed for the most modern High-Performance Computing and has been thought to be ready for the next generation Exascale systems.

Mainly written in C++, Chronos uses OpenMP directives for shared memory processing and CUDA for the use of GPU accelerators. Interprocessor communication for distributed computation is accomplished through the MPI protocol.

Within the OPTIMA project, we will develop the CPU-version of Chronos in order to create a CPU-FPGA version.

2.2.3 Libraries used

The user needs the following packages to compile and run Chronos:

- cmake (compatible versions from 3.10.2 to 3.20.0);
- gnu toolchain (compatible versions from 6.1.0 to 9.3.0);
- OpenMP (compatible version 4.5);
- MPI (compatible version 3.1);
- lapack (compatible version 3.8.0);
- parmetis (compatible version 4.0.3);

None of these packages will be developed within the OPTIMA project, they are only auxiliary libraries. Communications libraries of the specific target platform will be used, assuming they are fully optimized for the specific platform itself

2.2.4 License

The Chronos Library is freely accessible to research institutions. Chronos is not open source, but some portions of code can be shared on request, for research purposes only.

2.2.5 Interest for the scientific community

The solution of linear systems of equations is a central problem in a huge number of applications in both engineering and science.

Within the OPTIMA project, the Chronos library for the solution of large and sparse linear algebra problems will be optimized for FPGA-based HPC systems.

Chronos is freely accessible to research institutions, so researchers will be able to interface Chronos to their simulators, evaluate the potential and enhance their computational performance.

2.2.6 Arithmetic considerations

In the current CPU-version of Chronos, the main kernels for the preconditioner computation and linear solver application rely on double precision arithmetic.

We plan to take advantage of the FPGA single precision for the preconditioner kernels while using the FPGA double precision for the linear solver kernels.

2.3 MESHFREE

[contributed by Fraunhofer ITWM]

2.3.1 Description

The simulation software MESHFREE has been developed since 1999 at the Fraunhofer ITWM [3.1] with regards to industrial applications. It is a meshfree approach to simulate flows and continuum mechanic processes. By not using a numeric meshing the method is versatile and efficient regarding simulations with moving boundary elements, free surfaces, boundary curves or fluid-structure-interaction.

Since 2018 the software has been extended by the solver library SAMG which is developed by the Fraunhofer SCAI.

2.3.2 Technical details

To allow for continuum mechanical simulation without a predefined mesh, MESHFREE uses moving point clouds on which generalized finite differences are constructed. Coefficients of the finite difference stencils are computed based on neighboring points and local least squares minimization procedures. Since points in MESHFREE solely act as a numerical discretization of the domain covered by the simulated material, they can be added and deleted to maintain point cloud quality with respect to stencil calculation. In this aspect, it is also different from some other meshfree methods like the Smoothed-particle hydrodynamics (SPH) or the Material point method (MPM). Both rely on mass points that cannot be easily added or deleted since they can be thought of as representing “chunks of material”.

On the basis of this finite difference formulation, a range of solvers for different continuum mechanical problems are realized in MESHFREE. Most prominently, an implicit solver for incompressible or weakly compressible flows as well as an explicit solver for compressible flows are available. The implicit solver is the most feature rich solver in MESHFREE and is based on a fractional step scheme. Other solvers include a simplified model for the calculation of thin films and a DEM solver for particle interactions.

The incompressible solver requires the solution of very large systems of equations in every computed timestep. These systems are solved iteratively by a BiCGSTAB algorithm.

On the technical side, MESHFREE is written in Fortran and almost exclusively in a procedural programming style. Standard compilations are only done under Linux (Currently RedHat7) and with the Intel Fortran and MPI compilers. Experimental compilation has now been done for gfortran (Update November 2022).

All data associated with points (position, velocity, etc.) is held in memory in one large real array, where each column represents a point. Additionally, for each point a list of neighboring points is maintained and held in memory. MPI-Parallelization is done by a domain decomposition approach, where the large point cloud array is distributed on the

available processes. In this decomposition, halo regions are based on the local size of neighborhoods for the stencil calculation.

One strength of MESHFREE is the efficient handling of large geometries. This is made possible by a shared memory approach, where the geometry has to be stored and handled only within one memory domain per node or NUMA domain.

2.3.3 Libraries used

No external libraries are used for tasks relevant for this project.

2.3.4 License

Currently, MESHFREE can be obtained in the following license policy:
A commercial license is offered for industrial users. Single academic users can obtain a free license. At the end of this project, we plan extended test licenses with FPGA ported versions for all users.

2.3.5 Interest for the scientific community

Within the scientific community there are also other groups developing meshless methods (SPH, Material Point Methods) and CFD codes. They might have different underlying mathematics, but share some organizational tasks. We plan to publish on the porting ideas in order to make these ideas available to a broad scientific audience [3.2].

Additionally, we make the ported code partially publicly available.

Free licenses are available for single academic users.

2.3.6 Arithmetic considerations

Currently the complete code is written in double precision. But certain tasks are feasible for single precision.

We currently object that the solution of our linear systems of equations is not feasible for single precision arithmetic.

2.4 LatticeBoltzmann-CFD

[contributed by Enginsoft]

2.4.1 Description

LatticeBoltzmannCFD is a free computational fluid dynamics solver based on the Lattice Boltzmann method and optimized for modern multi-core systems, especially GPUs (Graphics Processing Units) and possibly FPGAs.

The solver is based on the Lattice Boltzmann Method, which is conceptually quite simple to understand and which scales very well with increasing computational resources.

As for the first Application description, the selected code was implemented in Python with some specific implementations for CUDA. The list of requirements was huge but did, in principle, match the distribution set for JuMax. During the first implementation, difficulties arose with respect to the correct toolchain and libraries to be installed, that were not overseen before, mainly due to the strict structure of the host and required libraries/toolchains of the system that should be multiuser and could not be changed by the user without affecting the whole system.

In order to bypass this issue, a new solver was chosen, not requesting such tight dependencies and with a structure very similar to the original code.

The general goals addressed by the solver are as follows:

- Scalability: the code is designed to scale well with an increasing number of compute cores. This is particularly true for both the code engineerization and the type of equations used.
- Maintainability: the code is clean and easy to understand.
- Ease of use: defining new simulations and exporting simulation results is simple and can be automated with scripts.

Which are almost the same as the first original solver.

2.4.2 Technical details

The lattice Boltzmann equation (LBE) is a minimal form of Boltzmann kinetic equation which is meant to simulate the dynamic behavior of fluid flows without directly solving the equations of continuum fluid mechanics. Instead, macroscopic fluid dynamics emerges from the underlying dynamics of a fictitious ensemble of particles, whose motion and interactions are confined to a regular space-time lattice. Technically, the distinctive feature of LBE is a dramatic reduction of the degrees of freedom associated with the velocity space. In fact, particle velocities are restricted to a handful of discrete values $\vec{v}i=\vec{c}$ i, i=0,b, by assuming that at each site the particles can only move along a finite number of directions. By endowing this set with sufficient symmetries to fulfill the basic conservation laws of mass, momentum and energy, the LBE can be shown to quantitatively reproduce

the equations of motion of continuum fluid mechanics, in the limit of long wavelengths as compared to the lattice scale.

The distinctive features of LBE as a computational solver for fluid problems are its space-time locality, and the fact that information travels along the straight lines defined by the (constant) particle velocities associated with the lattice, rather than along the space-time dependent material lines defined by the flow speed. Due to these properties, the LB approach counts today an impressive array of applications across virtually all fields of fluid dynamics and allied disciplines, such as biology and material science.

Some generic code capabilities, intended as the OPTIMA project interest follows:

- runs on a single processor or in parallel on pure CPU nodes

- open-source distribution
- developed in C/C++ with almost no dependencies
- easy to extend with new features and functionality
- runs from an input script

2.4.3 Libraries used

The list of used libraries is:

- openmpi/openmp for thread communications (not binded to a specific version)
- metis/parmetis (just for partitioning)(not binded to specific version)

2.4.4 License

The application is licensed under the LGPL v3.

2.4.5 Interest for the scientific community

Advanced strategies for the efficient implementation of computationally intensive numerical methods have a strong interest for the industry.
Nowadays the Lattice-Boltzmann method (LBM) is one of the most popular methodologies in CFD, and its use is extended to a high number of different CFD applications. Furthermore, the advantage of using LBM has been consistently confirmed by many authors [4.1][4.2][4.3][4.4] for a large variety of problems and computing platforms.

There are some obvious advantages of LBM approach:

- Intrinsic linear scalability in parallel computing, because the collisions are calculated locally.
- Geometric complexity is not a challenge. This includes the solid moving and domain deformation.
- Efficient inter-phase interaction handling for multiphase flow because phase interaction is inherently included in the particle collisions.

- Accuracy not different from other methods
- Inherently unsteady (transient) simulations

And of course disadvantages:

- Memory requirement for large problems
- Computational intensity
- Turbulence model

The technology is promising for specific problems where traditional methods need too much discretization and thus the number of equations to be solved.

2.4.6 Arithmetic and memory considerations

LBE CFD is an accelerated general purpose CFD code that can take advantage of the underlying hardware.
The arithmetic used is the same as traditional CFD codes: SP or DP, with a preference for the first by assuming the user will know its limits (basically stagnation points and magnitudes of velocities).
Despite the ease of the method, which can be efficiently parallelized, it needs a considerable memory capacity, which is the consequence of a dramatic fall in performance when dealing with large simulations. Those aspects will be investigated through the project.

2.4.7 Implementation approach

The code is written in standard C/C++, with minimum requirements.
The entire domain can be spread out on several CPUs and possibly FPGAs via a thread model that is responsible for the boundary interactions between the partitions. The current approach did show impressive performance gains in the computational part, completely counterbalanced by the time needed by the CPU to prepare data and copy memory structures. Extensive modifications have been done on the code in order to provide a monolithic kernel for solving the equations, by also monitoring and profiling the entire code.
The subsequent optimization part of the project will be dedicated to implementing thread calculation in order to reduce the bottleneck of the CPU and provide multi FPGAs capabilities.

3. Open Source Libraries

3.1 Specification updates

Based on the results obtained from “D6.1 - 1st Evaluation Results”, this section lists a set of updates on the OOPS library specifications:

- The OOPS library will be compatible with the latest U55c Alveo card from Xilinx, which will be used at the final OPTIMA hardware platform.
- BLAS L2 kernels that require a packed matrix representation (tpmv, spmv) will support a row-major format.
- BLAS L1, L2 and L3 kernels will support single-precision floating data format.
- The Sparse Matrix - Vector (SpMV) kernel will support double-precision data format.
- Computer-Aided Engineering (CAE) solvers (as stated in Table 2, see next) will support single-precision floating data format.

Following sections 3.2 - 3.6 provide the original library description that was reported in D2.1, included in this deliverable for convenience.

3.2 Description

Scientific software, as well as industrial applications, is based on calculations such as vector operations, linear / differential equations, and matrix multiplications. Consequently, towards enhancing performance, OPTIMA has set Objective 3, which is the development of the **O**ptima **OP**en **S**ource (OOPS) library. In a nutshell, OOPS will be an optimized set of software routines that may be used by industrial / scientific software and applications, which will take advantage of the OPTIMA hardware platforms. OOPS will drastically reduce the effort of mapping primitive computation kernels onto reconfigurable logic that is integrated in the OPTIMA hardware platforms, and improve the execution time and energy efficiency of the mapped computations. OOPS will be integrated into the OPTIMA toolflow and programming environment, enabling seamless utilization of the available hardware resources by software developers, without requiring advanced skills or extensive experience in hardware development.

The rest of this chapter will focus on

- presenting currently available libraries developed and maintained both by academic / scientific institutions and hardware vendors;
- listing software toolflow / dependencies for known current implementations
- discussing arithmetic representation for the candidate libraries and supported precision formats;
- finalizing the OOPS kernel list that will be mapped onto the OPTIMA hardware platforms, exposed to application developers over a simple API.

3.3 Related Work - technical details

High performance computing (HPC) applications from various scientific domains commonly use similar computational building blocks (e.g., matrix operations), share computational and data access patterns (e.g., stencils) and sometimes share even common algorithms/methods (e.g., partial differential equation solvers [PDEs]). In this section, we discuss some of these fundamental computational modules and the state-of-the-art and state-of-practice of their acceleration on FPGAs (shown in Table 1).

Table 1 - Existing and related attempts of mapping/accelerating HPC kernels on curent FPGAs

Category	Kernels	Publicly available implementations	Platforms	Toolchain	Data Precision
Blas	L1 (vector-vector): asum, axpy, dot, imax, nrm2, rot, rotm, scal,swap L2 (vector-matrix): gemv, ger, syr L3 (matrix-matrix): gemm, gemm systolic, sr2k, syrk, trsm	Xilinx Vitis Blas Library [5.2]	Xilinx Alveo and VU9	Xilinx Vitis	SP and short
		FBLAS [5.1]	Intel Stratix 10/Arria 10	Intel FPGA OpenCL SDK	SP and DP
Patterns	2D/3D Stencil Computations	Stencil [5.7]	Xilinx UltraScale+	Xilinx Vitis	SP, HP and FP
		Stencil Stream	Stratix 10 GX/SX 2800	Intel OneAPI	SP
		Stencil Optimizations [5.9]	Intel Stratix V/Arria 10	Intel FPGA OpenCL SDK	SP
Methods	LUD, CFD, 1D/2D/3D FFT	Rodinia [5.9]	Intel Stratix V	Intel FPGA OpenCL SDK	SP
		OpenDwarfs [5.10]	Altera FPGAs	Intel FPGA OpenCL SDK[ΠΜ1]	SP
		HPCC_FPGA [5.5]	Xilinx [Ultrascale+] / Intel [Stratix 10]	Xilinx Vitis Intel FPGA OpenCL SDK	SP
		FFT3D-FPGA [5.12]	Intel Arria 10/Stratix 10	Intel FPGA OpenCL SDK	SP

		Rodinia-HLS [5.11]	Xilinx's Alveo U200	Xilinx Vitis	SP

BLAS [5.17]. The Basic linear Algebra Subprograms (BLAS) are routines that provide standard computational building blocks for basic matrix and vector operations. The first level of BLAS (L1) routines perform scalar, vector and vector-vector operations. The L2 performs vector-matrix operations, while L3 performs matrix-matrix operations. BLAS kernels are efficient, portable and widely used in the development of high-quality linear algebra software. Both academia and industry has been involved in the development of high performance BLAS kernels targeting FPGA platforms.

State-of-the-art FBLAS [5.1] is an open-source implementation with multiple optimized L1/L2/L3 kernels (Table 1) including, among others, specialized matrix routines (triangular and symmetric matrices). FBLAS is developed with Intel FPGA OpenCL SDK targeting Intel platforms and evaluated on Arria 10 and Stratix 10. State-of-practice Vitis Libraries [5.2], Xilinx's official open-source libraries, include a BLAS implementation that targets the Alveo cards. Their approach for BLAS organization differs the usual approach of computational patterns, as the L1 consists of the low-level hardware primitives of routines, L2 includes all L1 operations to form a single blas kernel, and L3 consists of the host executable code and the hardware L2 kernel. It is implemented using OpenCL API for the host executable code, and augmented C++ with HLS primitives for the hardware kernel. It supports fewer matrix-vector and matrix-matrix kernels compared to FBLAS. Similar works from academia, such as Spector [5.3] and OpenDwarfs [5.4], implement some BLAS kernels like gemm, while HPCC FPGA [5.5] includes various benchmarks based on the same kernels.

Computational patterns/Stencils. Stencil computations are a class of algorithms which update elements in a multidimensional grid based on neighboring values using a fixed pattern - the stencil. Stencils are an important computation pattern in HPC, used for solving differential equations, weather, seismic and fluid simulations, and convolution neural networks. Stencil codes are a popular target for FPGA acceleration in HPC, due to their regular access pattern and temporal locality. Zohouri et.al. [5.6] implement 2D and 3D stencils for Intel's platforms, taking advantage also of spatial locality to enable large input sizes beyond a platform's on-chip memory. Litch et.al. [5.7] provide optimized stencil hls code for Xilinx's Ultrascale+ FPGAs and StencilStream [5.8] is a generic open-source stencil simulation library for FPGAs developed by Paderborn University targeting Intel's platforms using Intel's OneAPI and C++.

Methods. Apart from basic computational blocks, there are also complex scientific kernels that appear frequently in HPC and are actively developed for hardware accelerators.

Lower–upper decomposition or factorization (LU) is an algorithm to calculate the solutions of a set of linear equations. LUD is a common dense linear algebra benchmark and Zohouri et.al [9] and Krommydas et.al. [5.10] provide FPGA implementations for Intel platforms, while Cong et.al [11] for Xilinx's Alveo U200 card.

Fast Fourier Transformations (FFT) is a fundamental building block for digital signal processing. Meyer et.al. [5.5] implement a 1D FFT for Xilinx and Intel platforms while Ramaswami et.al. [5.12] implement high-level description kernels from one to three dimensions for Intel targets. The latter library is already used from industry as well, as it is used from CP2K as a part for their quantum chemistry package.

Computational fluid dynamics (CFD) is an engineering analysis method with direct application to many thermal (heat transfer) and fluid (flow behavior and regime) models and simulations. Common CFD solvers typically use Navier Stokes or Lattice Boltzmann analysis methods, requiring matrix computations (matrix-matrix multiplication, matrix vector product) with both dense and sparse datasets.

A wide range of industries such as automotive, biomedical and aerospace rely on fast CFD analysis. OpenFOAM [5.13-5.14] is an open-source library written in C++ that features a wide range of solvers employed in CFD. It uses MPI to provide parallel multi-processor functionality and there are no official open-source versions for hardware accelerators (FPGAs). [5.9], [5.10] and [5.11] develop an FPGA accelerated CFD solver for the three-dimensional Euler equations for compressible flow, as part of the Rodinia benchmark suite (Intel and Xilinx platforms). ByteLake enterprise [5.15] provides a proprietary library [5.16] with optimized CFD kernels for Xilinx's Alveo card. PETSc [5.19] is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It supports MPI, and GPUs through CUDA or OpenCL, as well as hybrid MPI-GPU parallelism. PETSc is intended for use in large-scale application projects and includes a large suite of parallel linear, nonlinear equation solvers and ODE integrators that are easily used in application codes written in C, C++, Fortran and Python.

3.4 Toolchain and dependencies

High-Level Synthesis (HLS) is an automated design process that imports a design of a desired behavior in a high-level description language (C++) augmented with HLS primitives, and creates the equivalent Register Transfer Level (RTL) design in a hardware descriptive language (Verilog, VHDL). Vitis is a software development platform from Xilinx that enables the acceleration of applications on heterogeneous hardware platforms (CPU + FPGA). OpenCL API is used for the host executable code, a C++ like language capable of managing the system's heterogeneity. In order to use Vitis, a license is needed which can be purchased from Xilinx. Furthermore, Vitis is mainly used to target the new state-of-the-art Alveo Datacenter Acceleration Cards, but QFDBs as well, that are the main target platform for the Optima project.

3.5 Arithmetic representation and precision

One of the main concerns of the OOPS library is data precision, a parameter which will strongly affect both the performance of the library and the estimated resources (DSP, LUTs, BRAM etc) of the kernel in a FPGA. While most HPC kernels and arithmetic libraries in general purpose processors implement both single and double floating point data precision, the use of the latter option will have several drawbacks. As seen on Table 1, all kernels are initially implemented in single precision, a data type which will benefit the most from

FPGAs. The adoption of double precision might be necessary for some kernels like aerospace or seismic and fluid simulation, but it is not recommended, as it will negatively affect the estimated resources. FBLAS is the only open-source library that implements double precision, but it reports the estimated resources only for the single precision subroutines. We expect at least a 2x (and possibly up to 4x) resource overhead with the adoption of double precision. Furthermore, the performance gains might be worse, as the DSPs in state-of-the-art FPGAs are designed for single and lower precision kernels.

On the other hand, there is an approach from the community to minimize the precision of data types, to either half precision or fixed point. The use of lower precisions is strongly recommended, as the resource consumption of such kernels is minimized, while more data can be fetched each second into kernels. The majority of kernels are implemented with the adoption of streaming interfaces where data are imported, modified and exported from the kernels. Nevertheless, such approaches must be carefully implemented. Most kernels are designed to utilize high precision data types, and the conversion to an equivalent lower precision can make the kernel utilization in a scientific application domain not feasible.

3.6 OOPS library and kernels

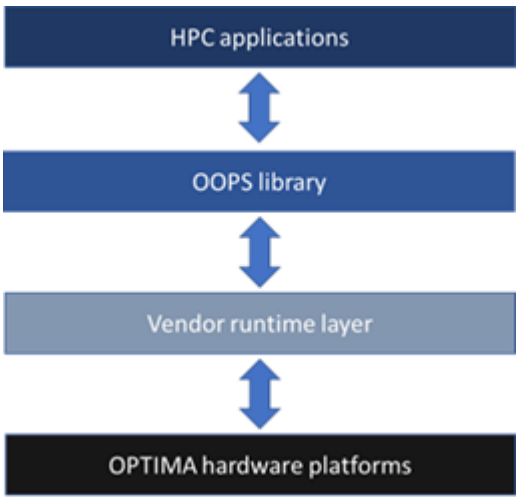

Figure 1 - OOPS intercommunication with the OPTIMA toolflow.

Figure 1 illustrates how the OOPS library will be integrated to the OPTIMA toolflow. OOPS will expose an API in the form of function prototypes towards the application layer, and target the OPTIMA hardware platforms. Developers will be able to utilize OOPS kernels by including a small set of files in the application source code. On the other hand, OOPS kernels will leverage the device vendor runtime layer to

- transfer data from the host processor to hardware kernels;
- initiate and monitor data processing;
- send output results back to the application layer.

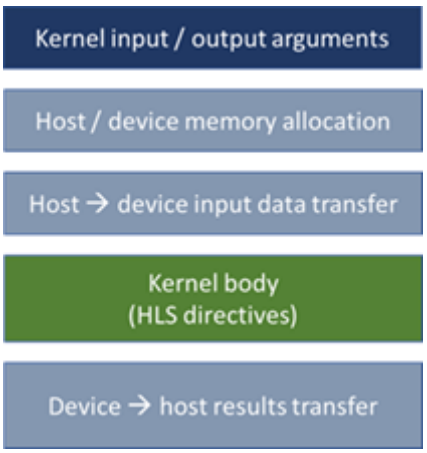

Figure 2 - OOPS kernel internal structure.

Figure 2 illustrates the general structure of OOPS kernels implemented in the OPTIMA platforms. Kernels will expose a set of input / output arguments to developers. Before initiating data processing, each kernel will utilize vendor-specific APIs to allocate memory for all input / output arguments in the FPGA, as well as transfer data between the host processor and the attached FPGA device. Towards optimizing performance and resource utilization, all OOPS kernels will include vendor-specific High-Level Synthesis (HLS) directives. These directives allow CAD synthesis and implementation tools to identify and unroll for-loops, exploit data parallelism, set a target pipelining interval (e.g., interval of 1 cc is fully pipelined that usually requires more resources than larger intervals, however produces an output every one clock cycle), as well as optimize data transfer mechanisms (e.g., using multiple DMA channels and / or double buffering).

Table 2 - The OOPS library set to be implemented onto the OPTIMA hardware platform

Category	**Kernels**	
BLAS	L1	ROT, ROTM, SWAP, SCAL, COPY, AXPY, DOT, DSDOT, NRM2, ASUM, IAMAX, AMIN, AMAX
	L2	GEMV, GBMV, SYMV, SBMV, SPMV, TRMV, TBMV, TPMV, TRSV, TBSV, TPSV
	L3	GEMM, SYMM, TRMM, TRSM
Sparse Linear Algebra	SpMV	
General Computer-Aided Engineering (CAE) solvers	PETSc	Preconditioners: Jacobi (DOT, SpMV, SCAL, COPY) Direct solvers: LU factorization / decomposition Krylov: CG

Table 2 lists the set of OOPS library functions that will be mapped and optimized on the OPTIMA platform. With respect to the original BLAS library [5.17], OOPS will support most BLAS subroutines to enable fast and accurate vector-vector, vector-matrix and

matrix-matrix operations. The majority of the L1 subroutines are core operations for linear solvers for matrices such as Conjugate Gradient and Jacobi, solvers that we will discuss later. The importance of Level-1 Blas Library leads to prioritizing the implementation of these subroutines on FPGA platforms. Nevertheless, many scientific applications domains are dominated by matrix-vector and/or matrix-matrix multiplications. GEMV and GEMM kernels have been already implemented on similar works as discussed on related work. However, BLAS library also includes calculations with special types of matrices such as symmetric or triangular. While the classic kernels could solve any type of matrices, the implementation of specialized kernels is necessary, as the performance boost and benefits from FPGAs would be even better.

OOPS will also support sparse matrix vector multiplication, since it is a widely used operation in many application domains, such as numerical analysis, graphics, graphs, and conjugate gradients. Finally, with respect to CFD/FEM solvers, OOPS will focus on implementing a set of kernels from the PETSc suite:

- The Jacobi preconditioner and provide a Jacobi iterative linear system solver. Prior work on the field has considered pipelining, loop unrolling and dataflow for forting Jacobi solvers on FPGAs [5.20]. The Jacobi method often converges rather slowly compared to the more sophisticated methods and thus is seldom used as a stand-alone solver, but rather as a preconditioner for more advanced iterative methods like conjugate gradient.
- LU and Cholesky factorizations; LU factors a matrix as the product of a lower triangular and an upper triangular matrix. LU decomposition (LUD) is a common dense linear algebra solver. The Cholesky factorization factors a matrix into the product of a lower triangular matrix and its conjugate transpose. These kernels can be significantly improved using FPGA technology as shown in previous works [5.21] [5.22].
- Conjugate Gradient iterative algorithm, applicable to sparse systems that are too large to be handled by a direct implementation or other direct methods such as the Cholesky decomposition.

4. Global overview

In the table below the key characteristics of each application are summarized in order to be discussed in D2.2 for the platform pairing.

In this section for each application, presented previously, we identify the programming, essential storage, and communication characteristics, their current and projected scales, as well as their principal bottlenecks in today's HPC machines and the critical parts of them to be accelerated. We can in this way compare them side by side and provide a synthetic overview of all of them.

Table 3 - Application summary of specific characteristics

Application	**Programming model**	**Storage/ communication characteristics**	**Scale: current / target**	**Bottleneck**	**Critical parts of the code**	**Arithmetic needed**
Robot Simulation	C++, pthreads, OpenMP	1-10 GB of storage per simulation. Bandwidth intensive at ML checkpoints	Unknown	DNN	Image processing	SP/DP
Underground Analysis	C++ OpenMP MPI	1500GB max scale/multi-node communication	1000M equations/1000M equations	Computing power	Preconditioner initialization PCG solver	SP/DP
MESHFREE	FORTRAN	1-100 GB of storage per simulation. Bandwidth intensive at MPI synchronisation points	Unknown	Computation intensive complex task	Pointcloud Organisation	SP/DP
Lattice Boltzmann CFD	MPI PThreads	1-100 GB of storage per simulation; Almost Nearest neighbor communication;	up to 100M cells/ 100-1000x needed	Storage / data bottleneck. Network bottleneck mainly due to latency. Both computational and memory intensive.	Efficient solver parallelization targeted to the problem	SP

Table 4 - The OOPS library set to be implemented onto the OPTIMA hardware platform

Category	**Kernels**	
BLAS	L1	ROT, ROTM, SWAP, SCAL, COPY, AXPY, DOT, DSDOT, NRM2, ASUM, IAMAX, AMIN, AMAX
	L2	GEMV, GBMV, SYMV, SBMV, SPMV, TRMV, TBMV, TPMV, TRSV, TBSV, TPSV
	L3	GEMM, SYMM, TRMM, TRSM
Sparse Linear Algebra	SpMV	
General Computer-Aided Engineering (CAE) solvers	PETSc	Preconditioners: Jacobi Direct solvers: LU factorization / decomposition Krylov: CG

5. Matching Applications to Systems (Updated Version)

The matching criteria presented in the previous version are here slightly updated in order to better fit with the OPTIMA HARDWARE PLATFORM. The updated status of the matching is the following:

Robot Simulation: The MaxJ platform turned out to be a working solution to implement the Robot Simulation code. It was initially chosen due to unavailability of the ExaNest platform at the time CYB wanted to start working on the implementation. However, it turned out that the lack of GPU was impacting the overall performance as the bottleneck was on the physics simulator side (especially camera image generation) and not in the FPGA-based machine learning side. Therefore, a mid range GPU was added to one of Jumax machine to run the simulator side efficiently and take full advantage of the FPGA capabilities. Evaluation, software adaptation and deployment work is currently on-going to adapt to the new system configuration.

Underground Analysis: Underground Analysis software relies on the CHRONOS library that will be ported to FPGAs during the project. The FPGA version of the CHRONOS library relies on the L1 BLAS and SpMV that are available by OPTIMA as part of OPen Source (OOPS) library. OOPS library is available on both Alveo prototype and ExaNeSt. In particular, for the development and preliminary testing phase, the Alveo prototype will be used. Then the application was ported to the ExaNeSt prototype, with the results being less promising than the ALVEO accelerator card. The final implementation will be done using the new ALVEO based prototype.

MESHFREE: MESHFREE can also be ported to both platforms. Previously, the code was only compiled with the Intel Fortran compiler on x86. Using the code on ExaNest would have consisted of additional efforts to porting, optimization, and verification of the code and its dependencies on ARM. JuMax with an x86 host CPU had a clear advantage for MESHFREE. Hence, porting efforts were started there. Meanwhile, the code can be compiled with gfortran and has also been preliminarily tested on ARM. However, this is still at an experimental stage.

LatticeBoltzmann-CFD: LatticeBoltzmann-CFD has the solver written in C/C++ which is memory bound, and therefore better fitting the JuMax platform. The compilation for the CPU version can be done with any compiler, the FPGA acceleration will take advantage of the Maxeler's Dataflow programming model. It is also possible to test multiple FPGAs (with the necessary dedicated RAM) directly orchestrated from the same host.

A summary of the recommended application matching is given in Table 5.1.

Application	Partner	Platform(s)	Tools and Environment
Underground Analysis	M3E	Alveo,ExaNest	MPI, Parallelware

Robot Simulation	CYB	JuMax	Parallelware
MESHFREE	FRAUN	JuMax	MPI, Parallelware, GPI-2
LatticeBoltzmann-CFD	ES	JuMax	MPI, pthreads, Python
Open-Source Libraries	ICCS	Alveo, ExaNest	C/C++, OpenCL, Xilinx XRT

TABLE 5.1: OPTIMA Applications Matching platform

6. Concluding remarks

In this document we have presented the updated specifications of applications in the OPTIMA project together with the proposed mathematical OpenSource Library for FPGAs. For each of them we briefly presented the scientific/ technical context and then we discussed technical and implementation details based on the experience of WP3 and first implementation of them on the OPTIMA hardware. The work done will continue with the optimization of the applications in WP5 with the platform matching described in Chapter 5.

7. References

Robot Simulation

[1.1] Michel, O., Webots: Professional Mobile Robot Simulation, Journal of Advanced Robotics Systems, 2004, Vol. 1, Number 1, Pages 39-42.

[1.2] Cyberbotics Webots web page. https://cyberbotics.com

[1.3] Webots Github web page: https://github.com/cyberbotics/webots

M3E - Underground Analysis

[2.1] Janna C., Isotton G., Frigo M., Spiezia N. and Tosatto O., ATLAS Web page. https://www.m3eweb.it/atlas, 2020.

[2.2] Janna C., Isotton G. and Frigo M., Chronos Web page. https://www.m3eweb.it/chronos, 2020.

MESHFREE

[3.1] MESHFREE web page. https://meshfree.eu

[3.2] MESHFREE publication list. https://www.meshfree.eu/en/media-publications.html#1

LBM-CFD

[4.1] Bernaschi M, Fatica M, Melchionna S, Succi S, Kaxiras E. A flexible high-performance Lattice Boltzmann GPU code for thesimulations of fluid flows in complex geometries. Concurrency Computat.: Pract. Exper. 2010;22 :1.

[4.2] Rinaldi, Dari, Vnere, & Clausse A Lattice-Boltzmann solver for 3D fluid simulation on GPU, 2012

[4.3] Performance modeling and analysis of heterogeneous lattice Boltzmann simulations on CPU-GPU clusters, Parallel Computing, Vol. 46, June 1, 2015 (Feichtinger, Habich, Kstler, Rude, & Aoki, 2015)

[4.4] Pivanti, M., Mantovani, F., Schifano, S., Tripiccione, R., Zenesini, L.: An optimized lattice boltzmann code for bluegene/q. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) Parallel Processing and Applied Mathematics. LNCS, vol. 8385, pp. 385–394. Springer, Heidelberg (2014)

OPEN SOURCE LIBRARIES

[5.1] Tiziano De Matteis, Johannes de Fine Licht, and Torsten Hoefler. 2020. FBLAS: streaming linear algebra on FPGA. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '20). IEEE Press, Article 59, 1–13.

[5.2] https://www.xilinx.com/products/design-tools/vitis/vitis-libraries.html

[5.3] Q. Gautier, A. Althoff, Pingfan Meng and R. Kastner, "Spector: An OpenCL FPGA benchmark suite," 2016 International Conference on Field-Programmable Technology (FPT), 2016, pp. 141-148, doi: 10.1109/FPT.2016.7929519.

[5.4] K. Krommydas, A. E. Helal, A. Verma and W. Feng, "Bridging the Performance-Programmability Gap for FPGAs via OpenCL: A Case Study with OpenDwarfs," 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2016, pp. 198-198, doi: 10.1109/FCCM.2016.56.

[5.5] Marius Meyer, Tobias Kenter, Christian Plessl, “Evaluating FPGA Accelerator Performance with a Parameterized OpenCL Adaptation of the HPCChallenge Benchmark Suite” 2020 arXiv:2004.11059

[5.6] H. R. Zohouri, A. Podobas and S. Matsuoka, "High-Performance High-Order Stencil Computation on FPGAs Using OpenCL," 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2018, pp. 123-130, doi: 10.1109/IPDPSW.2018.00027.

[5.7] Johannes de Fine Licht, Maciej Besta, Simon Meierhans, and Torsten Hoefler. 2021. Transformations of High-Level Synthesis Codes for High-Performance Computing. IEEE Trans. Parallel Distrib. Syst. 32, 5 (May 2021), 1014–1029. DOI:https://doi.org/10.1109/TPDS.2020.3039409

[5.8] https://software.intel.com/content/www/us/en/develop/tools/oneapi/application-catalog/full-catalog/stencilstream.html

[5.9] Hamid Reza Zohouri, Naoya Maruyama, Aaron Smith, Motohiko Matsuda, and Satoshi Matsuoka, "Evaluating and Optimizing OpenCL Kernels for High Performance Computing with FPGAs," Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC'16), Nov. 2016

[5.10] Krommydas, K., Feng, Wc., Antonopoulos, C.D. et al. OpenDwarfs: Characterization of Dwarf-Based Benchmarks on Fixed and Reconfigurable Architectures. J Sign Process Syst 85, 373–392 (2016). https://doi.org/10.1007/s11265-015-1051-z

[5.11] Jason Cong, Zhenman Fang, Michael Lo, Hanrui Wang, Jingxian Xu, Shaochong Zhang. "Understanding Performance Differences of FPGAs and GPUs". The 26th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM 2018 short paper), Boulder CO, May 2018, pp. 172-175.

[5.12] A. Ramaswami, T. Kenter, T. D. Kühne and C. Plessl, "Efficient Ab-Initio Molecular Dynamic Simulations by Offloading Fast Fourier Transformations to FPGAs," 2020 30th International Conference on Field-Programmable Logic and Applications (FPL), 2020, pp. 353-354, doi: 10.1109/FPL50879.2020.00065.

[5.13] H.G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object orientated techniques. Computers in Physics, 12(6):620 - 631, 1998.

[5.14] https://www.openfoam.com/

[5.15] https://www.bytelake.com/en/

[5.16] https://marcrojek.medium.com/accelerating-cfd-with-fpga-616cd93d733f

[5.17] Lawson, C. & Hanson, Richard & Kincaid, David & Krogh, Fred. (1979). Basic linear algebra subprograms for FORTRAN usage. ACM Trans. Math. Softw.. 5. 308-323. 10.1145/355841.355847.

[5.18] Dongarra, Jack & Croz, Jeremy & Hammarling, Sven & Duff, Iain. (1990). A Set of Level 3 Basic Linear Algebra Subprograms. ACM Transactions on Mathematical Software (TOMS). 16. 1-17. 10.1145/77626.79170.

[5.19] Balay S., Gropp W.D., McInnes L.C., Smith B.F. (1997) Efficient Management of Parallelism in Object-Oriented Numerical Software Libraries. In: Arge E., Bruaset A.M., Langtangen H.P. (eds) Modern Software Tools for Scientific Computing. Birkhäuser, Boston, MA.

[5.20] Ignacio Bravo, César Vázquez, Alfredo Gardel, José L. Lázaro, Esther Palomar, "High Level Synthesis FPGA Implementation of the Jacobi Algorithm to Solve the Eigen Problem", *Mathematical Problems in Engineering*, vol. 2015, Article ID 870569, 11 pages, 2015.

[5.21] J. Luo, Q. Huang, S. Chang, X. Song and Y. Shang, "High throughput Cholesky decomposition based on FPGA," *2013 6th International Congress on Image and Signal Processing (CISP)*, 2013, pp. 1649-1653, doi: 10.1109/CISP.2013.6743941.
[5.22] Altera Corporation, “Cholesky Solver Reference Design Datasheet”, 2014