Deliverable D2.3

Evaluation Criteria

2021-08-31

Work package:	WP2 UPDATED SPECIFICATIONS AND EVALUATION CRITERIA	
Editor:	ICCS	Dionisios Pnevmatikaos
Author(s):	ICCS	Dimitris Theodoropoulos
	ICCS	Chloi Alverti
	ICCS	Panagiotis Miliadis
	M3E	Giovanni Isotton
	FRAUN	Elisa Thiel
	ES	Gino Perna
Reviewer #1	ES	Gino Perna
Reviewer #2	ITWM	Jens Bender
Dissemination level	Public	
Nature	Report	

Date	Author(s)	Comments	Version	Status
21-06-07	Dimitris Theodoropoulos	Initial version	0.1	First
21-04-08	Pavlos Malakonakis	Section 2.5.1		
21-09-05	Gino Perna	Review	1	
21-09-08	Jens Bender	Review	1	

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.

Executive Summary

This deliverable focuses on amending the original KPIs that were provided in Table 1 of the DoA. KPIs amendment is driven by objectives 1 and 4 that target highly productive and programming environments and open HPC systems supported by FPGAs. As a result, these updated KPIs will enable the comparison of the OPTIMA platform against currently available ones with respect to productivity, as well as performance and energy benefits.

D2.3 provides an overview of existing programming environments used by the project application developers, as well as the OPTIMA development flow. Moreover, it presents the baseline platform that will be used for performance and energy comparison of applications when executed with and without hardware support. Finally, D2.3 provides the updated KPIs for each application respectively.

Contents

Executive Summary 1
List of Abbreviations 3
1. Introduction 4
2. Programming Flow and Evaluation 7
2.1 Programming Flow and Evaluation 7
2.2 Underground Analysis Development Flow 7
2.3 MESHFREE Development Flow 7
2.4 Lattice-Boltzmann CFD Development Flow 8
2.5 OPTIMA Platforms Development Flow 8
3. Application Execution Evaluation 11
4. Programming and Execution Evaluation Criteria 12
4.1 KPI Updates 12
4.2 Programming KPIs 13
4.3 Execution KPIs 13
5. Concluding remarks 14
6. References 15

List of Abbreviations

CP	CheckPoint
CUDA	Compute Unified Device Architecture
D2.2	Deliverable 2.2
DoA	Description of Action
DNN	Deep Neural Network
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Units
HPC	High Performance Computing
LBM	Lattice-Boltzmann method
MPI	Message Passing Interface
NUMA	Non-Uniform Memory Access
NVM	Non-Volatile Memory
OpenAL	Cross-platform 3D audio API
OpenCV	Open Source Computer Vision
OpenGL	Open Graphics Library
OpenMP	Open Multi-Processing.
OS	Operating System

1. Introduction

The main goal of this deliverable is to update the original KPIs provided in Table 1 of the original DoA, by fine-tuning their target values with respect to each OPTIMA application. All target values are amended to ensure that Objectives 1 and 4 are met.

Objective 1: Apply, evaluate, and optimize a set of easy-to-use and highly productive programming environments for FPGA-based HPC systems

Objective 4: Provide an open system that any industrial and/or academic partner can use in order to experiment with FPGA-based HPC systems.

The final set of OPTIMA KPIs will also drive the effort of addressing the EuroHPC-09-2019 challenge:

"To improve industrial software and codes for industrial users to fully exploit the new capabilities of extreme performance HPC environments. This includes aspects such as novel algorithms, efficiency, scalability, refactoring, porting and optimization to novel HPC hardware and software architectures of increased supercomputing performance"

For convenience, Table 1 lists all KPIs presented in the DoA.

Table 1 - Original KPIs that were provided in the DoA.

KPI#	KPI description	Target Value
1	Application development time when compared with that of conventional FPGA-based tools: Ratio of the time required for application development using the programming tools of OPTIMA and the corresponding time required using conventional tools (TL) for FPGA-based systems over the later one, $T=(TTL-TOPTIMA)\ /\ TTL$	$0.3 < T \leq 0.5$ A least 2x and probably 3x acceleration in the development time when compared with the conventional single FPGA-tailored tools
2	Application development time when compared with tools used for programming GPU/Many-core : Ratio of the time required for application development using the modelling and programming tools of OPTIMA and the corresponding time required using conventional tools (CL) for programming many-core/GPU systems over the later one, $A=(ACL - AOPTIMA)\ /ACL$	$1.0 < C \leq 1.3$ Similar (up to 30% more) development time when compared with that needed for efficiently programming HPC systems consisting of GPUs and/or Many-cores
3	Acceleration: Ratio of the performance difference of the applications optimized and executed on the OPTIMA platform and the ones executed on conventional (CA) systems over the later one,	$3 < Ra \leq 5$ for HPC systems with only CPUs

	Ra=($ROPTIMA$ - RCA)/RCA	At least 3x, probably 5x higher performance 1.5 < Ra≤3 for HPC systems having also GPUs At least 1.5x, probably 3x higher performance
4	Modelling accuracy: Ratio of the modelling accuracy difference using OPTIMA's machine learning framework and that of the conventional (CA) machine learning approaches over the later one, Ma = ($MOPTIMA$–MCA) / MC	0.2 < Ma≤0.3 At least 20% and probably 30% more accurate Machine Learning related results
5	Cost effectiveness: Ratio of the precision accuracy of the machine learning models achieved by OPTIMA over the energy cost to achieve such accuracy normalized to that of a conventional system, Q=(($POPTIMA$ / $EOPTIMA$)– (PCA / ECA)) / (PCA / ECA)	12 < Q ≤ 15 At least 12x and probably 15x higher precision accuracy per unit of energy consumed
6	Performance/Energy improvement over CPU: Ratio of performance per unit of power consumption of conventional CPU-based HPC systems against that achieved by the heterogeneous OPTIMA platform X= ($PCPU$–$POPTIMA$)/$PCPU$	20 < X < 50 At least 20x and probably 50x better performance per unit of power
7	Performance/Energy improvement over GPU: Ratio of performance per unit of power consumption of GPU-based HPC systems against that achieved by the FPGA-based OPTIMA platform Y= ($PGPU$–$POPTIMA$) / PGP	3 < X < 5 At least 3x and probably 5x better performance per unit of power over GPUs
8	Scaling effectiveness: We will measure OPTIMA's scale effectiveness with respect to HPC application needs, meaning that we evaluate the degree that the OPTIMA framework supports the development of small to large scale HPC applications.	The exact formula for this KPI as well as the target value will be defined in WP1 along with additional evaluation criteria.
9	OpenLibrary FPGA porting: We will measure OPTIMA's effectiveness in porting open libraries at the individual (key) functionality (F) and as a whole (L).	8 < F < 16, 2 <= L We target the porting of at least two domain specific libraries and more than 8 lower level (but important and general) functions.

The rest of this deliverable is organized as follows. Section 2 focuses on the development flow for all OPTIMA applications when mapped and executed on currently available platforms, such as nowadays workstations, HPC / GPU-supported servers, etc. The section also describes the benefits of both OPTIMA hardware platforms development tools with respect to productivity. Section 3 describes the final evaluation platform that will be used to compare all amended KPIs. Section 4 lists the updated KPIs separately for each application. Finally, Section 5 concludes this report.

2. Programming Flow and Evaluation

This section presents the development flow for all OPTIMA applications, as well as both hardware platforms (ExaNest and Maxeler).

2.1 Programming Flow and Evaluation

The main programming language will be C or C++. We may also use Python at a higher level to compare with GPU and CPU implementations on a fair basis. We will develop robotics simulations that make use of FPGA acceleration for controlling the robot behavior. The development effort will include the simulation models including simulation proto files, controller programs written in C, C++ and/or Python and the documentation. These examples will be published as open-source software. There might be some overlap with other deliverables as we will use standard libraries for machine learning and vision which may also be used in other deliverables. In that case, we could reuse the results of the other deliverables to leverage on them and save development time.

2.2 Underground Analysis Development Flow

Chrons is mainly written in C++ and its design is strongly object-oriented to facilitate its maintainability and possibility to be interfaced to other softwares. Inter-process data communication is performed using MPI while OpenMP is used for intra-node communications.

The porting on FPGA systems will make use of the OPTIMA OPen Source (OOPS) package developed within the OPTIMA project. In particular, the MPI communication between nodes will be preserved, using OpenCL APIs for host/device communications. For the innermost kernel, OOPS library will be used.

The performance evaluation will be performed on industrial test cases taken from M3E collection (https://www.m3eweb.it/matrixcollection/).

2.3 MESHFREE Development Flow

The software is mainly written in Fortran with MPI directives for inter-process communications. The code has been developed over the past 20 years. So far, there has not been a development on a GPU platform to compare with. The software development is in general scientifically driven, that means that novel algorithms are developed and incorporated to tackle scientific questions. As the software is then used in industrial applications, the development is balanced with performance optimization.

The first phase will consist of implementing and testing isolated MESHFREE tasks on the target platform. In the second phase, we will develop an integrated version of MESHFREE on the target platform.

2.4 Lattice-Boltzmann CFD Development Flow

The software is implemented in Python with the possibility to generate code for CUDA C/C++ and simplified OpenCL from a templating library. This combination makes the overall system a very powerful one, making it possible to significantly shorten development time without sacrificing any computational performance. Communications are handled by a mix of OpenMPI and pthreads to handle compatibility on the boundaries of the partitions.

The porting will first be tested on the target platform as is, in order to check all libraries and dependencies. Then an incremental approach will be followed by starting to implement on FPGAs the critical parts of the code and the necessary bindings with the application. This will ensure progress monitoring and measuring of improvements on the way. The all work will take advantage of the OPTIMA platform, especially foreseeing the MAXJ potential for acceleration, by following the MAXJ docs and suggestions.

Final evaluation will be performed on well-known test-cases in order to measure all required parameters for KPIs, by monitoring Scale-Effectiveness, Energy consumption, cpu-load, communications.

2.5 OPTIMA Platforms Development Flow

The two following sections will summarize the development flow of each platform that will be available for this project:

ExaNest-based platform Development Flow: D2.2 (Section 3.1.1) presented the development tools that are required for the implementation of applications targeting the ExaNest-based platform. This section focuses on the programming flow and the required effort from the user to port an application to the ExaNest-based platform.

The Vitis Unified Software Platform is the tool that incorporates all the required functionality that is required for the development of an application to run on the ExaNest-based platform. Each application can be separated into 3 distinct coding segments. The first segment is the host application, the second takes care of the software - hardware interface (XRT) and lastly the third segment is the hardware accelerated kernel description. In Figure 1 the v++ compiler outputs the hardware design, XRT takes care of the software interface while g++ produces the host application.

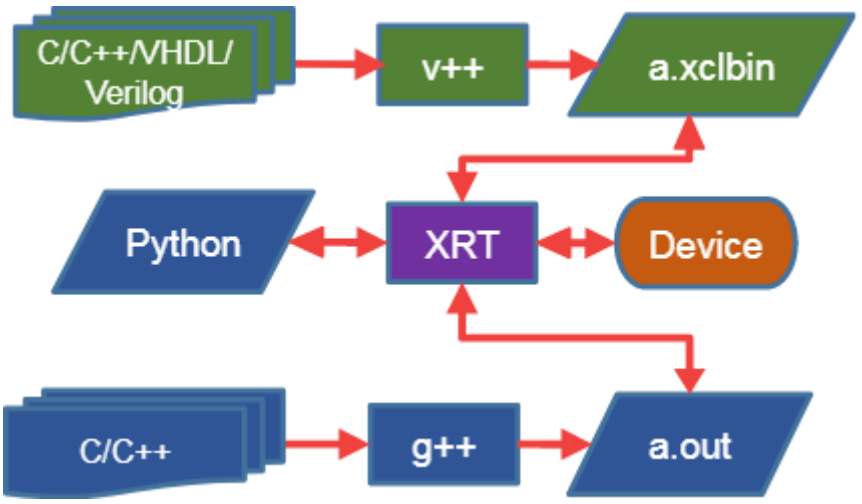

Figure 1 - User application compilation and execution [1]

The first code segment, the host application, basically includes all the application host code. The host application can be written in either C/C++ or python. The application host code has to be inputted in the Vitis Unified Software tool and any dependencies have to be resolved before compiling. Dependencies could involve missing libraries or code modifications required by the fact that the ExaNest-based platform has ARM processors. This procedure should be relatively easy and straightforward. After the compilation process, the host application's functionality can be verified using software emulation, which is done with the help of QEMU emulator.

Afterwards, the code segment that is to be accelerated using the FPGA has to be separated from the host application and added to a separate file, which will be the accelerated kernel. A copy of the source code should be also available for verification reasons. The second code segment involves the implementation, on the host code, of the hardware/software interface using the OpenCl API (XRT). A code sample of the XRT API can be found in D2.2.

Finally, the third and last segment is the accelerated kernels implementation. The accelerated kernels must be written in C/C++. The tool also supports Register Transfer Level (RTL) description for the kernels. As mentioned above, the accelerated kernel's source code has to be separated from the host application. Then it has to be modified in order for it to be synthesizable. For example, dynamic memory allocation (malloc) is not supported in an FPGA implementation. After the corrections towards synthesizability, software emulation and hardware emulation can be invoked for the verification of the whole application.

When these code segments are complete, the first hardware implementation is ready to be deployed to the platform. The initial implementation of an application to be executed in FPGA is relatively simple and requires very little modifications and additions to the actual application code. Usually though, a software-oriented implementation of a kernel will not provide the best performance when deployed on hardware. Several optimization iterations may be required for the kernel to exploit the FPGA capabilities, and as a result provide the best available performance. Also, depending on the application, after the initial implementation, a hardware architecture design step may be required before moving to the optimization iterations. A hardware architecture designed for the specific kernel could significantly reduce the optimization iterations. The HLS tools have made the

implementation of applications targeting FPGAs significantly faster, relative to RTL description. Finally, and most importantly, they have allowed software engineers to be involved in the hardware acceleration using FPGAs. The implementation of an application targeting an FPGA could be considered even simpler than a GPU implementation. With XRT, GPUs and FPGAs use the same software/hardware communication API, and while GPUs require the description on a specific language (Cuda, OpenCl), the C/C++ source code of the application, with minor changes, can be used for the FPGA implementation. The description of the development tools and sample code segments can be found in D2.2.

Maxeler-based Platform Development Flow: Maxeler provides a high-level dataflow-oriented programming model in its MaxCompiler where kernels are described in MaxJ, a Java-based meta language. The focus of this tool and the surrounding development approach is on productivity to guide the designer quickly to an optimal solution quickly while avoiding time-consuming iterations with the FPGA place & route tools. By targeting Maxeler's inhouse-developed FPGA-based Dataflow Engines (DFEs) or third-party platforms such as Xilinx Alveo with the highly productive MaxCompiler tool flow, developers can realise one to two orders of magnitude improvements in terms of performance and energy efficiency as compared to conventional systems. A detailed description can be found in D2.2.

3. Application Execution Evaluation

This section describes the evaluation setup for all OPTIMA applications. As mentioned in D2.2, the ExaNest-based platform is based on Quad-FPGA Daughter Boards (QFDBs), and each QFDB supports four tightly coupled Xilinx Zynq Ultrascale+ MPSoCs as well as 64 Gigabytes of DDR4 memory. The Ultrascale+ MPSoCs tightly couple FPGA resources with quad-core ARM Cortex-A53 processors operating at 1.2GHz. On the other hand, the Maxeler host CPU model is an AMD EPYC 7601 (32-core). The machine integrates 2 such CPUs with 1 TB DDR4 ECC RAM, as well as 8 accelerator cards, each with a Xilinx Virtex Ultrascale+ VU9P chip.

Table 2 - Application evaluation onto the OPTIMA platforms.

Application	Implementation	Baseline CPU	OPTIMA platform
Underground Analysis	only on this platform	ARM Cortex-A53	ARM Cortex-A53 with FPGA
Robot Simulation	primary target platform		
Lattice Boltzmann-CFD	secondary target platform		
MESHFREE	only on this platform	AMD EPYC 7601	AMD EPYC 7601 with FPGA
Robot Simulation	secondary target platform		
Lattice Boltzmann-CFD	primary target platform		

Table 2 shows how applications will be evaluated. Each application will be first mapped and executed to the target platform's CPU, to acquire baseline performance and energy footprints. Results will then be compared against the hardware-supported version of each application. Finally, as mentioned in D2.2, the Robotic Simulation and Lattice Boltzmann-CFD applications will be further explored to decide the final implementation platform.

4. Programming and Execution Evaluation Criteria

This section focuses on amending the original programming and execution evaluation criteria with respect to each application.

4.1 KPI Updates

Table 3 - KPI amendments compared to the original ones.

KPI #	Original KPI	Changes
2	Application development time when compared with tools used for programming GPU/Many-core: Ratio of the time required for application development using the modelling and programming tools of OPTIMA and the corresponding time required using conventional tools (CL) for programming many-core/GPU systems over the later one, $A=(ACL - AOPTIMA)/ACL$	Evaluation formula is updated to $A=AOPTIMA/ACL$
3	Acceleration: Ratio of the performance difference of the applications optimized and executed on the OPTIMA platform and the ones executed on conventional (CA) systems over the later one, $Ra=(ROPTIMA - RCA)/RCA$	Evaluation formula is updated to $Ra=ROPTIMA/RCA$
8	Scaling effectiveness: We will measure OPTIMA's scale effectiveness with respect to HPC application needs, meaning that we evaluate the degree that the OPTIMA framework supports the development of small to large scale HPC applications.	Scaling effectiveness will be evaluated using the ratio of the ideal execution time when running an application onto N FPGAs versus the experimental (measured) one: SE = ExTime (ideal) / ExTime (actual) ExTime ideal (N) = Execution Time (1) / N + Communication Overhead (N) Target values: $0.7 < SE < 1$

Table 3 lists a set of updates on the KPIs that were decided.

4.2 Programming KPIs

Table 4 - Target values of applications on programming KPIs.

Application	Programming KPI	Target Value
Robotics Simulations	1	$0.3 < T \leq 0.5$
	2	$1.0 < A \leq 1.5$
	5	$12 < Q \leq 15$
	9	$F \geq 4, L = 1$
Underground Analysis	1	$0.3 < T \leq 0.5$
	2	$1.0 < A \leq 1.3$
	9	$8 < F < 16, L = 1$
MESHFREE	1	$0.3 < T \leq 0.5$
	2	$1.0 < A \leq 1.3$
	9	$8 < F < 16, L = 1$
LatticeBoltzmann CFD	1	$0.3 < T \leq 0.5$
	2	$1.0 < A \leq 1.3$
	9	$8 < F < 16, L = 1$

Table 4 lists the target values of all applications with respect to the project's programming KPIs.

4.3 Execution KPIs

Table 5 - Target values of applications on execution KPIs.

Application	Programming KPI	Target Value
Robotics Simulations	3	$3 < Ra \leq 5$ for HPC systems with only CPUs $1.5 < Ra \leq 3$ for HPC systems having also GPUs
	6	$20 < X < 50$

	7	$3 < Y < 5$
	8	$0.7 < \mathrm{SE} < 1$
Underground Analysis	3	$3 < Ra \leq 5$
	6	$20 < X < 50$
	8	$0.7 < \mathrm{SE} < 1$
MESHFREE	3	$3 < Ra \leq 5$
	6	$20 < X < 50$
	8	$0.7 < \mathrm{SE} < 1$
LatticeBoltzmann CFD	3	$3 < Ra \leq 5$
	6	$20 < X < 50$
	8	$0.7 < \mathrm{SE} < 1$

Table 5 lists the target values of all applications with respect to the project's programming KPIs.

5. Concluding remarks

This report presented the amended set of the OPTIMA KPIs, which are fine-tuned for each application. These updated KPIs will be later used for the comparison of the OPTIMA platform against the currently used ones with respect to productivity, as well as performance and energy cost to solution.

6. References

[1] XRT and Vitis™ Platform Overview