H2020 FRAMEWORK PROGRAMME
ICT-01-2014: Smart Cyber-Physical Systems

PROJECT NUMBER: 645496

Agile, eXtensible, fast I/O Module for the cyber-physical era

D3.3 – Software and Report on Application Porting

Due date of deliverable: 31st January 2018
Actual Submission: 14th February 2018 (agreed extended date)

Start date of the project: February 1st, 2015 Duration: 36 months

Lead contractor for the deliverable: UNISI

Revision: See file name in document footer.

Project co-founded by the European Commission within the HORIZON FRAMEWORK PROGRAMME (2020)	
Dissemination Level: PU	
PU	Public
PP	Restricted to other programs participant (including the Commission Services)
RE	Restricted to a group specified by the consortium (including the Commission Services)
CO	Confidential, only for members of the consortium (including the Commission Services)

Change Control

Version#	Date	Author	Organization	Change History
0.1	09.01.2018	Antonio Rizzo	UNISI	Initial version
0.2	12.01.2018	David Oro	HERTA	Added SVS section
0.3	16.01.2018	Nicola Bettin	VIMAR	Added SLH section
1.0	29.01.2018	Antonio Rizzo	UNISI	First document draft

Release Approval

Name	Role	Date
Antonio Rizzo	WP Leader	29.01.2018
Roberto Giorgi	Coordinator	14.02.2018

The following list of authors will be updated to reflect the list of contributors to the document.

Antonio Rizzo, Francesco Montefoschi, Sara Ermini
Department of Social, Political and Cognitive Sciences
University of Siena – UNISI – AXIOM

David Oro
RD department
Herta Security – HERTA – AXIOM

Nicola Bettin
RD department
VIMAR spa – VIMAR – AXIOM

TABLE OF CONTENTS

GLOSSARY 5

Executive summary 6

1 Document structure 7
1.1 Document structure 7
1.2 Tasks involved in this Deliverable 7

2 Smart Video Surveillance Application Porting 8
2.1 Software Parallelization on the AXIOM Platform 10

3 Smart Home Living Service Porting 16
3.1 SHL Application 16
3.2 Porting of the Application to AXIOM board 16
3.3 Porting the system to the Video door entry system 20

4 User Interface Design and Testing 22
4.1 Smart Video Surveillance Scenarios 22
4.1.1 First Scenario 23
4.1.2 Second Scenario 23
4.1.3 Third Scenario 24
4.2 Smart Home Living Scenarios 24
4.2.1 First Scenario 24
4.2.2 Second Scenario 26
4.2.3 Third Scenario 26
4.3 Experience Testing 27
4.3.1 Concluding remarks 36

5 Confirmation of DoA objectives 37

6 Conclusions 38

7 References 39

TABLE OF FIGUREs

FIGURE 1 - OFFLOADING OF THE MAIN SVS APPLICATION COMPONENTS TO THE ULTRASCALE+ MPSOC CLUSTERS. 9
FIGURE 2 - SVS APPLICATION RUNNING ON THE AXIOM BOARD. 10
FIGURE 3 - GLSL SHADERS USED IN THE UI OF THE SVS FOR SCALING THE VIDEO OVERLAY 11
FIGURE 4 - GPU DEBUGGING OF THE IMAGE PYRAMID TEXTURE MIPMAP GENERATION 12
FIGURE 5 - XDMA LINUX KERNEL MODULE PERFORMING CPU/FPGA MEMORY TRANSFERS DURING FPGA LBP KERNEL EXECUTION. 13
FIGURE 6 - GENERAL OVERVIEW OF PARAVER SHOWING A PROFILE TRACE OF THE LBP KERNEL FPGA ACCELERATOR PERFORMING EXECUTIONS ON THE ULTRASCALE+ RECONFIGURABLE LOGIC. 14
FIGURE 7 - ZOOMED IN PARAVER VIEW OF THE PREVIOUS LBP KERNEL FPGA ACCELERATOR PROFILE TRACE 14
FIGURE 8 - FPGA UTILIZATION OF THE LBP KERNEL ON THE ULTRASCALE+ ZU9EG MPSOC. 15
FIGURE 9 - EXECUTION TIME OF THE THREE SHL APPLICATION KERNELS ON THE AEP AND THE AXIOM BOARD. THE RESULTS DEPICTED ABOVE WERE OBTAINED BY MANUALLY ENABLING AND DISABLING OMPSS@SMP DIRECTIVES 17
FIGURE 10 - PARAVER TRACE OF THE AXIOM_BIO_IDENTIFICATION APPLICATION ON THE AXIOM BOARD. 18

FIGURE 11 - EXECUTION TIME OF THE MODULE IN THE AXIOM BOARD WHEN THE ALGORITHM WAS PROCESSED ON THE SMP CORE AND ON THE FPGA ACCELERATOR. 19
FIGURE 12 - FPGA RESOURCES UTILIZED BY ANISOTROPIC SMOOTHING MODULE ON THE AXIOM BOARD. 19
FIGURE 13 - VIMAR DEMO SYSTEM. OF THE SHL APPLICATION 21
FIGURE 14 - SCREENSHOTS OF THE MINIMAL USER INTERFACE OF THE AXIOM_BIO_IDENTIFICATION APPLICATION. THE SCREENSHOTS SHOW THE MAIN MENU OF THE APPLICATION, THE "ADD USER" MENU, THE" RUNNING MENU" CONFIGURATION, AND THE "RUNNING WINDOW". 21
FIGURE 15 - THE SCREENSHOT OF THE MAIN SCREEN OF THE USER INTERFACE FOR THE AXIOM SVS APPLICATION. THE IMAGE SHOWS IN THE MIDDLE OF THE SCREEN THE WINDOW OF STREAMING VIDEO, THE MAIN MENU NAVIGATION ON THE TOP AND THE TWO SIDEBARS WITH WIDGETS FOR THE FACE RECOGNITION HISTORY AND ARCHIVE. 23
FIGURE 16 - THE SCREENSHOTS ARE ABOUT THE MOBILE APPLICATION. THEY SHOW THE SET UP AND THE INTERACTION WITH THE SMART VIDEO DOOR ENTRY SYSTEM. THE TWO IMAGES SHOW THE ACTION OF UNLOCKING THE DOOR WITH THE SMARTPHONE APP. 25
FIGURE 17 - THE SCREENSHOT SHOWS THE UI FOR TRAINING VOICE RECOGNITION SYSTEM FROM MOBILE APPLICATION. .. 25
FIGURE 18 - THE SCREENSHOTS ON THE LEFT SHOW THE UI FOR SETTING THE ACCESS KEY FOR DIFFERENT USERS USING MOBILE APPLICATION. THE SCREENSHOT ON THE RIGHT SHOWS THE ALERT SENT BY NOTIFICATION SYSTEM TO NOTIFY THE OWNER THAT SOMEONE ENTERED THE HOUSE. 26
FIGURE 19 - THE TWO SCREENSHOTS ARE ABOUT THE AIRBNB SCENARIO: AN EXAMPLE OF THE UI FOR THE GUEST-USER WHO IS GOING TO RENT A HOUSE (LEFT), AND THE MOCKUP OF THE INTERFACE FOR HOME OWNER WHO IS GOING TO GUEST SOMEONE IN HIS LOFT (RIGHT). 27
FIGURE 20 - WIREFRAME OF THE HOME OF THE SVS APPLICATION. ON THE PAPER WAS SKETCHED THE FIRST IDEA TO ADD A SPINNER TO FILTER FEATURES OF THE USERS DETECTED. 28
FIGURE 21 - MOCKUP OF THE MAIN VIEW OF THE SVS APPLICATION DESIGNED WITH THE QT QUICK TOOL. THE SPINNER ON THE TOP-CENTER OF THE SCREEN IS DEVELOPED AND TESTED BY REAL USERS REVEALING PROBLEMS OF TASK EXECUTION. 29
FIGURE 22 - THE IMAGE ON THE LEFT SHOWS A USER TESTING THE NEW DROP-DOWN OVERLAY FOR FILTERING TARGETED USERS USING THE SVS APPLICATION FOR DESKTOP. THE IMAGE ON THE RIGHT IS A SCREENSHOT OF THE UI TESTED. 29
FIGURE 23 - THE IMAGE SHOWS THE PAPER SKETCH AND ANNOTATION OF THE TROUBLES REVEALED DURING THE TESTING AND IDEAS FOR SOLVING THOSE PROBLEMS. 30
FIGURE 24 - THE TWO IMAGES SHOW THE SCREENSHOTS OF THE UI FOR THE HOME OF SVS APPLICATION AND THE SKETCHED NOTES AND REVIEWS ANNOTATED AFTER THE TESTING SESSION. 30
FIGURE 25 - THE IMAGE SHOWS THE WIREFRAME OF THE STATISTIC TAB BAR OF THE SVS APPLICATION AND THE ANNOTATION FOR REVIEWING THE INTERFACE AND IMPROVE THE USABILITY IN THE SECOND MOCKUP. 31
FIGURE 26 - THE IMAGES ARE ABOUT THE THREE DIFFERENT MOCKUPS DESIGNED FOR SH APPLICATION. ON THE LEFT THE FIRST IDEA, IN THE MIDDLE THE UI REVIEWED AFTER THE FIRST TESTING, AND ON THE RIGHT THE LAST VERSION DEVELOPED AFTER UX ANALYSIS. 32
FIGURE 27 - A SCREENSHOT ABOUT THE LOGIN/REGISTER PROCEDURE ON APPLICATION. 33
FIGURE 28 - TWO SCREENSHOTS OF SH APPLICATION: ON THE LEFT A STEP OF THE IRIS RECOGNITION TRAINING SYSTEM, ON THE RIGHT A STEP OF THE VOICE RECOGNITION TRAINING. 33
FIGURE 29 - A PHOTO ABOUT THE TESTING OF THE VIDEO DOOR ENTRY SYSTEM: THE USER IS TRAINING THE IRIS RECOGNITION SYSTEM. 34
FIGURE 30 - A PHOTO ABOUT A USER LOOKING AT LED FEEDBACK ON THE DEVICE DURING THE TRAINING PROCEDURE. 34
FIGURE 31 - A PHOTO DURING THE TESTING SHOWS THE USER LOOKING AT THE CAMERA AND THE BIAS OF UNDERSTANDING FEEDBACK. 35

GLOSSARY

AEP – AXIOM Evaluation Platform
AMBA – Advanced Microcontroller Bus Architecture
API – Application Programming Interface
ARM – Instruction set architecture developed by ARM Holdings Ltd.
CNN – Convolutional neural network
CPS – Cyber Physical System
CPU – Central Processing Unit
DDR – Double Data Rate RAM
DRM – Direct Rendering Manager
DSP – Digital Signal Processing
EGL – Embedded-System Graphics Library
FIFO – First In First Out
GLSL – OpenGL Shading Language
GPS – Global Positioning System
GPU – Graphic Processing Unit
HD – High Definition
GUI/UI – Graphical user interface
HDL – Hardware description language
HLS – High-level synthesis
IP – Intellectual property or Internet Protocol (depending on the context)
ISA – Instruction Set Architecture
KMS – Kernel Mode Setting
LBP – Local binary pattern
LED – Light Emitting Diode
LibAv – Open-source libraries derived from the FFmpeg project to handle multimedia data
Mali – A GPU microarchitecture developed by ARM Holdings Ltd.
Mercurium – OmpSs compiler
Nanos++ – OmpSs runtime
MPSoC – Multiprocessor System-on-Chip
OpenCV – Open Source Computer Vision Library
OpenGL ES – Reduced specification of the OpenGL standard that targets embedded devices
PL – Programmable Logic
PS – Processing System
Qt – Cross-platform application framework developed by The Qt Company
QML – Qt Meta Language
ROI – Region Of Interest
SHL/SLH – Smart home living scenario developed for the AXIOM project
SMP – Symmetric multiprocessing
SoC – System on chip
SPro – Open source speech signal-processing toolkit
SVS – Smart surveillance scenario for the AXIOM board
TCP – Transmission Control Protocol
UX – User Experience
WebRTC – Web Real-Time Communication
XDMA – AXIOM FPGA Direct Memory Access kernel module

Executive summary

This Deliverable reports the steps followed in the porting process of the Smart Video Surveillance and Smart Home Living use case applications to the heterogeneous FPGA-based architecture available on the AXIOM board. As such, the source code of the applications leveraged the OmpSs@FPGA programming model and related runtime libraries to automate HDL code generation, placement, routing, synthesis and execution on the reconfigurable logic. This toolchain is also used for abstracting the API interface to load/unload and schedule kernel launches of the bitstreams of synthesized accelerators, while hiding the underlying complexity of low-level host/device and device/host memory transfers. Preliminary experiments of OmpSs@Cluster were also conducted to show the potential of scaling and offloading computations to two AXIOM boards.

Additionally, it also describes the decisions adopted during the process of designing and prototyping the required user interfaces, as well as the steps followed for implementing and accelerating them on the on-die GPU included in the UltraScale+ SoC.

Finally, a test of the user experience was also conducted with the aim of evaluating the envisioned new services and their usability as implemented in the prototypes (e.g., determining how an inexperienced end user might interact with the different software components).

1 Document structure

1.1 Document structure

This Deliverable contains a report on the work carried out for finishing the porting process of the SVS and SLH scenarios to the AXIOM platform. The datasets collected and used to train both systems (demographic facial estimation system and voice recognition system) will be publicly released on an open data set repository (e.g. OpenAIRE [1] or Zenodo [2]) in order to be open accessible to other researchers. Additionally, it also includes the steps followed for designing and validating the graphical user interfaces designed for both use cases. The reported work is organized as follows:

- Section 2 describes the porting activity for the SVS scenario;
- Section 3 describes the porting activity for the SLH scenario;
- Section 4 describes the user interface design and testing for both scenarios;

1.2 Tasks involved in this Deliverable

This Deliverable is the result of the work developed in the following tasks:

- **Task 3.2**: Proof of Concept and Porting of SHL and SVS Case Studies
  Selection, envisioning and refinement of Scenarios to be put into scene by prototypes of AXIOM architecture in the domain of Smart Living Home and Smart Video Surveillance.
  Porting of the Smart Living Home Application and Smart Video Surveillance to the OmpSs Programming Model. [3]
  Partner UNISI (Interaction Design group) will envision new scenarios for the using the AXIOM CPS platform. UNISI will take care of the "Role Prototyping" of the App/Service, while addressing the two challenges of services/system integration and appealing user experience. UNISI will define the Interaction Design pattern in the design of the application on the AXIOM CPS. UNISI will carry out the Conception and Definition of the user experience in adopting into scene the new enabling CPSs.
  Partner VIMAR will design and develop algorithms for real-time data management used in the Home Automation application. Partner VIMAR will develop a modular, cost and power effective software architecture including a reference version and the porting of such reference version to the OmpSs programming model (the hardware part will be developed in WP6). Partner HERTA will design and develop algorithms for real-time face recognition used in the smart surveillance application. Partner HERTA will optimize the fine-grained parallel algorithm for FPGA accelerator and will develop and port their application to AXIOM architecture using OmpSs.
  Partner BSC will give support to VIMAR and HERTA for the porting of the applications to OmpSs.
- **Task 3.3**: Testing for User Experience
  Testing the AXIOM CPS mainly focusing on User Experience in the domain Smart Video Surveillance and Smart Living/Home.
  Partner UNISI will verify the user feedback of the real-life scenarios in cooperation with partners HERTA and VIMAR.

2 Smart Video Surveillance Application Porting

The Smart Video Surveillance use case application closely resembles the BioSurveillance [4] commercial software currently offered by partner HERTA. Since this product targets Windows platforms, and offloads compute intensive data-parallel kernels and video decoding to discrete NVIDIA GPUs, a major redesign was required in order to port the inner code-logic to the AXIOM platform.

As it was disclosed in Deliverable D3.2, the main architecture of the SVS application is made up of several engines that are written in ANSI C and C++ 11 languages. However, some parts of the code were more suitable for being offloaded to the reconfigurable logic (PL domain of the UltraScale+ MPSoC) using the OmpSs@FPGA programming model. More particularly, the most time-consuming parts of the application were the LBP face classifier, and the CNN inference engine used for performing the demographics estimation of detected faces. These kernels feature a data-parallel arithmetic intensity that makes them suitable for targeting the power-efficient DSPs available in the reconfigurable logic of the PL domain.

Since the Xilinx UltraScale+ ZU9EG SoC does not include and on-die H.264/HEVC video decoder, the video acquisition engine (i.e. video demuxing and slice decoding) was implemented by means of the LibAv open source collection of libraries [5]. As such, this part of the pipeline relies on a multi-threaded software implementation, which heavily exploits the ARM Cortex A57 processors available in the PS cluster.

Figure 1 summarizes which components of the SVS application exploit the two different clusters of the UltraScale+ chip included on the AXIOM board. The multiple steps depicted closely resembles a pipeline, in which the SVS main module orchestrates and schedules tasks on top of the ARM CPU cores, the FPGA reconfigurable logic, and the ARM Mali GPU.

After the initial H.264 slice video decoding is completed, it is also required to build a multi-level synthetic Gaussian pyramid. This pyramid is a step needed before issuing the kernel execution of the LBP face classifier, as the boosted cascade of features only work with a fixed-sized sliding window. Since the SVS application must be able to localize faces of arbitrary sizes, the Gaussian pyramid must include downscaled and subsampled copies of the decoded video frame. Additionally, it is also needed to avoid excessive memory transfers between the ARM CPUs hosts and the on-die BlockRAM, which must cache within the FPGA reconfigurable logic (PL domain) the image working set in order to both reduce latency and increase the memory bandwidth.

The construction of such pyramid is a step that is more suitable to be computed on the on-die ARM Mali GPU (PS domain). By relying on OpenGL ES texture mipmapping, it is possible to simultaneously compute bilinear filtering and scaling simply by performing fetches from the GPU texture cache. This step is transparently offloaded to the GPU by relying on the `glGenerateMipmap()` function, which automatically builds the required mipmap.

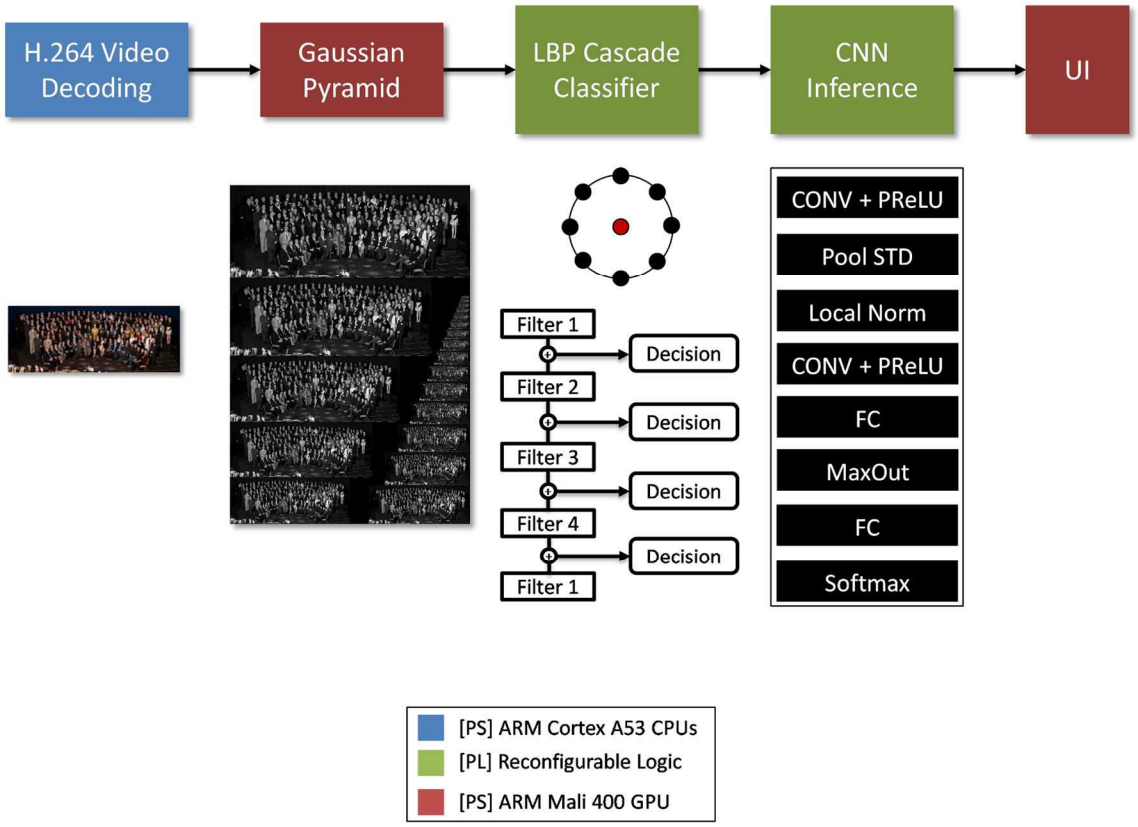

Figure 1 - Offloading of the main SVS application components to the UltraScale+ MPSoC clusters.

At this point, the demographic CNN inference is performed taking as an input a subset of detected faces, which are stored into a FIFO queue. Since there is no tracking algorithm involved, the SVS application creates a thread, then randomly selects a face from the abovementioned subset, and finally rescales it to the input size of the first layer of our pre-trained CNNs (64 x 64 pixels). For more details regarding the architecture of HERTA's proprietary CNNs, please refer to Section 2.3 included in Deliverable D3.2.

Finally, after all the pipeline stages are successfully completed, the user interface of the SVS application must show graphically the expected outcome of the facial analysis of the video footage. This UI is implemented on the AXIOM board on top of the X.org X11 server [6], using the Qt framework libraries [7], and heavily relies on OpenGL ES and GLSL GPU shader computations to efficiently map textures, perform coordinate transformations, render animations, and apply antialiasing filters. Therefore, as in the case of the Gaussian pyramid, these latter processes are also offloaded to the on-die ARM Mali GPU included in the PS cluster.

2.1 Software Parallelization on the AXIOM Platform

The main SVS application is implemented as a C++ class, and relies on a classic Model-View-Controller (MVC) design pattern for abstracting memory allocation, scheduling, and execution of the low-level compute-intensive kernels from the UI. Such kernels are implemented in isolated ANSI C files, enriched with OmpSs `#pragma` annotations, and also perform the required API calls to the XDMA runtime libraries in order to correctly allocate the FPGA accelerator input/output buffers and data structures.

On the other hand, the UI is implemented using the QML rendering engine offered by the Qt Framework set of libraries. This engine simplified substantially the development, as it enabled both HERTA and UNISI teams to code the different UI components using the simpler JavaScript language.

As Figure 2 depicts, the SVS application shows the real-time decoded video footage, which can be retrieved either from a remote H.264 IP surveillance camera or from a file stored in the filesystem of the microSD card plugged in the AXIOM board.

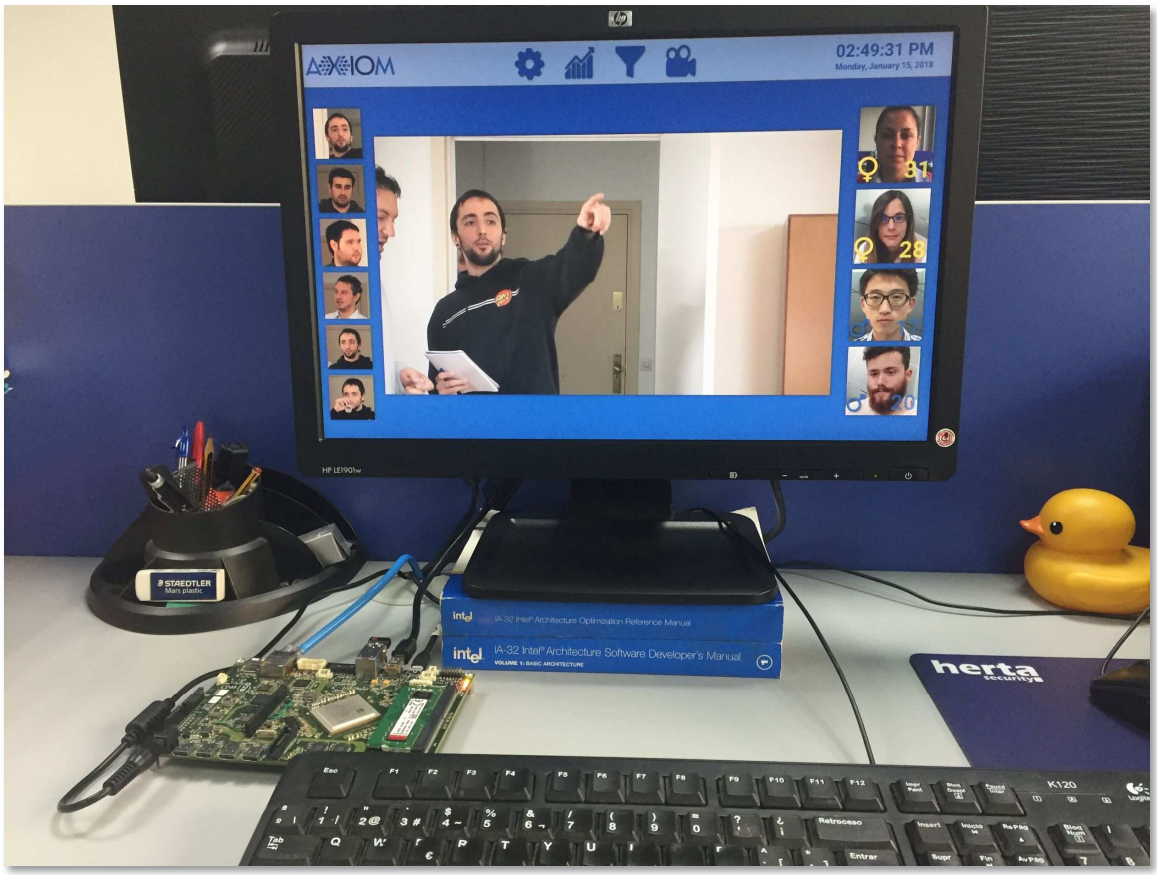

Figure 2 - SVS application running on the AXIOM board.

For each frame of the video footage, the decoded H.264 slices are merged into a `uint8_t` buffer, which is later passed as an argument to the subsequent kernels (i.e. `YUV_to_RGB`, `pyramid_compute`, `LBP_cascade_evaluation`, and `CNN_inference`) for further processing. It should be noted that the `YUV_to_RGB` kernel is only required for display, as the remaining kernels only work with luminance components. Therefore, the output of this kernel must be mapped into an RGB texture by calling its corresponding function `glTexImage2D()` from the OpenGL ES 2.0 specification.

Since the input video is usually larger than the overlay window of the UI, it is also required to perform real-time frame subsampling, and texture coordinate transformation to properly keep the aspect ratio. This process is also computed in the UI code using a combination of GLSL pixel and vertex shaders. As such, the shaders rely on data-parallel GPU SIMD instructions, which are then compiled to the target GPU ISA and executed by the GLSL ARM Mali JIT compiler included in the `libMali.so` driver.

```
/* Initialize GLSL shaders */
shader = new QOpenGLShaderProgram();
shader->addShaderFromSourceCode(QOpenGLShader::Vertex,
"attribute highp vec4 vertices;"
"varying highp vec2 coords;"
"void main() {"
"    gl_Position = vertices;"
"    coords = (vertices.xy / 2.0) - vec2(0.5,0.5);"
"}");

shader->addShaderFromSourceCode(QOpenGLShader::Fragment,
"precision mediump float;"
"uniform sampler2D texture;"
"varying vec2 coords;"
"void main()"
"{"
"   gl_FragColor = texture2D(texture, coords);"
"}");

shader->bindAttributeLocation("vertices", 0);
shader->link();
```

Figure 3 - GLSL shaders used in the UI of the SVS for scaling the video overlay.

As a consequence, enabling GPU acceleration of both OpenGL ES 2.0 and GLSL shaders (see Figure 3) in the UltraScale+ MPSoC of the AXIOM board was a major challenge, and required to develop patches to the official Qt 5.4 codebase, which are available at the private AXIOM Wiki page [8]. Typically, X11 for `aarch64` is not officially supported anymore by ARM in the Mali 400 GPU, as the company only provides EGL and OpenGL ES 2.0 drivers for the Android stack.

Therefore, we had to rely on experimental drivers of the X11 DRM/KMS OpenGL ES stack directly obtained from Xilinx, which provided an unoptimized code path to get 2D and 3D acceleration running on the AXIOM board. Similarly, the mini-DisplayPort output was turned on by interfacing the KMS driver with the `zynqmp_dp` kernel module. The final interaction between the Qt framework and the X11 server was implemented using as a glue layer the `libxcb` library.

By relying on the abovementioned combination, even though officially unsupported and highly experimental, we were able to get the SVS application GPU-accelerated UI displayed on an external screen.

The backend of the SVS application included the low-level calls to the kernels that were selected for parallelization using the OmpSs programming model (see the Appendix included in Deliverable D3.2). Since the most time critical part of the SVS pipeline is face localization, the main parallelization efforts were devoted to efficiently offloading the LBP kernel to the reconfigurable logic using OmpSs@FPGA.

In this particular kernel, the inner nested `for()` loops iterate a 48x48 sliding window over a multi-scale Gaussian image pyramid while evaluating the pre-trained cascade of LBP features. Unfortunately, due to the limitations in size of the BlockRAM included in the ZU9EG chip (4 MB), it was not possible to cache both the classifiers and the image pyramid on die. As an example, a given uncompressed H.264 1920x1080 grayscale video frame (i.e. UV components already removed) would consume 2 MB of the BlockRAM. Therefore, since the dimensions of the synthetic image pyramid corresponding to a 1080p input resolution are 1920x3936 pixels (22 scales), the memory requirements for on-die memory storage (7.2 MB) would be higher than the available on-die BlockRAM.

At this point, there were two major options considered for the FPGA implementation of the LBP kernel: i) either to fetch all video frame pixels directly from the external DDR4 memory; or ii) to cache video frame chunks together with the face classifier cascade on chip. The second option was chosen due to the higher memory bandwidth and lower latency of the BlockRAM, when compared to the external off-chip DDR4 memory. Additionally, since the FPGA reconfigurable logic is directly attached to the AMBA interconnect, and does not access the L1/L2 cache memory hierarchy included in the PS cluster, the DDR4 data path is not capable of automatically improving data locality. As such, it is required to manually cache data on die in the BlockRAM to hide latencies when accessing the external DDR4 memory.

An alternative third option was also considered, which consisted in performing multiple kernel launches of the `LBP_cascade_evaluation` kernel using as an input an OpenGL ES mipmapped texture. This texture was automatically generated by the ARM Mali 400 GPU (see Figure 4 - GPU debugging of the image pyramid texture mipmap generation) from the input video frame. However, this option was later discarded, as it required too many kernel launches, and the kernels computing the smaller mipmap scales yielded a very low occupancy of the DSP units thus decreasing the overall throughput. On top of that, it also degraded the performance of the face classifier, as it only generated the 10 scales offered by the standard OpenGL ES mipmap specification.

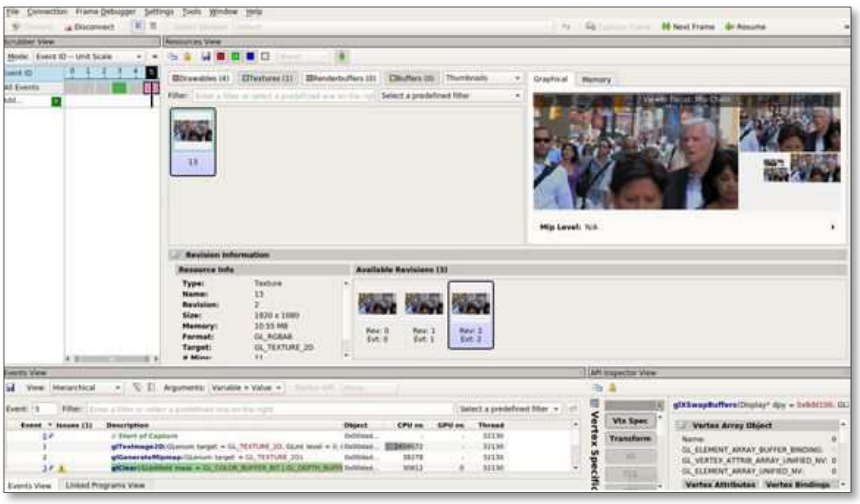

Figure 4 - GPU debugging of the image pyramid texture mipmap generation

Therefore, the AXIOM board implementation of the LBP kernel manually cached on die the `LBP_SCORES`, `LBP_THRESHOLDS`, and `LBP_FILTERS` arrays (see Appendix in Deliverable D3.2) containing the boosted cascade of classifiers. Similarly, the image pyramid working set was also split into

`NUM_BLOCKS` chunks and stored in the BlockRAM. Finally, FPGA task execution was scheduled for execution by annotating the kernel with the following simple OmpSs@FPGA directives:

```
#pragma omp target device(fpga) copy_in(width[0:0], height[0:0]) onto(0,1)

#pragma omp task
```

At the end of the code, the `#pragma omp task` directive was used to wait for the execution of the kernel and pending memory transfers containing the coordinates of localized faces. Unfortunately, since the cascade evaluation algorithm uses a `break` statement to quickly discard image regions, Vivado HLS suffered from optimization issues and required additional `#pragma HLS tripcount` and `#pragma HLS PIPELINE II=1` hints to improve the execution performance.

An additional wrapper function was developed to register the variables used as arguments for the kernel, and the internal buffers to automatically conduct XDMA memory transfers. These transfers were performed at the low-level by the `xdma.ko` Linux kernel module developed by BSC (see Figure 5), and also leveraged BSC's runtime libraries `libxdma.so` and `libxtasks.so`. The abovementioned variables were registered by calling `xdmaGetDMAAddress()` and `xtasksAddArg()` functions for each of the internal buffers and kernel arguments.

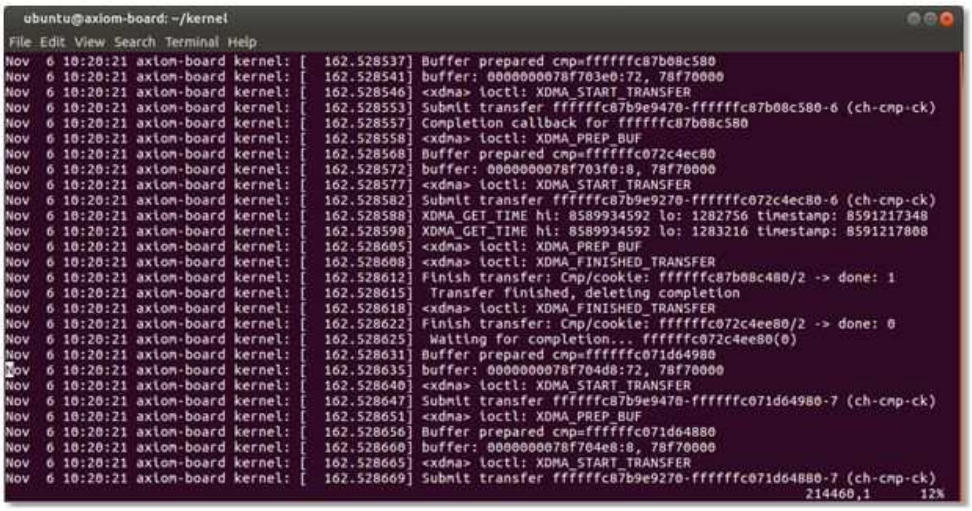

Figure 5 - XDMA Linux kernel module performing CPU/FPGA memory transfers during FPGA LBP kernel execution.

As the Paraver [9] profile trace shows in Figure 6, our implementation performed the FPGA execution of the LBP kernel tasks in a serialized manner, and therefore did not overlap computations with DMA memory accesses. Also, it did not speculatively exploit any other kind of intra-frame parallelism to further increase throughput. The reason of this limitation was due to the interaction between the threading mechanism of the UI Qt framework libraries, and the integration with the underlying OmpSs@FPGA and related runtime libraries, which would have required adding large mutual exclusion regions to avoid crashing the SVS application.

Similarly, Figure 7 shows a zoomed in version of the former trace, in which CPU/FPGA DMA transfers, CPU threads, and executions to the FPGA LBP kernel accelerator are highlighted. The latency observed between the FPGA accelerator calls were due to the time spent by CPU threads unpacking, demuxing, and decoding H.264 slices plus the computation required for constructing the image pyramid.

It should be noted that LibAv, Qt and OmpSs relied on their own customized thread scheduling mechanisms and thread pooling data structures. Therefore, the only way for successfully integrating them was to serialize OmpSs@FPGA kernel launches.

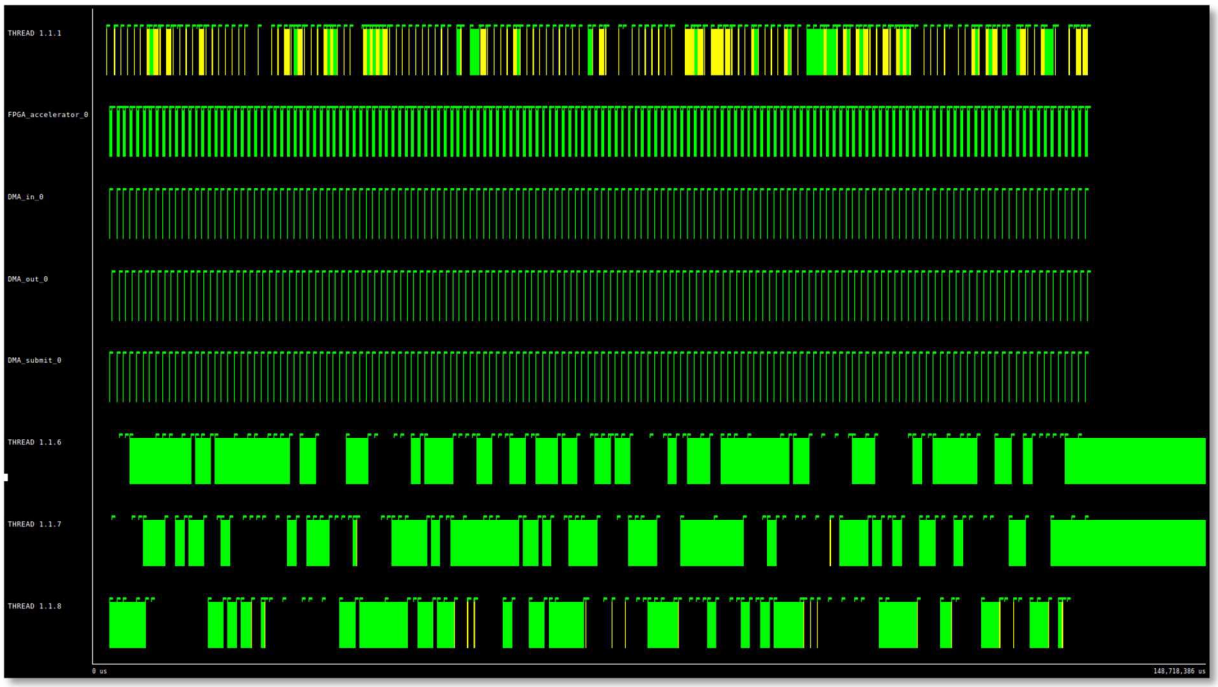

Figure 6 - General overview of Paraver showing a profile trace of the LBP kernel FPGA accelerator performing executions on the UltraScale+ reconfigurable logic.

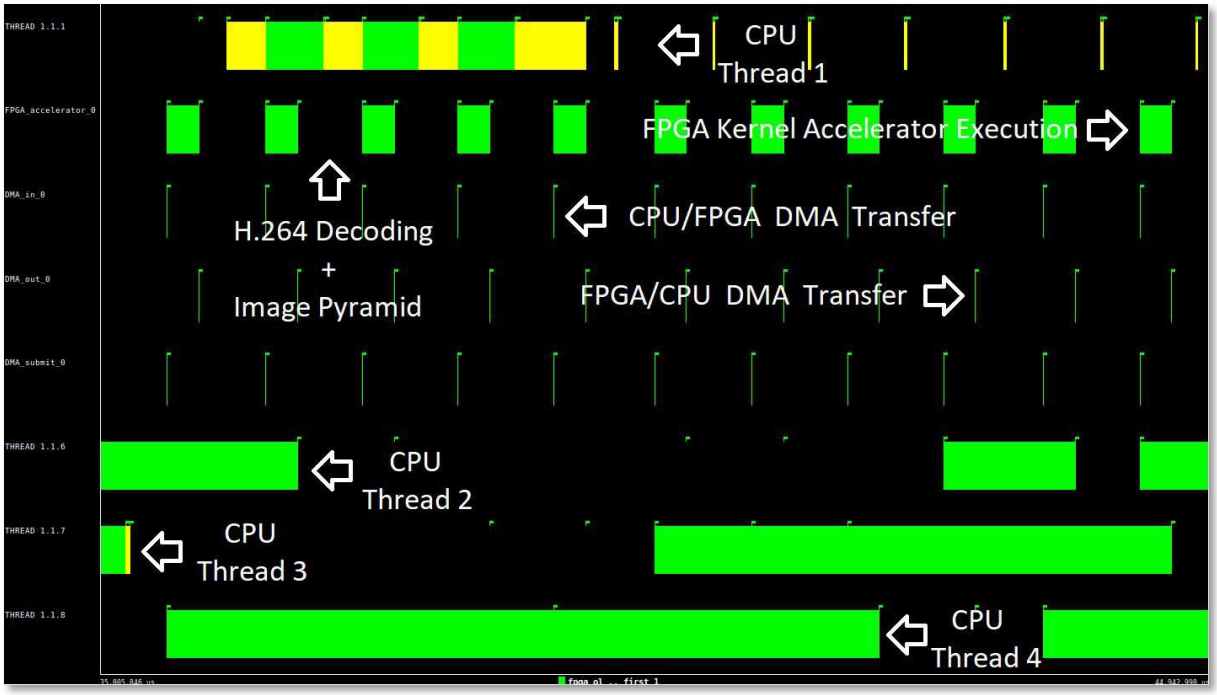

Figure 7 - Zoomed in Paraver view of the previous LBP kernel FPGA accelerator profile trace

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	43	-
FIFO	-	-	-	-	-
Instance	1380	8	10160	12784	-
Memory	0	-	128	4	-
Multiplexer	-	-	-	545	-
Register	-	-	1311	-	-
Total	1380	8	11599	13376	0
Available	1824	2520	548160	274080	0
Utilization (%)	75	~0	2	4	100

Figure 8 - FPGA utilization of the LBP kernel on the UltraScale+ ZU9EG MPSoC.

However, data parallelism was partially exploited within the LBP kernel by Xilinx's HLS compiler, which automatically generated the FPGA bitstream from the C code already parsed and extended by BSC's Mercurium `fpgacc` compiler. As Figure 8 shows, the generated bitstream clocked at 300 MHz consumed 8 DSP48E units. Unfortunately, this means that only 8 of the 2520 available DSP48E units on the ZU9EG chip were used by the FPGA kernel bitstream during execution. As such, there is still room for further improvement by increasing the utilization of the FPGA resources.

Since one of the main objectives of OmpSs@FPGA was to completely avoid dealing with low-level digital design at the HDL level, it was difficult to increase the utilization of the FPGA computational resources taking into account that logic synthesis, placement and routing algorithms behaved as black boxes, and thus were uncontrolled variables not targeted by the AXIOM toolset. Additionally, the irregular nature of the nested `for()` loops of the LBP face classifier proved to be a major challenge for the automatic C-to-HDL logic synthesis algorithms.

Another kernel that was parallelized using `#pragma` annotations and the OmpSs@FPGA toolchain was the matrix multiply kernel. This particular kernel is used for performing the convolutions required for the CNN models trained for demographics estimation. An extensive analysis of such parallelization is presented in Deliverable D4.2 and D4.3.

In the particular case of the `YUV_to_RGB` kernel, even though it was annotated and benchmarked with OmpSs@SMP directives (see Figure 8 included in Deliverable D3.2), we found that it was more worth to perform the color space conversion on the ARM Mali GPU rather than offloading these computations to the reconfigurable logic. Typically, OpenGL ES 2.0 deals with different texture formats and automatically performs channel and color space conversions when encoding/decoding pixel formats for GPU rendering. Therefore, the latencies derived from OmpSs@FPGA kernel launches, internal data structure allocation, and memory transfers (i.e. CPU/FPGA and then CPU/GPU for display) made the low-level FPGA offloading a highly uncompetitive solution for improving the performance of color space conversion in the SVS application.

Finally, a detailed analysis of the aggregated power consumption of the SVS application was conducted when performing face analysis on several videos. The gathered results are shown in detail in Deliverable D7.3.

3 Smart Home Living Service Porting

3.1 SHL Application

The *AXIOM_Bio_Identification* application developed by VIMAR during Task T3.2 analyzes a multimedia stream usually broadcasted by TCP/IP through the Ethernet network, and identifies users located in front of the audio and video recording devices. The user identification task is done by combining a cascade of several algorithms, which were presented in Section 3.1 of Deliverable D3.2.

The output of the application is displayed on the Linux console and stored in a log file. The output reports the result of all identification tasks, thus summarizing the following information:

- Presence/absence of human eyes in the input video stream
- Presence/absence of human voice in the input audio stream
- Iris ID stored in the database, if the video frame contains the iris of a known user
- Speaker ID stored in the database, if the audio trace contains the voice of a known user
- Data consistency statistics of features extracted by both iris and speaker recognition engines

The voice and the iris models/templates of known users are generated by the *AXIOM_Speaker_training* and the *AXIOM_Iris_training* applications, respectively. These applications were developed by VIMAR during Task T3.2. The main purpose of such applications is to generate iris and voice models of known users and to store them into the *AXIOM_Bio_Identification* databases.

3.2 Porting of the Application to AXIOM board

The *AXIOM_Bio_Identification*, the *AXIOM_Speaker_training* and the *AXIOM_Iris_training* applications were developed on x86_64 PC workstation with Ubuntu Linux 16.04.1 operating system. Such applications are based on several open-source libraries and multimedia frameworks, and were initially compiled targeting the x86_64 architecture. This external software is included and linked to the three applications at compile time. The main libraries used in VIMAR applications are OpenCV [10], ALIZE [11], SPRO [12] and WebRTC [13], while the multimedia framework used is GStreamer [14], which was enriched with a collection of plug-ins. These applications were tested on a PC using an input stream broadcasted from a video surveillance IP Camera.

The three main kernels that emerged as the most time-consuming parts of the SHL application are the following: *Feature extraction module*, *Anisotropic smoothing module* and *Iris recognition module*. These kernels were isolated, profiled, and initially annotated using OmpSs@SMP directives. The results were reported in Section 3.5, 3.6 and 3.7 of Deliverable D3.2. Moreover, further advanced profiling and optimizations were extensively studied and, as such, the full source code of these modules was shared with both BSC and UNISI partners, which proposed additional strategies to increase the throughput and reduce the execution latency of the selected kernels. The toolset used for conducting these experiments consisted in the Mercurium compiler [15], the Nanos++ runtime, and Extrae library [16] for generating Paraver traces.

In order to test, validate and profile our code in the initial AXIOM Evaluation Platform (AEP), all applications, libraries, and frameworks were compiled using the Mercurium compiler. The final binary was linked with the OmpSs runtime system, and targeted the ARMv7 Xilinx ZC706 board. The results obtained initially were presented in Deliverable D3.2, where several benchmarks depicting the execution time of the three main modules were briefly discussed. As in the case of PC experiments, the

experiments conducted on the AEP platform also relied on video streams broadcasted from a remote surveillance IP-based camera.

Finally, the three VIMAR application modules were natively compiled targeting the ARMv8 of the AXIOM board. For the experiments on the AXIOM board, we relied on the libraries and GStreamer framework customized by partner SECO, which were included in version 1.3 of the AXIOM filesystem image. The remaining required libraries were compiled from scratch targeting the ARMv8 architecture of the ARM Cortex A53 cores of the AXIOM board. As in the case of the Zynq7000, the applications were also tested using a video surveillance IP camera. The three selected kernels were profiled and its corresponding speed up analyzed after having enabled/disabled the OmpSs@SMP annotations on both Zynq7000 and the AXIOM board. The obtained results are depicted in Figure 9 included below.

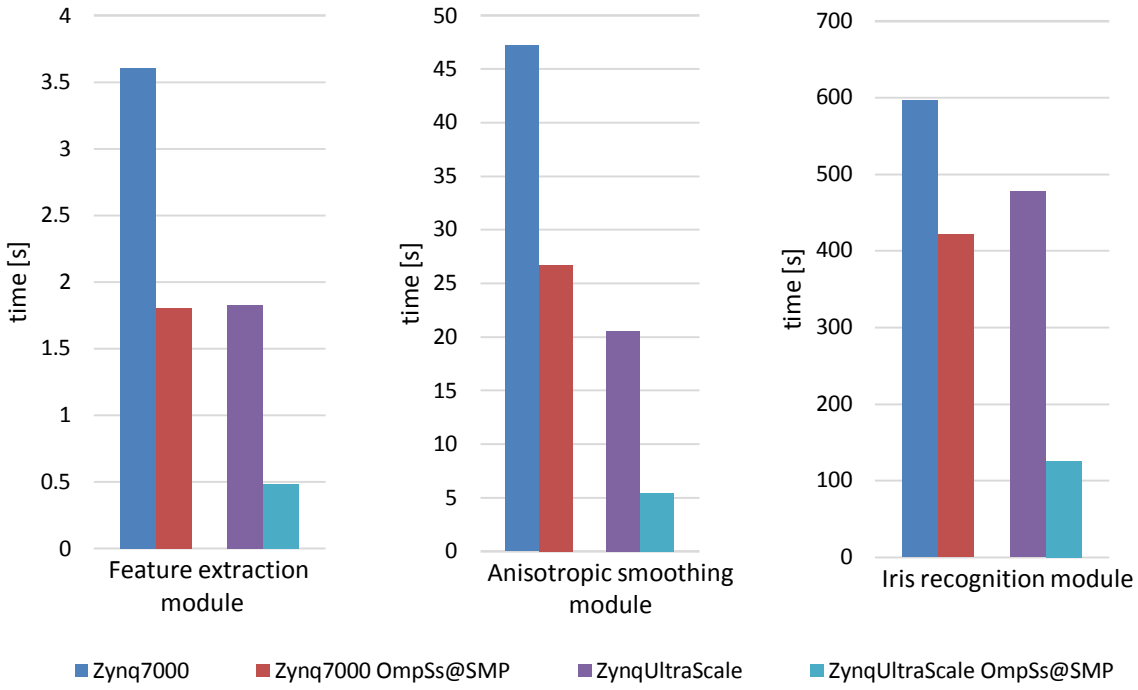

Figure 9 - Execution time of the three SHL application kernels on the AEP and the AXIOM board. The results depicted above were obtained by manually enabling and disabling OmpSs@SMP directives.

As Figure 9 shows, the speedup obtained was close to 2x when the kernels were executed on the AXIOM board versus the executions performed on Zynq7000 (blue and violet columns). These results are in line with the increased frequencies, and instruction issue mechanisms of the ARM Cortex cores included in the UltraScale+ chip of the AXIOM board.

Furthermore, the obtained results show an average speedup of 3.8x when the selected kernels were executed on the AXIOM board using OmpSs@SMP (violet and turquoise columns). This speedup is related to the increased core counts of the MPSoC included on the AXIOM board (i.e. four ARM Cortex A53), which are more advanced than those included in the SoC used on the AEP (i.e. two ARM Cortex A9). As it was originally reported in D3.2, the speedup achieved using OmpSs@SMP on the Zynq-7000P platform equaled 1.8x (blue and red columns) on the Zynq7000. As such, the results obtained demonstrate that the OmpSs@SMP porting from the initial AEP platform to the AXIOM board was correctly done, and it also substantially benefited from exploiting the underlying resources of a more powerful MPSoC.

After the porting and evaluation of the kernels to the AXIOM board, the *AXIOM_Bio_Identification* application was ported to the final board. Figure 9 shows the generated trace of the execution of the application on the AXIOM board. In the shown case, only the OmpSs@SMP directives of the *Iris Recognition Module* have been introduced in the application.

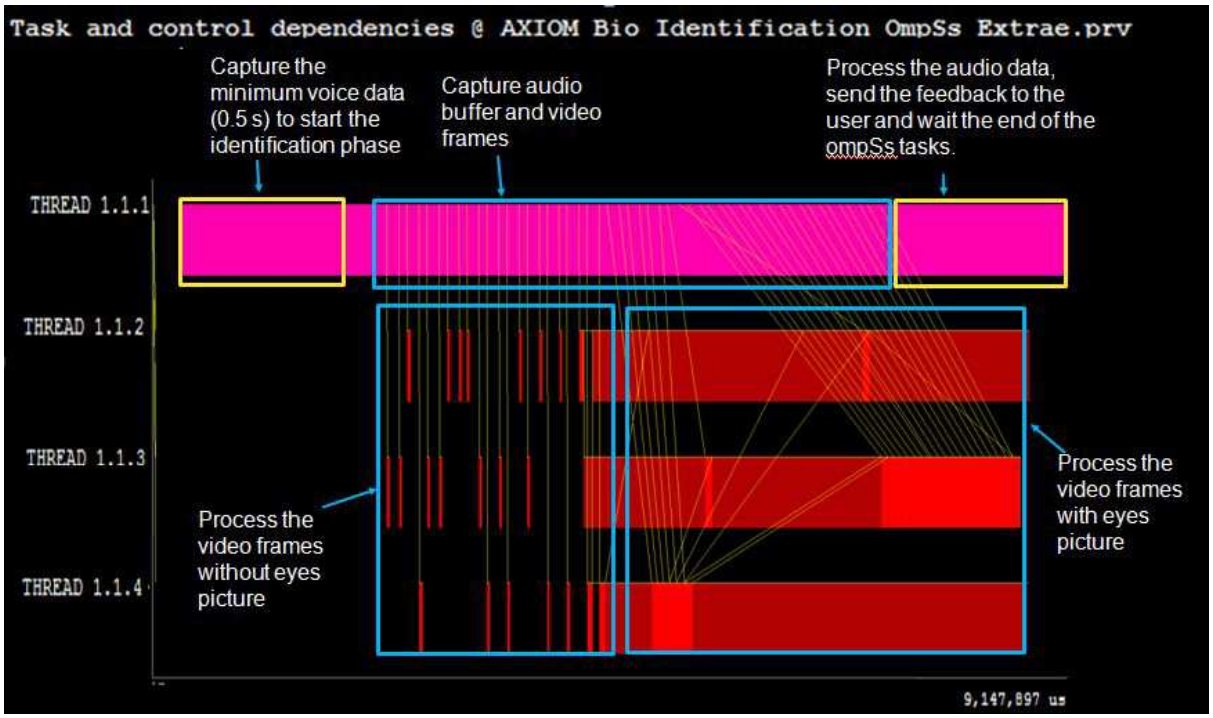

Figure 10 - Paraver trace of the AXIOM_Bio_Identification application on the AXIOM board.

Additionally, Figure 10 shows the usefulness of exploiting task parallelism in a complex sequential application using OmpSs to significantly reduce the execution time. The location and size of tasks (depicted in dark and light red colors) are related to the input video stream that is received from a remote surveillance IP-based camera. The trace shows the main application phases. In the first phase the application records and processes only the audio data, it waits the minimum voice signal to start an identification. In the second phase the application records audio and video buffers and it schedules tasks to process the frames, if no eye is detected inside the frames then the frames are discarded, otherwise they are deeply analyzed. In this phase the iris recognition is done. In the third phase of the application the speaker identification is executed with the audio recorded and the result of iris and speaker identification is sent to the user. The application waits the end of all the tasks scheduled before restarting the detection.

In order to efficiently exploit OmpSs@FPGA for offloading computations to the FPGA fabric, substantial efforts were carried out to cross-compile the SHL application and kernels to the ARMv7 and ARMv8 architectures starting from the original x86_64 implementation. These cross-compilations were required by the Xilinx high-level C/C++ synthesis tools, which were internally invoked by the Mercurium compiler to generate FPGA bitstreams. These cross-compilations were implemented by developing from scratch several scripts, and by later including the required platform-specific code in all *Makefile* components. The final binary building process was extensively tested on both ARMv7 and ARMv8 architectures.

In contrast with the initial evaluations, the execution time of the *Feature extraction module* on the AXIOM board using OmpSs@SMP directives was considered negligible compared to the execution

times of the modules related to the image and video processing. This consideration focused our work to porting on the FPGA fabric the video processing tasks.

The workflow of the algorithm and the size of the input and output data arrays were the main properties that allowed to obtain an effective porting of the *Anisotropic smoothing module* to the FPGA resources. The algorithm implemented in this module consists of a loop with *m* iterations in which all image pixels are read and written several times. In all iterations, the algorithm applies a stencil computation on each pixel, using the neighboring pixels. This module processes a set of regions of interest (ROIs) of the input video frames defined by the iris recognition algorithm. In the *AXIOM_Bio_Identification* implemented, the maximum area of these ROIs is 260x260 pixels. This limited size enables to develop an OmpSs@FPGA accelerator in which all the pixels of the ROI are saved inside the FPGA BlockRAM resources. The OmpSs@FPGA directives were introduced in the *Anisotropic smoothing module* to target the OmpSs tasks to the FPGA accelerators of the AXIOM board. As such, the Mercurium compiler was used to generate the binary `elf` file targeting the ARMv8 cores, and the proper bitstream to program the FPGAs. The module required an extensive code refactoring in order to properly synthesize the FPGA kernel accelerators from the C/C++ code implementation due to the restrictions imposed by the Xilinx HLS compiler. Low-level optimizations were also investigated by annotating the source code with the HLS directives parsed by Vivado HLS.

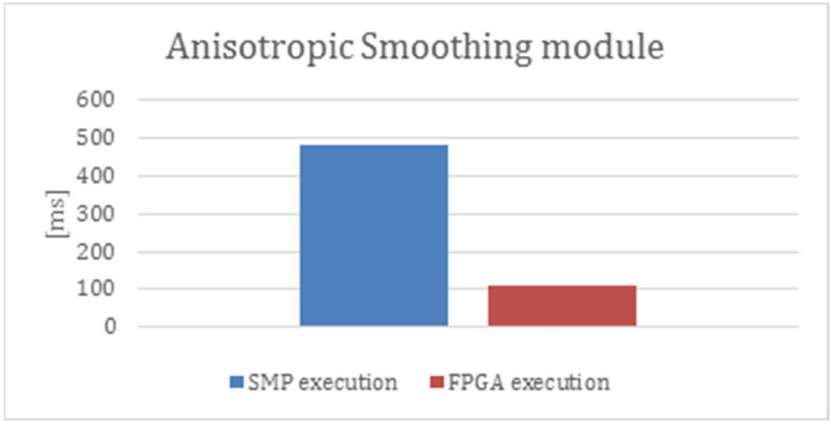

Figure 11 - Execution time of the module in the AXIOM board when the algorithm was processed on the SMP core and on the FPGA accelerator.

Resource	Utilization	Available	Utilization %
LUT	74392	274080	27.14
LUTRAM	1469	144000	1.02
FF	44312	548160	8.08
BRAM	235	912	25.77
DSP	12	2520	0.48
IO	25	204	12.25
BUFG	1	404	0.25

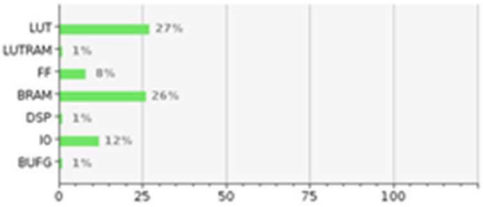

Figure 12 - FPGA resources utilized by Anisotropic smoothing module on the AXIOM board.

Figure 11 shows the execution time of the *Anisotropic smoothing module* on the AXIOM board when the algorithm processes an input ROI of 260x260 pixels, and compares both SMP and FPGA executions. On the other hand, Figure 12 summarizes the FPGA resources consumed by the *Anisotropic smoothing module* in the Zynq UltraScale+ ZU9EG MPSoC.

The *Anisotropic smoothing module* with the OmpSs@FPGA directives was embedded in the *Iris recognition module*. The iris recognition algorithm processes all input video frames, and for each eye that is found on a given frame, it executes four times the *Anisotropic smoothing module*.

In order to speed up the SHL application, OmpSs@SMP directives were added to the *Iris recognition module*. The tasks created with OmpSs@SMP opened the door to the parallel processing of several eyes recorded in the input video frames. Additionally, the OmpSs@FPGA directives used in the *Anisotropic smoothing module* enabled us to execute part of the data processing on the FPGA accelerator while instantiating several bitstream accelerators on the same device. More particularly, the low resource utilization of the FPGA attributed to this kernel enabled us to synthesize and exploit two simultaneous instances.

Both OmpSs@FPGA and SMP directives were added to the final SHL application, which was also profiled for estimating its power consumption. The results of these experiments are reported in Deliverable D7.3.

3.3 Porting the system to the Video door entry system

To enable the User Experience evaluation of the *AXIOM_Bio_Identification* application on a real environment, the AXIOM board was interfaced with the ELVOX Door Entry IP Pixel (VIMAR product 41006 code) and the Multimedia Video Touch Screen 10in IP (VIMAR product 21553.2 code). Figure 13 shows the VIMAR demo system.

The ELVOX Door Entry IP Pixel is the new generation of video door entry device produced by VIMAR and it is used to record the audio and video data that is processed within the *AXIOM_Bio_Identification* application, which targets the AXIOM board. The software of the ELVOX Door Entry IP Pixel device was conveniently modified to send multimedia streams, and also to receive control commands from the *AXIOM_Bio_Identification* application. In order to reach a good degree of operation of the iris recognition algorithms, both the lens and the LED used to illuminate the face of the person in front of the camera were appropriately modified. The algorithms and input parameters were also fine-tuned in order to adapt the iris recognition algorithm to the application. As such, the system was initially adjusted to process a video stream coming from a Full-HD video surveillance camera, and later downsampled to 720x578 pixels, which was the input resolution of the video door entry system. To reduce the distortion introduced by the lens and increase the iris recognition algorithm reliability, an operation of rectification of the images coming from the video device was added in the *AXIOM_Bio_Identification* application.

Finally, a minimal user interface was also developed to simplify the interaction with the *AXIOM_Bio_Identification* application. The interface has the goal to show the state of the process and the results of the identification and give to the user the possibility to start the iris and voice training operation. The interaction of the user with the application is performed using a touch screen, which enables the redirection of the input and the output of the users. Figure 14 shows three screenshots of the minimal user interface.

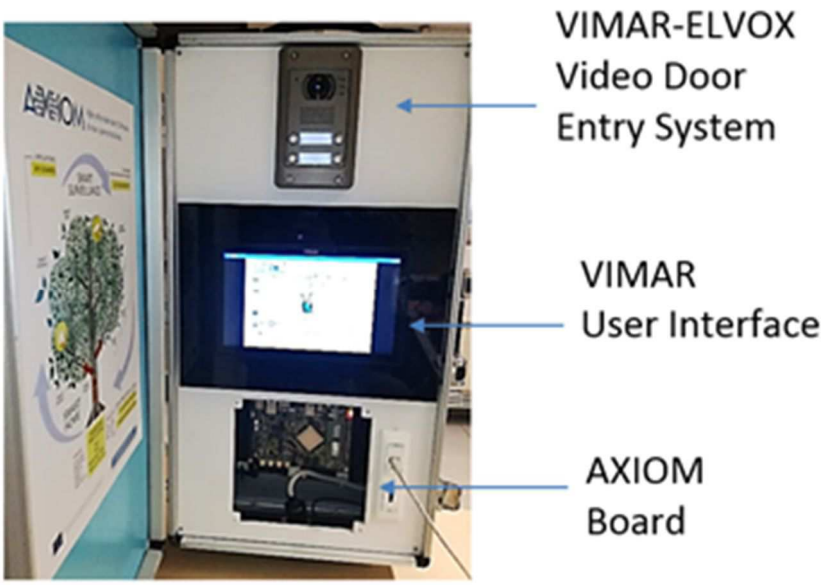

Figure 13 - VIMAR demo system. Of the SHL application

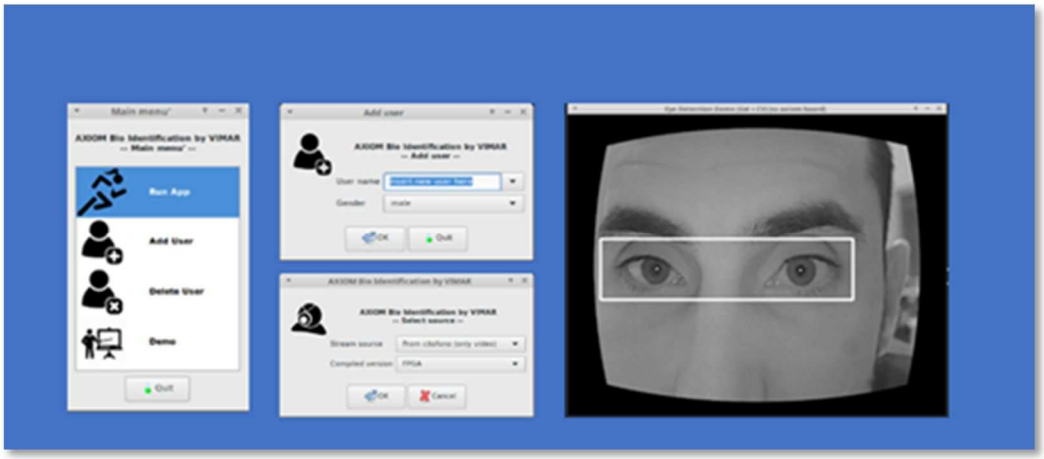

Figure 14 - Screenshots of the minimal user interface of the AXIOM_Bio_Identification application. The screenshots show the main menu of the application, the "add user" menu, the" running menu" configuration, and the "running window".

4 User Interface Design and Testing

This section reports on the testing of the envisioned App/Services based on the AXIOM CPS [17]. The testing was carried out using the Scenario-Based approach, using the same scenarios produced in D3.2 plus the new scenarios produced with the help of the end users. The conducted test adopted the 5-80% approach (5 users are typically sufficient for detecting 80% of problems) [18]. The goals of the testing procedure were mainly oriented to the assessment of the role the new services would play in the life of the end-users. For this reason, we implemented the services in the form of interactive prototypes that can allow the exploration of possible implications of the new services as described in the proposed Scenarios. The *mis-en-scène* of the scenarios performed by the end-user by means of a *Cognitive Walkthrough* also allow defining user performance metrics based on interactions breakdowns, establishing a baseline of user performance, and identifying potential design concerns to be addressed in order to improve the efficiency and end-user experience [19] [20].

More into detail, the test objectives are:

- To determine experience acceptance and usability problems in the user interface and content areas expressed as breakdowns in the interaction. Potential sources of breakdowns may include:
    - Navigation breakdowns – e.g. failure to locate functions, excessive keystrokes to complete a function, failure to follow recommended screen flow.
    - Presentation breakdowns – e.g. failure to locate and properly act upon desired information in screens, selection errors due to labeling ambiguities.
    - Control usage breakdowns – e.g. improper toolbar or entry field usage.
- Establish baseline user performance and user-satisfaction levels of the user interface for future development.

At the end of each sessions, the participants received a session of debriefing where they were free to express their concern and/or interest for the services/application they explored through the *mis-en-scène* of the scenarios.

4.1 Smart Video Surveillance Scenarios

Many physical store retailers are now installing surveillance cameras throughout their stores. These cameras are used to track customer traffic during shopping trips, and also to record in detail what any given customer was doing, where the customer goes in the store, how long the customer stays in a particular place, where the customer goes next, and so on. But beyond simply recording customer movement and activity, the cameras themselves are part of an environment that is equipped with enough computing power and storage to analyze all of those data and then, produce insights about the customers drawn from all of those data, along with specific recommendations designed to promote the customer shopping experience and ideally, increase profitability. The potentialities of the AXIOM CPS fit well with this edge computing scenario.

One use case would be sending customer specific and also shopping trip specific recommendations to a store salesperson, who would then wander over to the customer. Basically, the salesperson would intercept that customer and, armed with the knowledge of what that customer has been doing on the particular shopping trip, enable a much more tailored sales conversation. Or if the customer's identifiable to the analytical environment that's located at the edge, let's say through some sort of frequent customer card that was swiped on the way into the store, or perhaps through the GPS of the customer's phone or maybe through facial recognition, the system could then send a very tailored text message to

the customer, something along the lines of, "*If you're looking for more workout gear, you might try our new arrivals section at the front of the store*". Tailored one-to-one sales-oriented insights such as these are not new.

With edge analytics, all of these capabilities are recurring somewhere at the edge, on the fringes of the enterprise rather than within a centralized environment.

4.1.1 First Scenario

AXIOM boards are installed in a smart mall to capture video and images regarding the behaviors of visitors inside the building. The AXIOM cameras use a face recognition system to monitor and identify people inside the building. A desktop application allows the retailer to control and monitor all the data captured by the board. The main interface of the application shows the video real time streaming of all the people passing by the smart mall entrance and walking inside all the different areas and corners of the shops. In this way retailers have control on the AXIOM board activities and can understand the distribution of the crowd, the interests and the behavior of visitors. The camera identifies the customers and sends messages about their identities: age, the ethnicity and the gender of the people in the video. The user can monitor in real time all the shopping corners of the smart mall, and then have an instant insight on the customers that are inside the building and understand which action or which shopping activity they are doing. The retailer can understand which area is more crowded or captures the interest of visitors. According to the data visualized and captured by the streaming video, the retailer can modify specific shopping corner or decide security actions.

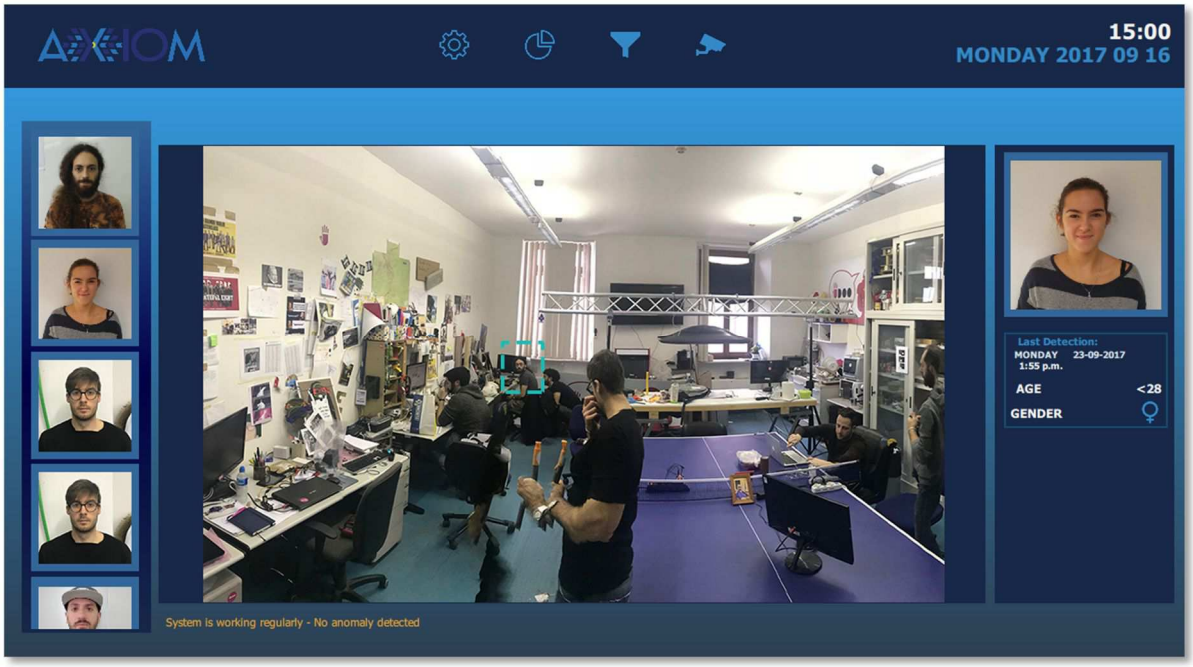

Figure 15 - The screenshot of the main screen of the user interface for the AXIOM SVS application. The image shows in the middle of the screen the window of streaming video, the main menu navigation on the top and the two sidebars with widgets for the face recognition history and archive.

4.1.2 Second Scenario

The retailer can select one or more subjects identified by the smart AXIOM system and save them in the History area on the right. This allows to compare shopping behavior when a client is alone or is

doing shopping with friends or relatives; if the shopping is for her/himself or for others, what kind of interactions occurs among them. A simple drag and drop enable the process of grouping these contacts inside the archive shared by all the administrators and users of the system (see Figure 15). The archive is useful to recall info about specific subjects for security purpose or to trace the customer behavior.

4.1.3 Third Scenario

In the third scenario, the AXIOM application enables retailers to monitor the client behaviors inside the smart mall, and make data analysis to decide the storage in order to perform marketing actions.

The retailer can match info about biometrics analysis with client behaviors. Client behavior is classified according three archetypes: *The ant*: "ant" clients are those who follow the path proposed by the shopping mall, taking time to observe most of the exposed objects. They stop frequently and the overall visit is long in time. Ant visitors usually move close to shelves and avoid empty spaces. *The butterfly*: the "butterfly" performs a sort of "pendulum" visit. They frequently change the direction of visit, moving from the shelves to shelves without following the proposed path. The visit is mostly guided by the "affordance" of the elements in the physical space. *The grasshopper*: the "grasshoppers" see only items they are interested in, without following the proposed path. The visit is mostly guided by personal interests and pre-existing goals about the elements of the mall. The grasshopper crosses empty spaces, stops rarely, and the times spent to observe single selected items can vary according to the fitness of the item to the goal [21].

The filter tab of the interface efforts the user to filter all the data achieved and selects specific target (e.g. "*Women under 25 Black*"). Inside the Analytics tab the retailer can study the classification of a specific consumer behaviour. In this tab, it would be possible to match the data from the AXIOM camera with the shopping card of the customers in order to obtain a wide dataset of info.

4.2 Smart Home Living Scenarios

The more we order food, clothing, and household items online, the more we need security around our front doors.

A smart doorbell might sound silly–you still have to get up to open the door even if the doorbell tells you who's there. Besides, knowing your friend has arrived is what texting is for. Yet as reported by Google smart doorbell *"...is the product that customers have been asking us for. It's our No. 1 most requested product.*"

And it is easy imagine why. Existing security cameras often give an eagle's eye view of a home but overlook the last foot or so of your entryway. With the VIMAR envisioned doorbell, we have a security camera and home assistant in one. Say DHL rings the doorbell to drop off a package, and you aren't home. You get a notification on your phone and can instruct him, either personally through doorbell microphone or through a pre-canned message, where to leave the package. Or consider a cleaning lady coming to your door, you can give access to specific time and person! The smart doorbell is going to be a key component of the smart home [22].

4.2.1 First Scenario

People interviewed expressed the need to open and close their home door without using a physical key. For example, users would like to be free to go out for running and take with them only a smartphone or even with nothing at all.

According to that evidence, in the first scenario we designed the AXIOM App to enable users to exit and enter their home using a digital key which can be activated or disabled through their mobile phone. The AXIOM App shows the state of the door directly in the Home screen interface. An icon indicates if the door is locked or unlocked, and the user can decide accordingly to that if he want to open or close and interact with it choosing to Lock or Unlock the door (see Figure 16).

In this scenario, the user can choose to open the door without the mobile application, just using the Iris and/or Voice Recognition System installed in the AXIOM Smart Video Door.

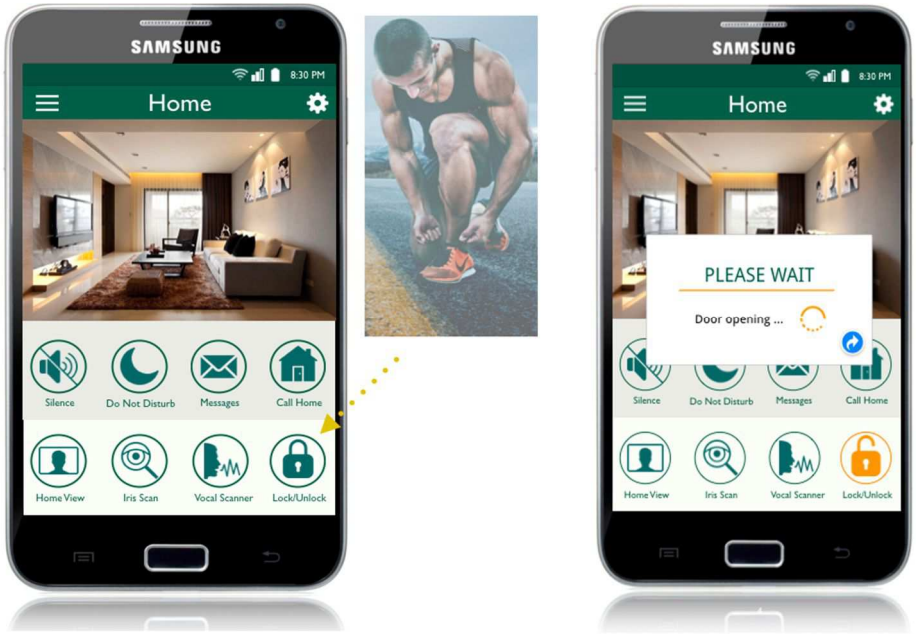

Figure 16 - The screenshots are about the mobile application. They show the set up and the interaction with the smart video door entry system. The two images show the action of unlocking the door with the smartphone app.

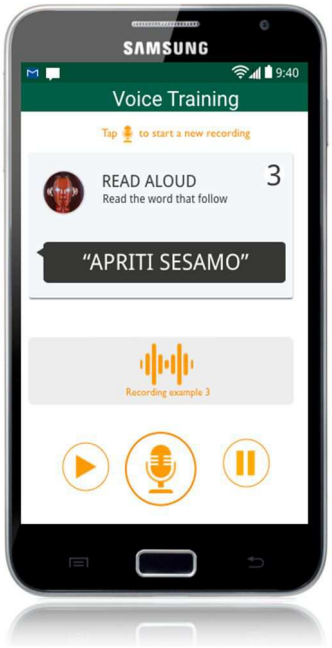

Figure 17 - The screenshot shows the UI for training voice recognition system from mobile application.

The system can be trained through the AXIOM App (see Figure 17) or by the device itself in order to recognize the voice and/or the iris of the home owner in order to let him enter the door using the Smart Video Entry System.

4.2.2 Second Scenario

In the Second Scenario users use the AXIOM App to select a list of contacts that can use the Access Key to enter the house without any further permission or physical key. The home owner can use the AXIOM App for Owners to enable one specific contact to access the building during a predefined period of time: from day X to day Y, from hour X to hour Y. The guest (e.g., *the baby sitter, a family member*) can use the companion AXIOM App for guests (see Figure 18) to train the AXIOM Smart Door System in order to be recognized by the Iris and/or Voice recognition system.

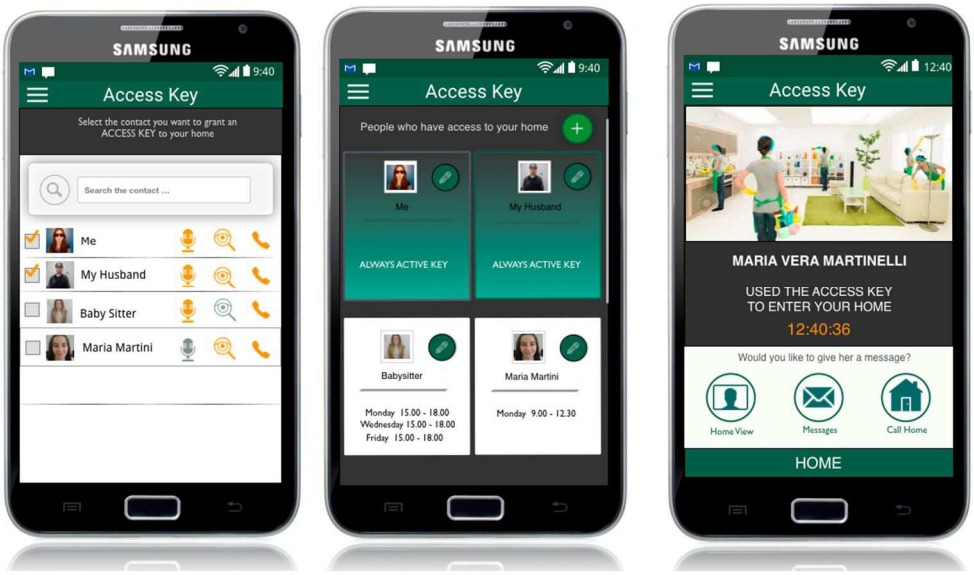

Figure 18 - The screenshots on the left show the UI for setting the access key for different users using mobile application. The screenshot on the right shows the alert sent by notification system to notify the owner that someone entered the house.

4.2.3 Third Scenario

The third Scenario widens the Smart Bell features to people who use services such as Airbnb or similar other platforms that enable people to rent a house/rooms or host guests for a defined period of time (see Figure 19). The user logged in the AXIOM App can monitor the destination of the incoming guests, and send the Access Key to a list of people with access granted to the building. The home owner can send a Digital Access Key to the next guest and grant him/her to enter the house during the stay. The guest can use the Access Key permission to enter the house at the day of arrival without contacting the owner or spending time to wait for the physical key.

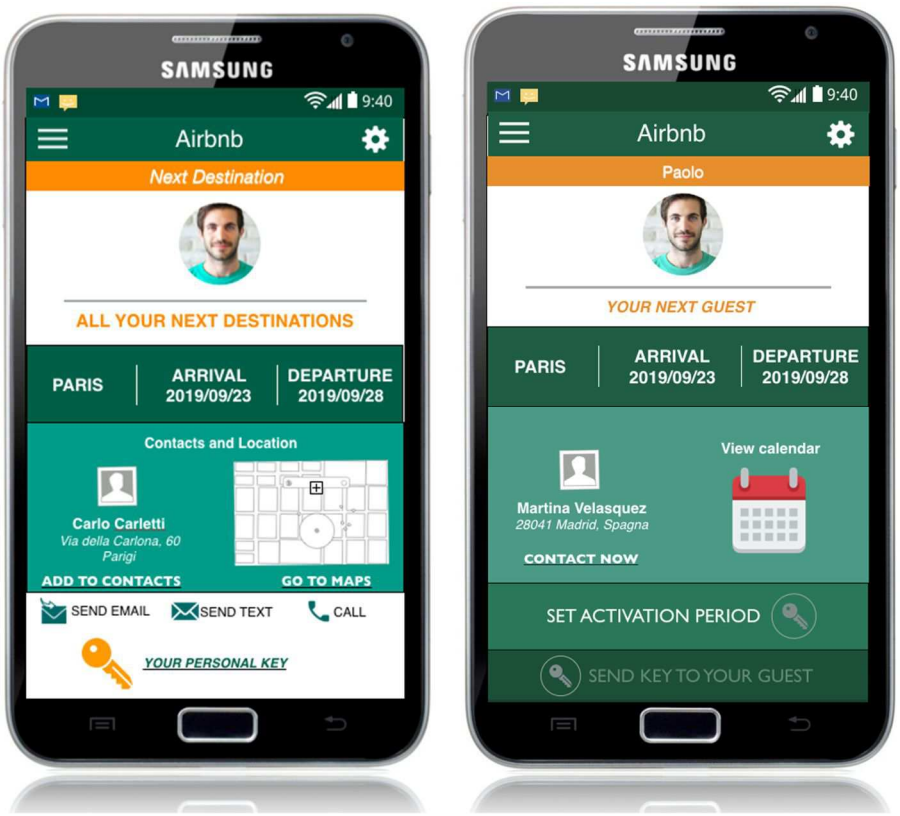

Figure 19 - The two screenshots are about the Airbnb scenario: an example of the UI for the guest-user who is going to rent a house (left), and the mockup of the interface for home owner who is going to guest someone in his loft (right).

4.3 Experience Testing

As it has been reported in the previous sections, the user experience testing was devoted mainly to the possible role the services would play in the life of the final users. The look & feel of the prototypes was mainly used to give the possibility to explore the implications of the envisioned scenarios and refine the possible interactions. The UX testing was conducted mainly inside the University of Siena involving five users [23] for each application. For both scenarios, SVS and SHL, people has been selected in order to engage five users sorted as follow: 1 stakeholder (Outlet VdC and Immobiliare.it), and 4 end-users with differences in gender, age and backgrounds.

At the beginning of each session, a facilitator introduced the AXIOM Project and briefed the participants asking them to evaluate the service and the related application. Then the participants were invited to complete a series of tasks and describe all their actions and thoughts talking aloud. During the tests all the participants described aloud their actions, misunderstanding, needs and general problem in reaching their final goal.

The final goal for each single task were the same for all the participants. During all the performance the facilitator observed participants and annotated information about their behaviors, interactions with the system and all the remarks expressed during the execution of the different tasks (e.g. overcoming the breakdown, commenting the interactions). As an output, we obtained a report of all the possible improvements and actual constraints related to the UI of the applications. The list of critical aspects was related to: navigation breakdowns, presentation breakdowns and control usage problems.

The final result is a completion rate in both applications testing of about 90% for each task: only in a few cases the participants were not able to overcome a given breakdown. All the scenarios evaluated were interesting and worthwhile to be implemented.

For the HERTA testing we asked the users to perform task related to the three main scenarios of the smart mall. We tested the task related to the main tab bar: LIVE streaming. For this particular tab, we noticed that the UI enabled users to easily monitor the path of customers inside the shops area and understand the navigation path through the tab bar menu and different windows. Some difficulties were related to the filter option.

In the first mockups made with Balsamiq [24] tool (see Figure 20), we designed the filters as a spinner to provide a quick way to select one value for each category of user properties.

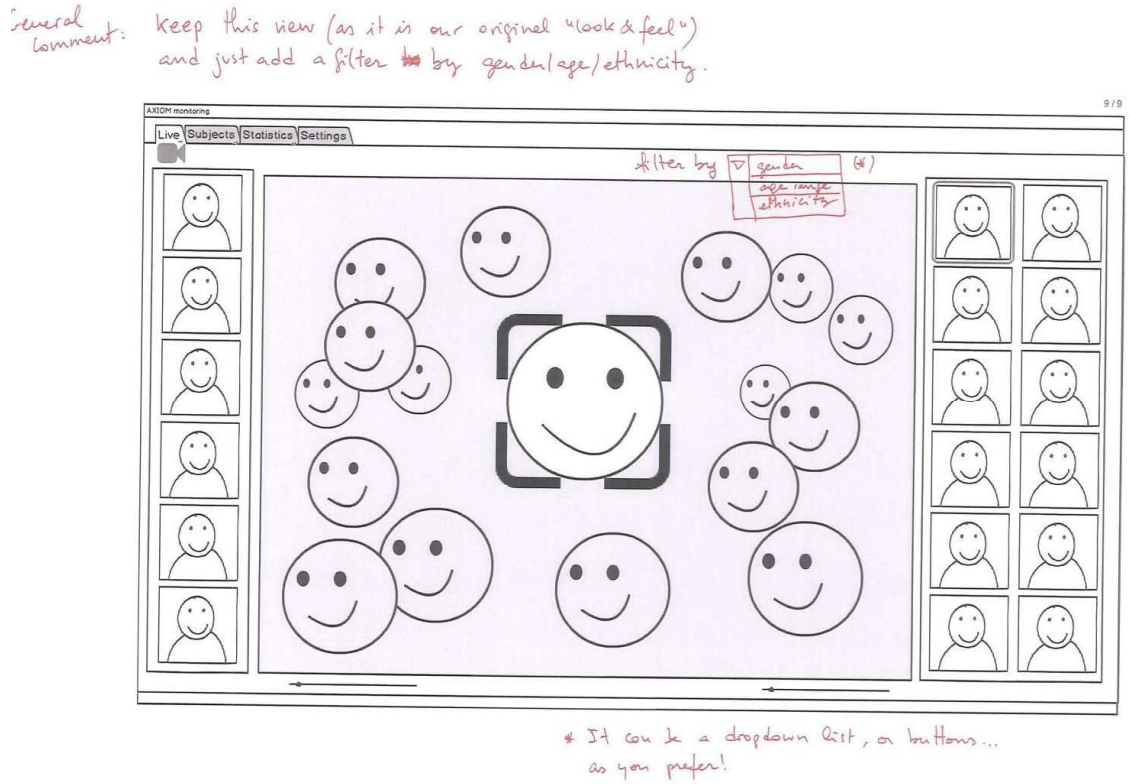

Figure 20 - Wireframe of the Home of the SVS application. On the paper was sketched the first idea to add a spinner to filter features of the users detected.

During the tests users were confused by all the elements in the screen view (see Figure 21) and often unable to predict how their filter selection will affect the resulting output.

As it shown in the images below, we redesigned the Filter selector options as a drop-down overlay menu (see Figure 22). In this way, all the contents which are not useful to complete the task of selecting a specific target of customer are covered and not completely visible to the user.

Figure 21 - Mockup of the main view of the SVS application designed with the Qt Quick tool. The spinner on the top-center of the screen is developed and tested by real users revealing problems of task execution.

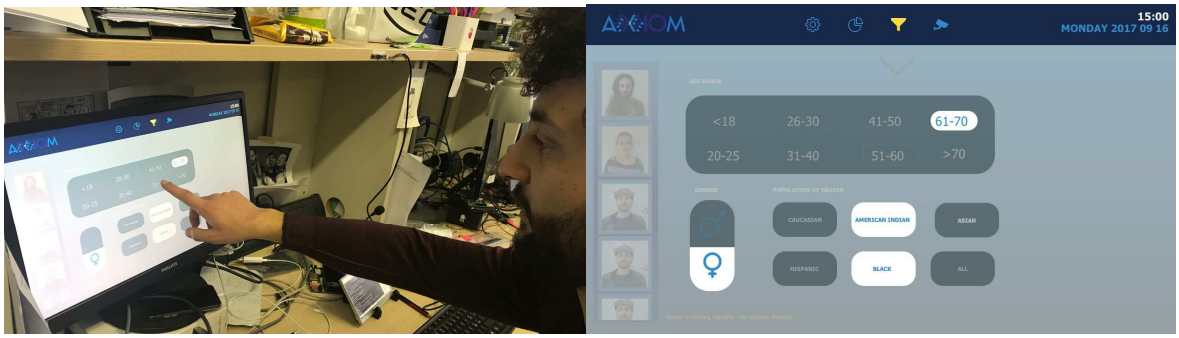

Figure 22 - The image on the left shows a user testing the new drop-down overlay for filtering targeted users using the SVS application for desktop. The image on the right is a screenshot of the UI tested.

This last version of the UI helped us on preventing errors, and guided users on selecting actions that will accomplish their current task. Additionally, we annotated more reviews about the UI (see annotations in Figure 23 and Figure 24).

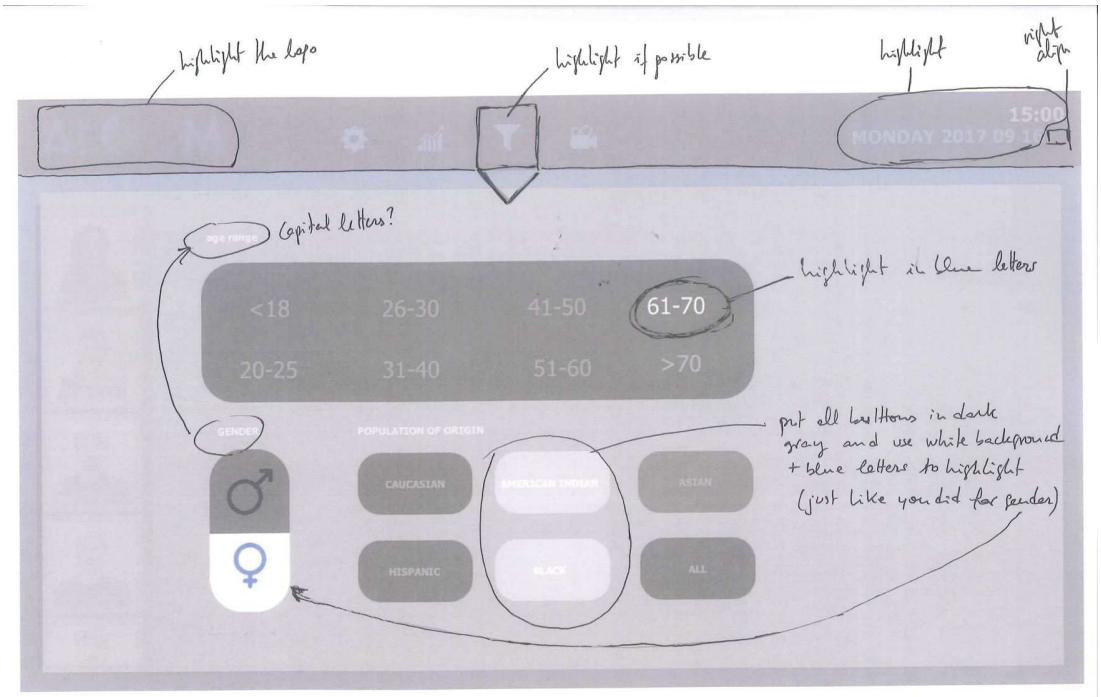

Figure 23 - The image shows the paper sketch and annotation of the troubles revealed during the testing and ideas for solving those problems.

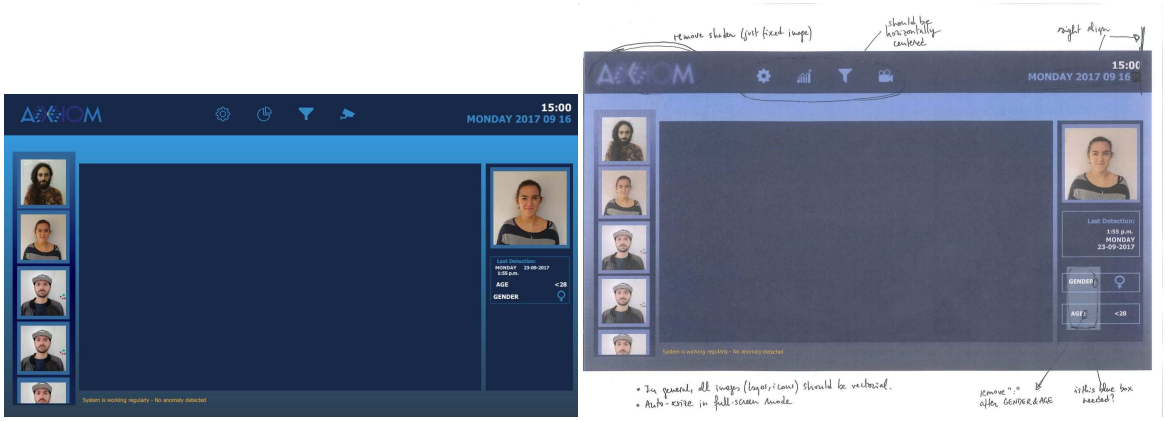

Figure 24 - The two images show the screenshots of the UI for the Home of SVS application and the sketched notes and reviews annotated after the testing session.

We tested also the Statistic Tab Bar (see Figure 25) asking users to analyze data about customers and explore information collected to extract unexpected or relevant patterns of shopping behaviors. We confirmed our expectations: users needed first an overview of all the data collected. Also, users filtered, and zoomed in the details to focus on specific information on demand. Therefore, we needed to avoid an information overload effect. In order to overcome this issue, we tested a series of possible solutions, which offered to end users the possibility to zoom in on items of interest or filter out uninteresting information. From the collected results, we concluded that users would enjoy understanding the relationship among different customer information.

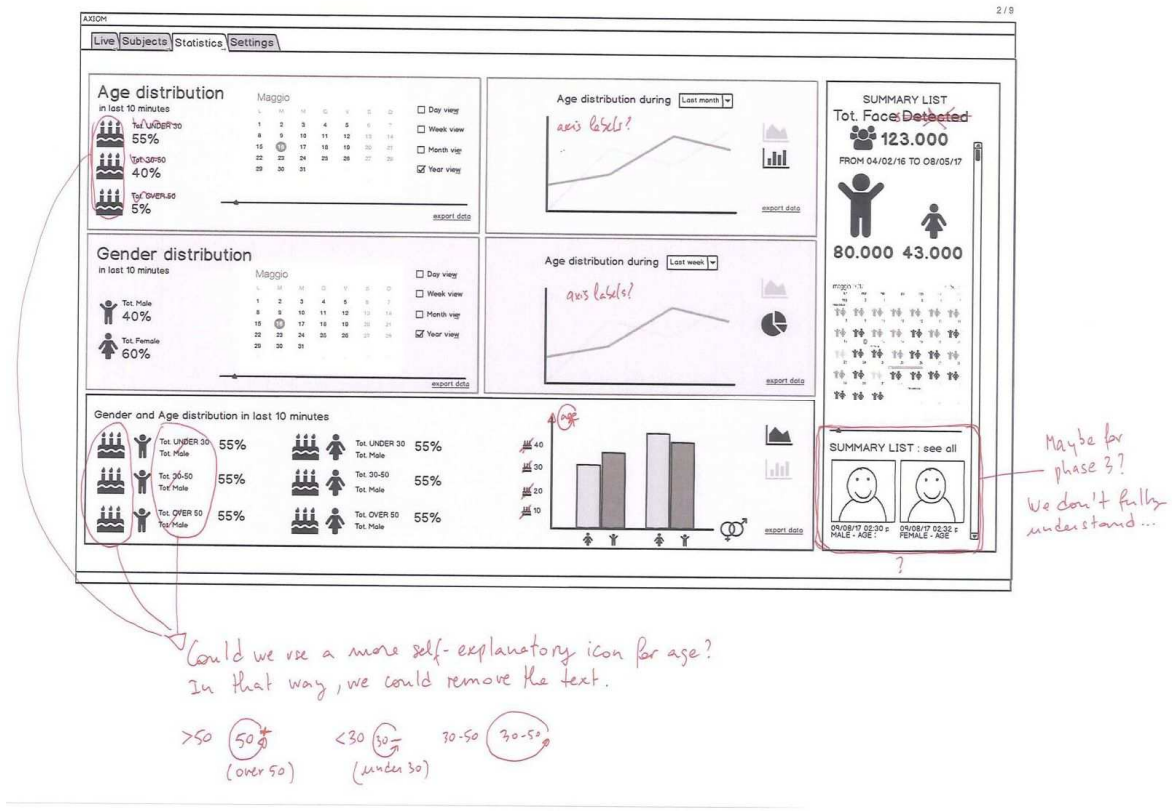

Figure 25 - The image shows the wireframe of the Statistic tab bar of the SVS application and the annotation for reviewing the interface and improve the usability in the second mockup.

The main breakdowns about the SVS application were the following:

- Navigation breakdowns - the first version of the main menu bring users to errors. The tab bar was redesigned as a menu with four icons corresponding to the four main windows of the application.

- Presentation breakdowns – in the statistic view was revealed some labeling ambiguities about the icons used to identify age range. Users were unable to select the correct items and filter age range of customers archived.

- Control Usage Problems - The classification of the client behavior according to three patterns was problematic since a given user can start *Grasshopper* a then become *Butterfly*.

Main concerns/interest:

- Concerns: It was not clear how to exploit all the potentialities of the service. The interface, even if it has been greatly simplified, is not yet intuitive for people with no-background in retail services.

- Interest: The potentialities seems amazing. The transformation of some user behavior from Grasshopper to Butterfly is a clear sign of success. The browsing of data according to the calendar and partnership is very intriguing.

For the SHL application, the main issue is related to the setting of the temporary access. The screen for setting the entries offers a huge range of options - such as the calendar, time slot, and multiple period selection - and in some case has been perceived as confusing. We tested different layouts (see Figure 26) with users to refine an UI which minimize the complexity to accomplish the task of programming access key without assistance of the facilitator. In the final version we opted for a UI solution which can be perceived as familiar as possible by user: elements and actions can be recognized because it recall patterns based on the previous experience of the users with the most downloaded/installed applications of Calendar and Agenda planning.

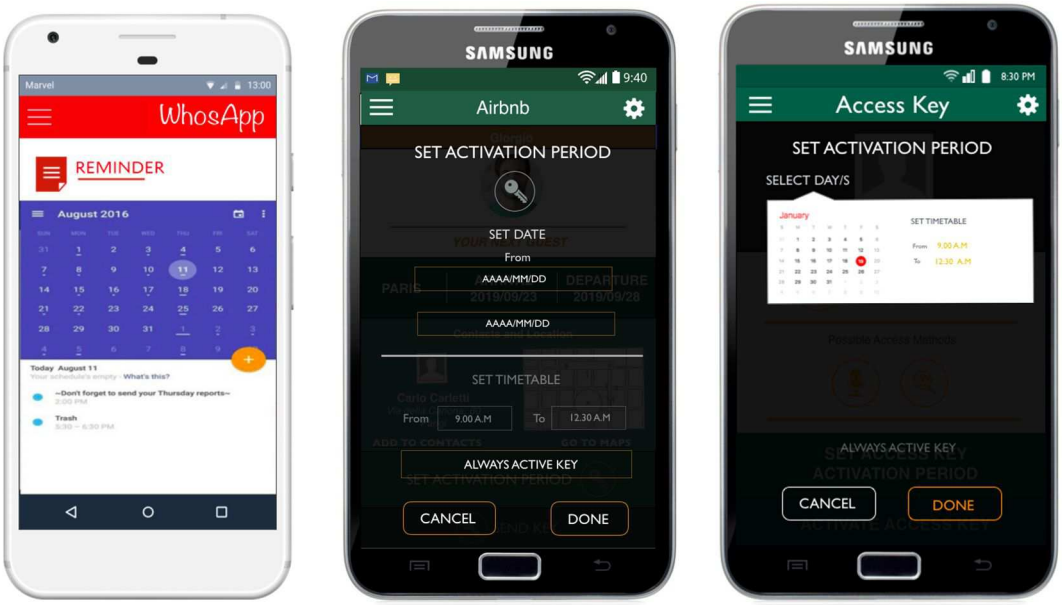

Figure 26 - The images are about the three different mockups designed for SH application. On the left the first idea, in the middle the UI reviewed after the first testing, and on the right the last version developed after UX analysis.

The stakeholder Immobiliare.it tested the mockup related to the Airbnb scenario and renting experience. The main feedback is positive and we collected some requests about follow ups and new features to develop.

For example, the user would like to simplify the login/register process in order to automatically sync the user data with their customer database to put together all the personal information of Airbnb service (see Figure 27) with the previews history of the user as an Immobiliare.it customer.

About the AXIOM VIMAR doorbell we tested the training process with the application and with the video door device. In the training process done with the application (see Figure 28) users were able to understand the overall workflow, follow step by step instructions and capture a training set of data. In this UI is still impossible during the training process to evaluate if the input will be sufficiently refined and clean to run the system with minimal error output. For e.g. in the voice training, the user can see the volume of the audio captured but he can't see if he is speaking too fast or if quality of speech is low. The user has to make a qualitative auto-evaluation of the audio training because the application can offer only a quantitative evaluation requiring a minimum number of files to capture and save but can't understand if the audio is qualitatively correct.

Figure 27 - A screenshot about the login/register procedure on application.

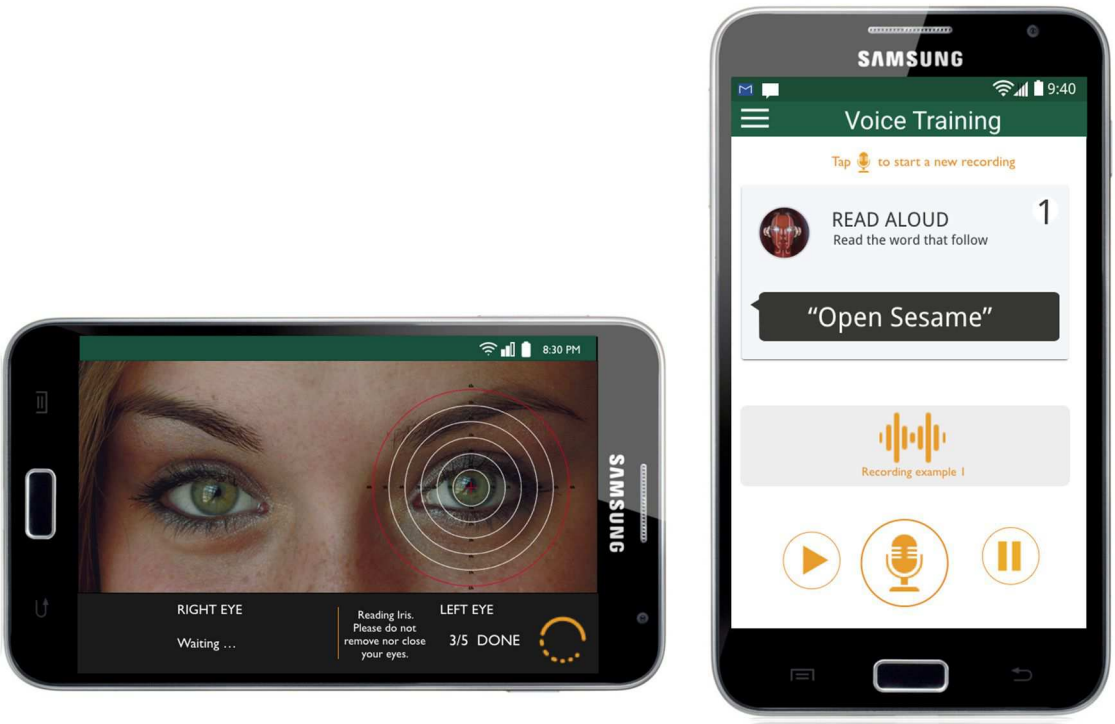

Figure 28 - Two screenshots of SH application: on the left a step of the Iris Recognition training system, on the right a step of the Voice Recognition Training.

Figure 29 - A photo about the testing of the video door entry system: the user is training the iris recognition system.

About the training process with the video door device, the users were able to reach the final goal of train their iris/voice recognition system but needed to be assisted by a facilitator to finalize correctly the steps.

In fact to train the video doorbell to recognize a subject iris, the device's camera must capture ten shots of the same person's iris, five for each eye (see Figure 29).

Figure 30 - A photo about a user looking at led feedback on the device during the training procedure.

The video door has 3 LEDs. Each LED can be in red or green status (see Figure 30). The red color indicates a failure in the system execution, the green indicates a correct result of the task completed.

The LEDs are used as a feedback for the user during the training phase to understand if the subject stands up in the right position and the camera can recognize the two eyes and capture correctly the iris photos. While LEDs are useful to indicate if the position is right or wrong, they are limited in suggesting how to correct it. For example, the user is not able to understand if he needs to get closer to the camera or perhaps he must take a step right or left to be selected by the camera. The LEDs are situated near the camera, some inches above the touch screen displaying the subject photo. All the participants expressed the desire to see their image in the same space range to easily correct the position without looking away. The trouble of users to assess the state of the system known as golf of evaluation [25] leaves users without feedbacks from their actions and unable to understand if they are doing the right actions to achieve the goal (see Figure 31). The same problem described for the application is present while training with the physical device: the process is blocked only in case of the uploaded data is under the minimum number required to end the procedure, and the qualitative validation of the input data is auto-evaluated by users.

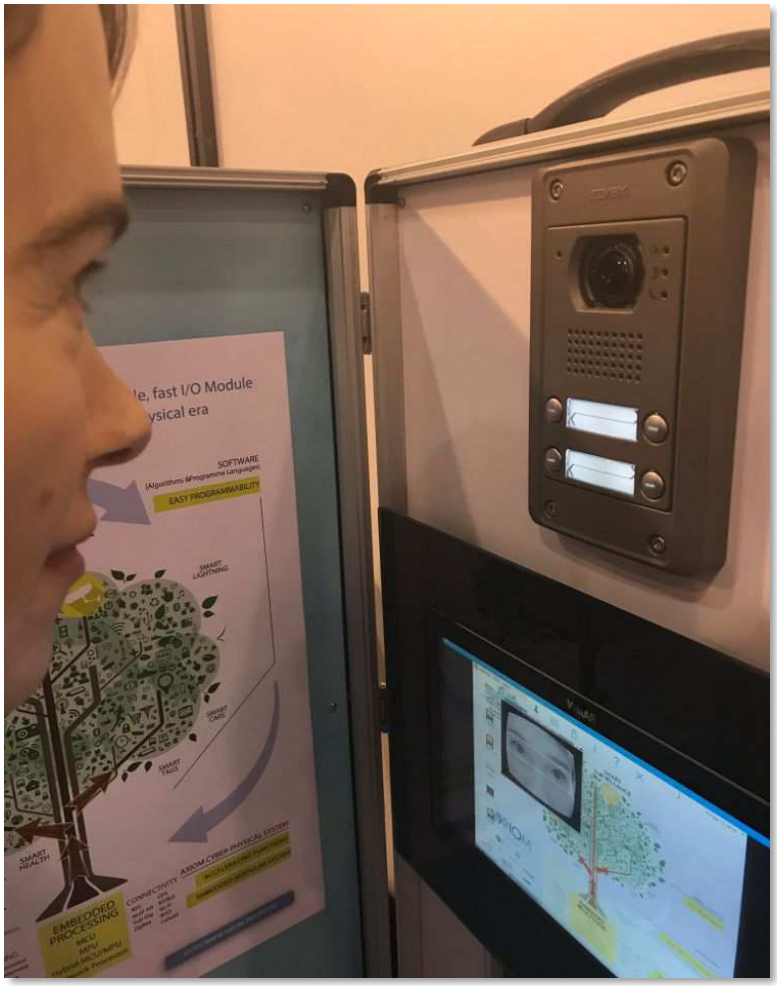

Figure 31 - A photo during the testing shows the user looking at the camera and the bias of understanding feedback.

For the SHL application the main breakdowns revealed are:

- Navigation breakdowns – difficulties revealed in activation period workflow to set Access Key privileges and timetable for guests;
- Presentation breakdowns – some users were confused by the Key Icon to understand when it was enabled/disabled, if it needs to be set or if it was just activated;
- Control usage breakdowns – troubles during the training of the video door entry system and understanding the feedback to control the quality of the data sample (as explained above).

Main concerns/interest:

- Concerns: It was not clear where and what to expect as feedback in some of the interactions. More interactions with other app/services available in the smartphone were expected (e.g. messaging, photos);
- Interests: The new services were very much appreciated and inspired extensions toward the office and the Condominium. There were proposals also in respect to hotels and holiday villages. An interesting proposal came from one of the stakeholder in the direction to use a blockchain framework to increase the service security.

4.3.1 Concluding remarks

The testing with the final users, who were completely unaware about the AXIOM project, showed that the possible new services enabled by the AXIOM CPS were very much appreciated and the exploration through the *mis-en-scène* of the scenarios allowed to capture the interest and involvement of the participants. At the same time, the results allowed to point out both further potentialities and margin of improvement in the current implementation of the services in the prototypes.

Other related publications of this project can be found in the references [26]-[51].

5 Confirmation of DoA objectives

PLANNED	DELIVERED
DELIVERABLE: SCENARIOS PORTING	
• Porting of the SVS application to the OmpSs Programming Model	Implementation steps and results are shown in Section 2. SVS software application successfully ported to the AXIOM board, and exploiting all components of the heterogeneous UltraScale+ MPSoC (FPGA, GPU and CPUs).
• Porting of the SHL application to the OmpSs Programming Model	Implementation steps and results are shown in Section 3. SHL software application successfully ported to the AXIOM board, and exploiting all components of the heterogeneous UltraScale+ MPSoC (FPGA and CPUs). Initial experiments conducted on OmpSs@Cluster on two AXIOM boards.
DELIVERABLE: USER EXPERIENCE TESTING	
• Testing the AXIOM CPS mainly focusing on User Experience	User Experience validation is discussed in Section 4. Multiple UI interfaces were developed for both SVS and SHL use cases.

6 Conclusions

In this deliverable, we have described the steps followed for porting both SVS and SHL use case applications to the AXIOM platform. The orchestration of the different software components exploiting the main features of the UltraScale+ MPSoC (PS and PL clusters) was achieved by means of the OmpSs programming model.

Critical parts of the source code exploited OmpSs@FPGA annotations and runtime libraries for offloading compute-intensive kernels to the FPGA reconfigurable logic. As it was originally planned, such modifications required minimal source code modifications. Minor HLS pragmas were added for providing hints to the Vivado HLS compiler and thus further increase the performance of the generated bitstream. On the other hand, XDMA API calls were also added for registering the variables and arguments used by the kernels that were offloaded to the FPGA. Additionally, such runtime libraries also greatly contributed to handle CPU/FPGA memory transfers in a transparent manner. More particularly, it completely avoided HERTA and VIMAR having to develop a Linux DMA driver for mapping input/output data to the FPGA accelerator through the AXI bus.

In order successfully complete WP3, additional unscheduled work had to be carried for enabling the graphical output, which was required to display the video footage and the UI (SVS use case), and to show on screen the iris overlay in real-time (SHL use case).

The parallelization approach followed by OmpSs, quickly enabled HERTA and VIMAR to port and offload real applications featuring highly compute-intensive sequential kernels to a data-parallel FPGA-based architecture. This porting process was completed on time, and most importantly without having prior knowledge on both the underlying architecture of the FPGA nor HDL design. Similarly, the learning curve of the OmpSs parallel programming model specification was very steep and, as such, it did not require extensive study.

Moreover, the selected use cases simultaneously exploited the different PS and PL resources of the heterogeneous UltraScale+ MPSoC, and implemented real-world solutions that could potentially address multiple markets such as smart home, surveillance, access control and retail among the most important ones. In order to demonstrate the feasibility of the AXIOM platform for targeting those market segments, several user interfaces were designed and later studied for further refinement. Also, the user experience of both applications was tested for assessing the appeal of such services and fine-tuning the human-machine interaction.

7 References

[1] Open Air Repository, https://www.openaire.eu/
[2] Zenodo Repository, https://zenodo.org/
[3] Duran, A., Ayguadé, E., Badia, R. M., Labarta, J., Martinell, L., Martorell, X., & Planas, J. (2011). OmpSs: a proposal for programming heterogeneous multi-core architectures. Parallel Processing Letters, 21(02), 173-193.
[4] Herta BioSurveillance, http://www.hertasecurity.com/sites/default/files/pages/files/BIOSURVEILLANCE_eng.pdf
[5] LibAv – Open Source audio and video processing tools, https://www.libav.org/
[6] X.Org Foundation, https://www.x.org/wiki/
[7] QT Framework, https://www.qt.io/
[8] QT5.4 X11 patch, https://wiki.axiom-project.eu/images/f/f5/Axiom_qt5.4_patch.zip
[9] Pillet, Vincent, et al. "Paraver: A tool to visualize and analyze parallel code." Proceedings of WoTUG-18: transputer and Occam developments. Vol. 44. No. 1. IOS Press, 1995.
[10] OpenCV, http://opencv.org/
[11] Alize, http://alize.univ-avignon.fr/
[12] SPro, https://www.irisa.fr/metiss/guig/spro/spro-4.0.1/spro.html
[13] WebRTC, https://webrtc.org
[14] GStreamer Framework, https://gstreamer.freedesktop.org/
[15] Balart, Jairo, et al. "Nanos Mercurium: a research compiler for OpenMP." Proceedings of the European Workshop on OpenMP. Vol. 8. 2004.
[16] Extrae BSC tool, https://tools.bsc.es/extrae
[17] D. Theodoropoulos, S. Mazumdar, E. Ayguade, N. Bettin, J. Bueno, S. Ermini, A. Filgueras, D. Jimenez-Gonzalez, C. Alvarez Martinez, X. Martorell, F. Montefoschi, D. Oro, D. Pnevmatikatos, A. Rizzo, P. Gai, S. Garzarella, B. Morelli, A. Pomella, R. Giorgi, "The AXIOM platform for next-generation cyber physical systems", Microprocessors and Microsystems, 2017. DOI:10.1016/j.micpro.2017.05.018
[18] Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User-centered design. Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications, 37(4), 445-456.
[19] Rizzo, A., Marchigiani, E., & Andreadis, A. (1997, August). The AVANTI project: prototyping and evaluation with a cognitive walkthrough based on the Norman's model of action. In Proceedings of the 2nd conference on Designing interactive systems: processes, practices, methods, and techniques (pp. 305-309). ACM.
[20] A. Rizzo, F. Montefoschi, M. Caporali, A. Gisondi, G. Burresi, R. Giorgi, "Rapid Prototyping IoT Solutions Based on Machine Learning", Proc. European Conf. on Cognitive Ergonomics 2017, New York, NY, USA, 2017, pp. 4. DOI:10.1145/3121283.3121291.
[21] Veron E and Levasseur M. (1983) "Ethnographie de l'exposition", Paris, Bibliothèque publique d'Information, Centre Georges Pompidou.
[22] R. Giorgi, N. Bettin, P. Gai, X. Martorell, A. Rizzo, "AXIOM: A Flexible Platform for the Smart Home", Springer Int.l Publishing, Cham, 2016, pp. 57-74. DOI:10.1007/978-3-319-42304-3_3
[23] Why You Only Need to Test with 5 Users by JAKOB NIELSEN on March 19, 2000, https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/
[24] Balsamiq, https://balsamiq.com
[25] Norman, D. A. (1986). Cognitive engineering. User centered system design, 31, 61.
[26] R. Giorgi, "Accelerating Haskell on a Dataflow Architecture: a case study including Transactional Memory", Proc. Int.l Conf. on Computer Engineering and Applications (CEA), Dubai, UAE, Feb. 2015, pp. 91-100. ISBN: 978-1-61804-276-7
[27] R. Giorgi, "Exploring Future Many-Core Architectures: The TERAFLUX Evaluation Framework", Elsevier, 2017, pp. 33-72. DOI:10.1016/bs.adcom.2016.09.002
[28] R. Giorgi, "Transactional Memory on a Dataflow Architecture for Accelerating Haskell", WSEAS Trans. Computers, vol. 14, 2015, pp. 546-558. ISSN: 1109-2750
[29] N. Ho, A. Mondelli, A. Scionti, M. Solinas, A. Portero, R. Giorgi, "Enhancing an x86_64 Multi-Core Architecture with Data-Flow Execution Support", ACM Computing Frontiers, Ischia, Italy, May 2015. DOI:10.1145/2742854.2742896
[30] L. Verdoscia, R. Vaccaro, R. Giorgi, "A matrix multiplier case study for an evaluation of a configurable Dataflow-Machine", ACM CF'15 - LP-EMS, May 2015, pp. 1-6. DOI:10.1145/2742854.2747287
[31] G. Burresi, R. Giorgi, "A Field Experience for a Vehicle Recognition System using Magnetic Sensors", IEEE MECO 2015, Budva, Montenegro, June 2015, pp. 178-181. DOI:10.1109/MECO.2015.7181897,
[32] D. Theodoropoulos, D. Pnevmatikatos, C. Alvarez, E. Ayguade, J. Bueno, A. Filgueras, D. Jimenez-Gonzalez, X. Martorell, N. Navarro, C. Segura, C. Fernandez, D. Oro, J. Saeta, P. Gai, C. Scordino, A. Rizzo, R. Giorgi, "The AXIOM

project (Agile, eXtensible, fast I/O Module)", IEEE Proc. 15th Int.l Conf. on Embedded Computer Systems: Architecture, MOdeling and Simulation, July 2015, pp. 262-269.DOI: 10.1109/SAMOS.2015.7363684

[33] A. Mondelli, N. Ho, A. Scionti, M. Solinas, A. Portero, R. Giorgi, "Dataflow Support in x86-64 Multicore Architectures through Small Hardware Extensions", IEEE Proc. DSD, August 2015, pp. 526-529.. DOI: 10.1109/DSD.2015.62

[34] C. Alvarez, E. Ayguade, J. Bueno, A. Filgueras, D. Jimenez-Gonzalez, X. Martorell, N. Navarro, D. Theodoropoulos, D. Pnevmatikatos, C. Scordino, P. Gai, C. Segura, C. Fernandez, D. Oro, J. Saeta, P. Passera, A. Pomella, A. Rizzo, R. Giorgi, "The AXIOM Software Layers", IEEE Proc. 18th EUROMICRO-DSD, Aug. 2015, pp. 117-124. DOI:10.1109/DSD.2015.52

[35] D. Jiménez-González, C. Álvarez, A. Filgueras, X. Martorell, J. Langer, J. Noguera, K. Vissers. "Coarse-grain performance estimator for heterogeneous parallel computing architectures like Zynq all-programmable SoC". arXiv preprint arXiv:1508.06830.

[36] R. Giorgi, "Scalable Embedded Systems: Towards the Convergence of High-Performance and Embedded Computing", Proc. 13th IEEE/IFIP Int.l Conf. on Embedded and, Oct. 2015, pp. 148-153. DOI:10.1109/EUC.2015.34

[37] R. Giorgi, A. Scionti, "A scalable thread scheduling co-processor based on data-flow principles", ELSEVIER Future Generation Computer Systems, Amsterdam, Netherlands, vol. 53, Dec. 2015, pp. 100-108. DOI:10.1016/j.future.2014.12.014

[38] P. Burgio, C. Alvarez, E. Ayguade, A. Filgueras, D. Jiminez-Gonzalez, X. Martorell, N. Navarro, R. Giorgi, "Simulating next-generation Cyber-physical computing platforms", Ada User Journal, vol. 36, no. 4, Dec. 2015, pp. 259-263. ISSN: 1381-6551

[39] L. Verdoscia, R. Giorgi, "A Data-Flow Soft-Core Processor for Accelerating Scientific Calculation on FPGAs", Mathematical Problems in Engineering, vol. 2016, no. 1, Apr. 2016, pp. 1-21, (article ID 3190234). DOI:10.1155/2016/3190234

[40] R. Giorgi, "Exploring Dataflow-based Thread Level Parallelism in Cyber-physical Systems", Proc. ACM Int.l Conf. on Computing Frontiers, New York, NY, USA, 2016, pp. 6. DOI:10.1145/2903150.2906829

[41] C. Alvarez, E. Ayguade, J. Bosch, J. Bueno, A. Cherkashin, A. Filgueras, D. Jiminez-Gonzalez, X. Martorell, N. Navarro, M. Vidal, D. Theodoropoulos, D. Pnevmatikatos, D. Catani, D. Oro, C. Fernandez, C. Segura, J. Rodriguez, J. Hernando, C. Scordino, P. Gai, P. Passera, A. Pomella, N. Bettin, A. Rizzo, R. Giorgi, "The AXIOM Software Layers", ELSEVIER Microprocessors and Microsystems, vol. 47, Part B, 2016, pp. 262-277. DOI:10.1016/j.micpro.2016.07.002

[42] S. Mazumdar, E. Ayguade, N. Bettin, S. Bueno J. and Ermini, A. Filgueras, D. Jimenez-Gonzalez, C. Martinez, X. Martorell, F. Montefoschi, D. Oro, D. Pnevmatikatos, A. Rizzo, D. Theodoropoulos, R. Giorgi, "AXIOM: A Hardware-Software Platform for Cyber Physical Systems", 2016 Euromicro Conf. on Digital System Design (DSD), Aug 2016, pp. 539-546. DOI:10.1109/DSD.2016.80

[43] G. Llort, A. Filgueras, D. Jiménez-González, H. Servat, X. Teruel, E. Mercadal and J. Labarta. "The Secrets of the Accelerators Unveiled: Tracing Heterogeneous Executions Through OMPT". In International Workshop on OpenMP (pp. 217-236). Springer, Cham. DOI: 10.1007/978-3-319-45550-1_16

[44] R. Giorgi, S. Mazumdar, S. Viola, P. Gai, S. Garzarella, B. Morelli, D. Pnevmatikatos, D. Theodoropoulos, C. Alvarez, E. Ayguade, J. Bueno, A. Filgueras, D. Jimenez-Gonzalez, X. Martorell, "Modeling Multi-Board Communication in the AXIOM Cyber-Physical System", Ada User Journal, vol. 37, no. 4, December 2016, pp. 228-235. ISSN: 1381-6551

[45] A. Rizzo, G. Burresi, F. Montefoschi, M. Caporali, R. Giorgi, "Making IoT with UDOO", Interaction Design and Architecture(s), vol. 1, no. 30, Dec. 2016, pp. 95-112. ISSN: 1826-9745

[46] M. Wagner, G. Llort, A. Filgueras, D. Jiménez-González, H. Servat, X. Teruel, and E. Ayguadé. "Monitoring Heterogeneous Applications with the OpenMP Tools Interface". In Tools for High Performance Computing 2016 (pp. 41-57). Springer. DOI: 10.1007/978-3-319-56702-0_3

[47] R. Giorgi, "AXIOM: A 64-bit reconfigurable hardware/software platform for scalable embedded computing", 6th Mediterranean Conf. on Embedded Computing (MECO), June 2017, pp. 113-116. DOI:10.1109/MECO.2017.7977173

[48] D. Theodoropoulos, D. Pnevmatikatos, S. Garzarella, P. Gai, A. Rizzo, R. Giorgi. "AXIOM: enabling parallel processing in cyber-physical systems." Reconfigurable Computing Workshop, Lausanne, CH. Sep 2016. Pp.1-2.

[49] J. Bosch Pons, "Asynchronous runtime for task-based dataflow programming models." Jul. 2017. Master's Thesis. Universitat Politecnica de Catalunya.

[50] S. Mazumdar and R. Giorgi. "A Survey on Hardware and Software Support for Thread Level Parallelism." arXiv preprint arXiv:1603.09274 (2016).

[51] C. Scordino and B. Morelli. "Sharing memory in modern distributed applications". In Proceedings of the 31st Annual ACM Symposium on Applied Computing (pp. 1918-1921) ACM, April 2016.