HW/SW Co-Design Framework for Mixed-Criticality Embedded Systems considering Xtratum-based SW Partitions

1\textsuperscript{st} Vittoriano Muttillo  
\textit{University of L’Aquila}  
L’Aquila, Italy  
vittoriano.muttillo@graduate.univaq.it

2\textsuperscript{nd} Luigi Pomante  
\textit{University of L’Aquila}  
L’Aquila, Italy  
luigi.pomante@univaq.it

3\textsuperscript{rd} Patricia Balbastre  
\textit{Universitat Politècnica de València}  
Valencia, Spain  
pbalbastre@ai2.upv.es

4\textsuperscript{th} José Simó  
\textit{Universitat Politècnica de València}  
Valencia, Spain  
jsimo@ai2.upv.es

5\textsuperscript{th} Alfons Crespo  
\textit{Universitat Politècnica de València}  
Valencia, Spain  
acrespo@ai2.upv.es

Abstract—Heterogeneous parallel devices are becoming widely diffused in the embedded systems application field since they allow to improve time performances and other orthogonal metrics (e.g., cost, power, size, etc.) at the same time. In such a context, the introduction of safety requirements, as dictated by the relevant standards (i.e., DO-178 B/C and RTCA/DO-254 in airborne systems, ARINC 653 for avionics software, ISO-26262 in automotive domain, etc.) while considering shared resources on a heterogeneous parallel HW platform, adds further challenges to industrial and academic research. This kind of platforms that execute tasks with different levels of criticality are commonly called mixed-criticality embedded systems. So, the main problem in their management is to ensure that low criticality tasks do not interfere with high criticality ones. The final goal is to allow several applications to interact and coexist on the same platform. For this, the exploitation of virtualization technologies (i.e., hypervisors) allows to guarantee isolation and to satisfy certification requirements but introduces scheduling overhead and new HW/SW partitioning challenges. In such a scenario, this work focuses on a framework for modeling, analysis, and validation of mixed-criticality and real-time systems based on an existing "Model-Based Electronic System Level HW/SW Co-Design" methodology. The main contribution of this work is the integration of the considered framework with Xamber tool in order to provide systems implementations by exploiting a design space exploration able to consider Xtratum-based SW partitions.

Index Terms—HW/SW Co-Design, Heterogeneous Parallel Systems, Design Space Exploration, Mixed-Criticality, Hypervisor

I. \textbf{INTRODUCTION}

In the last thirty years, there has been an exponential increase in the diffusion and evolution of \textit{Embedded Information and Communication Technologies} (Embedded ICT). As a consequence, the presence of embedded systems in everyday life is constant and, often, almost invisible. Embedded systems have common characteristics like the periodic execution of a single application (or a very limited set of applications) and the reactivity to external triggers possibly in Real-Time (RT). More in general, other than the main \textit{Functional Requirements} (FR), it is possible to identify several \textit{Non-Functional} ones (NFR) normally relevant in the embedded systems domain (e.g., cost, size, power/energy, etc.) and well known in advance.

During system development, the exploitation of proper HW/SW technologies normally allows to satisfy all the design requirements. However, it is not simple to identify the ones to be used, given the great number of heterogeneous available HW/SW components and the strong interdependence among the involved tasks. Furthermore, nowadays it is also possible to implement different functionality on a single chip to reduce manufacturing cost and design time. Such systems can include several heterogeneous processors (i.e., by following the classification provided in [1]: \textit{General-Purpose}, GPP; \textit{Application Specific}, ASP; \textit{Single Purpose}, SPP), memories, and a set of interconnection links among them. In this context, the correct choice of the processor technologies (mainly as GPP/ASP that execute SW vs SPP that implement specific functionality directly in HW) plays an essential role in the design activity and HW/SW Co-Design methodologies with related \textit{Electronic Design Automation} (EDA) tools, are of fundamental support for the designers. Unfortunately, there are no fully engineered general methodologies defined for this purpose and often the best option is still to refer to designers experience.

In such a context, an additional challenge is the recent switch from single-processor/core to (heterogeneous) parallel HW/SW platforms used to execute embedded applications with different levels of criticality (i.e., \textit{Mixed-Criticality Embedded Systems}, MCESs). The main problem in the management of a MCES is to ensure that low criticality applications do not interfere with high criticality ones. This type of systems can be found in many domains such as aerospace (e.g., \textit{Integrated Modular Avionic, IMA} [2]) and automotive industry [3]. Critical and non-critical applications can be further divided
by identifying different criticality classes. The goal is always to allow these applications to interact and coexist on the same platform, but a proper management of such Mixed Criticality (MC) systems becomes a very complex task that poses several challenges also from the implementation point of view [4]. The basis for integrating various critical applications are the isolation mechanisms that allow to enforce temporal and spatial separation [5]. As an example, according to this approach, embedded applications with different levels of criticality can be allocated on different “partitions” by exploiting Hypervisors (HPVs) technologies, which can be verified and validated in isolation (e.g., PikeOS [6], Xtratum [7]). Another approach is to allocate them on different HW components. The identification of the best solution is not always possible so heuristics methods are needed to support MCESs designers.

At Electronic System Level (ESL) of abstraction, there are very few works that introduce Mixed-Criticality (MC) issues directly into a HW/SW co-design flow.

So, according with the scenario described above, this work extends an existing open-source SystemC-based HW/SW Co-Design Environment for Heterogeneous Parallel Systems ( [8], [9]) by introducing RT and MC requirements as additional non-functional constraints. More in detail, the focus is on: the extension of the Design Space Exploration (DSE) approach to take into consideration MC constraints; the introduction of the “SW partition” concept as provided by HPV technologies; the integration of Xamber external tool for HPV configuration and (semi)automatic code generation.

The remainder of the paper is organized as follows: Section II presents related works that consider mixed-critical requirements into the whole design flow. Section III describes the reference HW/SW co-design framework. Then, Section IV analyzes experimental results. Finally, Section V closes the paper with some conclusions and future works.

II. HW/SW Co-Design for Mixed-Criticality Applications

A remarkable number of research works have focused on ESL HW/SW Co-Design of Dedicated Heterogeneous Multi-Processor Systems (D-HMPS). As presented in [10], the HW/SW Co-Design has a long history of more than 30 years. The main problem is related to (automatically) find a solution (in terms of final platform implementations) able to consider at the same time different FR/NFR starting from an ESL of abstraction, in a HW/SW Co-Design context. Since this work focuses particularly on RT and MC requirements, in the following, the most similar works in the literature are analyzed and compared with the proposed one.

In this scenario, AUTOFOCUS3 [11] proposes a model-based development process introducing safety-oriented constraints associated to computing components. The tool assigns the levels of criticality to application tasks and computing resources, avoiding the allocation of high-criticality tasks to low-criticality resources. AUTOFOCUS considers a generic System "Logical Architecture", that is intrinsically non behavioural. Moreover, the safety constraints are related to SIL classification, meanwhile it avoids allocation of higher-criticality level tasks on different lower-criticality level resources, but they do not consider different criticality level tasks interference.

CONTREP [12] is a framework supporting UML/MARTE based modeling, analysis and design of Cyber-Physical Systems (CPS), also with MC constraints. It is based on the CONTREX UML/MARTE modeling methodology [13] with some SysML features integration and considers safety constraints into the different design activities. CONTREP allows to convert MARTE models into ForSyDe-SystemC simulatable models for a formal functional validation. CONTREP offers schedulability analysis producing models that can be processed by the MAST schedulability analysis tool [14]. CONTREP enables embedded software synthesis for heterogeneous multi-core platforms using eSSYN features [15]. The CONTREP framework allows the user to select a specific exploration tool for DSE (i.e., Multicube Explorer [16]). CONTREP is then combined with simulation-based tools (i.e., VIPPE [17]) to perform timing and power analysis. To support MC constraints, CONTREP applies a minor extension to MARTE profile, adding a criticality attribute to a NonFunctionalProperties subprofile as an integer to denote an abstract criticality level. CONTREP offers also MC-aware schedulability analysis and architectural mapping validation. Note that DSE does not consider MC requirements, while the CONTREP modeling methodology offers the possibility to check MC constraints fulfillment, and CONTREP does not consider HPV technologies into the design flow.

ForSyDe (Formal System Design) models [23] have been enabled as a design entry to an analytical DSE tool, called DeSyDe [19]. DeSyDe is a modular tool which provides a DSE for bare-metal applications, finding implementations for a set of tasks on a shared Multi-processor System-on-chip (MPSoC) starting from synchronous dataflow graphs (SDFGs) and a predictable model for target platform. DeSyDe also offers support for MC, in the sense that constraints on performance and cost metrics can be hard for some applications, and they will be implemented on specific safety resources, while other applications are provided with best-effort service on the remaining resources [24]. In that work, the concept of Mixed-Criticality System (MCS) is not-well established since it refers only to the complexity in implementing such MCS on specific (safety) resources.

The work in [20] [25] presents a design methodology, called OSSS (Oldenburg System Synthesis Subset), for MPSoC. The OSSS design entry point is the Application Layer (AL), the platform model is a Virtual Target Architecture Layer (VTAL), while a manual mapping between application tasks and system components allows to simulate system behaviour to check input requirements. Using FOSSY (Functional Oldenburg System Synthesizer) tool [26], it is possible to synthesize a specific target platform in a separate step using AL and VTAL models. In [27] [28] the authors extend the methodology introducing MCS constraints, presenting the OSSS-MC system methodology. In these works OSSS-MC partitions
system behaviour into tasks and shared objects, clustered using criticality levels, defining a criticality-dependant end-to-end execution time and a criticality-dependant behaviour regarding system mode of execution respect to criticality levels.

A work that considers HPV and a methodology to identify a set of HPV-based SW partitions is MultiPARTES [21]. It relies on Model Driven Engineering (MDE) toolset and offers HPV partitions identification (i.e., Xtratum software partitions) and application allocation. However, MultiPARTES considers only a fixed multi-core architecture, managing HPV partitions only in a homogeneous multi-processor platform.

Starting from this list of methodologies considering MC requirements, it is possible to classify them updating the table described in [29] with some modifications and new tools. Table I presents a classification of this different ESL methodologies in terms of application and platform specification, DSE support and refinement activities. The specification column specifies the application models, in terms of Model of Computation (MoC) and meta-models, and the corresponding specification languages, and the platform architecture, in terms of heterogeneous/homogeneous multi-processor ones. From an implementation point of view, the Model of Structure (MoS) represents the system architecture and structure. MoS may be a netlist with a semantics limited to describing component connectivity or a Transaction Level Model (TLM) that tries to abstract as-much-as-possible architectural concepts. In order to estimate performances (in terms of timing, power/energy, cost/area etc.), a Model of Performance (MoP) is defined as a model where each individual elements is associated to a quality numbers respect to specific given implementations. The granularity is a measurements of the accuracy associated to each implemented elements in different solutions. It may be cycle-accurate (using RTL representations), instruction-set-accurate (using also Instruction-Set Simulators, ISS), or task-accurate (with estimations at a high level of abstraction). This quality numbers are used by the DSE step to find different implementation alternatives. Finally, starting from the specification, the synthesis is driven by decision making (into the DSE step) and system refinements (in terms of ESL synthesis activities) with both computational and communication elements. In the Table I, all the previous classification criteria are taken into account, where a full circle implies that the ESL aspect is fully supported, while an open circle means a partial support (and a partial automation). It is worth noting that this work fully support all the ESL synthesis criteria presents in the Table I, while future works will try to complete the missing features in terms of communication refinements and also external tools integration, to make advanced comparisons and to validate the final methodology.

III. REFERENCE HW/SW CO-DESIGN FRAMEWORK

In the context of MCESs, this work exploits an existing open-source ESL HW/SW co-design framework ( [8], [9]), and extends it by introducing the possibility to consider also MC and RT requirements (the extended framework is called HEPSYCODE-MC: HW/SW Co-Design of Heterogeneous Parallel Dedicated Systems with Real-Time and Mixed-Criticality Constraints).

While the general methodology has been described in [30] [31], this work proposes a specialization of the reference framework, in the context of MegaM@Rt2 [32] European Project, in order to define a DSE methodology able to take into account MCES based on Xtratum HPV [7]. In particular, the work focuses on agnostic models for partitioned MCES into multi-core systems and on generation of automatic projects/code of partitions, using model transformation between Xamber tool [33] and HEPSYCODE-MC approach (Fig. 1).

Xamber is a graphical configuration tool adapted to assist the user through completion of the configuration of partitioned systems, and provides an interface for capturing and editing the elements that are part of the system. Xamber generates the configuration file needed by a HPV (i.e., Xtratum) to execute the system. Meanwhile, Xtratum [7] is a bare metal HPV supporting paravirtualization for multiple architectures. Xtratum natively supports SPARC architecture and LEON processors.

Starting from HEPSYCODE-MC methodology (and related tools) and Xamber tool, an integration step has been performed in order to check overlapping functionality and to exploit HEPSYCODE-MC framework functionality.

The list of activities involves different modeling and design adaptation in the HEPSYCODE-MC HW/SW Co-Design Flow in order to introduce HPV technologies in the DSE step, by considering a System-Level RT MoC based on Communicating Sequential Processes (CSP), modified with some for-

### Table I: Classification of different ESL Methodology Approaches (considering MC issues).

<table>
<thead>
<tr>
<th>ESL Approach</th>
<th>Specification</th>
<th>Implementation</th>
<th>DSE 1</th>
<th>Decision Making</th>
<th>Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOFOCUS [11]</td>
<td>PN</td>
<td>HeMPES</td>
<td>Custom Component</td>
<td>TAPM</td>
<td>Comp. 6</td>
</tr>
<tr>
<td>CONTREP [18]</td>
<td>UML, SDF</td>
<td>HeMPES</td>
<td>MARTE &amp; SysML &amp; SystemC</td>
<td>TLM</td>
<td>TAPM</td>
</tr>
<tr>
<td>DeSyDe [19]</td>
<td>SDF</td>
<td>HeMPSoC</td>
<td>XML</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OSS/SMC [20]</td>
<td>OSS/SMC</td>
<td>HeMPSoC</td>
<td>SystemC</td>
<td>TLM</td>
<td>T/ISAPM</td>
</tr>
<tr>
<td>MultiPARTES [21]</td>
<td>UML</td>
<td>HeMPES</td>
<td>MARTE</td>
<td>TLM</td>
<td>TAPM</td>
</tr>
<tr>
<td>This work [22]</td>
<td>CSP</td>
<td>HeMPES</td>
<td>SystemC</td>
<td>TLM</td>
<td>T/ISAPM</td>
</tr>
</tbody>
</table>

mal communication constraints with respect to unidirectional point-to-point blocking channels that allow tasks communication in a deterministic network model.

Starting from a CSP System Behavioural Model (CSP-SBM), representing an executable model of the application behavior, splitting processes into pieces of code that represent tasks in the RT domain (creating the so called Process Interaction Model, PIM), it is possible to transform the initial CSP application model into the final Process to Task Graph Model (PTM), conform to the most used RT standards, as presented in [34]. After these assumptions and related transformation activities, the integration between HEPSYCODE-MC and Xamber has been realized with a methodology change in the HEPSYCODE-MC framework, as shown in Fig. 1.

The rest of the paper describes the integration activity in details.

A. HML Specification

The reference System-Level modelling language in HEPSYCODE-MC is the Hepsy Modeling Languages (HML), where the application is described by a process network connected via synchronous channels. In the HEPSYCODE-MC environment, the application described via HML is transformed into a System Behaviour Model (SBM). The SBM is a Communicating Sequential Process (CSP-based) executable Model of Computation (MoC) of the system behavior that explicitly defines also a model of communication among processes using unidirectional point-to-point blocking channels for data exchange. An example of HML application is shown in Fig. 2.

B. Metrics Evaluation

After the modeling step, several metrics evaluations and estimations have been performed and the execution time associated to each task has been estimated by means of HEPSIM [35].

C. Design Space Exploration

The DSE step is able to find a solution, in terms of HPV-based SW partitions in a heterogeneous multi-processor parallel scenario. In order to transform HEPSYCODE-MC input models into Xamber projects, it is needed to fix some parameters:

1) Only single core scenarios (with LEON3 processor cores) is permitted (Xamber support only single-core scenario at the moment);
2) No other basic HW components (in terms of extra processors connected into a heterogeneous distributed scenarios) are considered in the DSE step;
3) To consider a safety-critical scenario, a criticality level is assigned to different processes respect to their functionality;
4) A maximum number of 4 Xtratum SW partitions are allowed in the DSE.

D. HEPSYCODE-MC - Xamber Project Transformation

The final integration between HEPSYCODE-MC and Xamber has been realized with a methodology change in the HEPSYCODE-MC framework, as shown in Fig. 1. Using a transformation between XML schemas, the Partitioning solution, saved in a XML exchange file, has been translated into a Xamber compliant project, and schedulability analysis.
has been performed in order to find the best Hyperplan for
the initial task set, setting Xamber project parameters (in
terms of tasks, processors, partitions, RT parameters and so
on) from HEPSYCODE-MC Co-Analysis, Co-Estimation and
Partitioning steps.

The application model, the platform model, the partition
model and the mapping among these entities have been
transformed into a Xamber compliant project, using Java
Architecture for XML Binding (JAXB) technology. JAXB is
a software framework that allows Java developers to map Java
classes to XML representations using marshal transformation.

All the Xamber parameters are derived from the Hepsycyde
framework (i.e., processes execution time and partitions allo-
cation, End-To-End-Flow representation, IPC channels parti-
tions). This transformation allows also to use the Contrex
tool that performs schedulability analysis and find the best
hyper-plan for the reference application. After this activity,
Xamber produces the Xtratum Configuration file for SparcV8
architectures (file .xmc) and it is possible to perform 2 different
activities: (1) simulate the solution into the HEPSYCODE-MC
HPV Simulator engine (with hierarchical scheduling feature);
(2) check execution time (to check different HPV behavior
respect to the specific use case) implementing the proposed
input application.

E. HEPSIM Hierarchical Scheduling

In order to to simulate HPV timing behaviors, in the context
of this work, HEPSIM has been extended by implementing
a hierarchical scheduler, i.e., a second-level scheduler (i.e.,
HEPSCHED2, 2 Levels HEPSYCODE-MC SCHEDuler) with
respect to the standard SystemC one. HEPSCHED2 has been
implemented as a SystemC SC_MODULE containing a dedi-
cated HEPSCHED2 instance for each instance of processor
composing the system. Each HEPSCHED2 instance is imple-
mented as a SC_THREAD. The implementation of different
analysis mechanisms and scheduling policies in HEPSIM is
based on the instrumentation of code by means of macros and
their interaction with the HEPSCHED2. Macro S is inserted
as a prefix to the SystemC statements composing the SBM
to support the handshake mechanism for the scheduling of
processes as shown in Listing 1.

Listing 1. HEPSCHED2 Full Handshake: Interactions with macro S

```c
/* Macro S */
define S(X)
    pSystemManager->Increase(X);
    if(!pSystemManager->checkSPP(X))
        wait(pSchedulingManager->schedule[X]);
    .

/* HEPSCHED */
if (ready[ps.id]==true)
    schedule[ps.id].notify(SC_ZERO_TIME);
    wait(release[ps.id]);
.
/* Macro S */
wait(pSystemManager->upSimTime(X));
    if(!pSystemManager->checkSPP(X))
        pSchedulingManager->release[X].notify(SC_ZERO_TIME);

/* The handle goes to HEPSCHED2 */
```
step, the “Free Running Time” (FRT, the application execution time when all the processes are allocated on the same BB instance) has been estimated for the BB with LEON3 PU, and the values is equal to 0.0138695 s. The “Free Running Loads” (FRLs, the load evaluated for each process when all the processes are allocated on the same BB instance) and the average execution time for each process (called Average-Case Response Time, ACRT) are presented in Table III.

### TABLE III

<table>
<thead>
<tr>
<th>Process ID</th>
<th>FRL</th>
<th>ACRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.022503</td>
<td>263 μs</td>
</tr>
<tr>
<td>3</td>
<td>0.0401813</td>
<td>243 μs</td>
</tr>
<tr>
<td>4</td>
<td>0.0198998</td>
<td>175 μs</td>
</tr>
<tr>
<td>5</td>
<td>0.162515</td>
<td>1085 μs</td>
</tr>
<tr>
<td>6</td>
<td>0.119067</td>
<td>850 μs</td>
</tr>
<tr>
<td>7</td>
<td>0.0298497</td>
<td>263 μs</td>
</tr>
<tr>
<td>8</td>
<td>0.0282014</td>
<td>234 μs</td>
</tr>
<tr>
<td>9</td>
<td>0.0172465</td>
<td>175 μs</td>
</tr>
<tr>
<td>10</td>
<td>0.256044</td>
<td>2050 μs</td>
</tr>
<tr>
<td>11</td>
<td>0.179761</td>
<td>1554 μs</td>
</tr>
<tr>
<td>12</td>
<td>0.030513</td>
<td>291 μs</td>
</tr>
<tr>
<td>13</td>
<td>0.0859007</td>
<td>243 μs</td>
</tr>
<tr>
<td>14</td>
<td>0.0102816</td>
<td>17 μs</td>
</tr>
</tbody>
</table>

The FRLs have been calculated using the HEPSIM timing simulator. It is worth noting that such information are useful in order to find better tasks allocation among HP software partitions (also considering tasks workload, concurrency, ACRT parameters, and MC requirements). Considering single core scenarios (because Xamber do not support multicore architectures), only communication and criticality level are used to allocate and bind tasks on different partitions, then a schedulability analysis allows to refine the allocation in order to verify the correct behavior of tasks execution, considering also several system overhead (e.g., HPV context switch, tasks context switch, IPC). Moreover, HEPSYCODE-MC DSE produces one solution represented in Table IV.

### Listing 2
FirFirGCD Contrex Hyper-plan

Using this hyper-plan configuration into the HEPSIM simulator, the output in Table V has been produced, where each lap represents the execution time of one instance of the hyper-plan (in seconds). It is worth noting that the HEPSIM simulation follows the hyper-plan defined in the Xamber configuration tool (for each one of the 10 input triggers) without timing errors.

For the system implementation, the LEON3 General Purpose processor (GPP) has been considered. LEON3 is a 32-bit synthesizable soft-processor that is compatible with SPARC V8 architecture; it has a seven-stage pipeline and Harvard architecture, uses separate instruction and data caches and supports multiprocessor configurations in Symmetric Multiprocessor (SMP) mode. It represents a soft-processor for aerospace applications. The single-core reference implementation is shown in Fig. 5. The development board is the Xilinx ML605 Virtex-6 FPGA with 512 MB RAM.

Starting from GRLIB, a VHDLib library of IP cores for designing a complete system on chip centered around the LEON3 processor, the LEON3 processor has been customized with a system clock of 75 MHz per core and the following characteristics:

- 1 Cobham Gaisler LEON3 SPARC V8 Processor connected with AHB shared bus;
- 8 register windows;
- GRFPU High-Performance Floating-Point Unit;
- 2*8 KiB instruction caches, with 32 bytes per line with Least-Recently-Used (LRU) replacement algorithm;
- 2*4 KiB data caches, with 16 bytes per line with LRU replacement algorithm.

The final software partitioned system (suggested by the DSE activity) uses the Xtratum services to implement the FIR-FIR-GCD use case. All the processes are implemented as a bare-metal application into the partitions, where communication is allowed using sampling channels. Fig. 6 presents the comparison between the execution time on the real target (LEON3

---

* Unfeasible Solution (Minimum Cost function ≥ 1 [31])
single-core) and the simulation made with HEPSIM. The final average error estimation is under 2%, so the simulator is able to evaluate HPV timing behavior with a very limited error.

V. CONCLUSION AND FUTURE WORK

This work has presented an ESL HW/SW Co-Design approach able to take into account mixed-criticality and real-time constraints. The presented methodology, design flow and framework are able to drive the designer from the input specification to the final implementation solution, while offering timing simulation capabilities, DSE activities with the support of analysis tool, integrating this approach with external tools (in this scenario Xamber [33], but other tools are under evaluation). Despite of the obtained results, a lot of works...
should be made in future in order to consider the multi-core scenario [8] while introducing schedulability and real-time analysis, introduce fixed WCET values (taken from external tools) to be used also in the DSE step to improve allocation and binding of processes/tasks, and to improve performance, integrate other external tools to enhance HPESYCODE-MC functionality (i.e., art2kitekt [36], CHESS [37], CODEO [38]), and improve the hierarchical scheduling implementation considering Inter Partition Communication (IPC) overheads by means of benchmarking activities.

ACKNOWLEDGMENT

This work has been partially supported by the ECSEL RIA 2016 MegaM@Rt2, AQUAS and ECSEL RIA 2017 FitOptivis projects.

REFERENCES