2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

TwIRTee: design exploration with Capella and IP-XACT

Bassem Ouni[1], Philippe Cuenot[1][2], Pierre Gaufillet[1][3]
1 : IRT Saint-Exupéry, 118 route de Narbonne, 31042 Toulouse, France

***Abstract*— With the huge increase of embedded devices, model driven engineering has become increasingly important, coverering a large spectrum of abstraction levels. System models are exploited for design specification, system evaluation, verification and validation. Nowadays, no single modeling language or environment covers all these aspects. For instance, while the Capella tool fits the earliest stages of the development process, the IP-XACT standard provides means for tool capabilities to refine the hardware design during the later phases of the design. While using different modeling languages for different purposes is perfectly acceptable in a development process, it is important to guarantee that information remains consistent across all models. This is why a formalized bridge between Capella and IP-XACT has been built. In this paper, this transformation is described. The whole approach is illustrated by the design of TwIRTee – the INGEQUIP project's rover demonstrator.**

Keywords*— *Embedded systems, Model driven engineering, Model transformation, Capella, IP-XACT.

I. INTRODUCTION

This work forms part of the the INGEQUIP project being undertaken at the Toulouse *Institut de Recherche Technologique* (IRT) Saint-Exupéry. IRTs are new research structures established under the leadership of the French *Agence Nationale de la Recherche* (ANR). IRTs target the transfer of innovation from laboratories to industries. Engineers are gathered from companies of all sizes, from various industrial domains, and from public universities and national research agencies. The INGEQUIP project covers the space, aerospace and automotive systems domains.

The INGEQUIP project aims to study and to propose solutions for supporting closely integrated development of the main technical domains involved in embedded equipment engineering – system, electronic and software engineering. A key reasearch topic is to ensure the continuity and the consistency of artefacts across the complete engineering development chain. The relation between system engineering and domain engineering is made with model based techniques and model transformations. Transformations are used to fill the semantic gap between different modeling environments.

Engineering activity starts with requirements capture and refinement. The set of requirements is usually divided into two categories: functional requirements and non functional requirements. Functional requirements include the system's behavior, capabilities and characteristics as specified by stakeholders, whereas non-functional properties or requirements define criteria that can be used to evaluate the operations of the system. Specific modeling languages can be used to capture these specific aspects of the system: Capella/Arcadia [1] or SysML [2] for system engineering, AADL [3] or EAST-ADL [4] for detailed architecture description, UML [5] mainly for software architecture, AUTOSAR [6] as a dedicated language for automotive software architecture, SystemC [7] for hardware architecture with hardware/software simulation capabilities. In the INGEQUIP project, the complementary choice of languages is Capella, AADL, SystemC and IP-XACT. Capella has been selected as its functional representation is closer to the engineering culture in the embedded domain. AADL has been chosen because of its strong connection to formal behavioral verification tools and the ability to generate distributed code. SystemC was selected because the language is standardized and allows virtual platforms to be built for hardware/software co-simulation. However, commercial tools and related model libraries do not permit a complete interoperability between hardware platforms and system architectures. To cope with this issue it has been decided to interface Cappella with the hardware architecture with IP-XACT [8], the interchange format standard for hardware design and component description.

The combination of Capella and IP-XACT is well suited to provide most of the viewpoints commonly used by designers of embedded equipment at system and hardware levels: functional breakdown, logical and physical

[2] Seconded from Continental Automotive France, 1 avenue Paul Ourliac, 31036 Toulouse France

[3] Seconded from Airbus, 316 route de Bayonne, 31000 Toulouse, France.

architecture, hardware architecture. As the implementation of Capella is built on top of the Eclipse Modeling Framework, it allows tool extension to be easily developed (in Java), in particular for model transformations.
Hence inline with the goals of the INGEQUIP project, a transformation between Capella and IP-XACT has been defined and developed, ensuring a seamless transition from the system engineering level to hardware engineering level, facilitating the use of electronic system simulations for hardware/software exploration.

In this paper, section II starts by introducing the state of the art of model driven engineering. Section III then presents the process context, a first comparison between the semantics of Capella and IP-XACT, the mapping and the transformation between Capella physical components and IP-XACT concepts. The whole approach is illustrated in section IV by some elements extracted from the design of TwIRTee – the INGEQUIP robotic demonstrator of INGEQUIP. Finally section V presents our conclusions.

II. RELATED WORKS

To cope with early specification analysis, several studies have proposed similar transformations between high level models, with verification of functional and non-functional properties of the system, and with code generation.

In [9], UML-MARTE has been used to represent the various features of IP-XACT. This work proposes ways for IP-XACT to be extended with timing formalisms, while MARTE provides graphical editing functionalities and extension capabilities. While UML/MARTE allows the hardware architecture to be captured, still relation to system engineering is weak. Moreover, UML is a generic description not fully adapted to this scope.
In [10], AADL is used to model the properties of embedded system architectures, including application's tasks, hardware platforms and operating system services. This allows the characterization of energy overheads of an embedded operating system. The authors propose an AADL model transformation to a specified XML format serving as input to a multiprocessor simulation tool named STORM (Simulation TOol for Real-time Multiprocessor scheduling). AADL provides hardware and software architectures together with scheduling policies. STORM simulates the system behaviour with all its characteristics (task execution time, processor operating conditions, etc.). It generates the chronological ordering of all the scheduling events that occur at run time. It also computes various real-time metrics to analyse the system behaviour and its performances from various point of views. This approach only considers software characteristics of hardware platforms, which limits the use for this sort of hardware simulation during explorations.
In [11] and [12], the design flow for system electronic design, hardware architecture to hardware implementation (and simulation) is simplified thanks to High Level Synthesis techniques. The approach in INGEQUIP does not attempt to cover hardware implementation, but rather fills the gap between system engineering and hardware/software domains.

In this paper, the transformation from Capella physical model to IP-XACT concept is presented. It provides a way to transfer designs between various languages and therefore to build a consistent description of hardware components. Moreover it makes the hardware architecture virtualization faster and easier.

III. PROPOSED APPROACH

A. Principles

The first question is the level at which the Capella to IP-XACT transformations should be performed. Capella is clearly positioned on the most abstract part of the system development process with the Operational Analysis, focusing on the capture of stakeholder needs, and the System analysis, focusing on the functional definition of the system. Then the functional architecture is broken down into logical and physical architectures describing abstract representation of hardware and software components. Meanwhile, an IP-XACT hardware design model (supported by EDA tools) describes details of hardware components, design configurations and assemblies. Physical models in Capella share concepts with IP-XACT, in particular the definition of hardware components and communication between components of hardware architectures.

The system engineering approach and the modeling abilities of Capella can be summarized by the following layers of abstraction:

- At the operational level, the customer needs, the actors, the missions and the activities are described.
- At the system level, a Capella model defines how systems can satisfy the operational needs described

in the previous level.

- The Capella logical modeling level starts from functional and non-functional analysis and gathers related functions into logical components.
- The implementation of logical components is performed at the physical level. This physical architecture of the system introduces architectural patterns, services and components. It also takes into account implementation constraints and choices.

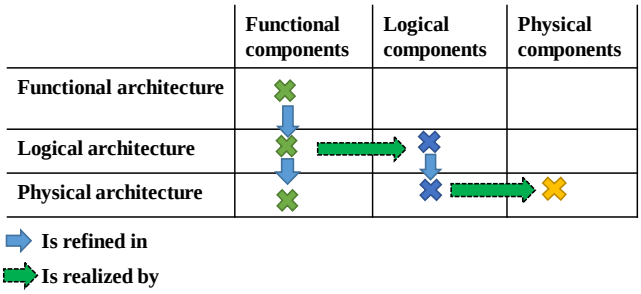

Figure 1 – Capella functional, logical and physical architectures

The Figure 1 summarizes the relationship between functional, logical and physical architecture in a Capella model. The physical architecture includes functional, logical and physical components. IP-XACT allows the specification of additional information about hardware components which are not necessarily known or required during system analysis. Typically such information would include port identification, hardware IP register definition, memory mapping information, bus abstraction detailed and pin allocation, etc. Consequently, the simplest translation between the two languages is at the physical architecture level. Considering that Capella is well suited to describing the logical architecture and a first abstract level of physical architecture, the aim is to generate as many as possible hardware artefacts in the IP-XACT models directly from the Capella description. Then, the assembly of hardware IP components is performed using EDA tools. Moreover, it is possible to generate SystemC skeletons (which must be extended manually) that are the basis for SystemC models, which capture of component behavior.

B. Electronic design process

The design process proposed in the INGEQUIP project is summarized in Figure 2. It depicts the complete set of engineering activities showing which tool environment and modeling language they use.

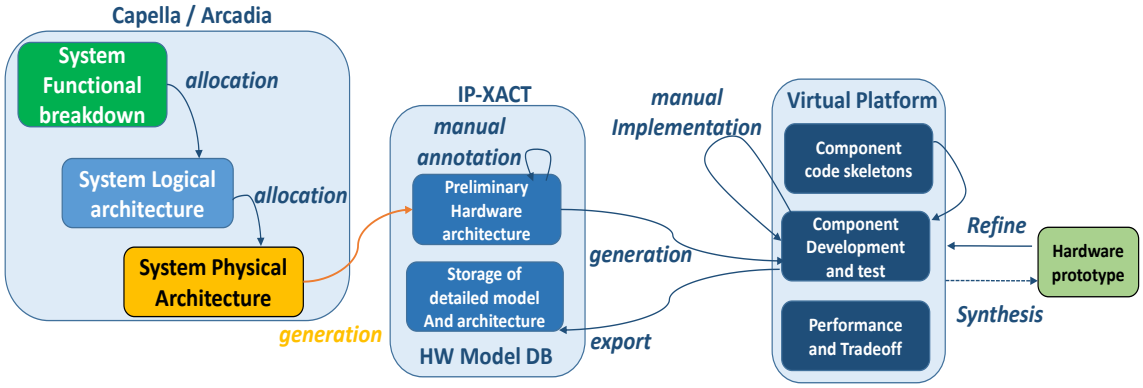

Figure 2- Equipement design process

First Capella captures the system by breaking down its high level functions. Then, it allocates the resulting elementary functions into the desired component of the logical architecture. The physical architecture captures the hardware topology of the system on which the software components are allocated for execution. Processing element such as core processors or hardware accelerators (FPGA, ASIC), memory banks and buses for interconnection can be modeled too (CAN, Ethernet). The construction allows the decomposition of complex SoC components (e.g. processors connected via bus controllers and bridges for specific hardware). At this point only structural information is gathered.

The designed electronic or hardware architecture is then transformed into IP-XACT to allow hardware domain engineering and to elaborate a virtual platform as an assembly of SystemC components. IP-XACT models are typically used for generating:

a) The virtual platform as an assembly of selected components (e.g. an electronic design is composed of a set of SoCs interconnected via, for instance, external serial communication busses (see Figure 7 of section IV).
b) The virtual platform, as an assembly of component interconnected with memory mapped bus and signals (see Figure 10 of section IV).
c) The initial description of hardware components which can then be refined to capture detailed design. The information captured can be a register definition for a hardware component, a memory space area for a bus controller, a pin logical to physical allocation for a bus transaction abstraction, a port pin number for a dedicated electronic signal, etc. This information is then exploited by EDA tools, to generate SystemC skeletons for models using TLM-2.0 bus transactions, including the code infrastructure (e.g. R/W access methods after address decoding, register declarations, etc.). The behavioral code of the hardware component has then to be written manually. When the component design is completed, it can be imported into a higher level IP-XACT design as in b).

The simulation capability of the virtual platform makes early verification and exploration of both hardware and software possible. Performance analysis can be used to ensure a robust and sound design.

C. Tool environment

The Eclipse modeling framework tool (EMF) [13] is used to transform Capella to IP-XACT models. The inputs are the Capella models and the meta-model of IP-XACT generated from its XML schema definition. This meta-model is described using the EMF language Ecore, which consists, in a simplified version, of the same concepts as a UML class diagram. As shown in Figure 3, the model transformation parses the physical components of the physical architecture in Capella, and generates the hardware design in IP-XACT.

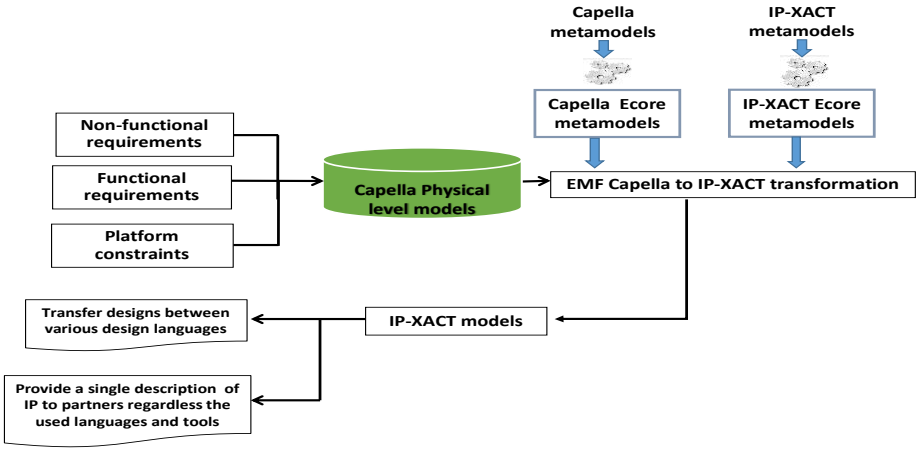

Figure 3- Capella/IP-XACT transformation methodology

D. Transformation algorithm

The Capella/IP-XACT transformation algorithm needs to explore the elements of the Capella physical model to generate the appropriate IP-XACT code, following the concepts mapping proposed in Table 1.

A Capella physical component node with 'kind' *SOFTWARE_EXECUTION_UNIT* is equivalent to an IP-XACT component with optional specialization as *processor*. Physical components having a 'nature' tagged as *BEHAVIOR* and 'kind' *SOFTWARE_APPLICATION*, which represents a software process, are not considered, as the transformation only manages hardware artefacts.

Case	Capella	Condition	IP-XACT
A	PhysicalComponent	nature = PhysicaComponentNature::*NODE* and kind = PhysicalComponentKind::*SOFTWARE_EXECUTION_UNIT*	CPU type
B	PhysicalComponent	other than above	Component type (including VLNV identifier, the model type, design type, view type)
C	PhysicalPort	See Bus extraction algorithm (The generated buses) C.1) Each physical port is parsed to a transactional port type	C.1) Transactional port type (Ports type, port type)

D	PhysicalLink	See Bus extraction algorithm (The generated buses) D.1) The Capella generated buses D.2) For each link between a physical component and a bus, a bus interface is generated in the physical component. This interface includes port mapping between the component's physical ports and the abstraction definition's logical ports D.3) The transformation searches links between components within the same design. They will be transformed into interconnections under the container component design with their active interfaces. D.4) Links between a component and its container are then transformed into interconnections under the container component design with an active and hierarchical interface.	D.1) Abstraction definition/Bus definition pair (with mandatory tags and links between them). D.2) Bus interface (including bus type, abstraction types, abstraction ref, port maps, VLNV) D.3) Interconnection Type (active interfaces referring bus types) D.4) Interconnection Type (active interface and hierarchical interface)

Table I. Mapping of Capella – IP-XACT concepts

Physical buses components are not native model element in Capella meta-models. For this reason, the algorithm extracts physical buses from Capella model elements (thanks to Capella Ecore Java APIs). It maps the buses to the IP-XACT concept of bus definition and to the abstraction definition types. Identifying buses achieved by exploring the Capella physical components and the physical links (as communication/transportation means linking node's Physical components and physical ports).

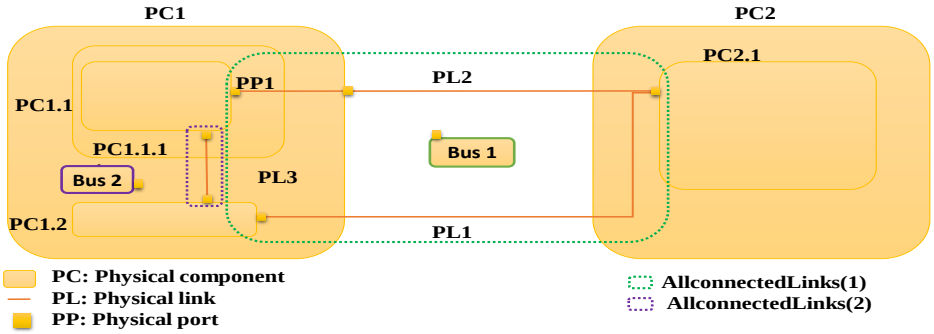

Figure 4 – Determination of connected physical links

As depicted in Figure 4, the physical links connected by physical ports are gathered in a list ("*AllconnectedLinks"*). For each element of the list, a physical bus component, with a bus port, will be generated to bind the physical links. Then, as showed in Figure 5, buses are connected to external ports (*tip ports*) which are ports of physical components having no subcomponents. This established link is called a *BusLink*.

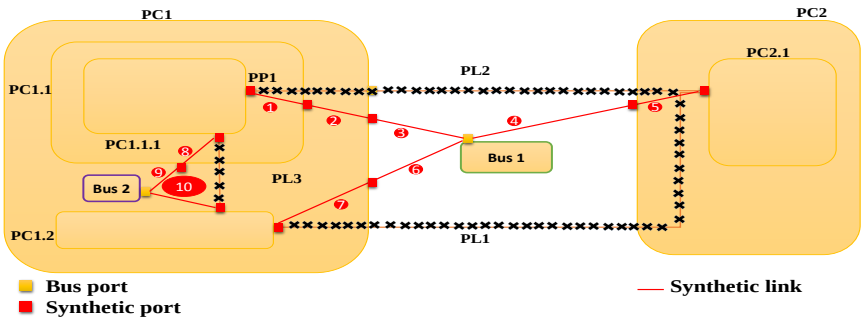

Figure 5 – Generation of synthetic links and ports

*BusLink*s are then divided into segments that don't cross physical component boundaries. These segments represent synthetic links. Synthetic ports include the *tipports* and new ports which are generated at each intersection between *BusLink*s and physical components. The algorithm describing buses extraction is detailed in algorithm 1. The output of this algorithm is the list of synthetic ports and links that will be mapped to IP-XACT elements.

The EMF APIs are then used to explore the contents of the Capella model and to generate the appropriate IP-XACT code following the Capella/IP-XACT concepts mapping proposed in Table I.

1. Get the physical architecture from Project in Capella model.
2. Extract the list of physical ports ***pps*** physical links ***pls*** and a cross reference between them.
3. Determinate the list ***allConnectedLinks*** including the sets of ***pls*** connected by ***pps*.**
4. **for** each element of ***allConnectedLinks***
    a. Get the category of the link
    b. Create the list of external ports ***tipports***
    c. Create a bus instance having the same name than the category
    d. Associate the bus instances with ***tipports*** in map ***bus2TipPorts*** (key=bus instance, value=tipports)
5. **end for**
6. Initialize the list of synthetic ports with the tip ports and synthetic links.
7. **for each** key in ***bus2TipPorts***
    a. Search the closest common physical component ancestor ***PcCommonAncestor*** of all ***tipPorts*** associated to key
    b. Create the Physical component ***PCBus*** that will model the bus instance
    c. Add a port **BusPort** to the ***PCBus***
    d. Add **PCbus** to **PcCommonAncestor**
    e. **for each** port ***p*** of ***bus2TipPorts[key]***
        i. Create a link from tip port ***p*** to ***BusPort***
        ii. Divide this link into segments called synthetic links that don't cross Physical Component boundaries
        iii. For each segment create intermediate ports – synthetic ports – as needed
    f. **end for**
8. **end for**

Algorithm 1 – Capella Bus generation algorithm

All the Capella ports are connected via a physical link. Actually all physical links are transformed into bus (bustype and abstractionref) in IP-XACT, connecting ports with transactional types. The impact is that simple ports connected by a signal are then managed as a bus. This limitation may be removed in the future by identifying more precisely the nature of physical links thanks to a specific property in Capella.

IV. APPLICATION TO THE TWIRTEE ROVER

TwIRTee is a three-wheeled autonomous rover developed within the INGEQUIP project. Its operational role is to move safely on some predefined tracks, along a map segment from a point A to a point B (a "mission"), and to optimize its path while avoiding collision with other rovers. To complete this mission, it is fitted with several sensors (camera, odometer sensors, global positioning…) and two main actuators (motors).

The development of the rover is not an objective *per se.* Indeed, TwIRTee is designed to allow us to investigate the engineering topics, namely: early validation, architecture exploration, performance prediction, and formal verification. Its missions, functions and architectural elements are designed to tackle or exercise one or several issues: for instance, the positioning function relies partially on imaging so as to exercise hardware/software exploration, co-design, and implementation in FPGA; the highly redundant architecture provides the experimental setup to perform early performance evaluations using virtual platforms (including dependability), etc. The computing platform of TwIRTee is composed of 2 COM(PPC)/MON(ZynQ) channels that host the main mission functions and one channel dedicated to power supply generation and motor control. The subsystems are mainly interconnected by a CAN bus.

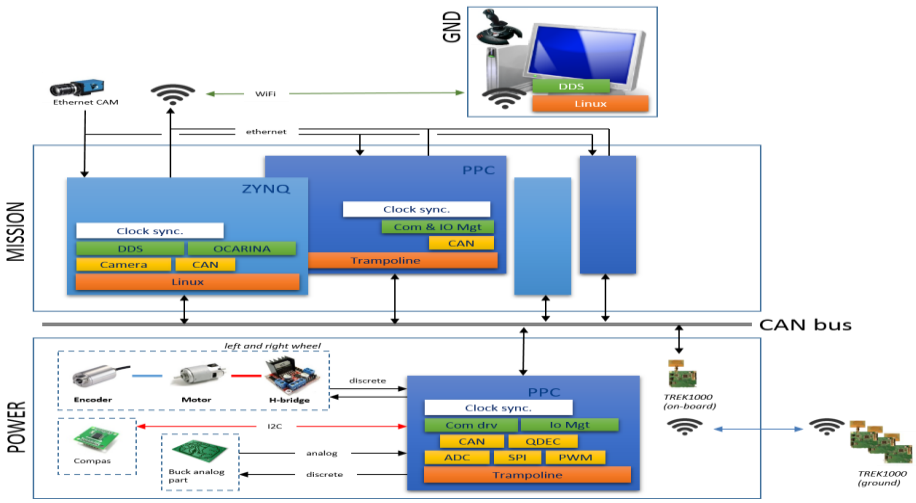

Figure 6 – TwIRTee rover architecture overview

To validate the Capella to IP-XACT transformation accordingly to the electronic design process defined in Section III B, subsets of TwIRTee's Capella physical architecture have been used. The first example, Figure 7, shows three ECUs, respectively COM_1, MON_1 from the mission cluster and the PowerUnit from the power cluster connected to a CAN bus interface. The IP-XACT file is then used to generate a Python script that assembles the models (PPC1, PPC2 and Zynq1) to create the virtual platform in the VLAB [14] simulator (one of the simulators used in the INGEQUIP project). This part is not described in this article.

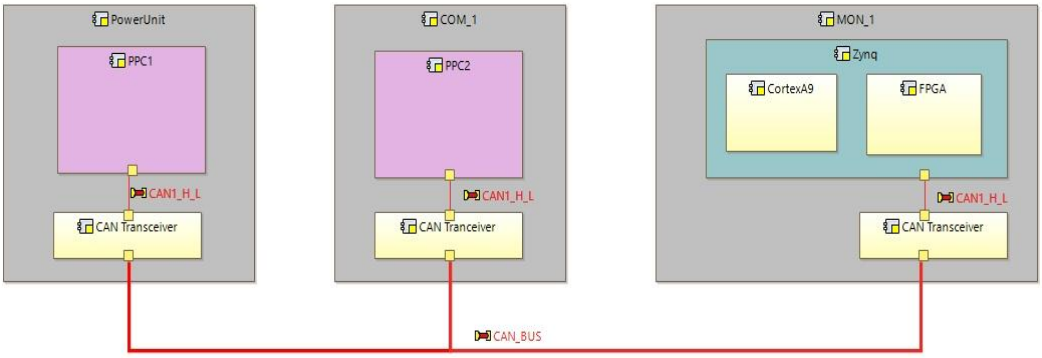

Figure 7 –Twirtee Capella physical view: COM_1/COM_1/PowerUnit ECU

An extract of the generated IP-XACT files are visible in the following Figure 7 and Figure 8.

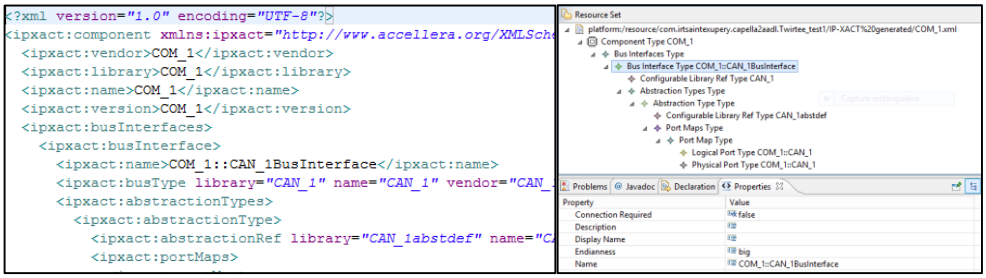

Figure 7 – COM_1 IP-XACT component

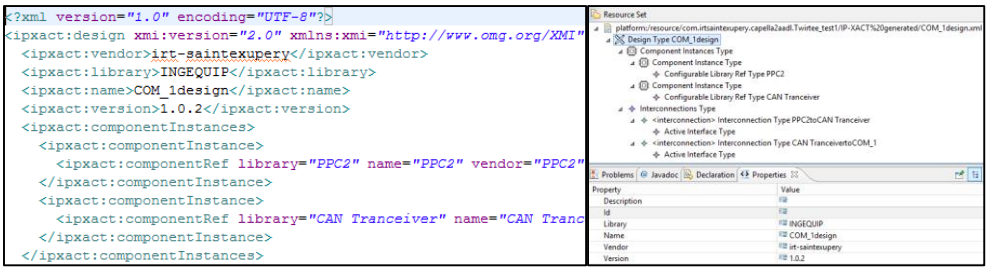

Figure 8 – COM_1 component's design in IP-XACT

Moreover, various other kind of buses can be managed by the transformation. The electronic design of the image processing, embedded in the Zynq, requires a complex architecture with a processor, a memory, an FPGA and a memory mapped bus with a controller and a bridge between the different buses. As showed in Figure 9, the AHB bus controller with master and slave components is connected to the peripheral bus (APB) via a bridge that performs the write/read I/O registers access to the hardware blocks implemented in the FPGA. The dedicated wired FIFO connections and the DMA controller accelerates the communication between the hardware blocks and the memory. They are both implemented in the FPGA. Then, the generated IP-XACT model is modified manually to describe the memory map region of the AHB_Bus_Controller and of the AHB_to_APB_Bridge, to define the logical and the physical port mapping in the bus abstraction definition and to declare the I/O registers of the hardware blocks (resizing, histogram equalization, etc.). For the hardware block, once the complete structural and connectivity descriptions are available, SystemC skeletons are generated.

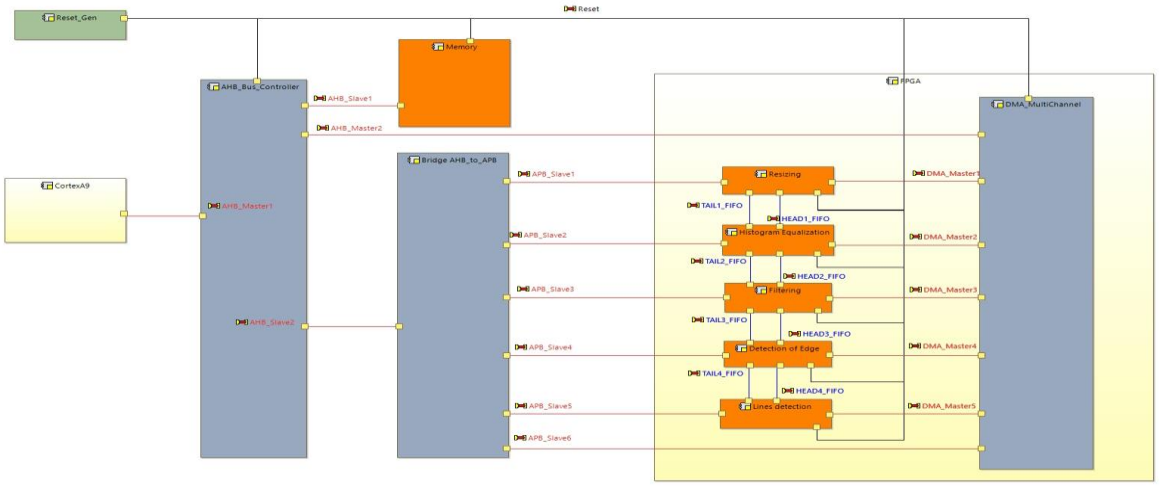

Figure 9 –Twirtee Capella physical view: COM_1 ECU design

V. CONCLUSION

Beyond co-design, a *"shared-design"* is probably the next step toward mitigation of the risks of hardware software integration. The experience in INGEQUIP shows that Capella is missing some major information related to the software domain: mainly the identification of tasks and real time properties. It is also limited in terms of bus-oriented architectures, though we have tried to address this issue in this paper. AADL, which supports such information, is maybe a better language to consider s we strive for a shared model of architecture. IP-XACT has shown its capabilities in terms of interoperability between EDA tools. Its extension to support low level software (e.g. drivers) also allows us to maintain the consistency between hardware dependent software architecture models and platform models. The challenge is therefore to ensure the mapping of such IP-XACT representations to AADL software descriptions.

ACKNOWLEDGMENT

The authors would like to thank all the people and industrial partners involved in the INGEQUIP project. This work is supported by the French Research Agency (ANR) and by the industrial partners of IRT Saint-Exupéry Scientific Cooperation Foundation (FCS): Actia Automotive, Airbus[(3)], Airbus Defense and Space, Continental Automotive[(2)], SAGEM, Systerel, Thales Avionics, ASTC Design Partners, Space Codesign Systems and GreenSocs.

References

[1] "Capella tool environement and Arcadia methodology," Thales group, 2015. [Online]. Available: https://www.polarsys.org/capella/arcadia.html.

[2] "SysML," [Online]. Available: http://www.omgsysml.org/.

[3] "Architecture Analysis and Design Langage (AADL), SAE standards," [Online]. Available: http://standards.sae.org/as5506/.

[4] "EAST-ADL," [Online]. Available: http://www.east-adl.info/Specification.html.

[5] "UML," [Online]. Available: http://www.uml.org/.

[6] "AUTOSAR," [Online]. Available: http://www.autosar.org/.

[7] SystemC. [Online]. Available: http://accellera.org/downloads/standards/systemc.

[8] I. c. society, 1685-2014 - IEEE Standard for IP-XACT, *"Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows", 2014.*

[9] C. André, F. Mallet, A. Mehmood khan and R. De Simone, "Modeling SPIRIT IP-XACT with UML MARTE," *In Design Automation and Test in Europe (DATE), MARTE Workshop*, 2008.

[10] B. Ouni, C. Belleudy and E. Senn, "Accurate energy characterization of OS services in embedded systems," *EURASIP Journal on Embedded Systems*, vol. 6, 2012.

[11] P. Coussy, D. Heller, and C. Chavet, "High-Level Synthesis: On the Path to ESL Design", *In Proceedings of the 9th International Conference on ASIC (ASICON)*, 2011.

[12] P. Coussy and A. Morawiec (Eds.), "High level Synthesis: from Algorithm to Digital Circuit", *Springer*, 2008.

[13] "Eclipse Modeling Framework," [Online]. Available: https://eclipse.org/modeling/emf/.

[14] "VLAB is a product of ASTC," [Online]. Available: http://www.vlabworks.com/vlab.