2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

TwIRTee design exploration with Capella and IP-XACT

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

***Abstract*— With the huge increase of embedded devices, model driven engineering becomes necesseray in order to cover a large spectrum of multiple abstraction levels. System models are exploited for design specification, system evaluation, verification and validation. Nowadays, no single modeling language and environment covers all these aspects. While Capella tool fits well to the most early stages of the development process, IP-XACT standard provides powerful capabilities to refine the design artifacts of the hardware point of view that appear during the latest phase of the design. While using different modeling languages for different purpose is perfectly acceptable in a development process, it is important to guarantee that information remains consistent across all models. This is why we build a formalized bridge between Capella and IP-XACT. In this paper, the transformation Capella /IP-XACT is described. The whole approach is illustrated by the design of TwIRTee – the robotic demonstrator of INGEQUIP – before concluding.**

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

I. INTRODUCTION

This work is achieved under the INGEQUIP project at the Toulouse *Institut de Recherche Technologique* (IRT) Saint-Exupéry. IRTs are new research structures established under the auspices of the French *Agence Nationale de la Recherche* (ANR). IRTs are aimed at favouring the transfer of innovation from laboratories to industries. Towards this goal, IRTs gather engineers coming from small to large companies from various industrial domains, and researchers from public universities and national research agencies. As an example, INGEQUIP covers the space, aeronautics and automotive systems domains.

INGEQUIP team targets to study and propose solutions for supporting closely integrated development of the main technical domains involved in embedded equipment engineering – system, eleectronics and software engineering. A key element for reaching such goal is to ensure the continuity and consistency of information in the whole chain of activities. In INGEQUIP, the choice has been to obtain this property by relying on models and model transformations. The set of requirements regarding an equipment 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. In order to design the system and associate to the design elements the realized functional and non functional requirements, several modeling languages are available among which AADL [1] – *Architecture and Analysis Design Language*, AUTOSAR [2], Capella [3], EAST-ADL [4], SysML [5] and UML [6] have been considered. While Capella, EAST-ADL and SysML fit system engineering, AADL, AUTOSAR and UML are focused on software and IP-XACT [7] refines the hardware architecture.

The transformation Capella/IP-XACT is adapted in INGEQUIP in order to provide most viewpoints commonly used by designers of embedded equipment at system and hardware levels: functional breakdown, logical and physical architecture, hardware architecture, formal behavioral descriptions; they are supported by a number of tools widely available: Capella and Eclipse Modeling Framework which allows to develop easily tools extensions.

Consistently with the orientation of INGEQUIP, a transformation has then been defined and developed for ensuring a seamless transition from system engineering stage to hardware engineering stage.

a Seconded from Airbus, 316 route de Bayonne, 31000 Toulouse, France.
b Seconded from Continental Automotive France, 1 avenue Paul Ourliac, 31036 Toulouse France.

In this paper, we therefore begin in section 2 by introducing the state of the art of engineering models transformation. Then, section 3 presents a first comparison between the semantics of Capella and IP-XACT and describes the mapping of the transformation between Capella physical model and IP-XACT. The whole approach is illustrated by some elements coming from the design of TwIRTee – the robotic demonstrator of INGEQUIP – in section 4 before concluding in section 5.

II. RELATED WORKS

In order to achieve an early analysis of the specification, the verification of functional and non-functional properties of the system, and even code generation for the targeted hardware platform, several studies have proposed comparable transformations between high level models.

In [8], the UML-MARTE has been used to represent the various IP-XACT features. The authors focus only on four top-level elements. This work allows possibility for IP-XACT to extend the formalism with timing aspects, while MARTE provides graphical editing functionalities and a way to experiment on extensions.
In [9], AADL is used for modelling the properties of embedded system architecture, including the application's tasks, the hardware platform and the operating system services in order to characterize the energy overhead of embedded operating system . The authors propose an AADL model transformation in order to be exploited by a multiprocessor simulation tool named STORM (Simulation TOol for Real-time Multiprocessor scheduling). AADL provides hardware and software architectures together with the scheduling policy; STORM simulates the system behaviour using all the characteristics (task execution time, processor functioning conditions, etc.) in order to obtain the chronological track of all the scheduling events that occurred at run time, and compute various real-time metrics in order to analyse the system behaviour and performances from various point of views.
In this paper, the transformation of Capella to IP-XACT models target to cover the various levels of abstraction when modeling systems. We take into account the system behavior and the hardware/software mapping. Next section will detail the transformation flow and how it exploits the complementarity of Capella and AADL in order to cover various embedded system aspects at high level modeling step.

III. CAPELLA/IP-XACT TRANSFORMATION APPROACH

A. The proposed approach

As Capella and IP-XACT partially overlap, the first question that has to be answered before defining the transformations from Capella to IP-XACT is the level at which this transformation 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 stakeholders' needs, and the System analysis, focusing on the functional definition of the system. As for IP-XACT, it doesn't offer support for such kind of analysis. As they deal with the system's architecture, Capella's physical models share lots of concepts with IP-XACT, in particular the capability to express hardware architecture components. IP-XACT however goes further by delivering IP descriptions of components to EDA tools, and for exchanging IP descriptions of designs between EDA tools. Also, IP-XACT details the communication between components under the hardware architecture.

Capella provides system modeling capabilities at several layers of abstraction:

- At operational level, the customer needs, the actors, the missions and the activities are described.
- At system level, a Capella model defines how system can satisfy the former operational need.
- Capella logical level modeling starts from functional and non-functional analysis and builds one or several decompositions of the system into logical components.
- The building of logical components is performed at physical level: the "final" architecture of the system introducing architectural patterns, services and components, and it makes the logical architecture evolve according to implementation, constraints and choices.

Figure 1 summarizes the relationship between functional, logical and physical architecture in a Capella model. The Capella model's physical architecture includes functional, logical and physical components. In this work, we interest to parse the physical components of this physical architecture.
Whatever the abstraction level of the considered components is, IP-XACT also brings the capability to specify additional information about the hardware components compared to Capella. Consequently, the simplest articulation between the two languages is at the level physical architecture. Considering that Capella is well adapted to describing the logical architecture and a first abstract level of physical architecture, and that it is a

good practice to limit the risk of data duplication and inconsistency between Capella and IP-XACT models, the best solution is to delay as far as possible the transfer of information.

To transform Capella to IP-XACT models, Eclipse modeling framework tool (EMF) [10] is used, the input of the proposed approach is the Capella graphical and Ecore model. Also, we generate from the XSD meta-model of IP-XACT an Ecore reference meta-model. As shown in figure 2, the model transformation will therefore take as input the most detailed physical architecture in Capella, and the design process will go on in IP-XACT.

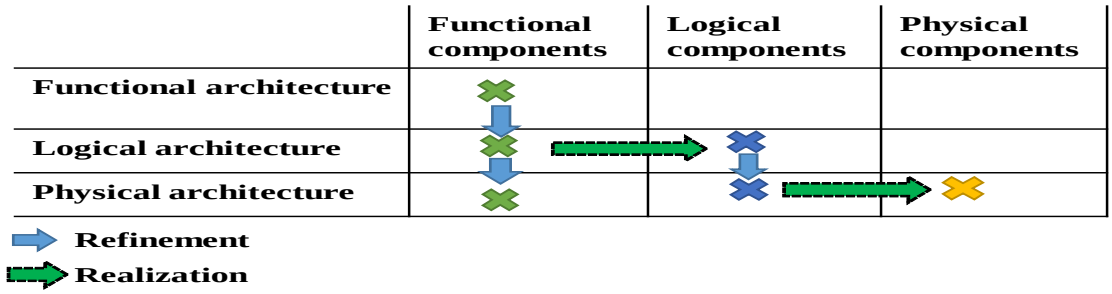

Figure 1- Capella functional, logical and physical architectures

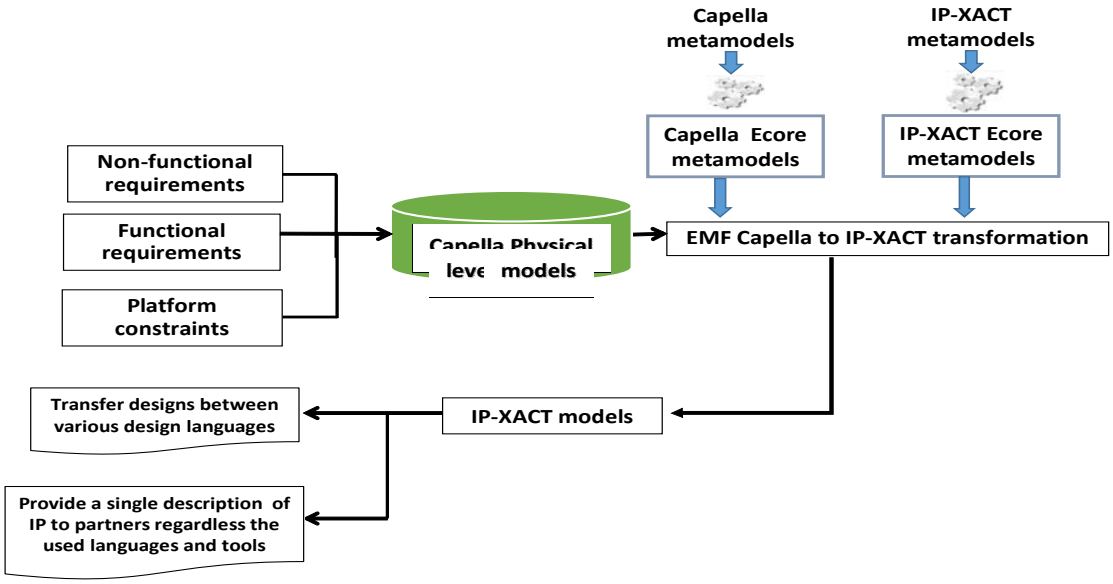

Figure 2- Capella/IP-XACT transformation methodology

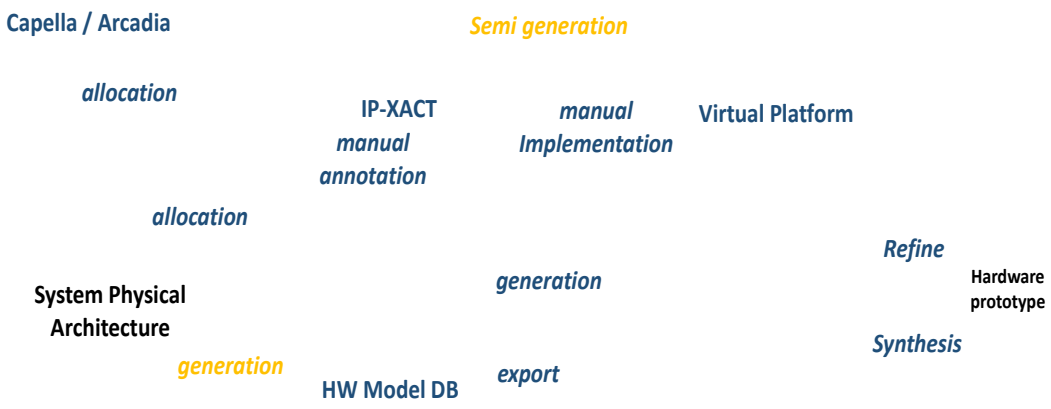

Figure 3: Equipment design process

The design process proposed in INGEQUIP project, summarized in Figure 3, depicts the engineering activities to support the conception of an electronic control unit and highlighted relationship between tool environment and modeling language. First Capella captures the system as introduced above, by taking care of modeling of function breakdown of the system, and then allocating them into the desired logical architecture. The definition of the physical architecture captures the hardware topology of the system on which the software components are allocated for execution. The processing element can be a processor or a hardware accelerator such as FPGA or ASIC. Moreover when electronic component architecture required to be detailed for the purpose of the future hardware detailed design, the construction allow the capture structural decomposition of complex component (can be processor, interconnect and specific hardware IP, etc...).

The hardware electronic architecture is then transformed into hardware domain interchange standard respectively IP-XACT, to gain benefit of supported feature for elaboration of virtual platform built with SystemC components. The relationship between a hardware physical component from Capella and an IP-XACT component is straightforward. The main purpose of the transformation is to make possible the assembly of the existing component and bus interface available from the SystemC library of virtual platform tool environment. Typical component can be for example a complete model of a microcontroller or a core processor connected with the required bus to specific hardware accelerator. When hardware accelerator need to be designed, its respective interfaces can be capture in IP-XACT (for example register declaration, memory region, etc...). The use of “template” feature of virtual platform tool permits to initiative the description of the hardware component to then implement the behavioral part. The overall electronic architecture with hardware component instantiation and binding is then performed according to relation and feature of IP-XACT import capabilities of the simulation tool or with a specific model to text transformation to comply with tool syntax. The verification and the validation of the complete electronic architecture are performed by simulation using the virtual platform. The performance and possible tradeoff analysis can be completed too in order to ensure a robust and sound design. It shall be noticed that all modification performed during simulation experimentation must re-imported in IP-XACT to capitalize electronic architecture independent of any simulation formalism.

B. The transformation flow

To transform Capella to IP-XACT models, we explore the contents of a Capella physical model and generate the appropriate IP-XACT code following the Capella/ IP-XACT concepts analogy proposed in Table 1. The equivalent of an IP-XACT processor and component are Capella node physical components with respectively a software execution unit and hardware kinds. A physical component with behavior nature and software application kind is not considered till we interest to the physical architecture.

Physical buses components are not modeled in Capella meta-models. For this reason, using Capella ecore meta-models, we generate Capella EMF code and exploit it using Java to extract physical buses from Capella model and map them to IP-XACT bus definition and abstraction definition types. The determination of buses is elaborated by exploring the Capella physical components, the physical links which are the communication/transportation means linking node Physical components, and physical ports. As depicted in figure 4, the physical links connected by physical ports are gathered in “*AllconnectedLinks”* list. 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 *BusLink*.

*BusLink*s are divided into segments that don't cross physical component boundaries. These segments represent synthetic links. Synthetic ports includes 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.

We exploit EMF plugin to explore the contents of a Capella model and generate the appropriate IP-XACT code following the Capella/AADL concepts analogy proposed in Table 1.

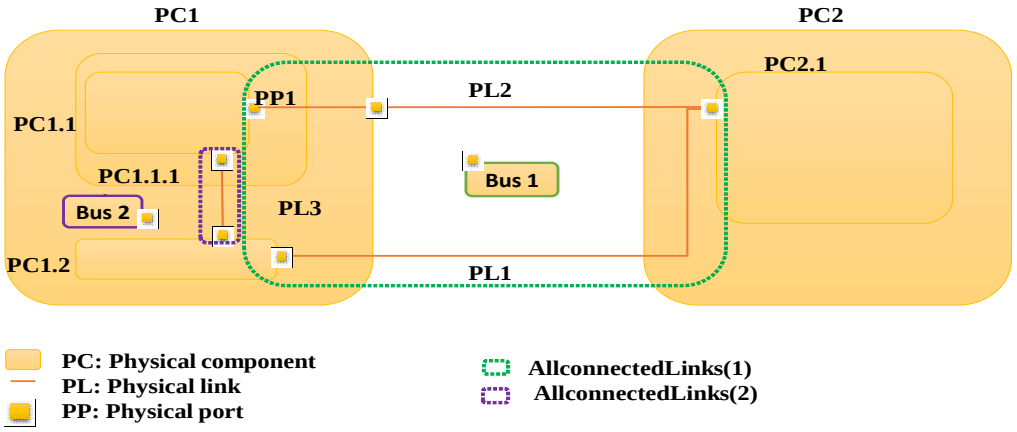

Figure 4 – Determination of connected physical links

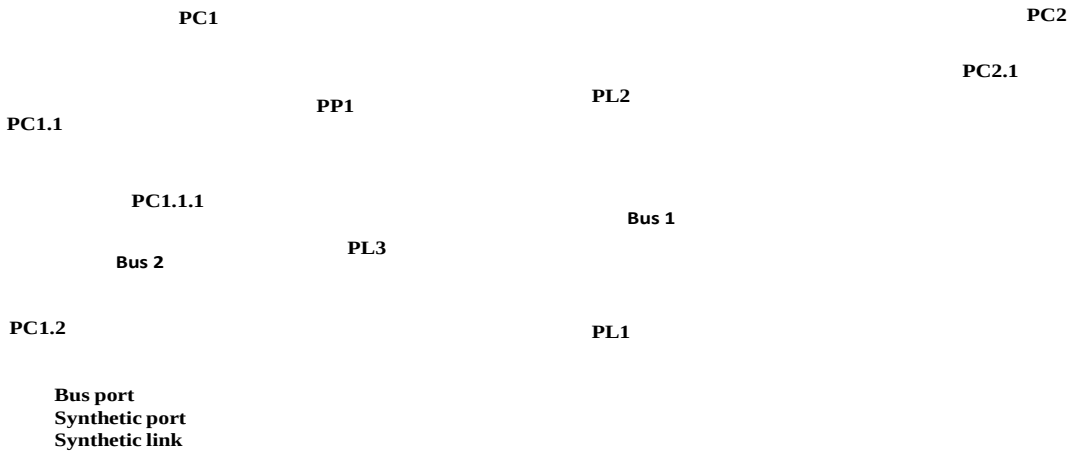

Figure 5 – Generation of synthetic links and ports

IV. APPLICATION TO THE TWIRTEE ROVER

TwIRTee is a three-wheeled autonomous rover developed within the INGEQUIP project. Its operational role is very simple: move itself on some predefined tracks from a point A to a point B (a "mission") while avoiding other rovers. To achieve this mission, it is fitted with several sensors (camera, odometry sensors, global positioning…) and two main actuators (motors).

The development of the rover is not an objective *per se.* Indeed, TwIRTee is designed so as to cover the major topics addressed in the project namely: early validation, architecture exploration, performance prediction, and formal verification. Furthermore, it is aimed at covering issues, and functional and architectural elements specific to the three industrial domains. Accordingly, missions, functions and the architectural elements are determined so as to tackle or exercise one or several issues: for instance, “the localization” function relies partially on imaging so as to exercise hardware / software space exploration and co-design; the highly redundant architecture provides the experimental setup to perform early performance evaluations (including dependability), etc.

The computing platform of TwIRTee is composed of 2 COM/MON channels that host the main “mission” functions and one channel dedicated to power supply generation and motor control. In order to cover issues related to the development of safety-critical Man Machine Interfaces, the system also contains a remote operator station. Figure 6, Figure 7 and figure 8 show the elements of TwIRTee design in Capella and their corresponding elements in IP-XACT.

Algorithm 1 – Capella Bus generation algorithm

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 of category
    d. Associate the bus instances with ***tipports*** in a BiMap structure ***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** element of ***bus2TipPorts***
    a. Search the closest common physical component ancestor ***PcCommonAncestor*** of ***tipPorts***
    b. Create the Physical component that will model physically the bus instance: ***PCBus***
    c. Add new properties to ***PCBus***: ***isBus*** and ***Bustype*** property
    d. Add a port **BusPort** to the ***PCBus***
    e. Add **PCbus** to **PcCommonAncestor**
    f. **for each** port ***p*** of ***bus2TipPorts***
        i. Create a link from tip port ***p*** to ***BusPort***
        ii. Divide this link into segments that don't cross Physical Component boundaries
        iii. Update the list of synthetic ports and links
    g. **end for**
8. **end for**

Table I. Mapping of Capella – IP-XACT concepts

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, we define a bus interface in the physical component with port mapping between the component's physical ports and the abstraction definition's logical ports D.3) We search links between components within the same design (friend component links), that will be transformed to interconnections under the container component design with their active interfaces. D.4) We search hierarchical links between a component and its container (component to container component links) that will be transformed to interconnections under the container component design with an active interface 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)

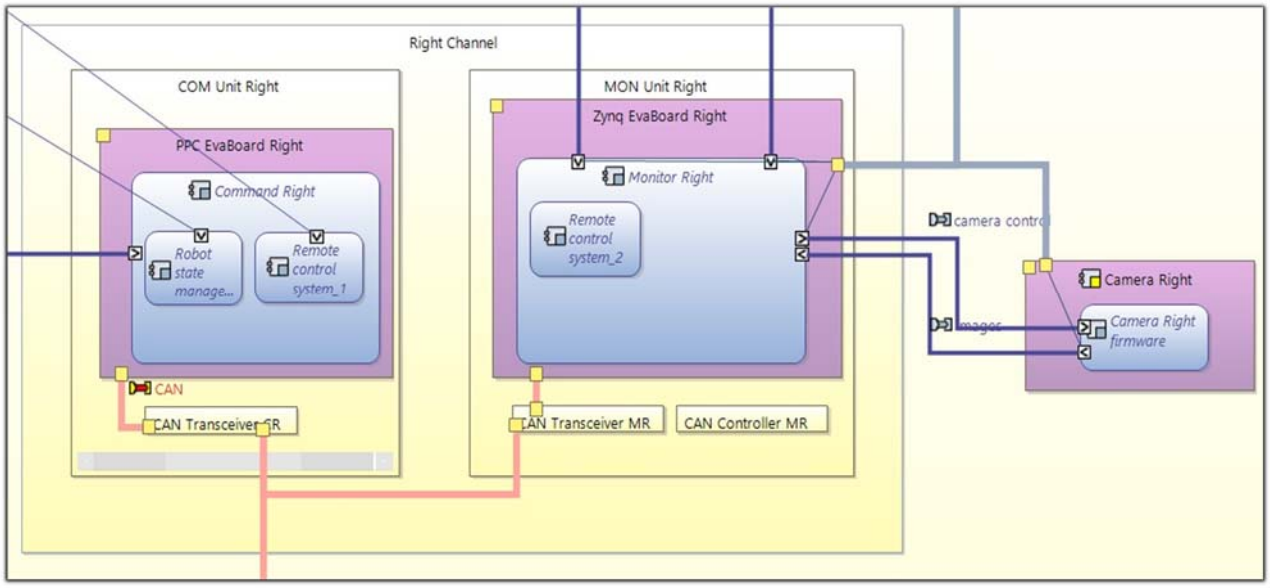

Figure 6 – Right channel design: Twirtee Capella model part

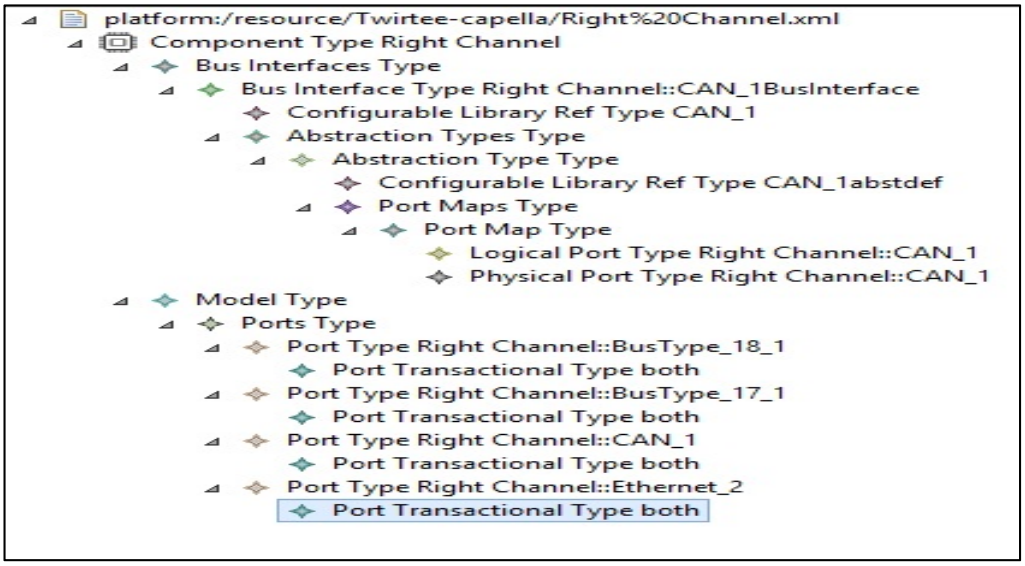

Figure 7 – Right channel IP-XACT component

V. CONCLUSION

In this paper, we presented a transformation from high level Capella physical architecture to a preliminary hardware architecture in IP-XACT. The goal of this approach is to implement a seamless development process from system definition to hardware components design. The approach has been applied and validated during the design of the TwIRTee rover demonstrator. The resulting physical model has been used to evaluate system performance and explore the design space.

ACKNOWLEDGMENT

The authors thank all 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 (a), Airbus Defense and Space, Continental Automotive (b), SAGEM, Systerel, Thales Avionics, ASTC Design Partners, Space Codesign Systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<ipxact:design xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014">
  <ipxact:vendor>Right Channel</ipxact:vendor>
  <ipxact:library>Right Channel</ipxact:library>
  <ipxact:name>Right Channeldesign</ipxact:name>
  <ipxact:version>Right Channel</ipxact:version>
  <ipxact:componentInstances>
    <ipxact:componentInstance>
      <ipxact:componentRef library="COM Unit Right" name="COM Unit Right" vendor="COM Unit Right" version="COM Unit Right"/>
    </ipxact:componentInstance>
    <ipxact:componentInstance>
      <ipxact:componentRef library="MON Unit Right" name="MON Unit Right" vendor="MON Unit Right" version="MON Unit Right"/>
    </ipxact:componentInstance>
  </ipxact:componentInstances>
  <ipxact:interconnections>
    <ipxact:interconnection>
      <ipxact:name>COM Unit RighttoRight Channel</ipxact:name>
      <ipxact:activeInterface busRef="COM Unit Right::BusType_18_1BusInterface" componentRef="COM Unit Right"/>
      <ipxact:hierInterface busRef="Right Channel::BusType_18_1BusInterface"/>
    </ipxact:interconnection>
    <ipxact:interconnection>
      <ipxact:name>COM Unit RighttoRight Channel</ipxact:name>
      <ipxact:activeInterface busRef="COM Unit Right::CAN_1BusInterface" componentRef="COM Unit Right"/>
      <ipxact:hierInterface busRef="Right Channel::CAN_1BusInterface"/>
    </ipxact:interconnection>
    <ipxact:interconnection>
      <ipxact:name>MON Unit RighttoRight Channel</ipxact:name>
      <ipxact:activeInterface busRef="MON Unit Right::BusType_17_1BusInterface" componentRef="MON Unit Right"/>
      <ipxact:hierInterface busRef="Right Channel::BusType_17_1BusInterface"/>
    </ipxact:interconnection>
    <ipxact:interconnection>
      <ipxact:name>MON Unit RighttoRight Channel</ipxact:name>
      <ipxact:activeInterface busRef="MON Unit Right::CAN_1BusInterface" componentRef="MON Unit Right"/>
      <ipxact:hierInterface busRef="Right Channel::CAN_1BusInterface"/>
    </ipxact:interconnection>
    <ipxact:interconnection>
      <ipxact:name>MON Unit RighttoRight Channel</ipxact:name>
      <ipxact:activeInterface busRef="MON Unit Right::Ethernet_2BusInterface" componentRef="MON Unit Right"/>
      <ipxact:hierInterface busRef="Right Channel::Ethernet_2BusInterface"/>
    </ipxact:interconnection>
  </ipxact:interconnections>
</ipxact:design>
```

Figure 8 – Right channel component's design in IP-XACT

REFERENCES

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

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

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

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

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

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

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

[8] 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.

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

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