An Efficient Design Methodology for Hybrid Avionic Test Systems

George Afonso
EADS Innovation Works
george.afonso@inria.fr

Rabie Ben Atitallah
INRIA Lille-Nord Europe
rabie.ben-atitallah@inria.fr

Nicolas Belanger
Eurocopter Group
nicolas.belanger@eurocopter.com

Martial Rubio
Eurocopter Group
martial.rubio@eurocopter.com

Jean-Luc Dekeyser
INRIA Lille-Nord Europe
jean-luc.dekeyser@lifl.fr

Abstract

Due to the growing computation rates of avionic test systems, using hybrid CPU/FPGA architectures becomes an interesting solution to meet the performance goals. To overcome the development complexity of these systems, using system level design tools is considered a vital premise. At this level, extremely challenging requirements are needed such as the appropriate programming model and the rapid system prototyping. Focusing these issues, this paper proposes first the usage of heterogeneous architectures to design innovative avionic test systems. Second, a Model-Driven Engineering (MDE) approach is explored to be used as an efficient design methodology for the target systems. Within this approach, the compilation is defined as a sequence of small and maintainable transformations, that allows to move gradually from a high-level description into models closer to the final implementation.

1. Introduction

For the past 20 years, *Test Systems* have always be considered as a *must do* in the avionic development cycle. In early 2008, the Eurocopter research department has undertaken a profound reflection on the vocation *Pro-Active Test Systems*. Hitherto, the test systems were based on real time specific CPU boards that run proprietary real time operating systems and plugged with Input/Output (I/O) boards to communicate with the equipments under test. In current industrial practice, the well-spread VME CPU boards are widely used. Due to the present test system performance requirement, an increase in the computation rates is needed, but it cannot be delivered by the VME CPU boards any-more. To overcome these drawbacks, the usage of multicore hosts allows an immediate increase in the capacity of computation. An important outcome of this transition is the refusal of the obsolete CPU boards. However, this solution cannot guarantee the real-time criteria while the execution of concurrent tasks due to the lack of an appropriate Operating System (OS) environment. In addition, this solution brings new communication latencies between the CPUs and I/O boards plugged in the VME backplane. Eurocopter has selected the PEV1100 VME Bridge solution [2, 3] from the Swiss company IOxOS. The PEV1100 allows a local host to interface with a VME64x bus using a PCI Express External cable which offers transparent access to I/O boards. To achieve higher communication performances, IOxOS Technologies has developed a dedicated interface between the PCI express and the VME64x bus. This interface is built with the latest FPGA technology.

In this work, our proposal is to make profit from the new available hardware computing resource (FPGA) and to make up hybrid avionic test systems. Indeed, FPGA technology could offer a higher computation rates comparing to CPUs up to 10x [1]. It could implement heavy models in a hardware fashion with the management of the parallelism degree to answer the real-time constraints of the application. The main challenge of hybrid (CPU/FPGA) architectures concerns the programming model and the design methodology. We need to deal with the heterogeneity of both hardware and software parts in order to obtain a fast system prototyping. In current industrial practice, manual coding is still widely adopted in the development of hybrid architectures, which is clearly not suited to manage the complexity intrinsic in these systems. For designers, this approach is very tedious, error-prone and expensive. To overcome this challenge, we propose in this paper the usage of a *Model-Driven Engineering* [4] (MDE) approach in the specific context of hybrid system design. The objective is to build a tool that supports new design methodology and turns automatically a high level specification of the system into an executable implementation. This paper is organized as follows. After Section 1 which presents the motivations and the related works, Section 2 describes the target avionic test system. For a better development productivity, an efficient design methodology is detailed in Section 3. In order to evaluate our approach, Section 4 presents the preliminary experimental results for a typical avionic test system.

978-1-4244-6850-8/10/$26.00 ©2010 IEEE

2. The Avionic test system

Fig. 1 presents a test loop system that simulates the helicopter behavior including three models: the *Flight Mechanic Model* (noted FM), the *Navigation Model*, and the *Automatic Pilot Model*. In the initialization phase, the FM model takes several parameters such as the initial position relative to the ground and the aircraft configuration file. As a main result, this model gives back an *equilibrium position*. In addition, it sends the *common data area* structure containing the position and the speed of the aircraft to the navigation model. This later computes the helicopter destination and sends it to the automatic pilot model via the *ordered roll* structure. Finally, the flight control is managed by the *automatic pilot*. Thanks to the current multicore architectures, the computing capacity is becoming important, nevertheless they cannot guarantee the performance of tasks in a real-time. This is especially important because a task overflow could yield to the test cycle failure.

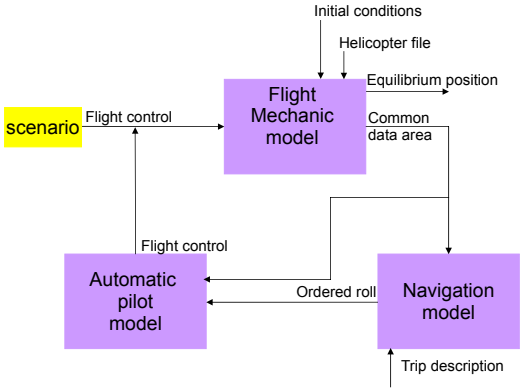

Figure 1: The simplified test loop system

The emergence and the maturity of FPGA circuits for distributed and reconfigurable architectures offer the opportunity to explore real time problems in the field of avionic systems. As of now, the FPGA is widely used in the field of I/O component in order to connect the real equipment with the CPU host. Among the main features mapped into the FPGA in the original architecture, we quote the fast serial link and RAM IPs (Intellectual property) which are needed to ensure communication between CPU and FPGA. This minimal configuration based on FPGA can be duplicated several times and connected together to build bigger test system or a complete simulator. Eurocopter expectation for the above described architecture is to prototype some models which can be eligible and relocated in the FPGA. The objective is to increase the performances of these models and to reduce the communication latencies by the means of embedding the different parts in the same chip. To do so, we need first to profile our avionic test loop in order to extract the complex models that will be implemented in the FPGA. Second, different hardware model configurations will be explored to reach an optimal well-balanced global system.

3. A Model-based design methodology

In order to deal with the design complexity of hybrid system, a new approach stretching from a high level specification to an efficient system implementation is needed. Our work is based on *Model Driven Engineering* [4] (MDE). MDE is a promising approach of software development where the whole design process is centred around three concepts: *model*, *metamodel*, and *model transformation*. A *model* is an abstraction of a system in which non-relevant details are hidden. It contains a set of concepts and relations in order to represent the system. A *metamodel* defines the available concepts and relations that can be used to create a model. It could be compared to the grammar of a programming language. As MDE promotes *all is model*, a system is built by developing a set of models at different levels in the design flow. Moving from an abstract model to a more detailed model is performed through a *transformation*. It could be seen as a *compilation process*. Applying successive transformations leads to *compilation chain*. MDE has also the option to make models executable. Indeed, a compilation chain often leads to *code*, the last transformation is thus a model to text corresponding to the code generation. More concretely, we propose the usage of the MARTE [1] standard UML profile to model the hybrid avionic test system.

3.1 Hybrid system modeling

At a high abstraction level, we use the MARTE profile to represent an hybrid system. In the MARTE specification, an application is a set of tasks connected through ports. Tasks are considered as mathematical functions reading data from their input ports and writing data on their output ports. Fig. 2 illustrates the modeling of the avionic test loop described in section 2. Its specification is a simple directed cyclic graph. In addition, MARTE allows to describe the hardware architecture in a structural way. Typical components such as HwProcessor, HwFPGA and HwRAM can be specified with their non-functional properties. Fig. 2 shows an example of an hybrid multiprocessor architecture. The main component of this architecture is composed of the Xeon-X3370 processor (multicore CPU) and the Virtex-4 Xilinx FPGA. Furthermore, MARTE provides the Allocate concept as well as the concept specially crafted for repetitive structures Distribute. This latter concept gives a way to express regular distribution of tasks onto a set of processors or FPGA resources. Fig. 2 illustrates two types of distribution (*timeScheduling* and *spatialDistibution*) depending on the target hardware platform. The different models of our avionic test loop can be mapped onto the host multicore processor, the embedded processor or the hardware resources in the FPGA.

[1] http://www.omgmarte.org

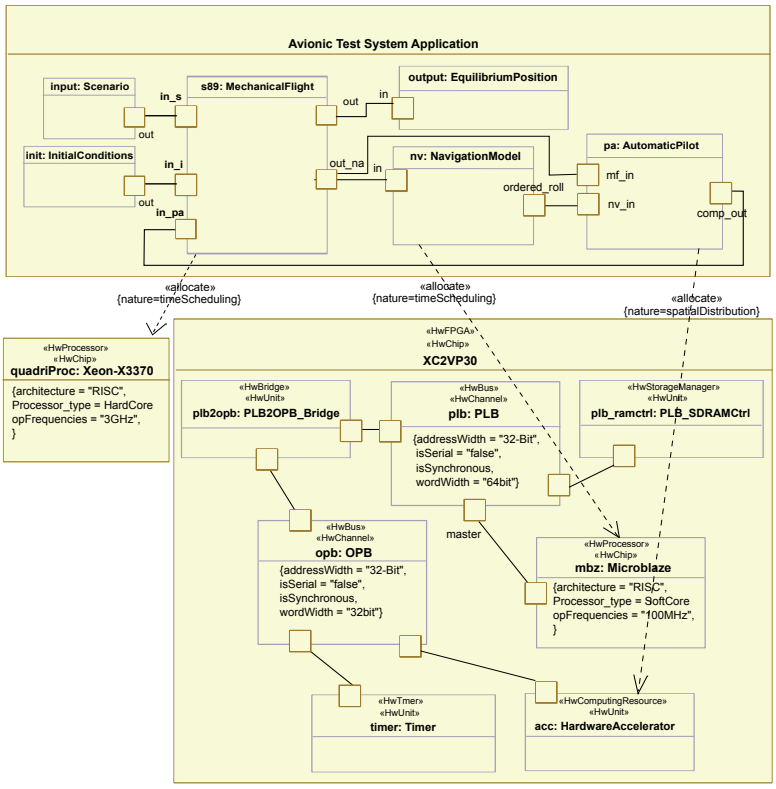

Figure 2: Hybrid system modeling with MARTE

3.2 Deployment

To transform the high abstraction level models into an implementation code, very detailed deployment information must be provided. In particular, each elementary component must be linked to an existing code. For this purpose, a *deployment* profile is introduced. A key point in our methodology is to favor the reuse of Intellectual Property block (IP). In this profile, we introduce the concept of *virtualIP* which expresses one functionality, independently of the target implementation (CPU or FPGA). It contains one or several *softwareIP* or *hardwareIP*, each one could be used to define a specific implementation at a given programming language (VHDL or C). Fig. 3 shows the *virtualIP* of the `virtualAutomaticPilot` which contains two *softwareIP* written first in C language for a software execution on CPU and secondly described in VHDL language for an FPGA implementation. Using the *ImplementedBy* dependency, the designer can select the adequate IP for each hardware and software component.

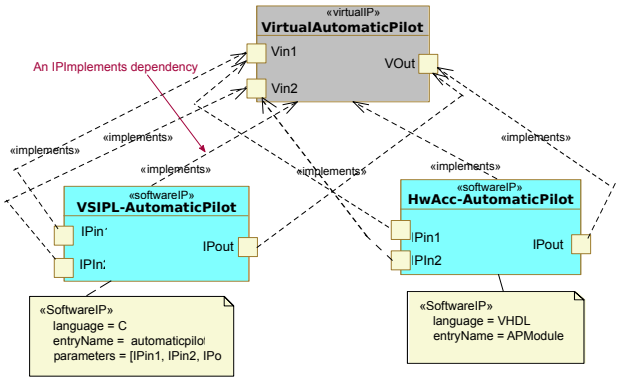

Figure 3: Example of the automatic pilot task deployment

4. Case study

In the case study, the proposed design methodology will be applied to our test loop system. For the system requirement, a time constraint of 20 ms is defined. The first step consists to profile the application using a software tool on the host machine in order to estimate the execution time consumed by each model. As a main result, the flight mechanic model takes up 90% of the processor utilization while other models occupies only 10% of the total execution time. After profiling and identifying the performance bottlenecks of the application, we have explored different hardware configurations to implement the flight mechanic model in the FPGA.

4.1 Implementation results

We considered the ML403 Virtex-4 Xilinx board as the hardware platform in which the model is implemented. In our baseline system (configuration A), all the functional blocks of the model are executed on a single Micoblaze processor running at 100 MHz. The processor communicates with the local BRAM memory (Block RAM), where the applications local data and instructions are stored on-

chip. Both instruction and data caches are also used. In the second configuration (noted B), CORDIC hardware accelerators are used with the Microblaze to improve the execution time of trigonometric functions intrinsic in the model. In the third configuration (noted C), we used the SDRAM external memory for instruction and data storage. The objective of this study is to obtain several configurations characterized with different tradeoffs in terms of performance and resource occupation in the FPGA. According to the implementation results, several remarks can be drawn. First, configuration A that runs the FM model as a software on the Microblaze has the highest execution time (30 ms). With respect to the performance constraint, the execution time of FM model is unorthodox. Thus, configuration A cannot be adopted. Configuration B that uses dedicated hardware accelerators increases the system performance up to 30x allowing a 0.9 ms execution time. However, it consumes a lot of resources: 100% of the available BRAM and 86% of the total number of slices. Configuration C offers a good tradeoff between execution time (0.94 ms) and area utilization. Indeed, this configuration requires only 13% of the BRAM and 81% of the total number of slices. In a common designer strategy, the choice of a given configuration will depend on several parameters such as the number of models and the test system constraints.

4.2 Toward automatic code generation

The above hybrid system specification using MARTE is considered the model with the highest abstraction level. The top part of Fig. 4 represents the various parts of the model. Starting from this high level, we propose to use MDE intermediate transformations until reaching the code corresponding to the target hybrid platform. By following a methodology which separates the compilation into a sequence of small transformations it is possible to manage the complexity of the tool and to maximize the reuse of transformations while compiling toward different target platforms. After the deployment step, we have defined the *Address computing* model. It allows to add information linked with the communication mechanism between the software model mapped on the host CPU and hardware model mapped on the FPGA. The second intermediate *Scheduling* model defines the ordering of task execution on the host CPUs. The need of a dynamic scheduling comes from the fact that due to the lack of details in the system communication part. Thus, the exact behaviour of the hardware cannot be predicted precisely enough to forecast the exact duration of each task. Due to the data dependencies between the tasks, the *Synchronization* model defines the appropriate scheme according to the task mapping information. We distinguish three synchronization schemes: software-software, hardware-software, and hardware-hardware. Starting from this level, we separate the system description onto two models according to the target platform (CPU or FPGA), hence the *RTL* and *Functional* models are obtained. From these models, the code generation phase is started. For each part, this step consists mainly of creating the hierarchical structure of the system, instantiating the IP blocks from the appropriate hardware libraries and connecting components together.

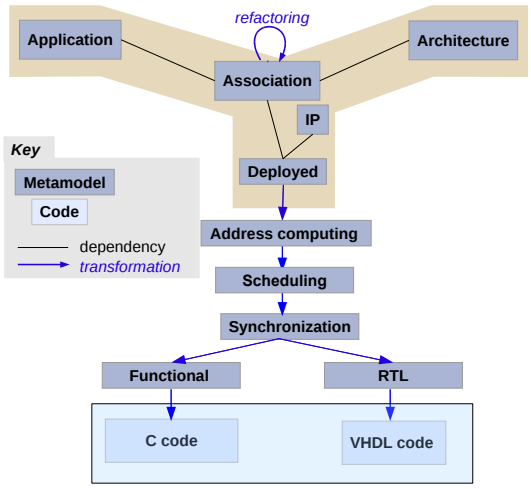

Figure 4: Compilation chain toward a hybrid system

5. Conclusion

In this paper, we have emphasized first the benefits of using hybrid CPU/FPGA architectures in the particular context of avionic test systems. Indeed, FPGAs bring performance and reliability to the avionic development cycle. Second, an MDE approach is proposed in order to deal with the design complexity of hybrid systems which are designed at a high abstraction level with the MARTE standard UML profile. MARTE models are refined toward low level implementation. In our work, we target code generation for heterogeneous CPU/FPGA architecture. To reach our objective, a compilation chain has been defined consisting of several model-to-model transformations. Eurocopter intends to adopt the MDE methodology for the next test benches generation.

References

[1] S. Asano, T. Maruyama, and Y. Yamaguchi. Performance Comparison of FPGA, GPU AND CPU in Image Processing. In *19th IEEE International Conference on Field Programmable Logic and Applications, FPL*, Prague, Czech Republic, Aug. 2009.

[2] N. Belanger, J. Bovier, J.-F. Gilot, J.-P. Lebailly, and M. Rubio. Multi-Core computers and PCI Express The future of data acquisition and control systems. In *ETTC International Conference*, Toulouse, France, 2009.

[3] N. Belanger, N. Favarcq, and Y. Fusero. An open real time test system approach. In *IEEE International Confererence on Advances in System Testing and Validation Lifecycle*, Porto, Portugal, Sept. 2009.

[4] Planet MDE. Model Driven Engineering, 2009. http://planetmde.org.