FPGA-Centric Design Process for Avionic Simulation and Test

Rabie Ben Atitallah, Venkatasubramanian Viswanathan, Nicolas Belanger, Jean-Luc Dekeyser

IEEE Transactions on Aerospace and Electronic Systems 54-3 (2018) 1047-1065

FPGA-Centric Design Process for Avionic Simulation and Test

RABIE BEN ATITALLAH
University of Valenciennes, Campus du Mont Houy, Valenciennes, France

VENKATASUBRAMANIAN VISWANATHAN
NOKIA X-HAUL R&D FPGA Lab, 59300, Valenciennes, France

NICOLAS BELANGER
Airbus Helicopters, Marignane, France

JEAN-LUC DEKEYSER
University of Lille, Villeneuve-d'Ascq, France

Real-time computing systems are increasingly used in aerospace and avionic industries. In the face of power challenge, performance re-quirements and demands for higher flexibility, hardware designers are directed toward reconfigurable computing using field programmable gate arrays (FPGAs) that offer high computation rates per watt and adaptability to the application constraints. However, considering re-configurable computing in the avionic design process leads to several challenges for system developers. Indeed, such technology should be validated along the verification & validation cycle starting with sim-ulation tools, passing through the test benches and finishing with the integration phase. For each step, the FPGA can play an essential role to achieve better performances, more adaptive systems, and cost-effective solutions. In this paper, we present a seamless FPGA-centric design process for avionic equipments. Along this process, we rede-fine the role of the FPGA circuits to cover the simulation, the test, and the integration steps. First, reconfigurable logics are used in the frame of heterogeneous CPU/FPGA computing in order to obtain high speed-up for real-time avionic simulation. The proposed environment supports dynamic execution model enabling reconfiguration during runtime to avoid the timing constraint violation. Second, the FPGA is used as a key solution to offer versatile test benches and to converge toward unified test and simulation tools. We have designed several commercial input output intellectual property systems with dynamic runtime reconfiguration capabilities, in order to mitigate component obsolescence and to provide increased flexibility and decreased design time. Third, at the integration phase, we use the conventional tools to make profit from reconfigurable technology in embedded avionic applications in order to deliver high computation rates and to adapt their functioning mode to provide reliability, fault tolerance, deterministic timing guarantees, and energy efficiency.

This work was supported by the ANRT (French National Association of Research and Technology) through CIFRE research grants in collaboration with Airbus Group and Nolam Embedded Systems.

Authors' addresses: R. B. Atitallah is with the University of Valenciennes, Campus du Mont Houy, Valenciennes 59300, France; V. Viswanathan is with the NOKIA X-HAUL R&D FPGA Lab; N. Belanger is with Air-bus Helicopters, Marignane, France; J.-L. Dekeyser is with the Univer-sity of Lille, Villeneuve-d'Ascq 59650, France. Corresponding author: Rabie Ben Atitallah, E-mail: (Rabie.BenAtitallah@univ-valenciennes.fr; v.venkatasubramanian@gmail.com; nicolas.belanger@airbus.com; jean-luc.dekeyser@univ-lille1.fr).

I. INTRODUCTION

Continuously growing aerospace industry competitiveness pushes avionic actors to revisit and strengthen their methodology and tools of the verification & validation (V&V) design cycle. In this perspective, the technical areas of simulation, test, and integration systems are currently in an unavoidable convergence path. For a long time, these different fields (simulation, test, and integration) were relying on different teams and tools, which is time consuming and error prone while switching between the design steps of the V&V process. Today, it is mandatory to converge toward common frameworks supported by cutting-edge hardware architectures.

As examples of target avionic systems, we quote control, collision avoidance, pilot assistance, target tracking, navigation and communications, amongst other functions. According to the characteristics of these functionalities, high computation rates should be achieved while performing intensive signal processing. Furthermore, these embedded systems often operate in uncertain environments. They should adapt their functioning mode to provide reliability, fault tolerance, deterministic timing guarantees, and energy efficiency. Undoubtedly, the essential feature of systems to reconfigure themselves (at the hardware or the software level) at runtime comes with additional complexity in the different design cycle steps. In the present industrial practices, different simulation tools and test benches are used for the verification of embedded avionic equipments (automatic pilot, guidance, etc.) dedicated to various helicopter ranges. This methodology calls for separate teams with different domain experts in order to achieve the simulation, the test and the integration of each part [1]. Today, this process is very complex and expensive to perform. Actually, there is an essential need of a seamless process that could help designers during the V&V cycle starting from a full software simulation to the integration phase.

In parallel, field programmable gate arrays (FPGAs) reconfigurable circuits have emerged as a privileged target platform to implement intensive signal processing applications. FPGAs offer inexpensive and fast programmable hardware on some of the most advanced fabrication processes. FPGA technology can embed parallel hardware components or several intellectual property (IP) due to the large number of programmable logic fabrics available on the chip. Such architectures can be customized at runtime using the dynamic partial reconfiguration (DPR) feature, a reconfiguration that can be done for all or for a subset of the IPs. The International Technology Roadmap for Semicon-

DOI : 10.1109/TAES.2017.2733858

ductors and high-performance and embedded architecture and compilation roadmaps promote the idea that adaptive architectures will dominate next-generation embedded systems, including those based on FPGAs. We are in line with this vision, indeed this new hardware paradigm opens many opportunities for research in aerospace and avionic industries since there are no standard process to take into consideration the FPGA as an essential part of the design process starting from a full simulation to the integration phase.

In this perspective, it is necessary to consolidate the usual design cycle with a solid system approach allowing to early check/validate the adequacy and the consistency of the reconfigurable technology to be embedded in the aircraft. Such a system approach can be conceived only if the means of V&V are present upstream. In another meaning there is a seamless process to simulate as soon as possible the solution relying on reconfigurable technology and after that testing on benches before embedding in the real world.

We extend and improve upon our previous works in order to overcome the above-mentioned challenges, and provide a complete framework for a reconfigurable avionic system. In [2], we propose a generic test environment that can adapt easily to the helicopter range and the unit-under-test (UUT). In [3], we present a prototyping environment for heterogeneous CPU/FPGA systems. In [4], we present a runtime reconfigurable and modular architecture using input output (I/O) IP cores, used in avionic applications. In this paper, we propose a complete FPGA-centric design process dedicated to avionic systems that calls for the convergence between simulation and test (S&T) domains and gives a possible solution to unify the development environment with a reduced cost and time-to-market. This objective is reached by relying on reconfigurable technology in order to perform faster real-time simulation, to reduce the time while switching between S&T, and to make the used hardware architecture more flexible. The proposed design process considers the following steps.

1) First, for the simulation phase, we propose the usage of the reconfigurable technology in the frame of generic and heterogeneous CPU/FPGA architecture that could implement intimately coupled hardware and software tasks. Relying on this architecture, a real-time simulation environment is developed. It supports a dynamic execution model to avoid the timing constraint violation during the simulation. In addition, this environment allows a context switch from a software node to a hardware node and vice versa at runtime and without a full simulation restart which reduces the verification time.
2) Second, for the test phase, the FPGA can host the avionic functionality as far as the communication protocol (ARINC429, MIL-STD-1553, etc.), which avoid the usage of specific I/O boards. We propose a modular, runtime reconfigurable, and IP-based approach for the avionic communication support. This support allows to manage dynamically different avionic communication protocols in order to consider UUTs (automatic pilot, guidance, etc.) in the test loop. Our hardware support leads to the convergence of S&T tools. Indeed, we can switch dynamically between a S&T phases in the same environment with just replacing the virtual model with the appropriate I/O protocol to communicate with the UUT using the DPR feature of the FPGA. This is another advantage that allows us to reduce the development time.
3) Third, for the integration phase, we discuss the main technological issues and industrial solutions for embedding FPGA based-avionic systems in the aircraft after the V&V from the previous phases taking into consideration different metrics such as reliability, timing constraints, power consumption, etc. The consideration of the reconfigurable part very early in the V&V design process of a new avionic equipment allows the easy integration on the final system relying on certified technologies.

This paper is organized as follows. After Section II, which presents the related works, Section III introduces the essential of S&T avionic domains and details the proposed FPGA-centric avionic design process. In Section IV, our solution of reconfigurable computing for simulation is illustrated. Section V presents an FPGA-centric solution for test systems. Technological issues and solutions for embedding avionic applications based on reconfigurable technology are enumerated in Section VI. To evaluate our approach, experimental results are presented in Section VII through several case studies. Section VIII analyzes the experimental results and offers a deep scientific discussion about the benefits of the proposed avionic design process.

II. LITERATURE REVIEW

In recent years, the feasibility of using reconfigurable hardware is being explored in the field of avionic, aerospace, and defense applications [5]–[10]. However, using FPGAs in such applications has its own challenges since time, space, power consumption, reliability, and data integrity are highly crucial factors. Some of these challenges are being addressed at the technology level, and some of them at the architectural level. One of the main challenges of using reconfigurable hardware specifically in space missions is that it has to be radiation- and fault tolerant. Single event upsets (SEUs) are induced by radiation. The environment where the avionic systems operate has unfavorable effects in these devices. Therefore, it is important to provide a fault-tolerant computing platform for such applications which are prone to radiation effects. The works done by [8] and [11] address and mitigate the effects of SEUs on FPGAs and provide a reliable computing platform. The book [12] introduces the concepts of soft errors in FPGAs, as well as the motivation for using commercial, off-the-shelf (COTS) FPGAs in mission-critical applications, such as aerospace. The authors describe a large set of soft-error mitigation techniques that can be applied in these circuits and propose methods for qualifying these circuits under radiation. Extensive work has been done in developing hardware/software codesign for an avionic communication system based on ARINC429 communication protocol [13]. Another related

work also proposes the configuration and deployment of infrastructure and related procedures of a distributed avionic communication system in FPGA [11]. Designing test systems based on modular I/O and FPGA technology provides increased flexibility and decreased cost, and helps mitigate component obsolescence [14]. Such works serves as the foundation for the usage of FPGAs in avionic applications. However, there is no a coherent design process that explicitly details the V&V of the reconfigurable hardware through the different phases: simulation, test, and integration.

Historically, tools used for avionic S&T have often been decoupled. This matter of fact could be explained by technical choices: real-time operating system (RTOS) competitiveness, hardware access, and models management capabilities. Due to the hard time-to-market requirement, practices have started to change during the last years. With the new technologies in the fields of hardware architecture and the emergence of virtualization solutions, aerospace actors are reconsidering their methodologies to verify and validate critical embedded systems. The result of this wide technological motion is the vital need to converge toward unified S&T tools.

For the past 20 years, the avionic test systems were based on real-time specific hardware architectures such as the well-spread Versa Module Europa (VME) CPU boards [15]. The VME bus is particularly efficient to allowI/O event management, multiprocessing synchronization, and a transparent access to the different hardware resources. Following the conventional industrial practices, Airbus Helicopters has integrated the VME bus as a standard backbone for the test benches of embedded helicopter systems. The proprietary test system named ARTIST is based on VME technology and the RTOS VxWorks. These technologies have been used for all helicopter benches in order to validate the avionic equipments.

Due to the present performance requirement, an increase in the computation rates is needed, but it cannot be delivered by the VME CPU boards anymore. Furthermore, this solution is considered as an expensive maintainable technology. To overcome these drawbacks, Airbus Helicopters recently decided to move to a "half generation" test system based on high-performance PC or workstation solution. Upcoming architectures are based on multicore computer plugged with I/O boards to communicate with the equipments under test. Airbus Helicopters has selected the PEV1100 VME Bridge solution [16], [17] from the Swiss company IOxOS. The PEV1100 allows a local host to interface with a VME64x bus using a peripheral component interconnect express (PCIe) external cable which offers transparent access to I/O boards. To achieve higher communication performances, IOxOS technology had developed a dedicated interface between the PCIe and the VME64x bus. This interface is built with the latest FPGA technology in order to implement PCIe end-point hardware cores.

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 OS environment. Furthermore, this solution brings new communication latencies between the CPUs and I/O boards plugged in the VME backplane.

Among existing avionic test systems provided by cutting-edge firms, we quote Aidass family [18] used in particular for Eurofighter EADS Military Air Systems, U-Test [17] developed by EADS Test&Services, and ADS2 from Techsat GmbH.[1] The proposed solutions are fully based on CPUs resources (PC or VME boards) and are close to Airbus Helicopters's solutions. These test systems can both deal with I/O management and simulation environments. Today, the management of increasing computing power relies on additional CPUs. For simulation dedicated tools, Airbus Helicopters's internal solution real-time simulation environment described in [1] does not support reconfigurable resources for virtual models management. Our perspective in this project is to consider the FPGA as an essential subpart of simulation, test, and embedded system architectures.

In [19], VanderLeest and White emphasize the usage of The Xilinx Zynq UltraScale+ MultiProcessor system on chip (MPSoC) as a promising hardware solution for avionic applications. The MPSoC provides a high-performance heterogeneous multicore processing system and FPGA in a single device with enhanced safety and security features. Combining this hardware solution with a safe and secure software supervisor satisfies the next generation of airborne computing requirements while respecting certification objectives. In addition, it has been proven that runtime reconfigurable hardware utilizes hardware resources much more efficiently. In [5], Zheng *et al.* propose a methodology for applications to be fault tolerant and sustain much longer using runtime reconfiguration capabilities. Using FPGAs for accelerating applications has shown significant performance improvements in aerospace applications [7], [8]. A general survey about dynamic adaptation in avionics systems is given in [20]. It aims at demonstrating that processing capabilities from reconfigurable computer architectures can offer high-integrity avionics systems with outstanding efficiency and effectiveness, as it is the case in other industrial domains. This study is performed considering manned and unmanned aircraft and spacecraft vehicles. In this work, we use runtime reconfiguration in the frame of avionic design process to achieve real-time simulation with better performances, to converge between S&T domains and to conceive more adaptive and reliable avionic systems.

The achievement of the proposed FPGA-based design process was performed through several development steps before reaching an acceptable maturity level. In our previous work [2], we emphasized the usage of reconfigurable technology to obtain a generic test environment that can adapt easily to the helicopter range and the UUT. In this paper, we will detail our new modular, runtime reconfigurable, and IP-based communication architecture for avionic test

[1] http://www.techsat.com

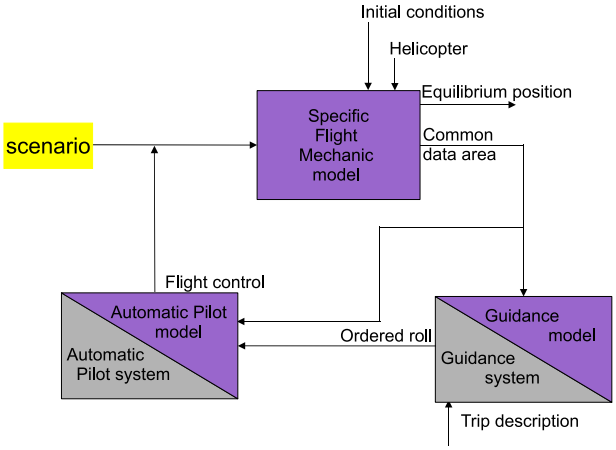

Fig. 1. Simplified simulation/test loop system.

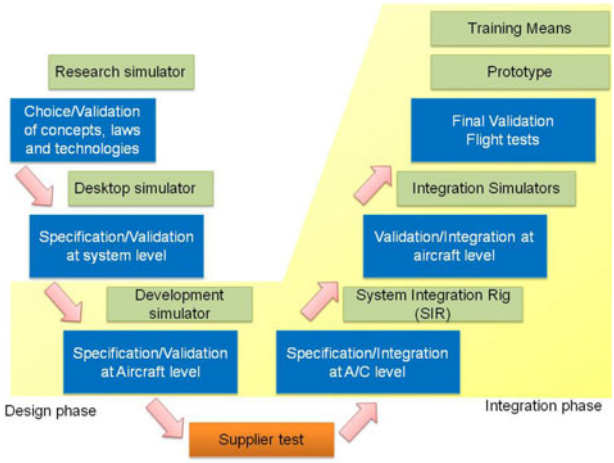

Fig. 2. New avionic equipment design cycle.

domain as will be detailed in Section V. This architecture was the objective of an international patent [21] registered in collaboration with the avionic leader Airbus Group. Our preliminary results about the advantages of heterogeneous CPU/FPGA architecture to implement avionic models are presented in [3]. In this paper, we go further into details to build an appropriate simulation environment for avionic systems based on this heterogeneous architecture that covers the execution model, the communication support, and the design methodology. In [4], we present a runtime reconfigurable and modular architecture using I/O IP cores, used in avionic applications. In this paper, we make profit from this functionality to support the convergence between the simulation and the test domains. Indeed, we can switch dynamically between a S&T phases with just replacing the virtual model with the appropriate I/O protocol to communicate with the UUT using the DPR feature of the FPGA.

III. RECONFIGURABLE-CENTRIC AVIONIC DESIGN PROCESS

A. Essential of Simulation and Test Avionic Domains

Avionic S&T domains target the validation of avionic embedded systems before the first test flight in order to increase the safety and to reduce the time-to-market. These phases are critical and have to respect constraints in order to provide the duplication of real flight conditions. In order to perform a complete simulation or test session, we need to modelize each part of the helicopter and the environmental parameters (weather conditions, geographical factors, etc.). The simulation phase relies totally on virtual models. Fig. 1 presents a simplified system simulation loop that simulates the helicopter behavior including three models: the *flight mechanic model*, the *guidance model*, and the *automatic pilot model*. In the initialization phase, the flight mechanic model takes into consideration several parameters such as the initial position relative to the ground and the aircraft configuration file to give back an *equilibrium position*. In addition, it sends the *common data area* structure containing the position and the speed of the aircraft to the guidance 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.

For the test scenario, each virtual model such as the automatic pilot can be replaced by the corresponding real avionic unit which calls for additional I/O communication hardware support. For example, the flight mechanic model can receive the *flight control* from a simulated model or from an I/O avionic interface (ARINC429, MIL-STD-1553, etc.) in the case of using a real automatic pilot system in the loop. These elements are essentials for the configuration of each test scenario depending on the UUT and the timing constraints.

In the current industrial practices, the design cycle of a new avionic equipment is following the V diagram illustrated in Fig. 2. As a first development step, the specification of the system allows a preliminary study about the hardware architecture of the new avionic equipment at different levels. To do so, different simulation (virtual) models are developed offering a first environment for the pilots to interact with the new functionality. After a virtual validation, the system specifications are transmitted for design. At the integration phase, the equipment is validated first through test benches (system integration rig) before final flight tests.

Today, different test benches are used for the verification of various helicopter ranges and UUTs (automatic pilot, guidance, etc.). Each test bench relies on a specific hardware architecture. This is due to the heterogeneity of the helicopter parts in terms of computing requirements and handled data structures. In general, several specialized CPU boards are needed to satisfy real-time constraints, which leads to sophisticated synchronization and communication schemes. In addition to this, dedicated avionic I/O boards (ARINC429, MIL-STD-1553, etc.) are required depending on the UUTs.

In conclusion, the presented design cycle calls for separate teams with different domain experts and several software tools in order to achieve each phase, hence this process is considered very complex and expensive to perform. Our objective is to converge toward a unified environment as

DOI : 10.1109/TAES.2017.2733858

shown in Fig. 2 with the yellow color. Our vision is centred around the reconfigurable technology that can play a key solution in such challenge.

B. Proposed Design Process

Facing the above challenge, we started studying the development of new design process basing on cutting-edge technology. The objective of this process is to bring reliability and competitiveness to the avionic industry. In the last quarter of 2009, we started the development phase with defining the main features and characteristics of a namely .

1) Generic: The proposed environment should be generic in order to support any helicopter range or avionic equipment. The hardware architecture should follow the generic aspect of the environment in order to support various computation nodes, avionic communication protocols, etc.
2) Scalable: The number of computing nodes or communication interfaces should be extensible according to the number of avionic systems.
3) Adaptive: When a simulation, test, or integration project is performed, the avionic models associated with a given helicopter range should be adapted according to the constraints (e.g., type, weight, size, etc.).
4) Dynamic: At runtime, we can replace an avionic model with another or a communication protocol with a second. In addition, in the same environment, we can switch between S&T phases or vice versa.

In order to satisfy the above requirements, we will rely on reconfigurable technology as an essential part of our environment for many reasons. For the first aspect, nowadays reconfigurable circuits such as FPGAs can host different computing nodes such as hard-cores, soft-cores, and hardware accelerators. Furthermore, it can be coupled with other computing nodes such as general purpose processor and interfaced with a widespread communication standards. For the scalability of the environment, FPGAs can be used to construct a network of computing nodes or parallel machines. In order to increase the productivity, FPGAs will be used in the frame of an IP-based design methodology promoting *All is IP* in order to favor the reuse and lead to more adaptive systems. In order to perform a given simulation, test, or integration project, the user needs only to select the appropriate IPs (hardware or software avionic models, I/O avionic protocols, etc.) according to the constraints. With the DPR feature, IPs can be managed at runtime to switch between different implementations and communication protocols.

These advantages of using FPGAs in the development of avionic systems are transverse to the design phases. As illustrated in Fig. 3, we redefine the role of the FPGA circuit to cover the simulation, the test, and the integration steps. In what follows we detail the design process.

1) At an early phase, we involve the reconfigurable technology in the design process for real-time simulation. The simulation phase is considered as an essential part of the industrial product manufacturing. In fact, it is required to validate the performance of complex equipments at an early phase through virtual models. For different avionic systems, specific real-time constraints should be fulfilled. This behavior has to be validated first at the simulation level before integrating the functionality into the real system. We propose the usage of FPGAs to design heterogeneous CPU/FPGA architecture that could implement intimately coupled hardware and software avionic models. The main objective is to deliver high-performance computing with real-time support. FPGA brings also dynamic reconfiguration capability to the system in order to deal with runtime model re-allocation. Furthermore, this step allows to verify the eligibility of a given model to be implemented as a cost-effective hardware solution comparing to a software implementation.
2) As a transition between the S&T phases, we propose first to use the FPGA as a bridge between virtual models and avionic equipments in the loop. At this level, reconfigurable technology is a key solution for the avionic I/O hardware obsolescence issue taking into consideration communication protocols as IPs. The huge logic budget available in nowadays FPGAs allows to use these circuits for computation as well as for communication at the same time. At this phase, there are also real-time requirements with more complexity coming from the data synchronization between virtual models and UUTs. Furthermore, we will support dynamic behavior in order to switch between a simulated model to the real equipment or to switch between different avionic protocols. The test phase enables the interaction of the new functionality with existing avionic equipment before the integration phase.
3) For the integration phase, we will rely on a standalone FPGA-based technology in order to carry out the avionic functionality. At this level, our concerns cover embedded constraint verification, fault tolerance, reliability, certification, etc.

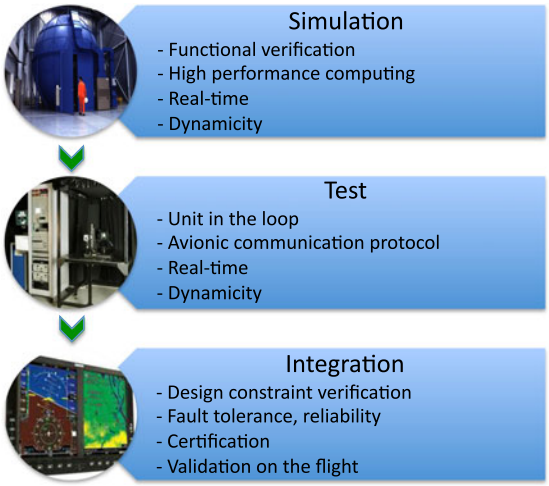

Fig. 3. Proposed design process.

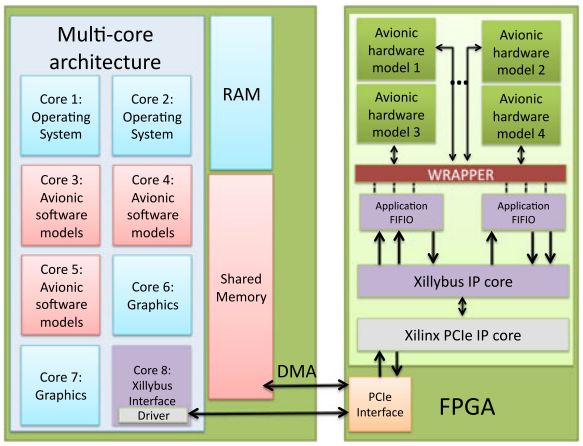

Fig. 4. Heterogeneous CPU/FPGA architecture for real-time simulation.

IV. RECONFIGURABLE COMPUTING FOR SIMULATION

As introduced is the previous section, the main goal of using reconfigurable computing for the simulation phase is achieving high computation rates with real-time capabilities. In order to meet these requirements, combination of general CPU and reconfigurable fabrics like FPGAs is necessary. In such systems, multicore processors provide high computation rates while the reconfigurable logic offers high performance per watt and adaptability to the application constraints. Designers could exploit the existing partitioning in the application (i.e., hardware-software and parallel-sequential hardware) which leads to several feasible implementations, whose performances vary with the chosen partitioning. With the management of the parallelism intrinsic in the application, FPGA technology could offer better performances comparing to CPUs or GPUs up to 10× [22] at lower frequencies. Using heterogeneous CPU/FPGA systems allows to adapt the architecture according to the application constraints and thus to optimize hardware resources. All these benefits emphasize system designers to redirect their efforts on reconfigurable computing for simulation domain.

Our expectation of 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 means of embedding the different parts in the same chip. To do so, we first need 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. Indeed, the FPGA technology could implement heavy models in a hardware fashion with the management of the parallelism degree to address the real-time constraints of the application.

A. Heterogeneous CPU/FPGA Hardware Environment

As illustrated in Fig. 4, we propose a scalable heterogeneous CPU/FPGA hardware environment composed mainly of two nodes. The first node is a general purpose multicore processor (i.e., AMD/Intel), while the second node represents an FPGA. The multicore will offer performance with a limited parallelism capability due to the fixed number of cores. FPGA is the support of the reconfigurable logics needed to implement challenging avionic models as hardware accelerators.

Within our environment, a great care has been devoted to the real-time aspect in order to satisfy tight computing and communication deadlines. In fact, nowadays operating systems (OS) such as Linux allocate dynamically tasks onto the available cores which may introduce latencies and lead to the timing constraint violation. This is due to the fact that general purpose OS do not support real-time functionalities. Processor affinity service is a modification of the native central queue scheduling algorithm in a symmetric multiprocessing OS. Each task (process or thread) in the queue has a tag containing the target processor or a core number in which it will be executed. In our architecture, we propose to allocate each kind of tasks (OS, avionic model, etc.) in the available cores under bounded soft real-time constraints. Fig. 4 shows an example of task allocation; cores 1 and 2 run the OS, core 3, 4, and 5 are dedicated to carry out the avionic models, the graphic part is mapped on cores 6 and 7, and finally the core 8 ensures the communication between the host and the FPGA module. For the reconfigurable part, several hardware models can be hosted in the FPGA while better performances are needed. Such heterogeneous CPU/FPGA architecture could implement intimately coupled hardware and software avionic models. The shared memory implemented in the software part allows data sharing between software and hardware avionic models.

As well as we need to optimize our avionic models in order to obtain better performances, we also need to focus on the communication which is crucial in heterogeneous architectures. The link has to be fast, efficient, and widely used in industrial systems. Nowadays, almost host machines or workstations are equipped with PCIe slots for expansion boards. In addition, a large range of commercial FPGAs integrate a hard endpoint PCIe core for industrial usage. Our proposal is to make profit from these features in order to design an efficient solution that can deal with the interoperability between hardware and software models mapped respectively on FPGA and CPU nodes with high throughput. In such architecture, communication latency with respect to the real-time constraints is considered the most important metric. Nevertheless, it is first necessary to define the application requirements in order to propose a customized solution that offers the better tradeoff between the communication bandwidth and the design cost.

B. Execution Model

For the execution model, each avionic model can be designed with different versions (i.e., software, hardware, etc.). A common high-level model is developed in order to encompass different functions which correspond to the different implementations. The necessary data (input, out-

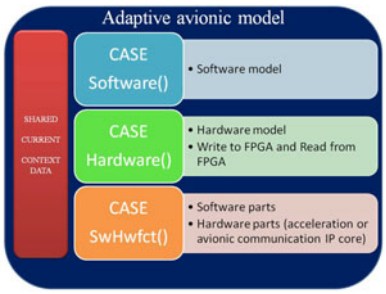

Fig. 5. Adaptive avionic model.

put, current context) is contained in a global data structure (stored in the shared memory in Fig. 4) allowing easier context switch from a software node to a hardware node and vice versa at runtime and without a full simulation restart.

Fig. 5 shows how a software/hardware (or vice versa) context switch can happen at runtime. In fact, the HwFunction() and SwFunction() share the same data context and the I/O data structure in order to perform the calculation node switching more efficiently. Let us highlight that the HwFunction() communicates with the hardware core using the Xillybus solution (will be detailed in the next section), and thus bringing a total transparency for the system. Our solution avoids additional timing cost for the software–hardware context switch. As a reconfiguration scenario, an anticipated overload alert can be generated for an avionic model reallocation in order to avoid the violation of timing constraints and thus the failure of the simulation phase. In our environment, the decision of the initial mapping is taken by an exact method while runtime allocation is ensured by a heuristic [23] depending on the simulation scenario requirements. Our runtime mapping heuristic is developed to deal with the model overloads and to make a decision about the dynamic context switch according to the available software or hardware implementations.

C. Xillybus: Making FPGAs Talk PCIe Easier

Due to the PCIe bus complexity, the communication in a heterogeneous architecture remains complex. Most of the time, all PCIe capabilities are not even required (i.e., prototyping), an abstracted communication level would improve the design cycle. Xillybus proposes a simple interface for the FPGA and the application designer: The FPGA application logic connects to the IP core through standard FIFOs (for read and write), and the user application on the host machine (Microsoft Windows or Linux) performs plain file I/O operations. Streaming data move naturally between the FIFO and the file handler opened by the host application. There is no specific and intrusive air position indicator involved, allowing the hardware and software designers to focus on the requirements of their application. This setting relieves the FPGA designer completely from managing the data traffic with the host. Rather, the Xillybus core checks the FIFOs empty or full signals (depending on data direction), and initiates data transfers when the FIFO is ready for it. As the number of streams and their attributes are configurable, this solution scales easily as the design requirements

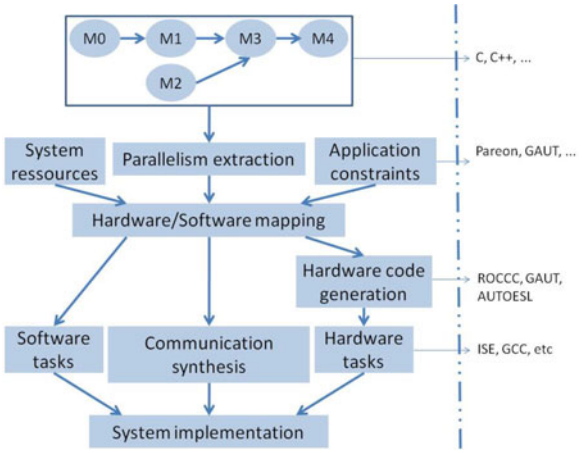

Fig. 6. Application design methodology for heterogeneous CPU/FPGA system.

expand. Fig. 4 depicts a simplified block diagram showing the connection of one data stream in each direction. The application on the computer interacts with device files that behave like named pipes. The Xillybus IP core and driver on the host offer efficient data streaming (using DMA) between the FIFOs in the FPGAs and their respective device files on the host. The Xillybus IP core implements the data flow utilizing PCIe transport layer level, generating and receiving transaction layer packets. For the lower layers, it relies on Xilinx official PCIe core, which is part of the development tools. Making the communication simple is sometimes not enough, the goal is to find the best tradeoff between simplicity, reliability, design time, and performance in order to address all requirements of our application.

D. Design Methodology

The above-described architecture is attractive for heterogeneous system prototyping and performance evaluation, however we need tools to help software designers to map application on such system. In current industrial practice, manual coding is still widely adopted in the development of heterogeneous 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 present a design methodology that covers the different development steps from software specification to the system implementation as shown in Fig. 6. First, we are considering a software application presented as a task graph containing different communicating models (M0, M1, etc.). All applications are not adequate to be implemented onto heterogeneous CPU/FPGA architectures; a complete analysis of the source code is needed to verify if a hardware implementation could bring better performances. In order to leverage the parallelism of the multicore CPU/FPGA architecture, tools such as Vector Fabrics Pareon [24] and GAUT [25] can find all data dependences by analyzing the C or C++ source code and extract the parallelism intrinsic in the application. Pareon analyzes partitions and maps applications on specific platforms as heterogeneous ones. It can

also estimate the performance of the parallelized software before implementing it. Moreover, it can trim any overhead in your hardware to reduce cost and ensure that all critical behaviors in your program are exercised. After this analysis, the developer will have key information for source code optimization. To perform the application mapping, system resources and application constraints are needed. This step requires a specific heuristic method to resolve a multiobjective exploration problem. After the mapping step, we need to develop some user hardware applications from the existing models (M1, M2, etc.). To make this step more efficient, tools such as riverside optimizing compiler for configurable computing (ROCCC) [26] can focus on FPGA-based code acceleration from a subset of the C language. ROCCC does not focus on the generation of arbitrary hardware circuits. Its objectives are to maximize parallelism within the constraints of the target device, optimize clock cycle time by efficient pipelining, and minimize the area utilized. It uses extensive and unique loop analysis techniques to increase the reuse of data fetched from off-chip memory. The communication synthesis step consists of generating the required CPU-to-CPU or CPU-to-FPGA communication interfaces depending on the selected mapping. Having all source code for a CPU/FPGA implementation, the compilation step, using GCC and ISE from Xilinx can be easily performed in order to map all the application onto the system.

V. RECONFIGURABLE COMPUTING FOR TEST

As stated in Section III-A, the test avionic domain calls for an additional hardware in order to communicate with avionic equipments. In current industrial practices, one of the biggest challenges of relying on different printed circuit boards (PCBs) for different requirements, is the hardware obsolescence issue. Ever-changing application requirements demand the customization of the I/O bus interfaces. Changing the hardware meant redesigning the entire board, with a lot of non-recurring engineering cost and significant time-to-market. The VMEbus International Trade Association group FPGA Mezzanine Card (FMC) standard solves the I/O obsolescence issue partially, with a single 400-pin connector with a potential overall bandwidth of 40 Gb/s. This essentially means that the I/O bus interface of a PCB is designed separately as a module and interfaced with the board using the FMC connector. Thus, every time an I/O bus interface needs a change, just the module changes, thereby avoiding a complete redesign. For the test phase, the FPGAs can be used for more than just computational purpose in order to improve the system performance. The introduction of FMC I/O standard has given a new purpose for FPGAs to be used as a communication platform. Taking into account the features offered by FPGAs and FMCs, such as flexibility and modularity, we have redefined the role of these devices to be used as a generic communication and computation-centric platform. Thus, in addition to the avionic models, FPGAs will implement I/O IPs such as ARINC429 in order to perform the communication with the UUT.

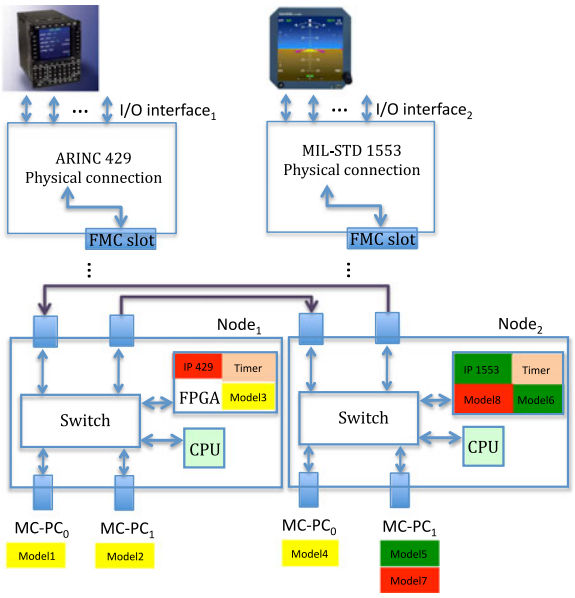

Fig. 7. Hardware architecture for test system.

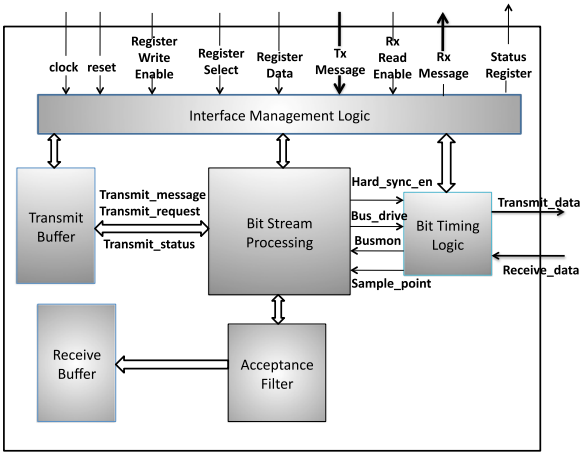

Fig. 8. CAN bus controller architecture.

A new modular, runtime reconfigurable, IP-based communication-centric hardware is proposed for avionic test application domain as illustrated in Fig. 7. The hardware architecture is composed of standard machines running virtual avionic models coupled with FPGA boards equipped with FMC connectors. These connectors ensure through the I/O interface just the physical connection with the avionic equipment. The communication protocol is implemented as an IP hosted on the FPGA and data are transmitted via the FMC. Thus, the test phase for a given equipment requires the instantiation of the appropriate I/O IP protocol, while the other avionic models remain virtual. Some models can be also hosted on the FPGA as the same level of the I/O IPs, which can reduce significantly the communication delays. We can rely on several FPGA boards in order to consider several avionic UUT on the loop.

With such architecture, avionic IP cores can be explored by system designers in different scenarios depending on the

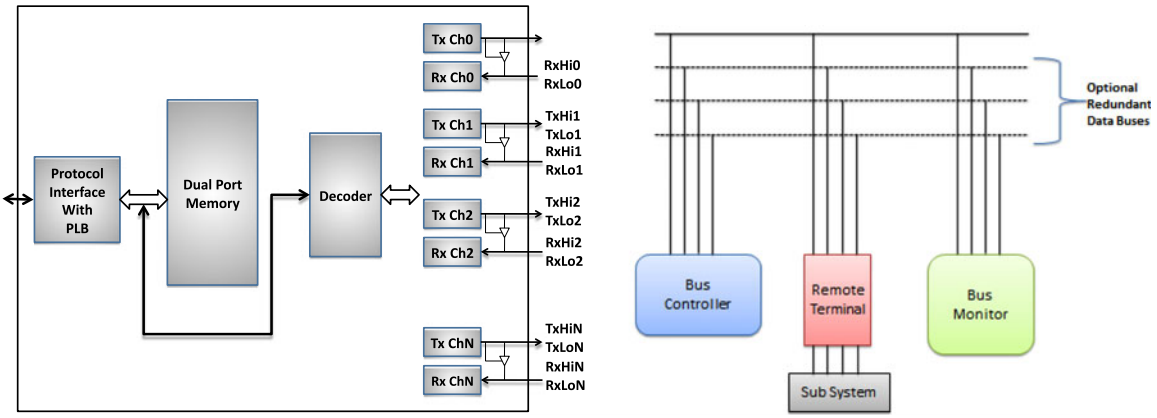

Fig. 9. ARINC429 bus architecture (left) and the MIL-STD-1553 bus architecture (right).

application requirement. Therefore, depending on the test scenario, the user can choose a communication IP core to be configured dynamically. This removes the need to have multiple/redundant systems, each for a different protocol. Moreover, when the IP core is reconfigured, the communication channels with the FMC is also reconfigured dynamically along with the protocol. In case the FMC module does not provide a corresponding interface for the communication core being reconfigured (which can be detected using the I^2C EEPROM in the FMC module), it can be swapped with another appropriate FMC module. Thus, eliminating the need to redesign the entire board based on a new I/O interface requirement.

In the next sections, we will detail the implementation of three widely used avionic I/O communication protocols: ARINC429, CAN Bus, and the MIL-STD-1553. These protocols are designed in the frame of an IP-based approach for the test phase.

A. Examples of Avionic Communication Protocols

1) *ARINC429 and CAN BUS:* ARINC429 is an application-specific technical standard for the avionic data bus used on most higher end commercial and transport aircrafts. It defines electrical characteristics, word structures, and protocol necessary to establish an avionic bus communication. For ARINC429, messages are transmitted at a bit rate of either 12.5 or 100 Kbps per second to other subsystems. The design supports up to 16 Transmit and 16 Receive channels, although the protocol permits upto 20 channels. The architecture of ARINC429 is shown in Fig. 9 (left). The CAN controller implements the Data Link Layer as defined in the document "BOSCH CAN Specification 2.0."[2] It implements a serial communication which efficiently supports distributed real-time control with a very high level of security. The design has two communication channels. The CAN bus protocol supports upto 16 channels, and has a maximum bandwidth of upto 1 Mbps. The transmission can be programmed to a random frequency using the configuration registers. The architecture of the CAN controller is shown in Fig. 8. Furthermore, since both the IP cores have been designed using generic IEEE 1076-1987 standard, they can be implemented on any FPGA device family and architecture.

2) *MIL-STD-1553:* The MIL-STD-1553 is originally serial military standard protocol that defines the mechanical, electrical, and functional characteristics of a serial data bus. It is now also being used in spacecraft on-board data handling subsystems, both military and civil. The architecture of the bus system consists of a bus controller (BC) controlling multiple remote terminals (RT) all connected together by a data bus providing a single data path between the BC and all the associated RT. The RT is used to interface with other user defined subsystems. There can also be one or more bus monitors; however, they are not allowed to do any data transfers, and are only used for recording the data for analysis. The protocol also supports several data buses to provide multiple redundant data paths upto a maximum of 4. The protocol follows very strict timing constraints and requirements and provides a maximum bandwidth of 1 Mbps. We have developed our own IP core according to the MIL-STD-1553 specification[3] and used it to evaluate our system. The architecture of the MIL-STD-1553 is given in Fig. 9 (right).

B. Toward the Convergence Between the Simulation and the Test Domains

Using the FPGA as a centric computation component for simulation as well as a communication centric component for test leads to the convergence toward a unified environment for S&T. We promote that *all is IP* (avionic models and I/O communication protocols) in our environment. For a given simulation or test project, we have to instantiate the appropriate IPs at the initial phase. In different scenarios, these IPs can be managed at runtime. For instance, a software avionic model can be replaced with a hardware implementation when more performance should be delivered. We can also switch dynamically between the S&T phases with just replacing the virtual model with the

[2]http://esd.cs.ucr.edu/webres/can20.pdf

[3]http://mil-std-1553.org/

DOI : 10.1109/TAES.2017.2733858

appropriate I/O protocol to communicate with the UUT using the DPR feature of the FPGA. Considering the number of avionic systems (from 10 to 100) that can be embedded depending on the helicopter range, several computing nodes are necessary. Hence, a unified environment for S&T can be a network of heterogeneous CPU/FPGA nodes with different I/O interfaces. A supervisor is needed to manage all the available resources at runtime. As all is considered as IP, a new simulation or test project is assimilated to resource allocation problem. We need also to deal with the communication and the reconfiguration models in order to meet real-time constraint and dynamic reallocation. These objectives are considered as future works, and we will focus only on one computing node as it will be illustrated in Section VII.

VI. TOWARD RECONFIGURABLE COMPUTING FOR EMBEDDED AVIONIC APPLICATIONS

After the validation of the avionic system through the simulation and the test phases, we need to integrate the FPGA-based solution on the aircraft as a standalone hardware gathering the computation and the communication parts. At this level, we meet the conventional methods and industrial tools used for fault tolerance, verification, and certification in order to address special requirements that demand powerful and highly reliable designs. Significant research and industrial efforts have been devoted at the circuit and Electronic Design Automation (EDA) levels to reach this objective. As an example, FPGA device manufacturers are collaborating with EDA tool vendors to resolve difficult problems like providing triple redundancy for dealing with SEUs issues in avionic applications.

Today, Xilinx offers on the 7 series FPGAs automatic detect and correct circuitry (CRC/ECC) with partial reconfiguration (PR). This technique scans and corrects 2-bit upsets in 20–30 ms for most devices and enables SEU logging and tracking. CRC/ECC operates independently of user design. With the Mentor Graphics (an EDA technology leader) and Xilinx collaboration, the tool Precision Hi-Rel synthesis software is provided. In addition, Xilinx offers TMRTool software for Space and other extreme reliability applications. Other FPGA vendors such as Actel and Altera are also providing their commercial solutions.

Today, triple-modular redundancy (TMR) techniques are widely used to mitigate radiation effects, but TMR incurs substantial overheads such as increased area and power requirements. In order to reduce these overheads while still providing sufficient radiation mitigation, Jacobs *et al.* in [27] propose a reconfigurable fault tolerance framework that enables system designers to dynamically adjust a system's level of redundancy and fault mitigation based on the varying radiation.

As the number and the complexity of embedded avionic systems have grown in nowadays aircraft, it became necessary for the Federal Aviation Administration to establish a baseline of minimum design flow steps for avionic equipment. DO-254 was formally recognized in 2005 as a standard for ensuring the highest level of safety in electronic avionic systems. It provides guidance for the design assurance of complex electronic hardware in airborne systems and equipment for use in aircraft or engines.[4] However, compliance with DO-254 requires assistance from the FPGA tools vendors because there are requirements to provide documentation and traceability. The tools vendors such as Synopsys have been doing a lot to provide easier and more comprehensive ways to ensure that compliance. In fact, tool assessment is a part of the DO-254 process that is meant to ensure that the tools used for hardware design and verification perform correctly.

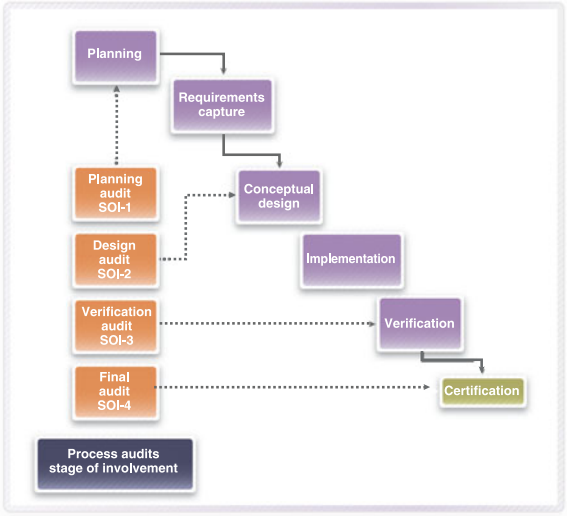

Fig. 10. DO-254 process flow from Synopsys.

A DO-254 compliant design is specified using a set of formal requirements. As part of the certification process, the applicant must prove that their implementation meets all of these requirements. A graphical illustration of the typical process flow is shown in Fig. 10.

The first step in the DO-254 process flow is the design specification using formal requirements leading to a verification plan that should be tracked along the process. The next step is the design implementation. FPGA implementation is typically verified through RTL simulation, to validate design intent, and code coverage analysis to ensure 100% coverage of all possible input signal combinations across a series of applied tests. However, while simulation results can be easily visualized, analyzed, compared, and requirements traceability easily maintained, the design behavior in real hardware cannot be easily traced back to simulation because it is simply not possible to achieve 100% specification coverage once the FPGA is physically mounted onto a circuit board.

To have full traceability you need to be able to compare the behavior of the physical outputs of the device with their corresponding RTL simulation results. However, rarely we can drive physically the hardware with all combinations of

[4]http://www.atego.com

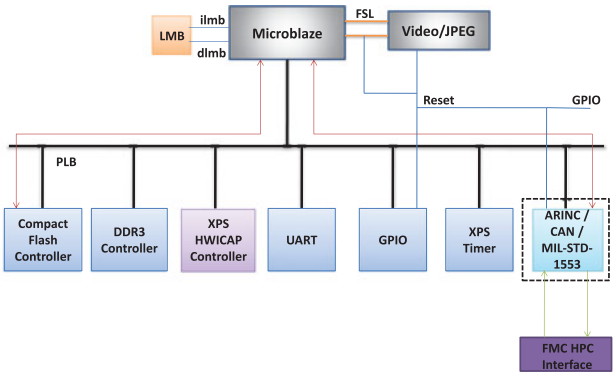

Fig. 11. Embedded hardware architecture with dynamic reconfigurable system.

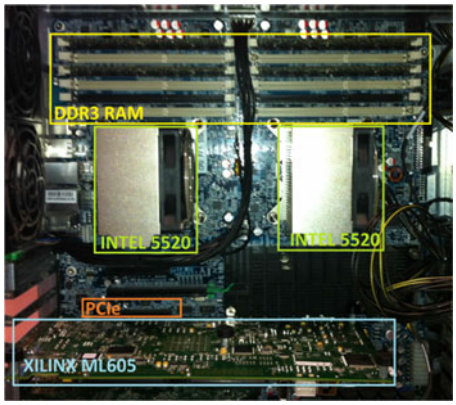

Fig. 12. Heterogeneous CPU/FPGA for real-time simulation.

stimuli. Even for the inputs, the creation of test vectors is an intensive and time-consuming manual task.

Accordingly, performing verification to satisfy DO-254 at the board-level is not only challenging and risky, it is sometimes just not feasible within project time scales, which is why engineers are increasingly adopting a so-called in-hardware verification methodology. Aldec provides a compliance tool set that enables DO-254 compliant design and implementation flow.

VII. EXPERIMENTAL RESULTS

In this section, experimental results will cover all the design steps in order to underpin the industrial relevance of the proposed FPGA-centric design process for avionic systems. First, we will demonstrate the benefits of using FPGA platforms to perform real-time simulation for avionic systems. Second, for the test phase, we will rely on the same technology in order to implement flexible I/O avionic protocols to establish the communication with the UUT (avionic equipment). Third, a real use-case scenario for related to a UAV avionic system is given. Our main objective is to demonstrate the smooth transition between the different phases and the capability to converge toward a unified environment for S&T.

A. Simulation Environment Results

To test our solution of heterogeneous CPU/FPGA hardware for real-time avionic simulation, a hardware experimental environment setup is essential. The hardware environment is based on a bi-processor Intel Xeon E5520 (quad-core) 2.27 GHz, 16-GByte DDR3 memory, and an ML605 Virtex-6 Xilinx board. The FPGA is plugged in the mother board through a PCIe slot that can support 2.5 GBytes/s throughput as illustrated in Fig. 12. Our software environment relies on Linux Debian and has been successfully tested with kernel versions ranging from 3.2.0-amd64 to 3.10.5-amd64. It is released under a proprietary software license. To satisfy soft real-time requirement, frequently imposed in industrial domain, we opted for processor affinity. In order to avoid the timing constraint violation, we have modified the standard kernel configuration in order to reduce the latencies. The CPU frequency scaling are disabled to keep the cores at their maximum frequency. We have also disabled the swap capability to be sure that no model will be "swapped." Another element that guarantees real-time requirements is the FPGA. However, its utilization can be useless because of the communication latencies with the CPU. In order to fulfill real-time requirements, it is important to find the best tradeoff.

TABLE I
Avionic Models Analysis

Results	Speed-up	Maximum number of useful threads	Synchronization overhead
Model A	1.2	2	1%
Model B	0	1	0%
Model C	2.4	3	0%
Model D	3.5	6	39%
Model E	3	4	29%

As stated before, the main objectives of using the FPGA at this level is improving performance versus software implementation, achieving high simulation speed-up, and fulfilling real-time requirements. To do so, we will analyze different avionic models in order to obtain different possible implementations on our heterogeneous multicore CPU/FPGA architecture. These models are used for an avionic simulation project such as the flight mechanic and the guidance models. According to the needed performance and the real-time constraints, the design will be tuned and improved as much as possible in order to be executed more efficiently on our architecture considering also switching at runtime between software and hardware configurations or vice versa.

Table I summarizes the experimental results obtained by analyzing the software models with Pareon tool. First, we measure the speed-up obtained after optimization. Second, threads must be created for parallel implementation strategy. This might be implemented through the use of POSIX calls creating the threads. The maximum useful number of threads is directly linked with the parallelism degree of the application. Indeed, as shown by many parallelism laws, there is always a limit number of useful calculation nodes, it is the same for the maximum of useful threads. But

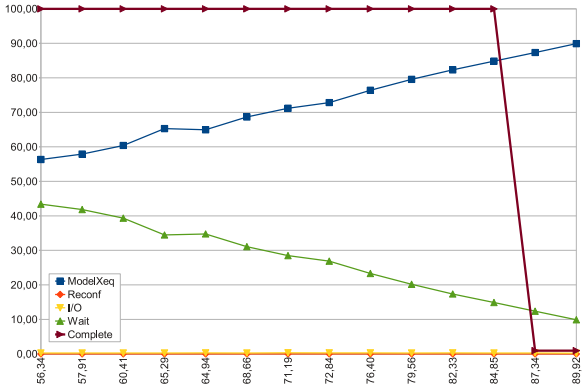

Fig. 13. Time repartition of a multimodel soft/hard simulation project.

multiple threads means more data synchronization. That means a less time delay while waiting for data and therefore less latency. Synchronization brings overhead, there is a tradeoff between the latency introduced by doing fewer synchronizations with more data and the overhead introduced by doing more synchronizations with less data. Pareon helps the user to get the best tradeoff.

Pareon shows a 1.2 speed-up for the model A with low synchronization overhead with only two threads. Model B cannot make profit from a parallelization strategy. For these two models, better performances can be achieved just with hardware implementation. Model C, D, and E offer higher parallelism degree with low synchronization overhead for model C. These models are suitable for multicore architecture or hybrid multicore CPU/FPGA implementation by splitting the models into different functions due to the low synchronization overhead.

Using our previous results, we decide to implement model A which is the *Flight Mechanic model* in order to observe the behaviour of such model in a VHDL hardware implementation. With the software version, we obtained a 20 μs of execution time with our host.

For the hardware implementation, we target the Virtex-6 Xilinx board (ML605) as execution support. A VHDL implementation of the *Flight Mechanic model* offers a 2 μs of execution time with 8% space occupation, this is mainly due to the usage of floating point calculation in this model. This result offers the opportunity to move model A from a processor to the FPGA in the case of timing constraint violation or an anticipated overload.

In the next experiment, we will consider the simulation loop presented in Fig. 1, the real-time period is set to 10 ms. The *Flight Mechanic model* is implemented in hardware and hosted on the FPGA while the other models are executed on processor. Our objective is to verify the stability of the simulator according to the global system load. While the load increases and upto certain limit, many runtime reconfigurations occur without any interruption of the simulation caused by a real-time period overflow. We highlight that the execution time of a software model increases according to the processor load which is not the case of a hardware model hosted on the FPGA.

Fig. 13 describes simulation model time repartition. The execution time of the model is proportional to the processor's load and inversely proportional to the "wait" duration. "Complete" corresponds to the percentage of successful simulation. Fig. 13 shows that the I/O time and the reconfiguration time are negligible. As soon as the load exceeds the migration threshold (87%), migrations occur. Then, as the load continues to increase the "too many consecutive migrations alert" occurs and stops the simulation which is mainly due to the software part. This result demonstrates that a heterogeneous CPU/FPGA architecture can be an efficient execution support for real-time simulation in avionic domain without referring to a dedicated and expensive solutions as discussed in Section II. As a conclusion, we highlight that the FPGA can bring performance in such domain plying mainly the role of a computing hardware accelerator.

Fig. 14. ML605 kit showing FMC connector loopback using MIL-STD-1553.

B. Test Environment Results

For the test environment, we will rely on the same technology in order to establish the communication with the UUT (avionic equipment) according to the generic architecture presented in Fig. 7. In fact, the ML605 board that we are using provides two FMC slots; one with a high pin count and the other with a low pin count. Hence, these slots can be used to host simultaneously two different FMC cards presenting I/O avionic interfaces as shown in Fig. 14. Different communication avionic protocols (ARINC249/CAN/MIL-STD-1553) are implemented and tested in real scenarios. To do so, a Microblaze processor is used in order to configure the registers and initiate data transfers. A number of frames are transmitted to an avionic subsystem using an appropriate communication protocol (ARINC429/CAN Bus/MIL-STD-1553) via an FMC interface. The communication protocol is selected and configured during runtime according to the request. Ideally, the data have to be transmitted to external subsystems. However, for testing purposes, we have done an external FMC loopback to verify if the transmission is correct as shown in Fig. 14. We then analyze the FPGA resource utilization, transmission characteristics, I/O pin requirement, and scalability for each core. The results are elaborated in the following sections [4].

1) *FPGA Resource Utilization:* Table II summarizes the area utilization of each avionic protocol with different configurations. While considering only one IP core active at a time (i.e., a design with only an ARINC429 16 channels or a CAN Bus 16 channels), the consumed area is about 37% of the logic blocks in the FPGA (with respect to flip flop

TABLE II
Area Utilization of the Avionic IP Cores

	Slices	FFs	LUTs	BRAMs
ARINC429-2 channel	1912	3198	7644	2
ARINC429-16 channel	47 920	44 267	59 400	2
CAN Bus-2 channel	689	1016	2754	0
CAN Bus-16 channel	10 176	15 437	41 096	0
MIL-STD-1553- Dual Redundant	1232	3432	8453	16

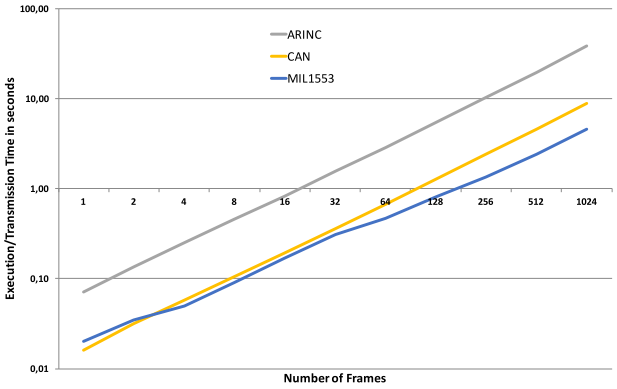

Fig. 15. Transmission time versus number of frames.

utilization). However, when three IP cores are implemented at the same time (in this scenario ARINC429 16 channels, CAN Bus 16 channels, and MIL-STD-1553), it consumes over 70% of the resources on the Virtex6 FPGA which is about 50% excess comparing to one IP core active at a time. This is a waste of hardware resources considering the fact that the FPGA can host also avionic models as discussed in Section V. Using PR is very relevant because it clearly eliminates the need for having multiple systems (corresponding to different UUTs) for the lack of hardware resources in traditional avionic systems.

2) *Transmission Times:* Fig. 15 shows the performance of our IP cores in terms of time taken for transmitting different number of frames. The system performance has been evaluated up to 1000 frames. The time measurements were done in the software using a Microblaze timer. The transmission and reception is done synchronized according to the protocols' internal clocks in preprogrammed frequencies. However, the time measured, also takes into consideration the overhead of configuring the registers, the overhead caused due to the transfer of status words, and the idle time between each transaction. From the graph shown in Fig. 15, it is seen that MIL-STD-1553 is the fastest in transmitting the frames. Although MIL-STD-1553 and CAN Bus have the same maximum bandwidth, the fact that MIL-STD-1553 is able to pack more data into a single message and to operate with minimal status feedback, gives it an extra edge in transmitting efficiently.

3) *Number of I/O Pins and Channels:* Each protocol requires a specific number of FPGA I/O pins to communicate to the external subsystems via FMC. The number of pins required is determined by the number of communication channels in the design. Table III shows the number of FPGA I/O pins required for each protocol according to the

TABLE III
FPGA I/O Pins Requirement for Each Protocol

Channel count	ARINC429	CAN Bus	MIL-STD-1553
Pins for 1 channel	8	3	6
Pins for 2 channels	16	6	12
Pins for 4 channels	32	12	24
Pins for 8 channels	64	24	–
Pins for 16 channels	128	48	–
Pins for 20 channels	160	–	–

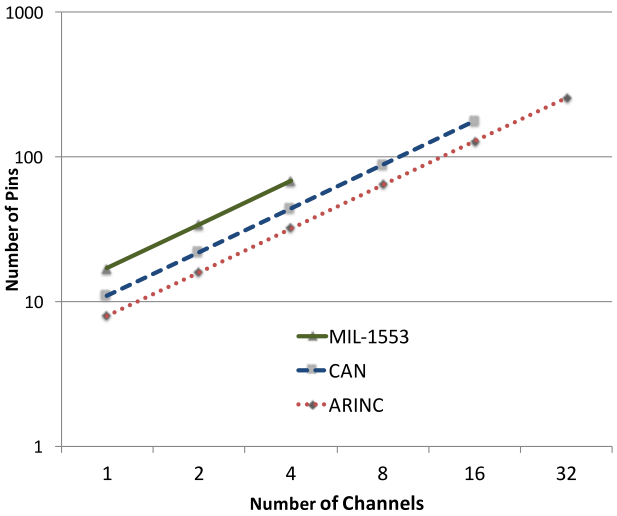

Fig. 16. Number of pins versus number of I/O channels.

number of I/O channels. However, some data in the table are not shown because CAN bus standard does not support more than 16 channels and MIL-STD-1553 is never used beyond 4 redundant buses. As seen in Fig. 16, the number of pins linearly increases with respect to the increase in the number of channels. This parameter is important for the following reasons. It is quite clear that packing all protocols at the same time requires about 230 user I/O pins on the FPGA. However, the number of FPGA user I/O pins that can be allocated for these communication protocols is restricted by the number of FPGA ports and FMC slots. While a mid-size FPGA (as used by many avionic systems) may have anywhere around 300 to 500 pins, using all these pins for FMC I/O will not leave sufficient pins for the other FPGA peripheral devices (i.e., Ethernet, SFP, LCD, memory, back planes, front panels, etc.). On the other hand, a high-end FPGA which may have up to 1200 user I/O pins are usually chosen to pack dense computational logic, hence using them for multiple communication protocols would not be very cost effective.

4) *Scalability:* The scalability of the system is given in term of number of transmission channels each core can accommodate, which is derived from the protocol specification. ARINC429 is capable of scaling up to a maximum of 20 channels and CAN up to a maximum of 16 channels. However, MIL-STD-1553 is only dual redundant. It is quite important to note that scalability of a protocol does not increase the bandwidth of the system. The bandwidth of the bus is dictated by the standard itself. However, the

number of channels only enables to the system to simply communicate with more number of subsystems at the same time which is the case when different UUTs are using the same avionic protocol.

According to these results, we demonstrated that the FPGA can host the avionic functionality, as far as the communication protocol, where designers can choose the appropriate I/O protocol according to the application requirement. Different parameters can be considered such as the area overhead for different configurations, pin utilization, I/O transmission time, etc. When a new protocol is required, it is enough to design the relevant I/O IP without any PCB replacement, and hence resolving the problem of hardware obsolescence. As an example of protocol that can be used in the future is the Avionics Full Duplex Switched Ethernet (AFDX), which is an avionics data network based on commercial 10/100Mbit switched Ethernet. AFDX is an avionics communication bus mandated by both Boeing and AIRBUS for civilian avionic communication systems. It is designed as a replacement for ARINC429. The point-to-point wiring and the limit on the maximum number of 20 end-systems in ARINC429, makes it highly unsuitable for avionic systems requiring scalable communication infrastructures. Ethernet standard is a well-proven communication standard in the industry, however it is not real time and it often loses packets.

In avionics, packet losses are unacceptable in many cases. For this reason, AFDX is derived from the commercial Ethernet standard with added features to achieve the required deterministic behavior for avionics applications. AFDX uses a special protocol on packets to provide deterministic timing and redundancy management, providing secure and reliable communications of critical and noncritical data. AFDX Switches incorporate functions for filtering and policing, switching (based on configuration tables), and network monitoring thus making it highly scalable compared to the ARINC429 standard.

C. Embedded Avionic Application Results

The use-case scenario for our architecture is a part of an UAV avionic system. The task of the system is to travel between different terrains; to take pictures, to encode sand either store them internally, or to transmit them a remote system. Depending on the scenario, an appropriate communication protocol (ARINC429/CAN Bus/MIL-STD-1553) has to be selected and configured during runtime. For instance, when a secure transmission is needed, the MIL-STD-1553 is chosen. Our avionic system is implemented respecting the architecture of Fig. 11. The Xilinx EDK and ISE tools were used to generate the bitstream. Initially the partial bitstreams are stored in the Compact Flash memory and are read when requested by the application. The operational frequency of the processor, buses, and the peripherals is 100 MHz. A C program is used to initialize and to interact with Microblaze and thus the underlying hardware in order to configure the registers and initiate data transfers. A communication protocol (ARINC249/CAN/MIL-STD-1553) runs in parallel with a JPEG encoder. The captured image is encoded, and is transmitted to an avionic subsystem using an appropriate communication protocol selected during runtime via an FMC interface.

TABLE IV
Area Utilization of the FPGA Design Blocks

	Slices	FFs	LUTs	BRAMs	DSP	ICAP
Microblaze	573	1737	1474	101	3	0
JPEG Encoder	2482	4112	6375	79	10	0
Microblaze peripherals	4937	4543	4051	3	0	1
Design with 3 IPs + JPEG	79 047 (27%)	68 569 (68%)	127 558 (77%)	150 (36%)	13 (2%)	1 (50%)

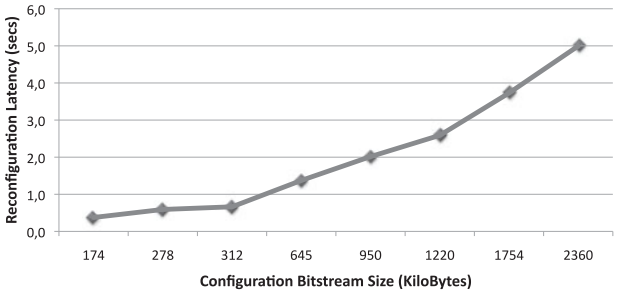

Fig. 17. Reconfiguration latency versus configuration bitstream size.

The area utilization of each hardware component is shown in Table IV. The Microblaze core occupies 573 slices and 101 Block RAMs (BRAM). This is mostly due to the large embedded memory used to store the software executable (application and OS kernel). The JEPG encoder needs over 2400 slices, 79 BRAM blocks and DSP blocks, since it buffers the input image frames and performs DSP algorithms to encode the image. About 4900 FPGA slices and 3 BRAMs are occupied by the peripheral devices and the PLB bus interface. These results show the hardware extra-cost of using a softcore processor (the Microblaze) for data transfer and dynamic reconfiguration management.

1) *Reconfiguration Latency and Application Profile:* We measure the time taken to dynamically reconfigure the system with our communication protocols. All the timing measurements shown are measured from the software. Partial bitstreams are stored in the Compact Flash and read when requested. Fig. 17 shows the reconfiguration latency versus the bitstream size. From the graph, it is quite obvious that the bigger the size of the bitstream the longer the reconfiguration latency. Bitstreams of size less than 500 KBps require less than a second to be reconfigured. Configuration stream of size 645 KBps which is the size of our bitstream (the IP cores with 2 channels), requires roughly about 1.3 s while bitstreams larger than a 2 MBps (corresponds to the IP core with 16 channels) have reconfiguration latency of few seconds. The read queue in the XPS_HWICAP controller buffers the configuration data before it is fed to the ICAP. However, we see that the throughput of the ICAP controller is less than the theoretical maximum because of the disk access overhead caused by the Compact Flash. It is inter-

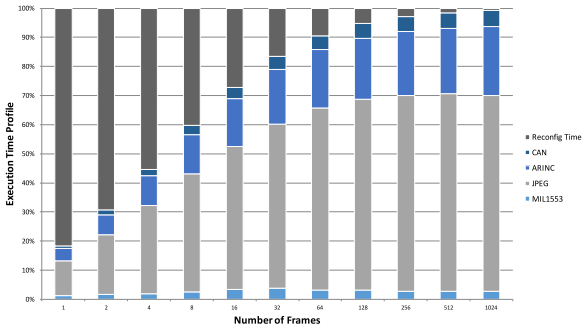

Fig. 18. Application profile.

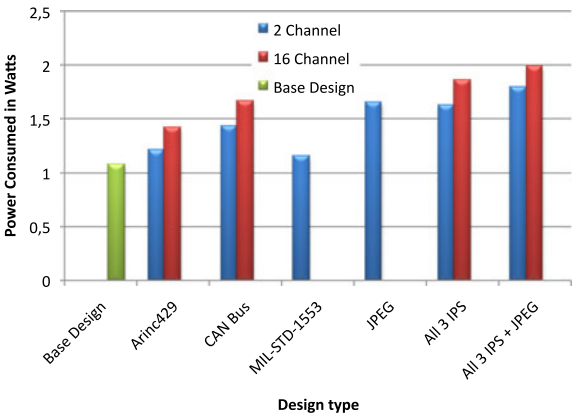

Fig. 19. Total power consumption of each design.

esting to note that the entire avionic protocol is swapped within seconds with uninterrupted system operation.

The reconfiguration latency can be completely hidden in many scenarios as shown in the profile of the application in Fig. 18. From this illustration, it is quite obvious that transmission time is quite negligible compared to the execution time of the application (JPEG). It is also seen that the reconfiguration time become also negligible with significant processed data. As PR means the ability to dynamically modify blocks of logic by downloading partial bit files while the remaining logic continues to operate without interruption, the reconfiguration phase can be anticipated during the application processing.

2) *Power Estimation:* The total power consumption of the design depends on several attributes of the overall design. But it is safe to assume that bigger the design more the power consumption is, unless special power saving mechanisms such as clock gating is applied, among other factors. From Fig. 19, the bigger the design, the greater the overall power consumed. We are comparing the overall power consumption of each design with respect to the design with ARINC429 16 channels, CAN Bus 16 channels, MIL-STD-1553, and JPEG application at the same time. The maximum power savings are about 400 mW when only one IP is present at a time. Although this difference may not be huge, as mentioned earlier, multigigabit avionic protocols are far more complex and power-hungry with several gigabit transceivers operating at the same time.

We highlight that the power consumption of a given hardware implemented on the FPGA depends on several parameters such as the target device, resources utilization, the operating frequency, etc. [28]. In general, the designer has the possibility to tune the hardware solution in order to reach the appropriate implementation that meets the application requirements in terms of performance, power consumption, etc. To do so, high-level synthesis (HLS) tools are used to explore the different hardware solutions and to produce efficient implementation at a reduced time-to-market.

VIII. EXPERIMENTAL RESULTS ANALYSIS AND SCIENTIFIC DISCUSSION

The main focus of our research is to demonstrate the benefits of FPGAs in avionic S&T systems, as flexible and runtime reconfigurable hardware, by characterizing their performances. Several cutting-edge features of FPGAs such as hardware acceleration and DPR are widely exploited as research topics. However, their benefits remain less known and even lesser used in industrial applications. Use of FPGAs in avionic applications has always been challenging due to the safety and certification issues. As FPGAs find more use cases, more efforts are being made to standardize the safety and certification processes. At this level, we have to bring a clear answer to the following question: *How to consider a new avionic functionality in the design process when an FPGA-based implementation is chosen*? The V&V process of this new equipment should be performed respecting the conventional industrial steps: a full simulation, passing through the test benches and finishing with the integration phase. For a long time, these different fields (simulation, test, and integration) were relying on different teams and tools, which is time consuming and error prone while switching between the design steps of the V&V process. Today, it is mandatory to converge toward common frameworks supported by cutting-edge hardware architectures.

Our research calls for the convergence between S&T domains and gives a possible solution to unify the development environment with a reduced cost and time-to-market. First, for the simulation step, the FPGA can be an essential component of the execution support coupled with other computing nodes (multicore, GPU, etc.). It can host avionic models eligible for efficient hardware implementation. At this level, the new functionality can interact with other virtual models in order to verify the overall system behavior. Furthermore, during the last two decades, several academic and industrial efforts have been devoted in order to increase the productivity of FPGA-based designs. This challenge is tackled by means of HLS tools. HLS approach in EDA is a step in the design flow aiming at moving the design effort to higher abstraction levels [29]. Today several existing HLS tools have shown their efficiency for producing acceptable design performances and shortening time-to-market [29], [30]. Fig. 12 illustrates our heterogeneous CPU/FPGA prototype for real-time simulation environment. It supports a dynamic execution model to avoid the timing constraint violation during the simulation. As described in Section IV-B, this environment allows a context switch from a software node to a hardware node

and vice versa at runtime and without a full simulation restart which reduces the verification time. In the first part of our experimental results, we have demonstrated a hardware–software partition for achieving up to ten times speed up in an avionic simulator. In addition, our system was also able to reconfigure during runtime to adapt to the increase in the workload of the application. Avionic simulations take up a lot of time in the system design cycle. For each hour of flight, at least 40 h of simulation is required. A 10× speedup of the workload would imply faster and more efficient design cycles in building avionic simulation.

Second, for the test step, the FPGA can host the avionic functionality as far as the communication protocol which avoid the usage of specific I/O boards. Designers can choose the appropriate I/O protocol (ARINC429, MIL-STD-1553, etc.) according to the application requirement. In addition, when an upgrade is required at the application or the communication level, there is no need to replace the PCB hosting the calculator but just updating the corresponding design or instantiating the relevant I/O IP, and hence it increases the productivity and reduces the cost. In the second part of the experimental results section, we have measured different performance characteristics of different avionic IP cores, such as area overhead for different configurations, pin utilization, I/O transmission time, etc. The comparative statement between different protocols and configurations will allow designers to fix their parameters according to the target equipment, the test scenario, etc. Furthermore, we make profit from the FPGA to support the convergence between the simulation and the test domains. Indeed, we can switch dynamically between the S&T phases in the same environment with just replacing the virtual model with the appropriate I/O protocol to communicate with the UUT using the DPR feature of the FPGA. This is another advantage that allows us to reduce the development time. For this reason, we have also measured runtime reconfiguration time. Fig. 14 illustrates the ML605 kit showing FMC connector loopback using MIL-STD-1553 I/O protocol. In this scenario, we are relying on the same board used for hardware acceleration in the simulation scenario. Hence, the same prototype environment can be used for S&T phases. Our complementary research on Scheduling Problem [31], [32] allows to share the available hardware resources (CPU/FPGA) between different simulation or test projects considering different scenarios offering an additional speedup in the V&V process.

The consideration of the reconfigurable part very early in the V&V design process of a new avionic equipment allows the easy integration on the final system relying on certified technologies. In the third part of the experimental results section, we have presented in detail the performances of a real application that we have designed. Here, we measure different performance characteristics of our system, such as area overhead for different system configurations, power consumption, application execution profile, etc. In addition to the application profile, the aforementioned parameters are also important to consider in order to choose a good partition strategy and improve system life cycle. As a final note to DPR, by dynamically reconfiguring a required communication protocol, we obtain significant improvement in area utilization, with no degradation in the performance of the application. A good partition strategy in addition to system speed-up should also provide hardware reuse and system scalability, thereby increasing the system life cycle. All these results represent a conceptual proof of FPGA-based next generation avionic S&T systems.

IX. CONCLUSION

In this paper, we have presented an FPGA-centric design process for avionic systems. We have redefined the role of the FPGA in the different design steps, namely the simulation, the test, and the integration phases. In the proposed process, a particular attention has been given to the smooth transition between the different steps relying on the same technology which yields to a reduced design cost and time-to-market. The main criteria of reconfigurable circuits in terms of performance, flexibility, and dynamicity have been exploited to define versatile avionic systems respecting several design constraints (real-time, area utilization, etc.). This research work addressed also the challenge of convergence between the S&T domains by proposing an FMC standard-based communication system. The pertinence of experimental results presents a concept proof of the proposed design process. Future works will deal first with the scalability of the system in term of number of computation nodes in order to cover very complex avionic systems or system-of-systems. Second, we will focus on the certification of the proposed concepts that should be used in the next generation of avionic systems.

ACKNOWLEDGMENT

The authors would like to thank the ANRT (French National Association of Research and Technology). The authors would also like to thank industrial partners for their fruitful collaboration, in particular, B. Nakache and M. Nakache.

REFERENCES

[1] G. Afonso, N. Damiani, N. Belanger, R. B. Atitallah, and M. Rubio
Hybrid and multicore optimized architectures for test and simulation systems
In *Proc. 6th Int. ICST Conf. Simul. Tools Tech.*, 2013, pp. 129–138.

[2] G. Afonso, R. Ben Atitallah, N. Belanger, M. Rubio, S. Stilkerich, and J. Dekeyser
Toward generic and adaptive avionic test systems
In *Proc. NASA/ESA Conf. Adapt. Hardware Syst.*, Jun. 2011, pp. 287–294.

[3] G. Afonso, R. Ben Atitallah, A. Loyer, J. Dekeyser, N. Belanger, and M. Rubio
A prototyping environment for high performance reconfigurable computing
In *Proc. 6th Int. Workshop Reconfigurable Commun.-Centric Syst.-on-Chip*, Jun. 2011, pp. 1–8.

[4] V. Viswanathan, R. Ben Atitallah, J.-L. Dekeyser, B. Nakache, and M. Nakache
Dynamic reconfiguration of modular i/o ip cores for avionic applications

In *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, Dec. 2012, pp. 1–6.

[5] W. H. Zheng, N. Marzwell, and S. Chau
In-system partial run-time reconfiguration for fault recovery applications on spacecrafts
In *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2005, vol. 4, pp. 3952–3957.

[6] B. Osterloh, H. Michalik, S. Habinc, and B. Fiethe
Dynamic partial reconfiguration in space applications
In *Proc. NASA/ESA Conf. Adapt. Hardware Syst.*, 2009, pp. 336–343.

[7] L. A. Cardona, J. Agrawal, Y. Guo, J. Oliver, and C. Ferrer
Performance-area improvement by partial reconfiguration for an aerospace remote sensing application
In *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, 2011, pp. 497–500.

[8] L. Sterpone, F. Margaglia, M. Koester, J. Hagemeyer, and M. Porrmann
Analysis of seu effects in partially reconfigurable sopcs
In *Proc. NASA/ESA Conf. Adapt. Hardware Syst.*, 2011, pp. 129–136.

[9] S. J. Lu, P. Siqueira, V. Vijayendra, H. Chandrikakutty, and R. Tessier
Real-time differential signal phase estimation for space-based systems using fpgas
IEEE Trans. Aerosp. Electron. Syst., vol. 49, no. 2, pp. 1192–1209, Apr. 2013.

[10] P. M. Nair and S. Ray
Wireless data acquisition system with signal processing and control modules within field programmable gate array (fpga)
In *Proc. Int. Conf. Elect. Electron. Signals Commun. Optim.*, Jan. 2015, pp. 1–6.

[11] M. Lanuzza, P. Zicari, F. Frustaci, S. Perri, and P. Corsonello
Exploiting self-reconfiguration capability to improve sram-based fpga robustness in space and avionics applications
ACM Trans. Reconfigurable Technol. Syst., vol. 4, no. 1, Dec. 2010, Art. no. 8.

[12] L. A. Tambara, J. Tarrillo, F. L. Kastensmidt, and L. Sterpone
Fault-tolerant manager core for dynamic partial reconfiguration in FPGAs
In *FPGAs and Parallel Architectures for Aerospace Applications: Soft Errors and Fault-Tolerant Design*. Cham, Switzerland: Springer, 2016, pp. 121–133.

[13] F. Cloute, J.-N. Contensou, D. Esteve, P. Pampagnin, P. Pons, and Y. Favard
Hardware/software co-design of an avionics communication protocol interface system: An industrial case study
In *Proc. 7th Int. Workshop Hardware/Softw. Codesign*, 1999, pp. 48–52.

[14] R. Bauer
Embedded synthetic instruments for o-level test: Modular io and fpga technology provide increased flexibility and decreased cost of test
In *Proc. IEEE Autotestcon*, Nov. 2015, pp. 284–287.

[15] N. Belanger and J.-P. Lebailly
Promoting avionic test systems as productivity enablers
In *Proc. 5th Int. Conf. Syst.*, Les Menuires, France, 2010, pp. 66–70.

[16] 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 *Proc. ETTC Int. Conf.*, Toulouse, France, 2009, pp. 35–42.

[17] N. Belanger, N. Favarcq, and Y. Fusero
An open real time test system approach
In *Proc. IEEE Int. Conf. Adv. Syst. Test. Validation Lifecycle*, Porto, Portugal, Sep. 2009, pp. 38–41.

[18] H. Plankl
Embedded solutions for development tests, component tests and system integration in the test centre
In *Aerosp. Test.*, Munich, Germany, 2009.

[19] S. H. VanderLeest and D. White
Mpsoc hypervisor: The safe and secure future of avionics
In *Proc. IEEE/AIAA 34th Digit. Avionics Syst. Conf.*, Sep. 2015, pp. 6B5–1–6B5–14.

[20] C. Insaurralde
Reconfigurable computer architectures for dynamically adaptable avionics systems
IEEE Aerosp. Electron. Syst. Mag., vol. 30, no. 9, pp. 46–53, Sep. 2015.

[21] G. Afonso, W. Godard, R. Ben Atitallah, and J.-L. Dekeyser
Test and simulation system
World Intellectual Property Organization Patent WO 2015/097105 A1, Jul. 2015.

[22] S. Asano, T. Maruyama, and Y. Yamaguchi
Performance comparison of FPGA, GPU AND CPU in image processing
In *Proc. 19th IEEE Int. Conf. Field Programmable Logic Appl.*, Prague, Czech Republic, Aug. 2009, pp. 126–131.

[23] O. Souissi, R. Ben Atitallah, A. Artiba, and S. E. Elmaghraby
Optimization of run-time mapping on heterogeneous cpu/fpga architectures
In *Proc. 9th Int. Conf. Model. Optim. Simul.*, Bordeaux, France, 2012, pp. 127–135.

[24] VectorFabrics: Pareon Analyze your sequential C code to create an optimized parallel implementation
2013. [Online]. Available: http://www.vectorfabrics.com/.

[25] P. Coussy, C. Chavet, P. Bomel, D. Heller, E. Senn, and E. Martin
GAUT: A High-Level Synthesis Tool for DSP Applications, P. Coussy and A. Morawiec, Eds. Dodrecht, The Netherlands: Springer, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-1-4020-8588-8_9

[26] J. R. Villarreal, A. Park, W. A. Najjar, and R. Halstead
Designing modular hardware accelerators in c with roccc 2.0
In *Proc. 18th IEEE Annu. Int. Symp. Field-Programmable Cust. Comput. Mach.*, Charlotte, NC, USA, May 2010, pp. 127–134.

[27] A. Jacobs, G. Cieslewski, A. D. George, A. Gordon-Ross, and H. Lam
Reconfigurable fault tolerance: A comprehensive framework for reliable and adaptive fpga-based space computing
ACM Trans. Reconfigurable Technol. Syst., vol. 5, no. 4, pp. 21-1–21-30, Dec. 2012.

[28] R. B. Atitallah, E. Senn, D. Chillet, M. Lanoe, and D. Blouin
An efficient framework for power-aware design of heterogeneous mpsoc
IEEE Trans. Ind. Informat., vol. 9, no. 1, pp. 487–501, Feb. 2013. [Online]. Available: http://dx.doi.org/10.1109/TII.2012.2198657

[29] R. Nane *et al.*
A survey and evaluation of FPGA high-level synthesis tools
IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 35, no. 10, pp. 1591–1604, Oct. 2016.

[30] W. Meeus, K. Van Beeck, T. Goedemé, J. Meel, and D. Stroobandt
An overview of today's high-level aynthesis tools
Des. Autom. Embedded Syst., vol. 16, no. 3, pp. 31–51, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10617-012-9096-8

[31] A. Ait El Cadi, R. Ben Atitallah, S. Hanafi, N. Mladenović, and A. Artiba
New mip model for multiprocessor scheduling problem with communication delays
Optim. Lett., pp. 1–17, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11590-014-0802-2

[32] A. Ait El Cadi, O. Souissi, R. Ben Atitallah, N. Belanger, and A. Artiba
Mathematical programming models for scheduling in a cpu/fpga architecture with heterogeneous communication delays
J. Intell. Manuf., pp. 1–12, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10845-015-1075-z