Avionic System based on Partial Reconfigurable FPGA

Rodrigo de Faria Romero
Electronics Engineering Division
Aeronautics Institute of Technology - ITA
São José dos Campos, Brazil
r_fromero@hotmail.com

Osamu Saotome
Electronics Engineering Division
Aeronautics Institute of Technology - ITA
São José dos Campos, Brazil
osaotome@ita.br

***Abstract* — the present paper proposes an avionic system hardware designed based on partial reconfigurable FPGA. A simple avionic system is presented as case study and point out on-chip redundancy applied to FPGA besides device redundancy. Reconfiguration times is measured and analyzed. The design is validated via practical experiment.**

Keywords — avionics; partial reconfiguration; FPGA

I. INTRODUCTION

Aviation electronic architectures are real-time embedded systems (RTES) composed of digital processing modules and communication buses. Avionic systems support functions such as navigation, guidance, flight control, passenger entertainment and air/ground communication. According to [1] avionics systems represent up to 40% of aircraft cost in civil sector and up to 50% in military sector.

Since 70s until 90s avionics systems were based on federated architecture concept where each software function is performed by an individual computer processing unit connected to its dedicated sensors and actuators [2]. The dependencies between standalone subsystems are well understood and a specific data communication bus, standardized in ARINC 429, isolate faults among functions with distinct criticalities. Federated architecture allows third-part development and certification of each software function according to global design specifications.

However, hardware redundancy requires that there are two (Dual Modular Redundancy) or three (Triple Modular Redundancy) computers for each software function. Along with this drawback avionics functionalities complexity increased naturally. More precise flight controllers, more complete entertainment systems for the passengers and general advances on digital systems lead to federated architecture limit. The large amount of equipment and electrical connections correspond to high maintenance costs and elevated weight and volume inside aircraft.

Front of this scenario, the first generation of Integrated Modular Avionics (IMA1G, or just, IMA) concept, based on centralizing and reutilization of communication and computation resources is adopted in order to keep the costs, weight and volume within reasonable limits. IMA architecture provides that multiple software functions with different criticality levels can be performed by a single computer processing unit. Therefore, twice or three times the same hardware copy is able to perform more than one software function redundancy, reducing weight and power consumption.

Strict and robust partitioning in terms of software and hardware ensures that each software function is allocated with its own non-shared virtual resources which avoids interference among different functions. It can be done according to ARINC 653 principles, where different functions resident in a same resource module must be partitioned respecting space and time constraints. In IMA principle communication resource and managements are described in ARINC 664 and communication flows are statically split into virtual links. ARINC 664 bus can be up to one thousand times faster than ARINC 429 bus applied to federated architectures.

On the other hand, computation sharing adds new indirect dependencies between subsystems and communication sharing might causes limitation of the communication flows. Shared resources may cause common-cause failures and communication bus overloads which cause delays and/or loss of data.

Besides IMA drawbacks, dependencies between applications and communication problems, fault tolerance, containment and reconfigurability leads to a natural trend for the second generation of IMA (IMA2G) [3]. Software function reconfiguration capability could allow that a hardware failure would be mitigated through software function reallocation to a safe module. Therefore, module reallocation correspond to a low level of redundancy and enables that the system remain operating until the aircraft has reached a maintenance location.

Furthermore, multicore and many-core architectures have already substituted single-core processors in almost all sectors, such as telecommunication and personal computers. Single-core processors are becoming even rarer, but an avionic system based on multicore processors would be even harder to certificate due to the greatest difficulty in ensure time predictability.

In this work we propose substitute software function reconfiguration by hardware partial reconfiguration (PR) thru Field Programmable Gate Arrays (FPGAs) instead of single-core or multi-core processors units. The rest of paper presents pros and cons of purposed approach, followed by a case study, its results and conclusions.

II. PURPOSED APPROACH

FPGAs are used to implement different complexity levels of digital circuits in the same die. These circuits are described by the designer through a Hardware Description Language (HDL), such as Verilog and VHDL, loaded in vendor IDE and then synthesized and implemented for the target FPGA. The IDE generates a full bit stream which is downloaded in FPGA memory. When power-up the device reads the full bit stream from FPGA memory and synthesize the described circuits using an array of combinational logic blocks and reconfigurable interconnects. Input/output blocks allow external signals to be connected with the internal signals of the FPGA.

In that connection, the FPGA technology provides the flexibility of on-site programming and re-programming through memory bit stream uploading. The flexibility can be increased allowing the modification of an operating FPGA design by means of partial bit streams. A partial bit stream can modify reconfigurable regions in the FPGA without compromising the integrity of the applications running on those parts of the device that are not being reconfigured [4]. The designer may specify distinct configurations for the system and use the IDE in order to generate partial bit streams in addition to full bit stream.

PR in run-time is present in modern FPGAs, such as Altera Stratix-V and Xilinx 7-series, along with advanced IDE, such as Altera Quartus and Xilinx Vivado. However, partial reconfiguration is not trivial due to possible device fragmentation and communication between static and newly implemented partition [5]. Mainly vendor's IDE provide partial reconfiguration controllers in order to manage with fragmentation and communication issues. A partially reconfigurable design requires bottom-up synthesis, i.e. each module has its own synthesis project. There is no optimizations across module boundaries.

In this scenario, avionics systems should be designed as hardware modules/components, taking advantage of concurrence and parallelism among process provided by FPGAs. This approach remains the modularity feature present in federated architecture.

If necessary, a single-core or multi-core soft processor can be implemented in FPGA and execute sequential instructions, but you should remember that there are certification problems about multi-core applied to avionic systems. There are still diverse System-on-Chip (SoC) devices which provide powerful microprocessor, like ARM, and FPGA in the same package.

In addition to abovementioned advantages, our hardware avionic system introduces on-chip redundancy for mixed criticality modules, besides the possibility of the devices redundancy. As well in federated and IMA architectures the purposed approach allows device replicating in order to mitigate faults, but also offer hardware function reallocation inside the FPGA thanks to PR technique. These features enhance system robustness, cost, area, and power reduction.

III. CASE STUDY

Ours purposed case study consists of a simple avionic system composed by four functions with different criticality levels, Fig. 1. Clock management is responsible for clock distribution among other modules. The aircraft strobe light function is used to help other pilots to recognize the aircraft's position during night operations and is part of static design. Navigation function receives data from accelerometer and gyroscope of Inertial Measure Unit (IMU) and calculates the aircraft's attitude. It is a high critical function that in a fault condition should be reallocated into a spare module or even replacing a less critical function. In our case study the low critical function is the passenger entertainment system which display some images on monitors and should be disassembled in order to provide required space for higher critical function that suffer a failure.

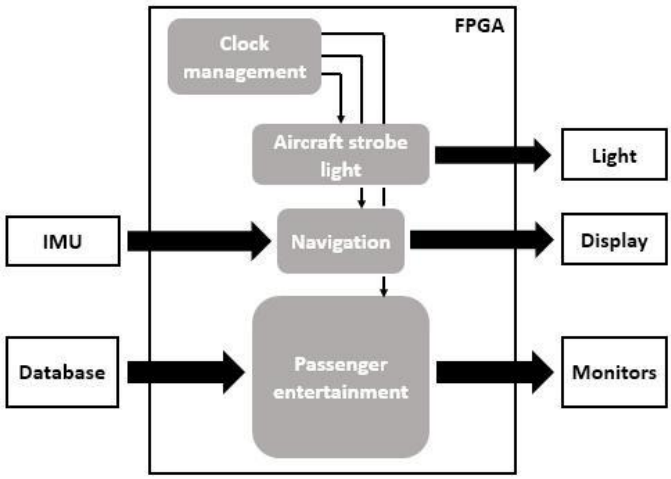

Figure 1 – Simple avionic system.

Each of four functions were described in VHDL language without using any Intellectual Property (IP) provide by FPGA's IDE vendor. Therefore, this avionic system could be implemented in any FPGA, regardless brand or model, since there is enough space inside the device. In our case, we implemented the design in Artix-7 FPGA present in Digilent Basys3 development board.

The lower and higher critical function modules must have the exactly the same inputs and outputs, in order to be implemented on a same FPGA partition. Therefore, either low as high critical function must be designed as a superset module, i.e. their inputs and outputs are of both function circuits. See Fig. 2.

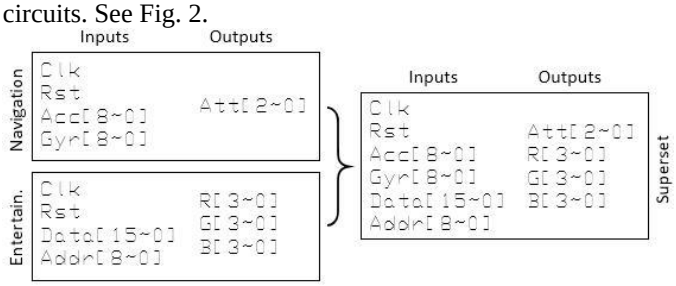

Figure 2 – Superset design.

Because both modules have spare signals, same inputs and outputs will not be used in each function. For inputs there is no problems, once an input signal can be simply left in high impedance. However, two or more outputs must not drive a same line due to the conflict situation between low and high digital levels. The synthesis tool is not able to detect mutual exclusion between two module outputs, so the mux ensures module outputs isolation. Front of this problem, Fig. 3 shows our purpose of using a Finite State Machine (FSM) in order to control the buses through multiplexer (mux).

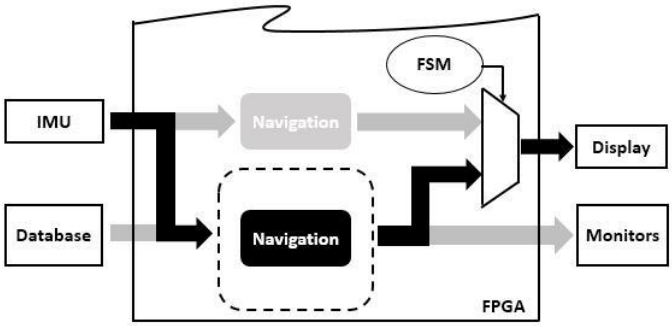

Figure 3 – FSM and mux for bus control.

When the fault is detected though a signal flag, the FSM changes the mux selection for swap the bus connection from the low to the higher criticality function. Therefore, this approach spends one mux per partition and can use one FSM for control all the mux.

In our experiment, there is no fault generator, so we connected a personal computer to FPGA via JTAG/USB interface. The designer may send a command via Vivado IDE in order to send the partial bit stream for FPGA partial reconfiguration process. The FSM detects the reconfiguration and swap the mux. The reconfiguration time is measure by means of digital logic analyzer connected to FPGA output. This signal toggle between the beginning and the end of the reconfiguration process. The gap correspond to reconfiguration spent time.

IV. RESULTS

First, we implemented the whole design through full bit stream download. According to the logic analyzer the full configuration spent 25.0 ms to be completed.

Still with respect to this experiment, we made use of partial reconfiguration in order to substitute a low critical function, passenger entertainment, by navigation function at run-time. The static portion, aircraft strobe light, remained operating during the 9.2 ms (in other words, 36 % of full configuration time) spent on reconfiguration process and carried on without been affected.

V. CONCLUSIONS

The avionic system proposed in this paper was tested through case study model and the reconfiguration time is small enough to keep the system operating until the aircraft has reached a maintenance location or even lost the control. The on-chip redundancy feature was validated via practical experiment using a Digilent Basys3 board and Xilinx Vivado. However, is important to point out that the purposed approach is valid for any FPGA, regardless of brand or model.

REFERENCES

[1] P. Bieber, F. Boniol, M. Boyer, E. Noulard, and C. Pagetti, "New challenges for future avionics architecture*", Journal AerospaceLab*, vol. 4, 2012.

[2] R. Wolfig and M. Jakovljevic, "Distribuited ima and do-297: Architectural, communication and certification attributes", in Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th, pp. 1.E.4-1-1-1.E.4-10, Oct 2008.

[3] B. Andrillon and D. Aviation, "*Contribution of integrated modular avionics of second generation for business aviation*", 2013.

[4] Vivado Design Suite User Guide: Partial Reconfiguration. UG909 (v2015.1) April 1, 2015.

[5] C. Bobda. Introduction to Reconfigurable Computing: Architectures, Tools and Applications. Springer, 2007.