DISTRIBUTED AND RECONFIGURABLE ARCHITECTURE FOR FLIGHT CONTROL SYSTEM

Manel Sghairi, Jean-Jacques Aubert, Patrice Brot; Flight Control System Department AIRBUS France 316 route de Bayonne, 31060 Toulouse, France

Agnan de Bonneval, Yves Crouzet, Youssef Laarouchi; CNRS; LAAS; Université de Toulouse; UPS, INSA, INP, ISA; F-31077 Toulouse, France

Abstract

New airplanes must meet rigorous requirements of aviation safety, operational reliability, high performance and energy efficiency at a low cost. To meet this challenge, we should optimize current system and take advantage of available technology for the next decade.

This work is aiming at proposing some evolutions for Flight Control System (FCS) and to build alternative FCS low-cost and safe architectures for the next decade with less hardware and software resources.

The main contribution of this paper is twofold. First, we will provide an incremental methodology to give guidelines for architecture optimization. Second, we will present a full distributed reconfigurable architecture for FCS based on smart actuators and digital communication network where all system functions are distributed to simplex Flight Control Computer (FCC) nodes and remote actuator electronics nodes (FCRM). Communication between FCC and FCRM will be based on Airbus embedded communication network (ADCN, Advanced Data Communication Network) [1] and a 1553 bus. We will use ALTARICA language to perform dependability evaluation at architectural level in order to check the effects and benefits of the new architecture on the dependability of FCS.

Introduction

Airplane performance and business pressures related to cost have been the main drivers to change flight control system from mechanical to digital Fly-By-Wire (FBW) design [2]. Technical improvements considered for the future, such as smart actuators/sensors with remote electronics and digital communication, will change drastically avionics architectures design for future commercial and military programs [3,4]. A FBW control system has several advantages over a mechanical system but equipments and architectures proposed for FBW critical systems such as FCS must meet stringent safety and availability requirements before they can be certified. For FCS, the probability of losing an aircraft critical function or of an occurrence of a critical failure must be less than 10^{-9} per flight hour.

Traditionally [5], FCS has used a centralized /federated architecture where a specific fault tolerant computer has performed all processing and authority. This architecture is inherently robust, because it is based on a high level of software and hardware redundancy. However, it can be very costly in terms of space, weight and power, and also wiring requirements between the elements of the system especially for large airplane. This also increases all continuous monitoring of "non-intelligent" components like actuators and sensors that the computers are performing at the present.

Given the high level of redundancy practiced, it seems interesting to try to propose alternative architectures with less hardware and software resources and to take advantage of technical improvements.

In this context, there is a great motivation for future programs to change current flight control architectures to more distributed and better optimized architectures as shown in Figure 1.

FCS architectures will be based on digital technologies and intelligent subsystems and offer many improvements on centralized architectures. They can help to reduce redundancy and the complexity of principal computing elements in FCS through the migration of some functions out of the FCC and the integration of smart subsystems.

978-1-4244-4078-8/09/$25.00 ©2009 IEEE.

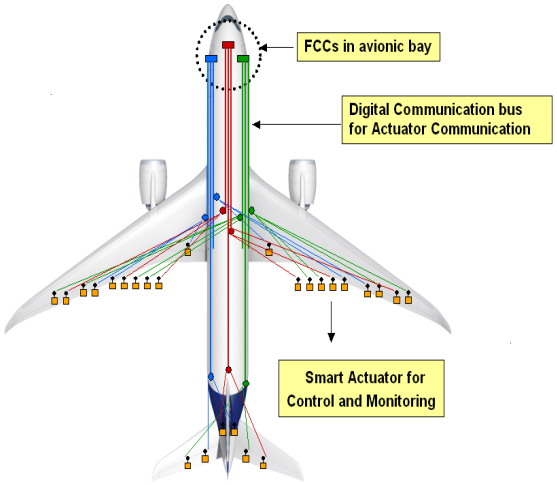

Figure 1. Full Distributed FCS Architecture

In this paper we propose a conceptual fully decentralized and reconfigurable architecture for FCS with architecture optimization and control distribution, where it is possible to use systems resources and new technologies better.

FCS is a very critical system and consequently must be carefully designed and exhaustively checked. We validate the proposed architecture through simulation using ALTARICA language (a high level formal description language) and SDT (System Design Tool) for system safety and reliability assessments.

The paper is organized as follows. This first section has presented flight control systems evolutions. The second section analyzes the state of the art of current FCS architectures. The third section gives an overview of an incremental methodology for architecture optimization. The fourth and fifth sections describe and analyze massive voting architecture, and illustrate the use of ALTARICA for dependability evaluation.

State Of The Art Of Current FCS And Their Requirements

Traditionally, FCSs have used a centralized and federate architecture where a specific computer has performed all processing and authority. In the context of our work we have analyzed a set of FCS architectures. The first subsection presents the Airbus and Boeing design. The second subsection presents a short analysis of redundancy, and the last subsection presents the system requirements identified.

Airbus And Boeing Design

The Airbus flight control system is based on many self-checking flight control computers [6]. Each FCC is composed of two software variants or units (command and monitoring unit) [7] whose results are compared. The command unit and the monitor unit are separated channels within a single computer.

Each channel has separate hardware and different software. If the results of the channels don't correspond (as checked by a comparing function) or are not produced at the same time then an error is assumed and system control switches to another computer. Computers communicate with each other through point-to-point digital communication in order to manage FCS redundancy taking into account different failure cases.

The Boeing PFCS (Primary Flight Control System) [8] comprises three Primary Flight Computers (PFCs), each of identical design and construction and four analog computers ACE (Actuator Control Electronic).

The PFCs compute control-surface position commands and transmit position commands to ACE via ARINC buses. The ACEs position the control surfaces using actuator systems. The ACE units act as an intermediate stage between the PFC and the pilot and actuators. Each PFC is identified as a channel and is composed of three dissimilar computing lanes [9]. Primary flight control system lines have all the same input signals and are all active. Their outputs are connected to a voter that compares these signals. Majority voting then chooses the correct signals. 2-out-of-3 voting can mask the faulty module. Each actuator is controlled by a single ACE and each ACE can receive orders from all PFCs.

All Flight Computers in Airbus and Boeing design are installed in the avionics bay and are connected directly by individual wires to all relevant sensors/actuators through point-to-point links. The relations between flight computer and actuators are arranged so that different computers control each actuator with priority order, so loss of a single computer will not mean loss of control of that surface.

System Analysis

The analysis of current flight control architectures shows that the design and implementation of such a safe system are realized through the combined use of redundancy and diversity (software redundancy) to minimize the probability of common mode failure between redundant units. It also shows that level of redundancy is very important.

This "over-redundancy" is justified by the need for a demonstration of safety and operational reliability especially for commercial airplane, which is guided by regulation authority and economic pressure.

However, given the high level of redundancy practiced, it seems interesting to try to propose alternative architectures on less hardware and software resources. To conduct this exercise, we first have to identify and classify all requirements to be met by flight control system architecture.

System Requirements

Safety And Civil Aviation Regulations

Fail-safe design concepts [10] are required by civil aviation regulations. The system has to meet the FAR/JAR 25 (Joint Aviation Authority/Federal Aviation Regulations) requirements for certification [11, 12]. It means that for a planned or existing system it must imperatively be possible to demonstrate its level of safety in order to be accepted by the authorities. This is to show that the system is robust against any considerable failure or combination of failures [13, 14].

The flight control system usually has two types of dependability requirements:

- Integrity: the system must not output erroneous signals. In particular, Flight Computer should not send incorrect information to the actuators.
- Availability: the system must have a high level of availability.

Economic Requirements

Operational reliability is very important for airlines to stay competitive. FCS must have sufficient redundancy of software and hardware components so that a failure will not disrupt the availability of the system services. The availability objective of flight control systems is to be able to dispatch the aircraft with one or more components failure, so aircraft may take off with one defective equipment. The airplane will have a large operational availability and relatively few maintenance hours, to enable airlines to organize easy maintenance for their fleet. It is required that the FCS be still usable with the expected level of safety, even if an equipment failure could not be repaired for several days (ie. before returning for maintenance). The number of successive flights under such conditions is limited.

Radiation Environment

Electromagnetic radiation should also be considered. The radiation must not affect data communication associated with the Fly-By-Wire system. Particularly, the system must be especially protected against over voltages and under voltages, electromagnetic aggressions, and indirect effects of lightning.

Manufacturing Faults

The choice of technological components and process development strategies [15] (quality control, rules for equipment design) are important factors to control reliability. Despite the precautions taken, a decline in production quality may occur in several defective components (less reliable). Thanks to the inclusion of additional redundancy, FCS provides sufficient margins to tolerate this kind of fault [16].

Incremental Methodology

Analysis of existing FCS architectures, and their requirements, lead us to introduce a brief overview of an incremental methodology to build a new architecture based on progressive requirements injection and distribution of the function of the system [17]. The question we are trying to solve is: what level of redundancy has to be achieved?

Flight control systems are complex. There are several subsystems (flight control computer nodes, actuator nodes, communication network,) with functional and structural dependency. Each subsystem has different timing and dependability requirements with different levels of criticality. For these reasons, a structured approach is necessary for architecture optimization. It is more natural to proceed in a gradual manner by building and validating the architecture step by step, this is the objective of the incremental methodology: starting

with a basic block architecture and then taking into account each requirement, which results in duplication of hardware or/and software or function migration. This approach allows us to analyze the real needs and justify each additional hardware and software cost.

The steps in the incremental methodology process are:

- Step 1: identification of all subsystem boundaries and requirements. At the start, we advise to define all principal subsystems without looking for their dependency.
- Step 2: allocation of tasks (system functions) under an optimizing criterion of the central control because FCCs are complex, big and expensive.
- Step 3: definition of safety objective per subsystem. Safety objective is the probability of system failure due to a subsystem failure.
- Step 4: choice of basic block architecture to meet functionality. Firstly, only necessary functional capabilities must be realized. A single component should be sufficient (one computer, one actuator or one switch…).
- Step 5: classification of requirements based on their criticality.
- Step 6: injection of the first requirement.
- Step 7: assessment of quantitative reliability and preliminary evaluation of the objective of the probability (we can use assumptions for calculation formula).
- Step8: use of hardware/software replication, function migration or reconfiguration to meet the probability objective with the first requirement.
- Step 9: iteration over all requirements.
- Step 10: iteration over all sub-functions.

This approach is part of a complete safety process methodology that allows us to define a new safe architecture for a complex real time safety-critical system.

Example

In this subsection we apply our approach on the most critical subsystem of the FCS: the flight control computer system where a single simplex FCC can handle all system processing and monitoring. But FCS must be designed to continuously provide service despite failure, so we need redundancy.

Flight control computer primary architecture is given by the necessary basic simplex node and software required for system functionality (laws computation). Extra hardware is then added to the architecture as hardware and software replication or dynamic software reconfiguration in order to meet the safety and availability requirements due to permanent and transient faults. In our approach, it is possible to reconfigure one or more FCC to meet dissimilarity requirement.

Dynamic software reconfiguration is a useful mechanism to adapt and maintain systems dissimilarity without need of other forms of reliability, such as redundancy. We consider that the probability objective is 10^{-9} per flight hour for the flight control computer system and the failure rate of one computer does not exceed 10^{-4} per flight hour. As a result, we are in need of additional redundant components, so other requirements should be injected.

We use assumptions to simplify the calculations of probability formula. The formula for probability calculation changes with the number of redundant equipments used to build a fault tolerant architecture and the MMEL (Master Minimum Equipment List) condition: for 3 BFCC primary architecture, and taking into account three requirements (integrity, availability and operational reliability). The probability is calculated as follows:

$$P1 \cong 3\lambda 1T0 \times 2\lambda 1T0 \times \lambda 1$$

Abbreviations And Acronyms

- BFCC: Basic Flight Control Computer
- MTBF: Mean Time Before Failure
- P: Mean Probability per flight hour for the system total failure.
- T1: Maintenance interval or MMEL rectification interval: number of flight hours performed without maintenance action.

- T0: Mean flight time
- $\lambda 1$: Failure rate of FCC ($\lambda 1 = MTBF^{-1}$).

The probability of system failure is simply the sum of individual BFCC failures probabilities. The formula is organized in three parts:

Initially, 3 active components exist. It is accepted to lose one component before or during the flight. This can occur during the time limit T1. The aircraft may take off with defective equipment. The number of successive flights under such conditions is limited to ten (T1 = 100 hours: 10 flights of 10 hours).

The aircraft performs 10 successive take-offs with BFCC 1. During the flight, it is tolerable to lose another computer; this can exist during time T0.

BFCC 1 failure must occur at first, followed by BFCC 2, and finally BFCC 3 failure. Last failure must occur during the flight to lose the whole system.

The last failure is catastrophic and should be shown to occur at a rate less than or equal to $1x10^{-9}$ per flight hour (combined with former failures) for computer flight control systems architecture. System failure must occur after triple combination failure (loss of three BFCC) without repercussion phase. P1 must be less than 10^{-9} per flight hour.

Under MMEL (Master Minimum Equipment List), P2 must be less than 10^{-8} per flight hour and P3 must be less than 10^{-9} per flight hour.

$$P2 \cong 2\lambda 1T0 \times \lambda 1$$

$$P3 \cong 3\lambda 1T1 \times 2\lambda 1T0 \times \lambda 1$$

This example shows that incremental methodology allows us to reduce the number of FCC nodes in the architecture.

Future Architecture For FCS

Currently, smart element (actuator and sensor) on current flight control system is capable of pre-processing data in digital form. Smart actuator comes with their own computational elements and will be equipped by flight control remote modules (FCRM). FCRM is typically an Application-Specific Integrated Circuit (ASIC) or a Field Programmable Gate Array (FPGA). But for commercial flight control system, overall critical function and authority is still retained within the primary flight computer. In other words, the FCC still makes all the important (safety critical) decisions and the smart subsystems interact intelligently with it.

Distributed architecture offers a number of improvements over centralized architectures by re-hosting data processing and control functionality from the primary computational elements into other subsystems and making them more and more intelligent. Next subsection presents a distributed architecture, with migration of some functions from FCC to FCRM nodes. Distribution refers to distribution of computing power, control and monitoring.

General Description Of The Massive Voting Architecture

With distributed flight control architecture, there are several possibilities to allocate the task of control laws and logic (monitoring, fault detection and handling). Our optimization strategy to build the massive voting architecture implies that redundancy management or voting and logic should be allocated to actuators nodes or shared between computer and actuator nodes. This give a high degree of hardware fault detection for both actuator and computer fault without extra hardware. Most voting algorithms do not demand high processing capability, so processing in the actuators nodes is not considered a limiting factor of a future distributed architecture on flight control systems. The massive voting architecture benefits from digital communication and new technology for smart actuators:

- Digital communication provides broadcast communication between FCC and actuator nodes.
- Digital communication is rapid responding to remote terminal especially for large airplane.
- Electronic for smart actuator can be designed with high degree of embedded computing capability.

We considered an architecture with N simplex, independent computer nodes, grouped into two groups (of N/2 elements). In this architecture, we

replicate the main hardware unit N times and their outputs are constantly voted by a massive voting algorithm implemented in M distributed actuator nodes. In some cases, there are two actuators per surface and one FCRM per actuator. Communication between FCC and FCRM will be based on Airbus current embedded communication network (ADCN, Advanced Data Communication Network) and a 1553 bus.

FCC are simplex, but FCRM are duplex (command/monitor architecture). Each FCRM has it own voter. FCCs have two software variant (S1 and S2) and two hardware variant (H1 and H2). Each voter on each FCRM needs to collect the output orders of all FCC nodes and of the two plane sides as shown in Figure 2.

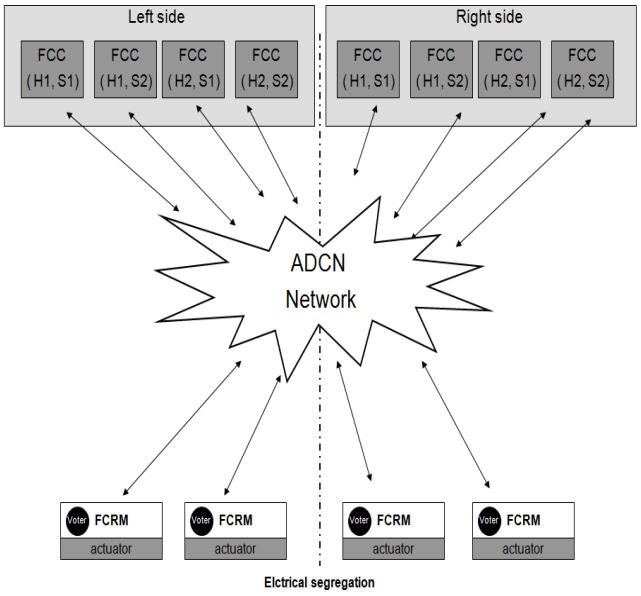

Figure 2. The Massive Voting Architecture

FCCs are connected to the ADCN network and can communicate to all actuator nodes trough a multi-master broadcast configuration. All FCC intra-communications are removed. Communication between flight control computer nodes and actuators nodes is organized as follows.

- Firstly, all FCC nodes calculate flight control laws and control-surface position commands for all actuators nodes (spoiler, elevator…) and then broadcast their message on the bus.
- Each actuator receives N/2 messages from each computer group at every application cycle (control law computation frequency).
- Secondly, each FCRM node achieves a massive voting to select a good order. In absence of fault all correct working voter should agree.

The voter may use different algorithms in the voting process of selecting correct order [18, 19].

Fault Handling

In massive voting architecture most fault handling is taken care of in actuator nodes. With several actuator nodes, each of them providing a feedback, a high degree of fault detection and fault location can be achieved.

Because FCC nodes are simplex, this requires a fault detection function to detect the faulty situations. First simple fault detection mechanism in FCC nodes use the output signal for inner consistency checking like parity checking or watchdog timers. The second fault detection mechanism is based on FCRMs feedbacks. If a fault occurs in an actuator node, that node will either be fail-silent or broadcast faulty command-words since the actuator has a command monitoring architecture. The command channel ensures the function allocated to the FCRM (voting, monitoring). The monitoring channel ensure that the command channel operate correctly.

Simulation

Up to 80 percent of the total cost of the life cycle of an airplane is set during the early design phase, so mistakes on architectural decision are expensive. To minimize risks, dependability analysis should be introduced early in the design process, and decision should be based more and more on simulation.

This section discusses modeling and dependability assessment of massive voting architecture with ALTARICA language. All dependability measures can be evaluated based on ALTARICA model but in this paper and for FCS we are just interested in safety and availability. We need to verify the effect of the massive voting architecture on system requirements, and application.

ALTARICA Language

ALTARICA is a formal language developed at LaBRI (Laboratoire Bordelais de Recherche en Informatique) jointly with French industrial partners (especially Dassault Aviation and Airbus) in order to model safety critical systems. ALTARICA is used for modeling both functional and dysfunctional behaviors of systems. ALTARICA is widely used by aeronautical industrialists [20].

Thanks to the language well defined semantics and syntax, safety assessments of ALTARICA models can be analyzed by numerous reliability or validation tools. Moreover, its capacity to realize compositional and hierarchical models is a great advantage when complex systems must be modeled [21]. An ALTARICA model is composed of several components linked together. Each system component is modeled by a node. A node is defined by three parts:

- declaration of variables and events
- definition of transitions
- definition of assertions

Most of the events of an ALTARICA model, that describe failure propagation in a system, represent failure modes of the components of the system. These events are mainly stochastic events: probability laws can be associated to them and later be used to evaluate the enforced quantitative requirement. The means of analysis on ALTARICA model are:

Interactive simulation:

- possible events may be triggered
- component icons and links color are updated

Automatic generation for a selected output value of:

- Fault tree
- Minimal Cut Sets (MCS)
- Minimal Sequence Sets (MSS)

Model-Checking:

- given a requirement, exhaustive exploration of reachable states in order to find a state where the requirement is not fulfilled
- production of a counter-example if the requirement is not fulfilled.

Application On Your Architecture

Architecture Modeling

Using SDT (System Design Tool) workshop of Airbus, we designed and implemented, a small ALTARICA model of the massive voting architecture for experimentation. In our simulation scenarios N is equal to six. The simulation model includes all systems communication, computing nodes (FCC and FCRM), electrical system and control surfaces and their failure modes to study failure propagation in the model as shown in Figure 3.

Massive voting architecture component has several failure modes:

- total loss
- detected erroneous functioning
- undetected erroneous functioning
- erroneous acquisition of data
- erroneous transmission on network

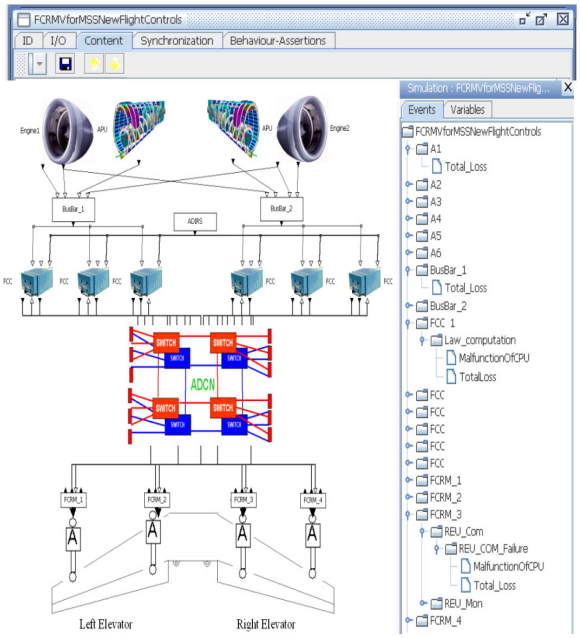

Figure 3. An Architectural Altarica Model

ALTARICA component model is composed of:

- A textual description (flow and events impacting the current state of the component) to describe both functional and dysfunctional behaviors as shown in Figure 4.

```
node S
  state
    status : {correct, lost};
  flow
    PowerFromElecGen: bool , in;
    ActivationOrderReceived: bool , in;
    OrderToSc: {correct, lost} , out;
  event
    sleeping_loss, acting_loss;
  trans
    PowerReceived and status = correct
            and not Activation |- sleeping_loss -> status := lost
    PowerReceived and status = correct
            and Activation |- acting_loss -> status := lost;
  assert
    OrderToSc = case {PowerReceived : status;
                      else lost};
  init
    status := correct;
edon
```

Figure 4. Altarica Node Textual Description

- A graphical representation (flow and icons updated to reflect the current state) as shown in Figure 5 of FCRM model.

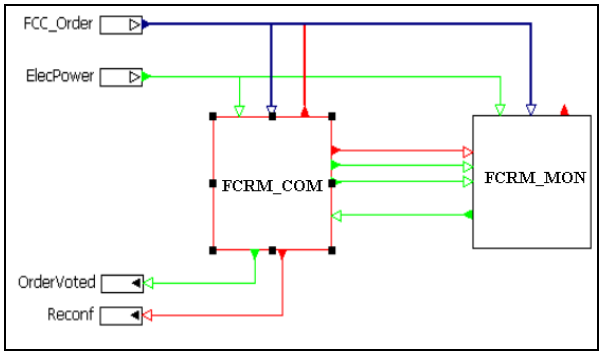

Figure 5. FCRM Node Graphical Representation

Safety Assessment With ALTARICA

After having modeled the architecture, we can perform dependability evaluation in order to check the effects and benefits of the new architecture on the dependability of FCS. We check the effect of failure occurrences on the system architecture by using SDT graphical interactive and automatic simulator.

Firstly, we use interactive simulation to validate each component behavior separately in order to verify system behavior and reaction in case of failure occurrence (by injection fault). Interactive simulation allows us to look at the consequence of each failure event in the architecture model (icons or textual updated to reflect the current state).

In test case one, the FCC1 sent a fault command to actuator nodes: undetected erroneous functioning event is trigged. Simulation shows that FCC1 failure has no influence in the surface control since the vote masks the faulty value and delivers the correct one with an negative acknowledgment to faulty FCC as shown in Figure 6.

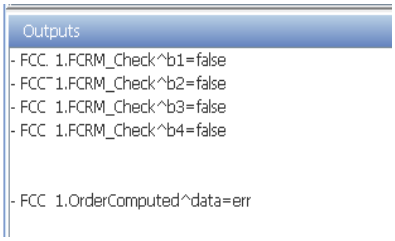

Figure 6. FCC Textual Simulation Result

Secondly we use automatic simulation to search MSS (minimum size sequence) or MCS (minimum cut sets) for event leading to FC for exhaustive validation.

The process for automatic simulation is as follows. First, the analyst defines all unsafe situations (called Failure Condition: FC) and associate a classification (minor, major, hazardous or catastrophic) and safety requirements (qualitative and quantitative). Then, he models the FC with a specific component (called observatory) integrated to the architecture model. Finally, SDT tool searches automatically all minimal combinations of failures leading to a given FC and compute the probability of FC. Architecture is valid only if all FC requirements are met. The result of automatic simulation for the "FC = Loss of both elevator control" must be less than 10^{-9} per flight hour, and it is shown in Figure 7.

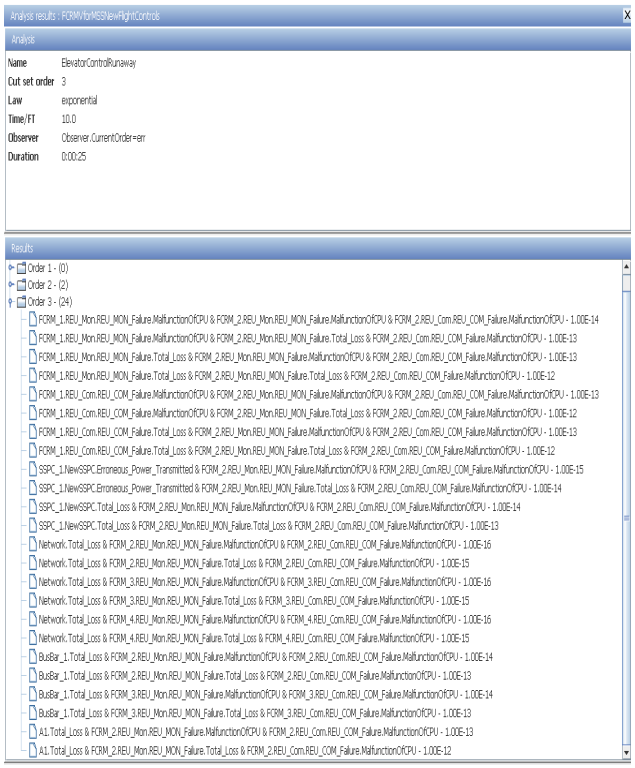

Figure 7. Loss of Both Elevator Control

Conclusions

Distributed FBW systems are the last step in the evolution of the traditional airplane FCS architectures. The evolution of microelectronic and communication technologies will continue to have an extreme influence on the FCS architecture.

This paper has shown the way that one could use so that using digital communication and smart actuators can eliminate the centralized architecture and reduce the number of centralized computers to achieve low cost that is vital for new aircraft without dependability degradation. Digitally distributed FBW architectures offer many improvements over centralized architecture. They can help to reduce redundancy and the complexity of principal computing elements in FCS architecture through the migration of computation functionality out of the FCC and the integration of smart subsystems. The processing tasks realized by central flight computers are also simplified, so that critical safety computing can now be more easily accomplished by low cost standard computing resources like IMA [22] or COTS (Commercial Off-The-Self) [23].

References

[1] Brajou, F. and P. Ricco, 2004, The Airbus A380 - An AFDX-Based Flight Test Computer Concept, in Proceedings of the 2004 IEEE AUTOTESTCON, San-Antonio, Texas, USA, September 20-23, pp. 460-465.

[2] Favre, C., 1994, "Fly-By-Wire for Commercial Aircraft: The Airbus Experience, in International Journal of Control, vol. 59, issue 1, January, pp. 139-157.

[3] Godo, E.L., 2002, Flight Control System with Remote Electronics, in Proceedings of the 21st Digital Avionics Systems Conference (DASC 2002), vol. 2, Irvine, California, October 27-31, pp. 13B1-1 - 13B1-7.

[4] Ahlstrom K. and J. Torin, 2002, Future Architecture of Flight Control Systems, in IEEE Aerospace and Electronic Systems Magazine, vol.17, Issue 12, December, pp. 21-27.

[5] Knight, J.C., 2002, Safety Critical Systems: Challenges and Directions, in Proceedings of the 24th International Conference on Software Engineering (ICSE 2002), Orlando, Florida, USA, May 19-25, pp. 557-550.

[6] Traverse, P., I. Lacaze and J. Souyris, 2004, Airbus Fly-By-Wire: A Total Approach to Dependability, in Proceedings of the 18th IFIP World Computer Congress (WCC 2004), Building the Information Society, Kluwer Academic Publishers, Toulouse, France, August 22-27, pp. 191-212.

[7] Brière, D. and P. Traverse, 1993, Airbus A320/A330/A340 Electrical Flight Controls – A Family of Fault-Tolerant Systems, in Proceedings of the 23rd IEEE International Symposium on Fault-Tolerant Computing (FTCS-23), Toulouse, France, June 22-24, pp. 616-623.

[8] Yeh, Y.C., 1996, Triple-Triple Redundant 777 Primary Flight Computer, in Proceedings of the IEEE Aerospace Applications Conference, Aspen, CO, USA, February 3-10, pp. 293-307.

[9] Yeh, Y.C., 2001, Safety Critical Avionics for the 777 Primary Flight Controls System, in Proceedings of the 20th Conf. on Digital Avionics Systems (DASC 2001), Daytona Beach, FL, USA, October 14-18, 2001, pp. 1C2/1.1C2/11.

[10] Avizienis, A., J.C. Laprie, B. Randell and C. Landwehr, 2004, "Basic Concepts and Taxonomy of Dependable and Secure computing, in IEEE Transactions on Dependable and Secure Computing, vol. 1, issue 1, Jan.-March 2004, pp. 11-33.

[11] ARP-4754/ED-79, 1996-97, Certification Considerations for Highly-Integrated or Complex Systems, published by SAE (Society of Automotive Engineers) no. ARP-4754, November 1996 and EUROCAE no. ED-79, April 1997.

[12] FAR/JAR 25, Airworthiness Standards: Transport Category Airplane, published by FAA, title 14, part 25, and Certification Specifications for Large Aeroplanes, published by EASA (former JAA), CS-25.

[13] DO-178B/ED-12, 1992, Software Considerations in Airborne Systems and Equipment Certification, published by RTCA, no. DO-178B, and EUROCAE no. ED-12.

[14] FAA (Federal Aviation Administration), 2000, System Safety Handbook, chapter 3:Principles of System Safety, December 30, 19 p.

[15] DO-254/ED-80, 2000, Design Assurance Guidance for Airborne Electronic Hardware, published by RTCA no. DO-254, and EUROCAE, no. ED-80, April.

[16] ARP-4671, 1996, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, published by SAE (Society of Automotive Engineers), December.

[17] Sghairi, M., A. de Bonneval, Y. Crouzet, J-J. Aubert and P. Brot, 2009, Challenges in Building Fault-Tolerant Flight Control System for a Civil Aircraft, in IAENG International Journal of Computer Science, vol. 35, n°4, January, pp. 495-499.

[18] Namazi A. and M. Nourani, 2007, Distributed Voting for Fault-Tolerant Nanoscale Systems, in Proceedings of 25th International Conference on Computer Design (ICCD 2007), Lake Tahoe, California, USA, October 7-10, 2007, pp. 563-573.

[19] Hardekopf B., K. Kwiat and S. Upadhyaya, 2001, Secure and Fault-Tolerant Voting in Distributed Systems, in Proceedings of 2001 IEEE Aerospace Conference (volume 3), Big Sky, Montana, USA, March 10-17, pp. 3/1117 - 3/1126.

[20] Bernard R., 2009, AltaRica Refinement to Support Safety Analyses, http://www.onera.fr/theses/journeesdesthes es/tis/actes/articles/jdt-tis-2009-article-bernard-romain.pdf

[21] Bieber P., C. Bougnol, C. Castel, J.-P. Heckmann, C. Kehren, S. Metge and C. Seguin, 2004, Safety Assessment with AltaRica - Lessons Learnt Based on Two Aircraft System Studies, in Proceedings of 18th World Computer Congress (WCC 2004), Building the Information Society, Kluwer Academic Publishers, Toulouse, France, August 22-27, 2004, pp. 505-510.

[22] Prisaznuk, P.J., 1992, Integrated Modular Avionics, in Proceedings of the IEEE National Aerospace and Engineering Conference (NAECON 1992), Dayton, Ohio, USA, May 18-22, pp. 39-45.

[23] Arlat, J., J.-P. Blanquart, T. Boyer, Y. Crouzet, M.-H. Durand, J.-C Fabre, M. Founau, M. Kaaniche, K. Kanoun, P. Le Meur, C. Mazet, D. Powell, F. Scheerens, P. Thévenod-Fosse and H. Waeselynck, 2000, Composants logiciels et sûreté de fonctionnement - Intégration de COTS, Hermès Science Publications, Paris, 2000, 158 p.

Email Addresses

Manel Sghairi: manel.sghairi@airbus.com
Jean-Jacques Aubert: jean-jacques.aubert@airbus.com
Patrice Brot: patrice.brot@airbus.com
Agnan de Bonneval: agnan.debonneval@laas.fr
Yves Crouzet: yves.crouzet@laas.fr
Youssef Laarouchi: youssef.laarouchi@laas.fr

28th Digital Avionics Systems Conference
October 25-29, 2009