Case Study: Safety Analysis of the NASA/Boeing Fly-By-Light Airplane Using a New Reliability Tool

Michael L. Ulrey • Boeing Defense & Space Group • Seattle
Daniel L. Palumbo • NASA Langley Research Center • Hampton
David M. Nicol • College of William and Mary • Williamsburg

Key Words: FMEA, Modeling & Simulation Methods, Software Tools:R&M and Safety

SUMMARY & CONCLUSIONS

This paper is a follow-on to a previous paper (Ref. 1), in which a new reliability tool called Reliability Performance Module (RPM) was described. Since that time, RPM has been used to perform a safety analysis trade study of the Primary Flight Control System (PFCS) architectures proposed for the joint NASA/Boeing Fly-By-Light/Power-By-Wire (FBL/PBW) airplane.

The purposes of this paper are to

- relate experiences and lessons learned from using RPM on the FBL/PBW safety analysis,
- present the results of the safety analysis.

The conclusions are that

- RPM is a powerful tool for performing reliability analyses of complex systems,
- the proposed FBL/PBW PFCS architectures meet all of the safety requirements except for one related to spoilers.

1. INTRODUCTION

In the following sections, we will describe the FBL/PBW PFCS architectures and the goals and scope of the study. We will review the important features of the RPM tool and how it was applied to this problem. We will discuus two modelling paradigms and compare their advantages and disadvantages. Finally, we will present the results of the study.

2. THE FLY-BY-LIGHT/POWER-BY-WIRE PROGRAM

The goal of NASA's FBL/PBW program is to demonstrate the technical feasibility and economic benefit of optical components and electrically powered actuators on civil transport aircraft. The Boeing Commercial Airplane Group has been contracted by NASA to design a modern FBL/PBW flight control system which meets current industry safety, reliability and performance requirements.

3. FBL/PBW PFCS STUDY OBJECTIVES & REQUIREMENTS

In the safety analysis of the FBL/PBW airplane PFCS, two candidate architectures were analyzed, the *baseline architecture* and the *final architecture.* Safety analyses of several *minimum dispatch configurations* of the final architecture were also performed. The purpose was to determine the robustness of the architecture in the presence of certain known failures at the start of the flight.

The objectives were to determine:

- Whether or not the baseline architecture meets the safety requirements,
- Whether or not the final architecture meets the safety requirements,
- The relative safety of the two architectures, and
- The safety of the final architecture given certain known failures.

The failure conditions studied were *loss of control* (LOC) and *hardover* (HO) of a single surface, either an aileron, an elevator, a spoiler or the rudder. Loss of control of a surface means that the surface assumes a position other than the one commanded, assumed to be relatively benign. Hardover is a more serious situation in which the surface assumes a maximum deflection position. The requirements for the various failure conditions are shown in Table 3.

In addition to the "nominal" cases, in which the airplane was assumed to dispatch in a full-up configuration, the LOC and HO cases were studied when dispatch occurred with either the left Primary Flight Computer (PFC) OFF, the right Interface Unit (IFU) OFF, or both. The OFF state indicates a non-operative but benign condition. The requirements for the various failure conditions, given the known failures, are shown in Table 5. The numbers in both Tables 3 and 5 are

0-7803-3112-5/96/$5.00 © 1996 IEEE

probabilities that the specified event occurs before the specified time.

4. THE FBL/PBW PFCS ARCHITECTURE MODELS

Schematics of the baseline and final architectures are shown in Fig. 1 and Fig. 2, respectively. In the interest of simplicity, only the rudder surface actuators are shown. A main feature of the architectures are their fiber optic busses. These busses are implemented via fiber optic star couplers.

The baseline architecture utilizes two distinct busses. A triplicated ARINC 629 bus serves the Primary Flight Computers (PFCs), the Air Data and Inertial Unit (ADIRU) and the Secondary Attitude and Air (data) Reference Unit (SAARU). The Interface Units (IFUs) act as a bridge between the 629 bus and a triplicated 1773 bus, which serves the Power Control Units (PCUs, or actuators). The IFUs also connect the pilot command sensors to the PFCs. The final architecture uses one triplicated 629 bus to connect the entire system.

Tables 1 and 2 list the numbers of each of the subsystems as well as the number of transmitters, receivers and connectors needed to implement the bus architectures. Note the large number of connectors. Since one of the objectives of the study was to determine the effect of optical connectors on the overall system reliability, all of them were modelled. However, large subgroups of them were in series and could be modelled as a single component. This reduced the number of components (or more correctly, single-step transitions, since some components have more than two states) to about 300 for the baseline architecture and about 150 for the final architecture.

Since the final architecture has fewer components, it is a less expensive alternative, especially since there is only one bus type (629) instead of two.

5. THE RELIABILITY TOOL RPM

To understand some of the modelling issues which are discussed in this paper, it is important to have a basic understanding of RPM. For a more thorough discussion, see Ref. 1.

RPM has an *executive,* which controls the activities of stimulating the *behavioral model,* and then observing the reaction. It also performs all probability calculations. The behavioral model is a BONeS representation of the system to be analyzed, produced by the design engineer using the BONeS graphical modelling capabilities. This model includes (constant) failure rates associated with component failures and recovery processes.

The construction of the behavioral model is the responsibility of the user, but once the model is accepted as valid and error-free, the executive performs a complete reliability analysis automatically.

The executive generates all possible sequences of failures, such as ("Left PFC FAILs") or ("Rudder PCU 1 FAILs", "Left 629 Star Coupler Turns OFF"), or ("Right IFU FAILs", "Pilot's Column Postion Sensor 1 Turns OFF", "Rudder PCU 2 Receiver Turns OFF"). Order is important, so that sequence dependencies are captured. The user can limit the failure space search by specifying either an upper bound on the number of component failures or a lower bound on the probability of a failure path. This is called *pruning.*

After the generation of each such component failure sequence, the executive inquires about the state of the behavioral model. If the model is in any of the specified failure states, the overall system failure probability is updated. If the system is alive but the pruning threshold has been crossed, an overall (conservative) pruning bound is updated. After each of these cases, another sequence is generated. If the system is still alive and the pruning threshold has not been reached, the executive adjoins another transition to the current failure path. This process continues until there are no more sequences to try.

6. THE BEHAVIORAL MODELS

The extensive modelling and simulation capabilities of BONeS allow one to experiment with various ways to model system behavior, fault propagation, and system health monitoring. We will describe and discuss two behavioral models which were proposed for the FBL/PBW architectures. For convenience, they will be dubbed the *signal model* and the *failure behavior model.*

7. THE SIGNAL MODEL

The BONeS block diagram of the signal model looks very similar to a schematic of the system, for example, resembling Fig. 1 or Fig. 2. The components and signals of this model are defined to mimic behavior of the corresponding actual components and signals. Fault propagation is accomplished by signals changing state as they pass through components which have failed. Health monitoring is done by checking the aggregate status of signals which have passed completely through the system, from pilot and sensor inputs all the way to actuator output.

There are certain complexities in the real system which are simplified or approximated in the model. For example, rather than represent exactly all of the possible analog and digital signals, with their various formats, flags, error checking bits, etc. we create a single signal object, which has just two fields. The first field is the *type,* such as sensor input, pilot command, PFC command, etc. The second field is the *error status,* which can be GOOD, PASSIVE, or ACTIVE. PASSIVE indicates that the signal is in error but a

validity flag has been set which is readable by "intelligent" components such as the PFC's. In other words, this is a detected error. ACTIVE inicates an undetected error, that is, the signal is in error but the validity flag still reads valid.

Similarly, the behavior of the components is simplified by viewing them as state machines. Some can be GOOD or OFF, while others can be GOOD, FAILED or OFF. The OFF state is a benign, non-operative state, while the FAILED state indicates that the component produces malicious output.

In the real system, the redundancy management system includes monitors, signal selection and fault detection algorithms, error detection and correction codes, transmitter inhibits, etc. which determine the output of a component as a function of the input signals and the component state.

The purpose of the redundancy management in the real system is to produce acceptable performance in the presence of certain combinations of detected and undetected failures. In order to represent this situation without modelling all of the minute details of the redundancy management hardware and software, we associate a table with each component (or sub-system) which specifies the state of the output given the states of the input signal(s) and the state of the component (or sub-system).

The behavior of individual components or sub-systems is thus simplified as follows. The output of any component is PASSIVE if the component is OFF and ACTIVE if the component is FAILED. If the component is GOOD, then the output is determined by the state(s) of the input signal(s).

There are two kinds of logic used in this case, which we will call "monitor" logic and "voter" logic. In the first case, if there are three input signals, the component "monitors" a primary signal, accepts and uses it if valid. If not valid, it goes to the secondary signal. If both are invalid, it goes to a tertiary signal. Of course, a signal with an undetected error will be wrongly accepted. In the second case the output is simply a function of the *numbers* of GOOD, PASSIVE and ACTIVE inputs, regardless of order, similar to voting.

This completes the description of *local* behavior in the signal model. We now turn to the *global* behavior.

After the executive has failed a selected set of components, two signals are injected into the behavioral model. The first is a *sensor signal* and the second is the *pilot input signal.* Refer to figures 1 and 2 for the following discussion.

The sensor signal is transmitted first to the inputs of the ADIRU and the SAARU. The ADIRU in turn produces two output signals, one on the left bus and one on the right bus, while the SAARU produces one output signal on the center bus. These signals are sent to the PFC's, and each PFC processes these signals, thus producing air and inertial reference data which is GOOD, PASSIVE or ACTIVE based on these inputs and the states of the corresponding PFC's. This data is saved to be combined with the pilot command input when it arrives.

Next, the pilot command input signal is sent to each of the appropriate control sensors on the flight deck (i.e., column, pedal or wheel position sensors). The output of the sensors then proceeds to the IFU's, where they are processed and transmitted to the PFC's. These signals are processed by each PFC, taking into account the state of the air and inertial reference data as well as the state of the pilot input signals.

The PFC's then generate surface movement commands, which proceed to IFU's in the baseline architecture, but directly to the PCU's in the final architecture. In the baseline, after processing by the IFU's, the signals proceed to the PCU's. In either case, each PCU produces an output based on the state of the input signals and the PCU.

The outputs of the PCU's on a given surface are then collected and the state of the given surface is declared OK, LOC (loss of control) or HO (hardover) by using rules similar to the voter logic for determining a component output signal state.

Suppose, for example, that the condition being studied is loss of control (LOC) of the rudder. Assume that the executive causes the PCU on the low rudder actuator (connected to the center 629 bus) to go from GOOD to FAILED and the left 629 bus star coupler to go from GOOD to OFF.

The result is that the output of the PCU on the low rudder actuator is ACTIVE, the output of the PCU on the mid rudder actuator is OFF and the output of the PCU on the high rudder actuator is GOOD. In this case, the surface condition would be declared LOC, since presumably the GOOD and ACTIVE actuators would cancel each other out, leaving the condition to be determined by the remaining PASSIVE actuator, which means that control of the surface has been lost, but it has not gone hardover.

The executive then computes the probability corresponding to the failure sequence ("Low rudder actuator PCU FAILED", "Center 629 bus star coupler OFF") which produced this condition, and this is used to update the probability of LOC of the rudder.

8. THE FAILURE BEHAVIOR MODEL

The failure behavior model, like the signal model, is represented as a block diagram in BONeS. The models are in

fact identical at the system and sub-system level. Differences between the two models occur only in the lowest functional level of the model. The major difference is in how signal packets are generated and propagated.

Recall that in the signal model, the system global condition is evaluated by simulating the sensor and command signal traffic. The arrival of a signal packet stimulates the local evaluation of the sub-system failure condition.

In a failure behavior model, failure condition messages are only created and transmitted when a sub-system undergoes a state change. The executive begins the analysis by stimulating a sub-system failure. This causes a failure description packet to be generated and injected into the system's signal paths, where it is communicated to and interpreted by other sub-systems.

The receiving sub-systems are then stimulated to respond to the failure condition in much the same way as they do in the signal model. A notable exception is that if the received failure message does not cause an observable sub-system state change, the sub-system does not propagate the message. An example of this would be when a voter module receives an ERROR signal which is effectively masked. In this way, only deviations from the current system state are reported. The failure simulation continues until message traffic ceases and the global system condition is thereby obtained.

The failure behavior paradigm is descended from the automated FMEA process that has been described in Refs. 2, 3 and 4. It is designed to mimic the actions of an analyst in propagating a component failure condition to a global system response. In this modelling paradigm, the analyst defines local behavior in a block structured environment. The executive produces the global system effects automatically.

It has been observed (Refs. 2, 4) that by focussing the analyst's attention to one component or sub-system at a time, fewer analysis errors are made. Encapsulation of sub-system behavior also allows automated consistency checks that would otherwise be impossible. This result is analogous to that obtained in modular software design.

9. COMPARISON OF SIGNAL AND FAILURE BEHAVIOR MODELS

The failure behavior model is computationally more efficient than the signal model, since it simulates the behavior of only those portions of the model that are affected by the failure. The signal model uses a lot of processing time sending signals through a large part of the system which is unaffected by the failure.

One of the advantages of the signal model is that it more closely resembles how the system functions. This can be an advantage when changes are made to the original design and a component's functional dependence changes.

An example of this was observed in the modelling of connectors. Using the failure behavior paradigm, a connector was modelled to send an "ERROR" message when the connector failed open. This is clearly not the correct behavior if, for example, the connector is on an unused leg of the star coupler. While this design dependence can be modelled (e.g., with a USED/UNUSED state variable), it is better to rely instead on the system operational simulation to infer this result. This will be most valuable in more complicated and less obvious situations.

Also, one of our goals is to integrate timing and reliability analysis capabilities in RPM. It seems a natural step from the signal model to a model which includes timing delays of the signals. That is, in addition to the current criteria for system failure based on the arrival of certain signals at the outputs, one can devise criteria for the *timely* arrival of such signals, then run the BONeS simulation to see if those criteria are met when some components have failed.

10. FBL SAFETY ANALYSIS RESULTS

Both architectures met all of the requirements shown in Table 3 except for the spoiler LOC requirements. This is due to the fact that each spoiler has but one PCU. The results for the two architectures were very close. However, the final architecture met each requirement by a wider margin than did the baseline.

See Table 4 for an example listing of dominant failure sequences for loss of control of the left elevator in the final architecture This table shows the top 20 events in order of decreasing probability. The reason that some events seem to be listed twice (with the order of the failures reversed) is that RPM respects the order of failures in its search. This is the basis of its ability to find sequence dependencies, although there are none evident in this table.

Of course, the program searches thousands (perhaps millions, depending on user-chosen pruning instructions) of such sequences in its computation of the overall system failure probability.

The minimum dispatch configurations studied were left PFC OFF, right IFU OFF, and both. The RPM tool made this aspect of the study very straightforward. For example, to do the left PFC OFF case, all that was needed was to set the state machine of the left PFC permanently OFF, and then run the analyses as before. Similarly for the other cases.

The results of the minimum dispatch configuration study are shown in Table 5. Note that there is only slight degradation in the 1-hour flight safety for loss of control given a PFC OFF, IFU OFF, or even both a PFC and IFU

OFF prior to start of flight. Note also that in the case of hardovers, the effect is not even visible.

Finally, RPM performance is documented in Table 6, which shows results corresponding to the rudder hardover case in the final architecture. The probability upper bounds include an estimate on the probability of pruned paths. The pruning bound gets progressively smaller as longer failure paths are explicitly generated. When going from 2-component failure sequences to 3-component failure sequences, for example, the decrease in the prune bound is greater in absolute magnitude than the increase in failure probability due to explicit consideration of more paths. This explains the drop in the upper bound as longer failure sequences are considered. Also, the execution time to compute a pruning bound is trivial compared to the time needed to determine the system condition after each and every failure sequence.

This table illustrates the tradeoff between execution time and tightness of the probability upper bound. For example, there is quite a jump in the execution time from considering all 2-component failures to considering all 3-component failures. However, the resulting decrease in the probability upper bound is small, and the bound corresponding to the 2-component failure level was sufficient to show compliance with the rudder hardover requirement. (see Table 3).

In fact, computation of bounds corresponding to component failure level 2 was sufficient to show compliance with the safety requirements in all cases.

However, in the minimum dispatch cases (see Table 5), calculations in all of the hardover cases were taken to the 3-component failure level in order to make clear any deviations of the non-nominal cases from the nominal one. It turned out that the differences were about 3 or 4 orders of magnitude too small to make a difference in the numbers shown in Table 5.

ACKNOWLEDGMENT

We are pleased to thank Sheldon Gerson of the Boeing Commercial Airplane Group for his guidance and advice on this project.

REFERENCES

1. "A Graphical Model-Based Reliability Estimation Tools & Failure Mode and Effects Simulator", D. M. Nicol, D. L. Palumbo & M. L. Ulrey, ***Proceedings of the 1995 Reliability & Maintainability Symposium.***

2. D.L. Palumbo, D.M. Nicol, "Advanced Techniques in Reliability Model Representation and Solution", *NASA Technical Paper 3242*, October, 1992.

3. D.L. Palumbo, "Automating Failure Modes and Effects Analysis", ***Annual Reliability and Maintainability Symposium***, 1994.

4. D.L. Palumbo, "Using Failure Modes and Effects Simulation as a Means of Reliability Analysis", ***11th Digital Avionics Systems Conference***, 1992.

BIOGRAPHIES

David M. Nicol
Department of Computer Science
College of William and Mary
P.O. Box 8795
Williamsburg, VA 23187-8795 USA
email:nicol@cs.wm.edu
Company Name: College of William and Mary

David M. Nicol received the B.A. degree in mathematics from Carleton College (1979), and the Ph.D. in computer science from the University of Virginia (1985). He is presently an associate professor of computer science at the College of William and Mary. He is associate editor for the ACM Transactions on Modeling and Computer Simulation, and for the ORSA Journal on Computing. He is on the steering committee for the Workshop on Parallel and Distributed Simulation, and has served as program chair, and also general chair for that Workshop's annual conference. He is widely published in the areas of performance modeling, parallel processing, and simulation.

Daniel L. Palumbo
MS 130
NASA Langley Research Center
Hampton, VA 23665-5225 USA
email:dlp@air12.larc.nasa.gov
Company Name: NASA Langley Research Center

Dan Palumbo received his BSEE and MSEE degrees from Rensselaer Polytechnic Institute in 1972 and 1973, respectively. H e is a Senior Computer Engineer with NASA Langley Research Center where his current interests are related to the deployment of distributed and integrated flight control systems.

Michael L. Ulrey
The Boeing Company, M/S 4C-72
P.O. Box 3707
Seattle, WA 98124 USA
Phone: (206)-655-3881
FAX: (206)-655-4018)
email: ulrey@kenyon.ds.boeing.com
Company Name: Boeing Defense & Space Group

Mike Ulrey is a Principal Engineer in the Advanced Vehicle Management Systems organization of the Boeing Defense & Space Group. He is currently responsible for the development of the RPM tool, to be included in an integrated design process for rapid prototyping of vehicle management systems. He has been involved in large applications software development projects for NCR, TRW, Raytheon and Boeing. He taught mathematics at Wichita State University for 6 years. He received a Ph.D. degree in mathematics from Ohio State University in 1973. He is a member of SIAM and IEEE.

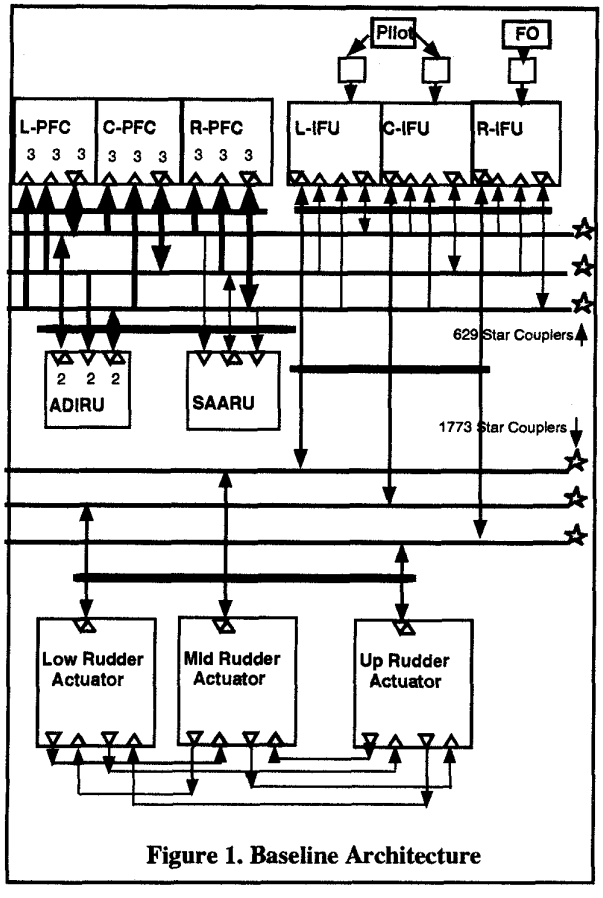
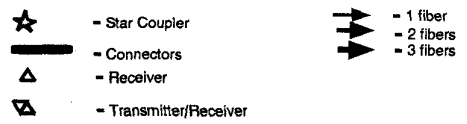


Figure 1. Baseline Architecture



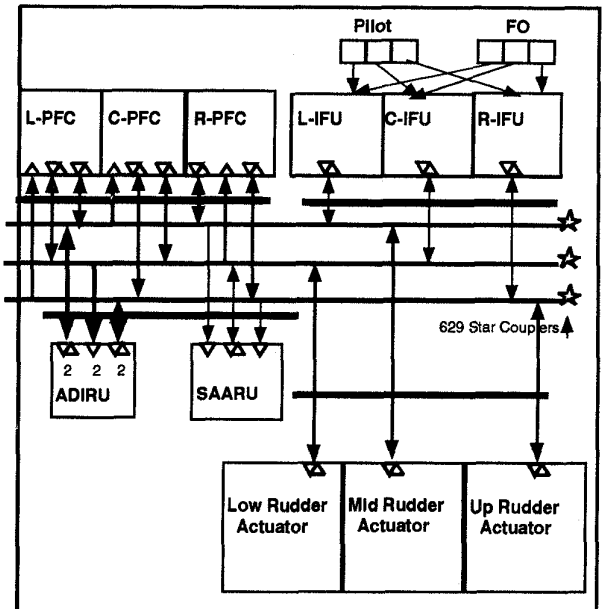


Figure 2. Final Architecture

Item	Number	Tx	Rx	629 Connections	1773 & Point-to-Point Connections
PFC	3	9	27	144	0
IFU	3	6	12	48	24
ADIRU	1	4	6	40	0
SAARU	1	1	3	20	0
Ail PCU	4	8	8	0	32
Elev PCU	6	18	18	0	84
Rud PCU	3	9	9	0	30
Spoil PCU	12	12	12	0	96
Tx	67	–	–	--	–
Rx	95	–	–	--	–
Flight Deck Sensors	11	–	–	--	–
Star Couplers	6	–	–	--	–
Connections	518	–	–	--	–

Table 1. Component Counts (Baseline Architecture)

Item	Number	Tx	Rx	629 Connections
PFC	3	6	9	60
IFU	3	3	3	24
ADIRU	1	4	6	40
SAARU	1	1	3	20
Ail PCU	4	4	4	32
Elev PCU	6	6	6	84
Rud PCU	3	3	3	30
Spoil PCU	12	12	12	96
Tx	39	--	--	--
Rx	46	--	--	--
Flight Deck Sensors	18	--	--	--
Star Couplers	3	--	--	--
Connections	386	--	--	--

Table 2. Component Counts (Final Architecture)

Surface/ Condition	Requirement (Probability in 1-hr Flight)	Baseline	Final
Aileron LOC	3.0e-7	< 2.47e-8	< 2.38e-8
Aileron HO	1.0e-9	< 3.22e-10	< 8.94e-11
Elevator LOC	3.0e-7	< 4.65e-10	< 2.31e-10
Elevator HO	1.0e-9	< 3.59e-10	< 8.82e-11
Rudder LOC	1.0e-9	< 4.67e-10	< 2.3e-10
Rudder HO	1.0e-9	< 3.59e-10	< 8.82e-11
Spoiler LOC	3.0e-5	< 6.25e-5	< 5.98e-5
Spoiler HO	1.0e-6	< 1.1e-8	< 1.0e-8

Table 3. Summary of Results Baseline vs. Final Architecture

- ADIRU (GOOD to FAILED)
- SAARU (GOOD to OFF) & ADIRU (GOOD to OFF)
- ADIRU (GOOD to OFF) & SAARU (GOOD to OFF)
- Center 629 Bus (GOOD to OFF) & ADIRU (GOOD to OFF)
- ADIRU (GOOD to OFF) & Center 629 Bus (GOOD to OFF)
- Elevator PCU1 (GOOD to OFF) & Elevator PCU2 (GOOD to FAILED)
- Elevator PCU1 (GOOD to FAILED) & Elevator PCU2 (GOOD to OFF)
- Elevator PCU1 (GOOD to OFF) & Elevator PCU3 (GOOD to FAILED)
- Elevator PCU1 (GOOD to FAILED) & Elevator PCU3 (GOOD to OFF)
- Elevator PCU2 (GOOD to OFF) & Elevator PCU3 (GOOD to FAILED)
- Elevator PCU2 (GOOD to FAILED) & Elevator PCU3 (GOOD to OFF)
- Elevator PCU2 (GOOD to OFF) & Elevator PCU1 (GOOD to FAILED)
- Elevator PCU2 (GOOD to FAILED) & Elevator PCU1 (GOOD to OFF)

Table 4. Primary causes of loss of control of left elevator in Final Architecture

Surface	Condition	Requirement	Nominal	L PFC OFF	R IFU OFF	Both OFF
Right Aileron	LOC	3e-7	<2.38e-8	<2.49e-8	<2.51e-8	<2.54e-8
Right Aileron	HO	1e-7	<8.94e-11	<8.94e-11	<8.94e-11	<8.94e-11
Left Elevator	LOC	3e-7	<2.31e-10	<1.34e-9	<7.63e-10	<1.87e-9
Left Elevator	HO	1e-7	<8.82e-11	<8.82e-11	<8.82e-11	<8.82e-11
Rudder	LOC	1e-7	<2.31e-10	<1.34e-9	<7.62e-10	<1.87e-9
Rudder	HO	1e-7	<8.82e-11	<8.82e-11	<8.82e-11	<8.82e-11

Table 5. Minimum Dispatch Results

Component Failure Level	Number of Paths Searched	Number of Failure Paths Found	Number of Paths Pruned	Upper Bound on Probability	Execution Time (3 Sparc 10's)
1	143	1	142	2.2e-7	2 sec
2	20,274	156	19,976	1.2e-10	227 sec
3	2,827,682	30,230	2,777,334	8.8e-11	33,363 sec

Table 6. Computing Effort vs. Tightness of Upper Bound for Rudder Hardover Condition (Final Architecture)