GRANT / AMES
76P.

IN-18306

INVESTIGATION OF AN ADVANCED FAULT TOLERANT INTEGRATED AVIONICS SYSTEM

FINAL TECHNICAL REPORT
covering the period
November 1983 - March 1986

School of Applied Science and Engineering Technology

(NASA-CR-176980) INVESTIGATION OF AN ADVANCED FAULT TOLERANT INTEGRATED AVIONICS SYSTEM Final Technical Report, Nov. 1983 - Mar. 1986 (University of Southern Colorado, Pueblo.) 76 p CSCL 01D G3/06

N86-28945

Unclas
43449

UNIVERSITY OF SOUTHERN COLORADO
2200 North Bonforte Boulevard Pueblo, Colorado 81001

NASA COOPERATIVE AGREEMENT NCC 2-277

INVESTIGATION OF AN ADVANCED FAULT TOLERANT INTEGRATED AVIONICS SYSTEM

FINAL TECHNICAL REPORT
covering the period
November 1983 - March 1986

Principal Investigator: Dr. W.R. Dunn
Co-investigator: Dr. D. Cottrell

Research Assistants: Joel Flanders
Alan Javornik
Michael Rusovick

Institution: University of Southern Colorado
Electronics Engineering Dept.
2200 N. Bonforte Blvd.
Pueblo, Colorado 81001

TABLE OF CONTENTS

Section	Contents	Page
	Tables	1
	Figures	2
	Notation	3
	Report Summary	4
1.0	INTRODUCTION: REPORT OUTLINE AND OBJECTIVES	5
2.0	AMAP IN THE ADVANCED ROTORCRAFT APPLICATION	7
2.1	Introduction	7
2.2	Advanced Rotorcraft Digital System	7
2.2.1	AMAP in the Integrated Digital System	7
2.2.2	AMAP Components	12
3.0	AMAP FAULT TOLERANCE REQUIREMENTS AND RELIABILITY GOALS	14
3.1	Introduction - Basic Definitions and Concepts	14
3.2	Fault Tolerant Digital Systems Design - Overview	14
3.3	Reliability Models and Goals	16
3.3.1	Reliability Models	16
3.3.2	Reliability Goals and Estimates	17
4.0	AMAP ARCHITECTURE TRADEOFF STUDY	19
4.1	Simplex System Reliability Analysis	19
4.2	Redundant AMAP Architectures - Theoretical	19
4.3	Standby Redundancy for AMAP/SANDAC IV	25
4.3.1	Diversity of Module Types	25
4.3.2	Register/Memory Reconfiguration	25

4.3.3 Flight Safety Fault Tolerance 26

4.3.4 Additional Hardware Overhead 26

4.3.4.1 Bus Redundancy 27

4.3.4.2 Power Distribution Faults 27

4.3.4.3 Power Supply Overhead 28

4.3.5 Additional Software overhead 28

4.4 Standby vs. Dual-Redundancy for AMAP/SANDAC IV 28

4.4.1 Introduction 28

4.4.2 Static vs. Dynamic Redundancy Management Requirements 30

4.5 Tradeoff Study Conclusion 32

5.0 HARDWARE AND SOFTWARE DESIGN CONSIDERATIONS FOR AMAP/SANDAC IV DUAL ARCHITECTURE 34

5.1 Introduction 34

5.2 Redundancy Management-System Level 34

5.2.1 System Redundant Architecture 35

5.2.2 System Redundancy Management 37

5.2.2.1 Authority Hierarchy 37

5.2.2.2 Cockpit Redundancy Management 38

5.2.3 System Fault Handling 38

5.3 Redundancy Management-AMAP/SANDAC IV 40

5.3.1 General Considerations 40

5.3.1.1 Fault Mechanisms and Effects 40

5.3.1.2 Fault Detection Strategy 41

5.3.1.3 Hardware vs. Software Redundancy Management 42

5.3.2 Built-in-Tests 42

5.3.3 Deadline Mechanisms 44

5.3.4 Software Assertions 44

5.3.5 Built in Redundancy Management Functions 45

5.3.6 Predictive Task Scheduling 45

5.3.7 Wraparound Tests 46

5.3.8 Cross-channel Testing 48

5.4 Fault Handling - AMAP/SANDAC IV 50

5.4.1 Dual Processor Fault Detection and Isolation 50

5.4.2 Cockpit Fault Monitoring and Reconfiguration Logical Requirements 53

5.5 Verification and Validation of Digital System Fault Tolerance 55

5.5.1 General Considerations 55

5.5.2 Verification Design Analyses of Fault Tolerant Systems 56

5.5.2.1 Reliability Analysis 57

5.5.2.2 Failure Modes and Effects Analysis 57

5.5.2.3 Fault Tree Analysis 58

5.5.2.4 Single-Point-of-Failure Analysis 58

5.5.2.5 Analysis Limitations 59

5.5.2.6 Verfication Documentation 59

5.5.3 Fault Tolerance Validation Testing 59

5.5.3.1 General Considerations 59

5.5.3.2 Hardware Fault Insertion 62

5.5.3.3 Resident Fault Simulation 62

5.5.4 Elimination of Man-Made Faults 65

6.0 CONCLUSION 67

REFERENCES 68

APPENDIX 69

TABLES

	TITLE	PAGE NO.
2-1	Major AMAP Functions	11
2-2	AMAP Modules for STAR Tests	13
4-1	Characteristics of Architectures of Figure 4-1	24
4-2	Characteristics of AMAP/SANDAC IV Architectures	31
5-1	Typical Built-in-Test Functions	43
5-2	Synchronous vs. Asynchronous Systems	49
5-3	Summary of Self-Test Methods	51
5-4	Typical Single Points of Failure	60

FIGURES

FIGURE		PAGE NO.
2-1	AMAP in the Advanced Rotorcraft System	8
4-1 (a)	Simplex System	21
4-1 (b)	Dynamic Redundancy with Single Spare	22
4-1 (c)	Static Dual Redundancy	23
4-2	Dual-bus/Dual-buffer implementation	23-A
5-1	Dual Processor Channels in Advanced Rotorcraft System	36
5-2	Authority Hierarchy	39
5-3	Wraparound Tests	47
5-4	Processor Fault Status	52
5-5	Cockpit/AMAP Redundancy Management Interface	54
5-6	Fault Insertion Device for Semiconductor Chips	63
5-7	Design Errors vs. Validation test time	66

NOTATION

ADAS	= Army Digital Avionics System.
ADOCS	= Advanced Digital Optical Control System.
AFTI	= Advanced Fighter Tactical Integration.
AMAP	= Army Multibus Avionics Processor.
AVRADA	= U.S. Army Avionics Research and Development Activity
BIT	= Built In Test
CBIT	= Continuous Built In Test
CPU	= Central Processing Unit.
EMI	= ElectroMagnetic Interference.
EPROM	= Erasable Programmable Read Only Memory.
FLIR	= Forward-Looking Infa-Red.
I/O	= Input/Output.
IFR	= Instrument Flight Rules.
LHX	= Army family of light helicopters for the 1990s and beyond; scout, light attack and utility.
LRU	= Line Replaceable Unit.
LSI	= Large Scale Integration.
MBIT	= Maintenance Built In Test
MTBF	= Mean Time Between Failures.
NOE	= Nap Of Earth.
PBIT	= Pre-flight Built In Test
PROM	= Programmable Read Only Memory.
RAM	= Random Access Memory.
RAMPS	= Redundant Asynchronous Microprocessor System.
REBUS	= REsident BackUp Software.
SANDAC IV	= A compact, modular microprocessor (68000) card family developed by Sandia National Laboratories.
STAR	= (Army) System Test bed for Avionics Research
VHSIC	= Very High Speed Integrated Circuits.
VLSI	= Very Large Scale Integration.
1553B	= A military standard number corresponding to a serial data bus.
68000	= Model number of a Motorola 16-bit microprocessor.

REPORT SUMMARY

This report presents an advanced, fault tolerant multiprocessor avionics architecture as could be employed in an advanced rotorcraft such as LHX.

The processor structure is designed to interface with existing digital avionics systems and concepts including the Army Digital Avionics System (ADAS) cockpit/display system, navaid and communications suites, integrated sensing suite, and the Advanced Digital Optical Control System (ADOCS).

The report defines mission, maintenance and safety-of-flight reliability goals as might be expected for an operational LHX aircraft. Based on use of a modular, compact (16-bit) microprocessor card family, results of an preliminary study examining simplex, dual and standby-sparing architectures is presented.

Given the stated constraints, it is shown that the dual architecture is best suited to meet reliability goals with minimum hardware and software overhead.

The report presents hardware and software design considerations for realizing the architecture including redundancy management requirements and techniques as well as verification and validation needs and methods.

1.0 INTRODUCTION: REPORT OUTLINE AND OBJECTIVE

This report was prepared for, the U.S. Army Avionics Research and Development Activity (AVRADA) which, during the report period is investigating an advanced computer architecture known as AMAP (Army Multibus Avionics Processor.)

The Army's AMAP development was intended to explore two concepts:

(1) the application of multiprocessing to avionics real-time data processing.

(2) Use of a compact, modular packaging scheme developed by Sandia National Laboratories called SANDAC IV.

The development of real time multiprocessing techniques is extremely important: near-future operational systems (such as LHX) pose a quantum jump in data processing requirements that will outstrip single-CPU capability. Concurrent application of the SANDAC IV packaging is intended to keep the expanded equipment requirement forced by multiprocessing into manageable equipment volumes and LRU counts.

This report develops a fault tolerant structuring of AMAP as it might be applied in an advanced application such as LHX.

Section 2 accordingly presents a top-down, baseline picture of AMAP as it might appear and function in an advanced rotorcraft system.

Section 3 explains the need for fault tolerant structuring of AMAP and states reliability goals for system maintenance and flight safety.

A tradeoff study based on candidate fault tolerant architectures and the reliability goals of section 3 is presented in section 4. This latter section also presents a candidate fault tolerant structure for AMAP (as employed in the baseline system of Section 2).

Detailed hardware and software design considerations needed to realize fault tolerant performance of this structure are discussed in Section 5.

A conclusion is presented in the final section of the report.

Although this report develops a preliminary, fault tolerant architecture for AMAP/SANDAC IV its principal purpose is to convey to the digital avionics designer/analyst the perspectives, tools and techniques leading not only to implementation of this architecture but any of its future variants.

2.0 AMAP IN THE ADVANCED ROTORCRAFT APPLICATION

2.1 Introduction

In this section, a baseline picture of AMAP, as embedded in an advanced Army rotorcraft, is developed. It is important to note that definition of digital hardware and software requirements against 1990's operational needs is in an early formative stage. The baseline rotorcraft system developed in this report is therefore based on a projected synthesis of several known Army development programs including AMAP. (These programs are discussed in more detail in the next section.) As indicated in the introduction (Section 1), a major objective of this report is not to reach a final definition of the 1990's digital system but to provide the avionics designer/analyst with some of the key perspectives and tools needed to reach this ultimate goal.

2.2 Advanced Rotorcraft Digital System

2.2.1 AMAP in the Integrated Digital System

Figure 2-1 depicts AMAP as connected to the major digital subsystems of the advanced rotorcraft. These subsystems include:

(1) Cockpit control/display system as currently being investigated in AVRADA's ADAS program (reference 1)

(2) Conventional Navaid and communications suite.

(3) Voice interactive signal processor (an ADAS cockpit-control extension)

(4) Integrated sensor and advanced communications suite (as defined for LHX in references 2, 3 and 4.)

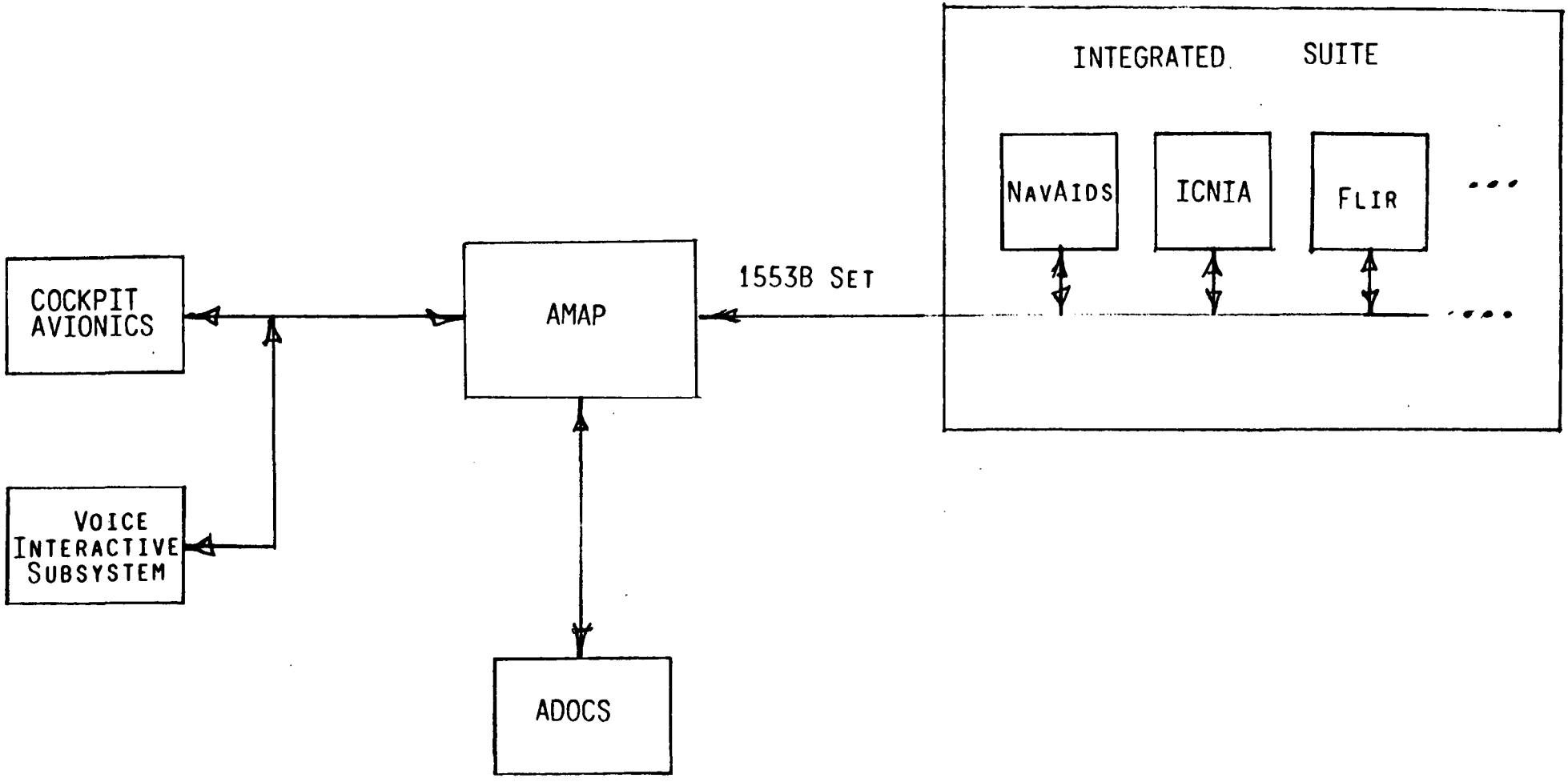


FIGURE 2-1 AMAP HARDWARE INTERCONNECT TO ADVANCED ROTORCRAFT DIGITAL SUBSYSTEMS

(5) The Army's Advanced Digital Optical Control System (ADOCS) as described in reference 5.

Functionally, cockpit elements and the AMAP processors constitute the central manager of the overall digital system of Figure 2-1. AMAP functions with regard to these subsystems (with the exception of ADOCS) include:

(1) All cockpit, flight management and navigation functions of the current (reference 1) ADAS processors.
(2) Management and support processing of the integrated sensor suite including sensor analytical redundancy management.

ADOCS is a "fly-by-wire" system interconnecting cyclic, pedal and collective cockpit controls to flight control actuators. ADOCS processors additionally provide stability and handling-qualities augmentation as well as limited maneuvering capability. Unlike AMAP, ADOCS is a flight critical system (i.e. loss of ADOCS function will most likely lead to loss of the aircraft.) In the analyses in this report it is assumed that AMAP will interface with ADOCS by:

(1) Receiving (redundant) autopilot and air data sensor for use in integrated sensor-fusion/analytical-redundancy algorithms.
(2) Transmitting limited authority (outer loop) navigation commands including:
    (a) Preprogrammed bob-up trajectories,
    (b) Memorized remask trajectories,
    (c) Conventional IFR approaches and departures (category II minimums.)

The foregoing AMAP functions are summarized in Table 2-1.

(It is noted that full authority navigation functions such as automatic NOE flight are not considered in the analyses of this report.)

TABLE 2-1

MAJOR AMAP FUNCTIONS

- COCKPIT CONTROL/DISPLAY FUNCTIONS
- NAVIGATION PROCESSING
- SUPERVISE/MANAGE VHSIC PROCESSING RELATED TO:
  - VOICE INTERACTIVE SUBSYSTEM
  - TARGET ACQUISITION/IDENTIFICATION SUBSYSTEMS
  - COMMUNICATIONS
- PROCESSOR SYSTEM REDUNDANCY MANAGEMENT
- FLIGHT CONTROL
  - PROVIDE (REDUNDANT) TRAJECTORY COMMANDS
  - PROVIDE OUTER LOOP CONTROL COMMANDS ONLY
- SENSOR/COMMUNICATIONS SUBSYSTEM REDUNDANCY MANAGEMENT

2.2.2 AMAP Components

In this report the 68000/SANDAC IV AMAP System projected for the Army's System Test Bed for Avionics Research (STAR) testing is considered. Here a simplex (i.e. non-redundant) system would contain the modules shown in Table 2-2 in the indicated quantities.

(Note: Table 2-2 does not include numeric processor or VHSIC based processor modules which would very likely be employed in the advanced rotorcraft system. Exclusion of these modules affects neither the validity of the analysis and design methods discussed in this report nor the presented conclusions.)

TABLE 2-2

AMAP MODULES FOR STAR TESTS

MODULE TYPE	QUANTITY
MASTER PROCESSOR	1
SLAVE PROCESSOR	3
1553B I/O SLAVES	3
GLOBAL MEMORY	1
STANDARD SERIAL/PARALLEL I/O	1
POWER SUPPLY	1

3.0 AMAP FAULT TOLERANCE REQUIREMENTS AND RELIABILITY GOALS

3.1 Introduction - Basic Definition and Concepts

This report addresses the fact that all digital system components are subject to physical failure. In analysing contemporary PC-card, electrical-contact-oriented rotorcraft avionics systems using the reliability analysis methods of MIL-HDBK-217, one finds that there are three primary forms of physical failure:

(1) Electrical interconnect failures such as open connector/ switch contacts, PC trace opens/bridges, open/shorted solder joints.
(2) Semiconductor device failures such as out-of-specification parameter shifts, metalization defects, and wire bonds.
(3) Discrete component failures such as opens/shorts in filter/ decoupling capacitors.

Physical failures can lead to physical fault defined as an unspecified and disruptive change in the logical function and/or of a timing digital component, assembly, subsystem, etc. Digital system faults may also arise from "man-made" faults in the form of improper specificatons, software errors, inadequate electromagnetic interference (EMI) protection, lack of understanding of thermal/ vibration environment, etc.

3.2 Fault Tolerant Digital Systems Design-overview

In the broadest sense, a fault tolerant digital system is a system which can continue to function correctly after the occurrence of (physical) faults and/or "man-made" faults. Its principal

characteristic is that it will employ additional hardware and/or software that would not be needed were the system free from faults. One would naturally seek to avoid, or at least minimize, this additional hardware and/or software overhead. Accordingly, the digital system in its non-fault tolerant form is analysed to determine the effects of faults on systems performance and reliability goals. If these goals cannot be met, fault tolerant design is then pursued by:

(1) Introducing hardware and/or software redundancy i.e. developing fault tolerant architecture(s)
(2) Designing hardware circuits and/or software algorithms that will make the architecture "work" i.e. developing redundancy management methods
(3) Evaluating the results of (1) and (2) through analysis and testing i.e. system verification and validation

As one might suspect this process is iterative, involving consideration of candidate architectures followed by analysis, consideration of modified architectures, further analysis, and so on.

Since the design activity is done against reliability goals, it is helpful to briefly discuss not only reliability goals (i.e. for AMAP) but reliability prediction models as applicable to fault tolerant system's design.

3.3 Reliability Models and Goals

3.3.1 Reliability Models

Reliability is defined as the probability that an item (e.g. component, subsystem, etc.) will perform satisfactorily for a specified period of time under a stated set of use conditions. In this report the single-parameter, exponential reliability model* is employed where,

$$R(t) = e^{-\lambda t} \qquad (2\text{-}1)$$

and

$R(t)$ = probability that item will operate without failure for time period, t (in hours)

e = base of natural logarithms

λ = item failure rate (in failures per hour), assumed to be constant for a given set of stress, temperature and part quality levels.

In this report, reliability calculations are based solely on physical failures, i.e. λ represents the physical failure rate of the hardware item.

Two companion definitions will be employed:

(1) Mean time between failures (MTBF) defined as the reciprocal of the item failure rate. I.e.,

$$\text{MTBF (item)} = 1/\lambda \qquad (2\text{-}2)$$

* This is considered to be a reasonable model for electronic components of the type employed in AMAP.

(2) Unreliability U(t), the probability of occurrence of a physical fault in an item. Here,

$$U(t) = 1-R(t) \qquad (23)$$

Note that the foregoing definitions apply to simplex (i.e. non-redundant) items. (Reliability calculations for systems employing redundant components are presented in Section 4.)

3.3.2 Reliability Goals and Estimates

The preceding subsection addressed notions of item or component reliability. This subsection discusses system reliability requirements or goals with specific consideration of AMAP reliability goals in the advanced rotorcraft application. In the next section, estimates of system reliabilities of candidate AMAP architectures will be calculated using component reliability data. This estimate will, as a result, correspond only to physical faults. I.e. it will not take into account "man-made" faults. In this sense, system reliability estimates consitute an upper bound which would be reached when all "man-made" faults are removed in system development.

Reliability goals represent the desired performance of the fielded equipment. There are three reliability goals for the advanced rotorcraft:

(1) Mission reliability

(2) Flight safety reliability

(3) Maintenance reliability

Mission reliability represents the probability that there will not be a mission abort due to failure of "mission-critical" components. Flight safety reliability corresponds to probability that aircraft and/or crew will not be lost due to failure of "flight-safety-critical" components. Maintenance reliability represents the probability that system components will not have to be replaced.

Based on the LHX study (references 2 and 3) and the ADOCS report of reference 5, the following reliability goals for AMAP are used in this report:

$$\text{Mission: } < 5 \times 10^{-5} \text{ /hr} \quad (\text{MTBF} = 20000 \text{ hrs.})$$

$$\text{Flight Safety: } < 10^{-7} \text{ /hr.}$$

$$\text{Maintenance: } < 1.5 \times 10^{-3} \text{ /hr.} \quad (\text{MTBF} = 667 \text{ hrs.})$$

4.0 AMAP ARCHITECTURE TRADEOFF STUDY

4.1 Simplex System Reliability Analysis

The appendix presents a preliminary reliability analysis for a simplex AMAP system employing the ten modules listed in Table 2-2. The analysis results show that the simplex AMAP system reliability (approx. 10^{-3} /hr.) does not satisfy LHX-level mission reliability goals (5×10^{-5} /hr.) and that a fault-tolerant design will be needed to meet the goals.

4.2 Redundant AMAP Architectures - Theoretical

AMAP is a multiple-module system. Although circuit design techniques could conceivably be invoked to realize individual, fault tolerant AMAP modules, it is far more practical to employ redundant modules. (The reasoning behind this statement will be seen in the subsequent discussion.)

In this subsection, a "first cut" is made to develop candidate redundant structures for AMAP. As it turns out, redundancy can be implemented in two ways:

(1) Dynamic Redundancy

A core of modules is supplemented with redundant hardware such that in the event of a fault, "good" hardware will be automatically substituted for the faulty hardware and correct operation continued. A well known approach for doing this involves use of stand-by-spare hardware (e.g. reference 2).

(2) Static Redundancy

Modules are simply replicated in duplex, triplex, quadruplex, etc. form. In the event of a fault, the faulted module is simply passivated and system operation taken up by the remaining, good modules. Static redundancy is employed in ADOCS (reference 5).

In this subsection, a system of n modules is considered structured in three ways:

(1) As a simplex system (to be used as basis for comparison)

(2) As a dynamically redundant system employing a single spare module.

(3) As statically redundant system in which all modules are simply duplicated

(Redundancy beyond single-sparing (dynamic redundancy) and duplication (static redundancy) are not considered since they represent "overkill" for the AMAP application.)

Figures 4-1(a) through 4-1(c) depict the above three configurations and also show equations for computing mission and maintenance reliabilities. To compare these three it is assumed that ten modules are employed and that each has a reliability of 10^{-4} /hr. I.e.,

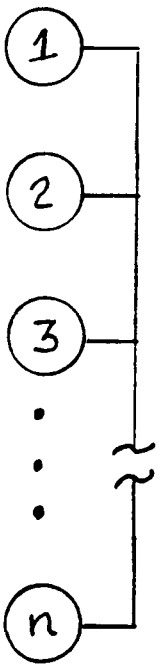
$$n = 10$$

$$q_o = 10^{-4}$$

Table 4-1 shows computed reliabilities for the three structures. Also shown are relative packaging weights and volumes based on the assumption that these parameters are directly proportional to module count.

ARCHITECTURE: SIMPLEX

STRUCTURE:



APPROXIMATE RELIABILITY EQUATIONS:

n MODULES

$$q_f \cong n q_{f_0}$$

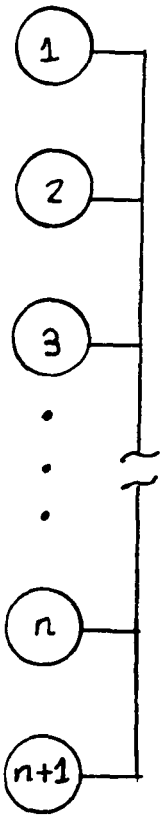
q_f =SYSTEM FAILURE PROBABILITY PER HOUR

q_{f_0} =MODULE FAILURE PROBABILITY PER HOUR

FIGURE 4-1(A)

ARCHITECTURE: DYNAMIC REDUNDANCY - SINGLE SPARE

STRUCTURE:



APPROXIMATE RELIABILITY EQUATIONS:

n MODULES + 1 SPARE

$$q_{f_1} \cong \binom{n+1}{2} q_{f_0}^2$$

$$q_{f_2} \cong (n+1) q_{f_0}$$

q_{f_1} =SYSTEM FAILURE PROBABILITY ONE HOUR (MISSION)

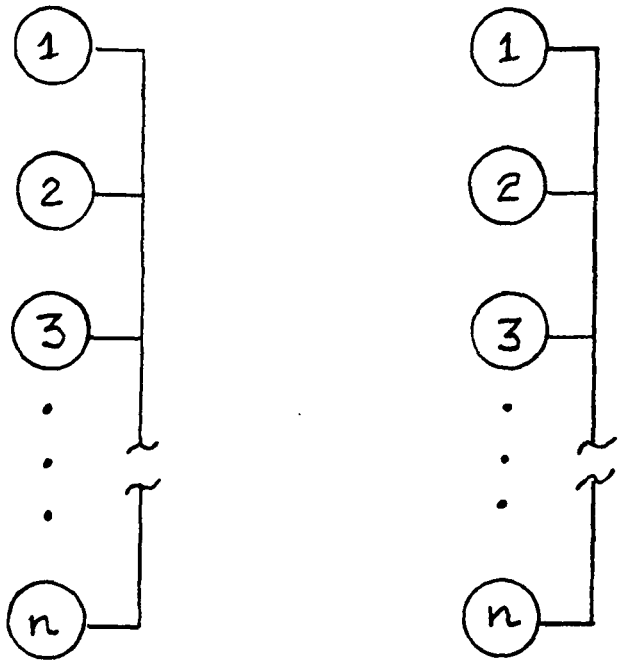
q_{f_2} =SYSTEM FAILURE PROBABILITY ONE HOUR (MAINTENANCE)

q_{f_0} =MODULE FAILURE PROBABILITY PER HOUR

FIGURE 4-1(B)

ARCHITECTURE: STATIC REDUNDANCY - DUAL

STRUCTURE:



APPROXIMATE RELIABILITY EQUATIONS:

n MODULES, DUPLICATED

$$q_1 \cong (nq_0)^2$$

$$q_2 \cong 2nq_0$$

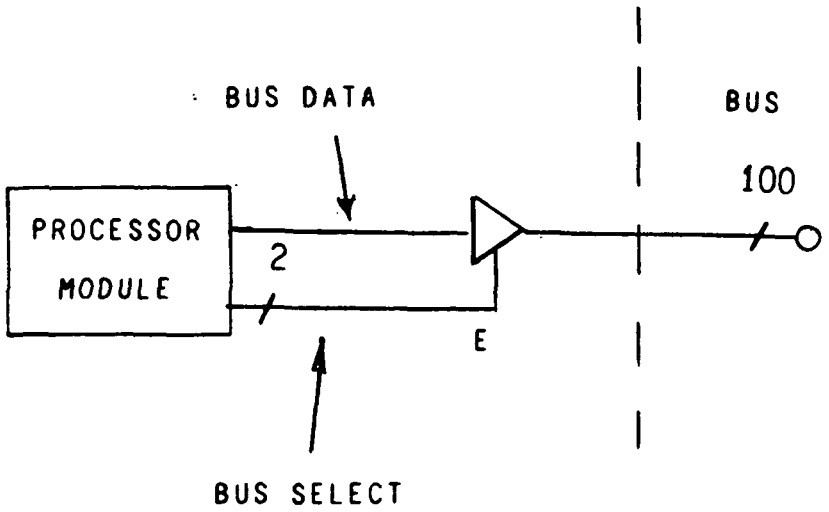
q_1 =SYSTEM FAILURE PROBABILITY ONE HOUR (MISSION)

q_2 =SYSTEM FAILURE PROBABILITY ONE HOUR (MAINTENANCE)

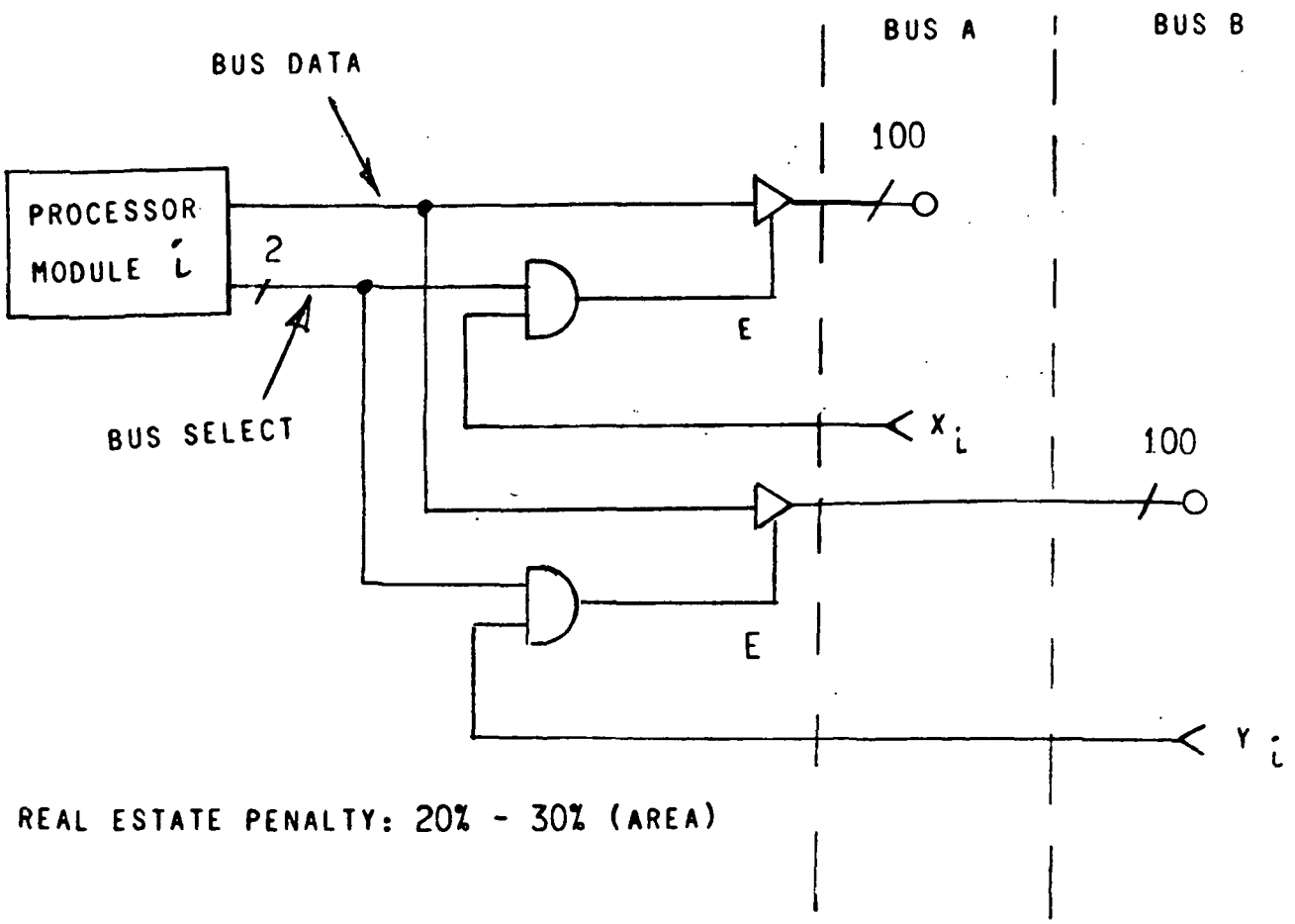
q_0 =MODULE FAILURE PROBABILITY PER HOUR

FIGURE 4-1(C)

SIMPLEX BUS INTERFACE



REDUNDANT BUS INTERFACE



BOARD REAL ESTATE PENALTY: 20% - 30% (AREA)

FIGURE 4-2 REDUNDANCY IN PARALLEL BUS INTERFACE

Table 4-1 - Characteristics of Architectures of Figure 4-1

Architecture	No. Modules	Hardware Reliability: Mission	Hardware Reliability: Maintenance	Weight/Volume
Simplex	10	10^{-3}	10^{-3}	1.0
Standby - Sparing (Theroetical)	11	5.5×10^{-6}	1.1×10^{-3}	1.1
Dual	20	10^{-6}	2×10^{-3}	2.0
Requirement		5×10^{-5} (max)	1.5×10^{-3} (max)	minimize

It is clear from the table that, at this level of analysis, the dynamic redundancy based on a single spare will not only satisfy AMAP mission and maintenance reliability goals but is far superior strategy to static duplication.

Dynamic reduncancy however involves considerable hardware and software overhead not required in the dual system. This is discussed in the next subsection.

4.3 Dynamic Redundancy for AMAP/SANDAC IV

This subsection discusses some of the practical implications of realizing standby redundancy for the SANDAC IV - based AMAP system.

4.3.1 Diversity of Module Types

The "first-cut" analysis in Section 4.2 carries the implicit assumption that modules are identical - i.e. the "spare" can replace any failed module. AMAP however consists of a family of modules, e.g. 6 distinct types are employed in the 10 modules of Table 2-2. In comparing AMAP module reliability estimates to the goals it is clear that a "spare" would have to be carried for each module type bringing the total count to 16 modules.

4.3.2 Register/Memory Reconfiguration

In reconfiguring a programmed - logic (e.g. microprocessor) system one must not only replace hardware but the contents of a failed unit's registers and data memory. Although a faulted module may contain correct register and memory contents, faults within an AMAP/SANDAC IV module will most likely block a spare module's accessibility to this information. To effect fault recovery, the

spare module will have to either:

(1) Reconstruct register/memory contents of the failed unit.

or

(2) Obtain "spare images" generated (by parallel computation) either locally or from some other module.

Data reconstruction is impractical:

(a) Values for pure counters and integrators cannot be reconstructed. These elements can however be expected to be widely employed in the advanced rotorcraft software algorithms.

(b) Processor reconfiguration times can introduce unacceptable transport delays in the software algorithms resulting in navigation/targeting errors and possible system instabilities.

Consequently some amount of spare parallel computation will be required in the dynamic redundancy approach. This would have to be done in the existing, or possibly additional, spares.

4.3.3 Flight Safety Fault Tolerance

AMAP computations leading to (ADOCS) flight control commands must, as a minimum, be duplicated in both hardware and software and results of both trans-mitted to the flight control system. (This would provide the flight control computers with a fail-detect-only capability and the require-ment to autonomously effect fail-safe recovery.) The duplicated computation would have to be done in the existing, or possibly additional, spare(s).

4.3.4 Additional Hardware Overhead

4.3.4.1 Bus Redundancy

SANDAC IV modules employ a simplex, parallel bus for inter-module communications. Module faults, most notably in interconnects and bus interface buffers, have a sizeable probability (Appendix A) of "jamming" the bus and taking the entire system down. Remedies for this would include both:

(a) Dual parallel bus.

(b) Isolation circuitry (e.g. dual buffers, analog switches, or relay networks).

Figure 4-2 shows a possible dual-bus/dual-buffer solution in which external signals (X_i and Y_i) could be generated by a non-faulted module to isolate faulted module i.

It is estimated that implementation of such a solution would entail a 20% to 30% increase in board area for each module. (It is believed that an analog switch network would require substantially more area; a relay network solution is not practical.)

4.3.4.2 Power Distribution Faults

In the SANDAC IV modules, the +5 VDC and $\pm$ 15 VDC rails constitute a single-point-of-failure in the sense that device breakdowns, connection "opens", trace shorts, etc. in a given module can propogate faults via the power bus into other good modules.

To protect the system from this probable type of fault, protection circuitry (e.g. LC filters and regulators) would have to be provided on each module for each supply voltage. (Estimated card area penalty: 10% - 20%).

4.3.4.3 Power Supply Faults

Dual power supplies are required. Implementation of this redundancy would very likely require additional load sensing and transfer circuitry on each power supply module. (Estimated card area penalty: 10% - 20%.)

4.3.5 Additional Software Overhead

Although the focus of this section is on hardware redundancy, it is well known that redundancy management software overhead for dynamic, stand by systems can be very high. Static redundancy management software typically commands some 10 - 40% of system memory and throughput resource. This figure can go to 70 - 90% for dynamic redundancy management. (The reasons for this will be seen in Section 4.4.) Additional software overhead translates to hardware overhead: i.e. additional slave processor(s) and memory.

4.4 Static Redundancy vs. Dynamic Redundancy for AMAP/SANDAC IV

4.4.1 Introduction

The foregoing paragraphs show that an implementation of AMAP using dynamic redundancy will involve the additional hardware overhead:

(1) Six spares would be required to cover the diversity of module types.

(2) Module circuit complexity would have to be increased to provide fault tolerance for parallel bussing and electrical power distribution resulting in a 30% to 50% increase in module volume.

(3) Some amount of hardware duplication would be required to

provide memory/register data "spares" and to meet flight safety requirements.

(4) Additional computational resources would have to be provided to support redundancy management software.

Under the assumption that items (3) and (4) could be accomodated using the spare modules, items (1) and (2) would represent the minimum hardware overhead needed to realize standby redundancy.

Table 4-2 shows characteristics for theoretical standby-redundancy, AMAP/SANDAC IV standby-redundancy and dual redundancy, Figures for the AMAP/SANDAC IV system are minimums. There figures show that in terms of hardware requirements, the static and dynamic architectures are roughly equivalent: both have comparable maintenance reliability; both satisfy mission reliability requirements.

4.4.2 Static vs. Dynamic Redundancy Management Requirements

Discussion to this point has been principally concerned with establishing survivability through modular hardware redundancy. Redundancy however must be "managed": if a module fails, the surviving modules must be able to detect the failure, isolate it and effect recovery. It has already been indicated (Section 4.3.5) that the hardware and software* overhead requirements for dynamic redundancy can significantly exceed those for static redundancy and in fact constitute the major function of the overall hardware/software system. This appears to be the case for AMAP/SANDAC IV. When employed in the dynamic, single-standby-redundancy structure, a faulted AMAP module can successfully transmit "bad" data and addresses to non-faulted modules contaminating (or "faulting") the latter. Unless corrected, this kind of propagated faults can lead to system failure. The root of this problem is the fact that the 68000 microprocessor architecture has a very limited amount of register/memory error detection correction coding. To insure system survivability against fault propogation:

* These two elements can be traded off one for the other.

Table 4-2 - Characteristics of Practical AMAP/SANDAC IV Architectures

Architecture	No. Modules	Hardware Reliability: Mission	Hardware Reliability: Maintenance	Weight/Volume
Dynamic Redundancy (theoretical)	11	5.5×10^{-6}	1.1×10^{-3}	1.1
Dynamic Redundancy (AMAP/SANDAC IV)	16	4.3×10^{-7}	2.1×10^{-3}	2.1
Static (Dual) Redundancy	20	10^{-6}	2×10^{-3}	2.0
Requirement		5×10^{-5} (max)	1.5×10^{-3} (max)	minimize

(1) A majority of "good" processors would have to monitor and validate each bus transaction. For example, a slave attempting to write global memory would first have to have the transaction validated by another slave and the master before the write could be effected.

(2) Each module's continous built-in-test would have to be very extensive. For example, RAM checksums would have to be computed for each memory access.

These expedients cut very deeply into overall system throughput capability. For the 68000 architecture (and for that matter any conventional fixed-instruction-set microprocessor) certain areas remain uncovered such as:

(1) Undetected PROM faults generating invalid op-codes

(2) The "unintelligent" modules such as the 1553B and general purpose I/O modules.

The above problems do not arise in the dual architecture since module failures within one module set do not affect the function of the other module set. (This statement must be somewhat qualified since dual modules will communicate with each other. As will be seen in the next section, fault propogation protection is easily handled with minimal demands on system throughput.)

4.5 Architecture Tradeoff Study - Conclusion

For AMAP/SANDAC IV employing the ten modules shown in Table 2-2:

(1) Static redundancy would appear to be superior to dynamic

redundancy in terms of hardware requirements.

(2) Redundancy management demands on system throughput would be significantly less for the statically redundant, dual architecture.

The dual architecture accordingly appears to be the "best" approach for meeting advanced rotorcraft mission, maintenance and flight-safety reliability goals.

5.0 HARDWARE AND SOFTWARE DESIGN CONSIDERATIONS FOR AMAP/SANDAC IV DUAL ARCHITECTURE

5.1 Introduction

The foregoing section developed AMAP module set duplication as a candidate redundant architecture for the advanced rotorcraft.

There are two remaining steps to complete the design process:

1) Definition of redundancy management hardware and software methods that will implement the fault tolerant design.
2) Verification and Validation steps to insure that the design meets both functional and fault tolerance requirements.

These two steps are the subject of this section.

Before proceeding, it is important to note that redundancy management methods are invoked as a defense against physical faults only. Although redundancy management methods can to an extent handle certain types of man-made faults, the latter are all hopefully found in the final verification and validation steps.

5.2 Redundancy Management-System Level

In section 4, the dual AMAP architecture was developed against what was essentially a simplex advanced rotorcraft system (Figure 2-1.) In this subsection, the structure and function of this system architecture is redefined in a manner that will satisfy both processor reliability goals and system reliability goals.

5.2.1 System Redundant Architecture

Figure 5-1 depicts the dual processor embedded in the advanced rotorcraft digital system. This proposed structure is similar to that of ADAS and features:

1) The dual AMAP module sets or channels.
2) Dual redundancy in cockpit control/display subsystems. (Cockpit functions are assumed to be mission-critical. It is further assumed that the overall cockpit system must be fail-operate to satisfy system reliability goals. Note that this does not necessarily imply that cockpit hardware must be duplicated "across the board")
3) Dual 1553B connections to the simplex sensor suite.
4) Dual 1553B connections to the flight control subsystem.
5) Cross-strapped 1553B connections to dual radio communications.
6) Inclusion of redundancy management functions in the cockpit control display subsystem cross-strapped with AMAP. (This is discussed further in Section 5.2.2.2)

In this Structure:

1) Both processors compute in parallel.
2) For sensor system processing, one processor's 1553B interface to the sensor and communications subsystems is active (receive and transmit); the other processor's 1553B interface to the sensor and communications subsystems is in standby (receive only.)
3) Both processors' 1553B interfaces to the cockpit and flight control subsystems are active (transmit and receive.)

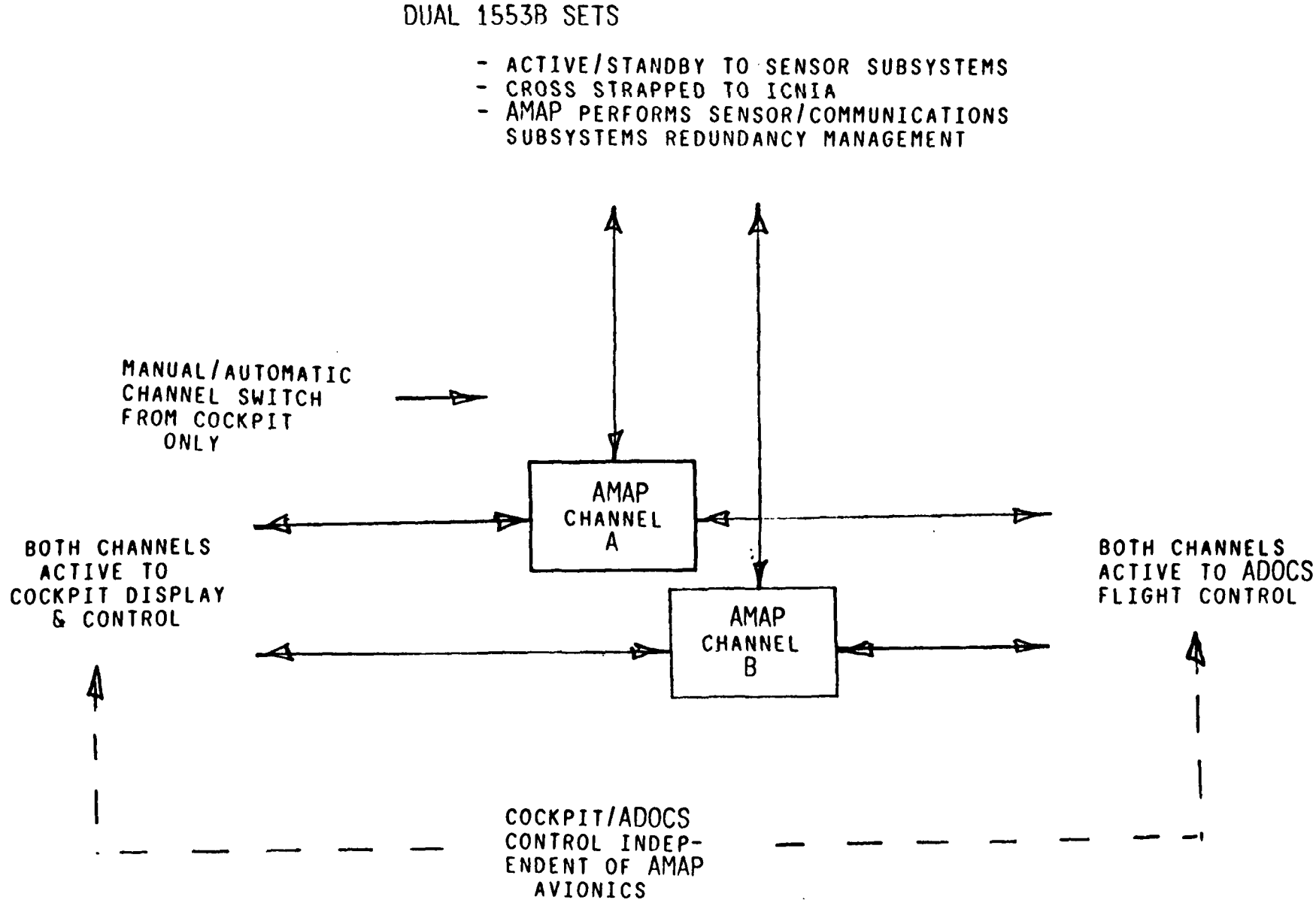


FIGURE 5-1 DUAL AMAP PROCESSOR CHANNELS

(Note: This system configuration, as defined, is based on the assumption that sensor and communications subsystems will employ conventional dual-1553B ports in active/standby mode. Dual active connections to cockpit are recommended; dual active connections to the flight control subsystem are mandatory where AMAP signals can effect flight safety.)

5.2.2 System Redundancy Management Design Considerations

5.2.2.1 Authority Hierarchy

Redundancy management involves not only fault detection and isolation but action to deselect, reconfigure and/or switch resources. Owing to the complexity of digital systems, one can not exclude the possibility of faults which result in fault-handling contentions between crew and the system or between elements within the system. For example, one cannot exclude the possibility of certain fault classes wherein pilot and computer (or one computer and another) "disagree" on the nature or location of faults and engage in a "fight" to assert control. For this reason, the system must be designed so that system elements have relative levels of authority, a higher authority element always having the capability of overriding element(s) of lower authority.

For the redundant avionics system, we would have, starting with the highest authority:

1) Crew decision/action.
2) Cockpit redundancy management subsystem (see below).
3) Dual AMAP channels.

4) Balance of digital system (sensor subsystems and communications subsystem.)

This hierarchy is illustrated in Figure 5-2.

Note that the flight control subsystem (ADOCS) is excluded in this list since its redundancy management considerations are completely independent of those of the avionics system.

5.2.2.2 Cockpit Redundancy Management Subsystem

Since ultimate authority for digital system management resides in the cockpit, panel avionics are required to display system fault status and permit the crew to alter (e.g. deselect, reconfigure, switch, etc.) resources at will.

This system is presently undefined but is seen to have the following requirements:

1) It must be fault tolerant not only within its own structuring but be capable of surviving all possible faults that can be generated by the subsystems it controls.
2) As will be seen, it will have to have some degree of (automatic) decision-making capability to support redundancy management of the dual AMAP system.

5.2.3 System Fault Handling

Following the authority hierarchy described in Section 5.2.2.1:

1) AMAP would utilize sensor subsystem BITE status, 1553B protocol (e.g. parity) and analytical redundancy (reference 2) to automatically detect failures in the sensor subsystems and deselect sensor(s) accordingly.

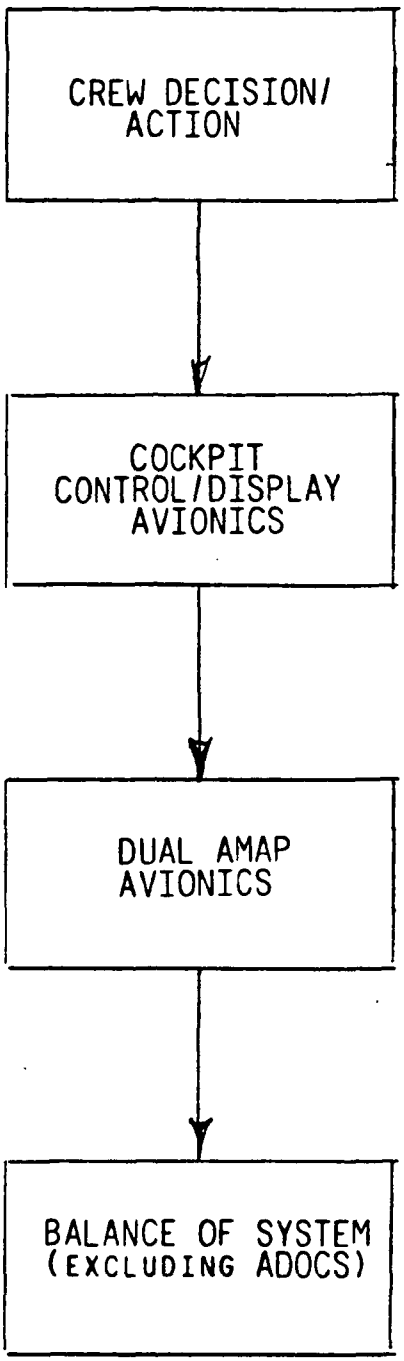


FIGURE 5-2 FAULT TOLERANCE AUTHORITY HIERARCHY

2) In the event of processor and/or 1553B channel failure, AMAP processors and the sensors would be manually or automatically switched to the standby processor and standby 1553B channels.

Before discussing the mechanics of effecting this switch (Section 5.4) it will be useful to examine, in more detail, the nature and effects of (physical) faults and how they are dealt with within the AMAP processor hardware.

5.3 Redundancy Management Techniques - AMAP/Sandac IV

Given that we have a dual active/standby structure for AMAP, our design objective is to develop methods to defect faults when they occur in the active channel and to effect the manual or automatic switch to the standby channel. An objective of this subsection therefore is to provide the avionics designer with both the general philosophy behind redundancy management and a "shopping list" of known redundancy management techniques.

5.3.1 General Considerations

5.3.1.1 Fault Mechanisms and Failure Effects

Although possible semiconductor and connector failure mechanisms are small in number, the number of possible failure states in a microprocessor system are virtually infinite. One cannot therefore pursue design of fault detection methods by enumerating all possible failure states.

We therefore take a more "macroscopic" view by noting that faults in a microprocessor system will in most cases result in three outcomes:

1) Incorrect sequential logic. (The most likely outcome here will be a system halt or "crash".)
2) Incorrect data values originating from faults in read/write store (assuming that memory has no parity protection) and I/O.
3) Incorrect frame rate resulting from oscillator/counter drifts/faults.

5.3.1.2 Fault Detection Strategy

Given a system of redundant channels, there are two basic strategies for detecting a faulted channel:

a) Each channel can perform self-diagnostics. When a fault is encountered, the channel declares itself "failed".
b) Channels can perform cross-diagnostics, "good" channel(s) detecting and identifying the "bad" channel(s) or at least the existence of disagreements.

The first of these strategies is preferred for two reasons:

The first is philosophical: under the self-diagnosis strategy a channel falsely declaring itself failed is indeed failed; under the cross-diagnosis strategy, a "bad" channel can declare a "good" channel failed thereby setting up a total system failure.

The second reason is practical: Self-diagnostics are easy to implement; cross channel diagnostics are much more difficult.

Emphasis in the following is therefore placed on self-diagnostic techniques.

5.3.1.3 Hardware vs. Software Redundancy Management Implementation

Redundancy management can be effected using hardware (parity checkers, comparators, "watchdog" timers, etc.) and/or software techniques. Since AMAP/SANDAC IV hardware is assumed to be a "given", emphasis in the following is on software techniques. (Some additional hardware requirements are however indicated; these are pointed out in the discussion).

5.3.2 Built-in-Test (BIT)

Processor and processor system built-in self-tests are performed to not only detect in-flight processor faults but to: (a) assist in maintenance, (b) provide preflight tests to assure that the processor system is correct. (Recall that mission reliability predictions are made under the premise that the system is fault-free when committed to mission operation.)

One can therefore identify three levels of BIT:

a) Maintenance Built-in-Test (MBIT.)

   Comprehensive test designed chiefly against field maintainability requirements.

b) Preflight Built-in-Test (PBIT.)

   Subset of MBIT functions designed to provide fast, preflight check of system integrity.

c) Continuous Built-in-Test (CBIT)

   Subset of MBIT and/or PBIT functions. Run in real time (each frame or in background across several frames) for purpose of detecting in-flight faults which do not affect program flow.

Typical BIT funcitons are shown in Table 5-1.

TABLE 5-1

TYPICAL MICROPROCESSOR BUILT-IN-TEST ELEMENTS

- CPU TESTS
  - INSTRUCTION SET TESTS
  - ALU LOGICAL FUNCTIONS
  - ALU ARITHMETIC FUNCTIONS
  - REGISTER TESTS
- ADDRESSABLE I/O AND INTERPROCESSOR COMMUNICATIONS
  - MONITOR VALIDITY OF PREPROGRAMMED TRAFFIC
  - TOKEN PASSING WITH DATA TRANSFERS
- NUMERIC PROCESSOR
  - ARITHMETIC CHECKS
  - FUNCTION COMPUTATION CHECKS
- MEMORY
  - PROM/EPROM CHECKSUMS
  - RAM PARITY
  - FULL ADDRESS/CONTENT TESTS (PREFLIGHT)
- TIMING
  - WATCHDOG TIMERS
  - INTRAPROCESSOR TIMING CHECKS

For AMAP, both master and slave processors would execute local BIT routines.

5.3.3 Deadline Mechanisms

As indicated earlier, microprocessor faults have a high likihood of disrupting intended sequential logic flow with the result that the system logic goes into a halted, fixed state.

Deadline mechanisms are a simple, effective means to detect this condition. The most widely-used mechanism of this sort is the so-called "watchdog timer". Here, an independent digital or analog timer is employed. In normal operation, the processor periodically (e.g. at the end of each computation frame) resets the timer. In the event of a fault-caused processor halt, the reset signal is not generated causing the timer to "time out" and flag the halt-state event. (The "watchdog" will also detect some oscillator failures.)

The "watchdog" principle can often be implemented without adding timer hardware. For example, the master and slave processors in an AMAP channel can each simply count frames and exchange frame counts. These multiple processors can accordingly "watch" one another and signal a fault condition when a frame count mismatch is encountered.

The reader can probably envision other (hopefully better) ways to apply this principle within the existing AMAP structure. (Additional "watchdog" hardware may be required for AMAP to cover the possibility of an entire processor channel entering a halt state.)

5.3.4 Software Assertions

Read/write (i.e. RAM) memory failures can result in incorrect data variables. (It is assumed that there is no RAM parity

checking.) Software assertions simply consist of code inserted in the application program which test the "reasonableness" of input data and computational results. Input or data memory failures resulting in unreasonable data values or data value changes in time can be flagged with these assertions. Assertion code blocks can be incorporated as a part of CBIT.

5.3.5 Built-in Redundancy Management

AMAP/Sandac IV hardware has several inherent features which can and should be employed to support fault detection including:

a) 1553B parity checks

b) Parallel bus protocols

c) Processor exception handlers

5.3.6 Predictive Task Scheduling

In designing combined sequential and parallel software tasks, one has two basic options:

a) Static (Predictive) Tasking. Task sequences are preplanned. A specific task sequence is executed only on the basis of polled input discretes (e.g. pilot mode selects).

b) Dynamic (Adaptive) Tasking. A task sequence is not known apriori, but occurs on the basis of interrupts and/or values of the input data.

In theory, dynamic tasking is superior in the sense that the "user" is serviced promptly and "dead time" tasks are avoided. In an avionics system however this superiority is not practically realized since:

a) Data changes and event response requirements are slow with respect to the system sampling (or frame) rate.

b) Task sequencing requirements are made not in the interest of rapid task execution but in getting the worst case task sequence done within the sampling period.

Static tasking on the other hand has a large potential benefit in detecting those sequential logic faults and timing faults that do not result in a processor halt. Since each possible task sequence is known in advance, processors withing the system can be programmed to verify that the correct sequence is indeed being executed. (Such programming could employ a combination of token - passing between processors and subframe counters.)

5.3.7 Wraparound Tests

Wraparound tests are designed to detect faults in processor I/O hardware. (In all of the foregoing redundancy management methods, software is employed to enable the CPU to check itself, memory, interprocessor communication integrity and timing. Input structure integrity testing is limited to parity checks and assertion testing; output integrity however cannot be determined via the CPU.) To effect wraparound testing one simply connects processor (parallel, serial, and 1553) outputs to corresponding inputs and executes I/O tests to verify that input and output hardware are functioning correctly.

The concept is illustrated in Figure 5-3. Hardware overhead is required to effect the wraparound test, specifically the (analog) switch network to effect the input-output connection.

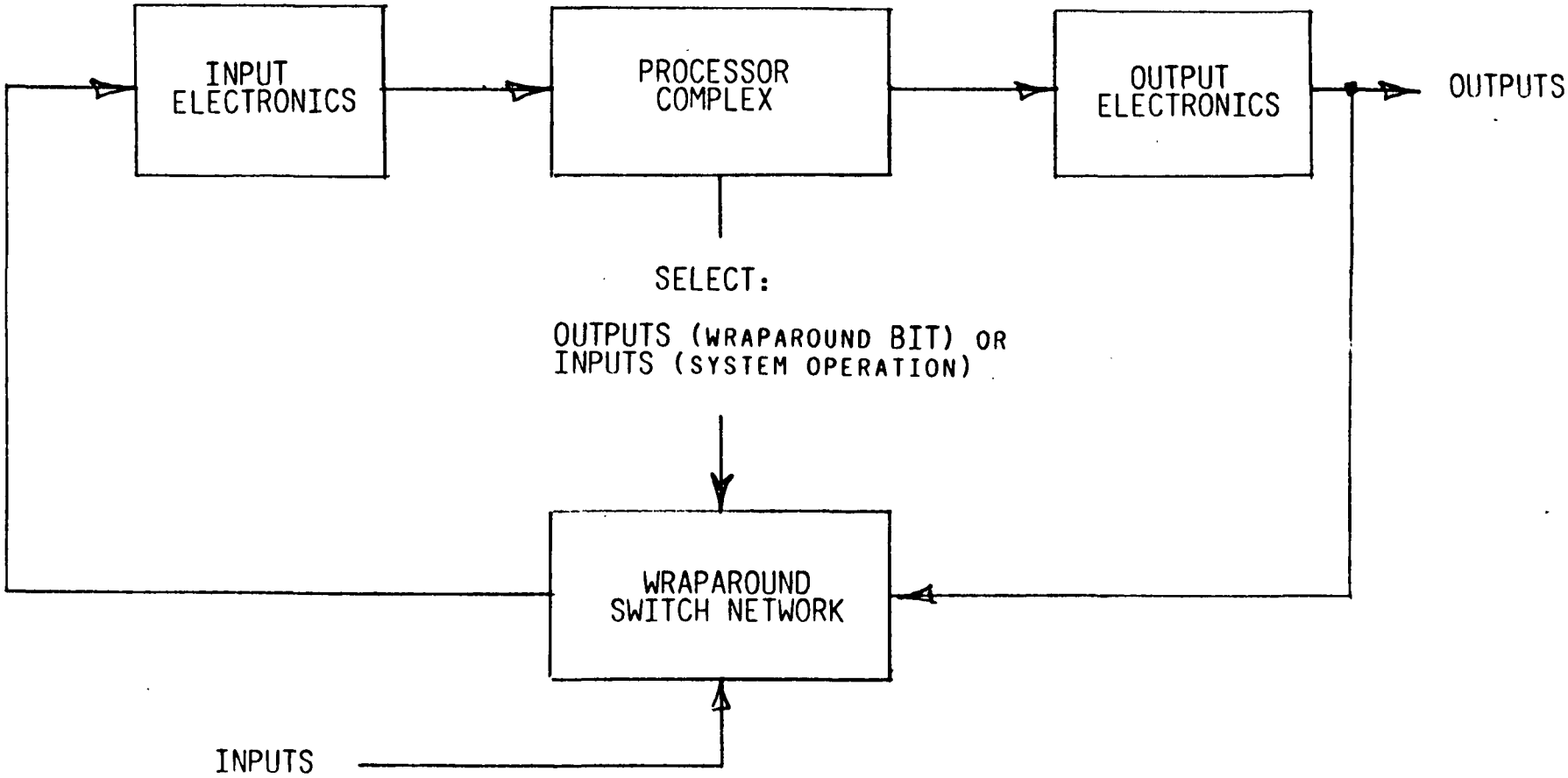


FIGURE 5-3 WRAPAROUND TEST STRUCTURE

5.3.8 Cross-Channel Testing

Redundancy management techniques presented to this point have focused on the preferred approach of having individual AMAP channels detect and announce their own faults.

Faults can also be detected externally:

a) Standby channel for example can monitor 1553B transmissions of the active channel and compare the latter's transmitted data values to its corresponding computed values. (Active channel transmissions to flight control subsystem would not be monitored since standby channel will also be transmitting to that subsystem.) If miscompares are encountered, the standby channel can signal the cockpit that a fault condition has been detected.

b) Flight control subsystem can likewise signal a fault condition on miscompare. (In this event, the flight control subsystem would have to revert to fail-safe mode of operation.)

c) Connected sensor and communications subsystems can, through parity checks and local data assertions, identify some (but not all) incorrect outputs from the active channel.

d) Cockpit Control/Display Avionics can likewise effect comparisons of processor outputs provided that the former have access to 1553B outputs.

To implement comparison monitoring, one must be concerned with synchronization (or lack thereof) of the dual AMAP channels. Pros and cons of synchronous and asynchronous strategies are summarized in Table 5-2.

Table 5-2 Synchronous vs. Asynchronous Redundant Digital Flight System

	SYNCHRONOUS	ASYNCHRONOUS
ADVANTAGES	- Cross-channel data differences provide positive fault indication in output voting plane. - Can use metastable algorithms (i.e. pure counters and integrators) in closed-loop operation	- Hardware channels independent
DISADVANTAGES	- Synchronization logic constitutes system single-point-of-failure	- Requires time-referencing for certain variables - Subject to nuisance trips in output voting planes - Requires asymptotically stable algorithms in closed loop application

5.4 Fault-Handling - AMAP/Sandac IV

5.4.1 Dual Processor Fault Detection and Isolation

In the event of a fault in an active channel, three events must transpire:

a) The fault must be detected,

b) The fault must be isolated to the active channel,

c) The "switch" must be made from the active to the standby channel. (I.e. the system must be reconfigured.)

Standby channel faults would be flagged for maintenance; flight would continue on the active channel with no backup.

As discussed earlier, dual processors can detect faults through:

a) Self-tests (These are summarized for the readers convenience in Table 5-3)

b) Cross-channel comparison of 1553B outputs.

As also discussed, identity of a faulted channel is more or less "guaranteed" through self tests whereas comparison monitoring can "guarantee" only fault existence.

Following the authority hierarchy discussed in Section 5.2.2.1, channel switch would be effected manually or automatically in the cockpit. To support implementation of this "switch", dual processor channels would have to provide status signals to the cockpit. These status signals are summarized in Figure 5-4.

TABLE 5-3

SUMMARY OF SELF-TEST METHODS

- BUILT-IN-TESTS (TABLE 5-1)

- DEADLINE MECHANISMS
  - WATCHDOG TIMERS
  - TASK SCHEDULE MONITORING

- ASSERTIONS
  - REASONABLENESS CHECKS ON COMPUTED DATA VALUES
  - ANALYTICAL REDUNDANCY

- BUILT-IN REDUNDANCY MANAGEMENT FUNCTIONS
  - 1553B PARITY
  - PARALLEL BUS PROTOCOL
  - EXCEPTION/TRAP HANDLERS

- PREDICTIVE SEQUENTIAL LOGIC FLOW
  - MULTIPLE PROCESSOR CHECKS ON REQUIRED TASK FLOW

- WRAPAROUNDS
  - COCKPIT CHECKLIST FUNCTIONS FOR MAINTENANCE AND PREFLIGHT BUILT-IN-TEST
  - DEDICATED HARDWARE TO TO EFFECT I/O CLOSURE

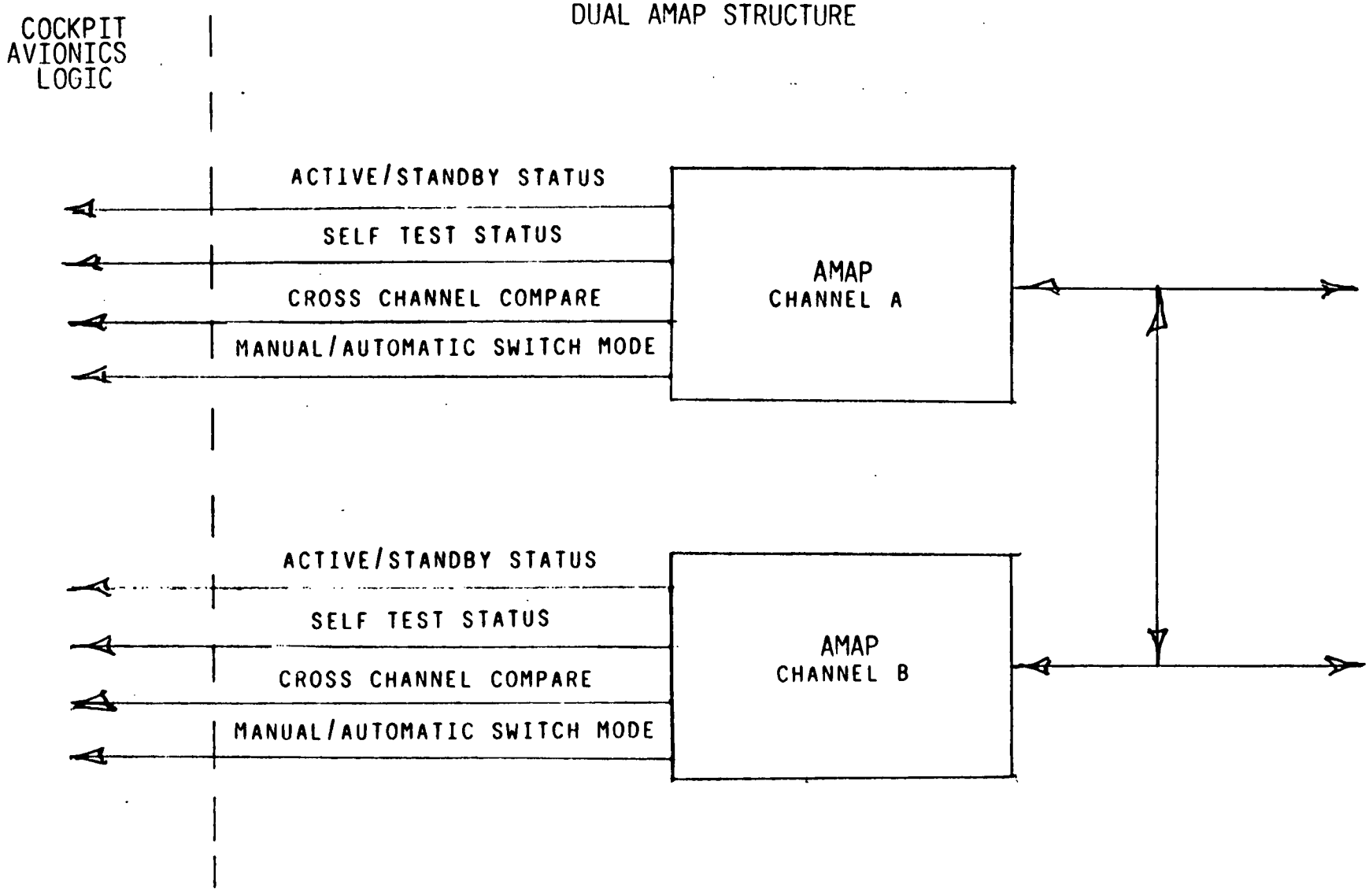


FIGURE 5-4 PROCESSOR STATUS SIGNALS TO SUPPORT REDUNDANCY MANAGEMENT

5.4.2 Cockpit Fault Monitoring and Reconfiguration Logic Requirement

Management of the dual processor redundancy, most particularly the switch of channels, would be effected with the cockpit control display avionics. At the present time, this system and its functions are undefined. This subsection therefore deals only with the embedded requirements of the cockpit avionics to effect fault detection, fault isolation and the switch to the standby unit. Such requirements would be refined (and quite possibly changed) as a part of the cockpit system detailed design.

Cockpit/AMAP redundancy management interface is summarized in Figure 5-5. Features of this interface include:

1) Circuit breaker disconnects to each channel.
2) Pilot can select:
    (a) Either processor channel in automatic mode enabling automatic channel switch, or,
    (b) Either processor in non-automatic mode (channel not switched)
3) Automatic mode would effect automatic switch to standby channel under the sole conditions of:
    (a) Standby channel self-test indicating no faults, and,
    (b) Active channel self-test indicating fault.

    Cross channel miscompares would only be announced; action would be left to pilot decision.

In implementing the foregoing cockpit functions one will probably have to address the question: "Which channel should be selected as the active channel?" In theory, it does not matter since the

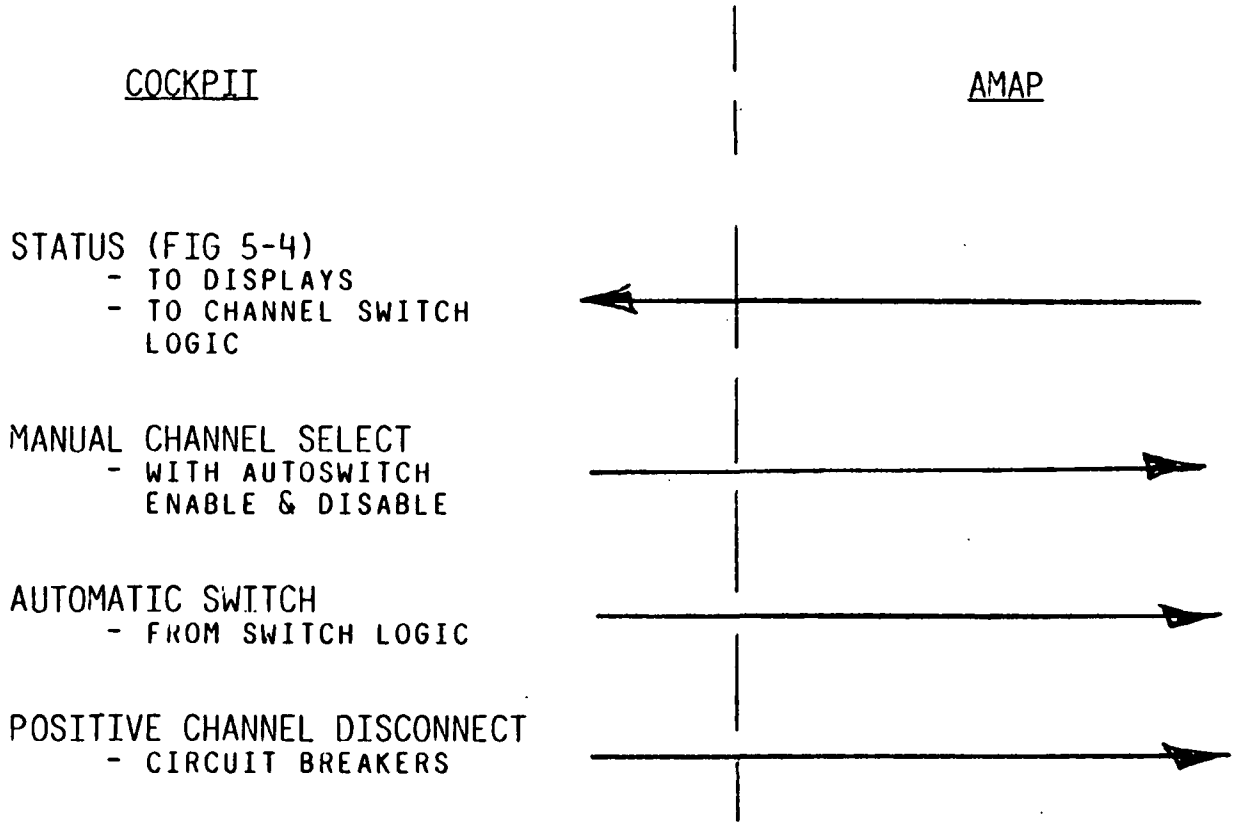


FIGURE 5-5 COCKPIT/AMAP REDUNDANCY MANAGEMENT INTERFACE

preflight built-in-test (PBIT) is designed to assure that both channels are perfect when flight operations commerce. From a practical standpoint however, one cannot design a "perfect" PBIT. For example, a standby channel may have a weak parallel output port connection which looks "good" on ground PBIT but suffers from intermittent "opens" from flight vibration. If the port is in standby, this fault will show up only after a switch from the active channels is made. These kinds of faults (frequently referred to as latent faults) tend to accumulate in non-exercised, standby systems.

An effective means of purging these kinds of faults is to periodically alternate active/standby roles of the two channels. Scheduling of active/standby roles could be incorporated in AMAP's built in maintenance - testing/logging system, designated roles being furnished automatically or as a crew checklist advisory.

5.5 Verification and Validation of Digital System Fault Tolerance

5.5.1 General Considerations

In the design, development and fielding of the fault tolerant digital system one seeks to satisfy not only system functional requirements but continued, correct system operation under all probable fault conditions.

The following paragraphs discuss some of the major techniques that have been employed in the past to address digital system fault tolerance. (No one has yet found a way to prove fault tolerance under all probable fault conditions.) These techniques are employed as a part of the engineering activity generally referred to as sytem verification and validation. Several definitions exist for these

terms. For the purposes of this report we will use the following definitions:

(Fault Tolerance) Verification - Process of establishing that the AMAP/SANDAC IV - based rotorcraft system design will continue to function correctly under all probable fault conditions.

(Fault Tolerance) Validation - Process of testing in the real environment or an environment nearly as real as possible that the system does continue to function under all probable fault conditions.

Verification activities are principally "paper" oriented, consisting of on-going design analyses begun at the early, preliminary design phase and continuing through completion of detailed, documented system design. A major emphasis in verification is to continually insure that (written) system specifications are being satisfied during the development process.

Validation activities on the other hand are concerned with the actual performance of the complete, piloted system in a full-up simulation or flight environment. Validation activities seek not only to verify specification correctness but the fact that actual system requirements are actually being satisfied.

Given these definitions, verification and validation activities as applicable to AMAP/SANDAC IV fault tolerance are discussed seperately as follows.

5.5.2 Verification Design Analyses of Fault Tolerant Systems

As indicated earlier in this report, a fault tolerant system definition evolves through an iterative sequence of candidate design

definition followed by design analysis.

The following briefly describe four principal analysis approaches that are frequently employed in fault tolerant system design analysis.

5.5.2.1 Reliability Analysis

Preliminary reliability analyses for AMAP/SANDAC IV were presented in the beginning part of this report (Sections 3 and 4). Such analyses provide "order-of-magnitude" accuracy and are intended to guide overall evaluation of architectural candidates.

As more detailed system definition evolves, one turns to more accurate, formal reliability prediction methods including:

(a) MIL-HDBK-217D, a piece-part reliability prediction tool. (Single channel reliability estimates; maintenance reliability estimates.)

(b) MIL-STD-756B, derives reliability equations for redundant system configurations.

(c) Reliability estimating computer programs such as the CARE III reliability modelling and analysis program recently released by NASA Langley.

5.5.2.2 Failure Modes and Effects Analysis (FMEA)

FMEA constitues a "bottom-up" approach for evaluating fault tolerant systems. Here, one identifies the probable failure modes that can occur at the component, module, and/or system level. For each identified failure mode, the system is then analysed to determine its fault response.

Probabilities are often associated with each failure mode so that a net probability can be assigned to the aggregate failure effects of all failure modes.

5.5.2.3 Fault Tree Analysis

Fault tree analysis "reverses" the FMEA and begins with a "top-level" event such as "total system failure." Given the "top-level" event, one then seeks to define all of the "second-level" events which can give rise to the former. Each "second level" event is then broken down into "third level" events, and so on. This process results in a tree structure, the lowest-levels of the tree constituting system component failures.

5.5.2.4 Single-Point-of-Failure Analysis

Redundant system realizations frequently contain single elements which when failed can lead to total system failure.

The "man-made" faults discussed in Section 3.1 can constitute single-points-of-failure. Redundant systems may moreover depend upon elements such as non-fault synchronization logic or simplex monitors whose physical faults can lead to system failure.

In effecting a single-point-of-failure analysis, one seeks through scrutiny of system documentation to identify all of the possible single-points-of-failure and to estimate the probability of occurance of each. The analysis can lead to one of two actions:

(a) Retaining the element(s) constituting single-points-of-failure where it is clearly demonstrated that system reliability requirements are not compromised, or

(b) Redesign including possible additional redundancy.

Common single-points-of-failure are summarized in Table 5-4.

5.5.2.5 Analysis Limitations

All of the foregoing analysis approaches are in reality ad hoc engineering approaches: reliability estimates are as good as the user's reliability model; correspondingly, there are no guaranteed ways of enumerating all failure modes, to generate complete fault trees or to identify all possible single-points-of-failure.

The analysis techniques do however collectively constitute somewhat independent, systematic frameworks with which the designer/analyst can eliminate design deficiencies that would otherwise produce serious setbacks during validation testing or lead to costly retrofitting in the field.

5.5.2.6 Verification Documentation

Fault tolerance analysis methods and results are invariably documented for the purposes of:

(a) Obtaining airworthiness approval.

(b) Guiding development of the validation test plan.

5.5.3 Fault Tolerance Validation Testing

5.5.3.1 General Considerations

Although design analyses are important ingredients in ultimately realizing viable complex digital flight systems, there is probably no better development tool to demonstrate design integrity (or to expose design weakness) than testing.

The AMAP/SANDAC IV development will undoubtedly go through several levels of testing:

Table 5-4 TYPICAL SINGLE-POINTS-OF-FAILURE

SINGLE-POINT-OF-FAILURE	POTENTIAL SOLUTION	TECHNOLOGY DEMONSTRATION
SOFTWARE	System reset with transfer to primitive and provable code.	- Classical Recovery Block - REBUS NASA Ames Dryden 1978-1984
EMI (all channels)	Manual/watchdog reset. Automatic recovery with measured/zero aircraft state estimate.	- Classical Retry - Microprocessor Experiments NASA Ames 1978-1982
SYNCRONIZATION LOGIC	Parallel asynchronous operation with static/ stable algorithms.	- RAMPS NASA Ames 1979-1982 - Shuttle Computers Synchronized - AFTI/F-16 Total Failure
GENERIC COMPONENT DEFECTS	100% Screen/Testing Select common componnents from dissemilar lots/processes	- Shuttle experienced failures - Shuttle flys generic hardware.
VLSI DESIGN/TOOLING SOFTWARE	Self test/finite state test	- VLSI on Boeing 757/767 Fly-by-wire Engine Control.

1) Module tests.
2) Individual channel bench tests (single AMAP LRU tested against simulated inputs and outputs.)
3) Bench tests with the redundant configuration.
4) Ground simulations with pilot-in-loop (as presently being done with ADAS).
5) Flight testing.

To test system fault tolerance one must, quite obviously, have faults as input stimuli. Although components can be expected to fail during development testing, such faults can be expected to comprise only an infinitesmal fraciton of all probable faults. It is therefore necessary to inject simulated faults during those tests performed as a part of system functional validation.

Two basic approaches for injecting faults are hardware fault insertion and software fault simulation. These are discussed in the following paragraphs. Before discussing these methods it is noted that fault injection exercises constitute part of a (written) overall system test plan. One must accordingly develop a fault injection test plan which hopefully will cover all the probable faults* that can occur during system operation. Results of fault tree and failure-modes-and-effects analyses provide key inputs to this test plan.

* One cannot, of course, test against all faults. For this reason, the verification analyses are frequently considered as additional bases for system validation.

5.5.3.2 Hardware Fault Insertion

With the prospect that several thousand faults may be injected during validation testing, one has the design challenge of introducing valid faults but doing so in a manner that will not damage system components. For example, short circuit faults to ground of high-current-carrying conductors are hard to simulate without producing over-voltage stresses on semiconductor junctions. Other types of hardware faults can however be safely and realistically simulated.* For example:

a) Connector open-contact-faults can be simulated using a relay or analog switch test rig temporarily placed between plug-connector interfaces within the system.

b) Semiconductor pin-level faults consisting of "stuck-at" and "open" logic levels have been simulated using the test setup illustrated in Figure 5-6. (Reference 6).

A well planned fault insertion setup will have the fault insertion hardware under (minicomputer) software control permitting input of a large number of fault patterns and automatic logging of fault response.

5.5.3.3 Resident Fault Simulation

Although the hardware fault insertion approach can provide realistic fault stimuli it has two major disadvantages;

a) Considerable effort must go into design and development of insertion hardware.

* USC understands that this is currently being done with ADAS at AVRADA

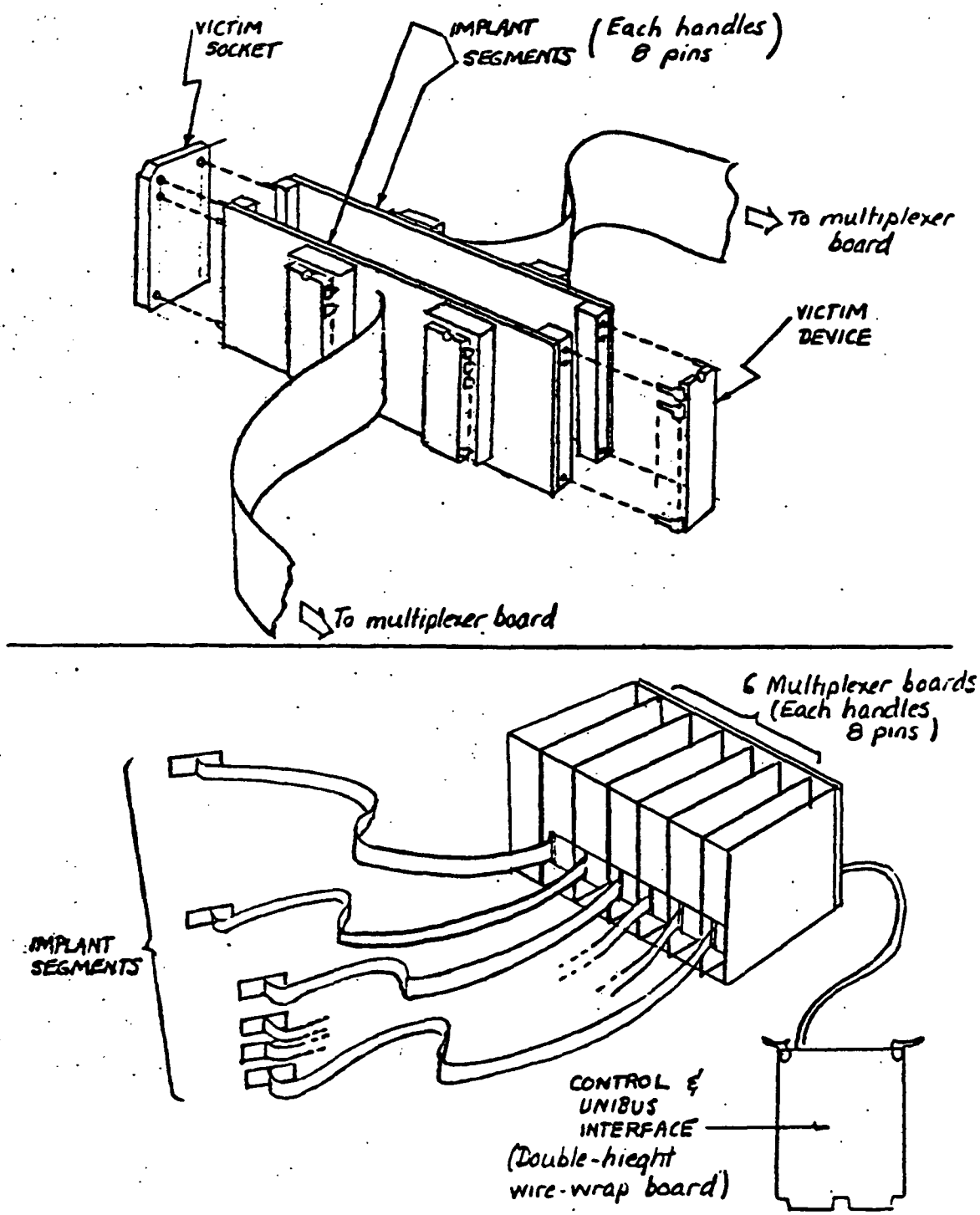


FIGURE 5-6(A) FAULT INSERTION SETUP FOR SEMICONDUCTOR DEVICES

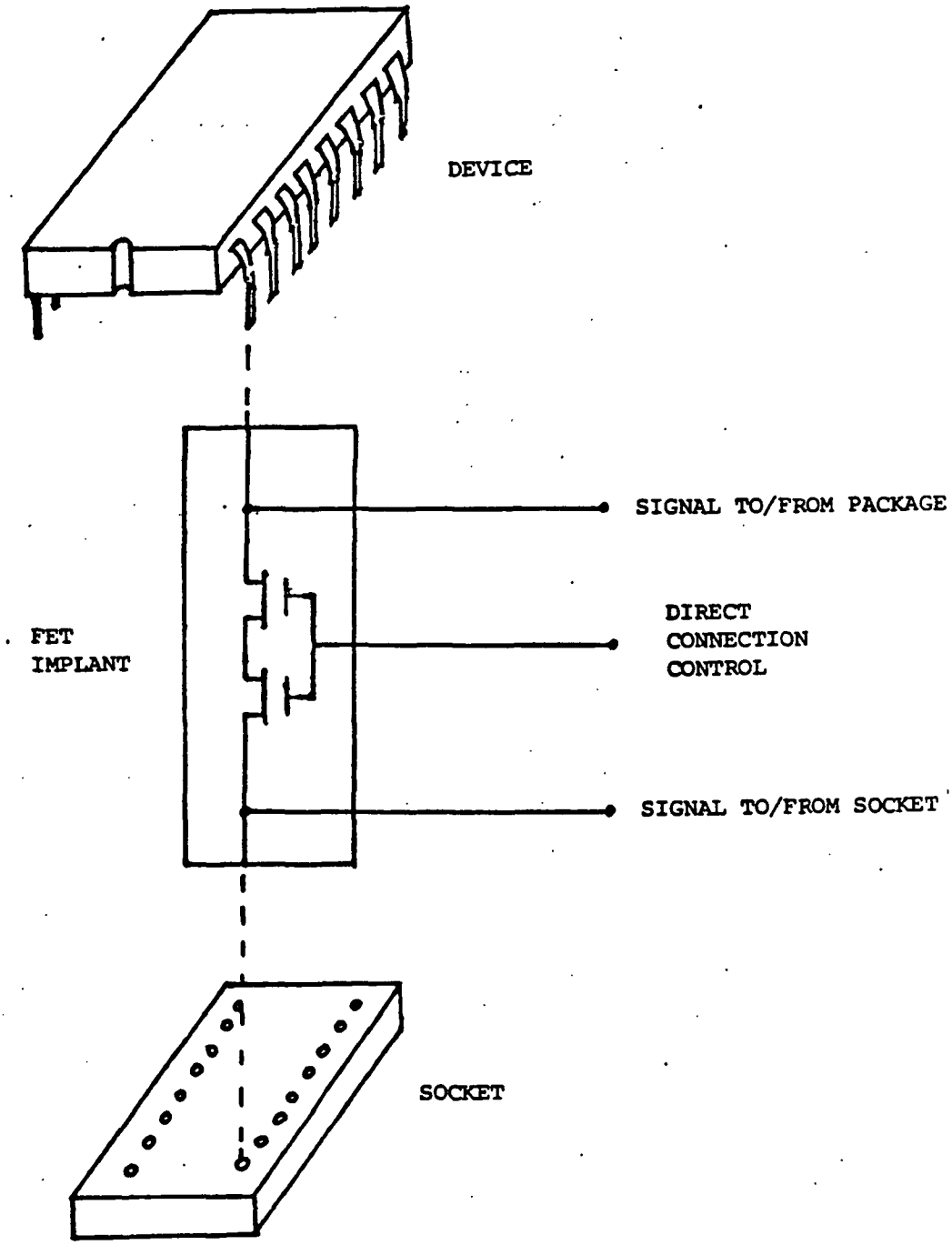


FIGURE 5-6(B) DETAIL OF INJECTION CIRCUIT
BETWEEN DEVICE AND SOCKET

b) Test set-up time can be prohibitive. (Insertion hardware would cover only a limited number of electrical contacts and chips. This hardware would have to be relocated several hundred times during validation.)

Many hardware faults can be simulated by colocating a fault simulation program with the applications programs in the master and slave processors. Upon external signal (provided through spare discrete inputs) this program could for example:

a) Execute a halt thereby simulating the effects of many sequential logic faults.

b) Fault memory locations. (EPROM would have to be temporarily relocated to RAM.)

c) Simulate faults in analog and discrete I/O.

5.5.4 Elimination of Man-Made Faults

Unlike physical faults, man-made faults (Section 3) can be eliminated through hardware and/or software re-design. Experience with fault tolerant digital flight control systems has shown that thorough ground integration and validation testing and flight testing* can expose in excess of 95% of man-made faults. By tracking design errors during testing one can obtain the (typical) history shown in Figure 5-7).

*For a system of the scale of AMAP/SANDAC IV/STAR: approximately 2000 hrs ground test; 50 hrs flight test.

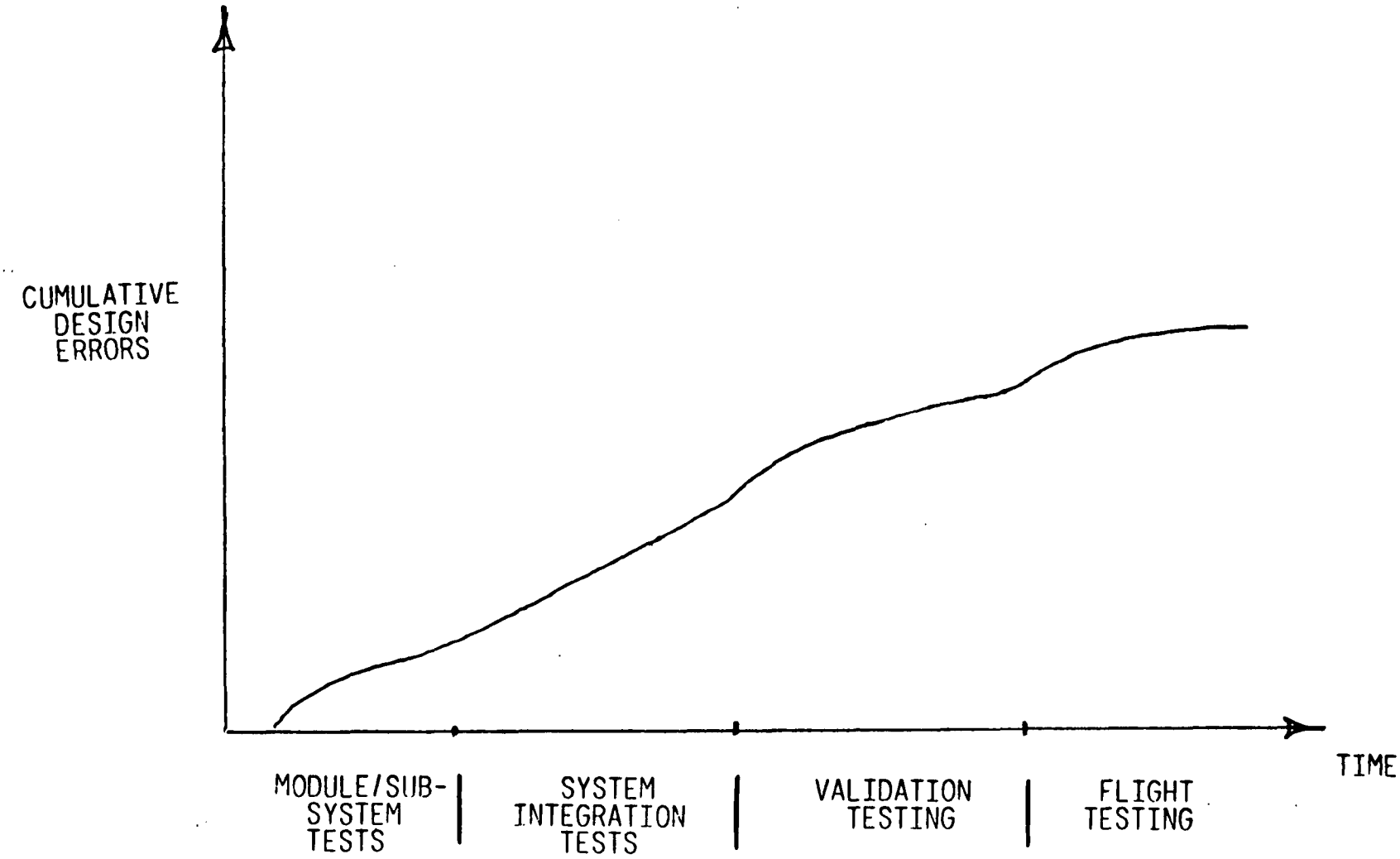


FIGURE 5-7 CUMULATIVE DESIGN ERRORS VS. DEVELOPMENT TIME

6.0 Conclusions

As stated at the outset, the principal objective of this report has been to provide the avionics designer with some of the perspectives, tools and techniques needed to realize the fault tolerant AMAP system in the advanced rotorcraft application.

Selection of static, dual redundancy for AMAP/SANDAC IV is based on information currently at hand. This choice however should be continuously re-examined as future AMAP and advanced rotorcraft system definition evolve from the AMAP/SANDAC IV development experience.

REFERENCES

1. J. Dasaro, C. Elliott, "Synthesis of an Integrated Cockpit Management System", AHS/NASA Specialists Meeting On Helicopter Handling Qualities, Palo Alto , Calif., April 1982.

2. G. Marner, R. Pruyn, ""LHX System Design for Improved Performance and Affordability", Journal of the American Helicopter Society, July 1983.

3. Boeing Vertol Company, "Integrated Subsystems and an Associated Cockpit Configuration Suitable for a Scout Version of a Family of Light Helicopters (LHX - Scout)"", Final Report of Conceptual Feasibility Study,
Report D210-12225-1, March, 1983.

4. J. Dasaro, "Evolution Toward a Multi-Bus Architecture for Army Helicopter Avionics Systems", AGARD Guidance and Control Panel 38th Symposium, Paper 45, Monterey, Calif. May 1984.

5. Boeing Vertol Company, "Advanced Digital/Optical Control System (ADOCS) Flight Demonstrator Program", Interim Technical Report D358-10045-1, May, 1983.

6. J.H. Lala, "Fault Detection, Isolation and Reconfiguration in FTMP: Methods and Experimental Results", 5th Digital Avionics Systems Conference, Seattle, Washington, Oct. 31 - Nov.3, 1983.

APPENDIX

PRELIMINARY RELIABILITY ANALYSIS OF AMAP/SANDAC IV PROCESSOR

Given a completed, detailed digital system design (including parts lists and component quality grades) one can formally employ the method of MIL-HDBK-217D to obtain a reliability prediction of the fielded system.

Where fine design detail is unavailable (e.g. in preliminary architectures tradeoff studies) digital system reliability must be estimated using nominal failure rate values for the system components. In the past, USC has used the following failure rates for estimating reliability of microprocessor-based digital flight systems:

Component	Failure Rate (per hour)
LSI Semiconductor chip	10^{-6}
Single connector contact	10^{-6}
Crystal	10^{-6}
Power Supply Reg./Cap.	10^{-5}
Discrete Logic, PC Boards Solder joints and feedthroughs	negligible

For the simplex (i.e. non-redundant) system, overall system failure rate is simply the sum of the failure rates of the individual components. For each AMAP/SANDAC IV module we would have roughly:

Item	Failure rate (per hour)
(10) LSI components	10^{-5}
(100) Contacts	10^{-4}
Balance	negligible
Total (per hour)	1.1×10^{-4}

From ten AMAP modules (Table 2-2 in main body of report) one would therefore have a failure rate of roughly 10^{-3} /hr.