NASA Contractor Report-187554

Advanced Information Processing System for Advanced Launch System: Avionics Architecture Synthesis

Jaynarayan H. Lala
Richard E. Harper
Kenneth R. Jaskowiak
Gene Rosch
Linda S. Alger
Andrei L. Schor

THE CHARLES STARK DRAPER LABORATORY, INC.
CAMBRIDGE, MA 02139

Contract NAS1-18565
September 1991

NASA
National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

NASA Contractor Report-187554

Advanced Information Processing System for Advanced Launch System: Avionics Architecture Synthesis

Jaynarayan H. Lala
Richard E. Harper
Kenneth R. Jaskowiak
Gene Rosch
Linda S. Alger
Andrei L. Schor

THE CHARLES STARK DRAPER LABORATORY, INC.
CAMBRIDGE, MA 02139

Contract NAS1-18565
September 1991

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

TABLE OF CONTENTS

Title **Page**

LIST OF ILLUSTRATIONS

LIST OF TABLES

ADVANCED INFORMATION PROCESSING SYSTEM FOR ADVANCED LAUNCH SYSTEM: AVIONICS ARCHITECTURE SYNTHESIS

1.0 INTRODUCTION

The goal of the Advanced Information Processing System (AIPS) is to achieve a validated fault tolerant distributed computer system architecture to meet the real time computational needs of advanced aerospace vehicles. One such vehicle is the Advanced Launch System (ALS) being developed jointly by the National Aeronautics and Space Administration and the Department of Defense to launch heavy payloads into low earth orbit at one tenth the cost (per pound of payload) of the current launch vehicles. An avionics architecture that utilizes the AIPS hardware and software building blocks has been synthesized for ALS. This report describes the AIPS for ALS architecture synthesis process starting with the ALS mission requirements and ending with an analysis of the candidate ALS avionics architecture.

1.1 Design for Validation Methodology

The ALS architecture synthesis process follows a new design for validation methodology that has been developed as part of the AIPS program to assure that fault tolerant computer system architectures for advanced applications meet the reliability, performance and other goals of the application. The design methodology is described in detail and compared to the conventional avionics design methodology in an accompanying report [1]. It is recapitulated here briefly to put various steps of the ALS architecture synthesis process in context.

In the AIPS design for validation methodology, a set of functional requirements is derived from the mission requirements and translated into avionics requirements, as shown in Figure 1-1. These avionics requirements are then mapped into hardware and software building blocks using a knowledgebase and future technology projections. Validation of the AIPS building blocks and generation of the knowledgebase and technology projections are goals of the AIPS program. The validation is being addressed using a combination of mathematical proofs, analytical models, and empirical test and evaluation. The architecture knowledgebase allows the designer to synthesize the architecture in accordance with rules and guidelines such that the fundamental principles of fault tolerance are adhered to and rationale for each design decision is related to overall mission requirements. The building block knowledgebase provides the designer with a detailed characterization of the performability and other important parameters of each building block. These characterizations, in conjunction with advanced technology projections, are used to project

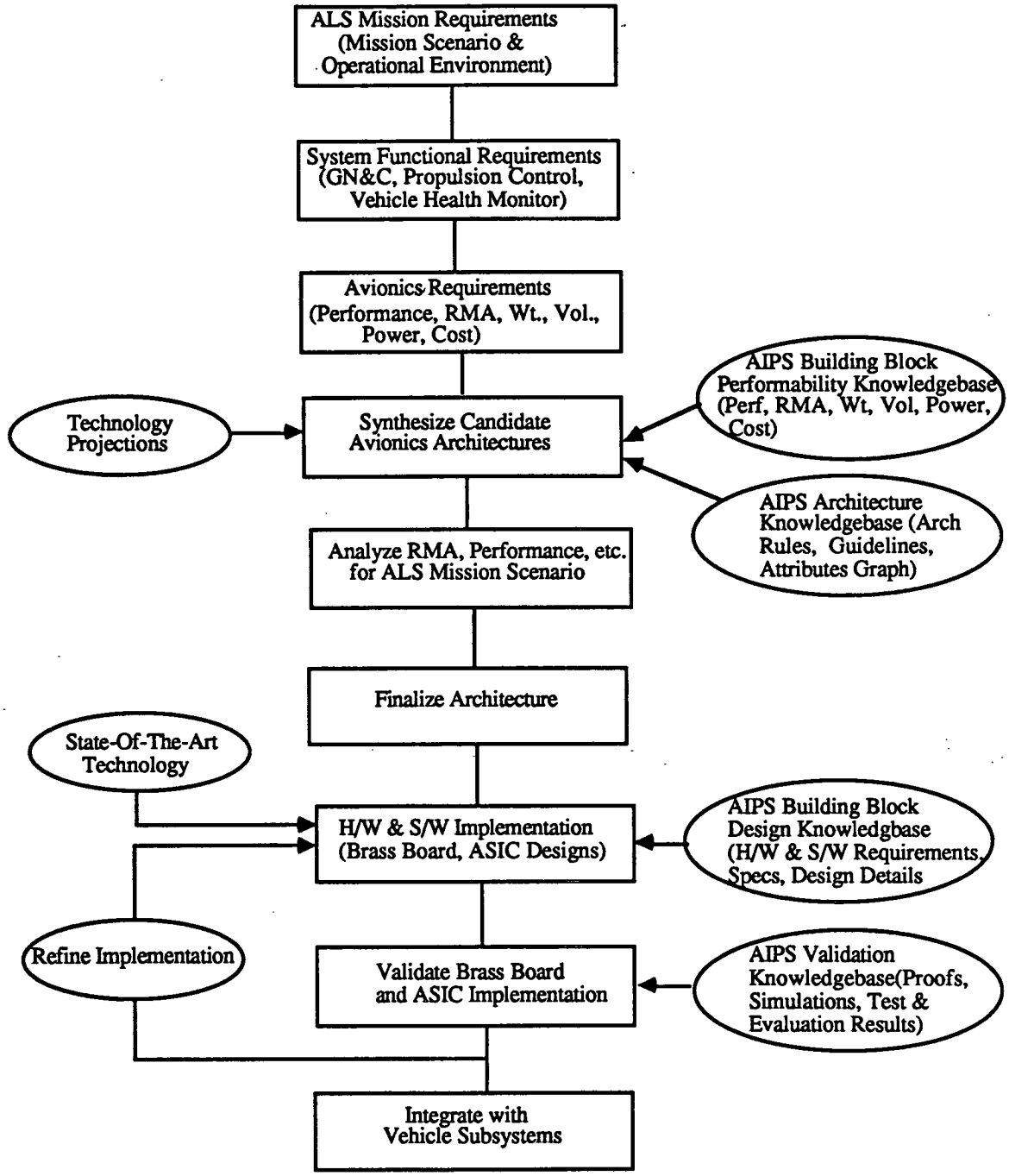

Figure 1-1. AIPS Design for Validation Methodology

building blocks implemented in state-of-the-art hardware and software technology. The AIPS hardware and software building blocks have been designed such that their major attributes such as Byzantine resilient fault tolerance, simplex programming model, reconfigurability, rigorous separation of redundancy management and applications software, etc. are not dependent on any specific technology of implementation. (The goal of the AIPS program is to validate these and other reliability and performance attributes of the AIPS hardware and software building blocks and make these building blocks available to programs like the ALS.) Furthermore, the system services are implemented such that their overheads become smaller as the processor and communication speeds increase. Therefore, the building blocks do not become obsolete with technology advancements. Their performance increases in direct proportion to improvements in processing and communication speeds.

This design methodology makes the architecture synthesis task much less of an art and personal judgement and provides a solid foundation of knowledgebase on which to base design decisions. In the conventional design methodology, validation of the architectural characteristics such as redundancy management, reliability, maintainability, performance, etc. is done in parallel with the validation of the specific hardware and software implementation of the design. The design for validation methodology decouples the validation of the architecture design from the validation of a specific hardware and software implementation of that design. The combination of architectural rules and guidelines, the prevalidated building blocks knowledgebase and the analytical models of the performability of the synthesized architecture assure a validated architecture. A validated architecture is defined to be an architectural concept that when implemented in hardware and software will meet various mission requirements such as reliability, throughput, transport lag, cost, weight, volume, power, etc.

Section 2 of this report describes the ALS requirements. The very high level requirements were collectively obtained from the three prime contractors, Boeing, General Dynamics, and Martin Marietta, and provided to Draper by Martin Marietta. These include the general ALS mission scenario and related parameters such as time on the launch pad, launch availability, mission duration and reliability. Other ALS requirements include ALS computational functions such as Guidance, Control, Navigation, etc. The functional requirements were translated into detailed computational requirements such as throughput, memory, processing lag, function iteration rate, I/O and interfunction communication rates, etc., by Martin Marietta with some feedback from and interaction with Draper on the format and content. These "raw" computational requirements were then converted into "derived" requirements by Draper with Martin Marietta's assistance. The conversion was necessary to accurately reflect the overheads of the Ada language and compiler to be used in programming AIPS for ALS compared to the assembly languages that have traditionally been used to program the launch vehicle avionics.

The AIPS architectural attributes, rules and guidelines, and reliability and performance models of the building blocks are described in detail in the accompanying report "Advanced Information Processing System: Design and Validation Knowledgebase" [1]. It also contains references to the more detailed hardware and software specifications and simulations that constitute the AIPS knowledgebase. This knowledgebase, which is quite large, is required to synthesize a validated ALS avionics architecture. Section 3 of this report briefly recapitulates the AIPS virtual and physical architectures and the key attributes of the hardware and software building blocks.

The AIPS architecture and its attributes are transparent to the microelectronics technology. The hardware building blocks can be implemented in the state-of-the-art technology for improved performance and reliability while still retaining all of the validated characteristics of the AIPS architecture. Similarly, the software building blocks, i.e., the system services, can be implemented using the latest Ada run time system and compiler. In order to project the reliability and performance that can be expected of the AIPS building blocks if they were implemented using the technology that will be available in the ALS time frame, a task to survey the technology was undertaken. An accompanying report "Advance Information Processing System for Advanced Launch System: Hardware Technology Survey and Projections" [2] describes the results of this survey in detail. Section 3.6 of this report describes the flight system characteristics, including hardware implementation, performance projections, and module failure rate projections, of the AIPS for ALS building blocks.

The projected building block performability characteristics are then used to configure a candidate AIPS for ALS avionics architecture that meets the ALS performance and reliability requirements. This process of configuring the building blocks is explained in greater detail in the next subsection 1.2. The resulting ALS avionics architecture is described in Section 4 of this report.

The reliability and performance of the candidate ALS avionics architecture have been modeled using the generic models of the AIPS building blocks developed earlier [1] but using the ALS specific parameters. The ALS specific parameters include the mission duration, launch pad time, repair strategy and other mission related parameters as well as the specifics of the AIPS for ALS building blocks such as the projected module failure rates, the fault detection, isolation and recovery times, various operating system service and redundancy management overheads, etc. Section 4.3 provides the details of the reliability and performance projections of the ALS avionics architecture.

Further refinement of the architecture would be necessary if the projected performability does not meet the ALS requirements. After the configuration has been revised to meet the requirements, the next step in the ALS avionics design would be the detailed design of the hardware and software building blocks using the state-of-the-art microelectronics and software technology. This was outside the scope of the present study

contract. Section 6 summarizes the results of this study with some thoughts on the steps necessary to reach a validated AIPS for ALS flight system hardware and software.

One of the prime drivers of the Advanced Launch System design is the reduction in the cost of launching a pound of payload by an order of magnitude over the current launch vehicles. The ALS on-board computer systems have the potential to reduce the operational cost by automating a number of functions that are now performed manually by "standing armies" of ground controllers, by making the launch window less vulnerable to weather through the use of adaptive navigation and guidance algorithms, by reducing the cost of launch pad operations through "maintenance free" fault tolerant computers and by reducing the cost of launch vehicle failures through the use of fault tolerance techniques. In addition, the cost of the avionics themselves can be reduced by changing the current design philosophy of single-string, non-fault tolerant systems built out of the highest quality (Class S) components with additional quality control checks at every stage from manufacture to launch to a philosophy of fault tolerant systems that do not use the most expensive components but are actually more reliable than single string systems. Section 5 of this report discusses in detail the impact of the AIPS for ALS architecture on the ALS cost.

1.2 Avionics Architecture Synthesis Overview

Architecture synthesis can be thought of as a constrained optimization problem. At the highest level the problem objective can be stated as: minimize cost subject to meeting all the avionics requirements. Cost studies, including the one described in Section 5 of this report, have shown that fault tolerant avionics constructed out of Class B components can be more cost effective for the ALS as a whole than single string avionics built out of Class S parts. Since the AIPS architecture provides prevalidated fault tolerant building blocks for ALS type applications, the architecture synthesis problem can be restated as: configure AIPS hardware and software building blocks to meet the ALS avionics requirements.

As discussed in the preceding section on the design for validation methodology, AIPS for ALS configuration(s) are defined using as inputs the AIPS architectural rules, guidelines and attributes, the projected reliability, performance, physical characteristics and other attributes of the building blocks, and the ALS avionics requirements. The process of matching the avionics requirements with the building block capabilities is a multidimensional problem. However, it can be simplified by decomposing the requirements into two orthogonal sets each of which can be mapped independently of the other as a first order approximation and each of which determines a different aspect of the architecture. The performance related ALS requirements such as throughput, memory, transport lag, input/output latencies, etc. determine the virtual avionics architecture. The reliability related ALS requirements such as probability of mission success, launch availability, launch pad maintenance, function criticality, etc. determine the physical avionics architecture.

The virtual avionics architecture definition includes the number of Fault Tolerant Processors (FTPs), allocation of functions to FTPs, partitioning of functions between the Computational Processor (CP) and the I/O Processor (IOP) within each FTP, the number and type of sensors and actuators and their interconnections to FTPs via the I/O networks, etc. The physical avionics architecture definition includes such parameters as the redundancy level of FTPs, the redundancy levels of Inter-Computer and I/O networks, the physical topologies of networks, the redundancy level of sensors, actuators and other I/O devices, the cross-strapping of I/O devices to channels of FTPs and the redundancy levels of interfaces, etc.

A preliminary virtual architecture can be defined by grouping and allocating functions to processing sites, grouping and allocating corresponding sensors and actuators to the same FTPs, and partitioning the functions in each site between the IOP and the CP. A subset of the AIPS System Services and modes of operation, such as scheduling of tasks, definition of I/O chains to acquire sensor data and send out actuator commands, etc. can then be selected to complete the virtual architecture definition. Criteria for grouping functions in one processing site include functions requiring time critical or high communication rates, physical location (e.g. propulsion controller to be located on the engine or recoverable avionics to be located in a separate module), like-criticality functions, etc. Some of the constraints for grouping and allocating functions to one processing site include the maximum useful throughput available in an FTP and the FTP data exchange bandwidth which is necessary to perform interactive consistency and internal congruent distribution of all the sensors connected to that FTP at the iteration rates necessary to support all the functions executing in that FTP. Once a preliminary allocation of functions has been completed, one can determine using the ADAS (Architecture Design and Assessment System) tool and simulations whether the performance criteria have been met. These include the transport lag for each function, inter-function communication latencies and rates, processor utilization and reserve throughput, etc. Functions can be reallocated, regrouped or number of processing sites added or deleted depending upon the results of performance modeling. The preliminary virtual architecture can be fine tuned using the results of the analytical models which are described in Section 4 of [1].

The process of defining, analyzing and fine tuning the physical architecture is similar to that for the virtual architecture; only the analysis tools and measures of merit are different. The analysis tools include the Markov models, combinatorial models, etc. The measures of merit include the probability of mission success, launch availability, probability of repair on the launch pad, etc.

Preliminary AIPS for ALS virtual and physical avionics architectures are presented in Sections 4.1 and 4.2, respectively, of this report. A preliminary set of reliability and availability projections are described in Section 4.3.

2.0 ADVANCED LAUNCH SYSTEM REQUIREMENTS

As explained in the introductory section of this report, one of the inputs to the ALS architecture synthesis process is the set of ALS requirements. The very high level requirements were collectively obtained from the three prime contractors, Boeing, General Dynamics, and Martin Marietta, and provided to Draper by Martin Marietta. These include the general ALS mission scenario and related parameters such as time on the launch pad, launch availability, mission duration and reliability. Other ALS requirements include ALS computational functions such as Guidance, Control, Navigation, etc. The functional requirements were translated into detailed computational requirements such as throughput, memory, processing lag, function iteration rate, I/O and interfunction communication rates, etc., by Martin Marietta with some feedback from and interaction with Draper on the format of the contents. These "raw" computational requirements were then converted into "derived" requirements by Draper with Martin Marietta's assistance. The conversion was necessary to accurately reflect the overheads of the Ada language and compiler to be used in programming AIPS for ALS compared to the assembly languages that have traditionally been used to program the launch vehicle avionics.

Section 2.1 describes the ALS functional requirements. The functional requirements were translated into processing requirements, described in Section 2.2.1, and I/O and interfunction communication requirements, described in Section 2.2.2. Section 2.3 captures the other ALS requirements such as reliability, maintainability, availability and operating environment.

2.1 Functional Requirements

Nine top-level ALS functions have been identified by Martin Marietta: Central Control and Processing, Winds Ahead Determination, Vehicle Power System Management, Steering and Staging Control, Sensor Processing, Propulsion Control, Command and Telemetry Processing, Range Safety and Destruct, and Programmable Payload Interface. Figure 2-1 shows the top level requirements along with the breakdown of each function into its next level component sub-functions for all except the Programmable Payload Interface. The function hierarchy is three levels deep, with functions at the third level assumed to be equivalent to executable, dispatchable tasks.

2.2 Performance Requirements

2.2.1 Processing Requirements

2.2.1.1 Advanced Launch System (ALS) Raw Requirements

The ALS computational and interfunction communication requirements were provided to CSDL in the format of two Hypercard™ documents: one describes processing requirements and the other describes interfunction communication requirements [3, 4]. The most recent such requirements received by CSDL were dated December 7, 1989.

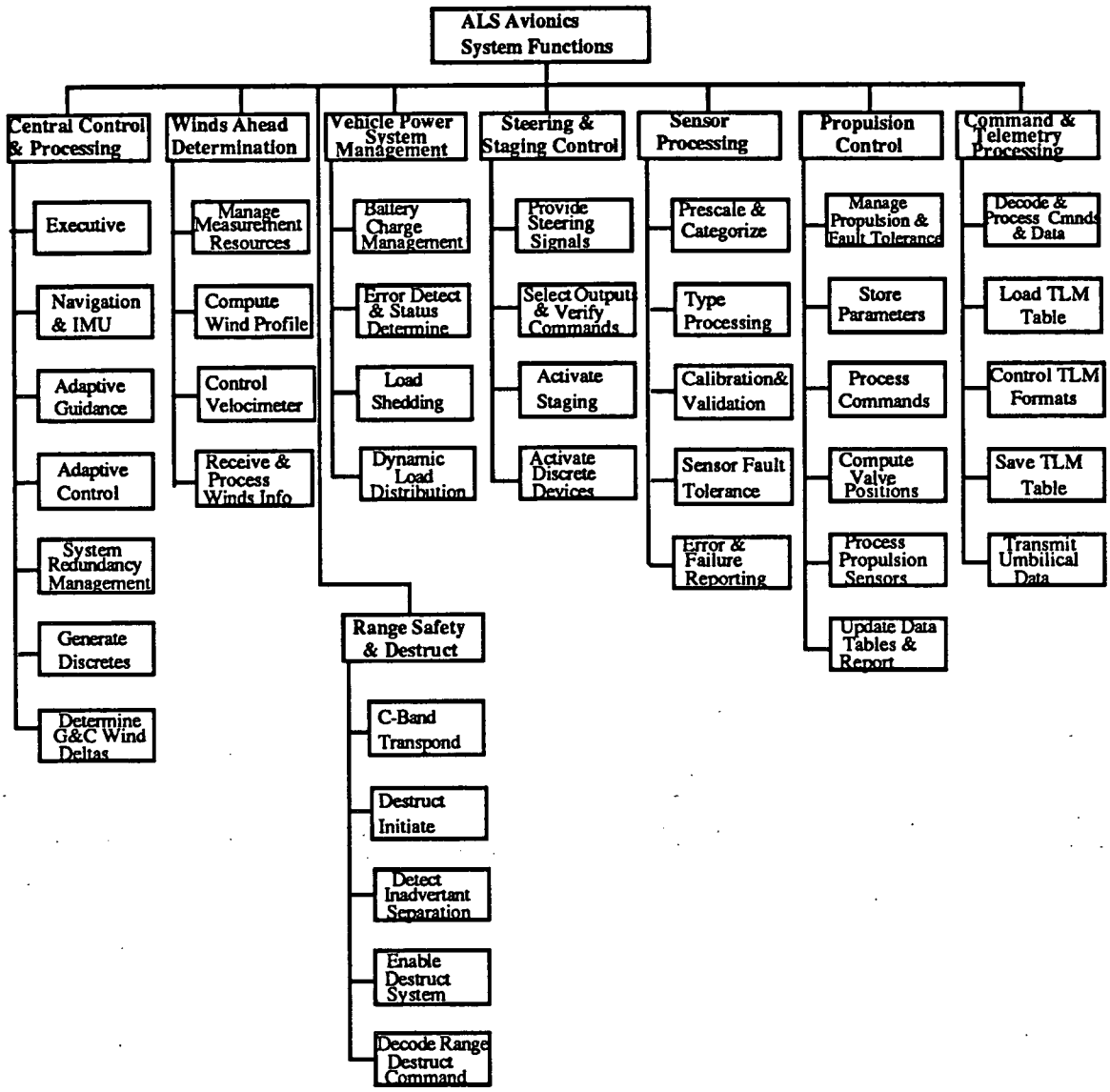

Figure 2.1. ALS Avionics System Functions

The ALS processing requirements are expressed by Martin Marietta in a hierarchical, dataflow-like representation. Currently, the function hierarchy is three levels deep, with functions at the third level assumed to be equivalent to executable, dispatchable tasks. Nine top-level functions have been identified. Computational requirements are given at multiple layers of the hierarchy, with the requirements of a higher-level function consisting of aggregate summations of the numerical requirements of its lower-level constituent functions. These aggregates have been observed to give an upper bound on the throughput requirements. To perform an accurate and meaningful synthesis of the avionics system, the following requirements are needed for each dispatchable task (level three in the representation hierarchy):

Frame rate
Throughput (or instructions per execution)

Throughput margin
Processing lag
Scheduling requirements (e.g., preemptible or nonpreemptible)
Task execution order dependencies
Inter-function communication requirements (bits per iteration, latency)

Given this data it is possible to construct a distributed schedule for the task suite and quantitatively perform system sizing and determine performance parameters. The current set of requirements does not yet possess this level of detail, so several simplifying assumptions are made to allow a rough system sizing. First, it is assumed that all tasks are preemptible within their frame by higher-frequency tasks, as long as the preempted tasks complete their execution within that frame. This allows the throughput utilization of a processor to be determined by summing the throughput requirements of the tasks it hosts. The second assumption is that the processing lag requirement will be met if the processor possesses the throughput to execute the requisite number of instructions of a task iteration within the processing lag. It is realized that these two assumptions may be mutually inconsistent since a task which is preempted within its frame may not meet its processing lag requirement. However, reconciliation of this potential inconsistency must be deferred until the detailed requirements are available.

CSDL has transferred the available numbers from the Hypercard documents to spreadsheets to facilitate their manipulation. The spreadsheets are presented as Tables 2-1 to 2-9 and are interpreted as follows.

Column A: Name and Number of Task(Note 1)

Column B: Frame Rate F(Note 1)

Column C: Throughput T0(Note 1)

Column D: Margin M(Note 1)

Column E: Processing Lag L(Note 1)

Column F: Margined Throughput TM = T0(1+M)(Note 2)

Column F Row 4: Total Margined Throughput Required = Σ TM(Note 3).

Column G: Instructions per Execution IE = TM/F

Column H: Instantaneous Throughput IT = IE/L.

Column H Row 4:

Maximum Instantaneous Throughput Required = max(IT).

Each task is identified by a name and a hierarchical number (Column A). The number indicates the place of the task or function in the three-level hierarchy of functional

requirements. For example, Kalman Filter (1.2.4) is Task 4 of the Nav/IMU (1.2) function which is sub-function 2 of the Central Control & Processing (1) function.

The Frame Rate (Column B) is the iteration rate in Hertz of the task. For example, Kalman Filter task must execute at 25 Hz.

The Throughput T0 (Column C) is the throughput, measured in instructions per second, required to perform a task. For example, Kalman Filter task is estimated to require 3.2 million instructions per second throughput.

The margin M (Column D) is the additional throughput requirement for a task. The margine is provided for the uncertainty in estimating the throughput or for growth purposes. The margin for Kalman Filter task, for example, is 0.5. The total throughput required for this task is T0(1+M) or 4.8 MIPS. This is called the Margined Throughput TM and is shown in Column F. The sum of all the margined throughputs for the tasks under a function is shown for each function in Column F, Row 4 of each table. For example, the total throughput for Central Control and Processing function is 21.8 MIPS, as shown in Table 2-1.

The maximum allowable processing lag for each task is shown in Column E. This is the interval from the time a task needs its inputs to the time it produces its outputs. The maximum allowable processing lag for Kalman Filter task, for example, is 36 milliseconds.

The number of instructions executed by a task per iteration is shown in Column G and is obtained by dividing the margined throughput TM by the frame rate F. For example, Kalman Filter task executes 210,000 instructions every iteration. The instantaneous throughput IT is the processing throughput required to execute a task within the requisite processing lag and is shown in Column H. For Kalman Filter task, this number is 5.83 MIPS.

Column H, Row 4 of every table indicates the maximum instantaneous throughput required for that function. For example, the Central Control and Processing function requires 10.72 MIPS to be able to perform the system identification task (1.4.3) within the required 10 msec processing lag. This number represents a lower bound on the throughput required of the processor

Notes:

1. From Martin Marietta ALS Specs.

2. Changed from TM = T0/(1-M) to prevent blowup at M=1.

3. Assuming preemptive scheduling.

Also, note that boldface italicized numbers represent CSDL guesses. Italicized numbers represent Martin Marietta-supplied aggregates and are not used in calculations unless otherwise noted.

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3	Central Control and Processing 1					Required		Required
4	Exec 1.1				sum:	21,829,396	max:	10,722,625
5	Central Control 1.1.1							
6		100	30,000	0.5	8	45,000	450	56,250
7	Timing & Sequence 1.1.2							
8		100	30,000	0.2	0.8	36,000	360	450,000
9	IPS Self-Test 1.1.3							
10		1	40,000	1	200	80,000	80,000	400,000
11	IPS Fault Detect & Mgmt 1.1.4							
12		25	50,000	1	10	100,000	4,000	400,000
13	Subsystem Status Monitor 1.1.5							
14		25	38,000	0.5	10	57,000	2,280	228,000
15	Mission Phase Sequence Gen 1.1.6							
16		100	5,000	0.2	5	6,000	60	12,000
17	Data Reporting 1.1.7							
18		50	20,000	1	1	40,000	800	800,000
19	Nav/IMU 1.2							
20	IMU Processing 1.2.1							
21		100	3,200,000	0.2	*10*	3,840,000	38,400	3,840,000
22	GPS Processing 1.2.2							
23		100	3,200,000	0.2	*10*	3,840,000	38,400	3,840,000
24	Error Compensation 1.2.3							
25		300	25,000	0.5	1	37,500	125	125,000
26	Kalman Filtering 1.2.4							
27		25	3,500,000	0.5	36	5,250,000	210,000	5,833,333
28	Nav Exec 1.2.5							
29		100	1,000	1	6	2,000	20	3,333
30	FDI 1.2.6							
31		50	132,000	0.5	0.5	198,000	3,960	7,920,000
32	Bending Processing 1.2.7							
33		100	1,200	0.2	0.1	1,440	14	144,000
34	Adptve Gdnce 1.3							
35	Two-body Linear Guidance 1.3.1							
36		50	37,500	0.2	15	45,000	900	60,000
37	Non-Linear Traj Shaping 1.3.2							
38		1	3,300,000	0.2	960	3,960,000	3,960,000	4,125,000
39	Contingency Control 1.3.3							
40		*1*	1,000	1	1	2,000	2,000	2,000,000
41	Adaptive Control 1.4							
42	Model Reference Adaptive Controller 1.4.1							
43		50	505,100	0.25	10	631,375	12,628	1,262,750
44	Classical Autopilot 1.4.2							
45		50	542,350	0.5	10	813,525	16,271	1,627,050
46	System Identification 1.4.3							
47		25	2,144,525	0.25	10	2,680,656	107,226	10,722,625
48	System RM 1.5 (includes CSDL RM; task throughputs approximated as 50KIPS/3)							
49	Fault Response 1.5.1							
50		25	17,000	1	40	34,000	1,360	34,000
51	Fault Determine & Isolate 1.5.2							
52		25	17,000	1	40	34,000	1,360	34,000
53	Configuration Manager 1.5.3							
54		25	17,000	1	40	34,000	1,360	34,000
55	Gen Discretes 1.6							
56	Validate Commands 1.6.1							
57		50	4,200	1	2	8,400	168	84,000
58	Update Outputs 1.6.2							
59		50	4,000	1	2	8,000	160	80,000
60	Discrete Error Handling 1.6.3							
61		*1*	1,000	1	0.5	2,000	2,000	4,000,000
62	Det G&C Wnd Ditas 1.7 (no task throughputs available)							
63		*1*	*29,000*	*0.5*	*20*	*43,500*		
64	Load Relief Algorithm 1.7.1							
65		100	0	0	10	0	0	0
66	Estimate Fluctuation Stats 1.7.2							
67		100	0	0	10	0	0	0

Table 2-1. Central Control & Processing

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3	Winds Ahead Determination 2					Required		Required
4					sum:	670,800	max:	656,842
5		*1*	*520,000*	*0.5*	*500*			
6	Manage Measurement Resources 2.1							
7		*1*	*100*	*1*	*1*			
8	Lidar Cal and Checkout 2.1.1							
9		1	0	1	1	0	0	0
10	Lidar BITE 2.1.2							
11		25	6,000	0.5	1	9,000	360	360,000
12	Lidar Fault Handling 2.1.3							
13		25	0	0	1	0	0	0
14	Lidar Health Monitoring 2.1.4							
15		50	3,000	0.5	1	4,500	90	90,000
16	Compute Wind Profile 2.2							
17								
18	Lidar mode Control 2.2.1							
19		1	0	1	1	0	0	0
20	Winds Measurement Fitting 2.2.2							
21		1	520,000	0.2	950	624,000	624,000	656,842
22	Vibration Compensation 2.2.3							
23		50	5,000	0.5	1	7,500	150	150,000
24	Bending Compensation 2.2.4							
25		50	10,000	1	5	20,000	400	80,000
26	Control Velocimeter 2.3							
27		*100*	*2,000*	*1*	*1*	*4,000*		
28	Redundant Lidar Configuration Control 2.3.1							
29		100	0	0	1	0	0	0
30	Lidat Power Control 2.3.2							
31		25	0	0	40	0	0	0
32	Receive and Process Winds Info 2.4							
33		*50*	*800*	*1*	*1*			
34	Detection 2.4.1							
35		1	0	1	1	0	0	0
36	Pulse Deconvolution 2.4.2							
37		1	0	1	1	0	0	0
38	Range Determination 2.4.3							
39		1	0	1	1	0	0	0
40	Doppler Frequency Estimation 2.4.4							
41		1	0	1	1	0	0	0
42	Data Collection & Formatting 2.4.5							
43		1	600	1	10	1,200	1,200	120,000
44	Check Range w/ Exp Values 2.4.6							
45		1	500	0.2	5	600	600	120,000

Table 2-2. Winds Ahead Determination

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3	Vehicle Power System Management 3					Required		Required
4					sum:	6,900	max:	138,000
5		*25*	*4,600*	*0.5*	*2*	*6,900*	*276*	*138,000*
6	Battery Charge Mgmt 3.1							
7		1	0	1	1	0	0	0
8	Charge Mgmt Exec 3.1.1							
9		1	0	1	1	0	0	0
10	Control Charging 3.1.2							
11		1	0	1	1	0	0	0
12	Monitor Charge 3.1.3							
13		1	0	1	1	0	0	0
14	Error Detection & Status Determination 3.2							
15		1	0	1	1	0	0	0
16	Pwr System Load Error Det 3.2.1							
17		1	0	1	1	0	0	0
18	Pwr Emergency Mgr 3.2.2							
19		1	0	1	1	0	0	0
20	Load Shedding							
21	3.3	1	0	1	1	0	0	0
22	Dynamic Load Distribution 3.4							
23		1	0	1	1	0	0	0
24	Pwr System Load Control 3.4.1							
25		1	0	1	1	0	0	0
26	Mission Power Profile 3.4.2							
27		1	0	1	1	0	0	0
28	Change Load Distribution 3.4.3							
29		1	0	1	1	0	0	0
30	Power-Up Initialization 3.4.4							
31		1	0	1	1	0	0	0
32	Monitor Load & Predict Deplete 3.4.5							
33		1	0	1	1	0	0	0

Table 2-3. Vehicle Power System Management

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Marqined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3						Required		Required
4	Steering & Staging Control 4				sum:	103,502	max:	1,800,000
5		*50*	*55,000*	*0.5*	*15*	*82,500*	*1,650*	*110,000*
6	Provide Steering Signals 4.1							
7		1	0	0	1	0	0	0
8	Identify Changed Cards 4.1.1							
9		1	0	0	1	0	0	0
10	Com Cmd to Prev & Limit Rate 4.1.2							
11		1	0	0	1	0	0	0
12	Gen New Steering Signal 4.1.3							
13		1	0	0	1	0	0	0
14	Report Steering Cmd Error 4.1.4							
15		1	0	0	1	0	0	0
16	Validate Steering Cmd Response 4.1 5							
17		1	0	0	1	0	0	0
18	Select Outputs and Verify Cmds 4.2							
19		*50*	*5,000*	*1*	*4*	*10,000*	*200*	*50,000*
20	Validate Discretes 4.2.1							
21		0.4	1,000	1	4	2,000	5,000	1,250,000
22	Activate Staging 4.3							
23		1	0	0	1	0	0	0
24	LFU Control & Monitor 4.3.1							
25		1	0	0	1	0	0	0
26	Verify & Sequence Power 4.3.2							
27		1	1	0	1	1	1	1,000
28	Ordnance BIT 4.3.3							
29		1	6,000	0.5	5	9,000	9,000	1,800,000
30	Fault handling 4.3.4							
31		1	1	0	1	1	1	1,000
32	LFU Redundancy Control 4.3.5							
33		1	0	0	1	0	0	0
34	Activate Discrete Devices 4.4							
35		1	0	0	1	0	0	0
36	Provide Discrete Outputs 4.4.1							
37		1	0	0	1	0	0	0
38	RCS Control 4.5							
39		1	0	0	1	0	0	0
40	RCS Error Generate 4.5.1							
41		1	0	0	1	0	0	0
42	Phase Plane Control 4.5.2							
43		1	0	0	1	0	0	0
44	RCS Command Generator 4.5.3							
45		1	0	0	1	0	0	0

Table 2-4. Steering & Staging Control

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3	Sensor Processing 5					Required		
4					SUM:	2,640,651	Max:	2,030,000
5		*25*	*636,000*	*0.5*	*34*			
6	Prescale & Categorize Sensor Data 5.1							
7		25	12,500	0.5	34			
8	Identify Format 5.1.1							
9		50	130,000	0.5	40	195000	3,900	97,500
10	Filter & Store 5.1.2							
11		50	812,000	1	16	1624000	32,480	2,030,000
12	Format-A Convert & Store 5.1.3							
13		25	5,000	1	40	*10000*	400	10,000
14	Format-B Convert & Store 5.1.4							
15		***1***	***1***	***0***	***1***	***1***	***1***	***1,000***
16	Type Processing 5.2							
17		*50*	*90,000*	*1*	*5*	*180000*	*3,600*	*720,000*
18	Compare With Limits 5.2.1							
19		***1***	***0***	***0***	***1***	***0***	***0***	***0***
20	Averaging & Voting 5.2.2							
21		***1***	***0***	***0***	***1***	***0***	***0***	***0***
22	Count Limits Errors 5.2.3							
23		***1***	***0***	***0***	***1***	***0***	***0***	***0***
24	Count Limits Errors 5.2.4							
25		***1***	***0***	***0***	***1***	***0***	***0***	***0***
26	Select Redundancy 5.2.5							
27		***1***	***0***	***0***	***1***	***0***	***0***	***0***
28	Calibration & Validation 5.3							
29		*25*	*112,500*	*0.5*	*5*	*168750*	*6,750*	*1,350,000*
30	Determine Criticality 5.3.1							
31		***1***	***0***	***0***	***1***	***0***	***0***	***0***
32	Manage Recovery 5.3.2							
33		***1***	***0***	***0***	***1***	***0***	***0***	***0***
34	Sensor Fault Tolerance 5.4.0							
35		*75*	*308,500*	*0.5*	*5*	*462750*	*6,170*	*1,234,000*
36	Limit Checks 5.4.1							
37		***1***	***0***	***0***	***1***	***0***	***0***	***0***
38	Consistency Checks 5.4.2							
39		***1***	***0***	***0***	***1***	***0***	***0***	***0***
40	Validation Error Report 5.4.2							
41		***1***	***0***	***0***	***1***	***0***	***0***	***0***
42	Error & Failure Reporting 5.5							
43		*1*	*100*	*0.5*	*5*	*150*	*150*	*30,000*

Table 2-5. Sensor Processing

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3	Propulsion Control 6					Required		
4					SUM:	20,000,000	Max:	26,666,667
5		50	10,000,000	1	15	20,000,000	400,000	26,666,667
6	Manage Propulsion and Fault Tolerance 6.1							
7		1	0	0	1	0	0	0
8	Prop Cntrllr Timing & Exec 6.1.1							
9		1	0	0	1	0	0	0
10	Prop Memory Mgmt 6.1.2							
11		1	0	0	1	0	0	0
12	Prop Fault Mgmt 6.1.3							
13		1	0	0	1	0	0	0
14	Update Data Tables & Report 6.2							
15		1	0	0	1	0	0	0
16	Report Engine Status 6.2.1							
17		1	0	0	1	0	0	0
18	Update Table 6.2.2							
19		1	0	0	1	0	0	0
20	Process Commands 6.3							
21		1	0	0	1		0	0
22	Monitor and Verify Commands 6.3.1							
23		1	0	0	1	0	0	0
24	Compute Valve Position 6.4							
25		1	0	0	1	0	0	0
26	Engine Control Loops 6.4.1							
27		1	0	0	1	0	0	0
28	Update Actuator Positions 6.4.2							
29		1	0	0	1	0	0	0
30	Compare Actuator Model 6.4.3							
31		1	0	0	1	0	0	0
32	Process Propulsion Sensors 6.5							
33		1	0	0	1	0	0	0
34	Sensor Data Scaling 6.5.1							
35		1	0	0	1	0	0	0
36	Redun Sensor Process and Qual 6.5.2							
37		1	0	0	1	0	0	0
38	Limit Monitor & Failure Detect 6.5.3							
39		1	0	0	1	0	0	0
40	Sensor Failure Handler 6.5.4							
41		1	0	0	1	0	0	0

Table 2-6. Propulsion Control

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3						Required		
4	Command & TLM Processing 7				SUM:	525,000	max:	525,000
5		25	350,000	0.5	40	525,000	21,000	525,000
6	Decode & Process Command Data 7.1							
7		1	0	0	1	0	0	0
8	Decode & Verify Commands 7.1.1							
9		1	0	0	1	0	0	0
10	Validate Ground Data 7.1.2							
11		1	0	0	1	0	0	0
12	Command Acceptance 7.1.3							
13		1	0	0	1	0	0	0
14	TLM Table Manager 7.2							
15		1	0	0	1	0	0	0
16	Format Telemetry 7.2.1							
17		1	0	0	1	0	0	0
18	TLM Transit 7.2.2							
19		1	0	0	1	0	0	0
20	Control Telemetry Format 7.3.1							
21		1	0	0	1	0	0	0

Table 2-7. Command & Telemetry Processing

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Thruput		Thruput (IPS)
3						Required		
4	Range Safety & Destruct 8				SUM:	5,400	max:	1,800,000
5		1	3,600	0.5	3	5400	5,400	1,800,000
6	Band Transpond 8.1							
7		1	0	0	1	0	0	0
8	Receive & Decode Signal 8.1.1							
9		1	0	0	1	0	0	0
10	Self-Test & timing 8.1.2							
11		1	0	0	1	0	0	0
12	Transmit C-Band Signal 8.1.3							
13		1	0	0	1	0	0	0
14	Command Receive & Decode 8.2							
15		1	0	0	1	0	0	0
16	Destruct Command Logic 8.2.1							
17		1	0	0	1	0	0	0
18	Select Power Source 8.2.2							
19		1	0	0	1	0	0	0
20	Sequece Destruct Signals 8.2.3							
21		1	0	0	1	0	0	0
22	Destruct System Control 8.3							
23		1	0	0	1	0	0	0
24	Safe & Arm ISDS 8.3.1							
25		1	0	0	1	0	0	0
26	Verify Loop Integrity 8.3.2							
27		1	0	0	1	0	0	0
28	ISDS Redundancy Manager 8.3.3							
29		1	0	0	1	0	0	0
30	Initiate Destruct 8.3.4							
31		1	0	0	1	0	0	0

Table 2-8. Range Safety & Destruct

	A	B	C	D	E	F	G	H
1		Frame	Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)		Lag (ms)	Thrupt		Thruput (IPS)
3						Required		
4	Programmable Payload I/F 9							
5		1	0	0	1	0		

Table 2-9. Programmable Payload I/F

2.2.1.2 Derived Requirements

The ALS throughput requirements discussed in the previous section were derived by Martin Marietta using certain assumptions that can have a substantial impact on the ALS avionics architecture and the number of processors and other hardware required to meet the performance requirements. In particular, the overheads for using Ada appeared too high in comparison with the CSDL experience with the new Ada compilers and Run Time Systems. As the Ada compilers have matured over the last few years, they have become much more efficient in generating code. Our recent experience indicates that newer Ada compilers generate code that is almost as good in code density and execution time as other high level language compilers such as C [22]. This appears to be true for a wide variety of programs including computationally intensive programs and operating system oriented programs.

In order to quantify the overheads of using Ada compared to an assembly language, and compare this overhead with the factor of 6 assumed by MM, a typical ALS application algorithm, the Lateral Acceleration Sub-System filter function (LASS), was chosen as a benchmark. This function had already been coded by MM in Ada. It had also been coded by CSDL in Ada using the CASE tool. Both versions were compiled with the Verdix 5.5 cross compiler and the XD Ada cross compiler produced by Systems Designers Software, Inc. and Digital Equipment Corporation. They were compiled with and without Ada constraint/range checking, thus producing a total of 8 versions. The Verdix 5.5 compiler is currently being used by CSDL on AIPS and a number of other projects. The XD Ada compiler is a newer, much more efficient compiler that has been undergoing Beta testing at CSDL for the past few months.

The eight versions of the LASS filter were run on the AIPS Fault Tolerant Processor (FTP) that was built for NASA Johnson Space Center (JSC) to obtain comparative execution times. The object codes of the eight versions were also analyzed to produce instruction counts. Figures 2-2 and 2-3 summarize the number of instructions and the execution times for the eight versions of the LASS filter function. Ideally, the execution time should be compared to the execution time of LASS function coded in an assembly language to determine the overhead of using Ada. However, the MM-supplied

requirements do not contain execution times of functions. They consist of an estimate of the number of instructions for each function. These estimates were arrived at by examining the function flow charts and using the following assumptions:

1. Time for 1 add = time for 2 instructions.
2. Time for 1 multiply = time for 4 instructions.
3. Time for 1 divide = time for 8 instructions.
4. Time for high order language (Ada) overhead (addressing, pragmas, etc.) = sum instructions for adds, multiplies and divides (#1+#2+#3) and multiply sum by 6.

Since the goal of the benchmarking was to calibrate the MM-supplied requirements (rather than to compare Ada to an assembly language on an absolute basis), we used the MM assumptions to compute the number of "assembly language" instructions for the LASS function. The total number of instructions, using the flow charts and the assumptions 1, 2, and 3, was calculated to be 74. This was multiplied by 6 (assumption # 4) to arrive at 444. This number can now be compared to the instruction counts in Figures 2-2 and 2-3.

H/W: AIPS Engineering Model JSC FTP 68020/68881/15.7 MHz

Compiler	Num of Instr per Iter	Time per Iter
Verdix 5.5 w checks	415	1.09 msec.
Verdix 5.5 w/o checks	341	0.85 msec.
XD Ada w checks	160	0.60 msec.
XD Ada w/o checks	115	0.43 msec.

Figure 2-2. LASS (MM Coded) Benchmark Results

H/W: AIPS Engineering Model JSC FTP 68020/68881/15.7 MHz

Compiler	Num of Instr per Iter	Time per Iter
Verdix 5.5 w checks	277	1.22 msec.
Verdix 5.5 w/o checks	247	1.14 msec.
XD Ada w check	109	0.54 msec.
XD Ada w/o checks	99	0.51 msec.

Figure 2-3. LASS (CSDL Coded) Benchmark Results

The following conclusions can be drawn from these benchmarks:

1. The XD Ada compiler consistently produces fewer instructions than the Verdix 5.5. compiler. The ratio varies from 2.5 to 3 depending on whether or not range checks are turned on.

2. The code produced by the XD Ada compiler consistently outperforms the Verdix produced code. The ratio approximately varies from 1.8 to 2.2. The code density and speed ratios between the two compilers are consistent with other benchmarks performed by CSDL.

3. The manually coded MM version produces more instructions than the CASE but generally executes faster. The CASE produced version passes many parameters with each iteration and the instruction used to push a parameter onto the stack apparently takes a longer time than the average instruction. Other reasons for this disparity have not been analyzed at this time.

4. Turning the range checks on or off has a much greater impact both in the number of instructions as well as execution time on the manually coded version than on the CASE produced version. This is due to the fact that CASE uses predefined functions for certain floating point operations which are not affected by turning the checks on or off.

Given the superior performance of the XD Ada compiler which represents state of the art in Ada compiler technology, it will be assumed here that the ALS will either utilize this compiler or an advanced Ada compiler of similar performance. The following conclusions are based on this assumption.

5. XD Ada compiler generates 99 to 160 instructions for the LASS filter compared to 74 counted in the flowchart, resulting in an overhead factor of 1.34 to 2.16. (As pointed out earlier, the number of instructions would be 444 using the MM overhead factor of 6 which would have been fairly representative of the earlier Ada compilers.) The ALS throughput requirements have been recomputed in the next section using an Ada overhead factor of 2.2 as mutually agreed to by MM and CSDL.

6. Martin Marietta used a further multiplicative factor of 2.3 to account for the "logical and environmental overhead" of scheduling and dispatching a task. This overhead is actually not dependent on the number of instructions in a given task. Instead, the overhead is a fixed number of instructions required for each *iteration* of the task, i.e. each context switch. The number of instructions for scheduling and dispatching a periodic task was measured using the Verdix 5.5 compiler and run time system (RTS). A periodic task dispatch takes 820 microseconds and was equal to 830 instructions. The instructions used for the context switch take less time (less than 1 microsecond per instruction) than typical computation intensive instructions. These numbers are currently not available for the XD

Ada compiler. However, based on the code produced by the two compilers and the fact that the XD Ada run time system is written in an assembly language while the Verdix run time system is written in Ada, our estimate is that approximately 500 extra instructions would be required for each iteration of each task. MM feels that there is still some overhead that is proportional to task size and a proportional factor of 1.5 in combination with a fixed overhead of 250 instructions per task per iteration should be used. The ALS requirements have been recomputed in the next section using these two factors.

A new set of ALS throughput requirements have been derived using the findings reported in the previous section. Using an Ada overhead of 2.2 rather than 6, and a fixed scheduling overhead of 250 instructions per task per iteration in combination with a proportional overhead of 1.5 rather than a factor of 2.3, the new ALS requirements are summarized in Tables 2-10 to 2-18. The following conclusions can be drawn from the modified requirements.

The reduced overheads of an advanced Ada compiler make the modified throughput requirements obviously much smaller than the initial requirements provided by MM. However, the effect of changing the scheduling overhead from one that is proportional to the task size to one that is dependent on the task iteration rate is not so obvious. The dispatch overhead for high frequency tasks is quite high: for example 25,000 instructions per second for a 100 Hz task. On the other hand, the requirements for large tasks are substantially reduced: for example, the Kalman filter function requirement (including margins) is reduced from 3.5 MIPS to .84 MIPS.

The net effect of these changes is that the overall ALS throughput requirements (including margins) for non-propulsion functions are reduced from about 26 MIPS to about 8.8 MIPS. Furthermore, the maximum instantaneous throughput (throughput required to perform the most demanding indivisible task) for non-propulsion functions is reduced from about 10.7 MIPS to about 3 MIPS. For propulsion control functions, the total throughput requirement reduces from 20 MIPS to about 4.8 MIPS and the maximum instantaneous throughput requirement reduces from 26.7 MIPS to about 6.4 MIPS.

The derived requirements were used to synthesize the ALS avionics architecture as discussed in Section 4.

2.2.2 I/O and Interfunction Communication Requirements

To obtain estimates of the Input/Output and Interfunction communication requirements, the ALS requirements Hypercard stacks provided by Martin Marietta [3, 4] were examined. The communication requirements contained in these documents are less complete than the computational requirements discussed above. For functions for which relevant communication requirements were absent from the Hypercard stacks but which were known to possess significant communications needs, other preliminary requirements documents were consulted [5, 6].

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thruput	Mod Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)	(IPS)		Lag (ms	Throughput		Thruput (IPS
3	Central Control and Processing 1						Required		Required
4	Exec 1.1					sum:	5,841,958	max:	3,036,730
5	Central Control 1.1.1								
6		100	30,000	*32,174*	0.5	8	48,261	483	60,326
7	Timing & Sequence 1.1.2								
8		100	30,000	*32,174*	0.2	0.8	38,609	386	482,609
9	IPS Self-Test 1.1.3								
10		1	40,000	*9,815*	1	200	19,630	19,630	98,152
11	IPS Fault Detect & Mgmt 1.1.4								
12		25	50,000	*18,207*	1	10	36,413	1,457	145,652
13	Subsystem Status Monitor 1.1.5								
14		25	38,000	*15,337*	0.5	10	23,005	920	92,022
15	Mission Phase Sequence Gen 1.1.6								
16		100	5,000	*26,196*	0.2	5	31,435	314	62,870
17	Data Reporting 1.1.7								
18		50	20,000	*17,283*	1	1	34,565	691	691,304
19	Nav/IMU 1.2								
20	IMU Processing 1.2.1								
21		100	3,200,000	*790,217*	0.2	***10***	948,261	9,483	948,261
22	GPS Processing 1.2.2								
23		100	3,200,000	*790,217*	0.2	***10***	948,261	9,483	948,261
24	Error Compensation 1.2.3								
25		300	25,000	*80,978*	0.5	1	121,467	405	404,891
26	Kalman Filtering 1.2.4								
27		25	3,500,000	*843,207*	0.5	36	1,264,810	50,592	1,405,344
28	Nav Exec 1.2.5								
29		100	1,000	*25,239*	1	6	50,478	505	84,130
30	FDI 1.2.6								
31		50	132,000	*44,065*	0.5	0.5	66,098	1,322	2,643,913
32	Bending Processing 1.2.7								
33		100	1,280	*25,306*	0.2	0.1	30,367	304	3,036,730
34	Adptve Gdnce 1.3								
35	Two-body Linear Guidance 1.3.1								
36		50	37,500	*21,467*	0.2	15	25,761	515	34,348
37	Non-Linear Traj Shaping 1.3.2								
38		1	3,300,000	*789,380*	0.2	960	947,257	947,257	986,726
39	Contingency Control 1.3.3								
40		*1*	1,000	*489*	1	1	978	978	978,261
41	Adaptive Control 1.4								
42	Model Reference Adaptive Controller 1.4.1								
43		50	505,100	*133,285*	0.25	10	166,606	3,332	333,212
44	Classical Autopilot 1.4.2								
45		50	542,350	*142,192*	0.5	10	213,289	4,266	426,577
46	System Identification 1.4.3								
47		25	2,144,525	*519,071*	0.25	10	648,839	25,954	2,595,356
48	System RM 1.5 (includes CSDL RM; task throughputs approximated as 50KIPS/3)								
49	Fault Response 1.5.1								
50		25	17,000	*10,315*	1	40	20,630	825	20,630
51	Fault Determine & Isolate 1.5.2								
52		25	17,000	*10,315*	1	40	20,630	825	20,630
53	Configuration Manager 1.5.3								
54		25	17,000	*10,315*	1	40	20,630	825	20,630
55	Gen Discretes 1.6								
56	Validate Commands 1.6.1								
57		50	4,200	*13,504*	1	2	27,009	540	270,087
58	Update Outputs 1.6.2								
59		50	4,000	*13,457*	1	2	26,913	538	269,130
60	Discrete Error Handling 1.6.3								
61		1	1,000	*489*	1	0.5	978	978	1,956,522
62	Det G&C Wnd Dltas 1.7 (no task throughputs available)								
63		*1*	*29,000*	*7,185*	*0.5*	*20*	***10,777***	10,777	
64	Load Relief Algorithm 1.7.1								
65		100	0	*25,000*	0	10	25,000	250	25,000
66	Estimate Fluctuation Stats 1.7.2								
67		100	0	*25,000*	0	10	25,000	250	25,000

Table 2-10. Central Control & Processing (Modified)

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thruput	Mod Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3	Winds Ahead Determination 2						Required		Required
4						sum:	348,823	max:	507,652
5		*1*	*520,000*	*124,598*	*0.5*	*500*			
6	Manage Measurement Resources 2.1								
7		*1*	*100*	*274*	*1*	*1*			
8	Lidar Cal and Checkout 2.1.1								
9		1	0	*250*	1	1	500	500	500,000
10	Lidar BITE 2.1.2								
11		25	6,000	*7,685*	0.5	1	11,527	461	461,087
12	Lidar Fault Handling 2.1.3								
13		25	0	*6,250*	0	1	6,250	250	250,000
14	Lidar Health Monitoring 2.1.4								
15		50	3,000	*13,217*	0.5	1	19,826	397	396,522
16	Compute Wind Profile 2.2								
17									
18	Lidar mode Control 2.2.1								
19		1	0	*250*	1	1	500	500	500,000
20	Winds Measurement Fitting 2.2.2								
21		1	520,000	*124,598*	0.2	950	149,517	149,517	157,387
22	Vibration Compensation 2.2.3								
23		50	5,000	*13,696*	0.5	1	20,543	411	410,870
24	Bending Compensation 2.2.4								
25		50	10,000	*14,891*	1	5	29,783	596	119,130
26	Control Velocimeter 2.3								
27		*100*	*2,000*	*25,478*	*1*	*1*	50,957	510	
28	Redundant Lidar Configuration Control 2.3.1								
29		100	0	*25,000*	0	10	25,000	250	25,000
30	Lidat Power Control 2.3.2								
31		25	0	*6,250*	0	40	6,250	250	6,250
32	Receive and Process Winds Info 2.4								
33		*50*	*800*	*12,691*	*1*	1	25,383	508	507,652
34	Detection 2.4.1								
35		1	0	*250*	1	1	500	500	500,000
36	Pulse Deconvolution 2.4.2								
37		1	0	*250*	1	1	500	500	500,000
38	Range Determination 2.4.3								
39		1	0	*250*	1	1	500	500	500,000
40	Doppler Frequency Estimation 2.4.4								
41		1	0	*250*	1	1	500	500	500,000
42	Data Collection & Formatting 2.4.5								
43		1	600	*393*	1	10	787	787	78,696
44	Check Range w/ Exp Values 2.4.6								
45		1	500		0.2	5	0	0	0

Table 2-11. Winds Ahead Determination (Modified)

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thru	Mod Thru	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3	Vehicle Power System Management 3						Required		Required
4						sum:	18,025	max:	500,000
5		25	4,600	7,350	0.5	2	11,025	441	220,500
6	Battery Charge Mgmt 3.1								
7		1	0	250	1	1	500	500	500,000
8	Charge Mgmt Exec 3.1.1								
9		1	0	250	1	1	500	500	500,000
10	Control Charging 3.1.2								
11		1	0	250	1	1	500	500	500,000
12	Monitor Charge 3.1.3								
13		1	0	250	1	1	500	500	500,000
14	Error Detection & Status Determination 3.2								
15		1	0	250	1	1	500	500	500,000
16	Pwr System Load Error Det 3.2.1								
17		1	0	250	1	1	500	500	500,000
18	Pwr Emergency Mgr 3.2.2								
19		1	0	250	1	1	500	500	500,000
20	Load SHedding 3.3								
21		1	0	250	1	1	500	500	500,000
22	Dynamic Load Distribution 3.4								
23		1	0	250	1	1	500	500	500,000
24	Pwr System Load Control 3.4.1								
25		1	0	250	1	1	500	500	500,000
26	Mission Power Profile 3.4.2								
27		1	0	250	1	1	500	500	500,000
28	Change Load Distribution 3.4.3								
29		1	0	250	1	1	500	500	500,000
30	Power-Up Initialization 3.4.4								
31		1	0	250	1	1	500	500	500,000
32	Monitor Load & Predict Deplete 3.4.5								
33		1	0	250	1	1	500	500	500,000

Table 2-12. Vehicle Power System Management (Modified)

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thru	Mod Thru	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)	(IPS)		Lag (ms)	Thruput		Thruput (IPS)
3							Required		Required
4	Steering & Staging Control 4					sum:	73,075	max:	505,435
5		50	55,000	25,652	0.5	15	38,478	770	51,304
6	Provide Steering Signals 4.1								
7		1	0	250	0	1	250	250	250,000
8	Identify Changed Cards 4.1.1								
9		1	0	250	0	1	250	250	250,000
10	Com Cmd to Prev & Limit Rate 4.1.2								
11		1	0	250	0	1	250	250	250,000
12	Gen New Steering Signal 4.1.3								
13		1	0	250	0	1	250	250	250,000
14	Report Steering Cmd Error 4.1.4								
15		1	0	250	0	1	250	250	250,000
16	Validate Steering Cmd Response 4.1.5								
17		1	0	250	0	1	250	250	250,000
18	Select Outputs and Verify Cmds 4.2								
19		50	5,000	13,696	1	4	27,391	548	136,957
20	Validate Discretes 4.2.1								
21		0.4	1,000	339	1	4	678	1,696	423,913
22	Activate Staging 4.3								
23		1	0	250	0	1	250	250	250,000
24	LFU Control & Monitor 4.3.1								
25		1	0	250	0	1	250	250	250,000
26	Verify & Sequence Power 4.3.2								
27		1	1	250	0	1	250	250	250,239
28	Ordnance BIT 4.3.3								
29		1	6,000	1,685	0.5	5	2,527	2,527	505,435
30	Fault handling 4.3.4								
31		1	1	250	0	1	250	250	250,239
32	LFU Redundancy Control 4.3.5								
33		1	0	250	0	1	250	250	250,000
34	Activate Discrete Devices 4.4								
35		1	0	250	0	1	250	250	250,000
36	Provide Discrete Outputs 4.4.1								
37		1	0	250	0	1	250	250	250,000
38	RCS Control 4.5								
39		1	0	250	0	1	250	250	250,000
40	RCS Error Generate 4.5.1								
41		1	0	250	0	1	250	250	250,000
42	Phase Plane Control 4.5.2								
43		1	0	250	0	1	250	250	250,000
44	RCS Command Generator 4.5.3								
45		1	0		0	1	0	0	0

Table 2-13. Steering & Staging Control (Modified)

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thru	Mod Thru	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)	(IPS)		Lag (ms)	Thruput Required		Thruput (IPS)
3	Sensor Processing 5					SUM:	753,335	Max:	516,685
4									
5		25	636,000	158,337	0.5	34	237505	9,500	
6	Prescale & Categorize Sensor Data 5.1								
7		25	12,500	9,239	0.5	34			
8	Identify Format 5.1.1								
9		50	130,000	43,587	0.5	40	65380	1,308	32,690
10	Filter & Store 5.1.2								
11		50	812,000	206,674	1	16	413348	8,267	516,685
12	Format-A Convert & Store 5.1.3								
13		25	5,000	7,446	1	40	14891	596	14,891
14	Format-B Convert & Store 5.1.4								
15		1	1	250	0	1	250	250	250,239
16	Type Processing 5.2								
17		50	90,000	34,022	1	5	68043	1,361	272,174
18	Compare With Limits 5.2.1								
19		1	0	250	0	1	250	250	250,000
20	Averaging & Voting 5.2.2								
21		1	0	250	0	1	250	250	250,000
22	Count Limits Errors 5.2.3								
23		1	0	250	0	1	250	250	250,000
24	Count Limits Errors 5.2.4								
25		1	0	250	0	1	250	250	250,000
26	Select Redundancy 5.2.5								
27		1	0	250	0	1	250	250	250,000
28	Calibration & Validation 5.3								
29		25	112,500	33,152	0.5	5	49728	1,989	397,826
30	Determine Criticality 5.3.1								
31		1	0	250	0	1	250	250	250,000
32	Manage Recovery 5.3.2								
33		1	0	250	0	1	250	250	250,000
34	Sensor Fault Tolerance 5.4.0								
35		75	308,500	92,522	0.5	5	138783	1,850	370,087
36	Limit Checks 5.4.1								
37		1	0	250	0	1	250	250	250,000
38	Consistency Checks 5.4.2								
39		1	0	250	0	1	250	250	250,000
40	Validation Error Report 5.4.2								
41		1	0	250	0	1	250	250	250,000
42	Error & Failure Reporting 5.5								
43		1	100	274	0.5	5	411	411	82,174

Table 2-14. Sensor Processing (Modified)

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thruput	Mod Thruput	Margin	Proc.	Margined	Ins/Exec	Instant
2		Rate (Hz)	(IPS)	(IPS)		Lag (ms)	Throughput		Thruput (IPS)
3	Propulsion Control 6						Required		
4						SUM:	4,807,609	Max:	6,410,145
5		50	10,000,000	2,403,804	1	15	4,807,609	96,152	6,410,145
6	Manage Propulsion and Fault Tolerance 6.1								
7		1	0	250	0	1	250	250	250,000
8	Prop Cntrlr Timing & Exec 6.1.1								
9		1	0	250	0	1	250	250	250,000
10	Prop Memory Mgmt 6.1.2								
11		1	0	250	0	1	250	250	250,000
12	Prop Fault Mgmt 6.1.3								
13		1	0	250	0	1	250	250	250,000
14	Update Data Tables & Report 6.2								
15		1	0	250	0	1	250	250	250,000
16	Report Engine Status 6.2.1								
17		1	0	250	0	1	250	250	250,000
18	Update Table 6.2.2								
19		1	0	250	0	1	250	250	250,000
20	Process Commands 6.3								
21		1	0	250	0	1	250	250	250,000
22	Monitor and Verify Commands 6.3.1								
23		1	0	250	0	1	250	250	250,000
24	Compute Valve Position 6.4								
25		1	0	250	0	1	250	250	250,000
26	Engine Control Loops 6.4.1								
27		1	0	250	0	1	250	250	250,000
28	Update Actuator Positions 6.4.2								
29		1	0	250	0	1	250	250	250,000
30	Compare Actuator Model 6.4.3								
31		1	0	250	0	1	250	250	250,000
32	Process Propulsion Sensors 6.5								
33		1	0	250	0	1	250	250	250,000
34	Sensor Data Scaling 6.5.1								
35		1	0	250	0	1	250	250	250,000
36	Redun Sensor Process and Qual 6.5.2								
37		1	0	250	0	1	250	250	250,000
38	Limit Monitor & Failure Detect 6.5.3								
39		1	0	250	0	1	250	250	250,000
40	Sensor Failure Handler 6.5.4								
41		1	0	250	0	1	250	250	250,000

Table 2-15. Propulsion Control (Modified)

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thruput	Mod Thruput	Margin	Proc.	Margined	ns/Exec	Instant
2		Rate (Hz	(IPS)	(IPS)		Lag (ms	Throughput		Thruput
3							Required		(IPS)
4	Command & TLM Processing 7					SUM:	134,918	max:	250,000
5		*25*	*350,000*	*89,946*	*0.5*	*40*	134,918	*5,397*	*134,918*
6	Decode & Process Command Data 7.1								
7		*1*	*0*	*250*	*0*	*1*	250	*250*	*250,000*
8	Decode & Verify Commands 7.1.1								
9		*1*	*0*	*250*	*0*	*1*	250	*250*	*250,000*
10	Validate Ground Data 7.1.2								
11		*1*	*0*	*250*	*0*	*1*	250	*250*	*250,000*
12	Command Acceptance 7.1.3								
13		*1*	*0*	*250*	*0*	*1*	250	*250*	*250,000*
14	TLM Table Manager 7.2								
15		*1*	*0*	*250*	*0*	*1*	250	*250*	*250,000*
16	Format Telemetry 7.2.1								
17		*1*	*0*	*250*	*0*	*1*	250	*250*	*250,000*
18	TLM Transit 7.2.2								
19		*1*	*0*	*250*	*0*	*1*	250	*250*	*250,000*
20	Control Telemetry Format 7.3.1								
21		*1*	*0*	*250*	*0*	*1*	250	*250*	*250,000*

Table 2-16. Command & TLM Processing (Modified)

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thruput	Mod Thruput	Margin	Proc.	Margined	ns/Exec	Instant
2		Rate (Hz)	(IPS)	(IPS)		Lag (ms)	Thruput		Thruput (IPS)
3							Required		
4	Range Safety & Destruct 8					SUM:	1,666	max:	555,435
5		*1*	*3,600*	*1,111*	*0.5*	*3*	*1666*	*1,666*	*555,435*
6	Band Transpond 8.1								
7		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
8	Receive & Decode Signal 8.1.1								
9		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
10	Self-Test & timing 8.1.2								
11		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
12	Transmit C-Band Signal 8.1.3								
13		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
14	Command Receive & Decode 8.2								
15		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
16	Destruct Command Logic 8.2.1								
17		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
18	Select Power Source 8.2.2								
19		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
20	Sequece Destruct Signals 8.2.3								
21		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
22	Destruct System Control 8.3								
23		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
24	Safe & Arm ISDS 8.3.1								
25		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
26	Verify Loop Integrity 8.3.2								
27		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
28	ISDS Redundancy Manager 8.3.3								
29		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*
30	Initiate Destruct 8.3.4								
31		*1*	*0*	*250*	*0*	*1*	*250*	*250*	*250,000*

Table 2-17. Range Safety & Destruct (Modified)

	A	B	C	D	E	F	G	H	I
1		Frame	MM Thruput	Mod Thruput	Margin	Proc.	Margined	ins/Exec	Instant
2		Rate (Hz)	(IPS)	(IPS)		Lag (ms)	Thrupt		Thruput (IPS)
3							Required		
4	Programmable Payload I/F 9								
5		1	0	250	0	1	250	250	

Table 2-18. Programmable Payload I/F (Modified)

Based on the throughput estimates discussed above it was concluded that all ALS functions except propulsion control can be performed by one FTP. (See Section 4.1 for a more detailed discussion). For brevity we denote this FTP the "Core FTP." According to [5] the propulsion controllers (also assumed to be FTPs) are assumed to reside at the engines; denote these as the "Propulsion FTPs." This functional partitioning guided the interpretation of the communication requirements. Specifically, in analyzing the communication requirements only the communications between the Core FTP and the vehicle sensors and actuators, the Core FTP and the Propulsion FTPs, and the Propulsion FTPs and the propulsion sensors and actuators were considered. Moreover, functions which did not possess communications requirements or had numerically insignificant requirements were excluded from this analysis.

Determination of temporal load profiles requires detailed knowledge of task and communication request scheduling, and this information is currently unavailable. Therefore the figures obtained from the requirements are at best average figures and primarily of use only in performing a rough sizing of the communications media, and to predict the average utilization of the media.

Using the assumptions and simplifications noted above, the following communication requirements were determined.

2.2.2.1 Core FTP I/O

This category comprises input from the vehicle's sensors to the Core FTP and output from the Core FTP to the vehicle's actuators. This I/O would take place over one or more AIPS I/O Networks or through memory-mapped I/O devices resident on the Core FTP's Shared or Private Bus.

Winds Information: "The winds information signal represents the echo returns from the electromagnetic probe signals emanated from the winds measurement signal. It consists of sixty points, obtained every sixth of a second [4]." Each sample is 16 bits. The average bandwidth required for this I/O function is thus

60 points/sample*16 bits/point*60 samples/sec = 56,470 bits/sec

Sensor Signals: This communication path consists of "...2500 non flight critical sensors and 600 flight critical sensors, or 3100 sensors. We assume most are sampled at 50 Hz rate [4]." Each sample is 16 bits. We assume that the non flight critical sensors are

simplex and the flight critical sensors are duplex (fail-operational), resulting in 2500 + 1200 = 3700 sensor reads per iteration. The resultant average bandwidth required is

3700 points/sample * 16 bits/point * 50 samples/sec = 2,960,000 bits/sec

Ground Downlink: "The ground downlink contains the telemetry stream which is normally routed through the radio link [4]." The average bandwidth required for this function is given directly by the requirements to be

8,400,000 bits/sec

Steering Signals: "Two actuator signals of 16 bits each are sent to up to 17 engines...(at a) 25 Hz rate [4]." The bandwidth required for this communication is

2 signals/engine * 17 engines * 16 bits/signals * 25 outputs/sec = 13,600 bits/sec

The aggregate average ALS Core FTP I/O bandwidth requirement is 11,160,340 bits/sec.

2.2.2.2 Core FTP - Propulsion FTP Communications

This category comprises communication between the Core FTP and the Propulsion FTP(s). This communication would most likely take place over the InterComputer Network.

Propulsion Commands: "The propulsion interface is fairly simple. Command examples: Engine Preparation, Engine Start, Engine BIT, Engine Shutdown. Assume there (are) at most 16 commands (resulting an a 4-bit command word) [4]." The output rate is 50 Hz. The bandwidth required for this function is

1 command/engine * 17 engines * 4 bits/command * 50 outputs/sec = 3,400 bits/sec

Propulsion Status: This signal is not described in [4] but is referenced in an earlier informal requirements document and corroborated via communication with Martin Marietta. It represents the propulsion control transmitting propulsion status to the command and telemetry functions, and is of sufficient magnitude to be considered here. This signal requires a bandwidth of

10,000,000 bits/sec

Total Engine Status: The propulsion control periodically reports the engine status to the Central Control and Telemetry functions [4]. Each engine possesses 100 bits of status information and the output is scheduled at a 25 Hz rate, resulting in a bandwidth of

100 bits/engine * 17 engines * 25 outputs/sec = 42,500 bits/sec

It is not known whether the Total Engine Status signal supplants the Propulsion Status signal described above.

Propulsion Memory Dump: The propulsion control function can be commanded to dump its memory contents to the Command and Telemetry processing. This is normally done only at prelaunch. The bandwidth requirements are given as

16,000,000 bits/sec

The aggregate average bandwidth required for communication between the ALS Core FTP and the ALS Propulsion FTP(s) is 10,045,900 bits/sec during flight and 26,045,900 bits/sec during prelaunch propulsion memory dumping.

2.2.2.3 Propulsion FTP I/O

The ALS Propulsion FTPs possess interfaces to the engine sensors and actuators. This communication would take place over one or more regionally partitioned I/O Networks, or through memory-mapped I/O devices resident on the FTPs' Shared and/or Private Bus.

Propulsion Sensor Signals: "There are 357 flight critical sensors, reporting at 25 and 50 Hz rates, with up to 16 bits accuracy [4]." We assume that this sensor complement suffices for 17 engines. While it is suggested in [4] that the worst-case of 50 Hz and 16 bits be used for all sensors, further conversations with Martin Marietta suggest that approximately 20 sensors are sampled at a 1000 Hz rate. Moreover, since propulsion sensors are flight critical, they are at least duplex, resulting in a total number of sensor reads of 337*2 = 674 per 50 Hz sample and 20*2=40 per 1000 Hz sample. The total bandwidth required is

674 points/sample * 16 bits/point * 50 samples/sec
+
40 points/sample * 16 bits/point * 1000 samples/sec = 1,179,200 bits/sec

Propulsion Actuator Controls: The propulsion control transmits actuator commands to the engines via this communication path. The requirements are obscure but appear to comprise 4 to 5 fail-operational/fail-operational (triplex) actuators per engine, each of which is provided with a 16-bit output value at an assumed 100 Hz rate. We assume that each engine therefore requires 15 outputs. Under these assumptions the bandwidth requirement is

15 actuators/engine * 17 engines * 16 bits/point * 100 samples/sec = 408,000 bits/sec

The aggregate average ALS Propulsion FTP I/O bandwidth requirement is 1,587,200 bits/sec.

2.2.2.4 Summary of I/O and Interfunction Communication Requirements

The aggregate bandwidth requirements for the three ALS categories are depicted in Tables 2-19. Figure 2-4 depicts a functional partitioning of the ALS virtual architecture along with the I/O and Interfunction communication bandwidth requirements.

Category	Bandwidth, bits/sec
Core FTP I/O	11,160,340
Core FTP - Propulsion FTP Communications	
prelaunch	26,045,900
flight	10,045,900
Propulsion FTP I/O	1,587,200

Table 2-19. Aggregate I/O and Interfunction Communication Bandwidths for ALS

2.3. RMA and Environmental Requirements

2.3.1. Reliability and Availability

The ALS mission scenario and Avionics Reliability and Availability requirements are summarized in the following paragraphs.

The ALS mission scenario comprises four phases, each of which has different RMA parameters. These phases are: (1) integration and checkout in a facility such as a Vehicle Assembly Building, (2) a period of extended launch pad residence, (3) the launch or the boost phase, and (4) on-orbit operation.

During vehicle integration the ALS avionics must be verified to be in a nonfaulty state prior to roll out to the launch pad. This implies a degree of testability which has not been quantified or modeled in the current study but is expected to be high because of the characteristically high diagnosability of the AIPS Byzantine resilient approach to fault tolerance.

The ALS remains on the pad in a condition of launch readiness for a period of up to one week, during which no repair or maintenance of the avionics is desirable to avoid costs associated with such actions. A "Launch with Faults" policy is assumed to be in effect such that the ALS can be launched with avionics system faults, but only if the avionics system is known to be capable of fault masking at the time of launch. The availability requirement used for the current study is that the ALS avionics system must have a 95%

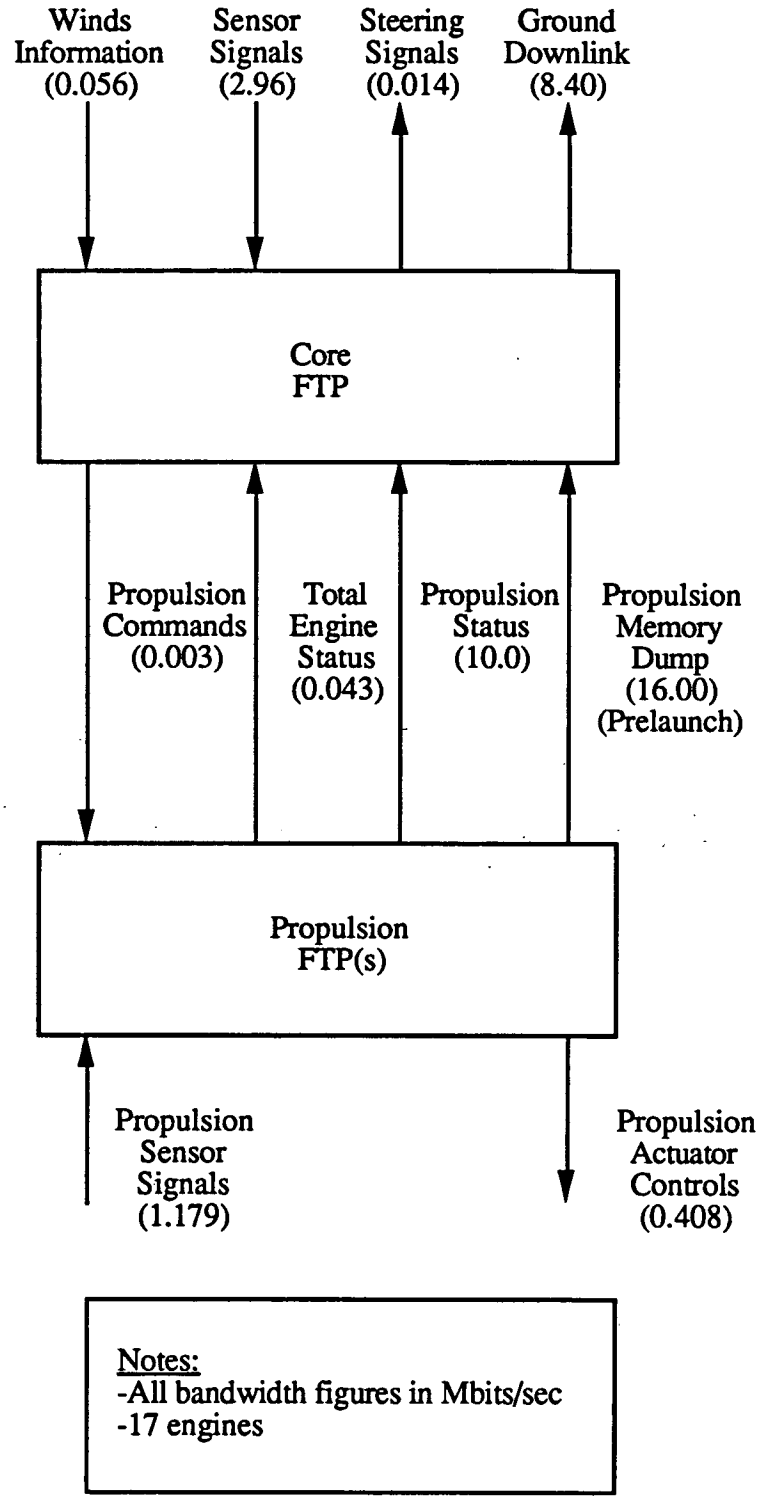

Figure 2-4. Functional Partitioning and I/O and Interfunction Communication Bandwidths for the ALS

availability, i.e., must be fault masking with a probability of 0.95, at the end of one week on the launch pad. The AIPS configured for the ALS consists of a number of triply or quadruply-redundant FTPs connected by a triply or quadruply-redundant InterComputer Network. The FTP complement is defined to be fault masking if and only if each FTP is capable of correct operation in the presence of any single active Byzantine fault. Representative fault masking FTP configurations are a triplex FTP which has suffered zero permanent faults and is not in the process of recovering from a transient fault, and a quadruplex FTP which has suffered, detected, and masked out the channel containing any single permanent fault (becoming in effect a fault-free triplex FTP), and is not in the process of recovering from a transient fault. A duplex FTP is not fault masking. The InterComputer Network is defined to be fault masking if and only if every FTP can transmit a sufficient number of uncorrupted copies of a message to every other FTP in the avionics suite to allow voting to generate a correct copy in the presence of any single active Byzantine fault in the InterComputer Network. Representative fault masking InterComputer Network configurations are manifold and include the case of a permanent-fault-free triplex IC Network and a quadruplex IC Network which has suffered a single permanent Byzantine fault and successfully detected the existence of that fault, in effect becoming a fault-free triplex IC Network.

The launch or the boost phase, i.e., the powered flight segment, is specified to be nominally of ten minutes duration. Finally, the launch vehicle enters the desired earth orbit. The amount of time in the orbit until the payload is positioned and released may vary. In the worst case, it may extend to as long as 48 hours. The maximum allowable probability of mission or vehicle loss due to avionics failure for the boost phase and the on-orbit operations is assumed to be 10^{-5}. Because launch occurs only if the avionics system is known to be capable of masking a single fault, no single fault can cause the loss of the ALS avionics during launch. However, during launch the avionics may fail either due to near-coincident multiple faults occurring in a subsystem designed to tolerate only one fault at a time, or by the exhaustion of redundant modules in a subsystem such that the subsystem can no longer perform its computational or communication functions. An example of the former failure mode is the occurrence of near-coincident error bursts on two layers of a triplex InterComputer Network. An example of the latter is a series of sequential covered faults resulting in the loss of the computational services of an FTP.

The operational environment for the ALS avionics on the launch pad, in the boost phase, and on-orbit is radically different. Different failure rates of components must be taken into account to model the system reliability and availability accurately. The avionics module failure rates for the three ALS mission phases are tabulated in Section 3.

2.3.2 Maintainability

One of the assumptions regarding the operation of the ALS is that launch pad avionics maintenance should be avoided as far as possible. However, in the 5% of the

launches (the "unavailability" requirement) the avionics would have suffered too many faults on the launch pad to provide a fault-masking capability, and it will become necessary to perform maintenance while the vehicle is on the launch facility. The avionics must be packaged in relatively small units and those units must be located in the vehicle where removal and replacement of failed components is facilitated. The Line Replaceable Module (LRM) maintenance philosophy adopted by the Joint Integrated Avionics Working Group (JIAWG) is one alternative for the maintenance approach.

The JIAWG has defined support systems and support techniques for vehicles which provide easy access to the equipment locations. Much of the JIAWG maintenance philosophy can still be used for ALS even though physical access to the ALS Line Replaceable Units (LRUs) will probably be not as easy as in an aircraft.

The modules within the ALS LRUs can be standardized on the Standard Electronic Module format E (SEM-E) form factor in accordance with MIL-STD-1389D. These modules, depicted in Figure 2-5, are designed for conduction cooled applications and can be used in extremely harsh environments (temperature, acceleration, shock and vibration).

2.3.3. Component Quality

Non-fault tolerant, single-string systems for use in space-borne systems must be designed such that the probability of failure is sufficiently small so as to "guarantee" the absence of failures during the useful lifetime of the avionics. This level of system integrity is accomplished by creating and maintaining a "pedigree" for each item used in the flight article. A system's pedigree begins with the use of components which have a traceable history. The components are manufactured on well controlled assembly lines where each phase of the manufacturing process is reviewed, inspected and certified. Electronic microcircuits are manufactured, inspected and tested in accordance with MIL-STD-883C, Class S. Discrete components such as resistors, capacitors and transistors are subject to equally demanding manufacturing and testing controls (e.g., the JAN-TX quality controls).

The individual components are then combined to create a subassembly and subassemblies are combined to build the components of the system where each phase of the manufacturing and assembly process has appropriate inspection and test requirements.

The AIPS, a fault tolerant system, does not require the "pedigree" constraints required for single-string systems. The ability to ensure the placement of a payload in the proper orbit is achieved through the redundancy of the system. Since failures are tolerated, individual components need not be MIL-STD-883C, Class S, or JAN-TX quality. A comparative cost analysis, described in Section 5 of this report, shows that an effective alternative to these components is the use of Class B and JAN-T components provided there is a means for properly handling faults.

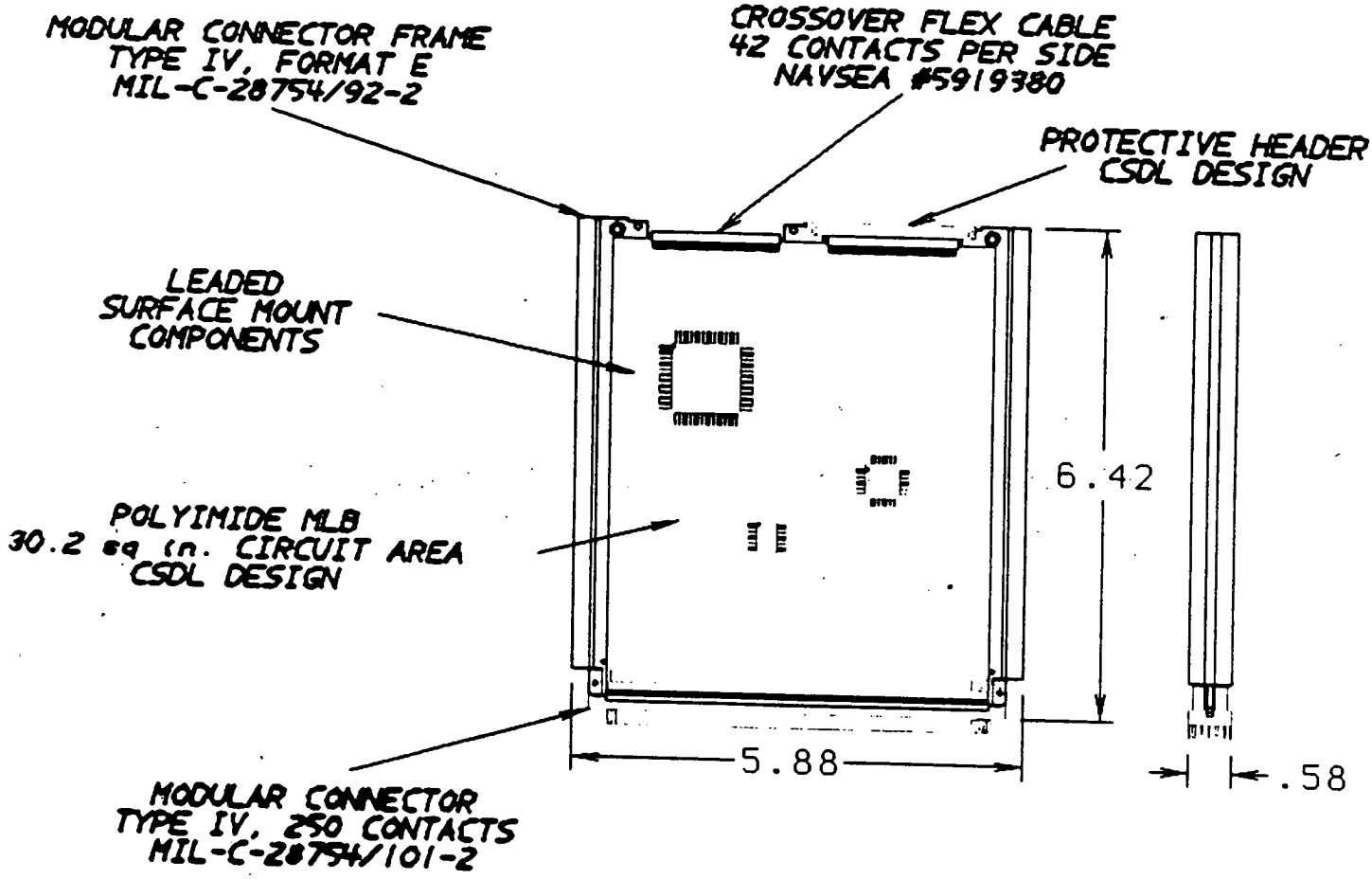

Figure 2-5. SEM-E Module

2.3.4. Radiation Hardness

A second issue for an ALS avionics suite is radiation hardness. Many space-borne systems must be radiation hard where the definition for hardness is determined by the purpose of the avionics. The significant difference between the ALS avionics hardness and the hardness of an arbitrary payload is the operational lifetime of the two systems.

A payload boosted to orbit by the ALS may need to be fully operational in a relatively high radiation environment for time periods exceeding ten years. The ALS mission, however, is complete after it has placed the payload in a specific orbit within a few hours after launch, typically 2 hours but in any case no more than 48 hours. Due to the extremely short mission times, the ALS radiation hardness is not constrained by a technology's total dose capability. The hardness of the ALS avionics will be constrained by the Single Event Upset (SEU) rate of the logic family. The AIPS architecture can tolerate transients caused by SEUs in the same manner that it can tolerate transient faults. However, if the radiation environment and the corresponding SEU rate is sufficiently high it can overwhelm the architecture's ability to tolerate these transients. Therefore, technologies which are less susceptible to SEUs are preferable even for the AIPS building blocks.

Upset rates for the CMOS, Bipolar and GaAs logic families have enjoyed continuous improvements in radiation tolerance during the time period of 1985 through 1990 and it is reasonable to expect continued improvements through 1993. With the total dose and transient dose improvements, both demonstrated and projected, the ALS avionics will not present significant problems with radiation tolerance.

2.3.5. Power Dissipation

Operation of the ALS avionics at a launch facility can be supported by auxiliary cooling. This cooling could be forced air or a recirculated liquid such as ethylene glycol. The avionics must also be capable of operating without degradation after support equipment has been withdrawn as well as during the launch and the 48 hours allocated for on-orbit maneuvers. System cooling capacity, therefore, will not be determined by the launch pad environment but by the ability to cool the avionics while in space.

The avionics cooling system for use in space will be comprised of one or more "cold plates", used to mount the avionics assemblies, and the cold plates will, in turn, be thermally connected to radiators via heat pipes. The radiators will be used to radiate the heat dissipated by the avionics into space. The thermal capacity of these cooling systems is severely limited and the power dissipation of the avionics must be minimized. Typical power capacities of radiative cooling systems are two to five Watts per square foot of the radiator area. Therefore, the avionic system power dissipation must be minimized.

Power dissipation minimization for the ALS avionics can be accomplished in three ways. The first approach is to use a non-saturating logic family such as CMOS. Other logic families such as bipolar are saturating logic and they are characterized by significant

power dissipation at all frequencies of operation. The second approach is the minimization of the clock frequency of the system. CMOS logic, while it is not a saturating logic family, does dissipate power during state transitions. The third approach for power minimization is the reduction of the bias voltage on the integrated circuits.

The total power dissipation of a CMOS-based system can be estimated by the following expression:

$$P_t=(C_{pd}f_i+C_Lf_o)V_{cc}^2 \qquad (1)$$

where

P_t = total power dissipated by the system

C_{pd} = gate capacitance of the devices

f_i = internal frequency of operation

C_L = external load capacitance

f_o = frequency of output signals

V_{cc} = supply voltage

As shown in the above expression, power dissipation is directly proportional to the frequency of operation, both internal and external, and the square of the supply voltage. The LRUs for the ALS avionics are described in Section 3.5.

2.4 Requirements Conclusions

The overall ALS mission scenario and RMA requirements were obtained from the three ALS prime contractors via Martin Marietta Astronautics Group. These included the ALS mission phases and durations, the launch availability and the probability of mission success.

The ALS computational and communication requirements were obtained from Martin Marietta Astronautics Group in the form of directed cyclic graphs depicting interfunction data dependencies and Hypercard stacks depicting numerical throughput and interfunction communication requirements. The requirements were presented as a three-level hierarchy.

Nine top-level functions reside at the top of the depiction hierarchy: Central Control and Processing, Winds Ahead Determination, Vehicle Power System Management, Steering and Staging Control, Propulsion Control, Command and Telemetry Processing, Range Safety and Destruct, and Programmable Payload Interface. Aggregate throughput estimates were given for many of these functions. These aggregates were observed to comprise an overestimation of the computational requirements of their constituent functions. The third level of the hierarchy represents atomically schedulable computational tasks. Currently, Central Control and Processing is the only function for which level-three task requirements are available.

Based on the data provided, the overall ALS throughput requirements were estimated to be approximately 8.8 MIPS for non-propulsion functions and 4.8 MIPS per engine for propulsion functions. The inter-FTP communication bandwidth requirements were estimated to be 26 Mbits/sec (prelaunch) between the Core FTP and the Propulsion FTP(s) (17 engines), the Core FTP I/O bandwidth requirement was estimated to be 11.2 Mbits/sec, and the Propulsion FTP(s) I/O bandwidth requirement was estimated to be 1.587 Mbits/sec (17 engines).

To more accurately estimate the throughput and bandwidth needs of the ALS avionics system the following information is desirable for each task: task frame or iteration rate, instructions per frame, throughput margin, processing lag, scheduling requirements (i.e., preemptible or nonpreemptible), task execution order and/or data dependencies, and inter-function communication requirements (i.e., bits per frame, source or destination of data, and latency requirements). The data presented by Martin Marietta served as an important starting point for determining the requirements of the ALS avionics. During the course of the CSDL-Martin Marietta interaction an active dialogue was set up which would have in time resulted in a more complete definition of a common requirements vocabulary and facilitated the acquisition of comprehensive requirements data.

3.0 AIPS ARCHITECTURE OVERVIEW

Over a period of about six years, the Draper Laboratory has been developing a fault tolerant distributed computer system architecture suitable for advanced launch vehicles. The overall program objective of the Advanced Information Processing System (AIPS) Program, has been to produce the knowledgebase which will allow achievement of validated fault tolerant distributed computer system architectures suitable for a broad range of aerospace vehicles. The architecture that has been conceived to meet these requirements is based on the notion of prevalidated building blocks.

The AIPS architectural attributes, rules and guidelines, and reliability and performance models of the building blocks are described in detail in an accompanying report "Advanced information Processing System: Design and Validation Knowledgebase" [1]. It also contains references to the more detailed hardware and software specifications and simulations that constitute the AIPS knowledgebase. This knowledgebase, which is quite large, is required to synthesize a validated ALS avionics architecture. The following subsections briefly recapitulates the AIPS virtual and physical architectures and the key attributes of the hardware and software building blocks.

3.1 Building Blocks

AIPS is a multicomputer architecture composed of hardware and software building blocks that can be configured according to certain design rules and guidelines to meet the specific requirements of a given application.

The hardware building blocks, as shown in Figure 3-1, are Fault Tolerant Processors (FTPs), Networks, and Interfaces. The FTPs are general purpose computers which can be built in varying redundancy levels from simplex to quadruplex, using one to four identical channels, to meet varying levels of reliability requirements. The networks are communication media and are composed of circuit-switched nodes linked together with full duplex links. The networks can be configured in various topologies such as a ring, braided mesh, irregular mesh, etc. Networks can also be made redundant. Networks are used to connect FTPs to input/output devices (these are called I/O networks) and to other FTPs (these are called Inter-Computer or IC networks). I/O and IC networks are built out of identical nodes and links. The interfaces are the building blocks that are used to interface a channel of the FTP to an I/O Network, called the I/O sequencer or IOS, and to the IC Network, called the IC Interface Sequencer or ICIS.

The software building blocks are the major software functions: local system services, input/output system services, inter-computer system services and the system manager. This software provides the services necessary in a traditional real time computer such as task scheduling and dispatching, communication with sensors and actuators, etc. The software also supplies the redundancy management services necessary in a redundant

computer and the services necessary in a distributed system such as inter-function communication across processing sites, management of distributed redundancy, management of networks, and migration of functions between processing sites.

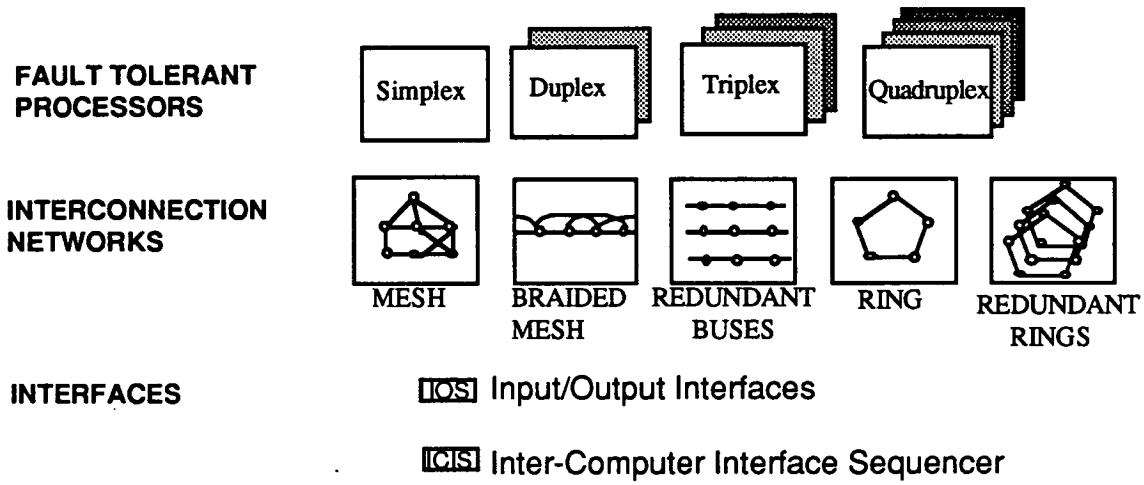

Figure 3-1. AIPS Hardware Building Blocks

3.2 Virtual Architecture

One of the important and unique attributes of the AIPS and other Draper-designed fault tolerant computers is that redundancy and its management are transparent to applications software. Furthermore, even most of the system software is unaware of the redundancy of the underlying hardware on which it executes. The only exceptions are those system services that are directly responsible for managing the redundancy. (They have to know about the existence of redundant hardware in order to manage it.) This AIPS attribute allows almost all of the software to be developed and validated on a simplex processor, in a software development environment familiar to most programmers and using mature tools. We call the architecture, as it appears to the programmer, the virtual architecture.

The AIPS virtual architecture is a conventional multicomputer architecture as shown in Figure 3-2. It consists of a number of processing sites each containing an FTP and the necessary external interfaces. The processing sites are linked together by an Inter-Computer or IC bus. An FTP at any particular processing site may also have access to varying numbers and types of I/O buses, which are separate from the IC bus. Separate buses to carry sensor data and intercomputer data are provided because the bandwidth and reliability requirements for these two classes of data in most realtime systems are very different. The I/O buses may be global, regional or local in nature. I/O devices on the global I/O bus are available to all, or at least a majority, of the AIPS FTPs. Regional buses

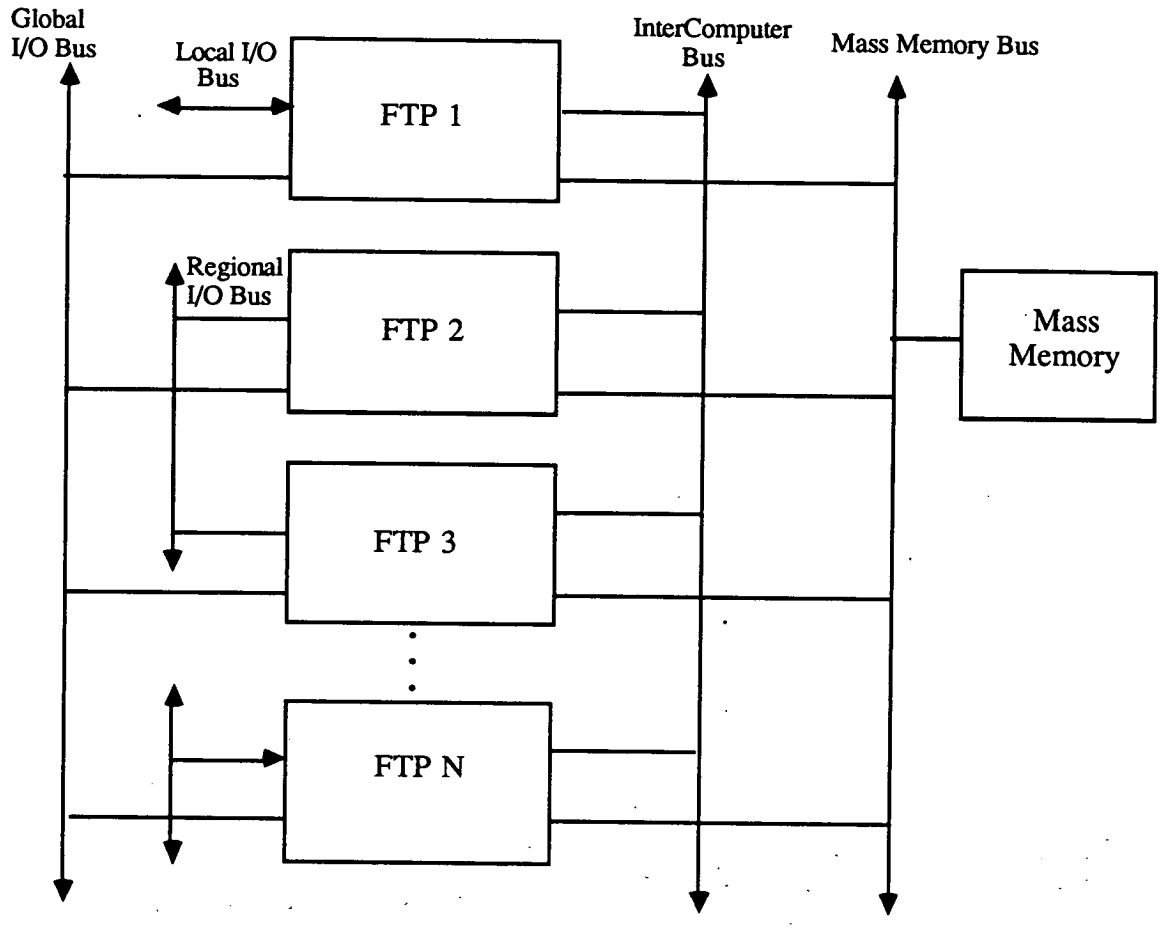

Figure 3-2. AIPS Virtual Architecture

connect I/O devices in a given region to the processing sites located in their vicinity. Local buses connect an FTP to the I/O devices dedicated to that computer. Additionally, devices may be connected directly to the internal bus of a processor and accessed as though the I/O devices reside in the computer memory (memory mapped I/O). The regional and global buses allow sharing of raw sensor data among functions that reside on different processing sites, thus reducing the overall system cost. They also allow functions to be migrated between FTPs in real time in the event of faults, damage or change in mission phase and work load. The memory mapped I/O is used to access time critical sensors to meet stringent transport lag requirements for real time control applications. (Transport lag is the time elapsed from reading a set of sensors to asserting an actuator command in response to that input).

The virtual architecture of a processing site is shown in Figure 3-3. It consists of three sections: a computational section, an I/O section, and the resources shared between them. The computational and I/O sections are identical, conventional processor architectures. Each consists of a processor, memory, interval timers and memory mapped I/O (which is unique to each processor). Although identical in hardware design, the

computational processor or CP is typically devoted to application functions (such as executing vehicle control law) while the I/O processor or the IOP is devoted to I/O functions such as reading and validating sensors and sending out actuator commands. The CP and IOP communicate with each other via the shared memory. Other resources shared by both processors include a data exchange mechanism which is used to exchange and vote data with other redundant channels in this FTP, a real time clock and interfaces to several I/O buses and the IC bus.

3.3 Physical Architecture

The parameters that define the physical architecture include redundancy levels of FTPs, interconnections of redundant channels in an FTP, redundancy level of sensors, actuators and other I/O devices, cross-strapping of I/O devices to channels of FTPs and redundancy level of their interfaces, redundancy levels of IC and I/O networks and their physical topologies. This section highlights some of the salient points of the architecture.

Figure 3-4 shows the physical architecture of the quad redundant AIPS FTP. It is designed strictly according to the fundamental fault tolerance theory. It complies with all the requirements for tolerating two sequential Byzantine failures of Fault Containment Regions (FCRs). In addition to redundancy, other features that provide hardware and software fault tolerance include watchdog timers, processor interlocks, a privileged operating mode, hardware and software exception handlers, and self tests. A majority of correctly operating channels can disable all outputs of a failed channel using the processor interlock mechanism. A channel that is failed active is thus prevented from transmitting erroneous data or commands on I/O networks and IC networks or to local I/O devices.

Figure 3-4 also illustrates how redundant sensors are connected to the redundant FTP channels. In this example, three redundant copies (S1, S2, S3) of a sensor S are attached to three of the four FTP channels. No cross-strapping of the sensors to FTP channels is shown for simplicity, although it is possible and likely for some critical sensors. The process by which all four FTP channels derive a congruent value of the sensor S is as follows. Channel A reads sensor S1 and all four channels then execute the two-round Byzantine resilient exchange algorithm which culminates in all four channels receiving a congruent value of S1, say V1. The process is repeated for sensors S2 and S3. Now all four channels have the same three sensor values, say, V1, V2, and V3. To obtain a valid sensor value V, the three sensor values must be compared and voted. However, a bit-for-bit voting of redundant sensors is usually not possible since sensors measure real world parameters such as pressure, temperature, angle, acceleration, etc. which are all analog quantities. Even under no fault conditions, digital representations of redundant sensor values differ from each other. That is, the values V1, V2, and V3 may be different even though the sensors S1, S2, and S3 are all operating correctly. However, since they do represent real world physical quantities, a number of reasonableness checks such as rate

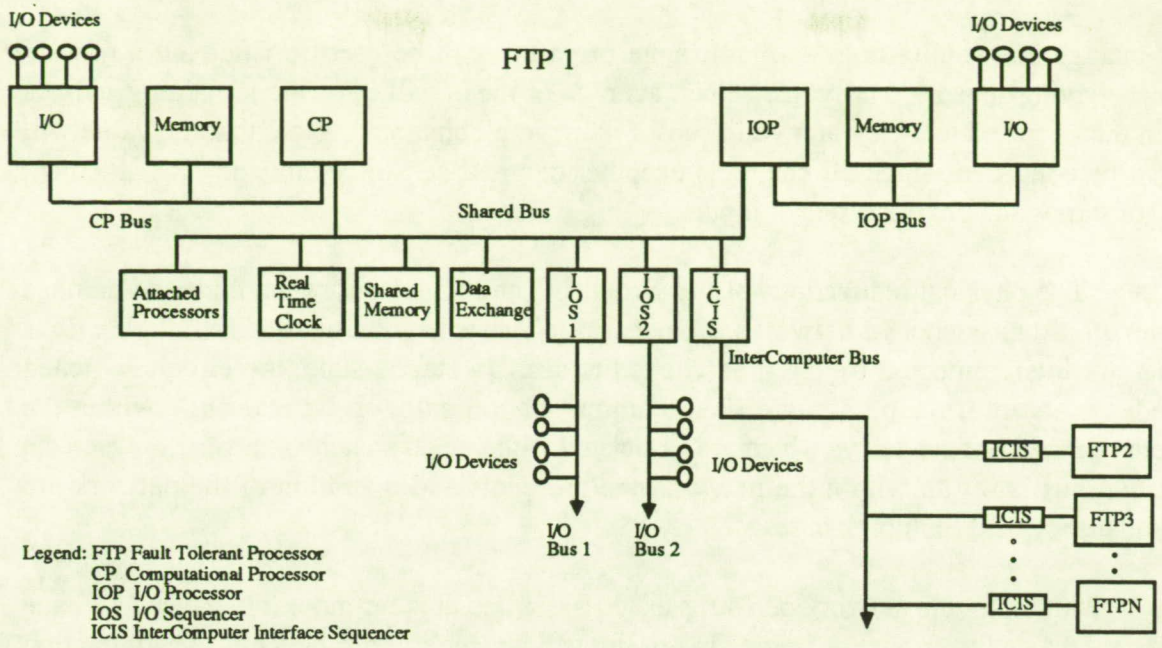

Figure 3-3. AIPS Processing Site Virtual Architecture

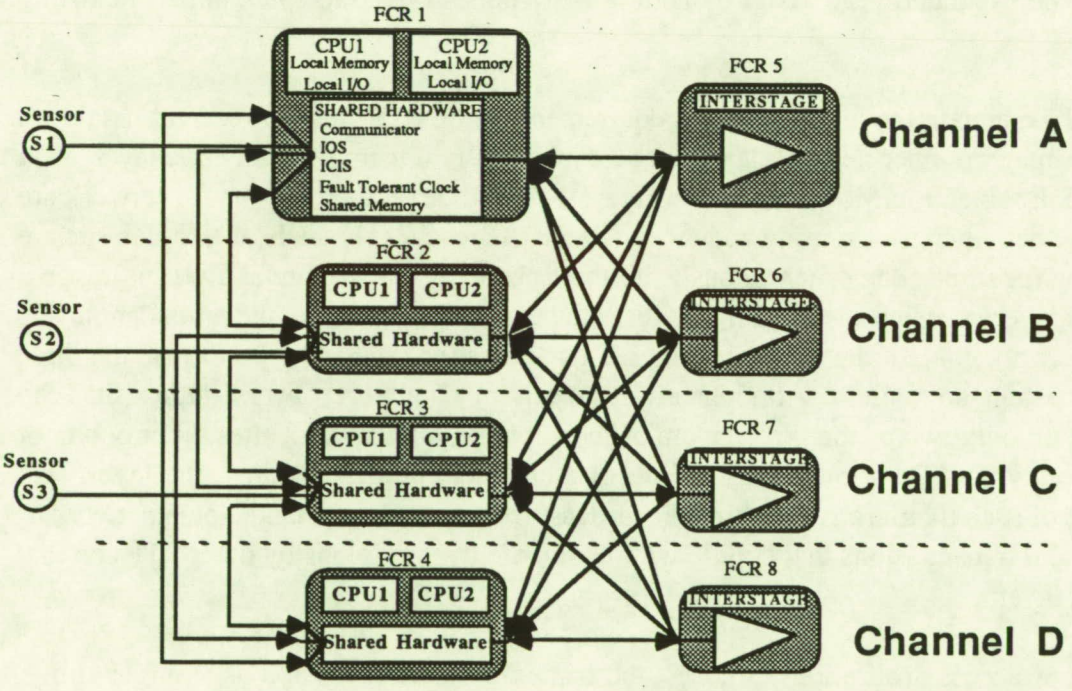

Figure 3-4. AIPS Quad Redundant Fault Tolerant Processor: Fault Containment Regions and Interconnections

of change and minimum/maximum range of values can be used to filter out a grossly misbehaving sensor. Mid-value select, average or mean values of the remaining sensors can then be used to arrive at a valid sensor value in all channels. Note that the value will also be congruent since all channels execute identical sensor redundancy management algorithm with congruent sensor inputs.

The physical realizations of the virtual I/O and IC buses are the fault and damage tolerant circuit-switched networks. A network consists of a number of full duplex links that are interconnected by circuit-switched nodes. In steady state, the circuit switched nodes route information along a fixed communication path, or "virtual bus", within the network, without the delays which are associated with packet switched networks. Once the virtual bus is set up within the network, the protocols and operation of the network are similar to typical multiplex buses.

Although the network performs exactly as a bus, it is far more reliable and damage tolerant than a linear bus. A single fault or limited damage can disable only a small fraction of the virtual bus, typically a node or a link connecting two nodes. By reconfiguring the network around the faulty element, a new virtual bus is constructed. The nodes are sufficiently intelligent to recognize reconfiguration commands from the network manager (explained in System Services Section) which is resident in one of the FTPs. The network can also be expanded very easily by adding more nodes linked to spare ports in existing nodes.

To maintain the fault tolerance requirements, each FTP channel receives data from all three intercomputer network layers but can physically transmit on only one layer, as in the AIPS Engineering Model shown in Figure 3-5. All three layers of the IC network are used together when transmitting and receiving data. Since all channels of a triplex site are executing the same code synchronously, all three channels (each channel transmitting on a different layer) transmit identical messages. Thus, within some skew, the redundant layers of the network contain the same message. This allows the receiving site to vote the three layers, masking any failure. Although always receiving on all three layers, duplex sites can transmit on only two of the three layers of the network, and simplex sites on only one of the three layers. Thus, malicious failure of a channel can disrupt only one layer. An example of such a failure is a continual broadcast (babbling) by a channel on a network or intermittent transmissions that collide with legitimate transmissions by other FTPs on that network layer.

For access arbitration purposes, the triplex network is treated as a single entity. FTPs, regardless of their redundancy level, contend for all three layers of the network. At the end of the contention sequence one, and only one, FTP may have access to all three layers of the network.

The IC networks, the FTP interfaces to the networks (ICISes), and the arbitration logic are designed in strict accordance with the fault tolerance theory [7]. The system provides error-masking capability for intercomputer communication between triplex (or higher redundancy level) FTPs. An arbitrary hardware fault, including Byzantine faults, anywhere in the system can not disrupt communication between FTPs of triplex or higher redundancy level.

3.4 System Services

Each processing site has the ability to operate autonomously, particularly for the performance of critical functions. However, the System Services allow the coordinated use of the entire information processing system to provide attributes superior to the more federated systems that are typical for current aerospace vehicles.

The Local System Services in each FTP include FTP initialization, a real time operating system, local resource allocation, FTP fault detection, isolation and reconfiguration (FDIR), and local time management. The real time operating system supports task execution management, including scheduling according to priority, time and event occurrence, and is responsible for task dispatching, suspension and termination. It uses the vendor-supplied Ada Run Time System (RTS), and includes additional features required for the AIPS real time distributed operating system.

FDIR has the responsibility for detecting and isolating hardware faults in the CPs, IOPs, and shared hardware. It is responsible for synchronizing both groups of processors in the redundant channels of the FTP and for disabling outputs of failed channel(s) through interlock hardware. The CPU hardware exception handling and downmoding/upmoding hardware in response to configuration commands from the system manager are also performed by the FDIR function in the FTP. It is also responsible for transient hardware fault detection and for running low priority self tests to detect latent faults. The local time manager works in cooperation with the system time manager to keep the local real time initialized and consistent with the universal time.

The I/O system services provide efficient and reliable communication between the user and external devices (sensors and actuators). The I/O system services software is also responsible for the fault detection, isolation and reconfiguration of the I/O network hardware and the IOS.

The IC user communication service is designed along the ISO's seven layer Open Systems Interconnect model. It provides local and distributed inter-function communication (point to point or broadcast mode) which is transparent to the application user. It provides synchronous and asynchronous communication, performs error detection and source congruency on inputs, and records and reports IC network errors to the IC network layer managers. The IC network manager is responsible for the fault detection, isolation and reconfiguration of the inter-computer network.

AIPS ENGINEERING MODEL CONFIGURATION

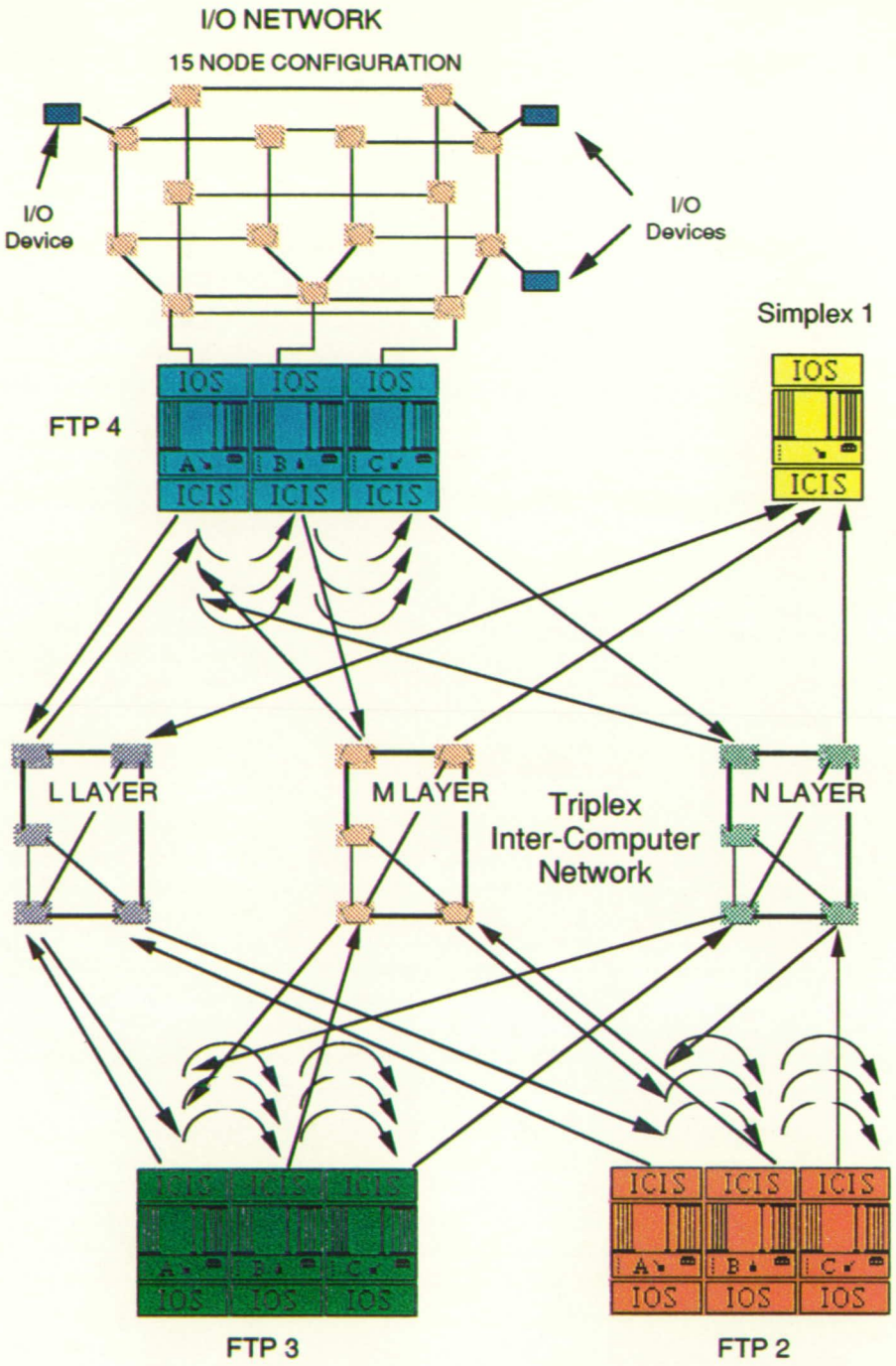

Figure 3-5. AIPS Engineering Model Configuration

The system manager is a collection of system level services: The system resource manager allocates migratable functions to FTPs. This involves the monitoring of the various triggers for function migration such as failure or repair of hardware components, mission phase or workload change, operator or crew requests and timed events. The system fault detection, isolation and reconfiguration (FDIR) is responsible for the collection of status from the inter-computer (IC) network managers, the I/O network managers, and the local GPC redundancy managers. It resolves conflicting local fault isolation decisions, isolates unresolved faults, correlates transient faults, and handles processing site failures. The system time manager, in conjunction with the local time manager on each FTP, has the job of maintaining a consistent time across all FTPs.

3.5 Flight System Characteristics of Building Blocks

3.5.1 Functional Building Blocks

The AIPS hardware building blocks are the Fault Tolerant Processors, the Input/Output and InterComputer Networks, and the network interfaces as shown in Figure 3-1 and are used to implement the AIPS virtual architecture and the FTP virtual architecture shown in Figures 3-2 and 3-3, respectively. The building blocks are comprised of the Fault Tolerant Processor Channel and the Communications Node and each of these are composed of smaller functional elements or modules. These hardware building blocks and the modules which make up the building blocks are described in the following paragraphs and are based on the technology projections for the 1992-1993 time frame. Additionally, the implementation considerations for performance, radiation hardness, power dissipation and maintainability outlined earlier in the requirements section have been incorporated into these building blocks.

3.5.1.1. Fault Tolerant Processor Channel

The Fault Tolerant Processor Channel is shown in Figure 3-6 and incorporates the FTP functions identified in Figure 3-3. The channel is comprised of two Central Processor Unit (CPU) modules, a Shared Devices module, one or more Input/Output Sequencer (IOS) modules, one or more InterComputer Interface Sequencer (ICIS) modules, and the Communicator and Interstage module. The channel's modules are interconnected using a shared bus which is common to all modules in the channel. These modules are described in Subsections 3.5.1.1.1 to 3.5.1.1.5.

Taken together, the hardware in a quad-redundant FTP will support at least a FAIL-OP, FAIL-OP mode of operation. A quad-redundant FTP will continue to operate correctly after two sequential, arbitrarily malicious faults in any two FCRs. Additionally, it can also survive certain combinations of triple and quadruple FCR failures. For example, sequential failures of two processors in different channels and their associated Interstages will be masked. After these four failures have occurred, the FTP will continue to function

correctly in a duplex mode (two channels still functional and voting inputs, state data and outputs). The fifth failure can be detected with near perfect coverage resulting in FAIL-STOP capability. Alternatively, if the operational requirements are such that it is desirable to continue operation in a simplex mode, then the probability of FAIL-OP after the fifth failure is directly related to the completeness of a channel's self-tests (self-test coverage).

3.5.1.1.1. Processor

The processor will be based on the MIPS, Inc., R3000 or similar performance 32-bit RISC architecture. The nominal frequency of operation will be 40 MHz which can provide 12 to 18 MIPS peak performance (DAIS mix). The CPU module will have both Random Access Memory (RAM) and Read Only Memory (ROM) and the combined total will be between one and four megabytes. The processor module will also have counters/timers used for support of the operating system and an interrupt controller. A local bus interface will be provided for expansion of the ROM if the size of the operating system and application software should require additional memory space.

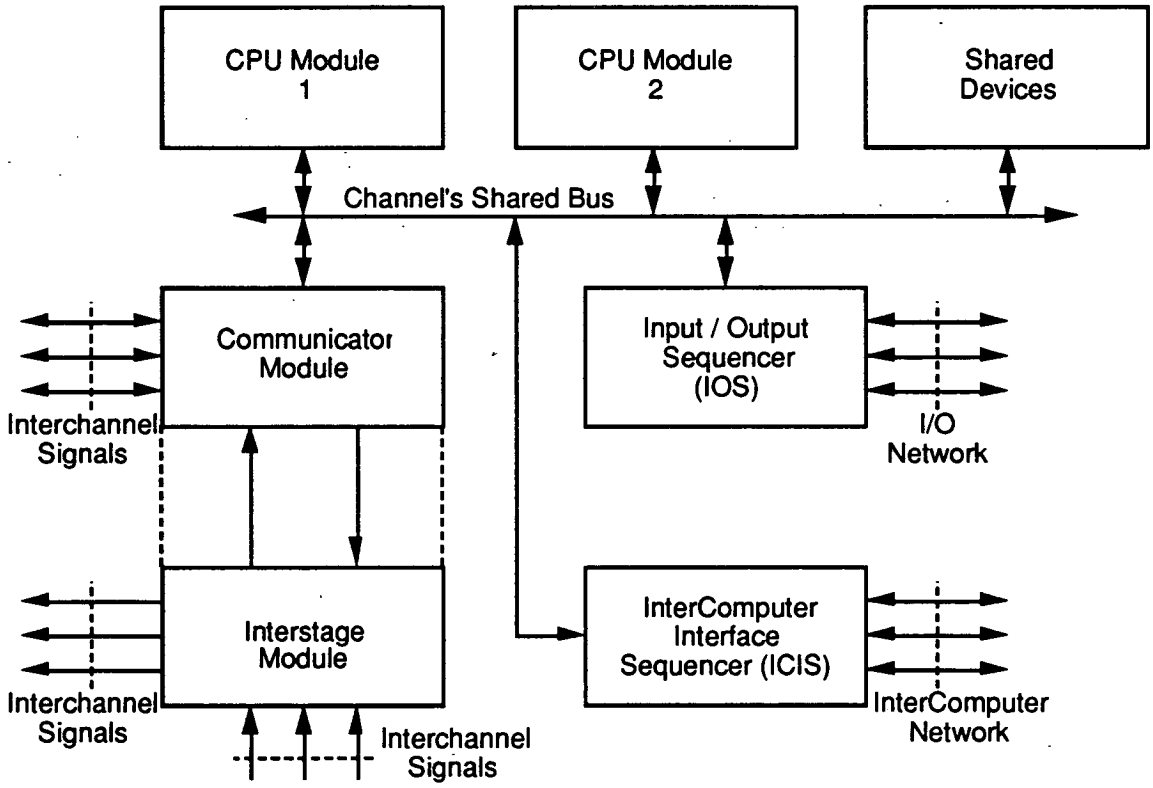

Figure 3-6. Fault Tolerant Processor Channel

The processor, executing instructions at the 40 MHz clock rate, must be provided instructions and data at that clock rate. Memory devices with access times less than the 25 nsec clock period, however, will continue to have relatively low bit densities (16 KBits to

256 KBits per integrated circuit) and their active power dissipations will be high (one to two Watts per integrated circuit). The CPU module's performance and power dissipation, therefore, will be determined by the memory architecture implemented on the module.

While it is possible to implement a cache memory system (very low access time memory implementation) to support the fast cycle times of the processor integrated circuit, the relatively small memory space required in the AIPS for ALS can be implemented such that all of the memory appears to operate as a cache. This can be accomplished by implementing the memory array as 64 bit or 128 bit wide memory rather than the 32 bit words needed by the integrated circuit. For example, if a memory array is implemented as 128 bits wide then the memory array is 64 KWords long (each Word is now 16 bytes). When a memory read cycle is accomplished, 128 bits of data or instruction are read from the memory and are held in a memory read buffer. Subsequent memory accesses are checked to determine if they are from a memory location contained in the buffer. If the access is from a location contained in the buffer then the information is provided to the microprocessor without the penalty of another memory access. Memory write cycles must be completed by a physical write to memory.

The other circuitry on the CPU module (timers, counters, interrupt controllers, etc.) are common to most microprocessor based systems used in real-time control systems. Their use in the CPU module is comparable to these other applications and to the AIPS proof-of-concept system. No special provisions are required for these functions.

3.5.1.1.2. Shared Devices

System level support circuits to be included on this module are the channel's shared memory (RAM), real-time clock, timers and interrupt controller.

The shared RAM in the channel is used for interprocessor communications and for intermediate storage of a limited quantity of data. The use of this memory as a communications area adds the derived requirement that its access time, including any shared bus overheads, be minimized. This memory, therefore, must have access times which are as short as possible. Unlike the memory to be implemented on the processor module, this memory space will be subjected to random accesses to data structures located in the shared space and no benefits can be accrued by providing a multiple word access capability.

3.5.1.1.3. ICIS and IOS Hardware

The InterComputer Interface Sequencer (ICIS) and Input/Output Sequencer (IOS) provide the FTP channel's interface to the redundant, fiber-optic communication paths needed in a network of fault tolerant computers and input/output devices. While the purposes of the IC network and the I/O network are different, the communications protocol used in each network is identical and common hardware will be implemented to interface to

both networks. A block diagram of this Network Interface Sequencer (NIS) module which replaces the functions of the ICIS and IOS is shown in Figure 3-7.

The communications protocol used on the networks will be based on the Fiber Distributed Data Interface (FDDI) standard with a bit rate of 100 MegaBits per second (MBPS) which is 50 times the 2 MBPS data rate of the AIPS engineering model network. FDDI is normally implemented as two counter-rotating physical rings and access to the rings is controlled using a token passing protocol as defined by IEEE-Std-802.5 (see Figure 3-8). The implementation of the IC and I/O networks for the ALS will be a modification of the FDDI standard where a modified Laning Poll will be used to access the network of virtual buses rather than the FDDI token.

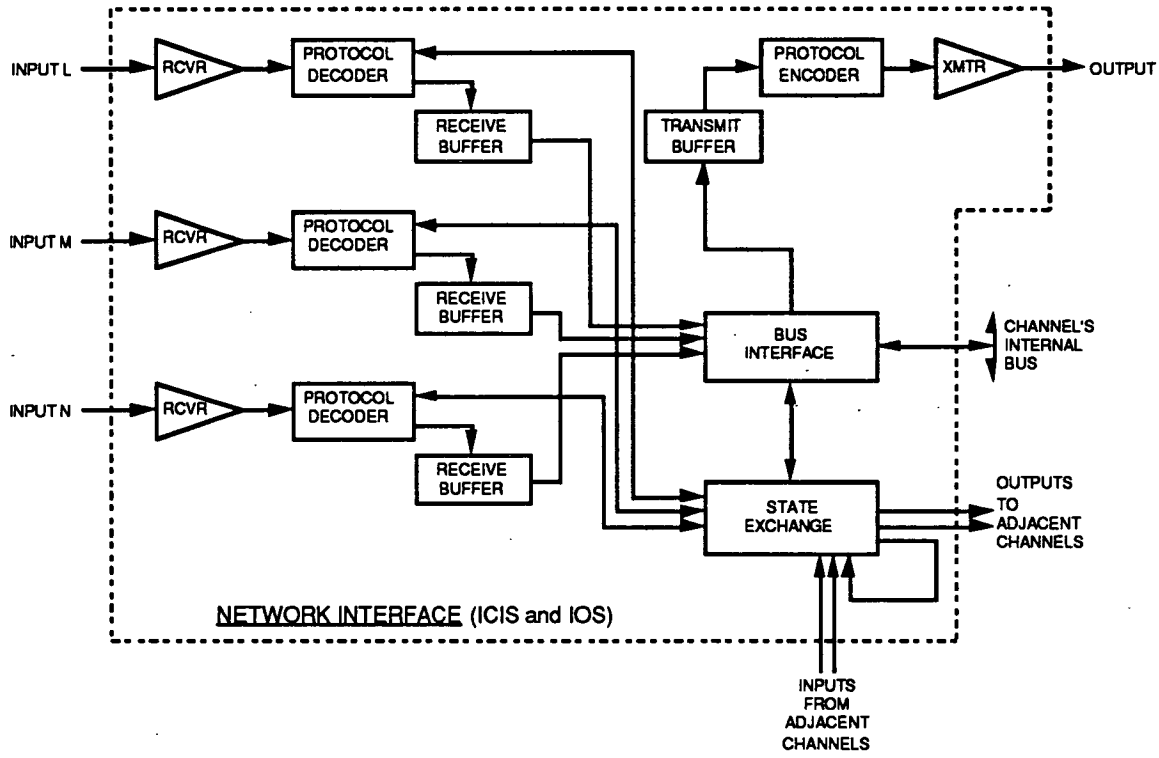

Figure 3-7. Network Interface Sequencer

The FDDI token consists of three bytes and its construction is specific to ring type networks. Two bytes are the starting and ending delimiters and these are used for phase-locked-loop (PLL) synchronization as well as marking the beginning and ending of the token frame. The third byte is the access control byte and it contains the token bit as well as other bits for monitor, priority and priority reservation. The monitor bit is used by the ring monitor (a station responsible for monitoring the operation of the ring) to detect "orphan" frames and thereby remove them from the network. The priority and priority reservation bits are used to establish the operating level of the network and to make "reservations" for future data transfers.

The Laning Poll supplants the FDDI token and is an extension to the poll implemented in the AIPS engineering model. The two significant modifications to the Laning Poll is the addition of the Q-Bit, which is required for quad-redundant processing sites, and the reduction in the arbitration timing.

The proof-of-concept system was built with triplex, dual and simplex processing sites and the Laning Poll was implemented accordingly; there were no provisions for the addition of quad redundant processing sites in the engineering model. The Q-Bit, which precedes the T- and D-Bits, allows a quad redundant processing site to poll for control of the network and, because the most significant bits are transmitted first, do so at a high priority than other processing sites with lower degrees of redundancy. The other bits of the Laning Poll will have the same significance and will operate like the existing network. The algorithm for the poll is:

```
FOR ALL i WHILE contending DO:
   TRANSMIT Pi on the NETWORK
      IF Pi = 1 and RECEIVED = 1 then CONTINUE_VIE
      IF Pi = 1 and RECEIVED = 0 then VIE_WON
      IF Pi = 0 and RECEIVED = 1 then VIE_LOST
      IF Pi = 0 and RECEIVED = 0 then CONTINUE_VIE
```

Figure 3-9 depicts the network arbitration timing for the AIPS engineering model system and the timing for an advanced technology AIPS with a comparable number of processing sites and Communications Nodes. The incorporation of technology projected to be available in 1992 will permit an improvement in the arbitration timing in excess of one order of magnitude. An AIPS network sized for the Advanced Launch Vehicle, however, requires far fewer processing sites and Communications Nodes and will require even less arbitration time. Overall, the network can be arbitrated in $1/16^{th}$ to $1/48^{th}$ the time required for the engineering model, depending on the number of nodes, as shown by the timing values in Figure 3-9.

The physical layer of the network is the FDDI implementation. It does not use Manchester encoding because this form of data transmission requires frequencies which are twice the data rate (200 MHz for 100 MBPS). The encoding scheme used at the physical layer is "four out of five" where each group of four symbols (binary digits) is encoded as a group of five bits on the medium. Sixteen of the 32 possible combinations are for the data bit patterns, three combinations are for delimiters, two are for control, three are for hardware signaling, and eight are reserved for future use.

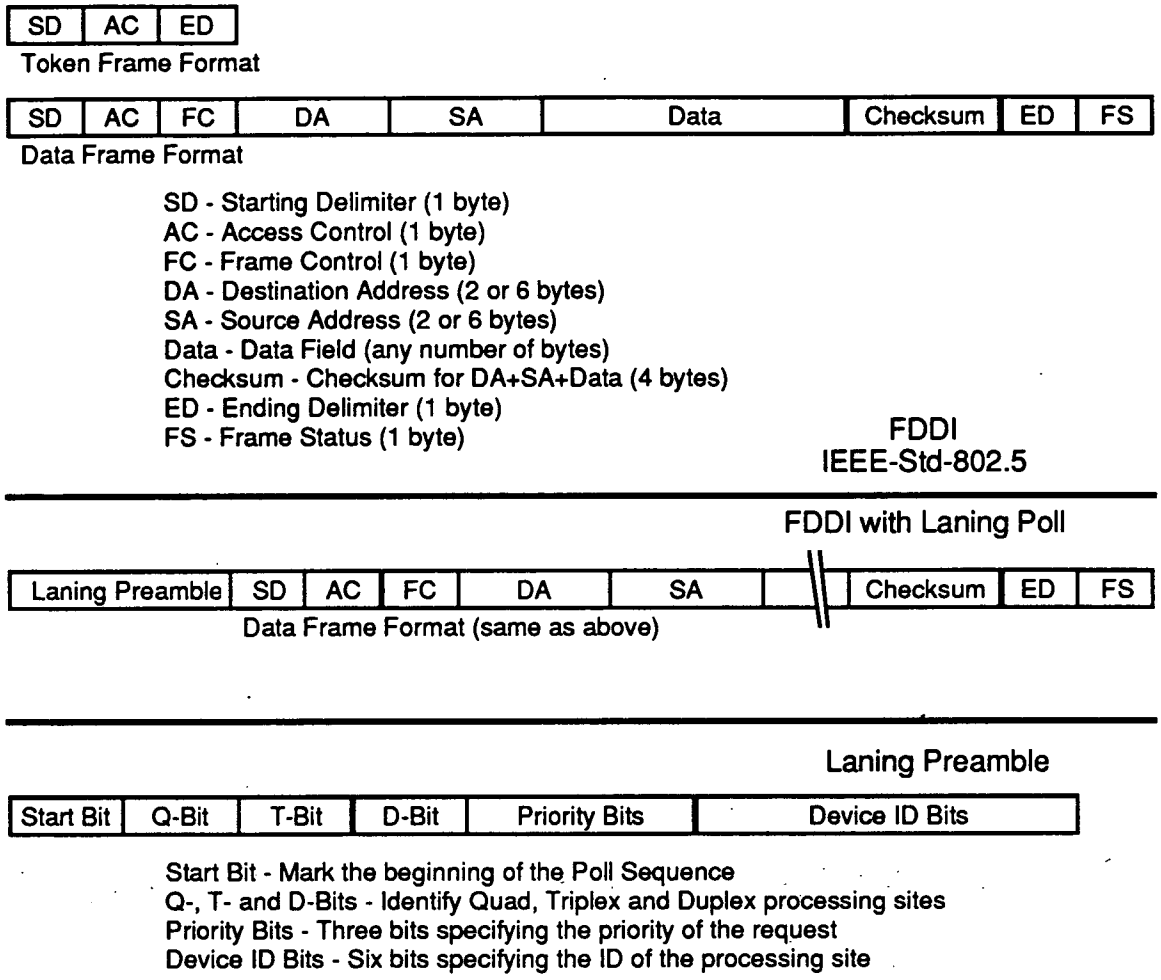

Figure 3-8. IC and I/O Network Data Frame Format

The Network Interface Sequencer module, when used as the interface to the InterComputer Network, will be connected to the three network layers as was done in the proof-of-concept system. The I/O network does not have the redundant layers, however, and the module design, with the three inputs and the state exchange logic, does not appear to be capable of operating with the single layer I/O network. The common interface module design will interface to the I/O network if a fiber optic beam splitter is used to provide the three parallel inputs required by a NIS module, and the state exchange outputs are "looped back" so the state exchange inputs are monitoring their own exchange outputs. In this way, an interface module connected to the I/O network will "vote" the single set of input data and the state machines and state exchange hardware will be "synchronizing" with itself.

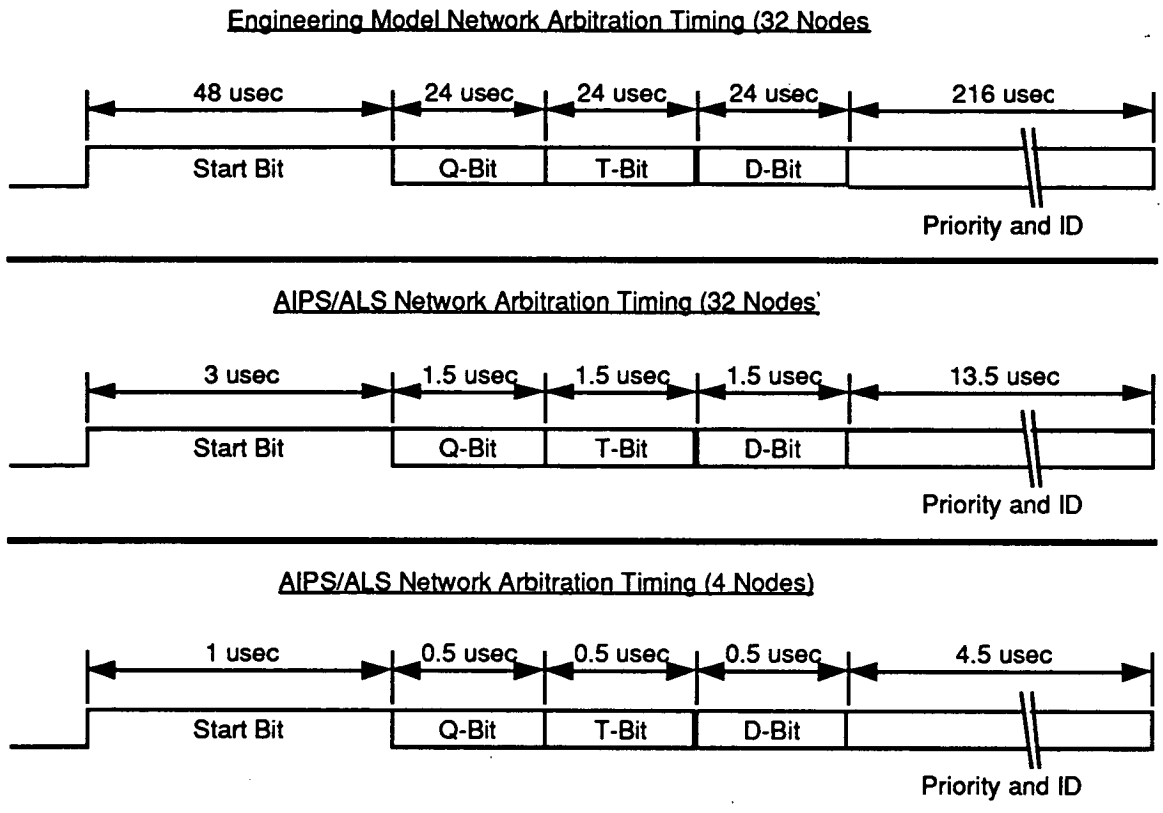

Figure 3-9. Network Arbitration Timing

3.5.1.1.4. Communicator and Interstage

The communicators, resident on each shared bus and thus shared by the processors in their respective channels, together with the Interstages are used to tie the four channels of the FTP together in a manner that guarantees that each FTP processor can maintain congruent inputs, state information and outputs. The interchannel hardware, the communicators and the interstages, is used to exchange and vote data, fault-tolerant clocks, and external interrupts.

The interstages provide the additional Fault Containment Regions (FCRs) and the connectivity required for an efficient implementation of the two round communication algorithm required to correctly handle Byzantine failures. To ensure the integrity of these additional FCRs, the interstages are powered by power supplies independent of the power supplies used to power the processors and shared hardware. In addition, all inter-FCR connections are made using fiber optic drivers, receivers and cables so that the integrity of the FCRs cannot be compromised. This electrical isolation prevents a fault in any given FCR from migrating past the boundaries of that region and corrupting other FCRs.

The communicator and interstage function will be based on the corresponding functions developed in the AIPS engineering model. The three fundamental purposes of the communicator and interstage hardware and its fiber optic data communication paths are: 1) Provide the paths for distributing data in one channel to all other channels; 2) Provide a mechanism for comparing results of the redundant channels; and 3) Provide a path for distributing and comparing timing and control signals such as the fault tolerant clock and external interrupts. As shown in Figure 3-6, the communicator and interstage functions reside in each channel and are accessed via the channel's shared bus.

Two types of data exchanges are possible. These are simplex exchanges and voted exchanges. The simplex exchange is used to distribute copies of data from one channel to all other channels in a congruent fashion. An example of such a data item is the value of a sensor that is available in only one channel. Voted exchanges, on the other hand, are used to compare and vote results of the redundant channels. An example is an actuator command produced by a control law implemented in all channels, which is to be voted before the command is issued to the actuator.

Data will be exchanged between the redundant channels one 16 bit word at a time. To perform a voted exchange, each processor writes the value to be voted in the transmit register. Writing to this register initiates a sequence of events in hardware which culminates with the voted value being deposited in the receive register of each processor. The processors in each channel can read the receive register at this point to fetch the voted value.

A significant portion of the existing communicator function has been implemented as an Application Specific Integrated Circuit (ASIC). The ASIC, a 6000 gate, 2.0 micron CMOS Configurable Gate Array, implements the data communicator function but the clock and interrupt communicator functions are implemented separately. The implementation for the Advanced Launch System will use a 20,000 gate, 0.8 micron CMOS gate array and will incorporate the data, clock and asynchronous interrupt communicator functions. The data paths will be serialized/deserialized using logic also on the new gate array. The serial data rate will be 128 MBPS which supports peak data exchange rates of eight megabytes per second.

The interstage functions will also be implemented using a 20,000 gate 0.8 CMOS gate array. This gate array will integrate the data, clock and asynchronous interrupt interstage functions. The communicator and interstage functions are all very similar and, as such, the two ASIC will have approximately 95% common designs.

Figure 3-10 shows the block diagram of the communicator and Figure 3-11 shows the block diagram of the interstage. Critical to the proper operation of the communicator is the fault masking action inherent in the design of the hardware voter. The voter compares the four parallel data streams on a bit-by-bit basis and produces a "majority" output bit for

each input bit. Any disagreements in the voting process are detected and the identity of the disagreeing input is stored in an error register. There are five bits in the error register: one bit for each of the four inputs and a fifth bit which is set to indicate that a pairwise split was detected. In the presence of a single fault, the voter produces the correct result and latches the identity of the faulty input.

A voter with four inputs can tolerate two faults if and only if it is has been configured to vote on only three inputs (assuming one of the faulty inputs is the input being ignored). Therefore, associated with each voter in the communicator is a voter mask register. This register contains a four bit mask used by the voter and permits voting of data only from the unmasked inputs.

The AIPS approach to achieving exact consensus is to use identical software running on identical, redundant, "clock deterministic" hardware operating in tight, micro-

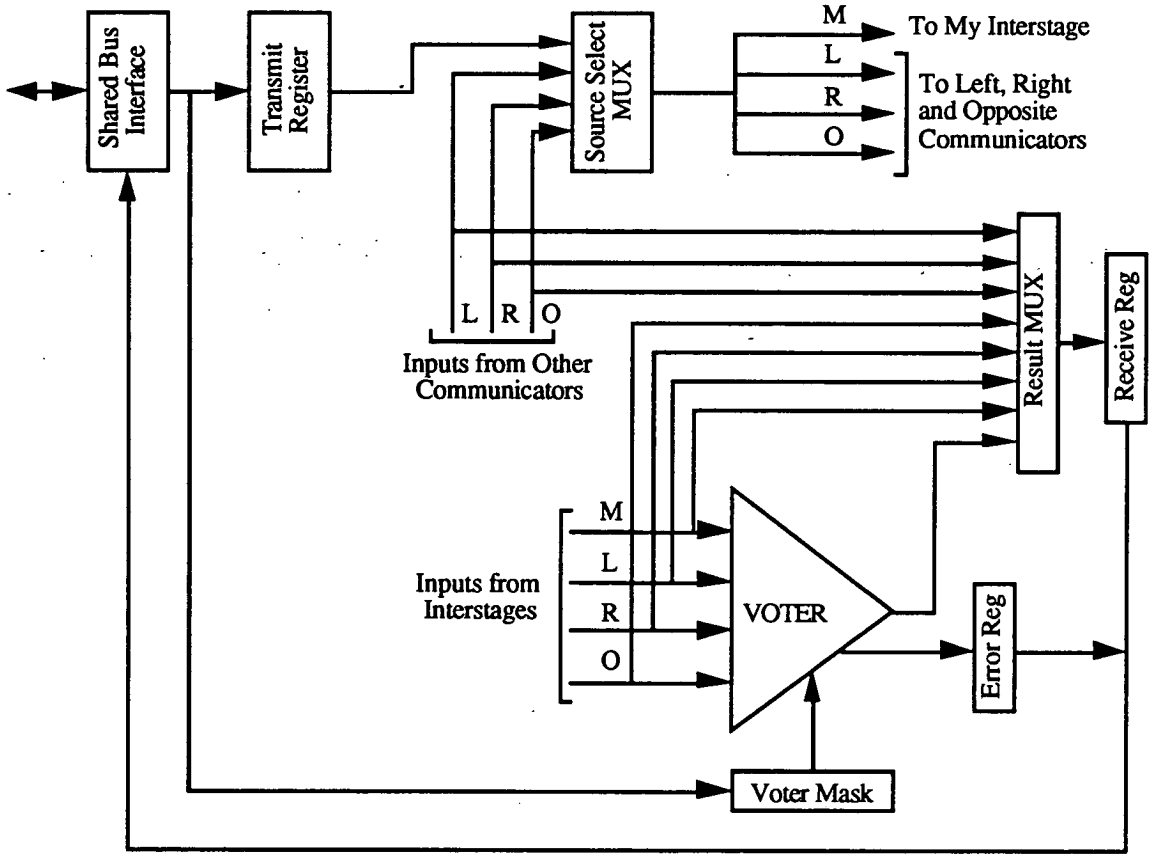

Figure 3-10. FTP Communicator

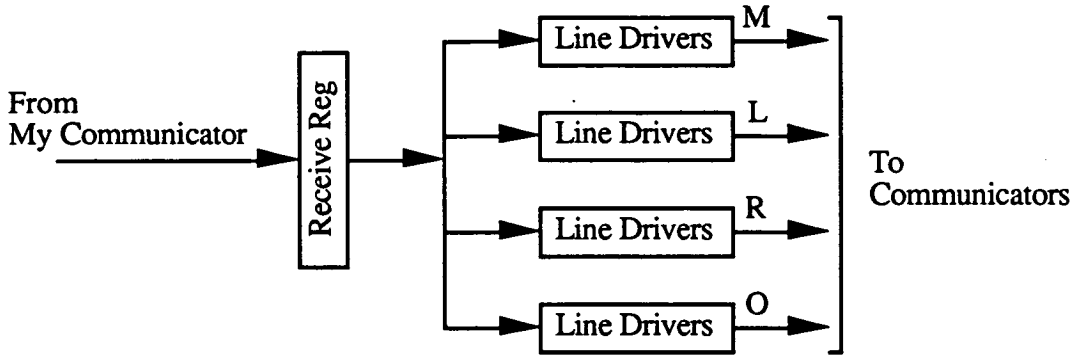

Figure 3-11. FTP Interstage

frame synchrony. As noted earlier, for hardware to be "clock deterministic", it must perform each of its operations in a fixed and predictable number of clock cycles.Synchronous operation of that hardware then allows an efficient, hardware oriented solution to the Byzantine Generals Problem. It relieves the software of the burden of maintaining process synchrony. In addition, it allows the necessary voting to be performed in hardware in a straightforward manner. This means that errors can be detected and masked as they occur, transparently to the application software.

One way to achieve synchronization of the redundant hardware would be to drive it with a single clock. However, a common clock would represent a potential single point failure. Therefore the AIPS FTP uses redundant clocks. To maintain synchrony, the clock signals are digitally phase-locked to one another. To ensure that they are resilient to Byzantine failures, they are exchanged and voted in a manner similar to the way data is exchanged and voted. In this case, exact consensus between channels is achieved on the relative phase of these clock signals and they are collectively known as the Fault-Tolerant Clock (FTC).

Each channel of the AIPS FTP maintains its own version of the FTC. Like the rest of the FTP, the FTC hardware is partitioned into eight FCRs as depicted in Figure 3-12. However, the network topology used to interconnect these FCRs is different. Unlike the communicators described above, each FTC communicator broadcasts its version of the FTC to all FTC interstages because each channel's version of the FTC is a separate, simplex clock source which must be exchanged and voted. Requiring each FTC communicator to broadcast its version of the FTC to every FTC interstage, four simplex source exchanges are performed in parallel with an additional voting plane at the interstages. The voted clocks produced at each FTC interstage are then broadcast back to all FTC communicators where they are voted the second time.

The frequency of the FTC is nominally 2 MHz which is too slow to drive most of the hardware (for example, the microprocessors require a 40 MHz clock). Therefore, a faster clock that is deterministically related to FTC is required to drive the hardware in each

channel. In the AIPS engineering model, that faster clock is a 16 MHz clock which, in turn, is used to produce FTC. For the ALS, the faster clock will be the 40 MHz clock needed by the microprocessors and their supporting logic.

3.5.1.1.5. FTC and 40 MHz Signal Determinism

During normal operation, each channel's FTC is continuously compared to the "majority" of all FTCs and adjusted accordingly. If a channel's FTC is in phase with the "majority", no action is taken. If the channel's FTC is too slow, its period is shortened by a known amount, and if it is too fast, its period is lengthened by the same known amount. The "majority" can best be defined by illustrating an example. Figure 3-13 shows four clock signals and the resulting "majority" that would be produced by the clock voter algorithm used in the AIPS engineering model. Under steady state, no fault conditions, the output of the voter algorithm is a very slightly delayed version (equivalent to the gate delay through the voter itself) of the second fastest input to the voter. The voter is performing a second edge detect algorithm. The "majority" output will only transition from one state to the next, following the second input signal to transition to the new state.

In order to maintain a deterministic relationship between FTC and the 40 MHz signal in each channel, whenever an adjustment is made to a channel's version of FTC, corresponding adjustments are made to that channel's 40 MHz clock. Clock pulses in the 40 MHz pulse train are either unaltered or stretched (one 20 MHz clock pulse in place of two 40 MHz clock pulses) to guarantee that there will always to be a deterministic number of clock pulses in each half period of the FTC. When a channel's FTC is in phase with the "majority", one 40 MHz clock pulse is stretched in that channel's 40 MHz clock signal (19 clock pulses in one period of FTC). Whenever the channel's FTC is late, Case 3 in Figure 3-14, this pulse stretching is suppressed, shortening the FTC period by one 40 MHz clock cycle while retaining 19 clock cycles in one FTC period. If the channel's FTC is too early, Case 2 in Figure 3-14, a second stretched clock pulse is included in the FTC period again keeping 19 clock pulses in one FTC period.

3.5.1.2. Communications Node

The circuit switched nodes of the IC and I/O networks are identical and are based extensively on the nodes developed for the AIPS engineering model. The Communications Node (CN) and the operation of each set of Port Logic is portrayed in Figures 3-15 and 3-16. Each node has five identical ports (only one port is shown in Figure 3-15) where a port's input and output are connected via fiber optic cables to a device's output and input port, respectively.

The FDDI data frame, described above, includes fields for the destination and source addresses. Each device connected to the IC and I/O networks (ICIS, IOS and CN)

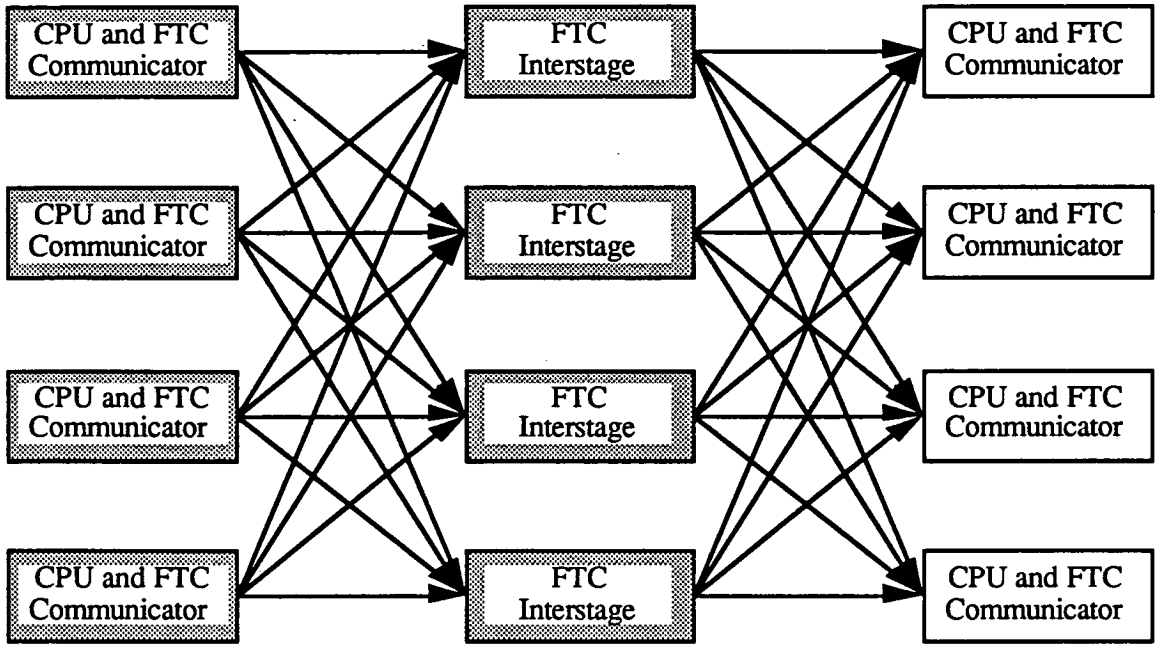

Figure 3-12. FTP Fault Tolerant Clock Topology

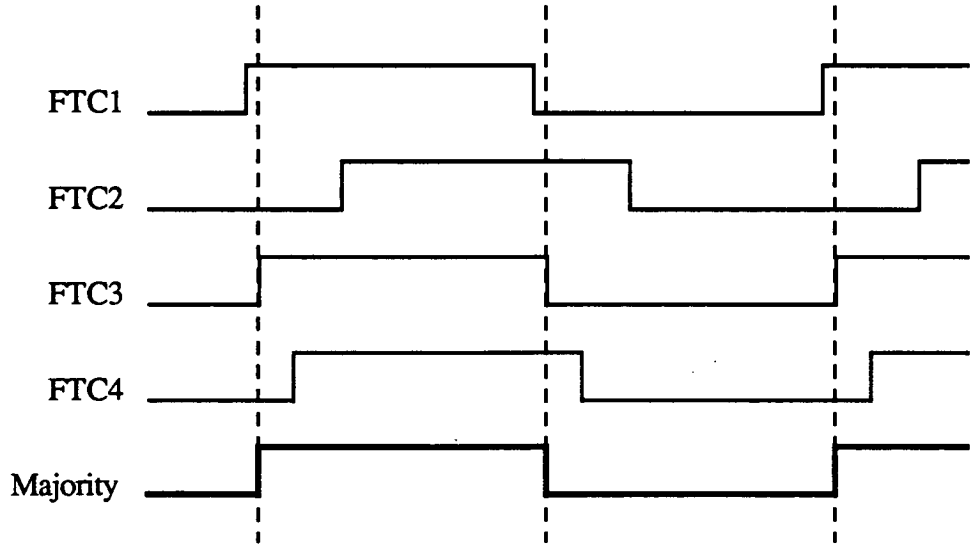

Figure 3-13. FTC "Majority" Voter Operation

will be provided with a unique 16-bit identification code and, during communications, the Destination Address (DA) field will be set to the desired destination's identification code. The Source Address (SA) field will always be used to identify the originator of messages and thereby specify which device should receive any required response messages.

3.5.1.2.1. Node Controller

There is one Node Controller in each CN and it is responsible for CN initialization, monitoring the communications traffic for CN messages, and enabling or disabling the Port Logic regeneration logic and transmitters as directed by the messages received via one or more Port Logic receivers.

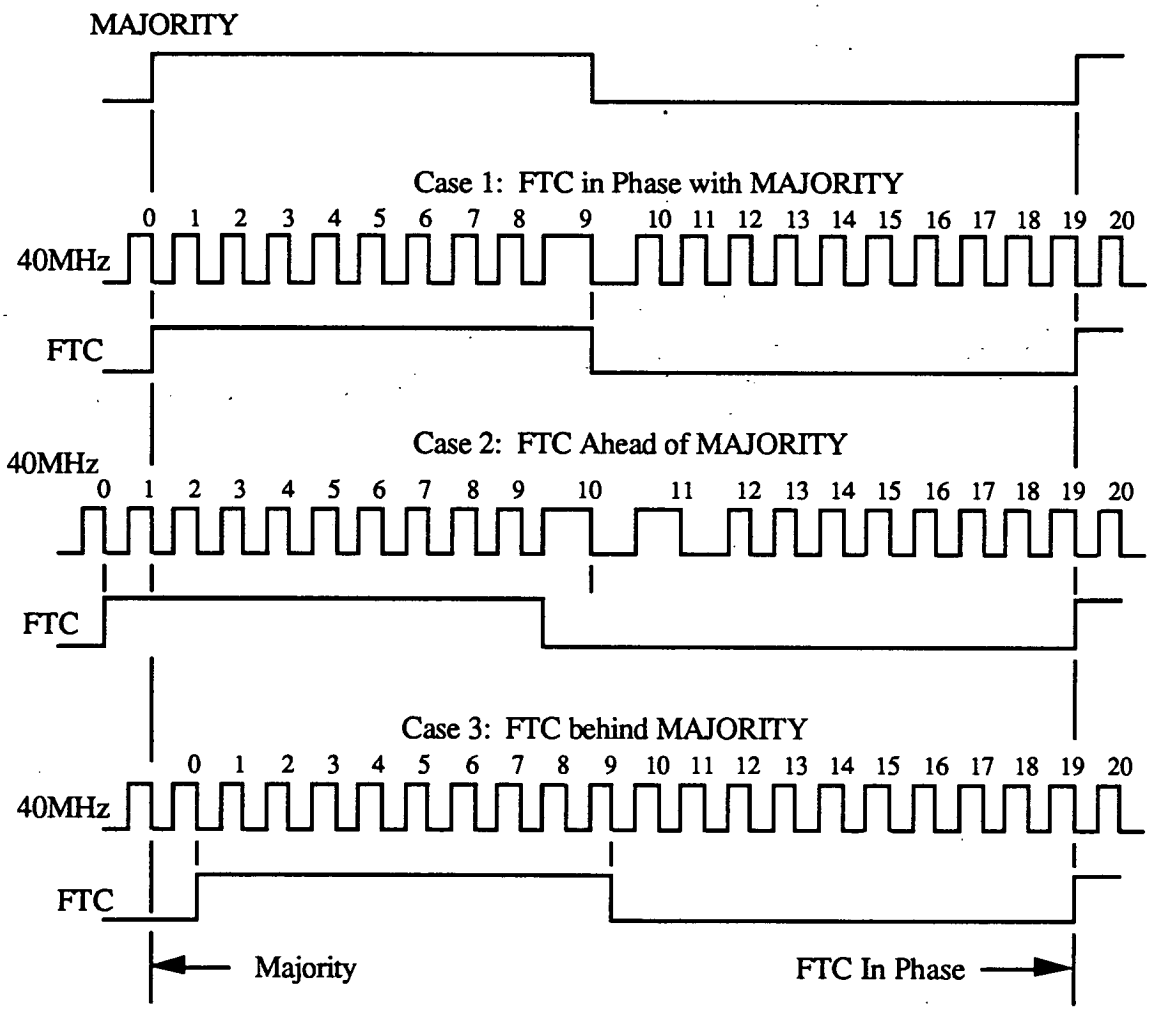

Figure 3-14. Clock Corrections

The Port Logic decodes all messages received on any input port and compares each DA to the 16-bit CN identification node. If the 16-bit address does not match the code for the CN, the message is automatically discarded by the Protocol Decoder. If the DA matches the CN's code the Protocol Decoder stores the message in the Receive FIFO and sets appropriate status bits to indicate the reception of a message the validity/invalidity of the message based on the Checksum. The Node Controller will then decode the message to determine if it is a valid message and, if it is a valid message, respond accordingly. If a reply message is required, the SA included in the original message will be used as the DA in the reply.

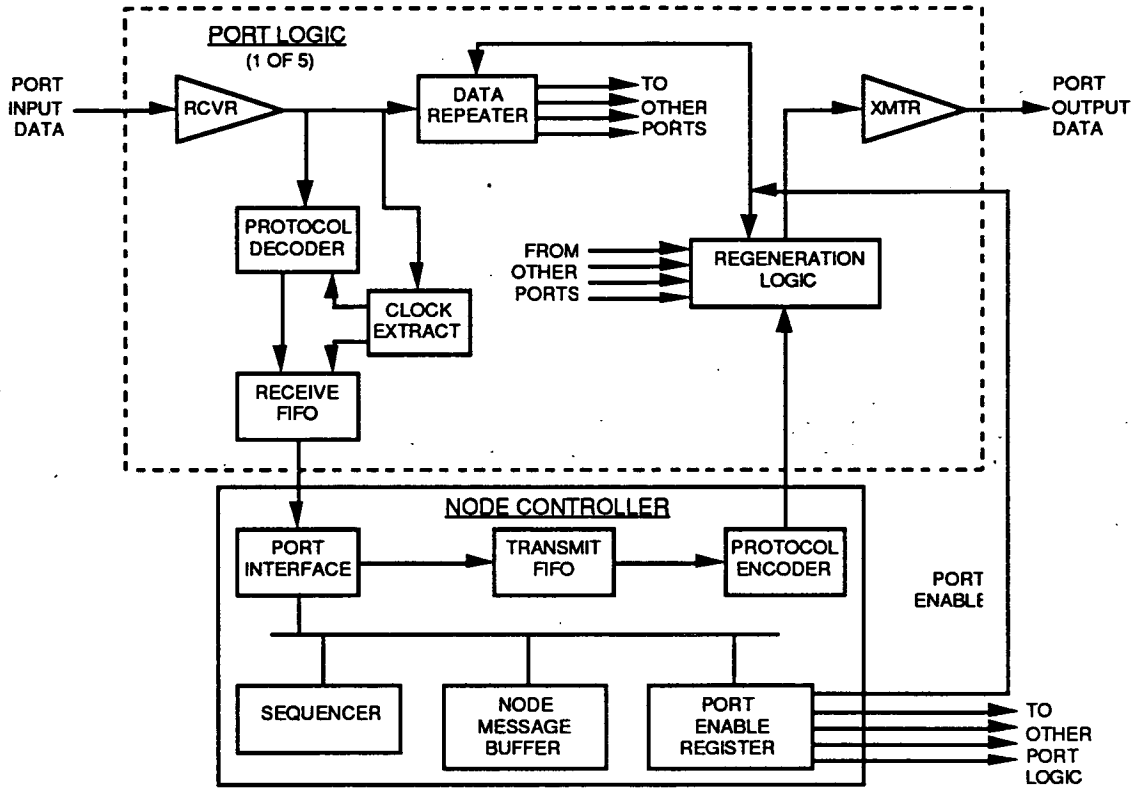

Figure 3-15. Communications Node

Examples of messages to be sent to CNs include requests for status and direction to perform CN reconfiguration. The Network Manager, resident in selected processing sites, can request status as an input to its network monitoring task and issue reconfiguration messages to establish communication paths or to change the port enable status of the CNs.

3.5.1.2.2. Communications Port Receiver/Transmitter

Each of the five CN port receiver/transmitter are identical and Figure 3-16 is the block diagram of these ports. Every port has a unique Port Enable signal which enables or disables the path from the receiver and the path to the transmitter. If the port is disabled

and is failure free, whatever data presented to the receiver is not repeated and whatever data is presented to the data encoder is ignored.

If the port is enabled and is failure free, the data presented to the receiver is repeated and sent to the regeneration logic in the other four ports. The enabled port's regeneration logic will encode a data stream which is the logical-OR of the other four port's receivers and the Node Controller transmitter.

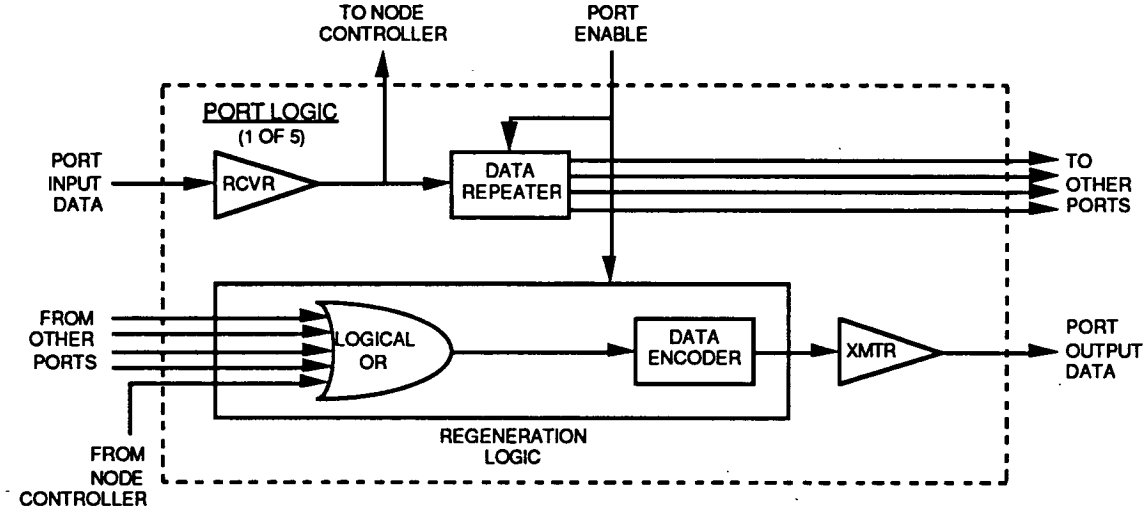

Figure 3-16. Node Port

3.5.2. Projected Performance Parameters

Some performance data was included in the above paragraphs. This section will summarize the parameters that affect the performance of the AIPS building blocks when they are implemented using the technologies projected to be available in the 1992-1993 time frame. The resulting performance projections are summarized in Section 3.5.4.

3.5.2.1. Fault Tolerant Processor Channel

The Fault Tolerant Processor channel will be based on RISC microprocessors with 40 MHz clocks. Additionally, these microprocessors will be supported by instruction caches which provide an average of one wait state per instruction fetch and a maximum of four wait states for cache misses. Data caches will not be implemented due to the size limitations of the SEM-E modules and four wait states will be required for all data accesses. The memory available on each SEM-E processor module will be four megabytes of random access memory (RAM) and read only memory (ROM), combined. The mix of RAM and ROM will be variable in 256 Kbyte increments and will be determined after detailed design of the application software.

Each FTP channel will have two processors. These processors may be allocated such that one performs all of the required computation while the other performs all input and output or they may each share the computation and I/O burden. As in the engineering model implementation each channel can be used in quad-, triplex-, dual-redundant or simplex processing sites. The design of the Communicator and Interstage hardware and the software supports the use of the same hardware at different levels of redundancy.

3.5.2.2. Communications Node

The IC and I/O Networks operate at a 100 MBPS signalling rate. At that signalling rate, the information transfer rate is a maximum of 96.4 MBPS if the data frames are transferring packets with 4,096 bytes of data. The information transfer rate will be correspondingly lower for data frames with fewer bytes in the information field. The design of the ALS Communications Node will be based on the engineering model and full-duplex communication is supported throughout the network. All IC and I/O Network connections will be made using fiber optic data links for electrical isolation and tolerance to electromagnetic interference due to lightning strikes and pyrotechnic devices.

Each Communications Node will be implemented using a single SEM-E module. Each module will provide five ports for connections to the processing sites or I/O devices and a Node Controller implemented in a 20,000 gate, 0.8 micron ASIC.

3.5.2.3. AIPS for ALS Avionics Packaging

The packaging concept for the AIPS for ALS avionics is the Standard Electronic Module - Format E (SEM-E) conduction cooled modules installed in base-plate cooled chassis. Module-to-module interconnections are to be made using a motherboard rigidly mounted to the chassis and the standard SEM-E 250 contact connector is used on all modules. Figure 3-17 illustrates the packaging concept for the AIPS for ALS avionics.

3.5.3. Hardware Failure Rate Projections

MIL-HDBK-217E provides the failure rate data and method for estimating reliability during early design phases [16]. The information needed to apply the method is: 1) Generic part types including complexity for microelectronics and quantities; 2) Part quality levels; 3) Estimate of the number and type of opto electronic components; 4) Estimate of the number and type of passive components (resistors and capacitors); 5) Estimate of the number of connectors and connector contacts; 6) Estimate of the number of other connections (e.g., device solder joints); and 7) Equipment environment.

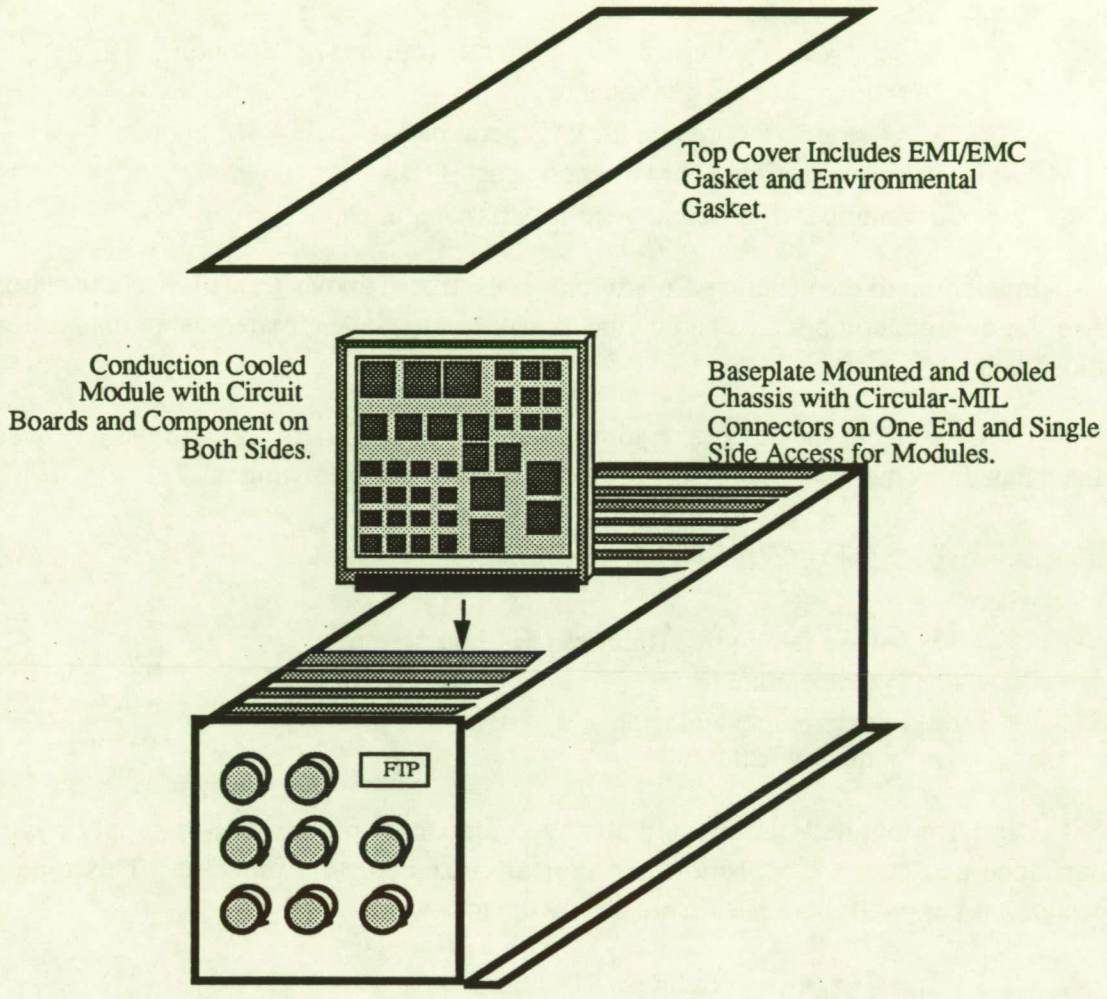

Figure 3-17. AIPS for ALS Packaging Concept

The expression for generic parts failure rate using this method is as follows:

$$\lambda_{Generic}=\sum_{i=1}^{n} N_i(\lambda_G\pi_Q)_i \qquad (3\text{-}1)$$

where

$\lambda_{Generic}$ = total generic parts failure rate (failures / 10^6 hours)
N_i = quantity of i^{th} generic part
λ_G = generic failure rate for i^{th} generic part (failures / 10^6 hours)
π_Q = quality factor for i^{th} generic part
n = number of different generic parts categories

In addition to the generic part failure rates estimated above, the failure contributions due to the discrete components and connections can also be estimated using expressions outlined below.

The opto electronic devices required for the IC and I/O networks and the FTP inter-channel data links have per part failure rates estimated by the following:

$$\lambda_p = \lambda_b\pi_T\pi_E\pi_Q \text{ failures / } 10^6 \text{ hours} \qquad (3\text{-}2)$$

where

λ_b = base failure rate (failures / 10^6 hours)
π_T = temperature factor
π_E = environmental factor
π_Q = quality factor

Digital integrated circuits and analog components require power supply bypass capacitance and ceramic capacitors are typically used for this function. This type of capacitor has per part failure rates estimated by the following:

$$\lambda_p = \lambda_b\pi_E\pi_Q\pi_{CV} \text{ failures / } 10^6 \text{ hours} \qquad (3\text{-}3)$$

where

λ_b = base failure rate (failures / 10^6 hours)
π_E = environmental factor
π_Q = quality factor
π_{CV} = capacitance factor

Film resistors are used to set predefined signal levels, specify analog amplifier gains and limit currents. They have per part failure rates estimated by the following:

$$\lambda_p = \lambda_b\pi_E\pi_Q\pi_R \text{ failures / } 10^6 \text{ hours} \qquad (3\text{-}4)$$

where

λ_b = base failure rate (failures / 10^6 hours)
π_E = environmental factor

π_Q = quality factor
π_R = resistance factor

Connectors (MIL-C-38999 Circular), to be used on the LRUs, have per part failure rates estimated by the following:

$$\lambda_p = \lambda_b \pi_E \pi_P \pi_K \text{ failures / } 10^6 \text{ hours} \quad (3\text{-}5)$$

where

λ_b = base failure rate (failures / 10^6 hours) = $0.02e^x$
$x = \frac{-1592}{T} + (\frac{T+273}{473})^{5.36}$; T = operating temperature
π_E = environmental factor
π_P = number of active contacts in connector
π_K = mating/unmating factor

Module connectors, to be used on the SEM-E modules, have per part failure rates estimated by the following:

$$\lambda_p = \lambda_b \pi_E \pi_P \pi_K \text{ failures / } 10^6 \text{ hours} \quad (3\text{-}6)$$

where

λ_b = base failure rate (failures / 10^6 hours) = $0.216e^x$
$x = \frac{-2073.6}{T} + (\frac{T+273}{423})^{4.66}$; T = operating temperature
π_E = environmental factor
π_P = number of active contacts in connector
π_K = mating/unmating factor

Other connections in the system, such as integrated circuit leads to circuit boards, have failure rates estimated by the following:

$$\lambda_p = \pi_E \sum_{i=1}^{n} N_i(\lambda_{bi}\lambda_{Ti}\pi_{Qi}) \quad (3\text{-}7)$$

where

π_E = environmental factor
N_i = number of connections of the i^{th} type
λ_{bi} = base failure rate of the i^{th} type connection
π_{Ti} = tool type factor for the i^{th} type connection
π_{Qi} = quality factor for the i^{th} type connection

The individual failure contributors, estimated using the above expressions, can then be summed and the result is the total estimated failure rate.

Tables 3-1 through 3-10 list the anticipated parts required for the implementation of the AIPS for ALS avionics and their generic failure rates. Included in the list are the integrated circuits, resistors used for setting signals to desired states, capacitors used for filtering noise from the supply voltages, an estimate of the number of gates required for each integrated circuit, and the number of electrical contacts. Tables 1, 3, 5, 7, and 9 are for operation while the vehicle is on the launch pad and Tables 2, 4, 6, 8, and 10 are for system operation during powered flight, i.e., the boost phase. These data are used in the MIL-HDBK-217E defined calculation of the module mean-time-between-failure (MTBF) which are also shown in the tables.

The failure rates of the AIPS/ALS avionics modules are summarized in Tables 3-11, 3-12, and 3-13 for the three ALS mission phases - launch pad, boost phase, and on-orbit - respectively. The on-orbit failure rates were obtained by using the space environmental factor $\pi_E = 0.9$.

Module:	CPU				Environment:	Ground Fixed	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	Microprocessor	1	105,000	168	0.2802	1.00	0.2802
2	Coprocessor	1	80,000	132	0.2802	1.00	0.2802
3	RAM	4	2,100,000	40	0.5416	1.00	2.1665
4	ROM	30	1,100,000	40	0.2447	1.00	7.3410
5	CPU ASIC	1	20,000	200	0.4103	1.00	0.4103
6	Bus Transceivers	10	200	20	0.1627	1.00	1.6270
7	Resistors	80	n/a	2	0.0037	1.00	0.2960
8	Capacitors	40	n/a	2	0.0170	1.00	0.6800
9	I/O Connector	1	n/a	350	3.17E-12	2.00	6.35E-12
						MTBF:	76445.6

Table 3-1. CPU Module Parts List (Ground Fixed)

Module:	CPU				Environment:	Missile Launch	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	Microprocessor	1	105,000	168	1.2143	1.00	1.2143
2	Coprocessor	1	80,000	132	1.2143	1.00	1.2143
3	RAM	4	2,100,000	40	1.2695	1.00	5.0781
4	ROM	30	1,100,000	40	0.7060	1.00	21.1800
5	CPU ASIC	1	20,000	200	1.8097	1.00	1.8097
6	Bus Transceivers	10	200	20	0.1627	1.00	1.6270
7	Resistors	80	n/a	2	0.0540	1.00	4.3200
8	Capacitors	40	n/a	2	0.0100	1.00	0.4000
9	I/O Connector	1	n/a	350	3.17E-12	1.50	4.76E-12
						MTBF:	27141.9

Table 3-2. CPU Module Parts List (Missile Launch)

Module:	Share Devices				Environment:	Ground Fixed	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	Real-Time Clock	1	1,500	168	0.0750	1.00	0.0750
2	Watchdog Timer	1	750	132	0.0391	1.00	0.0391
3	RAM	4	2,100,000	40	0.5416	1.00	2.1665
4	Maintenance ASIC	1	20,000	200	0.4103	1.00	0.4103
5	Bus Transceivers	10	200	20	0.1627	1.00	1.6270
6	Resistors	80	n/a	2	0.0037	1.00	0.2960
7	Capacitors	40	n/a	2	0.0170	1.00	0.6800
8	I/O Connector	1	n/a	350	3.17E-12	2.00	6.35E-12
						MTBF:	188896.8

Table 3-3. Shared Devices Module Parts List (Ground Fixed)

Module:	Share Devices				Environment:	Missile Launch	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	Real-Time Clock	1	1,500	168	0.3091	1.00	0.3091
2	Watchdog Timer	1	750	132	0.1627	1.00	0.1627
3	RAM	4	2,100,000	40	1.2695	1.00	5.0781
4	Maintenance ASIC	1	20,000	200	1.8097	1.00	1.8097
5	Bus Transceivers	10	200	20	0.2480	1.00	2.4800
6	Resistors	80	n/a	2	0.0540	1.00	4.3200
7	Capacitors	40	n/a	2	0.0100	1.00	0.4000
8	I/O Connector	1	n/a	350	3.17E-12	1.50	4.76E-12
						MTBF:	68683.4

Table 3-4. Shared Devices Module Parts List (Missile Launch)

Module:	COM/INT				Environment:	Ground Fixed	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	COM ASIC	1	20,000	168	0.4103	1.00	0.4103
2	INT ASIC	1	20,000	132	0.4103	1.00	0.4103
3	Oscillator	2	50	40	0.0830	1.00	0.1660
4	Bus Transceivers	10	200	20	0.1627	1.00	1.6270
5	F/O Transmitter	8	250	40	0.0698	1.00	0.5584
6	F/O Receiver	8	2,500	40	1.4704	1.00	11.7632
7	Resistors	80	n/a	2	0.0037	1.00	0.2960
8	Capacitors	40	n/a	2	0.0170	1.00	0.6800
9	I/O Connector	1	n/a	350	3.17E-12	2.00	6.35E-12
						MTBF:	62848.8

Table 3-5. COM/INT Module Parts List (Ground Fixed)

Module:	COM/INT				Environment:	Missile Launch	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	COM ASIC	1	20,000	168	1.8097	1.00	1.8097
2	INT ASIC	1	20,000	132	1.8097	1.00	1.8097
3	Oscillator	2	50	40	1.1000	1.00	2.2000
4	Bus Transceivers	10	200	20	0.2480	1.00	2.4800
5	F/O Transmitter	8	250	40	1.1552	1.00	9.2416
6	F/O Receiver	8	2,500	40	18.3091	1.00	146.4728
7	Resistors	80	n/a	2	0.0540	1.00	4.3200
8	Capacitors	40	n/a	2	0.0100	1.00	0.4000
9	I/O Connector	1	n/a	350	3.17E-12	1.50	4.76E-12
						MTBF:	5926.5

Table 3-6. COM/INT Module Parts List (Missile Launch)

Module:	NIS				Environment:	Ground Fixed	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	IEEE-802.5 ASIC	1	30,000	168	0.4103	1.00	0.4103
2	F/O Transmitter	1	250	40	0.0698	1.00	0.0698
3	F/O Receiver	1	2,500	40	1.4704	1.00	1.4704
4	RAM	8	2,100,000	40	0.5416	1.00	4.3330
5	Bus Transceivers	10	200	20	0.1627	1.00	1.6270
6	Resistors	80	n/a	2	0.0037	1.00	0.2960
7	Capacitors	40	n/a	2	0.0170	1.00	0.6800
8	I/O Connector	1	n/a	350	3.17E-12	2.00	6.35E-12
						MTBF:	112530.4

Table 3-7. NIS Module Parts List (Ground Fixed)

Module:	NIS				Environment:	Missile Launch	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	IEEE-802.5 ASIC	1	30,000	168	1.8097	1.00	1.8097
2	F/O Transmitter	1	250	40	1.1552	1.00	1.1552
3	F/O Receiver	1	2,500	40	18.3091	1.00	18.3091
4	RAM	8	2,100,000	40	1.2695	1.00	10.1561
5	Bus Transceivers	10	200	20	0.2480	1.00	2.4800
6	Resistors	80	n/a	2	0.0540	1.00	4.3200
7	Capacitors	40	n/a	2	0.0100	1.00	0.4000
8	I/O Connector	1	n/a	350	3.17E-12	1.50	4.76E-12
						MTBF:	25886.5

Table 3-8. ICIS/IOS Module Parts List (Missile Launch)

Module:	CN				Environment:	Ground Fixed	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	IEEE-802.5 ASIC	1	30,000	168	0.4103	1.00	0.4103
2	CNASIC	1	20,000	168	0.4103	1.00	0.4103
3	RAM	1	2,100,000	40	0.5416	1.00	0.5416
4	ROM	1	1,100,000	40	0.2447	1.00	0.2447
5	Oscillator	2	50	40	0.0830	1.00	0.1660
6	F/O Transmitter	5	250	40	0.0698	1.00	0.3490
7	F/O Receiver	5	2,500	40	1.4704	1.00	7.3520
8	Resistors	80	n/a	2	0.0037	1.00	0.2960
9	Capacitors	40	n/a	2	0.0170	1.00	0.6800
10	I/O Connector	20	n/a	20	3.17E-12	2.00	1.27E-10
						MTBF:	95694.5

Table 3-9. CN Module Parts List (Ground Fixed)

Module:	CN				Environment:	Missile Launch	
Part No.:	Part Name:	Quantity:	Gates:	I/O Contacts:	Generic Fail Rate:	Env/Temp Factor:	Fail./10^6Hrs:
1	IEEE-802.5 ASIC	1	30,000	168	1.8097	1.00	1.8097
2	CNASIC	1	20,000	168	1.8097	1.00	1.8097
3	RAM	1	2,100,000	40	1.2695	1.00	1.2695
4	ROM	1	1,100,000	40	0.7060	1.00	0.7060
5	Oscillator	2	50	40	1.1000	1.00	2.2000
6	F/O Transmitter	5	250	40	1.1552	1.00	5.7760
7	F/O Receiver	5	2,500	40	18.3091	1.00	91.5455
8	Resistors	80	n/a	2	0.0540	1.00	4.3200
9	Capacitors	40	n/a	2	0.0100	1.00	0.4000
10	I/O Connector	20	n/a	20	3.17E-12	1.50	9.52E-11
						MTBF:	9104.4

Table 3-10. CN Module Parts List (Missile Launch)

Channel Module Complement:	Qty:	Fail. Rate:	Aggregate:
CPU Module	2	1.31E-05	2.62E-05
Shared Devices Module	1	5.29E-06	5.29E-06
COM/INT Module	1	1.59E-05	1.59E-05
NIS Module	2	8.89E-06	1.78E-05
	Channel Failure Rate:		6.51E-05
	Channel MTBF:		15,351.4
Communications Node:	Qty:	Fail. Rate:	Aggregate:
CN Module	1	1.04E-05	1.04E-05
	CN Failure Rate:		1.04E-05
	CN MTBF:		95,694.5

Table 3-11. Channel and CN Failure Rates (Ground Fixed)

Channel Module Complement:	Qty:	Fail. Rate:	Aggregate:
CPU Module	2	3.68E-05	7.37E-05
Shared Devices Module	1	1.46E-05	1.46E-05
COM/INT Module	1	1.69E-04	1.69E-04
NIS Module	2	3.86E-05	7.73E-05
	Channel Failure Rate:		3.34E-04
	Channel MTBF:		2,991.9
Communications Node:	Qty:	Fail. Rate:	Aggregate:
CN Module	1	1.10E-04	1.10E-04
	CN Failure Rate:		1.10E-04
	CN MTBF:		9,104.4

Table 3-12. Channel and CN Failure Rates (Missile Launch/Boost Phase)

Channel Module Complement:	Qty:	Fail. Rate:	Aggregate:
CPU Module	2	2.54E-6	5.10E-6
Shared Devices Module	1	1.01E-6	1.01E-6
COM/INT Module	1	1.17E-5	1.17E-5
NIS Module	2	2.67E-6	5.35E-6
	Channel Failure Rate:		2.31E-5
	Channel MTBF:		43,290
Communications Node:	Qty:	Fail. Rate:	Aggregate:
CN Module	1	7.6E-6	7.6E-6
	CN Failure Rate:		7.6E-6
	CN MTBF:		131,529

Table 3-13. Channel and CN Failure Rates (On-orbit)

3.5.4. Performance Projections of AIPS for ALS Building Blocks

This section highlights some of the important performance projections of the AIPS building blocks for ALS. A complete projection of all the performance metrics is beyond the scope of this study. However, the performance summary given in this section is adequate to do a preliminary definition of the ALS architecture. The performance has been projected using several sources of information. The details of the expected microprocessor performance in the 1992 time frame are provided in the hardware technology survey and projections [2]. A complete set of metrics that are considered relevant for the ALS applications and the empirical performance data for a subset of these figures of merit are described in [1, 8]. The empirical data collected on the current versions of the AIPS hardware and software [1, 8] forms the basis which was used with the estimated parameters of the AIPS/ALS flight system, described earlier in Section 3.5.2, to project the performance of the AIPS/ALS flight system.

The AIPS/ALS performance is partitioned and summarized along the hardware building blocks. Table 3-14 is a summary of the performance projections of the FTP hardware building block and the Local System Services software building block. (The details of the Local System Services software are provided in [9]). Table 3-15 is a summary of the performance projections of the I/O network and IOS hardware building blocks and the I/O System Services software building block. (The details of the I/O System Services software are provided in [10] and [11]). Table 3-16 is a summary of the performance projections of the Inter-Computer (IC) network and ICIS hardware building blocks and the IC Communication Services software building block. (The details of the IC Communication Services software and the ICIS and IC network hardware are provided in [12].) The table presents the time for a site to site communication between a source application task executing on one FTP sending a message to a sink application task executing on a different FTP. It is the time from when the source application task calls the SEND_OUTPUT routine with a message until the sink application has the message available. The total projected time for the AIPS/ALS is 1.278 milliseconds.

1. FTP THROUGHPUT
    1.1 Raw CPU Throughput
        15 MIPS (DAIS Mix) Per Processor
        30 MIPS Per FTP
    1.2 Total Overheads
        1.2.1 FTP Redundancy Management
            CP 1% of Throughput
            IOP 1% of Throughput
        1.2.2 Cross-Channel Synchronization
            CP 5% of Throughput
            IOP 5% of Throughput
        1.2.3 IOP-CP Contention
            CP 5% of Throughput
            IOP 5% of Throughput
        1.2.4 I/O and IC Network Management
            CP 0
            IOP 10% of Throughput
    1.3 Useful FTP Throughput
            CP 89% or 13.35 MIPS
            IOP 79% or 11.85 MIPS
            FTP 25.2 MIPS
2. ADA RUN TIME SYSTEM OVERHEADS
2.1 Ada Run Time System on CP or IOP
    2.1.1 Timer Dispatch 432 instructions (27 μs)
    2.1.2 Local Event 196 instructions (12 μs)
    2.1.3 Simple Context Switch 100 instructions (6 μs)
2.2 IOP-CP Communication
    2.2.1 Global Event
        Source CPU (CP or IOP) 132 instructions (8 μs)
        Sink CPU (IOP or CP) 240 instructions (15 μs)

Table 3-14. AIPS/ALS FTP and Local System Services Projected Performance

3. REDUNDANCY MANAGEMENT

3.1.	Permanent FDIR (Fault Detection, Isolation and Reconfiguration)		
	3.1.1 No Fault Conditions		
		CP	736 instructions (46 μs)
		IOP	680 instructions (42 μs)
	3.1.2 Data Exchange Network Fault		
		CP	2024 instructions (126 μs)
		IOP	120 instructions (75 μs)
	3.1.3 Unsynchronized Channel		
		CP	1536 instructions (96 μs)
		IOP	430 instructions (270 μs)
3.2.	Transient FDIR		
	3.2.1 No Fault Conditions		
		CP	20 instructions (1.25 μs)
		IOP	56 instructions (3.5 μs)
	3.1.2 Data Exchange Network Fault		
		CP	464 instructions(64 μs)
		IOP	64 instructions(4 μs)
	3.1.3 Unsynchronized Channel		
		CP	728 instructions(45.5 μs)
		IOP	64 instructions(4 μs)
3.3	Average Fault Detection, Isolation and Reconfiguration Time		
		10 msec	
3.4	Inter-Channel Data Exchange		
	3.4.1 Fault Tolerant Clock Frequency		2 MHz
	3.4.2 Data Exchange Bandwidth		4 Mwords/sec or 64 Mbits/sec

Table 3-14. AIPS/ALS FTP and Local System Services Projected Performance (cont.)

Raw I/O Network Bandwidth = 100 Mbits/Sec

1.	COMMUNICATION MANAGEMENT OVERHEADS IOP	
1.1	I/O Processing	
	1.1.1 10 Transaction Chain	7040 instructions (440 μs)
	1.1.2 8 Transaction Chain	5600 instructions (350 μs)
	1.1.3 2 Transaction Chain	3360 instructions (210 μs)
2.	REDUNDANCY MANAGEMENT OVERHEADS	
2.1.	Network FDIR	
	2.1.1 Failed Channel	929 instructions (58 μs)
	2.1.2 Failed IOS	14800 instructions (925 μs)
	2.1.3 Failed Link (Leaf Node)	24134 instructions (1,508 μs)
	2.1.4 Failed Leaf Node	35344 instructions (2,208 μs)
	2.1.5 Failed Link (Branch)	24664 instructions (1,541 μs)
	2.1.6 Failed Branch Node	46264 instructions (2,892 μs)
2.2.	Network Growth	
	2.2.1 Full Diagnostics	473600 instructions (29,600 μs)
	2.2.2 No Diagnostics	81600 instructions (5,100 μs)
	2.2.3 Single Chain	19366 instructions (1,200 μs)

Table 3-15. AIPS/ALS I/O Network and I/O System Services Projected Performance

Raw IC Network Bandwidth = 100 Mbits/Sec

LOCATION	FUNCTION	PROJECTED PERFORMANCE
Source FTP	SEND_OUTPUT	493 instructions (30.8 μs)
Source FTP	Set Event	187 instructions (11.67 μs)
Source FTP	MSR Task	760 instructions (47.5 μs)
IC Net	Time on Network	67 μs
Sink FTP	Ave Time for Polling for msg	1000 μs
Sink FTP	ICIS RM	1293 instructions (80.8 μs)
Sink FTP	Context Switch	120 instructions (7.5 μs)
Sink FTP	MSR Task	426 instructions (26.7 μs)
Sink FTP	GET_INPUT	102 instructions (6.67 μs)

Total Task-to-Task Communication Time on IC Network = 1.278 msec

Table 3-16. AIPS/ALS IC Network and IC Communication Services Projected Performance

4.0 PRELIMINARY ALS AVIONICS ARCHITECTURE

As discussed in Section 1 on the design for validation methodology, AIPS for ALS configuration(s) are defined using as inputs the AIPS architectural rules, guidelines and attributes, the projected reliability, performance, physical characteristics and other attributes of the building blocks, and the ALS avionics requirements. The ALS avionics requirements were described in Section 2 and the performance and reliability parameters of the AIPS hardware and software building blocks, projected in the ALS time frame, have been summarized in Section 3.

The process of matching the avionics requirements with the building block capabilities is a multidimensional problem. However, it can be simplified by decomposing the requirements into two orthogonal sets each of which can be mapped independently of the other as a first order approximation and each of which determines a different aspect of the architecture. The performance related ALS requirements such as throughput, memory, transport lag, input/output latencies, etc. determine the virtual avionics architecture. The reliability related ALS requirements such as probability of mission success, launch availability, launch pad maintenance, function criticality, etc. determine the physical avionics architecture.

A preliminary definition of the AIPS for ALS avionics architecture has been carried out using the architecture synthesis process described in Section 1.2. Sections 4.1 and 4.2 describe the preliminary virtual and the physical avionics architecture for ALS, respectively. Using the projected performance and reliability parameters of the building blocks, analytical models of the ALS architecture were solved to predict the avionics reliability and the availability for the ALS mission scenario. These results are described in Section 4.3. Section 4.4 concludes with some thoughts on future work necessary to complete the ALS architecture synthesis.

4.1 Virtual Architecture

The ALS functions that require the highest throughput are IMU and GPS processing, Kalman filter, adaptive guidance and control, and propulsion control. Using the Martin Marietta supplied processing requirements, all the non-propulsion functions require a total of about 8.8 MIPS throughput (including margins) with a peak instantaneous throughput of about 3 MIPS to perform part of the Navigation/IMU Processing function. The total is the sum of throughput requirements for all tasks. The peak is the throughput required to perform an indivisible task. If a processor is not fast enough to execute all the tasks then the workload can be assigned to a number of parallel processors. However, a processor with at least the peak throughput is required to run the highest throughput indivisible task. The most demanding non-propulsion peak throughput requirement of 3 MIPS for the ALS results from the Bending Processing task, which is

part of the Nav/IMU function. The Bending Processing task executes only 300 instructions per iteration. At the 100 Hz iteration rate of this task, the average margined throughput requirement is about 0.03 MIPS. However, in order to meet the processing lag constraint of 0.1 msec, the peak throughput requirement rises to 3 MIPS. Since this is considerably below the expected processor throughput for the ALS time frame, this requirement does not pose a problem. However, if such a fast processor were not available, one would need to analyze the Bending Processing task further to see if it can be parallelized into two or more parts.

Since all of the non-propulsion functions taken together require less than the total useful throughput projected to be available in an ALS Fault Tolerant Processor, all of the non-propulsion functions can be allocated to a single FTP. We will call this the core FTP. If the total throughput requirement had exceeded the capacity of a single FTP, some criteria such as function criticality, interfunction communication rates, etc. would have had to be used to partition the ALS functions into groups and allocate them to different FTPs. The core FTP will have a growth margin of 16.4 MIPS (25.2 MIPS useful FTP throughput from Table 3-14 minus 8.8 MIPS throughput required for all non-propulsion functions from Section 2) or 186 per cent. This is in addition to the throughput margins already built into the processing requirements at the task level as described in Section 2.

Another constraint on allocating functions to a single FTP is the FTP data exchange bandwidth. All the sensor inputs must be made congruent using the data exchange hardware before they can be used by the applications functions. All the outputs are also usually voted before they are sent out to actuators. The output voting also uses the same data exchange hardware. The total I/O bandwidth requirement for all the non-propulsion functions is estimated to be 11.2 Mbits/sec. This is well within the projected bandwidth of 64 Mbits/sec of the FTP data exchange mechanism.

The ALS avionics functions allocated to the core FTP are listed below.

CORE FTP Functions

1 Central Control & Processing
2 Winds Ahead Determination
3 Vehicle Power System Management
4 Steering & Staging Control
5 Sensor Processing
7 Command & Telemetry Processing
8 Range Safety & Destruct
9 Programmable Payload

Each ALS engine requires a controller which must be capable of providing a useful total throughput of about 4.8 MIPS with a peak of about 6.4 MIPS. Since the controller is to be colocated with each engine, a dedicated FTP is necessary to host propulsion control functions for each ALS engine. A single FTP per engine would provide a growth margin of 20.4 MIPS or 400 per cent. If the colocation of the engine controller with the engine were not a high level requirement, a single FTP could be allocated to control a group of 4 engines since it will have the throughput and the data exchange bandwidth necessary to control 4 engines simultaneously. This would reduce the total avionics hardware substantially and would still leave a growth margin of 6 MIPS. However, it may complicate the logistics associated with assembling and testing each individual engine. The propulsion controller I/O bandwidth requirement of less than 0.1 Mbits/sec is quite modest and is easily accommodated by the FTP.

The virtual intercomputer communication architecture for ALS is quite straightforward. It will be a virtual bus that interconnects the core FTP to all the propulsion control FTPs. The bus bandwidth requirement is quite modest since all the non-propulsion functions are colocated in a single processing site. The communication bandwidth between the core FTP and all the propulsion FTPs in the ALS launch phase consists of propulsion commands going to the engine FTPs and the engine status data flowing back to the core FTP. The requirement for this is expected to be about 10 Mbits/sec which is well within the projected intercomputer bandwidth of 100 Mbits/sec for the ALS IC network.

The virtual I/O communication architecture for the ALS would consist of a number of parallel, virtual buses that interconnect ALS sensors and actuators to the FTPs. The sensors and actuators in each engine will be connected to the FTP controlling that engine on a local bus. The core FTP will interface with the IMU, GPS, and all the other sensors required to perform the non-propulsion functions. The number of parallel I/O buses required in the core FTP and the propulsion FTPs is determined by the number of sensors and the frequency of their access. Since the total I/O bandwidth required for all the core FTP functions is 11.2 Mbits/sec, a single virtual I/O bus should suffice. This preliminary I/O bus definition may be changed if the detailed performance modeling shows that some performance criterion such as the transport lag can not be satisfied with a single bus. The I/O bandwidth required for the propulsion FTPs is even less, only 0.1 Mbit/sec and can be easily satisfied with a single virtual I/O bus.

The vehicle health monitoring functions, their processing requirements or the number of sensors and the information flow associated with the sensors were not made available to CSDL during the course of this study. Therefore no conclusions can be drawn at this time about the virtual architecture that is required to read these sensors and perform the vehicle health monitoring functions. However, it is possible that the number of these

sensors and the frequency with which they are read can easily exceed not only the available bandwidth of I/O buses but also the capability of the processors to process the enormous amount of collected data in real time. If, on the other hand, the majority of the sensors are to be used to collect vehicle health data for launch pad monitoring (which would relax the real time processing constraint) and/or for post-flight analysis, then the problem can be dealt with quite effectively by providing a separate system which is dedicated to sensor collection (but no real time analysis) and telemetry.

4.2 Physical Architecture

The architectural parameters that determine the AIPS physical organization for the ALS avionics include: redundancy level of FTPs; redundancy levels of sensors, actuators and other I/O devices; cross-strapping of I/O devices to channels of FTPs and redundancy level of interfaces; trade-offs between FTP redundancy (triplex or quad) versus system redundancy (N or N+1 FTPs) for availability; redundancy level of intercomputer and I/O networks; and physical topologies of networks.

Since all ALS functions are flight-critical and cannot be suspended for more than a few milliseconds, a fault-masking computational architecture is required. Therefore, all FTPs need to be at least triplex at the launch time. Earlier reliability modeling of the FTP done for the ALS mission has shown that a triplex FTP has sufficient reliability to meet the ALS requirements for short mission durations (of the order of several hours). Also, due to the short mission time, no in-flight reconfigurations of the FTPs will be required to meet the reliability requirement. That is, after a channel fails in-flight in an FTP, the outputs of the failed channel to actuators will be disabled by the Monitor/Interlock circuitry. However, no effort would be made to recover the failed channel and retry it to ascertain if the channel failure was caused by a transient fault. However, for longer ALS missions, of the order of tens of hours, it may become necessary to change this policy. In the present AIPS FTP design, the realignment of the state of the faulted channel to the state of the good channels is performed in software. Although this process is relatively fast for small amounts of volatile memory, it can become unacceptably long for certain combinations of large RAM and high iteration tasks that cannot be suspended. A new, hardware implemented, channel realignment scheme has been developed to increase the speed of realignment and to do it in background without suspending any applications tasks [13,14]. This will allow the ALS FTPs to recover from transient faults and continue to provide the fault masking capability for longer ALS missions. The projected reliability of the ALS FTP as a function of mission time, computed using the module failure rates projected for the ALS time frame, is given in the next subsection.

The intercomputer bus will be physically implemented as a redundant network. A triplex redundancy level is required for dynamic masking of intercomputer communication faults. However, it may be possible to relax this requirement somewhat and use only dual redundant networks if sufficient progress can be made in the use of authenticated protocols. The physical topology of the IC network will be determined by the constraints imposed on the physical location of the core FTP and the propulsion control FTPs.

The I/O buses will also be implemented as redundant networks. Because of the inherent redundancy in the sensors and actuators, the I/O networks need not be triplicated. Dual redundant networks with a few spare links in each network are expected to meet the reliability and availability requirements of ALS. The I/O network topologies will be determined by the physical location of sensors and actuators on the launch vehicle.

The I/O and IC networks will be operated as static, non-reconfigurable buses in-flight for short missions. For mission lengths of tens of hours, it may become necessary to make them reconfigurable. The flight system performance projections, summarized in Section 3.5, show that the in-flight reconfigurations of networks can be accomplished without suspending any of the ALS applications tasks. The reconfigurability of networks is also intended to tolerate various link and node failures during the relatively long 1 to 2 weeks on the launch pad. This could obviate the launch pad repairs and launch delays resulting in a reduced overall cost of the ALS.

The core FTP would also be provided with a spare channel to obviate launch pad repairs. Thus, the core FTP would be quad redundant rather than triplex assuring a high probability of having at least a triplex level of redundancy at the scheduled launch time. The FTP will be operated as a quad unit with all four channels active and performing the same tasks in synchronism. This provides a better fault coverage for the fourth channel in comparison to a triplex with a standby spare channel. Since the number of FTPs in the candidate ALS architecture is small (only 1 is required to perform all the non-propulsion functions), this strategy of providing a spare channel in the FTP is quite cost effective. However, if the ALS requirements were to increase or the projected FTP throughput were to be significantly lower, more FTPs would be required to perform the ALS functions. A more cost effective strategy in this case may be to add a spare triplex FTP and provide the sensor and actuator cross-strapping necessary to reassign functions to processing sites. Use of global I/O networks enables a straightforward and cost effective sharing of sensors and actuators among a number of FTPs. The next subsection provides numerical modeling results on the availability of the ALS core avionics.

The sparing of the propulsion control FTPs may or may not be necessary depending upon the philosophy of launching the ALS with fewer than a complete

complement of engines. The baseline ALS vehicle design supplied to CSDL has up to 17 engines. If it is necessary to have all the engines operating prior to launch (which is the current launch philosophy), it will be necessary to make the engine control FTPs quad redundant as well. However, if a new philosophy of launch with failed components is applied to the ALS, and this can be more cost effective and safe if designed-in from the outset, then it is no longer necessary to make sure that every engine controller has a fault masking capability at the launch time.

4.3. Physical Characteristics

Each AIPS/ALS Fault Tolerant Processor channel will consist of six SEM-E modules -- two CPU modules, one Shared Devices module, one Communicator and Interstage module, and two Network Interface Sequencer modules. The SEM-E modules are 5.88 in. x 6.68 in. x 0.6 in. and weigh approximately 1 lb. each. Power supplies will also meet the SEM-E form-factor and will occupy two additional module slots in each channel. Power dissipation is estimated to be 8 W for each CPU module, 2 W for the Shared Devices module, 3 W for the Communicator and Interstage module, and 6 W for the NIS module. Power dissipation for each channel, therefore, is approximately 41 W including the power dissipation for 80% efficient power supplies. Each channel will be packaged in a single chassis and the volume of the chassis is approximately 902 cu. in. (8 in. x 11.5 in. x 9.8 in) or .52 cu. ft. The weight of each channel, including the power supplies and chassis, will be approximately 27 lb.

The projected physical characteristics of an ALS triplex and a quadruplex FTP are summarized in Table 4-1.

	Triplex FTP	Quad FTP
Power	124 W	165 W
Weight	81 lb.	108 lb.
Volume	1.5 cu. ft.	2 cu. ft.

Table 4-1. AIPS/ALS Fault Tolerant Processor Projected Physical Characteristics

It is illustrative to compare the physical characteristics of the AIPS/ALS FTPs to computers on-board current launch vehicles. Two examples of current generation of space launch vehicles are the space shuttle and the Titan.

Each of the three Space Shuttle Main Engines (SSME) has a dedicated engine control computer built by Honeywell. The current version, called the block II, is a dual redundant processor that controls a number of main propellent valves, solenoids and spark

igniters. The inputs to the controller include a number of pressure and temperature sensors. This computer weighs 200 lbs and consumes 490 Watts in the standby mode. During flight, the controller power consumption is 600 watts. The controller dimensions are 23.5" x 14.5" x 17.0" or about 3.35 cubic feet.

The Titan series of unmanned launch vehicles use Magic 352 computer produced by the Delco division of General Motors. This guidance and control computer is single string, i.e., has no redundancy. It weighs 68 lbs, consumes 220 watts of power and occupies 1.5 cubic feet.

It can thus be seen that the physical characteristics of the AIPS/ALS Fault Tolerant Processors are well within the realistic constraints that might be imposed on the launch vehicle avionics.

4.4 Reliability and Availability Projections

The reliability and availability models of the AIPS building blocks are described in detail in Section 4 of the accompanying report: Advanced Information Processing System: Design and Validation Knowledgebase [1]. These models were executed using the failure rates and recovery rates for a 1992-technology ALS FTP summarized earlier in this report in Section 3.5. The availability and the reliability of the AIPS for ALS avionics architecture are presented in Tables 4-2 and 4-3. Table 4-2 lists these results for the case where the core ALS FTP and all the propulsion FTPs are quadruply redundant. The results for the ALS configuration consisting of all triplex FTPs are summarized in Table 4-3.

Results have been tabulated for the baseline AIPS for ALS avionics architecture as well as several variations on the baseline. The baseline architecture, as described earlier, consists of a core FTP that performs all the non-propulsion functions and a dedicated propulsion control FTP for each engine. The baseline vehicle design from Martin Marietta consists of 17 engines. Therefore the baseline architecture consists of 18 FTPs. The number of FTPs in the avionics architecture is denoted by the term Critical Minimum Complement. Results have been tabulated for a CMC of 1, 2, 3, and 4, in addition to the baseline case of 18 FTPs. This allows one to examine the avionics availability and reliability only for a subset of functions such as the non-propulsion functions which require only 1 FTP to be operational. Furthermore, if a group of engines is allocated to a single FTP, as suggested in the earlier discussion on architecture synthesis, or the number of engines is fewer than 17, then new RMA numbers can be quickly obtained from these tables.

The tables present the per cent Launch Availability of the avionics architecture, for a specified complement of FTPs, in column 1. The Launch Availability is defined to be the probability of the CMC of FTPs being in a fault masking state at the end of a week of

operation on the launch pad. An FTP is capable of masking faults if it has at least three operational channels. The next column gives the probability of failure of the ALS avionics during the powered flight or the boost phase. A failure of any one FTP out of the CMC is considered a total avionics failure. Finally, the last column gives the same failure probability for the on-orbit phase of ALS.

The avionics reliability for the whole ALS mission, i.e., the boost phase plus on-orbit phase, R(ALS), can be calculated from the failure probabilities of the two mission phases, PF(Boost) and PF(Orbit), using the following equation:

$$R(ALS) = \{1\text{-}PF(Boost)\} * \{1\text{-}PF(Orbit)\}.$$

However, for small failure probabilities, as is the case for the ALS mission, the mission failure probability is approximately just the sum of the failure probabilities for each mission phase. That is,

$$PF(ALS) = PF(Boost) + PF(Orbit).$$

The RMA models and their underlying assumptions are described in detail in Section 4 of [1]. However, it is useful to briefly recapitulate some of the important assumptions here. These are as follows.

- The launch pad operations last 200 hours (approximately 1 week), the boost phase lasts 0.2 hours (12 minutes), and the on-orbit phase lasts 50 hours.
- Failure of any single module in an FTP channel results in the loss of that channel.
- In the boost phase, no recovery is attempted from transient faults, i.e., transients are treated as permanent faults.
- On-orbit, transient failure recovery is performed by the FTP in the background without suspending applications tasks.
- Transient faults occur 10 times more frequently than permanent faults.
- Permanent faults occur at a rate dependent on the mission phase. The permanent failure rates for the AIPS/ALS modules are as summarized in Tables 3-11, 3-12, and 3-13.
- The average recovery time from a fault is assumed to be 20 milliseconds.
- Quad and triplex fault recovery coverages are assumed to be 1.0. Duplex fault coverage is assumed to be 0.9.

Critical Minimum Complement	Launch Availability	Failure Probability (Boost Phase)	Failure Probability (On-orbit)
1	99.88%	$8.90*10^{-9}$	$5.33*10^{-7}$
2	99.76%	$1.78*10^{-8}$	$1.07*10^{-6}$
3	99.64%	$2.67*10^{-8}$	$1.60*10^{-6}$
4	99.52%	$3.56*10^{-8}$	$2.13*10^{-6}$
•	•	•	•
•	•	•	•
18	97.86%	$1.60*10^{-7}$	$9.59*10^{-6}$

Table 4-2. Availability and Reliability of ALS Avionics (Quad FTPs)

The following conclusions can be drawn from the analytical evaluation of the AIPS for ALS avionics architecture. For the baseline architecture, consisting of 18 FTPs, it is necessary to provide a quadruple level of redundancy, in order to meet the 95% launch availability requirement. The baseline architecture is expected to have 97.86% availability. This configuration will also meet the mission reliability goals. Specifically, the mission probability of failure is expected to be $1.6*10^{-7}$ for the boost phase, $9.59*10^{-6}$ for an extended on-orbit phase (50 hours), and a total of $9.75*10^{-6}$ for the whole ALS mission. This exceeds the goal of 10^{-5} just slightly.

Critical Minimum Complement	Launch Availability	Failure Probability (Boost Phase)	Failure Probability (On-orbit)
1	96.20%	$1.67*10^{-7}$	$1.01*10^{-5}$
2	92.50%	$3.34*10^{-7}$	$2.02*10^{-5}$
3	89.0%	$5.01*10^{-7}$	$3.03*10^{-5}$
4	85.6%	$6.68*10^{-7}$	$4.04*10^{-5}$
•	•	•	•
•	•	•	•
18	49.8%	$3.01*10^{-6}$	$1.82*10^{-4}$

Table 4-3. Availability and Reliability of ALS Avionics (Triplex FTPs)

The contributions to the unavailability and unreliability come predominantly from the propulsion avionics since 17 out of the 18 FTPs are for engine control. The availability of the non-propulsion avionics, which consist of just 1 FTP, is 99.88% for a quad and 96.2% for a triplex FTP. Similarly, the mission loss probability, attributable to non-

propulsion avionics, is $5.42*10^{-7}$ for the quad FTP. Evidently, one needs to reexamine the requirement of dedicating a controller to each engine. If, for example, an FTP was configured to control 4 engines, which it is capable of doing based on the performance projections, only 4 FTPs will be necessary to control 16 engines. This would reduce the probability of mission failure due to a failure of propulsion avionics to $2.16*10^{-6}$ using quad FTPs. The launch availability would improve to 99.52%.

The ALS vehicle and engine designers should seriously examine the option of integrating engine controllers outside the engine and with the core avionics. This would not only improve the overall ALS reliability and availability, as demonstrated above, but would also result in reduced weight, volume, power and cost.

4.5 Architecture Summary and Conclusions

A preliminary AIPS-based fault tolerant computer system architecture has been configured to meet the ALS performance, reliability, and availability requirements. A single quadruply redundant AIPS Fault Tolerant Processor, the core FTP, will perform all the non-propulsion functions required in the ALS. Additionally, there will be a propulsion control FTP dedicated to each engine. The core FTP will access the guidance, control, navigation and other sensors and actuators on one redundant I/O network. Each of the engine control FTPs will access engine sensors and actuators on a dedicated I/O network. The core FTP and all of the engine control FTPs will be connected by a fault-masking triply redundant intercomputer network.

For short ALS missions, lasting an hour or less, it will not be necessary to reconfigure the FTPs or the I/O and IC networks. Redundant hardware would provide sufficient fault masking capability to meet the ALS reliability requirement. However, for longer ALS missions, lasting 1 hour to 48 hours, it will be necessary to re-integrate FTP channels affected by transient faults and to reconfigure the I/O and IC networks. The performance projections show that these in-flight reconfigurations can be accomplished without suspending any of the ALS applications tasks. The reconfigurability of the networks is also intended to obviate expensive launch pad repairs.

The AIPS for ALS architecture defined here is preliminary in nature but shows that the ALS performance and reliability requirements can be met by the AIPS hardware and software building blocks that are built using the state-of-the-art technology available in the 1992-93 time frame. The level of detail in the architecture definition reflects the level of detail available in the ALS requirements. As the avionics requirements are refined, the architecture can also be refined as well as defined in greater detail with the help of analysis and simulation tools. For example, the functions in the core FTP need to be allocated to the

computational processor and the I/O processor. This requires a more detailed enumeration of interfunction communication requirements and I/O communication requirements. Also, no effort was expended on defining the detailed I/O architecture. This requires as inputs the sensor details such as the number and type of sensors, their failure rates, and so on. This information can be used to define redundancy levels of sensors and allocate sensors to different redundant layers of the I/O network.

Several variations on the baseline architecture presented here are also possible and should be modeled and analyzed. These include allocating several engines to one propulsion control FTP, investigating the effects of launch with failures, and using authentication for the I/O and IC networks.

5.0 IMPACT OF AIPS/ALS ARCHITECTURE ON ALS COST

5.1 Introduction

The main motivation of the National Aeronautics and Space Administration and the Department of Defense in sponsoring its Advanced Launch System program is to realize a substantial reduction in the recurring launch costs over the present launch systems. The avionics architecture selected for the ALS will have an impact on the recurring launch costs. However, the impact is not limited to the cost of the avionics system itself or its cost in terms of the weight or physical displacement it will add to the vehicle. Being mission critical, the reliability of the avionics suite will directly influence vehicle failure. Obviously, a failure of the vehicle can incur very sizable costs. A failure while awaiting launch on the launch pad may require repair and incur the cost of the repair and the cost resulting from interrupting and delaying the launch. A failure during launch can cause a loss of the vehicle and of the payload it is carrying. If fault tolerance is also an attribute of the avionics suite, there is the possibility of exploiting this trait operationally to reduce costs. Therefore, to address the requirement of reducing recurring launch costs, this characteristic of the avionics suite needs to be accurately assessed as it is being designed and developed. In this way, when design freedom exists, choices can be made which will ultimately reduce launch costs.

The primary objective of this study is to demonstrate a methodology for investigating the impact of the avionics suite on the recurring launch cost of the ALS. All the factors influencing cost are investigated, however, this study focuses on the methodology for quantifying the contribution to the recurring launch costs due to the reliability and availability characteristics of the avionics suite.

Two secondary objectives are pursued. The first is to evaluate the impact of using an AIPS Fault Tolerant Processor (FTP) as the avionics computer on the recurring launch cost of the ALS. The second objective is to investigate the effect of a number of design and operational parameters on the recurring launch cost of the ALS.

5.2 Problem Definition

5.2.1 General Description

The anticipated mission of the ALS is to launch payloads into orbit about the earth. For the purpose of this study, the mission is considered to have two separate phases. The first phase of the mission begins when the vehicle is delivered to the launch site, assembled and then parked on the launch pad. While on pad the vehicle would undergo testing, be loaded with fuel (if necessary), have its status monitored and undergo the other normal preflight activities associated with an orbital rocket launch. The vehicle would probably remain on the pad for a relatively long length of time. The vehicle would most likely be parked a week or more before the scheduled launch date. When the scheduled launch time

did arrive, if the vehicle is judged to be fit for launch, the engines would be ignited and the vehicle launched. Launching the vehicle would end the first phase and begin the second. The second phase encompasses the time from launch through ascent until the payload is delivered to orbit. The time of the launch/ascent phase would be comparatively short — on the order of minutes or hours.

While parked on the pad, the avionics system would be monitored. The Vehicle Health Monitoring System (VHMS) is assumed to be a passive subsystem of the ALS whose function includes relaying the operational status of the avionics system to the ground operations center. The ground operations center has the ability to control the progress of the countdown to launch. So, based on the known operational status of the avionics system, the ground operations center may proceed with the countdown sequence or interrupt the launch before ignition of the engines.

The bulk of the avionics suite would be carried to orbit along with the payload. Since the avionics suite includes the sensors, actuators and other equipment which would be located on the initial stages of the vehicle, parts of the system would be discarded as the initial stages are separated and abandoned. However, it is assumed that the bulk of the avionics suite — including the avionics computer — would be located in the final stage and brought to the delivery orbit of the payload.

Some other assumptions are made to limit the scope of this study. First, the mission is assumed to be unmanned. The missions of the ALS may include manned missions. However, associating a monetary cost with the loss of human life brings in complications which detract from the objective of this study. (The issue should be approached as a safety requirement for the vehicle and not a parameter which can be traded-off against cost.) Second, the vehicle is assumed to be non-recoverable. Again, the ALS may include recoverable subsystems. But, assuming the ALS to be non-recoverable simplifies the analyses. Third, the ALS is presumed to have a high probability of mission success.

5.2.2 Contributors to Cost

The contributors to the cost of the avionics suite are divided into three categories — the cost of the avionics system itself, the cost of its weight, and the cost of its unreliability. The cost of the avionics system itself is subdivided into fixed and recurring costs. The fixed costs are the design and development associated with producing the avionics suite, exclusive of the components which will actually perform the mission. This includes the development and construction of prototype, validation and verification testing, etc. — any costs incurred to produce the working design of the avionics system. The recurring costs are the costs of the avionics suite directly related to the specific mission. That is, the actual costs to manufacture the system and integrate it with the vehicle given that a validated design exists. This includes the costs resulting from any design changes to the hardware or software and any testing (acceptance, validation, etc.) which are mission specific.

The second category for the cost of the avionics suite is the cost of weight. Since the bulk of the avionics system rides with the payload all the way to orbit, it's weight subtracts directly from the ALS payload lift capability. Therefore, a cost penalty is associated with the weight of the avionics system.

The third category is the cost of unreliability. The cost of unreliability differs from the other two costs in that the actual cost is contingent upon a failure occurring. Therefore the cost will depend on a probabilistic event occurring — a failure of one or more components within the avionics suite.

The cost of unreliability is broken up into two subcategories which relate to when the cost is incurred — on the launch pad and during launch/ascent. The cost of unreliability on the launch pad is derived from actions taken while the vehicle is on the launch pad because of detected failures within the avionics suite. While the ALS sits on the pad awaiting launch, a failure of a component in the avionics system can occur. If the failure is detected then there is the option of taking some action. If the system is fault tolerant and sufficient capability for launch still exists then a decision could be made to continue the countdown. If sufficient capability for launch no longer exists or it is not desirable to launch with the available configuration, then the countdown can be interrupted and the failure repaired. Therefore, the cost of unreliability on the launch pad is identified to have two contributors, the cost of interrupting the count-down and the actual cost of repairing or replacing the failed component.

Failures of the avionics system occurring while the vehicle sits on the launch pad may go undetected. The immediate effect is that no costs are incurred as a result of the first phase of the mission. However, these failures may manifest themselves at the time of launch and would incur the same costs as catastrophic failures during the launch/ascent phase.

The second subcategory of the cost of unreliability is the cost due to avionics system failures during launch/ascent. The cost of a failure of the avionics suite during ascent is the cost of the entire mission (avionics system, launch vehicle, payloads and operational costs associated with the mission), the cost of downtime following the failure, the cost of a post launch failure analysis, and less tangible costs such as the cost of launch unavailability, the loss of user confidence, and damage to the national image. Note that the failure modes leading to a vehicle failure during flight may include both avionics system component failures during the flight and/or during the time on the launch pad.

5.2.3 Architecture Definitions

To demonstrate the applicability of the methodology, the architectures to be focused on for the avionics suite should be typical of the systems utilized for this application. However, if the systems being analyzed are very complex, the task of modeling them will detract from the primary objective — illustrating the methodology. Therefore, with the

intention of satisfying the primary objective in what is a limited study, the two architectures investigated are defined as simplified versions of two potential candidates for the ALS avionics.

The two generic architectures defined for the focus of this study are shown in Figures 5-1 and 5-2, respectively. Architecture 1, shown in Figure 5-1, is an architecture usually proposed for this type of system. Architecture 2 is presented in Figure 5-2 and is a distilled version of an AIPS FTP. This permits a comparative evaluation of the impact to the recurring launch cost of using an AIPS FTP as the avionics computer. More comprehensive descriptions of the two defined architectures are in the following sections.

5.2.3.1 Architecture 1

Architecture 1 represents an elementary method of incorporating fault tolerance into the avionics system. The three channels (Channel-1, Channel-2 and Channel-3) are identical computers. Each channel individually processes the information from its dedicated sensor and generates an output to control an actuator. (Note that in the actual avionics system, each channel would receive data from many sources and generate outputs to many other subsystems.) In this generic representation, each of the three sensors (Sensor-1, Sensor-2 and Sensor-3) measures the same quantity. Therefore, in the absence of failures each channel would ideally produce the same output. The outputs of the three channels would then be voted at the actuator and a majority consensus would determine the control of the actuator. That is, if the output of one channel was in error (because of a fault in a sensor or actuator), two of the three channels would still be providing the correct output. The Voting Actuator would ignore the incorrect output and follow the output of the two valid channels.

A number of relevant attributes are assumed to be associated with Architecture 1. These are listed below:

1. The replicated sensors are wired to different processing channels.
2. No sensor data is being exchanged between the channels.
3. The channels are asynchronous.
4. There is no fault masking of channel failures.
5. No Fault Detection, Isolation and Reconfiguration (FDIR) is being done within the avionics system; errors due to sensor or channel failures will not be removed from the input to the voting actuator.
6. The VHMS monitors the health of all components.

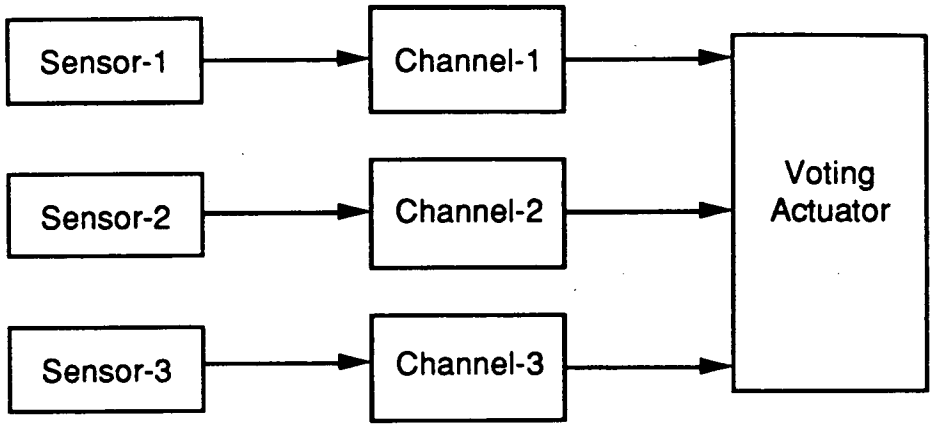

Figure 5-1. Architecture 1

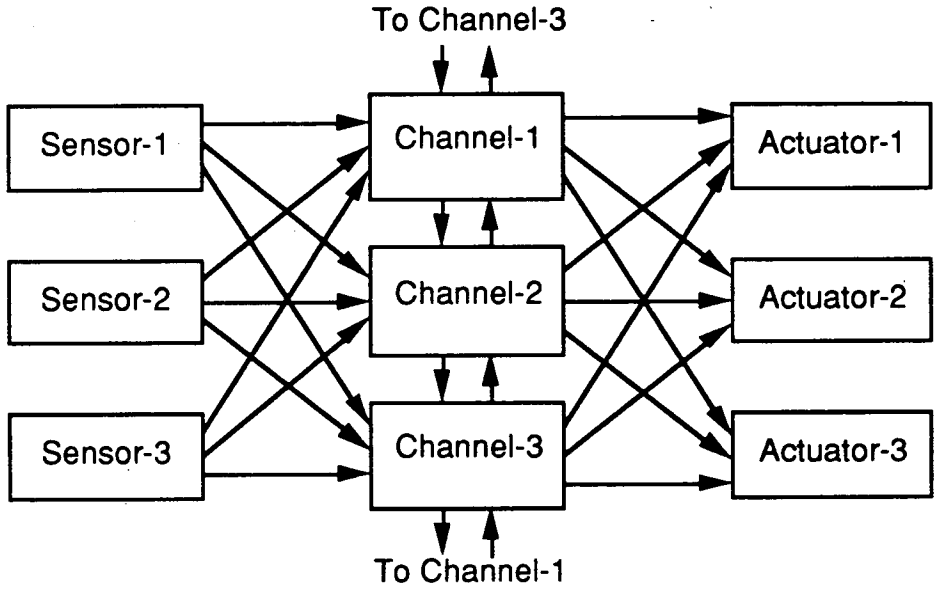

Figure 5-2. Architecture 2

Architecture 1 contains a single point of failure—the failure of the Voting Actuator. Architecture 1 is a generic representation of a fault tolerant architecture typically proposed for the avionics suite of this type of system. A single string system is replicated and some element must arbitrate between the outputs of the strings. In this case, the Voting Actuator acts as the arbitrator and represents a single point of failure for the system. If the Voting Actuator is highly reliable and its potential failure is easily predictable (which are both plausible properties for the Voting Actuator), then the redundancy within the architecture can be expected to significantly contribute to the launch reliability of the avionics suite. Note that the difference in the redundancy levels for the three functional groups (Sensors, Computers and Actuator) does not preclude a balanced design with regard to the reliability allocation for these groups. A balanced design balances the reliability budget allocated to

C-2

each of the distinguished functional groups, not the means utilized to achieve that reliability.

5.2.3.2 Architecture 2

Architecture 2 represents a very basic application of the AIPS technology into the avionics system of the ALS. Channel-1, Channel-2 and Channel-3 represent the three channels of a triplex FTP. Sensor-1, Sensor-2 and Sensor-3 are the same three sensors used for Architecture 1. However, their outputs are now hard-wired to each channel of the FTP. The replicated actuators (Actuator-1, Actuator-2 and Actuator-3) individually perform the same function as the Voting Actuator in Architecture 1. However, they do no voting of channel outputs. Only the output of one channel would be enabled to control only one actuator at any one time.

The relevant attributes of Architecture 2 are listed below:

1. The replicated sensors are cross-strapped to all processing channels.
2. In the absence of sensor failures, mid-value selection of sensor data is performed within the channels.
3. There is source congruency between channels .
4. There is micro-frame synchronization of the channels.
5. Frequent, exact comparisons of channel outputs and intermediate results are performed in order to detect computational failures.
6. There is fault masking of channel failures.
7. Failed components are isolated from the control chain.
8. The health of all components are monitored by the FTP and relayed to the VHMS.

The core FTP of the preliminary ALS avionics architecture is a quadruply redundant AIPS FTP. The rational for defining Architecture 2 as a triplex FTP is to make it more comparable to Architecture 1. Keeping the redundancy level similar to the more typical architecture utilized for this type of application allows a cost comparison to be made regarding the implementation of the redundancy and not just a more general comparison between two architectures.

5.3 The Cost Model

The cost model is implemented as a number of spreadsheets in Microsoft® Excel running on an Apple® Macintosh™ computer [1]. The use of a spreadsheet program allows the easy intermingling of a large number of factors in a tractable manner. This proved to be

[1] Specifically, version 1.5 of Microsoft® Excel is utilized and all of the results shown in this study are generated on an Apple® Macintosh IIci with version 6.1.4 of the Finder file and version 6.0.4 of the System file.

very advantageous in the development of the cost model. In total, eleven linked spreadsheets comprise the cost model. Appendix A presents the highest level spreadsheet for the baseline run. The highest level spreadsheet provides the interface of the cost model — it contains the alterable input parameters and provides the more relevant outputs of the model.

To facilitate discussion of the cost model, the organization of the spreadsheet of Appendix A is followed. Subsections 5.3.1 through 5.3.5 of this report correlate directly with Sections I through V of the spreadsheet and are referred to as the cost model is presented.

5.3.1 System Parameters

The first section of the spreadsheet shown in Appendix A lists the alterable input parameters of the cost model. The individual entries are discussed in the sections which follow. The general subcategories are discussed here.

The first four subcategories list the relevant parameters of the avionics suite and then the ALS from an increasingly wider prospective. The "Component Attributes" subcategory presents the relevant parameters pertaining to the components of the architectures presented in Section 5.2.3. The "Avionics System Attributes" subcategory includes the cost parameters related to the entirety of the avionics suite for each of the two architectures. Subcategory C, "Vehicle Attributes", contains the parameters associated with the vehicle which impact the cost model. The "Fleet Attributes" contains only one parameter: the number of vehicles in the fleet.

The fifth subcategory, "Operational Attributes" lists the operational parameters which impact the cost model. These are divided between two further subcategories — "On the Pad" and "During Launch/Ascent". The individual parameters are discussed in Section 5.3.4.

5.3.2 Cost of System

The cost of the avionics system calculated in Section II of the spreadsheet is the cost per vehicle to design, develop and construct a fleet of vehicles. It is calculated from

$$C_{system} = \frac{C_{dd}}{n} + (\sum C_{component}) + C_{construction} \qquad (5\text{-}1)$$

This states that the cost of the avionics system, C_{system}, is the sum of three terms. The first term is the Design and Development Cost, C_{dd}, divided by the Number of Vehicles in the Fleet, n. C_{dd} represents all the fixed costs of the system. $\sum C_{component}$ and $C_{construction}$ are the recurring costs. $\sum C_{component}$ is the sum of the costs of the individual components

and $C_{construction}$ is all of the other recurring costs. $C_{construction}$ is taken from the Construction Cost (excluding cost of parts) entry of Section I.B of the spreadsheet.

5.3.3 Cost of Launch Weight

Since the bulk of the avionics system rides with the payload all the way to orbit, it's weight subtracts directly from the ALS payload lift capability. Section III of the spreadsheet associates a cost penalty with this weight. The cost of the avionics system launch weight, C_{weight}, is obtained from

$$C_{weight} = W \bullet X_{payload} \tag{5-2}$$

where W is the weight of the avionics system (the sum of the weights of the components and the Weight of the Integration Hardware listed in the cost spreadsheet) and $X_{payload}$ is the Payload Launch Cost. The payload launch cost is the estimated cost for delivering payloads to the ALS parking orbit.

5.3.4 Cost of Unreliability

Section IV of the spreadsheet computes the costs incurred as a result of the unreliability of the avionics system. The costs due to the unreliability differ from the two previous categories in that the cost does not materialize until a probabilistic event occurs — a failure or sequence of failures which results in either a repair action being taken or a mission failure. Therefore, the cost associated with a repair action or of a failed mission must be weighted by the probability of the event occurring. This necessitates modeling the reliability of the avionics system.

The cost of the unreliability of the avionics system is conceptually expressed in the equation

$$C_{unreliability} = P_r C_r + P_f C_f \tag{5-3}$$

where

$C_{unreliability}$ = Cost of unreliability

P_r = Probability that a repair action is taken

C_r = Cost associated with repair action

P_f = Probability that a mission fails

C_f = Cost associated with a failed mission.

The first term of Equation (5-3) accounts for the cost of unreliability while on the launch pad. The second term is the cost of unreliability during launch. Markov models are

utilized to calculate the probabilities of P_r and P_f. The costs of C_r and C_f are calculated directly from the cost entries in Section I of the spreadsheet.

The models utilized to calculate the reliability of Architectures 1 and 2 are based on a single mission scenario. The mission begins when the vehicle is rolled out to the launch pad. It then sits on the pad for a specified number of days. If no repair action is taken, the engines are ignited and the vehicle launched. The vehicle would then ascend from the launch pad to the orbit the payload is to be delivered to. The mission ends successfully when the payload is deposited into orbit.

Three outcomes are possible from this mission scenario. The first is the successful completion of the mission. The second is that a decision is made to interrupt the countdown to perform a repair. The third is that a mission failure occurs and the vehicle and payload are lost. Note that this is slightly different from the actual mission whereby the mission continues after a repair action is taken. However, if in the actual mission the probability of two or more repair actions taking place is much smaller than the probability of only one occurring, then the second and third outcomes of the mission scenario can be used as calculations of P_r and P_f, respectively.

The cost of unreliability is dependent on the repair strategy for detected failures within the avionics system. While the vehicle sits on the launch pad, following each detected failure of a component, a decision must be made as to whether or not to interrupt the countdown and repair the failure (or failures). Even for the simple architectures defined in this study many strategies are possible. The most viable ones are defined here. For Architecture 1, the analyzed repair strategies are:

1. Repair when the first failure is detected. All components must be declared operational for the launch to occur.

2. Delay repair until either the failure of the Voting Actuator or both Control Chains is detected. (A Channel and its dedicated Sensor are referred to as a Control Chain. A failure of either the Channel or its dedicated Sensor results in the failure of the Control Chain.) The vehicle is permitted to be launched with the detected failure of one Control Chain.

3. No repairs are performed. The vehicle is launched independent of the component failures detected while it resides on the pad.

For Architecture 2, the analyzed repair strategies are:

1. Repair when the first failure is detected.

2. Delay repair until either the failure of 2 Sensors or 1 Channel or 2 Actuators is detected. The vehicle is permitted to be launched with only 2 Sensors and/or 2

Actuators detected as operational. All 3 Channels must still be detected as operational to allow the launch.

3. Delay repair until either the failure of 2 Sensors or 2 Channels or 2 Actuators is detected.

4. Delay repair until either the failure of 3 Sensors or 2 Channels or 3 Actuators is detected.

5. No repairs are performed.

The calculation of the two terms of Equation (5-3) are respectively performed in Sections IV.A and IV.B of the spreadsheet and discussed in the two sections which follow. Section IV.C of the spreadsheet sums these two terms and reports the total cost due to the unreliability of the avionics system.

5.3.4.1 On the Launch Pad

Section IV.A of the spreadsheet computes the cost of unreliability on the pad for the two architectures analyzed. The first subsection contains the "Intermediate Calculations". The "Intermediate Calculations" are the adjustments made to the failure rates and the calculation of the cost of interrupting the countdown. The second subsection calculates bounds for the cost of unreliability on the pad.

The base failure rates are adjusted to account for the environment of the launch pad and the fact that the avionics system may not be powered up continually while the vehicle sits on the pad. The expression

$$\lambda_{effective} = \pi_E \left[\frac{\left(\frac{t_{on}}{t_{off}}\right) + \frac{1}{\left(\frac{\lambda_{on}}{\lambda_{off}}\right)}}{\left(\frac{t_{on}}{t_{off}}\right) + 1} \right] \lambda_{base} \qquad (5\text{-}4)$$

is used to adjust each respective base failure rate λ_{base} to provide the effective failure rate $\lambda_{effective}$. All of the parameters are taken directly from the Section I.E.1 of the spreadsheet. π_E is the Pad Environmental Factor. (t_{on}/t_{off}) and $(\lambda_{on}/\lambda_{off})$ are the Ratio of Hours On/Off and On/Off Failure Rate Scale Factor, respectively.

The expression to calculate the cost of interrupting the countdown is taken from Reference 3. The Cost of Interrupting the Countdown, $C_{interrupt}$, is calculated from

$$C_{interrupt} = \frac{(T_d + T_c)(X - 1)Y}{(S - 1)} \qquad (5\text{-}5)$$

For the ALS, it is assumed that repairs are not made on the pad. If repairs are needed, the entire vehicle is brought back to the vehicle assembly area for rework. The Repair Time, T_d, reflects the time to move the entire vehicle from the pad to the assembly building, where the failed unit is replaced, and back to the pad. The total lost time is computed as this Repair Time plus the time which had already been invested in the countdown when the failure occurred, T_c. Once a vehicle comes back for repair, it is assumed that surge mode is activated until the original launch schedule is reestablished. The Surge Work Rate, S, is a scale factor which represents the increase in the nominal work rate during surge mode. The Overtime Cost, X, is also a scale factor which represents the increase in the nominal Operating Cost, Y, during surge mode.

The cost associated with a repair action is the Cost of Interrupting the Countdown plus the possible cost of replacing the defective components. Therefore, this cost is dependent on the failure mode which led to it. So, the first term of Equation (5-3), the cost of unreliability while on the launch pad (C_{pad}), is more accurately expressed as

$$C_{pad} = \sum_i P_{r,i} C_{r,i} \tag{5-6}$$

where $P_{r,i}$ is the probability of a specific repair action occurring and $C_{r,i}$ is the cost of this repair action. Sections 5.3.4.1.1 and 5.3.4.1.2 present the Markov models utilized to calculate the probability of occurrence of the repair actions and their associated costs for Architectures 1 and 2, respectively.

5.3.4.1.1 Architecture 1

Figure 5-3 presents the Markov model for Architecture 1 for the pad phase of the mission. Table 1 describes the states defined by Figure 5-3. In Figure 3, λ_s, λ_c and λ_a are the effective failure rates of each Sensor, Channel and Actuator, respectively. σ is the VHMS Detection and Interruption Rate and c_σ is the VHMS Self-Test Coverage Probability.

The Markov model shown in Figure 5-3 is utilized to generate the cost of unreliability on the pad for all of the defined repairs strategies for Architecture 1. Enough resolution is incorporated into the model to also make it compatible with the Markov model used for the second phase of the mission. State 1 represents the initial state of the avionics system — no failures of its components. States 2 and 3 individually represent the occurrence of a first fault in the avionics system. In the case of a fault in one of the Control Chains (State 2), two ensuing events can happen. The VHMS could detect the failure (State 4) or a near coincidental fault of a second component occurs before this detection can take place (State 5). Note that it is conservatively assumed that the VHMS is unable to interpret coincidental faults and, at worst, detects neither fault — resulting in an uncovered failure of a component. The transitions emanating from State 3 are analogous to those from

State 4, with the exception that a detection probability, c_σ, is now associated with the detection of the Actuator fault. This is because the Actuator is a simplex component and it is assumed the probability of detecting its failure is less than one.

State 4 represents the failure configuration with one Control Chain providing incorrect commands to the Voting Actuator. Architecture 1 continues to operate properly in this failure mode. However, the occurrence of another fault will fail the avionics system. The reason for distinguishing between States 7, 8, 9 and 10 is to discriminate between uncovered and covered failures of the avionics system.

For the first defined repair strategy for Architecture 1 — repair when the first failure is detected — States 4, 6, 7, 8, 9 and 10 are the states which represent a repair action has been taken. States 1 and 2 are modes from which a successful launch would occur. Being in States 3 or 5 would result in a system loss when the vehicle is launched. For this repair strategy, the cost of unreliability on the pad is

$$C_{pad} = P_6 C_6 + P_{4,7,8,9,10} C_{4,7,8,9,10} \tag{5-7}$$

where P_6 and $P_{4,7,8,9,10}$ are the probabilities of being in State 6 and States 4, 7, 8, 9 or 10, respectively. C_6 and $C_{4,7,8,9,10}$ are the respective costs to repair from these states. A distinction is made between State 6 and the States 4, 7, 8, 9 and 10 because different failure modes triggered the repair actions and greater modeling accuracy can be attained by distinguishing between the costs of these repair actions.

State	Description
1	No Failures
2	Fault in one Control Chain
3	Fault in Actuator
4	Detected fault in one Control Chain
5	Undetected failure of one or more components
6	Detected failure of Actuator
7	Fault in one of two remaining Control Chains
8	Fault in Actuator following previous detected failure
9	Detected failure preventing safe launch
10	Undetected failure of one or more components

Table 5-1. States of Architecture 1 Pad Markov Model

The "First Detected Failure" entry of Section II.A.2.a is calculated directly from Equation (5-7). P_6 and $P_{4,7,8,9,10}$ are obtained by solving the Markov model for Architecture 1. The bounds result because the defective components may or may not need to be replaced. So, C_6 and $C_{4,7,8,9,10}$ range between $C_{interrupt}$ and $C_{interrupt}$ plus the maximum number of components which may have to be replaced. The entries for the other two repair strategies are calculated analogously.

5.3.4.1.2 Architecture 2

The Pad Unreliability Cost for Architecture 2 is calculated by the same methodology used for Architecture 1. However, Architecture 2 is divided up into three independent Markov models. One model is constructed for the Sensors, one for the FTP and one for the Actuators. The three models are then combined to produce the probabilities of interest.

Figure 5-4 and Table 5-2 present the Markov model for FTP. Table 5-3 defines the symbols in Table 5-2. This model is based on Markov models used for previous reliability studies of the FTP.

Note that a unique structure is recognizable in the FTP Markov model. Starting at State 1, the state of the FTP with no failures of its components, a fault can occur in one of the three components (Processor, Memory and Interstage) which are distinguished in each of its three channels. These faults can be either permanent or transient and are separated among States 2 through 7. From these states, one of two events can occur. The first event is the FDIR processes could detect, isolate and reconfigure by eliminating the appropriate channel if it is a permanent fault (State 8) or resynchronize the affected channel if it is a transient fault (return to State 1). The second event is that a second, near coincident fault can occur before the FDIR processes are completed and the result is conservatively assumed to be an uncovered failure (State 9).

From State 8, as in State 1, a fault can occur in one of the three components in each of the remaining channels. These are divided among States 10, 12, 14, 16, 18 and 20. Additional resolution is incorporated at this failure level of the model to separate between the detection process and the other processes of FDIR. Since the coverage probability at this failure level is most likely to be dominated by the isolation and reconfiguration processes, separating the detection processes for the respective faults yields a more accurate model. The correct detection of these faults is modeled, respectively, as State 11, 13, 15, 17, 19 and 21. Note that there is a distinction between a missed detection (State 24) and an incorrect isolation or reconfiguration (State 23). State 22 represents the FTP correctly operating with a single channel. The occurrence of the next fault fails the FTP. However, if the fault is covered (State 25) the launch may be interrupted, whereas, if it is uncovered (State 26) the launch may result in a mission failure.

Given the structure of the FTP Markov model and the fact that the rates of the FDIR processes are much faster than the component failure rates, it is possible to reduce the model to the form shown in Figure 5-5. Appendix B discusses the reduction techniques utilized to generate the time invariant transition rates of Figure 5-5. The operational states of the FTP Markov model are reduced to States 1^R (all three channels operational), 2^R (only two of the three channels operational) and 3^R (only one of the three channels operational). States 9, 23, 24, 25, and 26 are these same states in Figure 5-4.

The Markov models for the Sensors and the Actuators are of the same form as the FTP Markov model, but are smaller in size. A single, permanent failure rate is associated with each Sensor (or Actuator). Their Markov models would appear as Figure 5-4 minus States 2 through 6 and 10 through 19 with the parameters for the Sensors (or Actuators) replacing those for the Interstages. These models are then reduced to the form shown in Figure 5-5 with the techniques of Appendix B.

The three independent Markov models are combined to generate the entries of Section IV.A.2.b of the spreadsheet shown in Appendix A. For each of the defined repair strategies, the probabilities of the appropriate states of each of the three models are combined to calculate the bounds of the probabilities of the repair actions and weighted by the respective costs of these actions as in Equation (5-6).

Two points should be noted with regard to the input System Parameters spreadsheet. None of the repair strategies invoke a repair action when all three Channels are detected as failed. So, consequently, there are no Simplex Coverage Probability for any of the Channel components. Also, the spreadsheet only allows a single FDIR Rate (Recovery Rate) to be input for each type of component failure. The rates for the detection and isolation and recovery processes are conservatively assumed to be the FDIR Rate (Recovery Rate).

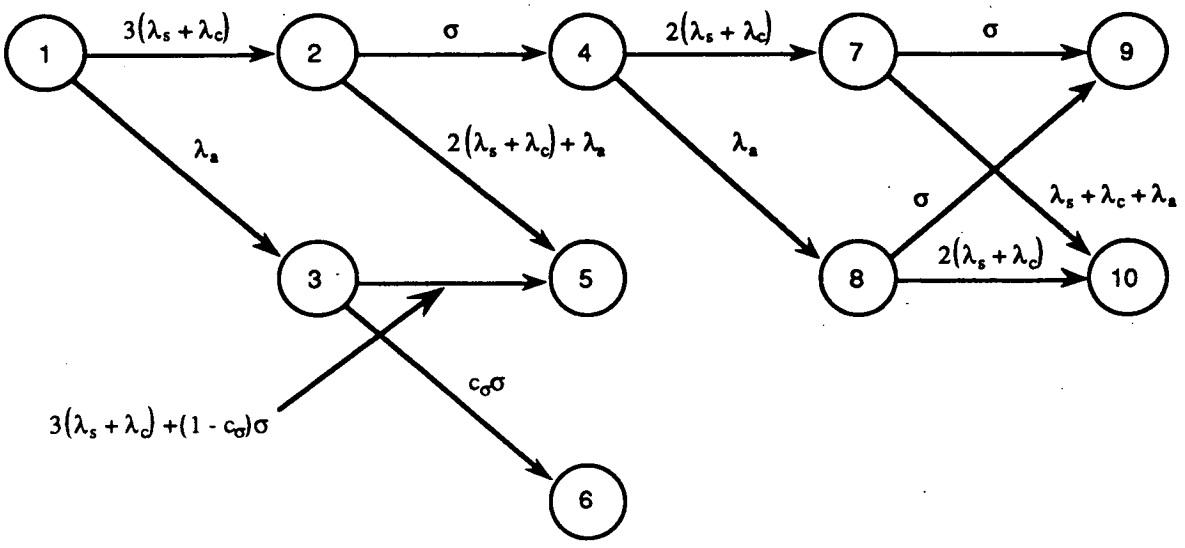

Figure 5-3. Pad Markov Model for Architecture 1

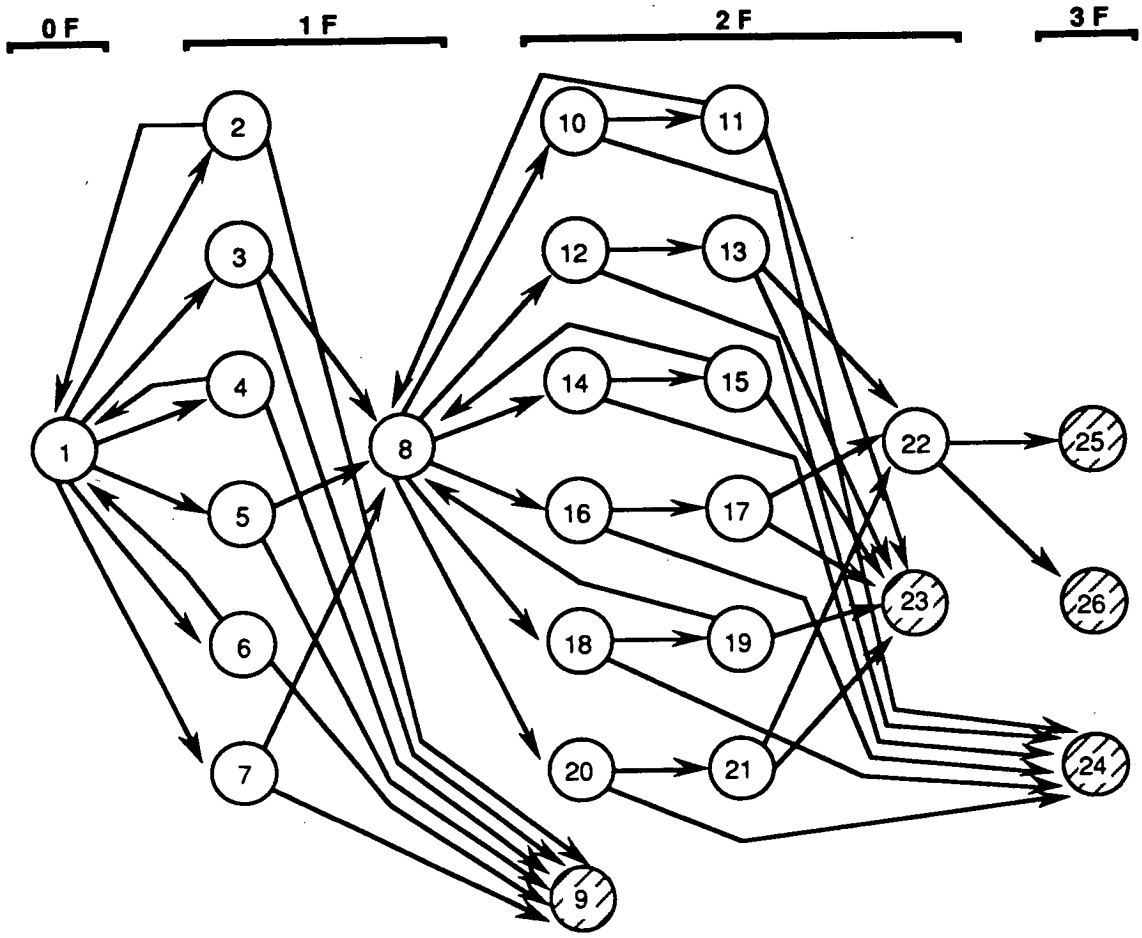

Figure 5-4. Triplex FTP Markov Model

From State → To State	Transition Rate
$1 \rightarrow 2$	$3\lambda_{pt}$
$1 \rightarrow 3$	$3\lambda_{pp}$
$1 \rightarrow 4$	$3\lambda_{mt}$
$1 \rightarrow 5$	$3\lambda_{mp}$
$1 \rightarrow 6$	$3\lambda_{it}$
$1 \rightarrow 7$	$3\lambda_{ip}$
$2 \rightarrow 1$	ρ_{pt}
$2 \rightarrow 9$	$2(\lambda_{pt} + \lambda_{pp}) + 3(\lambda_{mt} + \lambda_{mp}) + 3(\lambda_{it} + \lambda_{ip})$
$3 \rightarrow 8$	ρ_{pp}
$3 \rightarrow 9$	$2(\lambda_{pt} + \lambda_{pp}) + 3(\lambda_{mt} + \lambda_{mp}) + 3(\lambda_{it} + \lambda_{ip})$
$4 \rightarrow 1$	ρ_{mt}
$4 \rightarrow 9$	$3(\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + 3(\lambda_{it} + \lambda_{ip})$
$5 \rightarrow 8$	ρ_{mp}
$5 \rightarrow 9$	$3(\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + 3(\lambda_{it} + \lambda_{ip})$
$6 \rightarrow 1$	ρ_{it}
$6 \rightarrow 9$	$3(\lambda_{pt} + \lambda_{pp}) + 3(\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
$7 \rightarrow 8$	ρ_{ip}
$7 \rightarrow 9$	$3(\lambda_{pt} + \lambda_{pp}) + 3(\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
$8 \rightarrow 10$	$2\lambda_{pt}$
$8 \rightarrow 12$	$2\lambda_{pp}$
$8 \rightarrow 14$	$2\lambda_{mt}$
$8 \rightarrow 16$	$2\lambda_{mp}$
$8 \rightarrow 18$	$2\lambda_{it}$
$8 \rightarrow 20$	$2\lambda_{ip}$
$10 \rightarrow 11$	δ_{pt}
$10 \rightarrow 24$	$(\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
$11 \rightarrow 8$	$c_{pd}\gamma_{pt}$
$11 \rightarrow 23$	$(1 - c_{pd})\gamma_{pt} + (\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
$12 \rightarrow 13$	δ_{pp}
$12 \rightarrow 24$	$(\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
$13 \rightarrow 22$	$c_{pd}\gamma_{pp}$

Table 5-2. Transition Rates for FTP Markov Model

From State → To State	Transition Rate
13 → 23	$(1 - c_{pd})\gamma_{pp} + (\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
14 → 15	δ_{mt}
14 → 24	$2(\lambda_{pt} + \lambda_{pp}) + (\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
15 → 8	$c_{md}\gamma_{mt}$
15 → 23	$(1 - c_{md})\gamma_{mt} + 2(\lambda_{pt} + \lambda_{pp}) + (\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
16 → 17	δ_{mp}
16 → 24	$2(\lambda_{pt} + \lambda_{pp}) + (\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
17 → 22	$c_{md}\gamma_{mp}$
17 → 23	$(1 - c_{md})\gamma_{mp} + 2(\lambda_{pt} + \lambda_{pp}) + (\lambda_{mt} + \lambda_{mp}) + 2(\lambda_{it} + \lambda_{ip})$
18 → 19	δ_{it}
18 → 24	$2(\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + (\lambda_{it} + \lambda_{ip})$
19 → 8	$c_{id}\gamma_{it}$
19 → 23	$(1 - c_{id})\gamma_{it} + 2(\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + (\lambda_{it} + \lambda_{ip})$
20 → 21	δ_{ip}
20 → 24	$2(\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + (\lambda_{it} + \lambda_{ip})$
21 → 22	$c_{id}\gamma_{ip}$
21 → 23	$(1 - c_{id})\gamma_{ip} + 2(\lambda_{pt} + \lambda_{pp}) + 2(\lambda_{mt} + \lambda_{mp}) + (\lambda_{it} + \lambda_{ip})$
22 → 25	$c_{ps}(\lambda_{pt} + \lambda_{pp}) + c_{ms}(\lambda_{mt} + \lambda_{mp}) + c_{is}(\lambda_{it} + \lambda_{ip})$
22 → 26	$(1 - c_{ps})(\lambda_{pt} + \lambda_{pp}) + (1 - c_{ms})(\lambda_{mt} + \lambda_{mp}) + (1 - c_{is})(\lambda_{it} + \lambda_{ip})$

Table 5-2. Transition Rates for FTP Markov Model (Cont.)

Symbol	Description
λ_{pt}	Transient Processor failure rate
λ_{pp}	Permanent Processor failure rate
λ_{mt}	Transient Memory failure rate
λ_{mp}	Permanent Memory failure rate
λ_{it}	Transient Interstage failure rate
λ_{ip}	Permanent Interstage failure rate
ρ_{pt}	Transient Processor recovery rate
ρ_{pp}	Permanent Processor recovery rate
ρ_{mt}	Transient Memory recovery rate
ρ_{mp}	Permanent Memory recovery rate
ρ_{it}	Transient Interstage recovery rate
ρ_{ip}	Permanent Interstage recovery rate
δ_{pt}	Transient Processor detection rate
δ_{pp}	Permanent Processor detection rate
δ_{mt}	Transient Memory detection rate
δ_{mp}	Permanent Memory detection rate
δ_{it}	Transient Interstage detection rate
δ_{ip}	Permanent Interstage detection rate
γ_{pt}	Transient Processor isolation and reconfiguration rate
γ_{pp}	Permanent Processor isolation and reconfiguration rate
γ_{mt}	Transient Memory isolation and reconfiguration rate
γ_{mp}	Permanent Memory isolation and reconfiguration rate
γ_{it}	Transient Interstage isolation and reconfiguration rate
γ_{ip}	Permanent Interstage isolation and reconfiguration rate
c_{pd}	Duplex coverage probability of Processor failure
c_{md}	Duplex coverage probability of Memory failure
c_{id}	Duplex coverage probability of Interstage failure
c_{ps}	Simplex coverage probability of Processor failure
c_{ms}	Simplex coverage probability of Memory failure
c_{is}	Simplex coverage probability of Interstage failure

Table 5-3. Symbol Definitions for FTP Markov Model

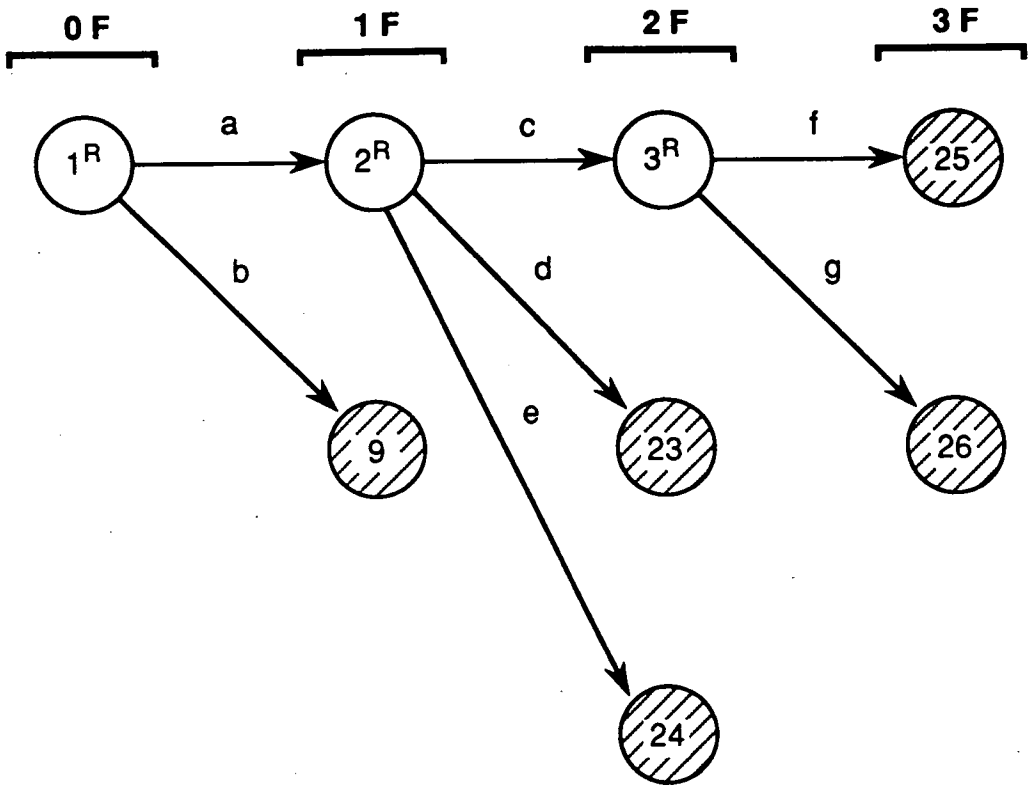

Figure 5-5. Reduced Triplex FTP Markov Model

5.3.4.2 During Launch and Flight

Section IV.B of the spreadsheet computes the cost of unreliability from avionics system failures during the launch and flight of the vehicle for each of the investigated repair strategies. The first subsection contains the "Intermediate Calculations". The "Intermediate Calculations" are the cost associated with a mission failure and the adjustments made to the failure rates. The second subsection of the spreadsheet calculates the bounds for the cost of unreliability during launch and flight.

The costs associated with the failure of the avionics system during ascent are discussed in Section 5.2.2. For the purposes of this study, this cost (C_f in Equation (5-3)) is limited to the cost of the avionics system, the launch vehicle, payloads and operational costs associated with the mission. It is calculated in the spreadsheet directly from

$$C_f = C_{system} + W_{payload} \cdot X_{payload} + C_{payload} \tag{5-8}$$

where C_{system} is the cost of the avionics system itself (calculated in Equation (5-1)); $W_{payload}$ is the Total Payload Weight; $X_{payload}$ is Payload Launch Cost; $C_{payload}$ is the Payload Value.

The base failure rates are adjusted to account for the environment of the launch/ascent, as was done for the on pad phase of the mission. However, for this phase of the mission the avionics system is assumed to be powered up for its entirety. So, the relation used to adjust the base failure rates becomes

$$\lambda_{effective} = \pi_E \lambda_{base} \tag{5-9}$$

where π_E is now the Launch Environmental Factor.

Sections 5.3.4.2.1 and 5.3.4.2.2 discuss the Markov models utilized to calculate the probability that a mission failure occurs in order to generate the cost of unreliability during launch.

5.3.4.2.1 Architecture 1

Figure 5-6 shows the Markov model for the launch/flight phase of the mission. Table 4 describes the states defined by Figure 5-6. During launch it is no longer possible to interrupt the countdown to repair failed equipment within the avionics system. Therefore, the VHMS no longer has an impact on the unreliability. Hence, the model depicted in Figure 5-3 collapses into the first 4 states of the Launch/Flight Markov model. State 5 accounts for the probability that a repair action is undertaken during the on pad phase of the mission.

The initial probabilities of the states of the Launch/Flight Markov model are derived from the probabilities of the states of the Pad Markov model at the end of the Time on Pad. The distribution is dependent on the repair strategy. For example, for the strategy which is to repair on the first detected failure the initial probabilities of the Launch/Flight Markov model are as listed in Table 5-5. The initial probabilities for the other two repair strategies are analogously made.

Section IV.B.2.a of the spreadsheet calculates the Launch/Flight unreliability cost of Architecture 1. The Launch/Flight Markov model is solved using the initial conditions for each of the repair strategies. The sum of the probabilities in States 3 and 4 at the end of the Flight Time is P_f. C_f has been previously calculated. Therefore, Launch/Flight unreliability cost is calculated according to the second term of Equation (5-3).

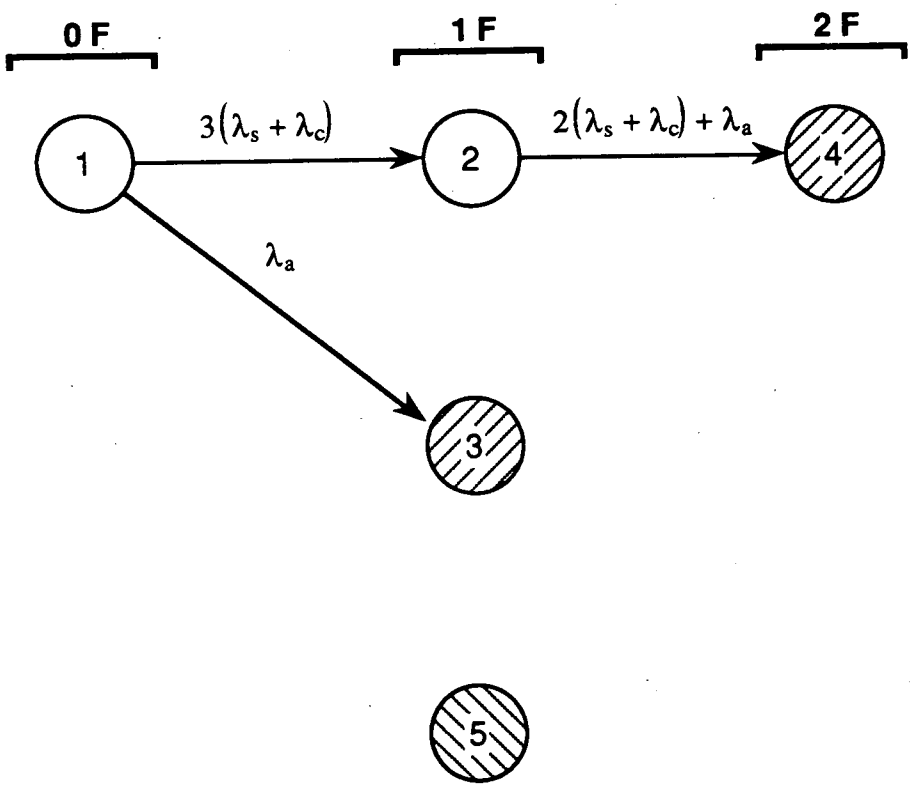

Figure 5-6. Launch/Flight Markov Model for Architecture 1

State	Description
1	No failures
2	Fault in one Control Chain
3	Fault in Actuator; System failure
4	System failure
5	Countdown interrupted to perform a repair

Table 5-4. States of Architecture 1 Launch Markov Model

State	Initial Probability is Probability at End of Time On Pad of Listed States of Pad Markov Model
1	1
2	2
3	3
4	5
5	4, 6, 7, 8, 9, 10

Table 5-5. Initial Probabilities of Launch/Flight Markov Model for First Detected Failure Repair Strategy of Architecture 1

5.3.4.2.2 Architecture 2

The Launch Unreliability Cost for Architecture 2 is calculated according to the same methodology used for Architecture 1. The three Markov models used to calculate the Pad Unreliability Cost for Architecture 2 are modified slightly to model the cost of unreliability during the launch/flight phase of the mission. An additional state, analogous to State 5 of

the Launch/Flight Markov model for Architecture 1, is added to each of the three Markov models to account for the probability that a repair action is undertaken during the on pad phase of the mission. It is also assumed that during the launch phase of the mission channels which are removed by the FDIR processes because of a transient fault are not brought back into service. Therefore, in the FTP Launch/Flight model transient faults are modeled as permanent faults.

As is done for Architecture 1, the initial probabilities of the Launch/Flight Models are derived from the probabilities of the states of the Pad Markov models at the end of the Time on Pad. This redistribution of the probability is dependent on the repair strategy and is done analogous to the example presented in Section 5.3.4.2.1.

Section IV.B.2.b of the spreadsheet calculates the Launch/Flight unreliability cost of Architecture 2. The three Launch/Flight Markov models are solved using the initial conditions for each of the repair strategies and combined to generated P_f. The entries of Section IV.B.2.b of the spreadsheet evaluate P_f for the Flight Time and multiply it with the previously calculated C_f to produce the Launch Unreliability Cost.

5.3.5 Total Cost

Section V of the spreadsheet adds the results of Sections II, III and IV and presents the result for each of the repair strategies for the two architectures. This is the total cost for the respective avionics system for the costs which are accounted for in this study.

5.4 Results

The cost model described here is used to predict the respective costs of the two defined avionics systems — Architecture 1 and Architecture 2. The Baseline System Parameters are listed as the appropriate entries of Section I of the spreadsheet shown in Appendix A. Sections II through V of the spreadsheet show the intermediate and final results using the Baseline System Parameters.

The Baseline System Parameters of Section I are estimates based on currently available technology and the current operational practices for commercial launch vehicles. The Component Attributes are chosen under the assumption that Class B parts are utilized. The Component Attributes and the Avionics System Attributes inputs for Architecture 2 are selected based on past experience in applying the AIPS technology to similar systems. The Component Attributes and the Avionics System Attributes inputs for Architecture 1 are chosen so that they mimic those of Architecture 2. In this way, the differences in the generated results from these two implementations are less obscured by the effect of the individual parts. (In actuality, it is reasonable to use equivalent values between Architectures 1 and 2 for the Component Attributes and the Avionics System Attributes. As justification, refer to References 23 and 24 which report the experience with an

asynchronous architecture similar to Architecture 1 for the Flight Control System of the AFT/F-16 aircraft.)

The Baseline System Parameters of Subsections C, D and E are, again, reasonable estimates based on experience. The CSDL Report CSDL-R-2109 [15] and Military Handbook MIL-HDBK-217E [16] served as sources for many of the Operational Attributes of Subsection E.

Sections II through V of the spreadsheet in Appendix A present the results produced by the cost model for the Baseline System Parameters.[2] Figure 5-7 organizes the more relevant data of the spreadsheet into a more enlightening form. The stacked bar chart shows the total cost, and breaks down the sources which make up the total cost, for each of the investigated repair strategies of the respective architectures. It is obvious that the total cost varies significantly between each of the repair strategies for each architecture and between the two architectures. These variations are mainly attributed to the cost of unreliability. The cost of the avionics system itself and the cost of its launch weight are independent of the repair strategy for any particular architecture and, relatively, do not vary much between the two architectures. The cost of its launch weight for both architectures represents a small contribution to the total cost.

The two classifications for the cost of unreliability (On the Launch Pad and During Launch/Ascent) monotonically vary with regard to the repair strategy, but in different directions. In going from the less tolerant repair strategy (First Detected Failure) to the most tolerant repair strategy (No Repairs), the cost of unreliability on the launch pad decreases, whereas the cost of unreliability during launch/ascent increases. For Architecture 1, these two effects are minimized for the repair strategy which delays repair until either the failure of the Voting Actuator or both Control Chains is detected. For Architecture 2 the optimal repair strategy with regard to cost is to delay repair until either the failure of 2 Sensors or 2 Channels or 2 Actuators is detected — this results in a total cost of about $362,000. Note that for both architectures, significant reductions in cost can be obtained by permitting the vehicle to be launched with detected failures within the avionics system.

Architecture 2 shows the greatest opportunity for minimizing cost. For the Baseline System Parameters, Architecture 1 has a greater potential for a failure during launch than Architecture 2. Figure 5-7 shows that, for Architecture 1, even when the policy is not to allow launch with any detected failures, the total cost is dominated by mission failures during launch. The only component of its total cost which can be reduced by allowing launch with detected failures (Cost of Unreliability — On the Pad) is relatively small.

---

[2] Note that in the Adjusted Reliability Parameters of the FTP_Launch_Model worksheet shown in Section IV.B.1 of the spreadsheet presented in Appendix A the transient failure rates of the FTP Channel Components are indicated as 0. During the launch phase of the mission, transient failures of a Channel have the same impact as permanent failures since a Channel will not be resynchronized once it is isolated off-line. The transient failure rate is accounted for in the respective permanent failure rate of the worksheet.

Alternately, the cost of unreliability for Architecture 2 for the case which permits launch only when there are no detected failures within the avionics system is dominated by the unreliability on the pad. Therefore, potential exists for significantly reducing the total cost by allowing launch to take place with some detected failures of its components. As pointed out, the optimal repair strategy is to delay repair until either the failure of 2 Sensors or 2 Channels or 2 Actuators is detected. Note that the total cost of this repair strategy is substantially less than that of any of those defined for Architecture 1.

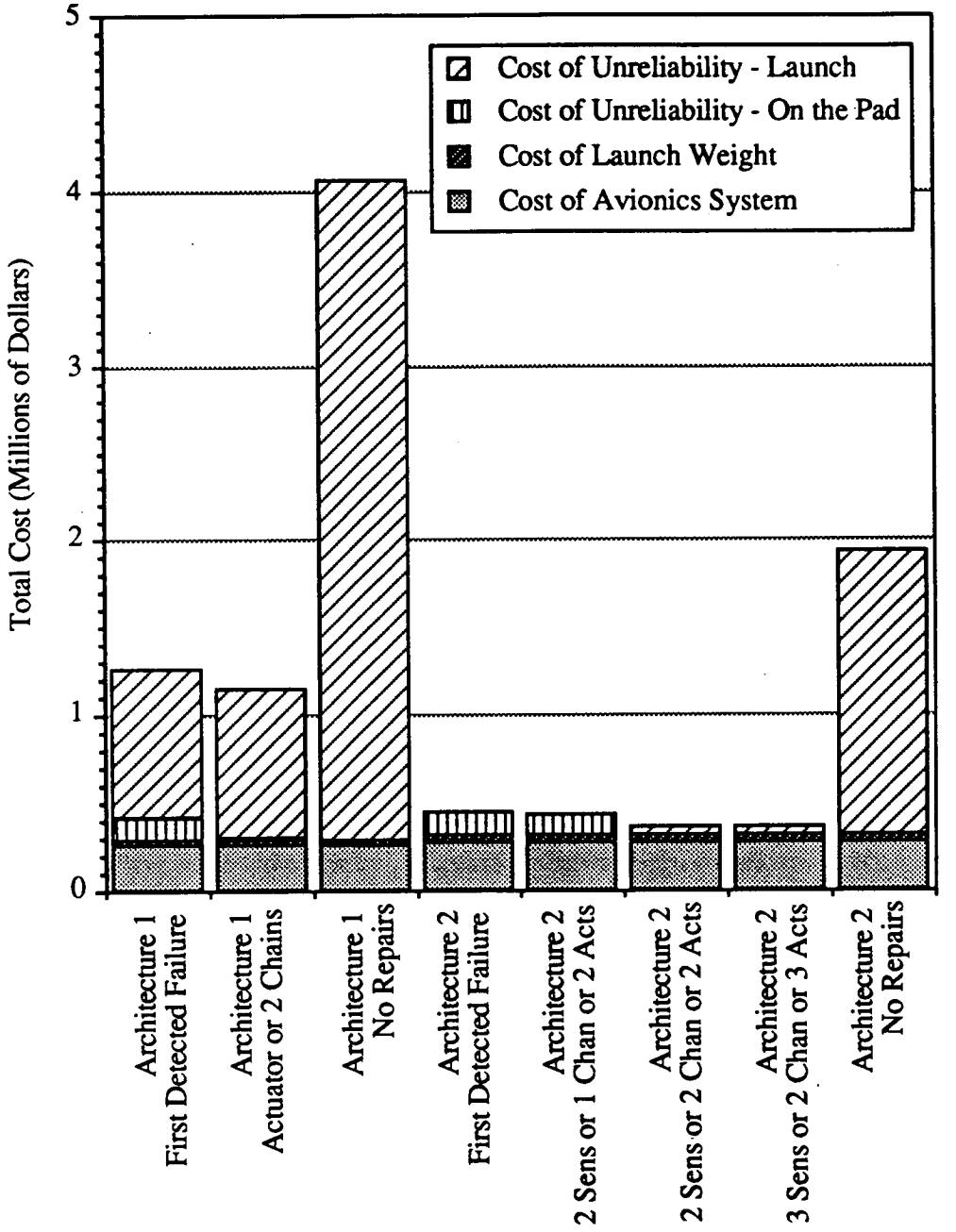

Figure 5-7. Baseline Results

Using the component quality factors listed in Table 5-6, Figure 5-8 is generated which shows the sensitivity of total cost to component quality. Table 5-6 is constructed from information in CSDL Report CSDL-R-2109 [15] and Military Handbook MIL-HDBK-217E [16]. Table 5-6 specifies the factor by which the failure rates of the components are multiplied by (Quality Factor) and their respective costs are multiplied by (Cost Factor) when a different quality level is assumed from the Baseline System Parameters. With the exception of the No Repairs repair strategy, utilizing B quality level parts shows itself to be the most economical. If the repair policy is not to repair any detected failures within the avionics system while the vehicle sits on the launch pad, using S quality level parts minimizes the total cost.

Quality Level	Quality Factor	Cost Factor
S	0.25	10.0
B	1.0	1.0
D	10.0	0.50

Table 5-6. Component Quality Factors

Figure 5-9 shows the sensitivity of the total cost for each of the repair strategies of each of the two architectures to the Payload Value. For each architecture, the sensitivity to Payload Value appears to be dependent on how lax the launch requirement is. The repair strategies which permit launch with more detected failures tend to be more sensitive to the Payload Value. However, the repair strategies of Architecture 2 (the architecture based on the AIPS FTP), excepting the No Repairs repair policy, are relatively insensitive to the Payload Value over the region shown.

Figures 5-10 and 5-11 present the sensitivity of the total cost for each of the repair strategies of each of the two architectures to the Time On Pad and Flight Time, respectively. As shown in Figure 5-10, the optimum repair strategy for each architecture appears to be least sensitive to the Time On Pad (Actuator or 2 Chains for Architecture 1; 2 Sensors or 2 Channels or 2 Actuators for Architecture 2). However, the first four repair strategies of Architecture 2 all remain less than the optimal strategy for Architecture 1 even when the Time On Pad is increased from the Baseline value of 7 days up to 30 days. As shown in Figure 5-11, increasing the Flight Time from the Baseline value of 0.167 hours amplifies the difference between the two architectures. All of the repair strategies of Architecture 2 are less sensitive to this parameter than any of the strategies of Architecture 1. The least tolerant repair strategies of Architecture 2, First Detected Failure and 2 Sensors or 1 Channel or 2 Actuators, are least sensitive to the Flight Time.

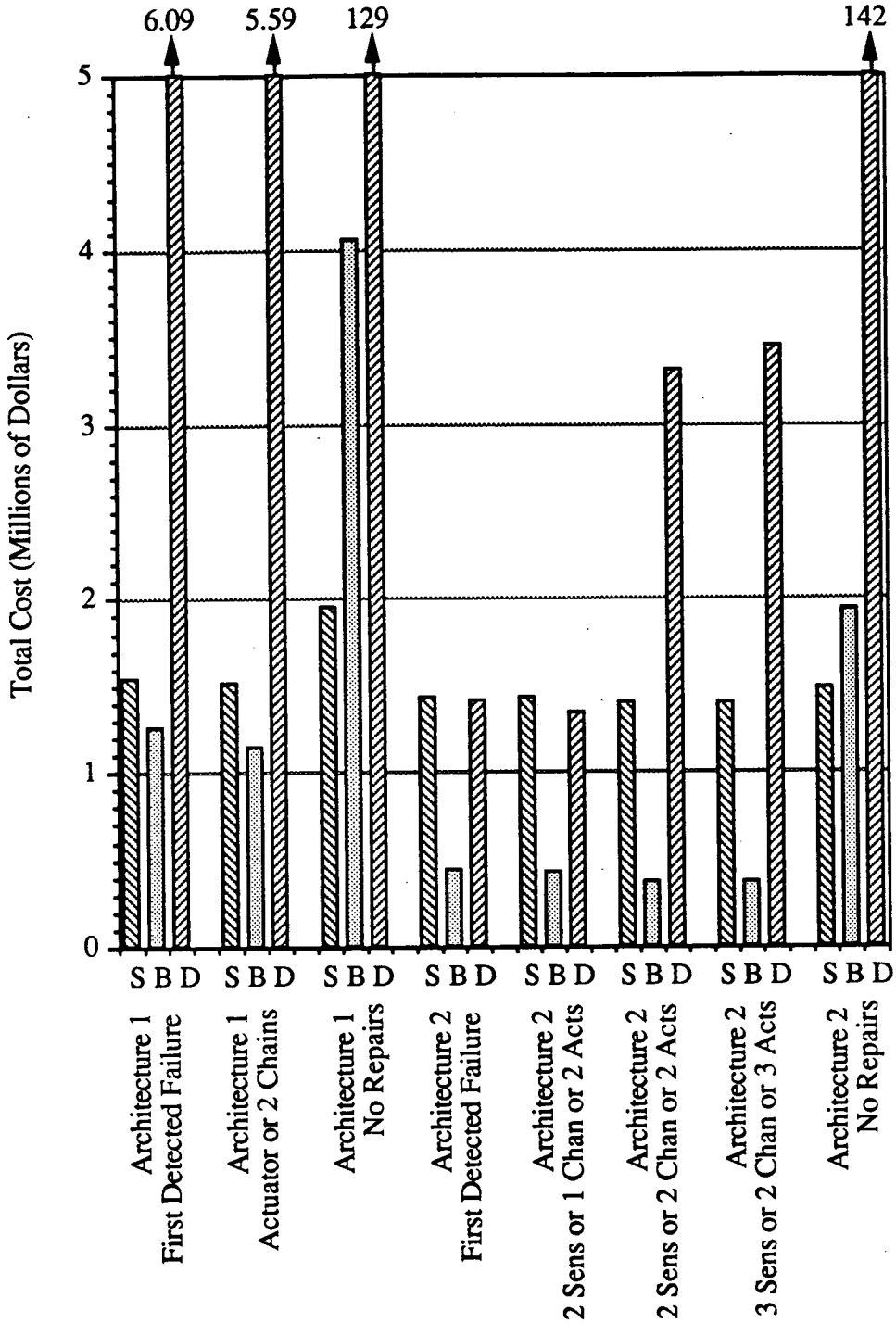

Figure 5-8. Sensitivity to Component Quality

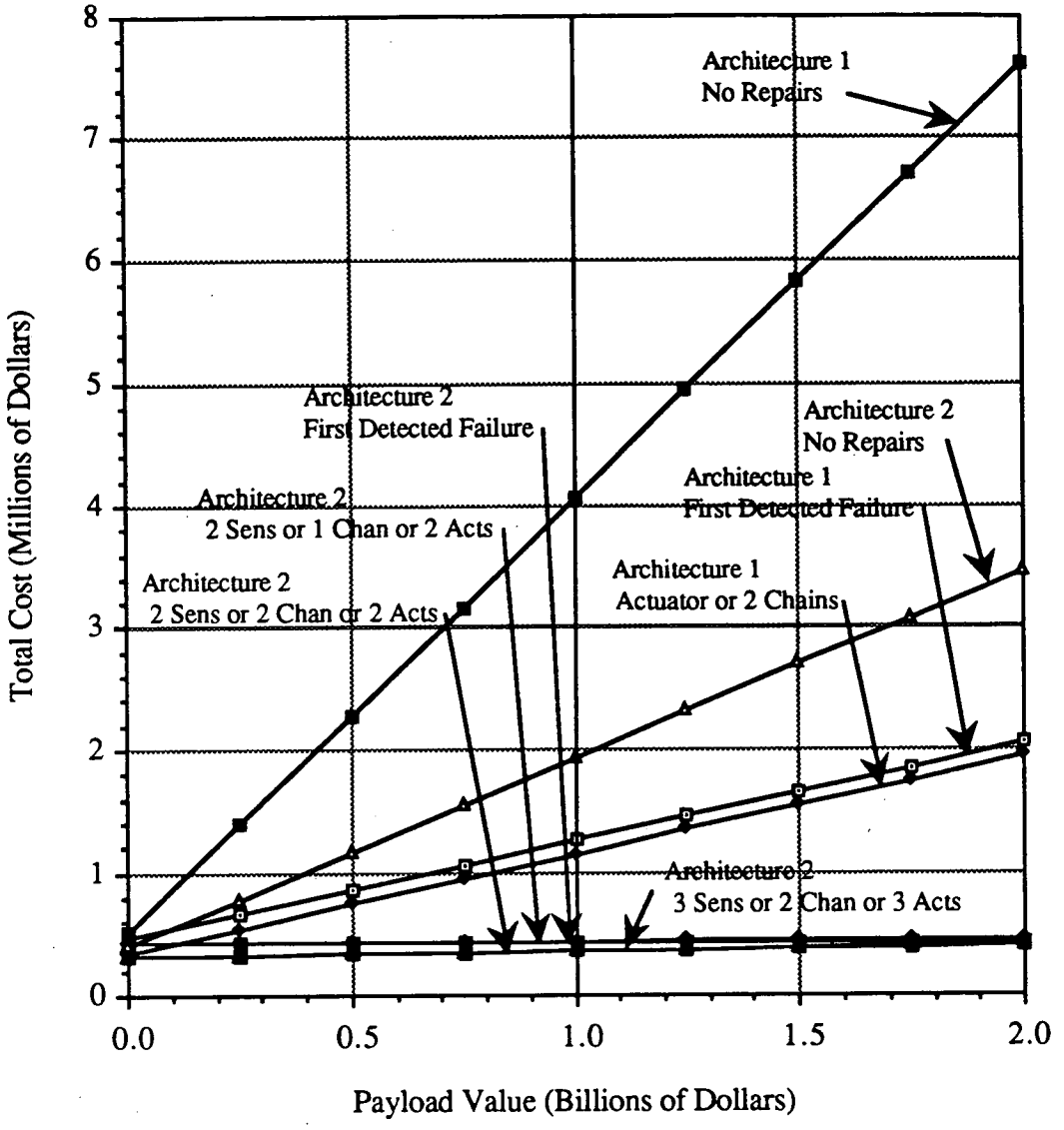

Figure 5-9. Sensitivity to Payload Value

Figure 5-12 displays the sensitivity of the total cost for each of the repair strategies of each of the two architectures to the Repair Time. For the region shown, little sensitivity to this parameter is observed. For each of the respective architectures, the strategies which allow the vehicle to be launched with more detected failures tend to be the least sensitive to Repair Time.

5.5. Conclusions and Suggestions for Future Work

A useful methodology has been demonstrated for investigating the impact of the avionics suite to the recurring cost of the ALS. The methodology evaluates the cost of the unreliability of the avionics suite and includes this along with a more traditional assessment of its cost. This allows a truer appraisal of its recurring launch cost to be made. During

design and development of the avionics system, this methodology would allow the designer to quantitatively predict the impact on cost that design decisions will have. In this way, when design freedom exists, choices can be made which will ultimately reduce launch costs. The methodology also measures the impact on cost that operational decisions can have — such as allowing the vehicle to launch with detected failures.

When the avionics system is viewed from the perspective of being mission critical, the cost of its unreliability can represent a significant portion of its total cost. Therefore, the parameters which affect its unreliability also affect its recurring launch cost. For the two architectures analyzed in this study, the impacts of two parameters are of particular interest. The first is the repair strategy which is employed for failures detected while the vehicle sits on the launch pad. Pre-empting launch on the occurrence of the first detected failure within the avionics system is not economical. Allowing the vehicle to launch with selected detected failures can potentially reduce the recurring launch costs. The second parameter of interest is the quality level of the components used. The use of Class B parts, as compared with Class S and D parts, minimizes the recurring launch cost for most of the repair strategies analyzed.

Based on the analysis performed, the AIPS FTP architecture (Architecture 2) shows itself to be potentially more economical than the more typical architecture employed for this application (Architecture 1). The more viable repair strategies of the AIPS FTP architecture show themselves to be much more cost effective than any of the strategies of Architecture 1 for the Baseline System Parameters. These repair strategies of the AIPS FTP architecture also appear to be more insensitive to a number of the major system parameters. For the AIPS FTP architecture, allowing the ALS to launch with detected failures in 1 Sensor and/or 1 Channel and/or 1 Actuator (delaying repair until failures are detected in at least 2 Sensors or 2 Channels or 2 Actuators) appears to be the most economical repair strategy.

Though the methodology presented here shows itself to be very useful in quantifying the impact of the reliability of the avionics system on the recurring launch cost of the ALS, the actual analysis performed is of limited benefit. The models of the two architectures which are the focus of the study are quite simplified in nature and lacked the complexity of an actual avionics system. Therefore, in the future it would be most useful to develop more detailed models that reflect the realistic architecture complexities and apply the techniques illustrated here to these models. More specific conclusions regarding the architecture and the most optimum repair policies could be made.

With regard to the methodology itself, some areas warrant further work. The costs associated with interrupting the countdown and a failed mission could be modeled better. The less tangible costs such as a delay in a launch schedule because of the need to do a repair or because a vehicle is lost in flight needs to be investigated further. The analysis

could also be expanded to include other critical subsystems of the ALS along with the avionics system. The methodology might also be enhanced to include reusable subsystems within the ALS and possibly compare the costs of these architectures with their expendable counterparts.

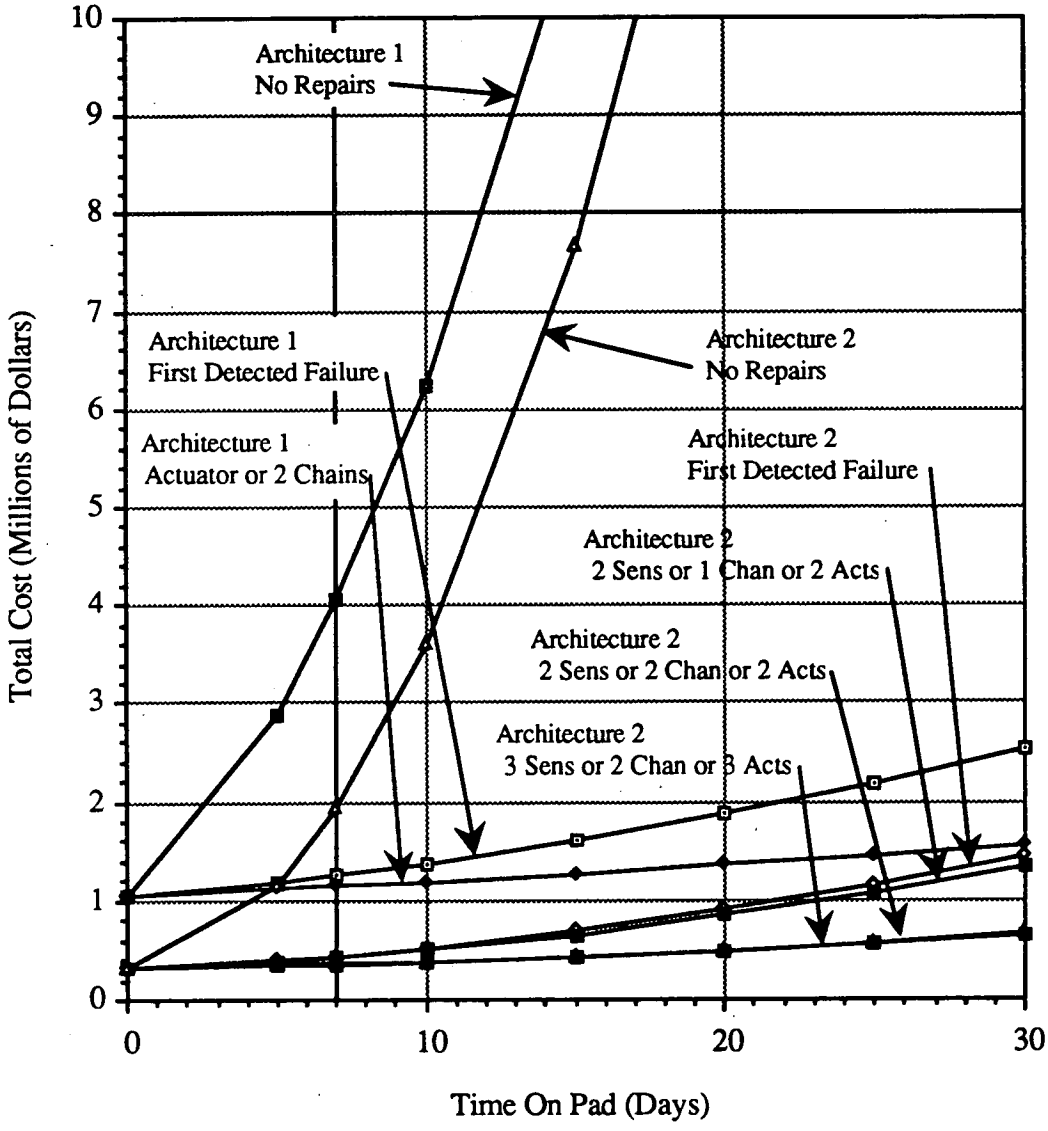

Figure 5-10. Sensitivity to Time On Pad

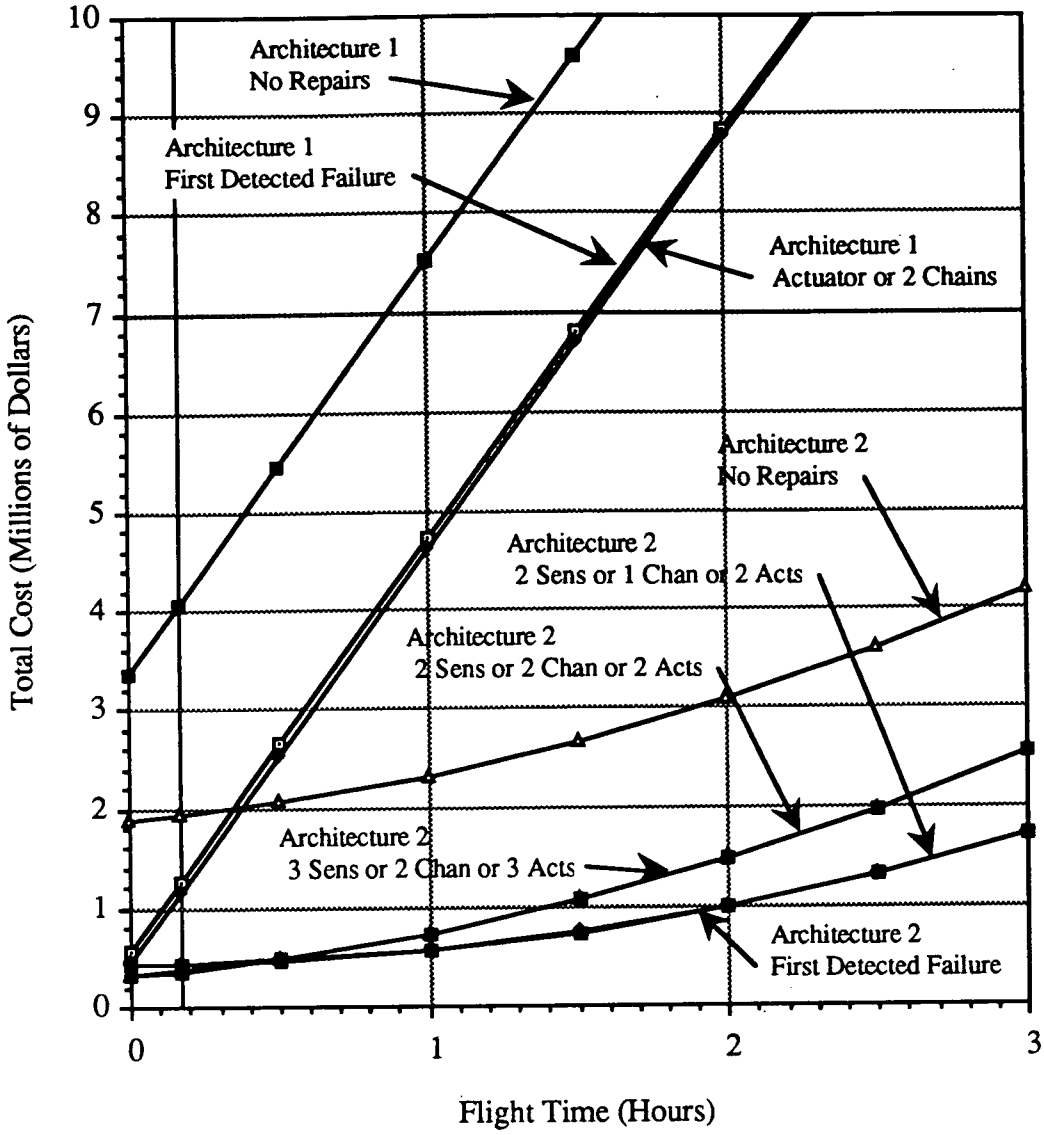

Figure 5-11. Sensitivity to Flight Time

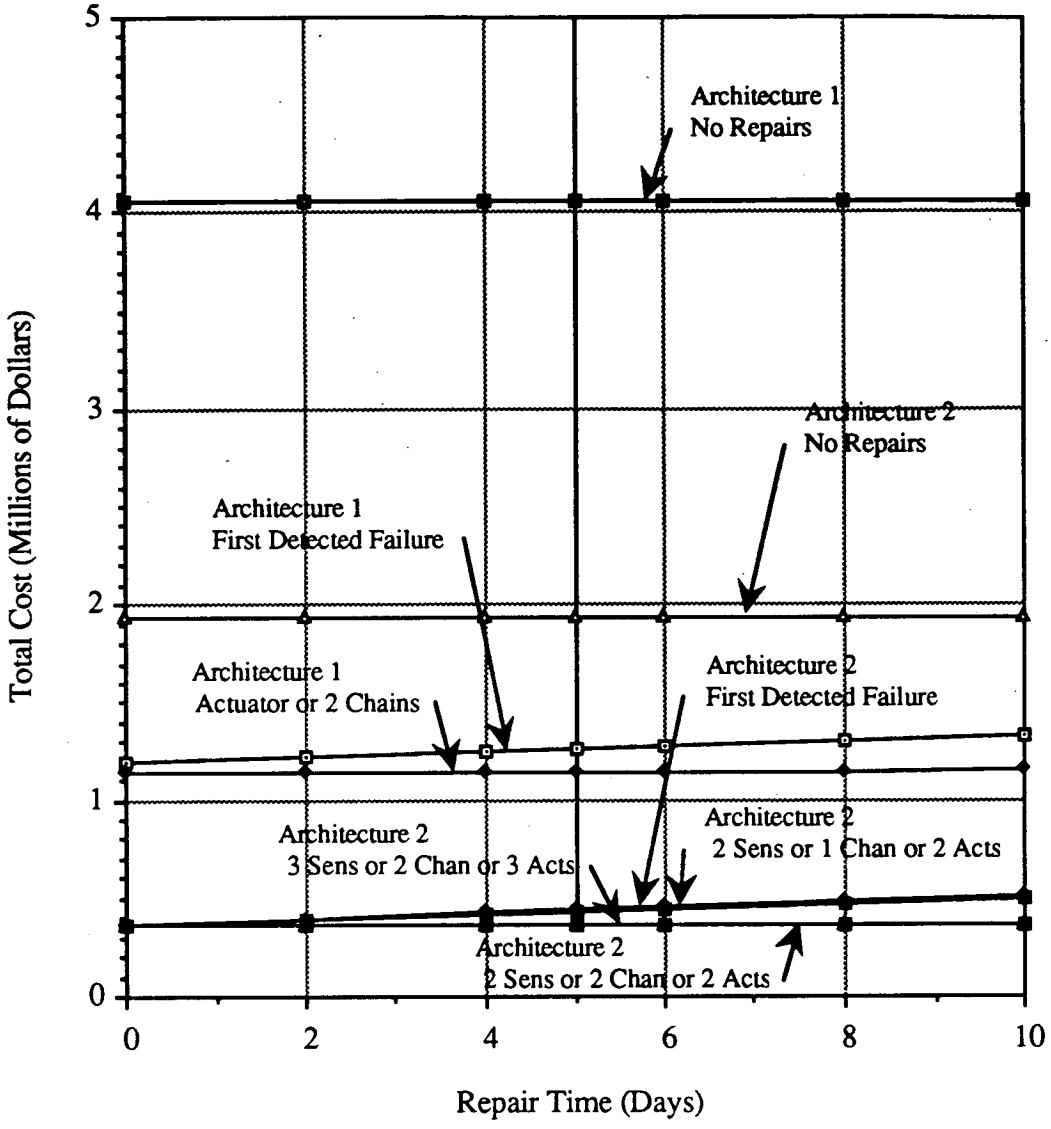

Figure 5-12. Sensitivity to Repair Time

6.0 SUMMARY AND CONCLUSIONS

The validated fault tolerant building blocks of the Advanced Information Processing System (AIPS) have been configured to meet the reliability, availability, performance and other avionics requirements of the Advanced Launch System (ALS) being developed jointly by the National Aeronautics and Space Administration and the Department of Defense to launch heavy payloads into low earth orbit. This report has described the AIPS for ALS architecture synthesis process starting with the ALS mission requirements and ending with an analysis of the candidate ALS avionics architecture. The ALS architecture synthesis process followed a new design for validation methodology that has been developed as part of the AIPS program to assure that fault tolerant computer system architectures for advanced applications meet the reliability, performance and other goals of the application.

The preliminary ALS avionics requirements were obtained by CSDL from the prime contractors via Martin Marietta. The detailed computational requirements were developed by Martin Marietta and jointly refined by MM and CSDL. The Reliability, Maintainability, and Availability (RMA) requirements for the ALS were defined for the launch pad operations, the launch or the boost phase, and for on-orbit operation. The ALS avionics availability requirement was defined to be 95%, i.e., the ALS avionics must have a fault masking capability with a probability of 0.95, at the end of one week on the launch pad. The avionics reliability requirement was defined in terms of the maximum allowable probability of mission failure or vehicle loss due to avionics failure. This failure probability should not exceed 10^{-5} for the mission duration which may vary from 10 minutes for short missions to as long as 48 hours for the longest ALS mission.

The computational requirements consisted of nine top-level functions: Central Control and Processing, Winds Ahead Determination, Vehicle Power System Management, Steering and Staging Control, Propulsion Control, Command and Telemetry Processing, Range Safety and Destruct, and Programmable Payload Interface. Aggregate throughput estimates were given for many of these functions. Each function was divided into subfunctions which was supposed to be further defined at the atomically schedulable task level. However, this process was carried out only for the Central Control and Processing function during the course of this study.

Based on the data provided by MM, the overall ALS throughput requirements were estimated to be approximately 8.8 MIPS for non-propulsion functions and 4.8 MIPS per engine for propulsion functions. The inter-function communication bandwidth requirements were estimated to be 26 Mbits/sec (prelaunch) between the non-propulsion and the propulsion functions (17 engines), the non-propulsion I/O bandwidth requirement was estimated to be 11.2 Mbits/sec, and the propulsion I/O bandwidth requirement was estimated to be 1.587 Mbits/sec (17 engines).

The flight system characteristics of the AIPS hardware and software building blocks were defined based on a survey and projection of technology expected to be available in the ALS time frame. The characteristics included the physical characteristics, performance projections, hardware implementation, module failure rates, and a packaging concept.

A quadruply redundant AIPS for ALS Fault Tolerant Processor will consist of eight SEM-E modules -- two CPU modules, one Shared Devices module, one Communicator and Interstage module, and Network Interface Sequencer modules, and two power conversion modules. The SEM-E modules are 5.88 in. x 6.68 in. x 0.6 in. and weigh approximately 1 lb. each. A quad FTP, including all enclosures, will weigh about 108 lbs, occupy 2 cubic feet and consume about 165 Watts of power. A triplex FTP will weigh about 81 lbs, occupy about 1.5 cubic feet, and consume about 124 Watts.

The raw throughput of the AIPS for ALS FTP, consisting of two 40 MHz RISC microprocessors per channel, is projected to be 30 MIPS using the DAIS mix benchmark. The useful FTP throughput available to the ALS applications tasks, after accounting for all the fault tolerance and core hardware redundancy management overheads (but not the sensor RM), is projected to be about 25 MIPS. The I/O and intercomputer network bandwidths will be 100 Mbits/sec. The internal data exchange bandwidth of the FTP will be about 64 Mbits/sec. The end-to-end communication time over the intercomputer network between functions located on different FTPs is expected to be less than 2 msecs per message. The average recovery time from a fault in the FTP is projected to be 10 msecs.

The raw hardware failure rates per hour, using Class B components, are projected to be: 6.5×10^{-5} per FTP channel and 1.0×10^{-5} per node on the launch pad; 3.3×10^{-4} per FTP channel and 1.1×10^{-4} per node during the boost phase; $2.3 * 10^{-5}$ per FTP channel and $7.6 * 10^{-6}$ per node in orbit. These are the permanent failure rates. Transients are assumed to occur at 10 times the permanent failure rates.

Using the building block performance and reliability projections, a preliminary AIPS-based fault tolerant computer system architecture was configured to meet the ALS avionics requirements. A single quadruply redundant AIPS Fault Tolerant Processor, the core FTP, will perform all the non-propulsion functions required in the ALS. Additionally, there will be a propulsion control FTP dedicated to each engine. The core FTP will access the guidance, control, navigation and other sensors and actuators on one redundant I/O network. Each of the engine control FTPs will access engine sensors and actuators on a dedicated I/O network. The core FTP and all of the engine control FTPs will be connected by a fault-masking triply redundant intercomputer network.

For short ALS missions, lasting an hour or less, it will not be necessary to reconfigure the FTPs or the I/O and IC networks. Redundant hardware would provide

sufficient fault masking capability to meet the ALS reliability requirement. However, for longer ALS missions, lasting 1 hour to 48 hours, it will be necessary to re-integrate FTP channels affected by transient fault and to reconfigure the I/O and IC networks. The performance projections show that these in-flight reconfigurations can be accomplished without suspending any of the ALS applications tasks. The reconfigurability of the networks is also intended to obviate expensive launch pad repairs.

Extensive analytical modeling of the AIPS for ALS architecture was carried out to predict its reliability and availability. For the baseline architecture, consisting of one core non-propulsion FTP and 17 propulsion control FTPs, all quadruply redundant, the launch availability is expected to be 97.9%. This configuration will also meet the mission reliability goals. Specifically, the mission probability of failure is expected to be $9.75*10^{-6}$ This exceeds the goal of 10^{-5} just slightly.

The contributions to the unavailability and unreliability come predominantly from the propulsion avionics since 17 out of the 18 FTPs are for engine control. The availability of the non-propulsion avionics, which consist of just 1 FTP, is 99.88%. Similarly, the mission loss probability, attributable to non-propulsion avionics, is $5.42*10^{-7}$. Evidently, one needs to reexamine the requirement of dedicating a controller to each engine. If, for example, an FTP was configured to control 4 engines, which it is capable of doing based on the performance projections, only 4 FTPs will be necessary to control 16 engines. This would reduce the probability of mission failure due to a failure of propulsion avionics to $2.16*10^{-6}$. The launch availability would improve to 99.52%.

The ALS vehicle and engine designers should seriously examine the option of integrating engine controllers outside the engine and with the core avionics. This would not only improve the overall ALS reliability and availability, as demonstrated above, but would also result in reduced weight, volume, power and cost.

The AIPS for ALS architecture defined here is preliminary in nature but shows that the ALS performance and reliability requirements can be met by the AIPS hardware and software building blocks that are built using the state-of-the-art technology available in the 1992-93 time frame. The level of detail in the architecture definition reflects the level of detail available in the ALS requirements. As the avionics requirements are refined, the architecture can also be refined as well as defined in greater detail with the help of analysis and simulation tools. For example, the functions in the core FTP need to be allocated to the computational processor and the I/O processor. This requires a more detailed enumeration of interfunction communication requirements and I/O communication requirements. Also, no effort was expended on defining the detailed I/O architecture. This requires as inputs the sensor details such as the number and type of sensors, their failure rates, and so on. This information can be used to define redundancy levels of sensors and allocate sensors to different redundant layers of the I/O network.

The data presented by Martin Marietta served as an important starting point for determining the requirements of the ALS avionics. During the course of the CSDL-Martin Marietta interaction an active dialogue was set up which would have in time resulted in a more complete definition of a common requirements vocabulary and facilitated the acquisition of comprehensive requirements data.

Several variations on the baseline architecture presented here are also possible and should be modeled and analyzed. These include allocating several engines to one propulsion control FTP, investigating the effects of launch with failures, and using authentication for the I/O and IC networks.

To complete the design of a validated ALS avionics architecture, the architecture synthesis process begun in this study needs to be followed up by the steps outlined in the design for validation methodology. Once an architectural configuration has been selected that satisfies the reliability, performance and other ALS avionics requirements, the next step is the detailed hardware and software design. The detailed design phase will utilize the AIPS building block design knowledgebase. Once the building blocks have been fabricated using the state-of-the-art microelectronics and the system services implemented using the latest Ada Run Time System and compiler, the validation of the integrated avionics system can commence. This validation pertains to the specific hardware and software implementation and not the architecture or the building block characteristics since these have been prevalidated. A test and evaluation of the avionics integrated with applications and actual or simulated I/O should confirm the predicted performability characteristics. Any discrepancies between the predicted and the actual performability should be minor and traceable to the detailed implementation phase rather than the architectural or building block design. These can be corrected by refining the implementation.

Since a prime design driver for ALS is the cost, a study was also undertaken to analyze the impact of the avionics architecture on the launch cost of ALS. A methodology was developed to quantify the contribution to the recurring launch costs due to the reliability and availability characteristics of the avionics. The resulting cost model was then used to predict and compare the costs of two different architectures for the ALS avionics. The two architectures modeled were defined as simplified versions of two potential candidates for the ALS avionics. Architecture 1 represents an elementary method of incorporating fault tolerance into the avionics system. Architecture 2 is a simplified version of the AIPS for ALS architecture defined in the current study.

A useful methodology has been demonstrated for investigating the impact of the avionics suite to the recurring cost of the ALS. The methodology evaluates the cost of the unreliability of the avionics suite and includes this along with a more traditional assessment of its cost. This allows a truer appraisal of its recurring launch cost to be made. During design and development of the avionics system, this methodology would allow the designer to quantitatively predict the impact on cost that design decisions will have. The

methodology also measures the impact on cost that operational decisions can have — such as allowing the vehicle to launch with detected failures.

When the avionics system is viewed from the perspective of being mission critical, the cost of its unreliability can represent a significant portion of its total cost. Therefore, the parameters which affect its unreliability also affect its recurring launch cost. For the two architectures analyzed in this study, the impacts of two parameters are of particular interest. The first is the repair strategy which is employed for failures detected while the vehicle sits on the launch pad. Pre-empting launch on the occurrence of the first detected failure within the avionics system is not economical. Allowing the vehicle to launch with selected detected failures can potentially reduce the recurring launch costs. The second parameter of interest is the quality level of the components used. The use of Class B parts, combined with a fault tolerant architecture, as compared with Class S and D parts, minimizes the recurring launch cost for most of the repair strategies analyzed. It should be noted here that the current philosophy for launch system avionics is to use the highest quality, that is, Class S, components in a single string, non-fault tolerant architecture.

Based on the analysis performed, the AIPS FTP architecture (Architecture 2) shows itself to be potentially more economical than the more typical architecture employed for this application (Architecture 1). The more viable repair strategies of the AIPS FTP architecture show themselves to be much more cost effective than any of the strategies of Architecture 1 for the Baseline System Parameters. These repair strategies of the AIPS FTP architecture also appear to be more insensitive to a number of the major system parameters. For the AIPS FTP architecture, allowing the ALS to launch with detected failures in 1 Sensor and/or 1 Channel and/or 1 Actuator (delaying repair until failures are detected in at least 2 Sensors or 2 Channels or 2 Actuators) appears to be the most economical repair strategy.

Though the methodology presented here shows itself to be very useful in quantifying the impact of the reliability of the avionics system on the recurring launch cost of the ALS, the actual analysis performed is of limited benefit. The models of the two architectures which are the focus of the study were quite simplified and lacked the complexity of an actual avionics system. Higher fidelity models that reflect the details of the architecture need to be developed.

With regard to the methodology itself, the costs associated with interrupting the countdown and a failed mission could be modeled better. The less tangible costs such as a delay in a launch schedule because of the need to do a repair or because a vehicle is lost in flight needs to be investigated further. The analysis could also be expanded to include other critical subsystems of the ALS along with the avionics system. The methodology might also be enhanced to include reusable subsystems within the ALS and possibly compare the costs of these architectures with their expendable counterparts.

7.0 REFERENCES

1. R. Harper, L. Alger, and J. Lala, "Advanced Information Processing System: Design and Validation Knowledgebase", NASA Contractor Report 187544, September, 1991.
2. R. Cole, "Advanced Information Processing System for Advanced Launch System: Hardware Technology Survey and Projections", NASA Contractor Report 187555, September, 1991.
3. Hypercard Stack "XPROCESSING," Martin Marietta Astronautics Group, Denver, CO 7 December 1989.
4. Hypercard Stack "ZARCS," Martin Marietta Astronautics Group, Denver, CO 7 December 1989.
5. S. Johnson, "IRAD ALS Requirements Summary," Martin Marietta Astronautics Group, Denver, CO, 8 November 1989.
6. "Future Launch Vehicle (ALS_PLUS) Requirements," Martin Marietta Astronautics Group, 29 October 1989.
7. J. H. Lala and S.J. Adams,"Inter-Computer Communication Architecture for a Mixed Redundancy Distributed System", Journal of Guidance, Control, and Dynamics, Vol. 12, No. 4, July-August 1989.
8. L. Alger, and J. Lala, "Performance Evaluation of a Realtime Fault Tolerant Distributed System", 23rd Hawaii International Conference of System Sciences, Kailua-Kona, Hawaii, January, 1990.
9. L. Burkhardt, L. Alger, R. Whittredge, and P. Stasiowski, "Advanced Information Processing System: Local System Services", NASA Contractor Report 181767, April, 1989.
10. T. Masotto and L. Alger, "Advanced Information Processing System: Input/Output System Services", NASA Contractor Report 181874, August, 1989.
11. G. Nagle, L. Alger and A. Kemp, "Advanced Information Processing System: Input/Output Network Management Software", NASA Contractor Report 181678, May, 1988.
12. L. Burkhardt, T. Masotto J. Terry Sims, R. Whittredge and L. Alger, "Advanced Information Processing System: Inter-Computer Communication Services", NASA Contractor Report 187556, September, 1991.
13. S. J. Adams, M. Dzwonczyk, "Techniques for Transient Error Recovery and Avoidance in Redundant Processing Systems", NATO/AGARD 49th Symposium on Fault Tolerant Design Concepts for Highly Integrated Flight Critical Systems, Toulouse, France, October 10-13, 1989.
14. S. J. Adams, "Hardware assisted Recovery from Transient Errors in Redundant Processing Systems," 18th International Symposium on Fault Tolerant Computing, Chicago, IL, June 1989.

15. Babcock, IV, P. S. and K. C. Hell, A Cost/Reliability Model of Electronic Functions, CSDL-R-2109, The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts, October 1988.
16. Military Handbook, Reliability Prediction of Electronic Equipment, MIL-HDBK-217E, Department of Defense, Washington, D. C., October 27, 1986.
17. H. Kando, T. Iwazumi and H. Ukai, "Singular Perturbation Modelling of Large Scale Systems with Multi-Time-Scale Property, " Int. J. Control, vol 48, No. 6, 1988.
18. K. Trivedi and R. Geist, "Decomposition in Reliability Analysis of Fault-Tolerant Systems," IEEE Trans. on Reliability, vol. R-32, No. 5, December 1983.
19. J. McGough, K. Smotherman and K. Trivedi, "The Conservativeness of Reliability Estimates Based on Instantaneous Coverage," IEEE Trans. on Computers, vol. C-34, No. 7, July 1985.
20. A. Bobbio and K. Trivedi, "An Aggregation Technique for the Transient Analysis of Stiff Markov Chains," IEEE Trans. on Computers, vol. C-35, No. 9, September 1986.
21. L. Segal and M. Slemrod, "The Quasi-Steady State Assumption: A Case Study in Perturbation, " SIAM Review, vol. 31, No. 3, September 1989.
22. J.T. Sims, "Performance Benchmarks for the VLSI - Fault Tolerant Processor," C.S. Draper Laboratory Internal Memorandum, February 1988.
23. Dale A. Mackall and S.D. Ishmael, "Qualifications of the Flight Critical AFTI/F-16 Digital Flight Control System," The 21st Aerospace Sciences Meeting, AIAA-83-0063, Reno, Nevada, January 1983.
24. Dale A. Mackall, "AFTI/F-16 Digital Flight Control System Experience," First Annual NASA Aircraft Controls Workshop, NASA Langley Research Center, Hampton, Virginia, October 1983.

APPENDIX A

HIGHEST LEVEL SPREADSHEET OF THE COST MODEL

I. System Parameters

A. Component Attributes

		Architecture 1	Architecture 2
Sensor	Base Failure Rate (/hour)	1.00E-05	1.00E-05
	Duplex Coverage Probability	not applicable	0.900
	Simplex Coverage Probability	not applicable	0.500
	FDIR Rate (/hour)	not applicable	1.00E+04
	Weight (pounds)	10	10
	Unit Cost ($)	10,000	10,000
Channel	Base Failure Rate (/hour)	1.48E-04	see FTP_Pad_Model [1]
	Dual Coverage Probability	not applicable	see FTP_Pad_Model [1]
	FDIR Rate (/hour)	not applicable	see FTP_Pad_Model [1]
	Weight (pounds)	10	10
	Unit Cost ($)	25,000	25,000
Actuator	Base Failure Rate (/hour)	1.00E-05	1.00E-05
	Duplex Coverage Probability	not applicable	0.900
	Simplex Coverage Probability	not applicable	0.500
	FDIR Rate (/hour)	not applicable	1.00E+04
	Weight (pounds)	10	10
	Unit Cost ($)	5,000	5,000

[1] From FTP_Pad_Model worksheet:

- **Reliability Parameters for Draper FTP**

Channel Component	Permanent Failure Rate (/h)	Transient Failure Rate (/h)	Duplex Coverage Probability	Permanent Recovery Rate (/h)	Transient Recovery Rate (/h)
Processor	1.867E-05	1.867E-04	0.900	9.000E+04	9.000E+04
Memory	1.280E-04	1.280E-03	0.900	9.000E+04	9.000E+04
Interstage	1.000E-06	1.000E-05	0.900	9.000E+04	9.000E+04

B. Avionics System Attributes

	Architecture 1	Architecture 2
Design and Development Cost ($)	10,000,000	10,000,000
Construction Cost (excluding cost of parts) ($)	50,000	50,000
Weight of Integration Hardware (pounds)	40	40

C. Vehicle Attributes

VHMS Detection and Interruption Rate (/hour)	1.20E+01
VHMS Self-Test Coverage Probability	.9000
Payload Launch Cost ($/pound of payload)	300

D. Fleet Attributes

Number of Vehicles in Fleet	100

E. Operational Attributes

1. On the Pad

Pad Environmental Factor	2.50
On/Off Failure Rate Scale Factor	100.00
Ratio of Hours On/Off	0.50
Time on Pad (days)	7.00
Repair Time (days)	5.00
Surge Work Rate	1.35
Overtime Cost (x Normal)	1.50
Operation Cost (per day)	$150,000

2. During Launch/Ascent

Payload Value ($)	$1,000,000,000
Total Payload Weight (pounds)	200,000
Launch Environmental Factor	13.0
Flight Time (hours)	0.167

II. Cost of System

	Architecture 1	Architecture 2
Cost of Avionics System	$260,000	$270,000

III. Cost of Weight

	Architecture 1	Architecture 2
Cost of Avionics System Launch Weight	$33,000	$39,000

IV. Cost of Unreliability

A. On the Pad

1. Intermediate Calculations

	Architecture 1	Architecture 2
Effective Failure Rate		
Sensor	8.50E-06	8.50E-06
Channel	1.26E-04	see FTP_Pad_Model [2]
Actuator	8.50E-06	8.50E-06

---

[2]From FTP_Pad_Model worksheet:

- **Adjusted Reliability Parameters**

Channel Component	Permanent Failure Rate (/h)	Transient Failure Rate (/h)	Duplex Coverage Probability	Permanent Recovery Rate (/h)	Transient Recovery Rate (/h)
Processor	1.59E-05	1.59E-04	0.900	9.00E+04	9.00E+04
Memory	1.09E-04	1.09E-03	0.900	9.00E+04	9.00E+04
Interstage	8.50E-07	8.50E-06	0.900	9.00E+04	9.00E+04
0	0.00E+00	0.00E+00	0.000	0.00E+00	0.00E+00
0	0.00E+00	0.00E+00	0.000	0.00E+00	0.00E+00

Cost of Interrupting Count-Down	$1,821,429

2. Pad Unreliability Cost

a. Architecture 1

On-Pad Repair Policy	Cost Lower Bound	Cost Upper Bound
First Detected Failure	$121,082	$123,371
Actuator Failure or Loss of 2 Control Chains	$5,006	$5,117
No Repairs	$0	$0

b. Architecture 2

On-Pad Repair Policy	Cost Lower Bound	Cost Upper Bound
First Detected Failure	$126,242	$127,975
Detected Failure of 2 Sensors or 1 Channel or 2 Actuators	$114,129	$114,129
Detected Failure of 2 Sensors or 2 Channels or 2 Actuators	$4,815	$4,815
Detected Failure of 3 Sensors or 2 Channels or 3 Actuators	$4,830	$4,830
No Repairs	$0	$0

B. During Launch

1. Intermediate Calculations

	Architecture 1	Architecture 2
Mission Cost	$1,060,260,000	$1,060,270,000
Effective Failure Rate		
Sensor	1.30E-04	1.30E-04
Channel	1.92E-03	see FTP_Launch_Model [3]
Actuator	1.30E-04	1.30E-04

2. Launch Unreliability Cost

a. Architecture 1

On-Pad Repair Policy	Cost
First Detected Failure	$845,535
Actuator Failure or Loss of 2 Control Chains	$848,574
No Repairs	$3,762,342

---

[3]From FTP_Launch_Model worksheet:

- **Adjusted Reliability Parameters**

Channel Component	Permanent Failure Rate (/h)	Transient Failure Rate (/h)	Duplex Coverage Probability	Permanent Recovery Rate (/h)	Transient Recovery Rate (/h)
Processor	2.67E-03	0.00E+00	0.900	9.00E+04	0.00E+00
Memory	1.83E-02	0.00E+00	0.900	9.00E+04	0.00E+00
Interstage	1.43E-04	0.00E+00	0.900	9.00E+04	0.00E+00
0	0.00E+00	0.00E+00	0.000	0.00E+00	0.00E+00
0	0.00E+00	0.00E+00	0.000	0.00E+00	0.00E+00

b. Architecture 2

On-Pad Repair Policy	Cost Lower Bound	Cost Upper Bound
First Detected Failure	$3,714	$3,746
Detected Failure of 2 Sensors or 1 Channel or 2 Actuators	$3,783	$3,785
Detected Failure of 2 Sensors or 2 Channels or 2 Actuators	$48,176	$48,177
Detected Failure of 3 Sensors or 2 Channels or 3 Actuators	$49,470	$49,474
No Repairs	$1,624,328	$1,624,328

C. Total Unreliability Cost

1. Architecture 1

On-Pad Repair Policy	Cost Lower Bound	Cost Upper Bound
First Detected Failure	$966,616	$968,906
Actuator Failure or Loss of 2 Control Chains	$853,580	$853,692
No Repairs	$3,762,342	$3,762,342

2. Architecture 2

On-Pad Repair Policy	Cost Lower Bound	Cost Upper Bound
First Detected Failure	$129,956	$131,721
Detected Failure of 2 Sensors or 1 Channel or 2 Actuators	$117,911	$117,914
Detected Failure of 2 Sensors or 2 Channels or 2 Actuators	$52,991	$52,992
Detected Failure of 3 Sensors or 2 Channels or 3 Actuators	$54,300	$54,304
No Repairs	$1,624,328	$1,624,328

V. Total Avionics System Cost (System, Weight, Unreliability)

A. Architecture 1

On-Pad Repair Policy	Cost Lower Bound	Cost Upper Bound
First Detected Failure	$1,259,616	$1,261,906
Actuator Failure or Loss of 2 Control Chains	$1,146,580	$1,146,692
No Repairs	$4,055,342	$4,055,342

B. Architecture 2

On-Pad Repair Policy	Cost Lower Bound	Cost Upper Bound
First Detected Failure	$538,956	$540,721
Detected Failure of 2 Sensors or 1 Channel or 2 Actuators	$526,912	$526,914
Detected Failure of 2 Sensors or 2 Channels or 2 Actuators	$461,995	$461,996
Detected Failure of 3 Sensors or 2 Channels or 3 Actuators	$463,305	$463,309
No Repairs	$2,033,481	$2,033,481

APPENDIX B

MODEL REDUCTION TECHNIQUE FOR FTP ANALYSIS

B.1 Introduction

There is an increasing demand for fault-tolerant control systems in high performance, critical applications. At the core of such systems, there is one or more fault tolerant processors (FTP's). The FTP receives information from sensors, sends commands to actuators and performs redundancy management. Given its critical role in any control system, the operation and performance of the FTP must be very carefully modeled if the safety and performance evaluation process is to be relied upon.

An FTP may contain computing elements, dedicated and/or shared memories, information replicating components, I/O-dedicated electronics, etc. The key feature of an FTP is the ability to handle faults in a controlled, timely manner that enables correct reconfiguration and uninterrupted (on the time scale of interest) operation. An accurate model must be able to describe in detail rate processes such as component failures, fault detection, fault isolation and reconfiguration.

The Markov modeling method has clearly emerged as the preferred approach with regard to the analysis of such processing systems. The ability of this approach to correctly capture rate processes and event sequence dependencies are of crucial importance. While discrete simulation approaches might be perfectly adequate from the point of view of modeling flexibility, computational efficiency strongly tilts the balance towards the Markovian approach. This is so because, for such highly reliable systems, the component failure rates are very low. Consequently, the fault occurrence event rate is so low as to require a prohibitively large number of trials in order to accumulate statistically meaningful and reasonably accurate results.

The Markov method however suffers from a major drawback. The number of states proliferate rapidly, often leading to an intractably large model. The FTP proper represents a system of moderate size, such that even a rather detailed model does not generally pose a major problem. However, when attempting to analyze the entire control system which the FTP is part of, maintaining the level of detail desirable for the FTP alone would most likely give rise to an unwieldy, perhaps even intractable model. Techniques such as aggregation, truncation and decomposition are used to mitigate this basic difficulty. Still, it would be highly desirable to devise a simplified model of the FTP which would greatly contribute to alleviating the space explosion problem while correctly preserving the main features of the detailed model.

The purpose of this appendix is to describe the development of such an approximate model. First a detailed Markov model of a triply redundant FTP is introduced. The model reduction technique is then presented, leading to an excellent approximate model for this

FTP. A complete analytical solution of the reduced model follows. To get a feel for the approximations involved, the procedure is shown applied to a simple example, for which analytical solutions are feasible for both the exact and reduced models. The appendix ends with a few concluding remarks regarding the reduction technique to be presented.

B.2 Detailed Triplex FTP Markov Model

A detailed Markov model for a Quad FTP is described in Section 4 of [1]. The model tracks separately, within a computational channel, the processor element, the associated dedicated memory and the corresponding interstage. Both permanent and transient failures are accounted for, along with the appropriate reconfiguration mechanism. Provisions are made for including common mode failures as well. This model provides a realistic paradigm of the fault occurrence/fault handling processes.

The cost analysis presented in Section 5 is focused on a comparison of two triplex architectures, one of which is based on an AIPS triplex FTP. The approach used in constructing the quad model was used to generate a similar triplex Markov model. In order to simplify the description of the model reduction method, some additional assumptions were made:

- the processor and its associated memory in one channel were treated as one component,
- the triplex coverage was assumed perfect and
- common mode failure was disregarded.

The Markov model for the triplex FTP, based on these assumptions, is shown in Figure B-1. The notation used for the state transition rates is the following:

- $\lambda^{n}_{ab\ (,sl)}$ is the failure rate for component **a** (where a = p for the processor and a = i for the interstage) from configuration **n** (where n = t, d or s, i.e., triplex, dual or single, respectively); **b** indicates the type of failure, (with b = t or p denoting transient or permanent failure); **sl** stands for system loss.
- ρ^{n}_{ab} is the rate of the reconfiguration process initiated by a **b**-type failure of component **a** starting from configuration **n**; here **a**, **b** and **n** have the same meaning as above.

State 1 represents operation with no failures. States 6 and 11 correspond to degraded modes of operation, namely operation with one channel failed and two channels failed, respectively. Finally state 12 denotes an aggregated system loss condition. This state is reached either as a result of incorrect reconfiguration or because of exhaustion.

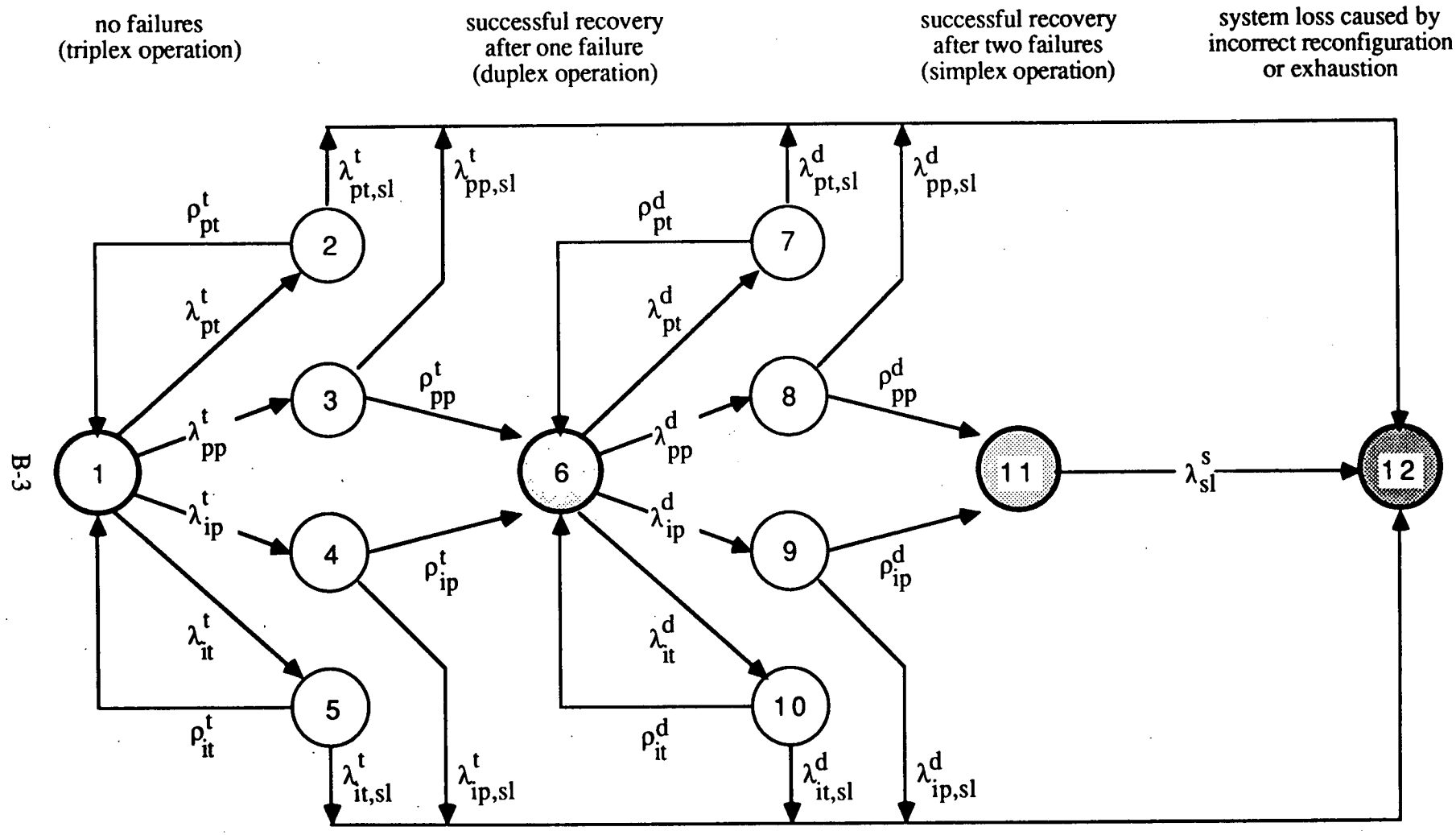

Figure B-1 Detailed Markov Model of a Triplex FTP

The group comprising states 2,3,4 and 5 is associated with the state of the system after one failure and the group including states 7,8,9 and 10 corresponds to the system after two failures. The states in these two groups are characterized by extremely short holding times, when compared to states 1, 6, 11 or 12. Specifically, the "fast" states represent intermediate configurations, persisting for only very short periods of time during the reconfiguration processes. The reader should note that both permanent and transient failures are accounted for, with a distinct reconfiguration path. Specifically, transient failures, i.e., states 2, 5, 7 and 10, are reconfigured back to their respective origin states, i.e., states 1 and 6. In contrast, permanent failures, i.e., states 3, 4, 8 and 9, lead to reconfigurations to the appropriate degraded operational modes, states 6 and 11.

This Markov model captures the key fault occurrence and handling processes in a triplex FTP, designed to withstand Byzantine faults. The actual model used in the cost analysis contains additional details, which makes it a realistic tool for studying the performance of an FTP processor.

It is perfectly feasible to use this model to analyze the FTP on a stand alone basis. Still, care must be taken in the solution technique to overcome difficulties caused by the very pronounced stiffness of the resulting system of ordinary differential equations. Indeed, there is an enormous discrepancy between the time constants characterizing the failure events and those associated with reconfiguration processes.

When the FTP must be analyzed as a subsystem within a much larger control system, the high level of detail in the model becomes a liability. This is so both because of the large state space the analyst will have to deal with and in view of the stiffness aspect mentioned above. It is thus natural to search for an approximation technique allowing a high level of fidelity, while providing a much more tractable model to work with. Such a technique will be described in the next section.

B.3 FTP Model Reduction

As already mentioned, the detailed Markov model has a number of states characterized by a very short time constant relative to the time scale of the failure events. This situation immediately suggests the possibility of a behavioral decomposition. The existence of distinct time scales is often encountered in control system applications and is exploited to generate a reduced model of the original system. Reference 17 presents a good review of this approach in the system control area. The same basic idea is also often used in simplifying various physical models (see, for example, [17]). These reduction techniques often rely on detailed eigenvalue analysis and consequently are rather cumbersome. The reduction techniques become considerably more appealing when it is obvious which states are characterized by fast time constants and which ones are "slow".

In the reliability analysis field, the need to model both the fault occurrence (slow) and the fault handling (fast) processes leads naturally to the situation previously described. There is a strong incentive to perform a systematic behavioral (or temporal) decomposition in order to both reduce the size of the state space and also remove the severe stiffness of the mathematical model. References [18] and [19] propose a decomposition approach. While the approach is well founded, it is quite impractical for complex applications because of some rather cumbersome probabilistic arguments used to determine aggregated transition rates. In [20] Bobbio suggests another approach, similar to that used in [21], which leads to a systematic and straightforward reduction model reduction procedure. The technique does not use a formal eigenvalue analysis, relying solely on an examination of the original, detailed model structure and transition rates. This technique will be applied to obtain a reduced FTP model.

The state transition rates may clearly be divided into two separate sets, one consisting of the *slow* rates, i.e., the failure rates, the other consisting of the *fast* rates, i.e., the reconfiguration rates. We can then partition the n states of the model into two disjoint and exhaustive subsets defined as follows:

- [S] is the set of n_S *slow* states (1, 6, 11 and 12), i.e., states with no outgoing transitions classified as fast,

- [F] is the set of n_F *fast* states, i.e., states with at least one fast outgoing transition.

For convenience, we further subdivide the set [F] into the subsets [F1] and [F2], corresponding to the fast states reached following a single failure (2, 3, 4 and 5) and two failures (7, 8, 9 and 10), respectively. The transition matrix and the associated probability vector are then reordered such that the equations governing the states in [S] become the first n_S equations, followed by the n_{F1} equations corresponding to the states in [F1] and the n_{F2} equations corresponding to the states in [F2], with $n_F = n_{F1} + n_{F2}$.

After this reordering, the equations describing the Markov model can be written as:

$$\frac{d}{dt}\begin{bmatrix} P_S \\ P_{F1} \\ P_{F2} \end{bmatrix} = \begin{bmatrix} D_0 & B_{01} & B_{02} \\ B_{10} & D_1 & O \\ B_{20} & O & D_2 \end{bmatrix} \begin{bmatrix} P_S \\ P_{F1} \\ P_{F2} \end{bmatrix} \qquad \text{(B-1)}$$

where:

$$D_0 = \begin{bmatrix} -\lambda_1 & 0 & 0 & 0 \\ 0 & -\lambda_6 & 0 & 0 \\ 0 & 0 & -\lambda_{11} & 0 \\ 0 & 0 & +\lambda_{11} & 0 \end{bmatrix}$$

$D_1 = \text{diag}(-\delta_2, -\delta_3, -\delta_4, -\delta_5)$

$D_2 = \text{diag}(-\delta_7, -\delta_8, -\delta_9, -\delta_{10})$

$O = \text{null matrix}$

$$B_{01} = \begin{bmatrix} \rho_{21} & \rho_{31} & \rho_{41} & \rho_{51} \\ \rho_{26} & \rho_{36} & \rho_{46} & \rho_{56} \\ 0 & 0 & 0 & 0 \\ \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 \end{bmatrix} \quad B_{02} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \rho_{76} & \rho_{86} & \rho_{96} & \rho_{10,6} \\ \rho_{7,11} & \rho_{8,11} & \rho_{9,11} & \rho_{10,11} \\ \lambda_7 & \lambda_8 & \lambda_9 & \lambda_{10} \end{bmatrix}$$

$$B_{10} = \begin{bmatrix} \lambda_{12} & 0 & 0 & 0 \\ \lambda_{13} & 0 & 0 & 0 \\ \lambda_{14} & 0 & 0 & 0 \\ \lambda_{15} & 0 & 0 & 0 \end{bmatrix} \quad B_{20} = \begin{bmatrix} 0 & \lambda_{67} & 0 & 0 \\ 0 & \lambda_{68} & 0 & 0 \\ 0 & \lambda_{69} & 0 & 0 \\ 0 & \lambda_{6,10} & 0 & 0 \end{bmatrix}$$

For facility, the notation follows the convention indicated in Figure B-2. The total outgoing transition rate from the fast state "k" is denoted δ_k and is given by:

$$\delta_k = \rho_{ki} + \rho_{kj} + \lambda_k$$

while the total outgoing transition rate from slow state "i" is denoted λ_i and is given by:

$$\lambda_i = \sum \lambda_{ik}$$

where the summation is implied over all the fast states "fed" by slow state "i".

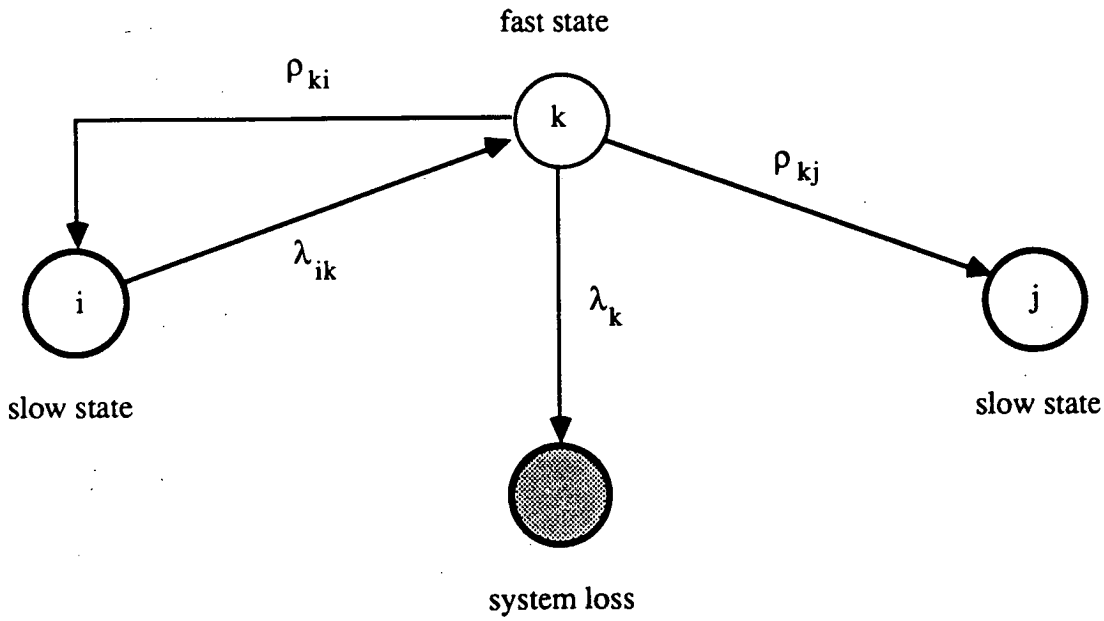

Figure B-2 Notation Convention

At this point, the key approximation is made that the *fast states reach their steady state well within the time scale of interest,* i.e., mission time. In other words, the fast states are assumed to respond instantaneously to changes in the slow states. Setting the temporal derivatives of the fast states' probabilities to zero leads to the following approximate expressions for the probability subvectors P_{F1} and P_{F2}:

$$P_{F1} = -D_1^{-1}B_{10}P_S \quad \text{and} \quad P_{F2} = -D_2^{-1}B_{20}P_S \tag{B-2}$$

These expressions for P_{F1} and P_{F2} are used to eliminate them in favor of P_S in the first n_S equations, leading to the following reduced system of equations for P_S:

$$\dot{P}_S = \left[D_0 - B_{01}D_1^{-1}B_{10} - B_{02}D_2^{-1}B_{20}\right] P_S = A_S^* P_S \tag{B-3}$$

It should be noted that the algebraic manipulations implied in (B-2) and (B-3) are particularly easy to carry out in our application because of the specific structure of the Markov model. Indeed, the submatrices D_1 and D_2 are strictly diagonal, making their inversion trivial. The significance of their strictly diagonal structure is that no direct coupling exists among the fast states. Moreover, B_{10} and B_{20} are very sparse, further reducing the computational effort required to obtain the approximate model equations.

The transition matrix of the reduced model has the following structure:

$$A_S^* = \begin{bmatrix} a_{11}^* & 0 & 0 & 0 \\ a_{21}^* & a_{22}^* & 0 & 0 \\ 0 & a_{32}^* & a_{33}^* & 0 \\ a_{41}^* & a_{42}^* & a_{43}^* & 0 \end{bmatrix} \tag{B-4}$$

where:

$$a_{11}^* = -\sum_{k=2}^{5} \frac{\lambda_{1k}}{\delta_k}\left(\rho_{k6} + \lambda_k\right), \quad a_{22}^* = -\sum_{k=7}^{10} \frac{\lambda_{6k}}{\delta_k}\left(\rho_{k,11} + \lambda_k\right), \tag{B-5a}$$

$$a_{21}^* = \sum_{k=2}^{5} \frac{\lambda_{1k}}{\delta_k}\rho_{k6}, \quad a_{32}^* = \sum_{k=7}^{10} \frac{\lambda_{6k}}{\delta_k}\rho_{k,11}, \tag{B-5b}$$

$$a_{41}^* = \sum_{k=2}^{5} \frac{\lambda_{1k}}{\delta_k}\lambda_k, \quad a_{42}^* = \sum_{k=7}^{10} \frac{\lambda_{ki}}{\delta_k}\lambda_k, \tag{B-5c}$$

$$a_{33}^* = -\lambda_{11}, \quad a_{43}^* = \lambda_{11}. \tag{B-5d}$$

The reduced formulation is thus fully defined in terms of the original model parameters. A more insightful and convenient, but still fully equivalent formulation, that clearly identifies the successful as well as the failed reconfigurations, can be obtained by rewriting the reduced model transition matrix in the following form:

$$A_S^* = \begin{bmatrix} a_{11}^* & 0 & 0 & 0 \\ -c^t a_{11}^* & a_{22}^* & 0 & 0 \\ 0 & -c^d a_{22}^* & a_{33}^* & 0 \\ -(1-c^t)a_{11}^* & -(1-c^d)a_{22}^* & -a_{33}^* & 0 \end{bmatrix} \qquad \text{(B-6)}$$

Here, the *equivalent* triplex and duplex coverages are obtained by simply comparing formulations (B-4) and (B-6):

$$c^t = \frac{\sum_{k=2}^{5} \frac{\lambda_{1k}}{\delta_k} \rho_{k6}}{\sum_{k=2}^{5} \frac{\lambda_{1k}}{\delta_k} (\rho_{k6} + \lambda_k)} \quad \text{and} \quad c^d = \frac{\sum_{k=7}^{10} \frac{\lambda_{6k}}{\delta_k} \rho_{k,11}}{\sum_{k=7}^{10} \frac{\lambda_{6k}}{\delta_k} (\rho_{k,11} + \lambda_k)} \qquad \text{(B-7)}$$

A few remarks are in order at this point. Since it was assumed that the triplex coverage (i.e., the detection and the isolation) is perfect, a system loss can be caused only by a coincident failure. Since the reconfiguration rate is many orders of magnitude greater than the failure rate, the equivalent triplex coverage is very nearly 1.0. In contrast, the situation for the dual operation, reached after a successful recovery following a single failure, is quite different. Here, the detection is still assumed perfect, but for isolation we must rely on self-test, which is assigned a probability of success $c_{d,isol} < 1.0$. Consequently, a transition to system loss may take place not only because of a coincident failure but also predominantly because of improper reconfiguration. As a result, the equivalent duplex coverage may be significantly less than 1.0.

A simple example will further clarify this important aspect. Disregarding transient failures and treating the processor/interstage set as one "component" with a failure rate λ and a reconfiguration rate ρ, we have, from (B-7):

$$c^t = \frac{\frac{3\lambda}{\rho + 2\lambda}\rho}{\frac{3\lambda}{\rho + 2\lambda}(\rho + 2\lambda)} = \frac{1}{1 + \frac{2\lambda}{\rho}} \qquad \text{(B-8)}$$

$$c^d = \frac{\dfrac{2\lambda}{c_{d,isol}\rho + (1-c_{d,isol})\rho + \lambda} c_{d,isol}\rho}{\dfrac{2\lambda}{c_{d,isol}\rho + (1-c_{d,isol})\rho + \lambda} [c_{d,isol}\rho + (1-c_{d,isol})\rho + \lambda]} = \frac{c_{d,isol}}{1 + \dfrac{\lambda}{\rho}} \qquad \text{(B-9)}$$

Since $\lambda \ll \rho$, then $c^t \approx 1.0$ and $c^d \approx c_{d,isol}$. The expressions (B-7) properly reduce to the simple model often used to represent a triplex FTP.

The reduced model obtained in this section is illustrated in Figure B-3. The three operating states (1, 2 and 3) in the reduced model correspond on a one-to-one basis to the operating states (1, 6 and 11) in the original model. The transition rates used in this model are given by the expressions (B-5) and and (B-7). From the initial n-state model (n = 12), an approximate model containing only n_S states ($n_S = 4$) has been obtained. Extensive numerical experimentation, comparing the original model with the reduced one, has consistently indicated an excellent agreement, proving the validity of this temporal decomposition technique in our application.

The simplicity of the reduced model allows a fully analytical solution, which will be introduced in the next section.

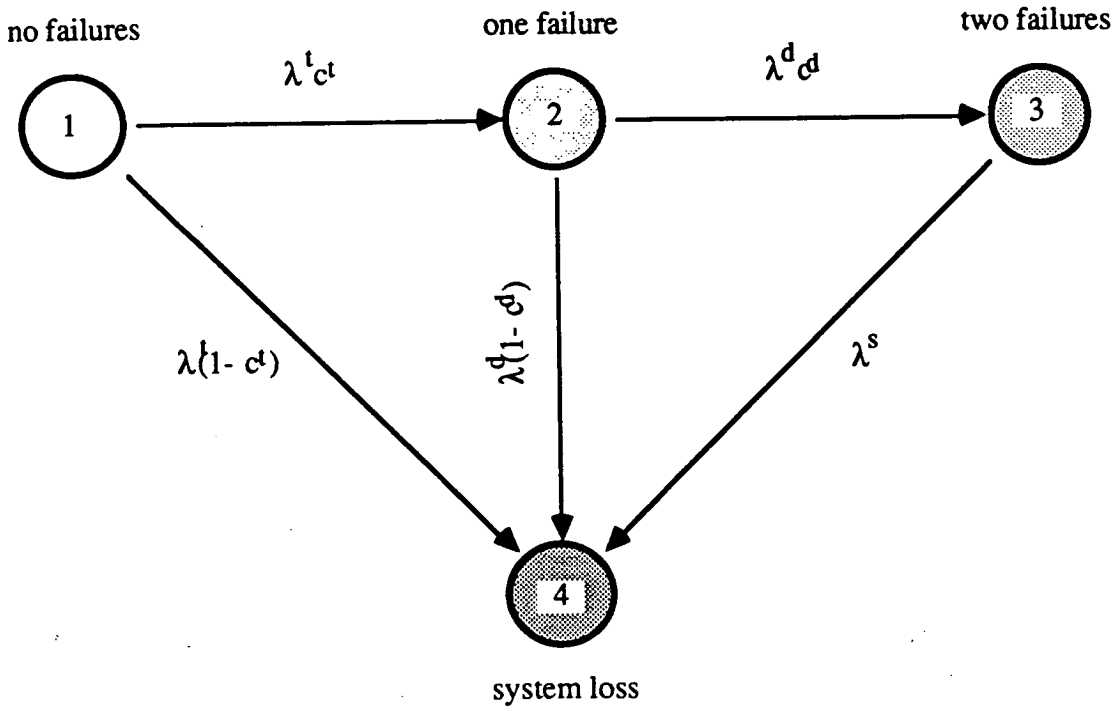

Figure B-3 Reduced Markov Model of a Triplex FTP

B.4 Analytical Solution of the Reduced FTP Model

The simple, non-cyclic structure of the reduced model allows a compact, analytical solution. The reduced model is basically a chain, with additional transitions to system loss before component exhaustion. It can be easily shown that the solution to the model depicted in Figure B-3 is:

$$P_1 = e^{-\lambda^t t} \tag{B-10a}$$

$$P_2 = \frac{\lambda^t c^t}{\lambda^t - \lambda^d}\left[e^{-\lambda^d t} - e^{-\lambda^t t}\right] \tag{B-10b}$$

$$P_3 = \frac{(\lambda^t c^t)(\lambda^d c^d)}{(\lambda^t - \lambda^d)(\lambda^d - \lambda^s)(\lambda^t - \lambda^s)}\left[(\lambda^d - \lambda^s)e^{-\lambda^t t} - (\lambda^t - \lambda^s)e^{-\lambda^d t} + (\lambda^t - \lambda^d)e^{-\lambda^s t}\right] \tag{B-10c}$$

The probabilities correspond to the three operational modes postulated for the triplex FTP, i.e., operation with no failures, with one failure and with two failures, respectively. For the particular case when $\lambda^t = 3\lambda$, $\lambda^d = 2\lambda$ and $\lambda^s = \lambda$, these formulas

take on an especially simple, compact form, i.e., a binomial formula modified to account for imperfect coverage,

$$P_1 = R_0^3 \qquad \text{(B-11a)}$$

$$P_2 = 3\, c^t\, R_0^2 (1 - R_0) \qquad \text{(B-11b)}$$

$$P_3 = 3\, c^t\, c^d\, R_0 (1 - R_0)^2 \qquad \text{(B-11c)}$$

where $R_0 = \exp(-\lambda t)$. In these formulas, the appearance of the coverage probabilities account for the obvious need for successful reconfiguration if the FTP is to continue to operate in a degraded mode.

This analytical formulation is a powerful tool for carrying out extensive parametric studies. The formulation can be easily adapted to a different type of FTP, for example a quad configuration.

B.5 Exact and Reduced Models for a Simple Example

It is instructive to examine a simple example to reveal the exact nature of the approximations involved in carrying out the model order reduction procedure outlined up to this point. A simple example will further clarify this important aspect. Let us consider a dual FTP, treating a processor/interstage set as one component, subject to both transient and permanent failures occurring at the rates λ_t and λ_p, respectively. The same reconfiguration rate, ρ, is assumed for both transient and permanent failures. The self-test coverage, accounting for the imperfect isolation characteristic of the dual architecture, is denoted by c. In view of the much greater rate of the reconfiguration process compared to the rate of failure events, the effect of a coincident failure will be disregarded. The Markov model incorporating all these assumptions is illustrated in Figure B-4. This model can be solved analytically to yield the following expression for the probability of system loss:

$$P_{SL}^{exact} = 1 - A\, e^{-s_1 t} - B e^{-s_2 t} - C\, e^{-\lambda t} \qquad \text{(B-12)}$$

where the coefficients and the exponents are given by:

$$A = \frac{\lambda [s_2 - 2\rho\,(1\text{-}c)]}{(s_1 - s_2)(s_1 - \lambda)}, \quad B = -\frac{\lambda [s_1 - 2\rho\,(1\text{-}c)]}{(s_1 - s_2)(s_2 - \lambda)}, \quad C = \frac{2c\lambda_p\rho}{(s_1 - \lambda)(s_2 - \lambda)}$$

s_1 and s_2 are the roots of: $s^2 + (2\lambda + \rho)\, s + 2\rho\left[\lambda_p + (1\text{-}c)\lambda_t\right] = 0$

and λ is the total failure rate, i.e., $\lambda = \lambda_t + \lambda_p$.

Applying the procedure previously outlined, the initial model is reduced to the approximate model shown in Figure B-5. The probability of system loss for this model is given by:

$$P_{SL}^{approx} = 1 - \tilde{B}e^{-2[\lambda_p + (1-c)\lambda_t]t} - \tilde{C}\,e^{-\lambda t} \qquad \text{(B-13)}$$

where

$$\tilde{B} = \frac{1 - 2c}{1 - 2c\left(\frac{\lambda_t}{\lambda}\right)}, \quad \tilde{C} = \frac{2c\left(1 - \frac{\lambda_t}{\lambda}\right)}{1 - 2c\left(\frac{\lambda_t}{\lambda}\right)}$$

It can be easily shown that if $\lambda \ll \rho$, then the roots of the quadratic are well approximated by:

$$s_2 \approx 2\left[\lambda_p + (1-c)\lambda_t\right] \quad \text{and} \quad s_1 \approx \rho \qquad \text{(B-14)}$$

Substituting (B-14) into the expression of the coefficients in (B-12) leads to the conclusion that :

$$A \to 0, \quad B \to \tilde{B} \quad \text{and} \quad C \to \tilde{C},$$
$$\text{as } (\lambda/\rho) \to 0$$

In addition, it is clear that the first exponential will decay very rapidly compared to the other two. Consequently, for any length of time sufficiently in excess of the time scale of the reconfiguration process (i.e., $1/\rho$), the approximate solution, (B-12), is in excellent agreement with the exact solution, (B-13).

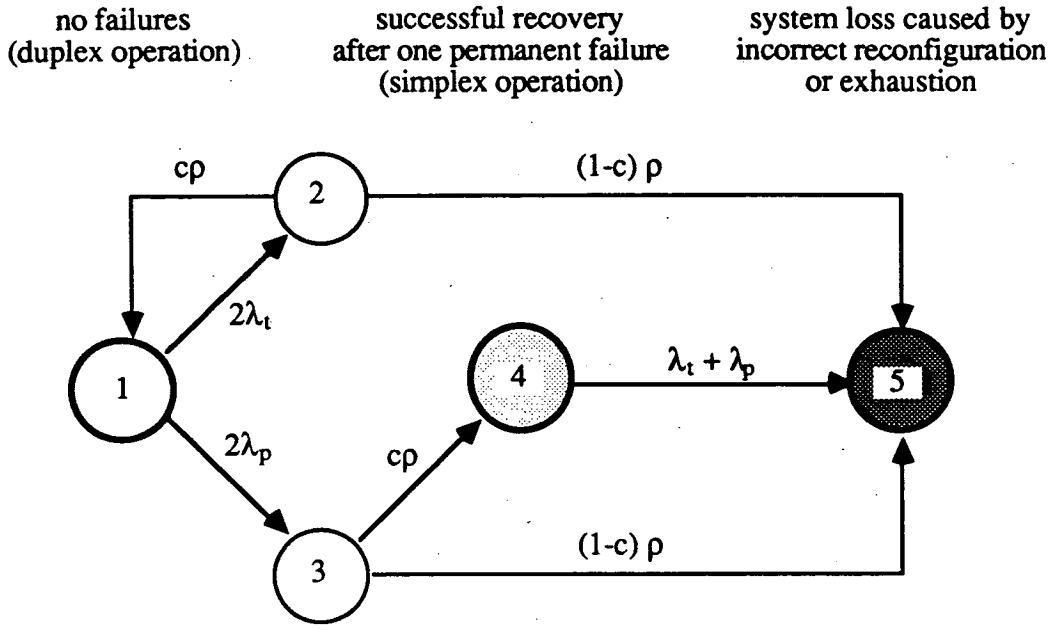

Figure B-4 Markov Model of a Dual FTP

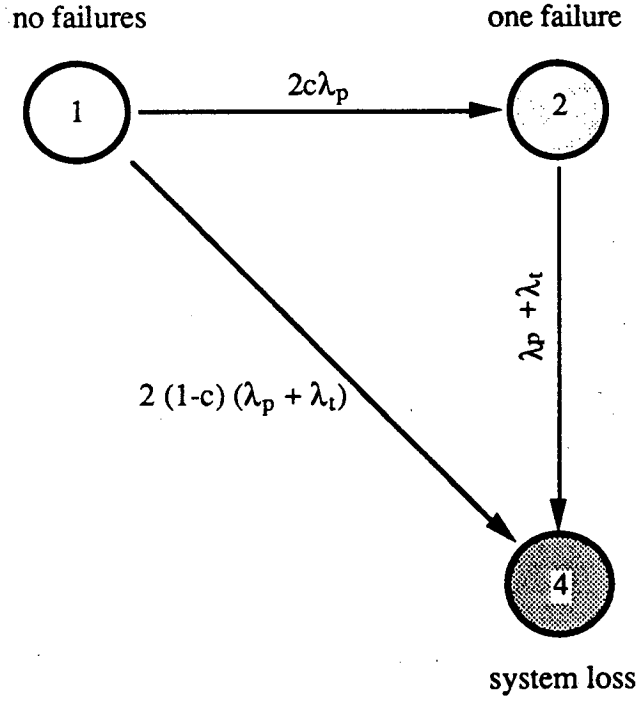

Figure B-5 Reduced Markov Model of a Dual FTP

It is interesting to note the expression of the *equivalent* dual coverage for this example. From Figure B-5 and according to equations (B-7), this effective coverage is given by:

$$c^d = \frac{2c\lambda_p}{2(1-c)\lambda + 2c\lambda_p} = \frac{c}{1 + (1-c)\dfrac{\lambda_t}{\lambda_p}} \quad \text{(B-15)}$$

The effective coverage is equal to the self-test coverage when the transient failures are disregarded. It decreases monotonically as the ratio of transient - to - permanent failures increases. This result is quite general, in spite of the simple model used to illustrate it.

B.6 Conclusions

A methodology enabling a systematic reduction of a complex FTP model has been developed. The technique relies solely on an examination of the structure of the transition matrix, with no eigenvalue analysis and coordinate transformation necessary. The reduced model captures all the relevant features of the detailed model and represents an excellent approximation. The simplicity of the reduced model allows a fully analytical solution, which is extremely effective especially when extensive trade studies are required.

The technique is illustrated on a simple but instructive example, for which an analytical solution is possible for both the exact and the reduced models. The high quality of the approximation is clearly shown. In addition, the crucial impact of the transient failures on the success of the reconfiguration process is demonstrated.

The methodology and the results presented herein provide considerable insight regarding the difficulties and the subtleties involved in the rigorous reliability and performance analysis of an FTP.

NASA National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-187554		

4. Title and Subtitle	5. Report Date
Advanced Information Processing System for Advanced Launch System: Avionics Architecture Synthesis	September 1991
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Jaynarayan H. Lala, Richard E. Harper, Kenneth R. Jaskowiak, Gene Rosch, Linda S. Alger, Andrei L. Schor	
	10. Work Unit No. 506-46-21-56

9. Performing Organization Name and Address	11. Contract or Grant No.
The Charles Stark Draper Laboratory, Inc. Cambridge, MA 02139	NAS1-18565
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	Contractor Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Final Report
Technical Monitor: Felix L. Pitts

16. Abstract

The Advanced Information Processing System (AIPS) is a fault-tolerant distributed computer system architecture which has been developed to meet the real time computational needs of advanced aerospace vehicles. One such vehicle is the Advanced Launch System (ALS) being developed jointly by the National Aeronautics and Space Administration and the Department of Defense to launch heavy payloads into low earth orbit at one tenth the cost (per pound of payload) of the current launch vehicles. An avionics architecture that utilizes the AIPS hardware and software building blocks has been synthesized for ALS. This report describes the AIPS for ALS architecture synthesis process starting with the ALS mission requirements and ending with an analysis of the candidate ALS avionics architecture.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Advanced Launch System Fault-Tolerant Avionics Architecture Reliability Redundant Digital Computers Distributed Processors	Unclassified-Unlimited Subject Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	158	