JPL PUBLICATION 82-1

A Study of the Selection of Microcomputer Architectures to Automate Planetary Spacecraft Power Systems

A. Nauda
Bucknell University
Lewisburg, Pennsylvania

January 1, 1982

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

JPL PUBLICATION 82-1

A Study of the Selection of Microcomputer Architectures to Automate Planetary Spacecraft Power Systems

A. Nauda
Bucknell University
Lewisburg, Pennsylvania

January 1, 1982

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

ABSTRACT

An advanced development program, conducted at the Jet Propulsion Laboratory, to develop the technology for autonomous operation of planetary spacecraft power systems, initiated a study to develop a methodology for selecting an optimum microcomputer architecture.

Unique to most applications of microcomputers, performance requirements such as throughput speed and data handling capacity, are not as significant to autonomous operation of a spacecraft power system as they are to more common applications such as signal processing and data manipulation. Planetary spacecraft power systems, however, are complex in terms of the number of different functions performed. Spacecraft power systems are also in a unique class, on which the total mission is dependent; therefore, reliability and fault tolerance are primary requirements.

Various microcomputer system architectures are analyzed to determine their application to spacecraft power systems.

Of the many microcomputer system architectures analyzed and discussed, no dominant system topology, applicable to automating spacecraft power systems, emerged. Indeed, there exists no standardized formula or common set of guidelines which will provide an optimum configuration for a given set of specifications.

Future work is shown to be necessary to develop performance and reliability models of alternate microcomputer architectures as a methodology for optimizing system design.

ACKNOWLEDGEMENT

The effort for this report was supported by the American Society of Engineering Education - NASA Summer Faculty Research Program.

CONTENTS

1. INTRODUCTION ---------- 1-1

2. POWER SYSTEM PERFORMANCE EVALUATION AND CONTROL REQUIREMENTS ---------- 2-1

3. FACTORS AFFECTING SELECTION OF COMPUTER ARCHITECTURE ---------- 3-1

3.1 TYPE OF COMPUTER ---------- 3-1

3.2 PERFORMANCE ---------- 3-1

3.3 RELIABILITY ---------- 3-2

3.4 OTHER CONSIDERATIONS ---------- 3-3
3.4.1 Ease of Development ---------- 3-3
3.4.2 Survivability ---------- 3-3
3.4.3 Flexibility ---------- 3-3

3.5 BOTTOM LINE ---------- 3-3

4. POSSIBLE MULTICOMPUTER ARCHITECTURES ---------- 4-1

4.1 COMPLETE INTERCONNECTION ---------- 4-1

4.2 PACKET SWITCHED NETWORK ---------- 4-2

4.3 REGULAR NETWORK ---------- 4-2

4.4 IRREGULAR NETWORK ---------- 4-3

4.5 HIERARCHY ---------- 4-3

4.6 LOOP OR RING ---------- 4-4

4.7 GLOBAL BUS ---------- 4-4

4.8 STAR ---------- 4-5

4.9 LOOP WITH SWITCH ---------- 4-5

4.10 BUS WINDOW ---------- 4-6

4.11 BUS WITH SWITCH ---------- 4-6

4.12 SHARED MEMORY ---------- 4-6

5. MULTICOMPUTER DESIGN METHODOLOGY ---------- 5-1

5.1 CHOICE OF ARCHITECTURE ---------- 5-1
5.1.1 Problem Definition (or Process Identification) ----- 5-1
5.1.2 Problem Decomposition ---------- 5-1
5.1.3 Process Interaction ---------- 5-2
5.1.4 Performance Requirements ---------- 5-2
5.1.5 Choice of System Architecture ---------- 5-2

6. HIERARCHICAL CONFIGURATION TECHNOLOGY ---------- 6-1

7. IMPLEMENTING FAULT TOLERANCE ---------- 7-1

8. CONCLUSIONS/RECOMMENDATIONS ---------- 8-1

9. REFERENCES ---------- 9-1

APPENDIX
A-1 Definition of Reliability ---------- A-1
A-2 Availability ---------- A-2
A-3 Reliability of Interconnected Components ---------- A-3
A-4 Effect of Redundancy on MTBF ---------- A-3
A-5 Fault Tolerance ---------- A-4

FIGURES

5-1 General Methodology for Choosing System Architecture Based on Data Transfer ---------- 5-3

6-1 System Reliability Increased with Redundant Control and Spare Local Computers ---------- 6-2

A-1 Typical Bathtub Curve of Failure Rate Versus Time ---------- A-2

TABLES

2-1 Typical Power System Functions and Their Computational Requirements ---------- 2-2

2-2 Estimated Power System Commands and Measurements for Autonomous Operation ---------- 2-3

A-1 Reliability of Series and Parallel Connections of Elements with Exponential Reliability Distributions ------ A-4

A-2 MTBF of Redundant Computer Configurations ---------- A-5

SECTION 1

INTRODUCTION

The trend of onboard computational capability, to satisfy more demanding mission requirements, has increased dramatically over the past 10 years. Ultra-reliable computer architectures are necessary for data acquisition and real time control. The purpose of this study is to investigate microcomputer system architectures with particular application to developing a design methodology for spacecraft power systems. Initial study effort has focused on two issues:

(1) Examination of several microcomputer architectures which may be suitable for spacecraft power system monitoring and control.

(2) Investigation of currently available redundancy/fault tolerance techniques.

Relatively speaking, it is easy to design a computer system, but it is very difficult to design a system that is optimized for a given set of requirements. In other words, techniques for logic design and system programming are fairly well understood, and there are also a number of techniques for analyzing the performance of a computer system. But designing a system that will perform well in a specific application area is a very intuitive undertaking. A good deal of experimentation is usually involved. This report examines possible design approaches, the trade-offs involved, and points out factors which affect the choice of computer system architecture for a spacecraft power system.

SECTION 2

POWER SYSTEM PERFORMANCE EVALUATION AND CONTROL REQUIREMENTS

To establish a basis for the performance requirements of a power system computer network, typical power system functions and estimated timing requirements were reviewed and are summarized in this section.

Current spacecraft power systems typically perform the following functions:

(1) Load switching

(2) Power processing

(3) Fault detection and correction

(4) Battery charging

(5) Battery reconditioning

The implementation and priority of the above functions may change subject to mission requirements and type of energy source.

Table 2-1 is a list of functions which have been identified (Ref. 1) as candidates for autonomous control on future planetary power systems.

The magnitude of the number of measurements and control commands necessary for autonomous monitoring and control of a planetary spacecraft power system is shown in Table 2-2.

Table 2-1. Typical Power System Functions and Their Computational Requirements

Power System Functions	Estimated Time Requirement	Computational Assessment
1. Fault Detection and Correction	10-ms response	Simple logical processing
2. Command Processing	1-ms decode time	Moderate logical processing
3. Relay Status Monitoring	As required	Simple logical processing
4. Relay Control	10-ms	Simple logical processing
5. Data Acquisition, Processing and Storage	All parameters every 100-ms	Moderate logical processing
6. System monitoring and diagnosis	1 s	Moderate logical processing
7. Subassembly Monitoring and Diagnosis	As required	Moderate logical processing
8. Load Sequencing and Control	100-ms response	Moderate logical processing
9. Load Equipment Monitoring and Diagnosis	100-ms response	Moderate mathematical processing
10. Power Capability and Margin Management	1 to 10 s	Complex mathematical processing

Table 2-2. Estimated Power System Commands and Measurements for Autonomous Operation

Subassembly	Analog Measurements	Relay Commands
Battery Electronics	8	14
Solar-Array Electronics	12	4
Power Control	13	6
Battery Charger 1	1	5
Battery Charger 2	1	5
Boost Regulator 1	2	0
30-Vdc Converter	2	4
Power Distribution	34	42
Battery	30	56
Total	103	136

Although this table summarizes a particular spacecraft power system design incorporating autonomous functions, it can be seen that hundreds of measurements and control commands are typically necessary (Refs. 2,3,4).

SECTION 3

FACTORS AFFECTING SELECTION OF COMPUTER ARCHITECTURE

The data processing requirements (speed, timing, and computational complexity) for a spacecraft power system are not as demanding as those for some other subsystems, (e.g. Attitude Control). However, the criticality of the power system to mission success dictates that the data processing and control functions be highly reliable. Some of the key factors which affect the selection of architecture are discussed below.

3.1 TYPE OF COMPUTER

Unless there are clear indications that a particular microcomputer is required (e.g. by throughput requirements), designers usually select one with which they are familiar and/or one whose development system is available.

3.2 PERFORMANCE

The first decision to be made is whether one computer can meet the throughput requirements demanded by the system. Designers find it difficult to generalize on the procedures and thought processes they use to make this decision. Some general comments follow.

Usually a synchronous executive is written where measurements and monitoring tasks are cycled through at a specified rate chosen by the rate at which the central computer needs data and/or the rate at which critical load management or survivability actions need to be taken. Any necessary calculations and logical decisions, along with subsequent corrective actions, must be processed well within the synchronous executive's cycle. Therefore, in deciding whether one computer is sufficient, the following steps may be taken:

(1) Translate the functional requirements of the system into precise specifications of required tasks.

(2) Develop efficient algorithms to accomplish those tasks.

(3) Examine the timing and memory requirements for the algorithms; this task includes:
   (a) Estimate lines of code necessary
   (b) Estimate time required for execution
   (c) Estimate memory requirements, including access time.

(4) Establish time line - or sequence of tasks - listing all tasks in the order (and at the frequency) which they must be executed for a sample executive clock cycle. This includes recording measurements, simultaneous checks for load faults, interrupts to eliminate faults, or control signals to relays within critical times for effective problem resolution, etc.

(5) Examine latency restrictions (i.e., Once an error is detected, how long can one wait until a corrective action must be taken) and look for ways to overlap tasks or condense the time line.

(6) Add a "comfortable" time margin for software overhead (executive control, computer communication software, etc). Some designers choose as much as 50 percent margin.

The result of this procedure is an estimate of the time required to process the necessary tasks with the chosen computer. If all tasks cannot be executed well within the synchronous executive's cycle, then a faster computer or a multicomputer system should be considered.

The throughput requirements of a spacecraft power system can generally be met by efficient use of a single computer. If more speed is necessary, a faster computer or more computationally efficient algorithms may be chosen. The decision of going to a multicomputer or distributed network is usually made for other reasons. These are discussed in subsections 3.3 through 3.5.

3.3 RELIABILITY

Computers for use in planetary spacecraft power systems will perform functions which are computation-critical and which require long life. The equipment cannot be maintained, repair is impossible, yet reliable operation is demanded for the duration of the mission (5 or more years). This imposes the most stringent fault tolerance requirements in a real-time environment to avoid jeopardizing the success of the mission. These stringent requirements can be met by a reliable version of a single computer system. (A fault tolerant uniprocessor - the self-testing and repairing (STAR) computer was developed at JPL (Ref. 5)). To achieve reliability, such a system usually requires redundancy and, if graceful degradation is desired, a fine partitioning of the computer system into programmable replacement modules.

Partitioning can occur at different levels. With many machines in the past, partitioning was at the subprocessor level. With current technology, it makes little sense to partition a system below the level of a microcomputer. Thus due to very large scale integration (VLSI) technology, the partitioning concept has evolved into an architecture in which individual computers make up the replaceable system modules. Such a distributed computer network is well suited to applications like the power system where the computing system controls a number of relatively autonomous (although possibly functionally interdependent) subelements (i.e., inverter control and load management).

Thus, even if a single computer can handle the throughput requirements, reliability goals may require a distributed network, particularly if the reliability goals include graceful degradation.

3.4 OTHER CONSIDERATIONS

By embedding small, dedicated processors into functionally partitioned subsystems, several advantages result.

3.4.1 Ease of Development

Subsystem designers, who are most familiar with their own equipment can develop independent software necessary for its peculiar control and/or fault diagnosis if a multicomputer approach is taken (i.e., software design is modularized). Also, if local subsystems are independent, local control frequently results in simpler higher level control and data handling programs.

3.4.2 Survivability

Graceful degradation is possible in distributed multicomputer systems because the total system can be designed to continue to operate despite individual computer failures.

3.4.3 Flexibility

As future spacecraft systems change in size and complexity, system redesign is simplified by incrementally deleting or adding microcomputers and modifying software.

These benefits are, however, accompanied by some disadvantages. The designer is faced with increased software complexity. Distributed systems typically require their own executives which must communicate with other executives in the system (or other systems). This also means the distributed system is more dependent on computer communication technology. In addition, overall diagnostic software development is usually more difficult in multicomputer systems.

3.5 BOTTOM LINE

The choice of using a single computer or a multicomputer network is a function of long-term design objectives. If the power system is to be custom redesigned for each mission, then a practical engineering approach will probably result in a single microcomputer with a standby unit to avoid a single point failure. However, if a general power system design is desired - one which is flexible and can be "programmed" to ease development efforts for different missions, then a distributed multicomputer approach seems more appropriate.

SECTION 4

POSSIBLE MULTICOMPUTER ARCHITECTURES

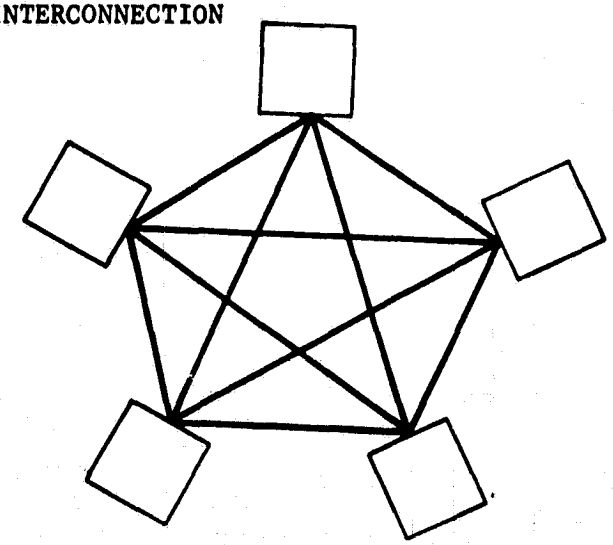
Six basic multicomputer architecture types (or interconnect technologies) are described in the literature:

(1) Shared memory
(2) Shared bus
(3) Loop systems
(4) Star configurations
(5) Hierarchical configurations
(6) Point to point interconnections

Each topology has certain attributes that affect its suitability for power system applications. These attributes are related to cost, reliability, performance (responsiveness, speed, throughput), ease of development, modularity, reconfigurability and survivability, and such physical parameters as volume, weight, and power consumption.

Some of the more common interconnect technologies (with a few variations of the basic six) are briefly compared based on selected design attributes in the discussion which follows. The architectures are discussed in the order of decreasing reliability based on vulnerability to a single component failure.

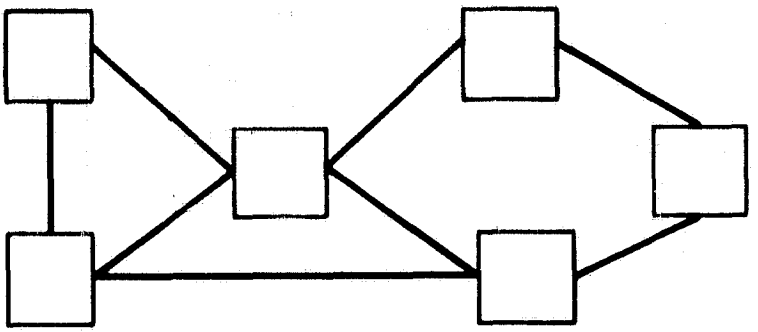
4.1 COMPLETE INTERCONNECTION



The completely interconnected architecture is conceptually the simplest design. Each processor is connected by a dedicated path to every other processor. Communications software becomes extremely complicated as the number of processors increases.

Cost: High - function of the number of micros in the system.
Modularity: Fair - number of ports on each micro is N-1.
Reliability: Most reliable-only local problem if micro fails. Redundant paths alleviate single link failures.

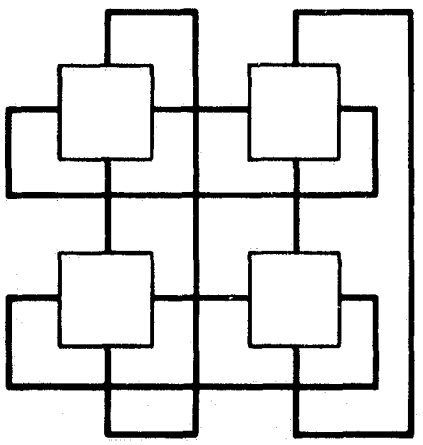
4.2 PACKET SWITCHED NETWORK



Messages are broken into packets and transmitted by way of available nodes. At least two paths exist between any two computers in the system.

Cost:	High - each node requires routing control.
Modularity:	Good.
Reliability:	Only local problem if a computer fails.

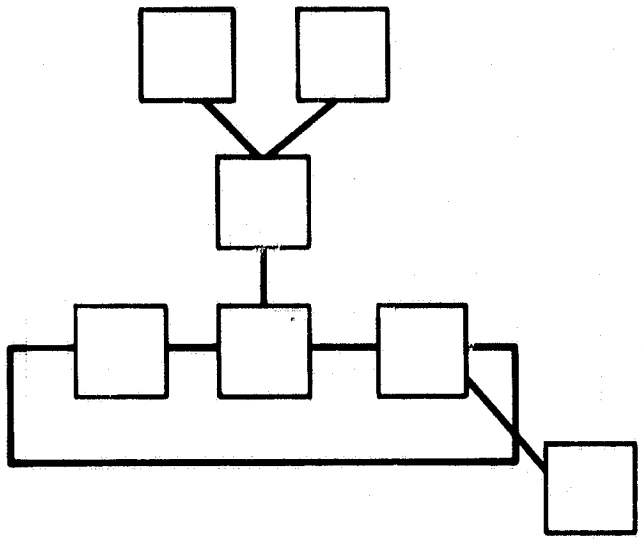
4.3 REGULAR NETWORK



Every computer is connected to its own neighbor and another computer above and below it. The network gets complicated if there are very many computers. The "tree" is a hierarchically structured variation with any computer able to communicate with its superior and its subordinates as well as its two neighbors.

Cost:	High - function of number of computers in system.
Modularity:	Poor.
Reliability:	Only local problem if computer fails. Redundant paths eliminate single connect failures.

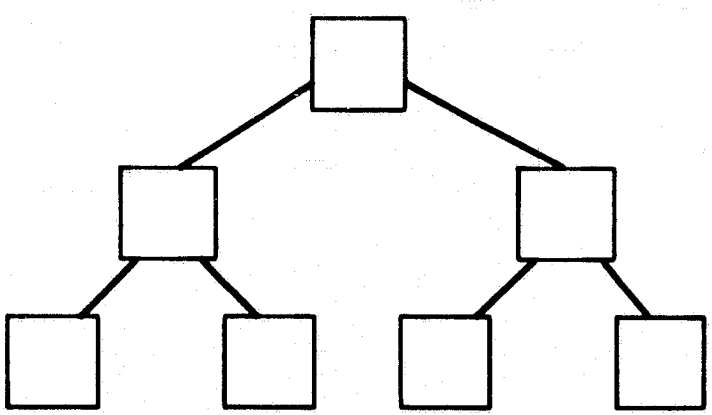
4.4 IRREGULAR NETWORK



The irregular network configuration has no consistent neighbor relationships. It is common in geographically dispersed networks where communication links control the design.

Cost:	Medium - function of distance between computers.
Modularity:	Fair.
Reliability:	Partial redundancy for link failure.

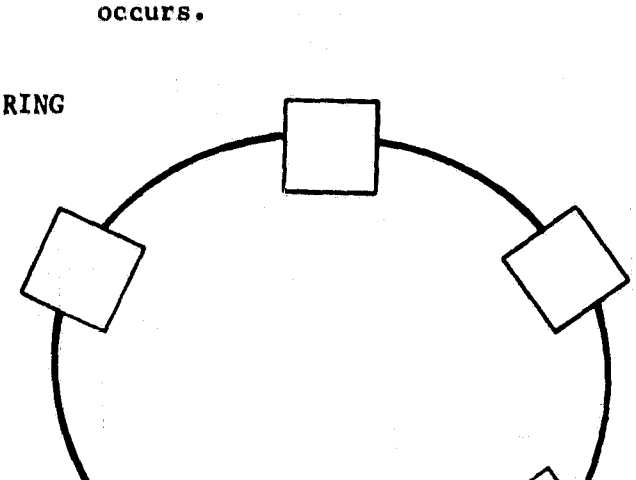
4.5 HIERARCHY



The heirarchy configuration is used in process control and data acquisition applications. The capabilities are specialized at lower levels and more general purpose at the top.

Cost:	Medium - function of distance between computers.
Modularity:	Good.
Reliability:	Systems operability reduced with single point failure, more serious the higher up the failure occurs.

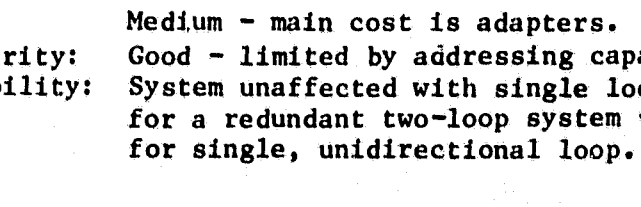
4.6 LOOP OR RING



Loop architecture evolved from the data communication environment. In this configuration, each computer is connected to two neighboring computers. The data can flow in both directions, but circulating traffic in one direction is less complicated.

Cost:	Medium - main cost is adapters.
Modularity:	Good - limited by addressing capability.
Reliability:	System unaffected with single loop failures for a redundant two-loop system - catastrophic for single, unidirectional loop.

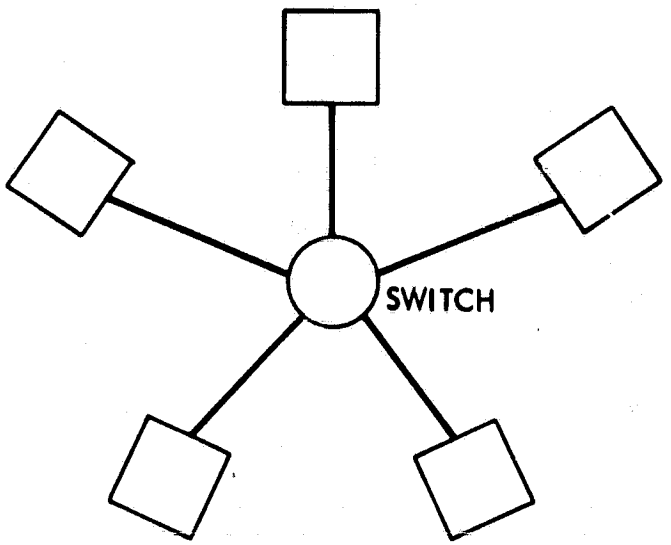
4.7 GLOBAL BUS



The use of a common or global bus requires some allocation scheme for sending messages from one computer to another.

Cost: Medium - main cost is bus adapters.
Modularity: Good.
Reliability: Only local problem if a computer fails - catastrophic with bus failure.

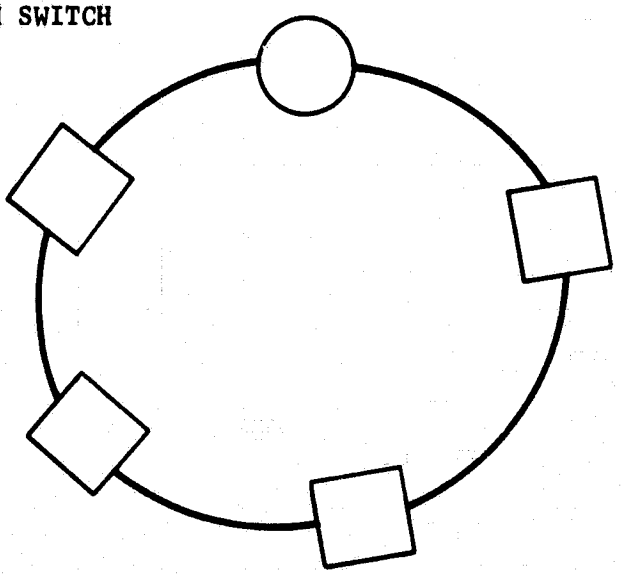
4.8 STAR



The star configuration has a central switching resource. Each computer is connected to the central switch. Traffic is in both directions.

Cost: Medium to Low - major cost item is switch.
Modularity: Good - until switch saturates.
Reliability: Only local problem if a computer fails - catastrophic if switch fails. Switch is possibly less reliable than bus or loop.

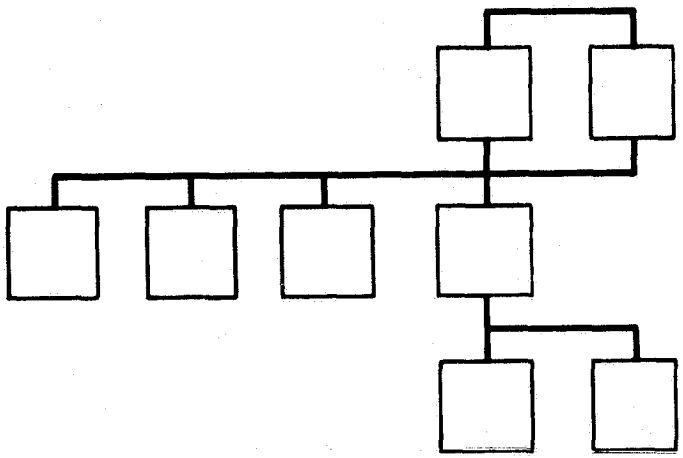
4.9 LOOP WITH SWITCH



This refinement of the loop provides a switching element that removes messages from the loop, maps their addresses, and replaces them on the loop properly addressed to their intended destination.

Cost:	Medium - main cost is switch.
Modularity:	Good-Fair, until switch saturates.
Reliability:	Catastrophic if either switch or loop fails.

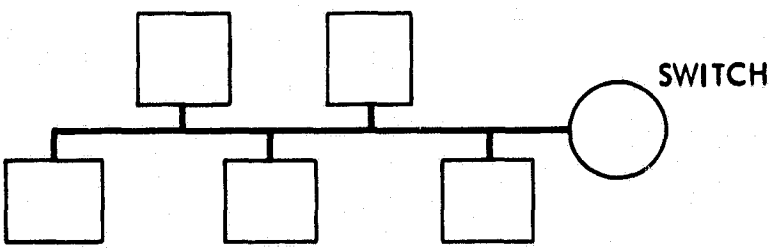
4.10 BUS WINDOW



The bus window configuration has more than one switch. Messages may be transmitted on the path they are received or on another. The switches provide "windows" for passing messages between buses.

Cost:	Low - main cost is switch.
Modularity:	Poor.
Reliability:	Serious contention problems. Partial system failure if switch or bus fails.

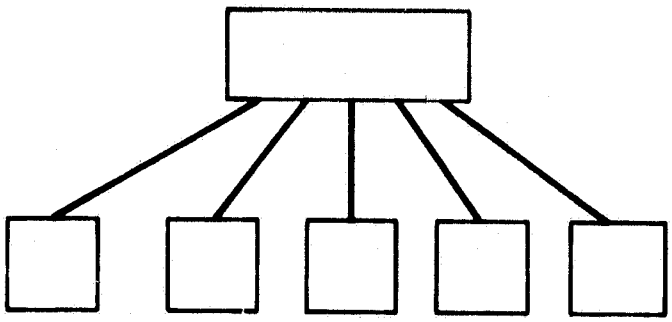
4.11 BUS WITH SWITCH



This is more like the global bus, since each computer is connected to the central switch and traffic flows from the originating computer to the switch, and from the switch to the destination computer. The computers share the path (bus) to share access to the switch.

Cost:	Low - main cost item is the switch.
Modularity:	Good - Fair, until switch saturates.
Reliability:	Catastrophic if bus or switch fails.

4.12 SHARED MEMORY



The most common way to interconnect computer systems is to communicate by leaving messages for one another in a commonly accessible memory. The key characteristic is that the the memory is used as a data path as well as storage.

Cost:	Low - main cost is multiported memory.
Modularity:	Poor - limited to number of memory ports.
Reliability:	Least reliable - catastrophic if memory fails.

In the design of multicomputer systems, the consideration of all possible interconnect technologies may not be necessary. Practical aspects of specific applications frequently lead to a limited choice of architectures. A methodology for making such decisions is discussed in the next section.

SECTION 5

MULTICOMPUTER DESIGN METHODOLOGY

Since the design of distributed microcomputer systems is an art dependent on experience, there exists no standardized formula to provide the optimal configuration. Key attributes in a typical data acquisition and control system are performance, reliability, availability, fault tolerance, and failure reconfigurability. Other attributes, but slightly less important, are life-cycle cost and modularity/growth. System design becomes a trade-off analysis weighing the relative contributions of alternate architectures to maximize the important attributes of the system. Although a methodology for an optimum universal design is virtually impossible, there are some general statements which can be made concerning the choice of microcomputer architecture and the subsequent implementation of fault tolerance.

5.1 CHOICE OF ARCHITECTURE

The design of a distributed microcomputer system is primarily a function of the experience of the designer. It is usually approached in a sequential fashion with the following considerations.

5.1.1 Problem Definition (or Process Identification)

It is not necessarily clear from the functional requirements what the consequences are of the specific tasks required by the system. It is necessary to determine as precisely as possible what is to be automated. This should also include the number and type of measurements, the number and type of controlling functions and signals, the relative criticality of each of the above, and the timing requirements. It is also necessary to identify the specific communication requirements in order to interface with other computers.

5.1.2 Problem Decomposition

This is a functional breakdown of the system requirements. The value of identifying major functional groups is that the designer will develop an understanding of the major subtasks to be performed by the system with a qualitative feel for the workload imposed by each function. One should identify critical functions, which may demand ultra-reliability, and those which may be allowed to gracefully degrade. This usually leads to allocating separate processor memory resources to handle different functional groups.

Some schemes have been developed to aid the designer with this task. Weitzman (Ref. 6) uses a structured set of data-flow primatives which are arranged in process architecture trees. This phase also embodies the experience of the designer.

5.1.3 Process Interaction

One would like to obtain and formulate adequate quantitative knowledge of the information flow between functions. Analytical tools available to develop an understanding of the various interrelationships between subprocesses include state exchange diagrams, process interaction diagrams, and N^2 charts (Ref. 6).

5.1.4 Performance Requirements

One must define as specifically as possible the system's physical performance requirements. These include

(1) Sizing of tasks

(2) Defining relationships between tasks

(3) System control for information movement and processing which involves identifying:

  (a) Information transfer strategy

  (b) Transfer control method

  (c) Transfer path structure

  (d) Shared and dedicated system resources.

5.1.5 Choice of System Architecture

In the selection of appropriate hardware and software elements and system structure, the consideration of all possible architectures may not be necessary. The pros and cons of information transfer strategy, control methods, and path structure may provide an indication of the most attractive solutions. Figure 5-1 indicates a general methodology for such choices.

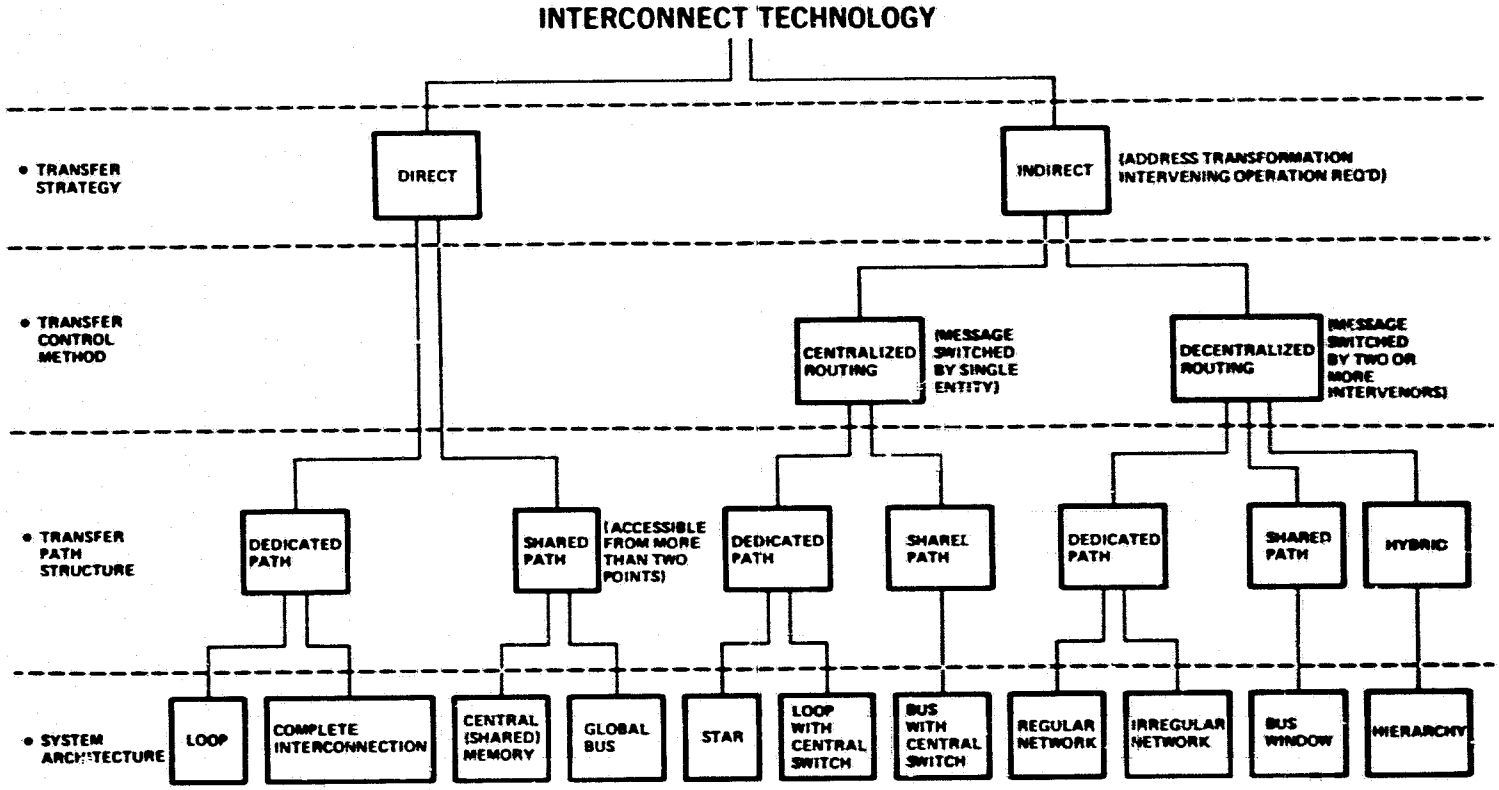


Figure 5-1. General Methodology for Choosing System Architecture Based on Data Transfer (Ref. 6)

SECTION 6

HIERARCHICAL CONFIGURATION TECHNOLOGY

Of all the interconnect technologies discussed, data acquisition and process control systems have most frequently been based on hierarchical architectures. This does not mean that loop or bus systems should be automatically ruled out. However, some design considerations for hierarchical structures should be mentioned.

A hierarchical configuration, as its name implies, consists of a tree structure of computers. In general, the capability of the computer increases as the top of the pyramid is reached. This is often due to practical rather than theoretical reasons. In a manner similar to a corporate organizational structure, the capabilities at the base are generally applications-dependent, with a special-purpose capability, dedicated to performing well-defined, specialized tasks, whereas the top of the organization has a more general-purpose capability, controlling and coordinating the entire system. In such a system, computer functions are usually distributed. The tedious repetitive functions and algorithms, such as data collection and reduction, are handled at the lowest levels, whereas data processing and command execution (control) are performed at the top. Typically, shared data bases are also stored at the top rather than distributed throughout the system.

The partitioning of overall system processing loads into approximately equal-size processing segments can make it possible to use one type and size computer in the system (at least in the lowest level). This has an advantage in that, since all computers are identical, the system may be implemented in such a way that a standby unit is always available and can be switched online to perform the tasks of any other computer in the system (should one become inoperative). Thus reliability can be improved without complete redundancy.

From a reliability standpoint, if a failure occurs in the computer located at the top of the pyramid, total system control is lost. This requires a redundancy along with doubling all communications paths at the top. An example of such a structure is shown in Figure 6-1. Thus, the addition of redundancy greatly increases the complexity of the system as well as software overhead. (Doubling of hardware does not necessarily double the reliability of the system. See the reliability discussion in the Appendix.)

Whether redundancy is used or not, hierarchical microcomputer systems should be designed to be capable of operating in a degraded mode. The loss of a single computer should result in the absolute minimum amount of information being lost and should not cause the entire system to cease functioning. To ensure operation in a degraded mode, the following design features should be incorporated:

(1) When a low level computer fails, all of its process outputs should be frozen and transfers should automatically be made to backup control by reconfiguring the system (e.g. by switching to a spare).

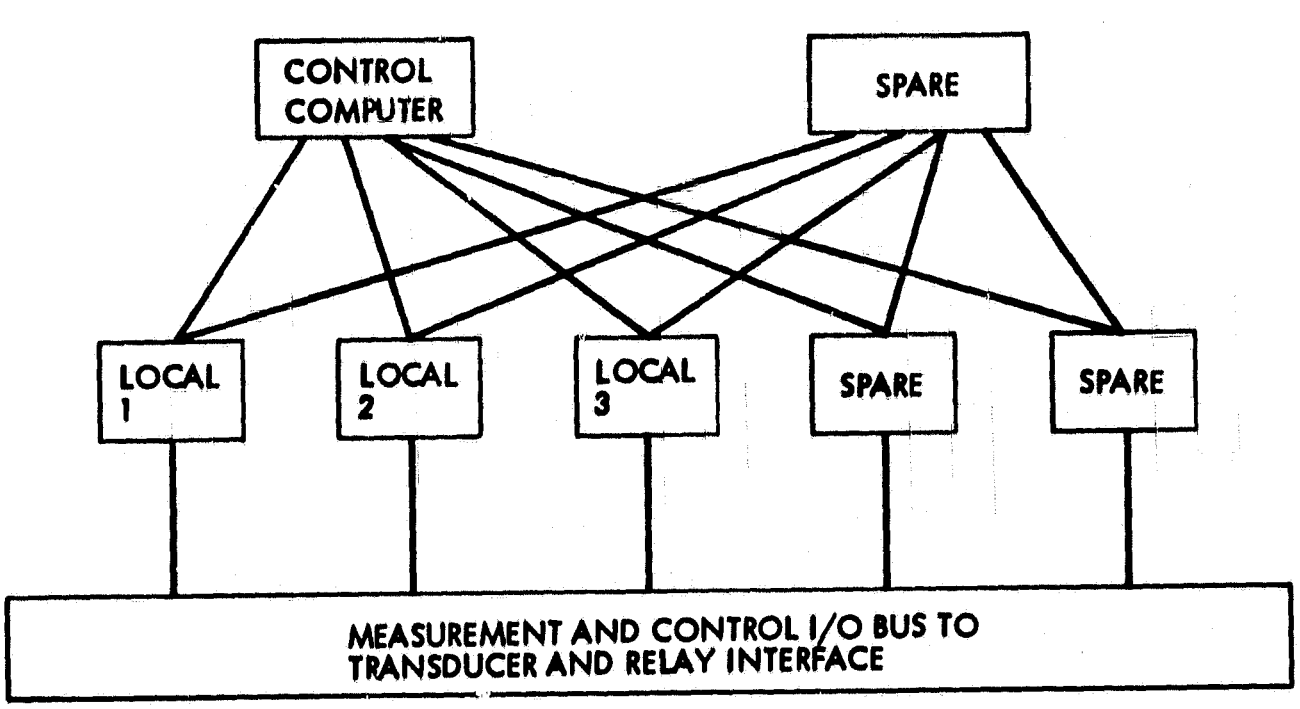


Figure 6-1. System Reliability Increased With Redundant Control and Spare Local Computers

(2) Each computer should be able to store, for a reasonable period of time, information destined for another computer. This information could be transmitted when the target computer becomes operational again.

(3) No computer should depend solely on information arriving from another computer. Crucial programs should always exist at the site where they are needed. Mathematical results or measurements should be replacements for old results (calibrations for example), and old results should continue to be used until new ones become available.

SECTION 7

IMPLEMENTING FAULT TOLERANCE

Recent literature, based on research, analysis, and experience accumulated over the past decade, indicates definite guidelines exist for the implementation of tolerance of physical faults in digital systems (Ref. 7). These are summarized as follows:

(1) Devise a fully satisfactory system according to given performance specifications, assuming fault-free conditions.

(2) Specify reliability goals for the system.

  (a) Explicitly identify classes of faults that are to be tolerated. (This usually limits the faults to less than all possible things that can go wrong.)

  (b) Specify quantitative reliability goals for each fault set.

  (c) Postulate a method to evaluate actual reliability.

(3) Select and incorporate fault-detection algorithms. This usually leads to the addition of new elements or software accomplishing parity checks, self-test programs, etc.

(4) Devise recovery algorithms which are evoked by signals from fault-detection algorithms and whose goal is to return the system to some level of normal operation, or to shut part of it down safely. Recovery consists of all actions that take place after the fault is detected. These may include:

  (a) Error correction

  (b) Fault location

  (c) Exclusion or replacement of failed parts

  (d) Recording of actions taken

  (e) Restart of normal operation.

  This may involve addition of spare computers or bus elements, or increase in memory size, etc.

  A special form of recovery results from the use of fault-masking techniques in which redundant elements instantly conceal the effect of faults without a separate fault detection being required.

(5) Evaluation is performed by means of modeling and/or simulation. Reliability prediction is compared to that of the original system. Degradation of performance is noted for each fault set.

(6) Refinement of design is performed. Initial evaluation is likely to demonstrate that various subsystems display unequal reliability contributions to the total system reliability.

Hardware implementation of fault tolerance to physical faults has led to several system design concepts. Triple modular redundancy (TMR), standby, hybrid, self-paging, and duplex redundancy techniques are some of the schemes discussed in the literature (Refs. 8-12).

Physical faults are not the only events that disrupt the specified behavior of digital systems. Many events can be traced back to some imperfections in the software that had remained unidentified. At least two approaches to software fault-tolerance design have appeared in the literature. They are the recovery block (Ref. 13) and N-version programming (Ref. 14). Both methods use some redundancies analogous to successful fault-tolerance approaches to physical faults.

SECTION 8

CONCLUSIONS/RECOMMENDATIONS

The choice between a single computer or multicomputer architectures is determined primarily by the system considerations of performance, reliability, and flexibility. For most power systems, a single microcomputer will handle throughput requirements. Multicomputer configurations may be chosen for the additional considerations of flexibility and ease of modification. In addition, distributed control of a multicomputer system may provide the benefits of graceful degradation and considerable fault tolerance. Hierarchical configurations are most frequently used in similar applications and appear to be an adequate compromise between maximizing fault tolerance and flexibility. Although not the only scheme possible, these systems can be made reliable with redundancies and/or spares, yet permit modular design for ease of development and modification.

A general design methodology is presented for both single computer and multicomputer systems. For either approach, a combined hardware/software fault tolerant design has the most advantages. Hardware redundancies increase the reliability of the physical systems, but extra software efforts can provide more than a computer system with built in spares - that is continued computational and control capability.

It is recommended that the power system be standardized including bus characteristics, power processing equipment, data bus interfaces, battery cells, etc. The benefits of a distributed multicomputer system can be gained by implementing a reuseable power system design, incorporating sufficient flexibility for expansion to a wide range of missions. In such a system, modification of control functions or system reconfiguration can become a matter of software manipulation rather than major hardware change. This can also permit development of analytical methods to model the system's performance and reliability. These tools, in the form of computer programs, can then be used to optimally reconfigure the system to suit new mission requirements.

SECTION 9

REFERENCES

1. A.O. Bridgeforth, "Preliminary Evaluation of the Application of Automated Power Systems Management (APSM) to a Spacecraft Power System (VO75) (APSM/VO75), Jet Propulsion Laboratory internal report No. 715-8, October 1979.

2. D. Rennels, "Architectures for Fault-Tolerant Spacecraft Computers," Proceedings of IEEE, Vol. 66, No. 10, October 1978, pp. 1255-1268.

3. D. Green and E. Perry, "Microprocessor Control for Standardized Power Control Systems," Proceedings of Southeaston (Atlanta, Georgia), April 10-12, 1978, pp. 242-245.

4. Martin Marietta Corp., "Design and Development of APSM," Final Report No. MCR-79-540, July 1979.

5. A. Avizienis, et al., "The STAR (Self-Testing and Repair) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design," IEEE TRANSACTIONS on COMPUTERS, Vol. C-20, November 1971, pp. 1312-1321.

6. C. Weitzman, Distributed Micro/Minicomputer Systems, Prentice Hall, 1980, pp. 204-255.

7. A. Avizienis, "Fault Tolerance: The Survival Attribute of Digital Systems," Proceedings of the IEEE, Vol. 66, No. 10, October 1978, pp. 1109-1125.

8. J. Abraham and D. Siewiorek, "An Algorithm for the Accurate Reliability Evaluation of Triple Modular Redundancy Networks," IEEE Transactions on Computers, Vol. C-23, July 1974, pp. 682-692.

9. F. Mather and P. de Souza, "Reliability Modeling and Analysis of General Modular Redundant Systems," IEEE Transactions on Reliability, Vol. R-24, December 1975, pp. 296-299.

10. J. Losq, "A Highly Efficient Redundancy Scheme: Self-Purging Redundancy," IEEE Transactions on Computers, Vol. C-25, June 1976, pp. 269-278.

11. T. Arnold, "The Concept of Coverage and its Effect on the Reliability Model of a Repairable System," IEEE Transactions on Computers, Vol. C-22, March 1973, pp. 251-254.

12. Ying-Wah Ng and A. Avizienis, "A Reliability Model for Gracefully Degrading and Repairable Fault Tolerant Systems," Proceedings of the 7th Annual International Conference on Fault Tolerant Computing (1977), pp. 22-28.

APPENDIX

APPLICABLE NOTES ON RELIABILITY

A-1 Definition of Reliability

The definition of reliability commonly accepted for engineering applications is the characteristic of a component or system, expressed by a probability, that it will perform a required function under stated conditions for a specific period of time. Models are usually developed to calculate and compare reliability of alternate systems. Since multicomputer systems frequently are required to carry out more than one type of function (e.g., load management and battery conditioning), separate reliability models for each of these functions may be necessary to make the problem more tractable.

Several parameters may have a marked effect on the reliability of a given system. These include: environmental conditions (temperature, humidity, vibration, etc.), operating conditions (voltage, current, power dissipation).

When comparing alternate systems, the relative system reliability can be measured both quantitatively and qualitatively.

Quantitative measurements:

- Mean time between failures (MTBF)
  Usually specified in hours, this can be related to component reliability and type of redundancy.

- Mean time to repair (MTTR)
  Also in hours, this can be minimized with built-in redundancy, real-time self check, and diagnostics.

- Failure reconfiguration time
  A reliable system requires redundant paths and/or microprocessors that can be activated as soon as a failure is detected. The time to reconfigure may be critical to avoid system failure.

Qualitative measurements:

- Graceful degradation
  This is applications related. One cash register failing is not a great loss (except on Friday nights), but one part of a measurement system in a spacecraft may be. Computers must be connected in such a way as to minimize the effect of failure on the total system.

- Fault tolerance
  This attribute allows a system to function when some component fails. The level or depth of fault tolerance depends on the fault set and recovery procedures.

A-2 Availability

Availability is a term frequently used when discussing reliability, since it also is measured by the same parameters. Availability is defined as the percentage of time a microcomputer system is up (available). It may be expressed quantitatively as follows:

$$A = \frac{MTBF}{MTBF + MTTR}$$

From this equation, it can be seen that availability can be improved by increasing the MTBF and/or decreasing the MTTR.

The ultimate in MTTR can be achieved by having spare units wired into the system either as hot or cold standbys. This combined with automatic fault-detection devices and an automatic reconfiguration capability that switches failed units out and backup units in, reduces MTTR to virtually zero.

Such fault tolerant design requires additional critical components, however, that are in turn subject to failure.

A-3. Reliability of Interconnected Components

The failure pattern of equipment placed in service can be categorized into three periods of operation, as illustrated in Figure A-1.

At the very beginning, any inherently weak parts that are the result of improper design, improper manufacture, or improper use usually fail fairly soon. The early failure rate decreases progressively and eventually

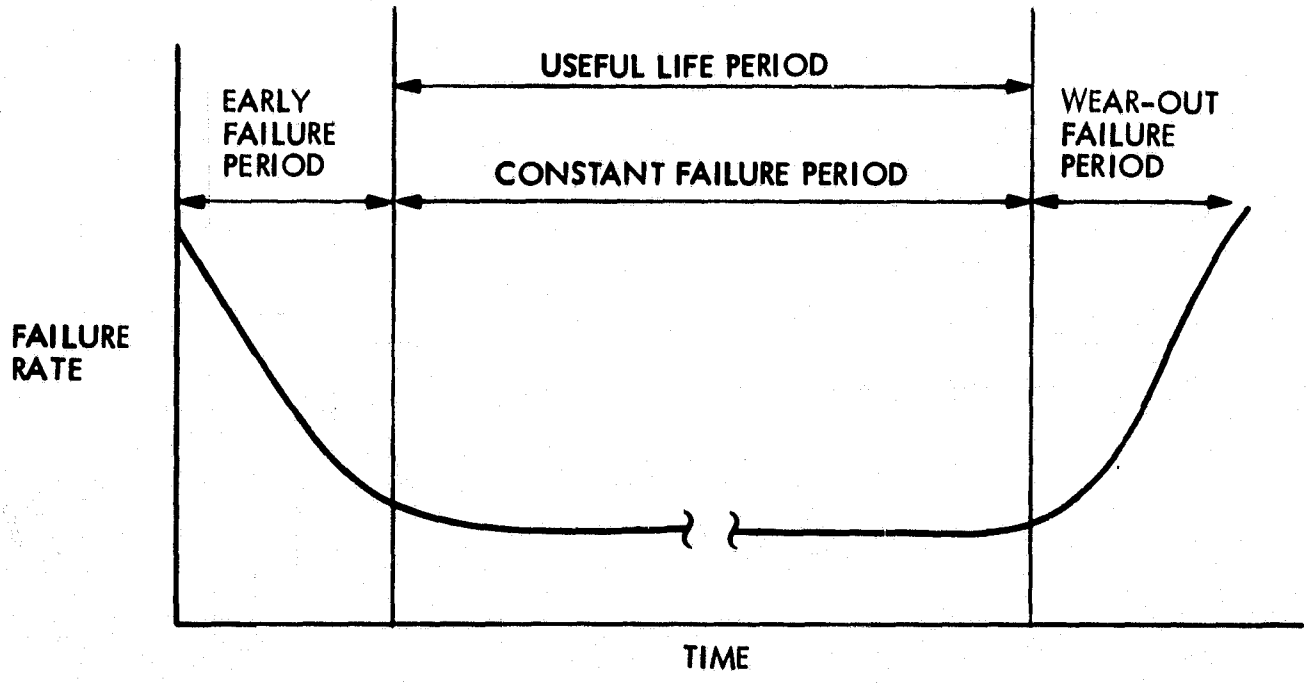


Figure A-1. Typical Bathtub Curve of Failure Rate Versus Time

levels off as the weak components are replaced (usually during tests under accelerated conditions). Spacecraft systems, which are non-repairable during missions should be operated for a period of time under varying conditions to ensure detection of early failures. After the early failures have been replaced, the components settle down to a long, relatively steady period at an approximately constant failure rate. The normal working life of a system occurs during this interval. In the wear-out period, the components rapidly deteriorate, and each component eventually wears out.

A reliability calculation may be made rather simply during the constant failure rate portion of the curve. The constant failure rate implies that the probability of failure is independent of age. A reliability function so characterized is the negative exponential distribution

$$R = e^{-\lambda t}$$

where

λ = failure rate, t = time.

It is assumed that at $t = 0$, all components are operational.

A physical system consists of many different types of components, each of which has a different instantaneous failure rate. The ultimate concern of the designer is the reliability of the total system.

Logically, the components are connected in either series or parallel (as with redundancies). The reliability of such interconnections of components (whose individual reliability functions are exponential) may be derived and is summarized in Table A-1.

It can be seen that the system reliability increases with the number of parallel paths and that it decreases with the number of units in series.

A-4. Effect of Redundancy on MTBF.

The overall reliability of a system may be improved by adding redundancy so that, if one unit fails, another is available to perform the necessary functions. There are active and standby types of redundancy. The parallel configuration discussed previously is an active type in which the redundant elements are continuously energized and used to perform the required circuit or system functions. In standby redundancy, the additional units are activated only when needed. The advantages of active and standby redundancy can be expressed in terms of the mean time between failures (MTBF). MTBF is a quantitative measure of reliability which may be expressed as the integral of the reliability function.

$$\text{MTBF} = \int_0^\infty R(t)dt = \int_0^\infty e^{-\lambda t}dt = \frac{1}{\lambda}$$

Table A-1. Reliability of Series and Parallel Connections of Elements with Exponential Reliability Distributions (Ref. 6)

Connection	Reliability
n series elements	$R = \prod_{i=1}^{n} R_i$
m parallel elements	$R = 1 - \prod_{i=1}^{m} (1 - R_i)$
m parallel paths with n series elements	$R = 1 - (1 - R^n)^m$

If the reliability functions of redundant computers have the same exponential form (i.e. identical failure rates), then the combined redundant system will have MTBF as given in Table A-2.

For the standby case, the spare unit remains unused until placed in service. The active redundant spare is used continuously and wears out along with the original. Thus, the MTBF of the standby configuration is twice the value of the one-unit configuration, whereas the active redundant pair (as calculated from the reliability of a parallel connection) has a smaller MTBF.

A-5. Fault Tolerance

Fault tolerance is the attribute of a digital system which makes it possible for a logic machine to continue with its specified tasks after the physical system suffers failures of its components. The implementation of fault tolerance is an approach to system design whose purpose is to increase reliability (or the probability that the system will function as designed).

Fault tolerance is the survival attribute of a logic machine because its purpose is to cause a return from error states back to a specified behavior, thus assuring the survival of the information processing system (Ref. 7).

The presence of fault tolerant features does not add any performance advantages during normal (fault-free) operation. On the contrary, fault-tolerance usually requires additional hardware and/or software that is redundant during normal operation and would be superfluous in a fault-free system.

Table A-2. MTBF of Redundant Computer Configurations

Configuration	MTBF
One unit, no redundancy	1/2
Two units, active redundancy	3/27
Two units, standby redundancy	2/7

To increase reliability, the only alternative to fault-tolerance is fault-avoidance, which requires the physical components and their assembly techniques to be perfect. As outlined previously, because of the constant rate of random failures (λ), the reliability of a system without redundancy is $R = e^{-\lambda t}$. The only way to increase system reliability is to force λ as close to zero as possible.

The exclusive use of fault-avoidance has two serious drawbacks:

- Cost of obtaining nearly perfect components rises very rapidly after failure rates have been reduced to threshold values that are characteristic of the physical parameters and manufacturing technology of the components.
- Since the system will cease proper operation upon the first failure or malfunction, manual maintenance is necessary.

Many systems have combined fault-avoidance and manual maintenance as a method to assure reliability (Ref. 7). This is not practical in space vehicles. There are strong reasons for the use of fault tolerance in spacecraft computer design:

- Initial investment in fault-tolerance can reduce the lifetime cost of the system.
- Space vehicles are placed in environments that do not allow access for manual maintenance.

Fault tolerant design involves implementing hardware and/or software redundancies, fault detection, and reconfiguration strategies. Typical steps in a recovery strategy (Ref. 8) are:

- Initial fault diagnosis
- Identification of faulty module
- Determine reconfiguration strategy

- o Perform reconfiguration
- o Condition new elements
- o Recover elapsed time
- o Rollback application programs

Methodology of implementing fault-tolerance is discussed in Section 7.