1989

Reliability analysis of real time computer architectures /

Bryan Robert Kris
Lehigh University

RELIABILITY ANALYSIS OF REAL TIME COMPUTER ARCHITECTURES

by

Bryan Robert Kris

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidancy for the Degree of

Master of Science in Electrical Engineering

Lehigh University

1988

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

DECEMBER 12, 1988
(date)

[illegible]
Professor in Charge

L. J. Vanin
Chairman of the Department

TABLE OF CONTENTS

ABSTRACT:

This paper will investigate several different realistic architectures for Real Time Computer Systems to determine which is a more optimum design choice with regard to cost and reliability, and to explore a new method for system reliability analysis. This new analysis method requires only the knowledge of the interfaces within a system to predict the reliability of the entire system. Special attention is paid to actual component failure modes and how they affect different architectures.

INDEX TERMS: Fault tolerance, real time control, gracefull degradation, component failure modes, reliabilty calculation techniques, computer system architectures.

Chapter 1

INTRODUCTION

Realtime Computer Systems are probably the least appreciated computer systems by both the public and computer systems designers. However, these computer systems are increasingly common in factory production equipment such as robots, milling machines, and lathes. They run chemical plants and automotive test equipment. Recently, medical equipment and commercial aircraft have been introduced with real time control systems. Often these computer systems are called "Embedded Control Systems" because they are small and are hidden from view within a larger system.

Real Time Computer/Control Systems endure some of highest legal liability exposure of any type of computer system. A failure of a machine tool control can injure or kill its operator; a malfunction of a medical life support system can kill a patient; a failure of a chemical plant controller or an aircraft stabilzation system can kill hundreds of people.

The increasing cost of liability insurance is making the need for ultra reliable industrial control systems more imperative. Not only is the actual risk of an equipment failure important to consider in its design, but its perceived risk may be more important to those that design, sell, use, or insure the equipment. Often, many real time control systems are operated by non technically oriented individuals in an atmosphere of worker versus management antagonism. If a failure results in an injury there may be a strong urge to sue for real and/or imagined injuries.

Real Time Computer Systems typically cost between twenty and eighty thousand dollars. The typical markup for a piece of industrial equipment is three times the cost to manufacture the item. The markup includes costs for development, insurance, profit and allowance for dealer profit margins. The cost of an embedded control system is often one half of the cost for the entire machine. For a fifty thousand dollar machine, the cost allocated for the control may be eight thousand dollars. These eight thousand dollars must include direct labor costs for assembly and test as well as mundain items such as cabinetry, wiring, fans, AC power mains disconnects, and power supplies. The cost of the "electronics" is often contrained to less than four thousand dollars on a fifty thousand dollar machine.

There are many phrases that have been used to describe or characterize Highly Reliable Computer/Control Systems, including Fault Tolerant, Fail Safe, Fail Soft, Fault Detection, Fault Masking, Redundancy, Fault - Confinement and Reconfiguration [1],[2]. To the user of the equipment, the minimum requirement would be to alert the user to a failure when it occurs to allow the operator to take corrective action. The system should be designed in anticipation of common types of failures to prevent the system from acting in a manner that would injure an individual while the operator is attempting to respond to the indicated error. A desirable feature would be to design the system so it continues to function in a limited but safe manner to aid the operator while he attempts to take corrective action to the indicated fault. The ideal system would continue to function without degradation until corrective action is taken.

The ability of a real time computer system to continue to function without any degradation in the presence of a fault usually requires the cost of triple redundant hardware, which is too expensive to implement in most industrial control systems. Most of the time, the redundant elements would be unused, they represent lost opportunities for increased system performance or reduced capital investment by the user.

Real Time Computer Systems (RTCS) are usually produced by small firms because of the limited market size for such products, often only several thousand systems per year. These firms are of moderate size, typically employing a few hundred people with annual sales of ten to one hundred million dollars. Small firms, because of their limited financial resources, have product design cycles of one to two years. They do not usually develop new technologies but apply existing technologies to their problem domains.

The conventional means used by small companies to design reliable real time computer systems is usually an adhoc approach that contains a measure of guesses, hopes, and a little fact. These adhoc approaches develop in response to the cost in time and engineering talent needed to analyze different architectures that could be used for new real time computer systems. The proper method for evaluating new architectures for reliability is to design a model of the proposed architecture and then analyze the design, component by component. Formal methods developed under government or university direction are often complex and vague. [1 - 3]. The analysis would take into consideration the type of component, its relationship to the rest of the system, and requirements of the system in a given application. This is not an easy task, and it can get tangled in a web of subjective reasoning.

There is a need to analyze the reliability of some common real time computer system architectures that are often employed by many small system manufacturers. This thesis is aimed towards analysis and design of real time computer systems. The results of this work will aid those designers that are looking for some "Cook Book" answers to their quest for more reliable real time computer system designs. The analysis should also give some direction towards new, more cost effective, and reliable designs. The analysis of some typical designs may shed some light on some "short cuts" that may be used to analyze the reliability of new real time computer systems.

Chapter 2 of this paper describes some of the basic theory of reliability analysis and the effects of component failure modes on a system's operation. The following chapter then contructs some realistic models of common real time control systems and analyzes their reliability and cost. A simple analysis method that examines only the interfaces within a system is proposed in chapter 4 of this thesis. The results of the conventional reliability analysis are also compared to this new, faster analysis method. Chapter 5 then proposes new real time computer architectures based upon the proposed interface analysis method that are cost effective and reliable.

Chapter 2

BASIC RELIABILITY THEORY AND DATA

2.1. Basic Theory

There are many texts that cover the field of reliability, from system analysis to component physics [4 - 8]. This thesis will limit itself to the most basic theory.

The classic picture used to describe the failure rate of a component with respect to time is called the bathtub curve, which graphs the failure rate versus time. The bathtub curve starts at a high failure rate initially because of infant mortalities. The failure rate then falls to a steady state value that represents the useful life of the device. At the end of this period, the failure rate is expected to rise due to wear out mechanisms and accumulated stress. During the useful life of a component, it is assummed that the failure rate will be relatively constant.

The constant failure rate is described by the exponential failure model. The reciprocal of the failure rate, Lambda, is called the Mean Time Between Failures (MTBF) and is an important parameter describing system reliability. The reliability of a device with respect to time may be described by the following equation

$$R(t) = \exp(-\lambda \times t) .$$

The failure rate for a system is the sum of the failure rates for its components. The reliability of a system is the product of the reliability of its components if every component failure will cause the system to fail.

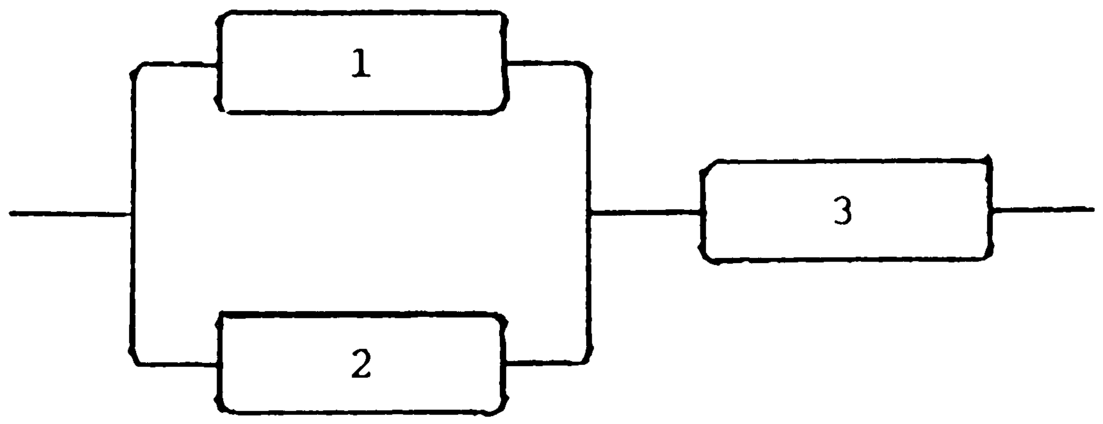
The reliability of a system with parallel elements where the system will function correctly even if one of the elements is faulty is described by the following equation

$$R_p = 1 - (1-R_1)(1-R_2) \ldots (1-R_n) .$$

Here the reliability of the parallel assembly is denoted by Rp and the reliability of each element is denoted by R1, R2, thru Rn. If the reliability of the elements is the same, then the reliability of a simple two element parallel system may be expressed as

$$R_p = 2R - R \times R .$$

The definition and calculation of system reliability can change depending on the definition of what constitutes a failure. The RAC Reliability Design Handbook (p.31) [4] gives a good example with a system built with three elements. Assume the following system architecture:



Let R1 = R2 = 0.85, and R3 = 0.99, and the time period is set at one hundred hours.

The nonredundant system, without element #2, would have a calculated reliability of Rs = R1 x R3 = 0.842 or a MTBF of 575 hours. If the redundant element #2 is added to the system, the reliability is calculated as Rs = R1 x R2 x R3 = 0.715 or the MTBF = 298 hours. At first glance it would appear that the extra components of the redundant element greatly reduced the reliability of the system. However, there are twp important aspects of reliability. The unscheduled maintenance reliability, is defined in terms of any failure that is considered significant and requires service before the system is regarded as fully functional. The mission reliability comes into picture when a failure is visibly detrimental to the operation of the system before it is considered a fault. The mission reliability is calculated as

$$Rs = [2R1 - (r1 \times r1)] \times R3 = 0.97,$$

which is considerably higher than the maintenance reliability.

The definition of what constitutes a failure will dramatically affect the calculated reliability of a system. The addition of circuitry to promote system reliability may harm a manufacturer's reputation for reliability if many service calls are generated by failures in the redundant circuitry. Even though most of the failures will not be "critical", the customer will only know that the "darn machine broke again".

2.2. Failure Modes And Failure Mechanisms

As far as this thesis is concerned, the term "failure mechanism" will represent the means by which the failure occured. The failure mode indicates the symptom of the failure as observed by the system. There are many opportunities for latent IC failures to be introduced into a system, including the manufacturing of the integrated circuits, and the handling of the components as they are packaged, tested, shipped, and assembled into systems.

During the manufacture of integrated circuits, the primary mechanisms of failure are metalization failures, poor wire bonds, and photolithography errors. During shipment and handling, the primary cause of failures is due to electrostatic discharges that damage the integrated circuit by vaporizing metalization or puncturing oxide layers. Another source of failures is the misuse of the device within a circuit, such as a logic design error that allows two different bus drivers connected to the same signal to be turned on at the same time. This contention can allow excessive currents to flow through the drivers and cause them to fail. Some system design flaws can cause problems through secondary effects such as excessive thermal cycling of the components that causes metal fatigue within the integrated circuit or the PCB.

The design induced failure is difficult to reduce to an identifiable quantity for purposes of discussion of fault tolerant architectures. For a specific class of design flaws, such as bus buffer contention, it may be possible to design a system to be resistant to that fault but the obvious question would be "If one can predict a system's response to a design failure, why not use that knowledge to prevent the design error from being made?". Many design errors can be independent of the system architecture, such as excessive thermal loading due to poor packaging of the system. For the purpose of this thesis, it will be assumed that the implementation of a system will be free of hardware design errors.

The obvious method to increase the reliability of a system is to use components that fail less frequently. Many of the failure mechanisms for integrated circuits may not become visible immediately to the system. Often the failures occur days or weeks after the event that provoked the failure. The most common means to guard against these "time bombs" is to burn-in the components and the completed assemblies at elevated temperatures to hasten the latent failure in becoming visible. The use of temperature cycling and vibration are also used to shake up the components in an attempt to expose the marginal devices or assemblies. The net result of these burn-in proceedures is to remove the infant mortalities. The components that do fail later, should do so randomly with respect to time and device type. The typical paper on reliability and/or fault tolerant systems design assumes that the failures are random in nature and that they are equal in effect to the system. This is blatantly not true, for example: If a bus connected device's output shorts to ground, it may disable all transactions on that bus. But if the device fails as an open, then the bus may function properly except for the failed device. The failure modes of integrated circuits should be considered in the design of fault tolerant real time controls.

Information on actual component failure modes and failure rates is limited to a few sources [9 - 11]. The premier source for information on the ways in which real components fail in real world systems is the Reliability Analysis Center at the Rome Air Development

Center at Griffiss Airforce Base in New York. According to the RAC Microcircuit Device Reliability Field Experience Analysis Study (1985), the major failure modes per device catagory are:

Non Hermetic Bipolar Digital Logic

Open	< 1.0 %
Short	< 1.0 %
Degraded	= 70.0 %
Functional Anomaly	= 20.0 %
Mechanical Anomaly	= 10.0 %

Non Hermetic MOS Digital Logic

Open	= 6.0 %
Short	= 1.0 %
Degraded	= 43.0 %
Functional Anomaly	= 48.0 %
Mechanical Anomaly	= 2.0 %

Non Hermetic Bipolar Interface

Open	= 11.0 %
Short	< 1.0 %
Degraded	= 53.0 %
Functional Anomaly	= 30.0 %
Mechanical Anomaly	= 5.0 %

Non Hermetic MOS Interface

Open	= 29.0 %
Short	= 29.0 %
Degraded	= 14.0 %
Functional Anomaly	= 0.0 %
Mechanical Anomaly	= 28.0 %

Non Hermetic Memory (MOS)

Open	= 1.0 %
Short	= 1.0 %
Degraded	= 74.0 %
Functional Anomaly	= 23.0 %
Mechanical Anomaly	= 1.0 %

Non Hermetic VLSI (MOS)

Open	= 6.0 %
Short	= 2.0 %
Degraded	= 17.0 %
Functional Anomaly	= 69.0 %
Mechanical Anomaly	= 6.0 %

The expected failure rate per million device hours for the different classes of devices and their technology is as follows:

Bipolar Logic	=	0.3
MOS Logic	=	0.3
Bipolar Interface	=	0.25
MOS Interface	=	0.25
MOS Memory	=	0.06 to 1.0
MOS VLSI	=	0.75

The RAC study found that memory failure rates seem to be independent of complexity and highly sporadic according to specific components or vendors. The reported failure rates for the different kinds of integrated circuits are about equal, and will be standardized as 0.3 failures per million hours.

2.3. Other Reliability Considerations

The reduction of the temperature in which a control system operates, and the reduction of system interconnections are means of achieveing greater system reliability, as detailed in Appendix A.

The temperature of the device has a strong effect on the expected device reliability. For bipolar devices, the failure rate doubles for each ten degree Centigrade rise in junction temperature. CMOS device failure rates triple for each ten degree rise in temperature.

An integrated circuit that was screened to 883B levels and used in a typical industrial control environment has the following calculated reliability:

Failure rate = 0.283 failures per million hours.

This calculated failure rate is very close to the observed failure rates for ICs as observed by RAC.

The only parameters for device reliability that the system designer has some control of are device screening and the ambient temperature in which the ICs operate.

The benefit of lower component operating temperature and device screening on the predicted reliability of a component are applicable to all designs of real time computer systems. Appendix B lists some typical components and their cost when procurred with different screening levels. A screening level of class D, which results in a doubling in predicted reliability for a component over an unscreened part, costs about fifteen cents per component. This additional cost may raise the component cost of a system by ten percent which is not excessive. Additional screening to level B will increase the component cost by four hundred percent! This extra cost prohibits the use of such components in lieu of proper system design in most reliable real time computer systems.

Chapter 3

CONSTRUCTION OF REAL TIME COMPUTER SYSTEM MODELS

3.1. Introduction

To accurately analyze different architectures for real time computer systems requires the development of realistic model designs. Simply drawing box diagrams of proposed systems will inevitably result in the over simplification of a design, and the results of any reliability analysis will be overly optimistic.

3.2. Performance Requirements

The purpose of a Real Time Computer System is to perform a productive, useful task. A machine tool control may supervise a machine with five axis of motion to a precision of one ten thousandth of an inch while the axes move at four hundred inches per minute. The motions of the axes might be synchronized with the revolution of the cutting tool, turning at ten thousand revolutions per minute. If a machine control has a resolution of 0.0005 inch and is moving at 400 inches per minute, then the control must calculate 13,333 points per second. If the machine motion is in four axes (table x,y,z and spindle), then 53,333 sets of calculations must be performed each second. Additionally, the control must perform other jobs such as interpreting the operator commands and displaying current machine position while communicating with other control systems or central computers. With resolution of 0.0005 inch and allowable dimensions of 1000 inches, math operations require a minimum of 26 bits of precision, which is usually handled as 32 bit integers or 64 bit floating point values. A point's calculation may require a minimum of two additions, four subtractions, and six comparisons for a simple circle. A three dimensional spiral may require two additions, a subtraction, four multiplies, and two divisions for each calculated point. A very important requirement of a machine tool control is its ability to guarantee that the path calculations will be done in real time as needed. A failure in delivering the proper commands to the machine's servo motors at the proper time can cause the destruction of the work piece through the introduction of DWELL marks or cause operator injury by overloading a cutting tool and causing the tool to shatter. In addtion, the machine tool control must monitor the operation of the servo motors to insure that they are positioning the work piece and cutting tool as commanded. One can thus see that the real time computing workload is beyond the capabilites of most moderate size computer systems.

Another example of a high performance real time system is medical equipment used to collect and display vast amounts of data, often requiring many CRT displays just to present the data in a convenient form. A hospital patient monitoring system may collect and display the following waveforms: ECG, EEG, Saturated Oxygen, multiple Blood Pressures (invasive and noninvasive), Respiratory O2, CO2, N2O, and Anesthetic Agent, as well as Breathing Pressure and Volume. In addition to real time waveforms, relatively slow changing data such as the temperature, and computed values such as the Pulse rate is collected and displayed. Information such as that from a Blood Gas Analysis machine may be acquired through communications links within the hospital. The collected data must be sorted and prioritized, and the proper alarm conditions must be recognized and annunciated. The data and alarm information is then cataloged in an internal memory for trending purposes. The trending of data can use large amounts of memory. For sixty data items that are sampled every six seconds with sixteen bit resolution for up to ten hours of duration, 720,000 bytes of ram is needed.

The collection and dissemination of information can impose a large I/O interrupt burden on the processor. A medical system might have five external communication channels operating at 9600 baud, an internal communication channel for ECG operating at 38.4 K baud, two channels operating at 4800 baud for data such as saturated oxygen (SAO2) and noninvasive blood pressure (NIBP), and a 9,600 baud link to communicate with a respiratory gas analysis device. These communication ports can generate 10,560 interrupts per second just for receiving data. If the ports are 70% utilized, then a realistic figure of 7,392 interrupts is obtained. These communication ports also have transmitting channels that are typically 20% utilized, they generate an additional 2,112 interrupts per second. The system will generate its own interrupts for use by its operating system. A total load of 10,504 interrupts per second is reasonable for a real time control system! A typical Interrupt Service Routine (ISR) may have twenty instructions. Therefore the work load for just handling these interrupts, may be more than 0.2 MIPS (Million Instructions Per Second). If a large operating operating system must be involved to process the acquired data, the interrupt overhead can easily exceed 1.0 MIPS.

To make real time industrial control systems more "User Friendly", many systems are incorporating large amounts of text containing operating instructions, warnings and equipment checkout lists. The displays have become more graphically oriented, often including diagrams on how to service the machine. These displays and tests require large amounts of read only memory (ROM or EPROM). A quarter of a megabyte would support about a dozen pages of text and half a dozen graphically oriented screens.

The CRT display(s) also require a lot of resources just to present a pleasant image to the operator. The control may have two monochrome displays with two level gray scale at a resolution of 640 by 480 pixels, requiring 153,600 bytes of RAM for video display refresh memory. If color and/or higher resolution displays were desired, the refresh memory requirement would grow much larger. A large processing load is imposed just to update the image with new data. These displays have a total of 614,400 pixels. A large portion (40%) of the display may be relatively static and require little updating except when a new screen is drawn. A smaller percentage (35%) of the screens display numerical data (in large fonts) that is updated several times a second. The remainder (25%) may display rapidly changing data such as waveforms or images that is updated every 20 milliseconds. This update rate requires about 16,650,240 accesses to the video refresh memory each second! The hardware is often designed to access multiple pixels per memory cycle to reduce the required bandwidth. If the video memory is designed to access sixteen pixels at the same time, the number of accesses can be reduced to about one million per second.

A typical real time control system may require the following minimum resoures:

1. CPU (s) sufficent for the task. (2 - 5 MIPS)
2. 512K bytes of ROM for program storage.
3. 512K bytes of RAM for data storage.
4. 5 - 8 Serial ports, to internal/external devices.
5. CRT Display logic, 1k x 1k by 4 planes (minimum).
6. Key Panel interface logic.
7. Audio Alarm Circuitry.
8. System Clocks and Real-Time Clock.
9. Machine dependent Analog Input/Output.
10. Machine dependent Digital Input/Output.

Appendix C lists some commercially available real-time control systems and the system resources that are allocated in their design.

3.3. System Cost/Complexity Constants

An analysis of a variety of microprocessor based PCBs (Appendix C) shows that an "average" integrated circuit costs about ten dollars when installed into a PCB and it occupies 1.7 square inches. Therefore, a standard real time industrial control system limited to four thousand dollars cost, would have about four hundred integrated circuits and occupies 680 square inches of printed circuit board area. Obviously, a system designed with VLSI components could be smaller with regard to PCB area and component count, but the component cost would be greater. Custom designed integrated circuits reduce system cost and size, which in turn allows fault tolerance logic addition; but the additional design effort needed to create custom integrated circuits reduces the design resources available to design the product itself. A given allocation of design resources limits the total system complexity. A modest size design team may be limited to "pushing commercially available ICs around a PCB into different configurations in a hope of reaching a reasonably effective design". The data in appendix C shows that in a typical system, fifty percent of the chips might be relegated to Input/Output functions. That leaves about two hundred chips to perform the computational tasks, store the program and data, and if there are any "left - over" to provide for the system's fault tolerance features. It is reasonable to assume that fault tolerance circuitry be nominally limited to fifteen percent of the system's total cost. The primary means to provide for a fault tolerant system might be embodied in architecture of the system[12 - 19]. The larger challenge is to design a moderate cost, fault tolerant architecture.

3.4. Overview of Model Architectures

The simplest and the most common architecture for real time industrial control systems is the standard micro processor bus system that incorporates a single processor with memory and Input/Output circuitry. Its logic may be totally contained on a single printed circuit card, or it may reside on many PCB cards that communicate with address, data, and control buses. Most personal computer systems are organized in this fashion. This format will, for the purposes of discussion, be called the "Standard Bus". The chief advantages of the Standard Bus is its simple design, ease of expansion (just add more PCB cards), and the availability of many board level micro processor based PCBs from a large selection of vendors. An example of such a system is the MultiBus system promoted by Intel Inc.. The chief disadvantage of the standard bus configuration is its lack of tolerance for faults that occur on the buses as there is no means to isolate faults. The cost of the standard bus is highly dependent on the implementation and partitioning of the system. If the system can reside on a single printed circuit board, the standard bus may be the most cost effective architecture. When the standard bus is divided into many different PCBs, the cost of the interface circuitry may dominate all other system components.

A more advanced version of the standard bus would be a scheme where multiple microprocessors would share resources on a Global Bus but they would have access to their own private resources (such as memory or I/O) over a Local Bus. These processors are considered to be tightly coupled because they share resources. An example of this archtitecture is Motorola's VME system. This kind of system will be called a "Complex Bus". Obviously, this scheme seems to promise the ability to maintain some form of continued system operation in the advent of a fault in the Global bus because the individual processors may continue to access their resources through their private (local) buses. The increased fault tolerance for this system inccurs the cost of the additional buses.

The Complex Bus scheme has a weakness where if a fault such as a "rampant processor" took control of the Global Bus, it may effectively lock out all of the other processors from gaining access to the global bus and the shared resources (such as memory). For the cost of some additional resources, each processor could be made self sufficent with regards to its basic needs for memory and I/O. These microprocessors could communicate with each other over a communications bus instead of a basic micro processor bus. Instead of an address being presented by a processor and a data value expected in return, the protocol would be at a higher level where commands would be issued and complex multibyte responses would be returned by another processor. This system is called a loosely coupled Distributed Peer Processor System, where each processor has approximately the same resources and status within the system. In this scheme, it would be difficult for a failed microprocessor to disrupt the operation of another processor. The main drawback of this design is its cost; each processor may be limited in performance and resources because of the need for redundant resources and the support circuitry to make these resources usable. If many processors are implemented, each may be reduced to a very small size. Instead of a single thirty-two bit processor, a dozen eight bit processors may be needed to support the work load. The limited processing power of a smaller processor may require large amounts of data be transfered between each processor, so each can process the data in parallel. A large portion of each processor's resources, hardware and processing power, may be spent on the communications channel. To reduce the latency of data transmissions between the processor modules, high speed parallel communication channels are often used. These parallel interfaces put additional hardware burden on each overloaded processor module.

The fourth architecture to be considered here is the "Octopus" system where a large central processor communicates with a satelite system of smaller processors over a number of slow speed serial channels. If the system can be partitioned into modules that require low bandwidth communications between the modules, then the hardware resources that would be spent on high speed communications can be devoted to more productive uses. The octopus system is a Master-Slave organization. If any of the slave processors fail, it will be unlikey to cause the entire system to fail. But if the master processor suffers a fault, the system might be disabled. The "Trick" to a successful system design with this archtecture is to design the slave processors to be able to operate the system in an automatic mode if the communication with the master processor is lost.

There are other architectures that are very fault tolerant through the virtues of massive redundancy. The space shuttle and modern aircraft may use three or more processor subsystems operating in parallel. The results of each processor is examined by a voting circuit. The voting circuit will detect errors by looking for outputs that do not agree with the others. If an error is detected, the faulty unit will be turned off or disregarded until it is repaired. This scheme is the most reliable known, but it is also the most expensive design.

Other special purpose architectures, such as arrays of processors, may be fault tolerant because the size of the processor, and its data, may be small compared to that of an entire system. An example: An image processing array could have some faulty elements that are responsible for processing the data for a few pixels within an image containing a million pixels. Such faulty pixels may never be noticed in the overall image.

It is reasonable to assume that technology will advance to allow the implementation of redundant or massively parallel arrays of processors on a single printed circuit board in the near future. But it will be an uncommon event within realtime industrial control systems because of the ever increasing demands from them for higher performance at lower cost. The current architectures such as the standard bus, complex bus, the distributed peer system (communications bus), and the octopus system are likely to remain in use for the foreseeable future.

3.5. Comparisons of the Design Examples

The detailed descriptions of the model systems are in Appendix E. The systems are described by lists of their components. The listed components are organized into functional groups with accompanying descriptions. The listed components are typical for the function performed but other devices may be applicable.

The overall integrated circuits (IC) count for the four design examples are:

Architecture	IC Count	I/O Cost	IF Cost
Standard Bus	152	40%	22%
Complex Bus	270	39%	33%
Communications Bus	222	40%	28%
Octopus System	153	58%	11%

The winner for lowest component count is the standard microprocessor bus which narrowly beats the Octopus system. The actual implementation of these two systems determines the lowest component count. If the octopus system is implemented with descrete microprocessors instead of microcontrollers, the standard bus design would have even a larger component count advantage. If the standard bus design is built on smaller printed circuit boards, the additional boards and their required bus interfaces would greatly increase its system component count.

The most expensive design is the Complex bus with almost twice the component count of the standard bus or the octopus system. The additional cost is primarily due to the redundant bus interfaces, and the redundant control logic and memory for the extra processor. Most applications can be mapped onto a complex bus system. The system can reside on one of the processors, and the user interface with its video graphics interface can reside on the other processor.

The communications bus example is most difficult to analyze because of its significantly different structure. If the design is adequate for the application, then its cost of 222 chips is approximately "average". If the design is implemented with 16 bit processors, its cost increases by at least forty more chips. The sixteen bit communications bus system is as costly as the complex bus. Like the complex bus, the predominant cost in this case also is the interface logic.

The Octopus system has less than one third, by component count, the interface ciruitry of the other systems. The obvious reason for this is the use of serial interfaces instead of the usual parallel interfaces.

Chapter 4

RELIABILITY ANALYSIS OF MODEL SYSTEMS

4.1. Conventional Reliability Analysis

The calculated reliability of each architecture model in this thesis is based only on the integrated circuits. The passive components such as carbon composition resistors and ceramic capacitors used for signal termination and power supply bypassing have failure rates that are one to two orders of magnitude lower than those for integrated circuits. Two failure rates are calculated for each architecture model, the service MTBF and the mission MTBF.

The service MTBF is just a summation of the failure rates for all of the integrated circuits. The mission MTBF calculation is more complex. It is assumed that each model system has enough diagnostic hardware and software to determine if non critical devices, such as I/O are functioning properly. This is needed to prevent the use of a system that appears to work but would give incorrect responses. The critical device failures are expected to either prevent the system from booting up and/or determining that a major fault has occurred. Often the microprocessor and/or system will incorporate a "watch dog timer" or other monitoring logic to shut the system down to prevent erraticoperation in the advent of a critical failure.

Typically, a bus connected device that suffers a shorted failure mode on a pin that connects to a bus will cause the system to fail catastrophically because the bused signal line becomes unusable. If the device fails as an open circuit, then the signal line is still usable by other devices. When a device is connected in series with a bused signal, then either a short or open will make the signal line unusable. This discussion leads to the following assumptions (rules) to determine which failures are critical to the system:

1. The Microprocessor, its clock, control, and reset circuitry are mission critical devices.
2. The bus interface buffers and transceivers are considered critical. The RAC data shows that a large percentage have opens or shorts as failure modes [11].
3. Any integrated circuit that can disable the processor or disable a majority of the system I/O through its failure will be considered critical.
4. Any integrated circuit, whose failure will completely disable the video displays is considered critical if there are no alternate means of displaying data to the operator.

5. Memory devices are not critical. The RAC data shows few memory failures as open or shorts. If the system boot code occupies a small percentage of the system memory, then a failure in the boot code is unlikely. If the boot code is readable, then assume system critical code will be duplicated in different areas of memory. CRC codes can be used to verify the integrity of each block of memory.

6. Bus connected devices that are on the other side of a bus buffer device from the critical microprocessor circuitry are not considered critical.

7. Microcontrollers, whose primary task are as I/O controllers, are not mission critical.

8. All non micro-processor bus connected I/O devices are not critical devices. Their failure will not disrupt the operation of the microprocessor.

There are failure modes that can disrupt almost any system, even if the device is not considered critical to the mission. For example: an I/O buffer could suffer a short between the system power and ground that would cause the voltage to other integrated circuits to fail. This event is highly unlikely because the typical power supply can generate a large enough current to burn out the bonding wires that supply power to the faulted integrated circuit before the system power fails.

Rule four, which states that any integrated circuit that can completely disrupt the video display will be considered mission critical, was added to list because of the recognition that "human factors" considerations are important. If the video interface is the only interface between the machine and the operator, then its complete failure will probably tramatize the operator and the machine will be considered hopelessly and dangerously broken. Even if the processor was still capable of maintaining control of the machine, no one would know it or believe it. Partial failures such as missing pixels, scrambled lines, or rolling images, would not cause such panic as would a complete failure. The operator would still see "something" and realize that the system was still attempting to operate.

A system built with multiple processing elements should be analyzed in a manner similar to a conventional system. Each processor module should be treated as a component and the above eight rules should be applied.

The standard bus design example's critical components were the processor, processor clock, reset logic, control decode PLDs, and the bus buffers for address, data, and control signals. The interrupt controller, interrupt signal buffers, and DMA signal buffers were not considered mission critical. It is assumed that the control decode PLDs will disable such functions until the processor deems that such functions are operating properly and are needed. On the I/O board, the 95C60 video processor, video clock, the buffers for address, data, and control are also critical to the mission.

The Complex bus design example has the same critical components as the standard bus and its adds the local bus buffers to the critical components list. The complex bus then pays dearly because there are two processors in the system, thus doubling the number of critical components! The common video board adds the video controller, clock, video control PLDs and the bus buffers to the critical device list. The logic used to share the video board between the processors is also critical. The I/O boards on the VMX buses however are not critical. The dual processor system does not provide enough fault isolation or redundancy to insure the continued operation of the system in case of a fault to be able to consider that each processor is not critical. This is not a clear cut decision but made as it was because of the heavy dependence on a common video and a lack of redundant I/O.

The communications bus design example list of critical components was primarily made up of the parts on the Internal Communications Controller. This board, should it fail, would disable both video display processor boards. Its failure would also prevent each I/O module from communicating with each other. The only components on the ICC board that are not critical to the mission are those devoted to the interface with the External Comminications Controller, and the audio output circuitry. In this rare case, the memory devices will be considered critical because it is unlikely that such as small processor subsystem would have additional resources to provide for redundant blocks of memory or program. The bus interface devices on the I/O modules are also capable of disabling the system if they should fail.

The Octopus system design example's critical component list only includes the usual, the processor, clock, control logic, video processor, video clock and logic. The lack of bus buffers greatly reduces the number of critical components as compared to the other design examples.

The Table 4.1 summarizes the results of the critical device analysis rules when they were applied to the example designs.

Table 4.1. Design example critical IC tally.

Design	Total ICs	Total Critical ICs	"OVERHEAD" Critical ICs
Standard Bus	152	48	37
Complex Bus	270	87	53
Communications Bus	222	37	13
Octopus System	153	20	18

The component counts can be converted into system MTBF values assuming a standard 0.3 failures per million hours MTBF for the integrated circuits as shown in table 4.2.

Table 4.2. MTBF Summary

Design	Service MTBF	Mission MTBF
Standard Bus	21,929 hours	69,444 hours
Complex Bus	12,345 hours	38,314 hours
Communications Bus	15,015 hours	90,090 hours
Octopus System	21,786 hours	166,666 hours

The standard bus design example appears to be a reasonable design choice for real time operating systems. The standard bus provides for a minimum component count, ease of system expansion, and the best service MTBF. But, the standard bus does not provide the optimum mission MTBF.

4.2. Reliability Calculation Through Interface Analysis

All usefull computer systems have a common heritage, they all contain at least one processor, memory, control logic, and input/output devices. Because of cost limitations, most designs allocate approximately the same amount of system resources to perform the same task as other designs. The basic difference between different systems is in their organization. The interfaces between the different components represent the discretionary differences in each design. These interfaces also define the paths for data transfer within a system, and they are often critical to the reliability of a system.

It should be possible to calculate the reliability of a computer system by analyzing the interfaces within the system. Such a procedure would save time, over more conventional methods, because the entire system would not have to be designed and analyzed. The procedure should take into consideration, the number of interfaces, the failure modes and rates of the components, and any redundancy built into the interface. The remainder of the system will be factored into the reliability equation as a constant based on the estimated total component count of the system.

Any interface can be modeled as a chain of four elements that connect one module to another module as shown in figure 4.1.

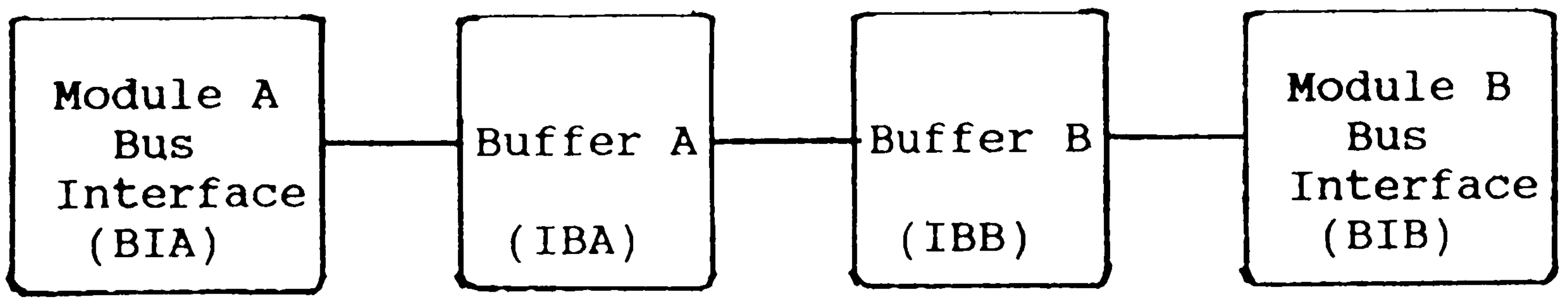


Fig.4.1. Interface between two modules.

Many interfaces may not include all of these elements, or elements may be combined within a component. Conversely, many components may be required to implement a single element of an interface. The module A bus interface represents the connection between the interface and the system bus of module A. Similarily, the module B bus interface is the connection between the interface and the main bus of module B. If these bus interfaces fail in a shorted mode, then the respective bus will be critically faulted. The buffers A and B are the physical buffers (amplifiers) that are typically used to provide the proper voltage and drive at the interface connection. These buffers are often built into the bus interface devices. It's possible to combine all four functions in a single device! It's common to use a bus buffer such as a 74ALS541 within a PCB to increase the drive capability of a microprocessor output. Thus interfaces may exist within a single PCB or may extend centimeters or kilometers to another PCB.

The portions on any interface that may be designed in a redundant manner are confined to the portion of the module interfaces that connect to each other through the interface buffers. Any redundant interface must have a circuit or device that selects, combines, or monitors the redundant paths. At some level, this path selection circuit is not redundant, and it must be connected to the system bus. To see this, consider that the module interfaces are implemented with VLSI octal UARTS, the buffers are RS-422 driver and receivers. To allow for fault tolerance, two different serial ports in each octal uart are used to implement a single communications channel. The portion of each octal uart that connects to the bus is shared by all of the serial ports and therefore is not redundant. The portion of the octal uart that is involved with a specific serial link may fail with out affecting the other serial links. This portion of the VLSI device may be considered to be a part of the redundant path.

A similar example could be designed with triple redundant open collector parallel bus buffers that have been resistively connected together at their outputs.

Their collectively tied inputs and their resistively tied outputs represent portions of the interface that are not redundant.

The interface model's reliability can be calculated easily if each element is considered separately. An interface circuit is shown in fig.4.2 to clarify Fig. 4.1.

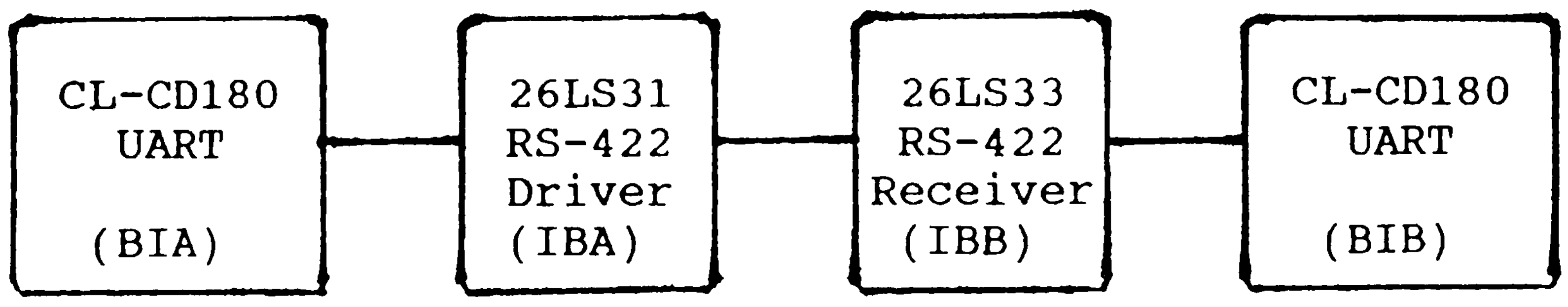


Fig. 4.2. Example Interface Circuit.

The portion of BIA that connects to the bus of Module A is by definition a critical component. The failure rate for the BIA interface element is represented by IBIAFR. It is assummed that one half of the failures for this portion of the interface occur on the bus boundary. The portion of these failures that are shorts, which are always critical, is represented by BIAFM. The failure rate for the bus side of BIA that is always critical can be calculated by the equation

$$FR = 0.5 \times IBIAFR \times BIAFM . \qquad (4.1)$$

The remainder of the other failure modes for this half of IBIA are taken as critical only if the signal path is critical. The fact of the criticality of a signal path is represented by the variable CP. If the path is critical then CP is equal to one, otherwise it is equal to zero. The redundancy of the signal path will greatly influence the criticality of the entire interface. The signal path redundancy is represented by the variable redundancy factor (RDF).

The RDF is calculated by taking a typical failure rate for an integrated circuit and modeling a system with two such components in parallel. The reliability is calculated for the parallel components using a period of time between two thousand hours (one working year) and twenty thousand hours (over two years continuous). The effective reduction in failure rate for the signal path was on the order of a factor of one thousand. For redundancy of more than two, the effective reduction in failure rates becomes on the order of one trillion.

The remainder of the bus portion of BIA, that may be critical if the signal path is critical, has a failure rate that is described by the following equation,

$$FR = (CP \times RDF) \times (1 - BIAFM) \times (0.5 \times IBIAFR \times BIAFM). \quad (4.2)$$

Finally, the failure rate for the portion of the bus interface IBA that connects to Buffer A is calculated by:

$$FR = (CP \times RDF) \times (0.5 \times IBIAFR). \quad (4.3)$$

This portion of the interface is critical only if the signal path is critical and only to the degree determined by the redundancy factor.

The interface Buffers A and B are critical only if the signal path is critical. The effective failure rate for these buffers and the module interfaces are dependent on the redundancy of the connection (RDF). The failure rate for both IBA and IBB is determined by:

$$FR = (CP \times RDF) \times (IIBAFR + IIBBFR). \qquad (4.4)$$

The BIB element of an interface, that connects the signal to module B, is divided into two halves. The half that connects with Buffer B (IBB) may be designed in a redundant manner. Its failure rate is described by equation

$$FR = (CP \times RDF) \times (0.5 \times IBIBFR) . \qquad (4.5)$$

The portion of the module interface that resides on Module B's bus can not be made redundant. This half of the IBB element is critical if the path is critical and its failure rate is determined by equation

$$FR = CP \times 0.5 \times IBIBFR . \qquad (4.6)$$

The total mission failure rate for a system is the sum of the mission failure rate of the interfaces plus the mission failure rate of the rest of the system. The total interface failure rate is the summation of the failure rates for each individual interface signal. The failure rate equation for a given interface signal is the sum of equations (4.1) through (4.6). The combined equation is called the Interface Reliability Analysis (IRA) equation in this thesis and reads:

$$\begin{aligned} FR = {} & (0.5 \times IBIAFR \times BIAFM) \\ & + [(CP \times RDF) \times (1 - BIAFM) \times (0.5 \times IBIAFR \times BIAFM)] \\ & + [(CP \times RDF) \times (0.5 \times IBIAFR)] \\ & + [(CP \times RDF) \times (IIBAFR + IIBBFR)] \\ & + [(CP \times RDF) \times (0.5 \times IBIBFR)] \\ & + [CP \times 0.5 \times IBIBFR], \end{aligned} \tag{4.7}$$

where IBIAFR = Individual Bus Interface A Failure rate

BIAFM = Fraction of BIA failures as shorts

Bipolar Interface	= 0.01
MOS Interface	= 0.29
MOS VLSI	= 0.02

CP = Critical Path, 0 = No, 1 = Yes

RDF = Redundancy Factor

1.0	= None
0.001	= Redundant signal paths

IIBAFR = Individual Interface Buffer A Failure rate

IIBBFR = Individual Interface Buffer B Failure rate

IBIBFR = Individual Bus Interface B Failure rate

The equation for the failure rate of an individual signal may be simplified greatly depending on the design of the interface. If the signal path is redundant, then RDF reduces most of the terms to insignificance and the IRA equation is reduced to:

$$FR = (0.5 \times IBIAFR \times BIAFM) + [CP \times 0.5 \times IBIBFR] \quad (4.8)$$

If the path is not critical then the equation reduces even further to:

$$FR = (0.5 \times IBIAFR \times BIAFM) \quad (4.9)$$

If the buffers are combined with the bus interfaces, and no redundancy exists in the critical signal paths, such as in a typical VME Bus interface, then the IRA equation reduces to

$$FR = IBIAFR + IBIBFR . \quad (4.10)$$

This is the typical form used in simple component count failure rate summation.

The total system mission failure rate is determined by summing together the failure rates for each interface signal. This is usually easy to do because many interfaces are usually groups of 8, 16 or 32 similar signals. The totalized interface failure rate is then added to a constant K that represents the failure rate of 15% of the estimated system component count. This fixed percentage of the system total represents the processors, clocks, control circuitry, and indispensible input/output circuitry that every system requires. This figure of fifteen percent matches the critical overhead of the four model systems.

4.3. Application of the Interface Analysis Method

For all the models under consideration, the following conditions exist:

1. The interface technology is bipolar; the incidence of shorts is regarded as insignificant.
2. All interface signals in the models are not of a redundant design.
3. All of the interface signals for the standard bus, complex bus, and communications bus models will be considered critical.
4. The interfaces for the standard bus, complex bus, and the communications bus are implemented in two levels. The BIA and IBA interface elements are combined, as are the IBB and BIB elements.
5. All of the interfaces are implemented with eight bit interface ICs. The standard failure rate of 0.3 failures per million hours per IC is assumed.

The Standard Bus example has 104 interface signals. The calculated failure rate for all of the interfaces is 7.8 failures per million hours. The fifteen percent allotment for critical overhead ICs, totals twenty three

ICs for the Standard Bus. The calculated total mission critical failure rate for the standard bus design is 14.7 per million hours.

The Complex Bus example has 384 interface signals. The calculated failure rate for all of the interfaces is 14.4 failures per million hours. The critical overhead IC allotment is forty one. The predicted mission critical failure rate for the complex bus is 26.7 per million hours.

The Communications Bus example requires a more sophisticated interpretation of the interface analysis technique. The "party-line" interface design, where many modules share a common bus is reduced to a simpler form for analysis. The six I/O processor modules with their separate bus interfaces will be treated as a single device. This device will have a bus interface that is six times more complex (in component count) that interfaces with the internal communication controller. The dual CRT controllers will also be treated in a similar manner with respect to their interface with the internal communications controller (ICC). The predicted failure rate for the sixteen data and control signals between the I/O modules and the ICC is 6.3 per million hours. The sixteen signals in the interface between the ICC and the

CRT modules are predicted to have 1.8 failures per million hours. The calculated total failure rate for this system including the critical overhead ICs is 18.09 per million hours.

The Communications bus example also offers another interpretation using the interface analysis method. The entire ICC module is just a portion of the interface between the I/O modules, that collect the data, and the CRT modules, which display the data. Treating the ICC as a black box of interface circuitry will increase the calculated interface failure rate to 13.5 per million hours. The calculated mission failure rate for the system is 23.5 per million hours. This rate is significantly greater then the results of the first interpretation. This second interpretation is probably not valid since the other design examples were not treated in this manner, but it does indicate that higher level analysis to large systems can yield results that are "in the ball park".

The Octopus System is another interesting test case for the Interface Analysis Method. The system has no interface signals that are critical by themselves, only the loss of a majority of them would be catstrophic to the system. The Interface Analysis Method only considers each interface signal by itself. The interface analysis method will count only the portion of the UART in the Octopus design that resides on the system bus and is expected to fail in a shorted mode. The predicted interface failure rate is 0.003 per million hours. If the interface method did consider the entire uart and its clock, then the interface failure rate would be calculated as 0.6 per million hours. The standard critical overhead brings the system total failure rate to 6.888 per million hours. In this case, the deficiencies of the interface analysis method do not greatly affect the result.

4.4. Comparison of Analysis Methods and Results

The combined results of the conventional analysis method and the interface analysis method are listed in the table 4.3.

Table 4.3. Comparison of Analysis Results

Design	Conventional Mission MTBF	Interface Method Mission MTBF
Standard Bus	69,444 hours	68,027 hours
Complex Bus	38,314 hours	37,453 hours
Communications Bus	90,090 hours	104,166 hours
Octopus System	166,666 hours	145,243 hours

The results are very close to each other, the maximum difference was only fifteen percent! The best correlation between the two methods was on the more conventional architectures. The communications design example demonstrated that there will always be areas of analysis that are subjective in nature, such as what constitutes an interface. The octopus design example highlighted a weakness of the interface analysis method, where the sum of a design behaves differently then each component.

The results indicate that the original hypothesis that the reliability of a system is primarily dictated by its interfaces is true. The interface Analysis method system offers the over worked and understaffed engineering departments that design real time computer systems a quick and accurate means for calculating the mission critical MTBF of a system.

Chapter 5

IRA SUGGESTED ARCHITECTURES

5.1. Future Design Considerations

The Interface Reliability Equation gives direction to the task of developing more cost effective and reliable realtime computer architectures. This chapter is devoted to the discussion of design modifications to improve reliability.

Note first that mission reliability can be improved by reducing the number and size of interfaces in a system. To reduce the system interfaces, one should use serial interfaces where possible. Secondly the technology of the circuitry on the system bus boundary should be chosen to minimize the chance of shorted failure modes which are catastrophic to the system. The choice of components that are known to have lower failure rates is also indicated by the equation as a good idea for the entire interface.

The equation also indicates that redundant interfaces be designed into the system to improve the reliability of the system. The level of redundancy should be kept at two or three. Any additional paths will only reduce an already insignificant failure rate for the signal path but it will probably greatly increase the complexity, cost, and failure rate for the interface circuitry on the system bus boundary. This increase in the boundary circuitry complexity will swamp any gains in reliability of the signal path.

5.2 A New Real Time Computer Architecture

The IRA equation can be used to guide the design of more reliable, cost effective real time computer systems. The new design will meet the performance criteria specified in chapter 3 while exceeding the mission MTBF of the previous four model designs. The new design will cost less than some of the previous model systems!

The primary statement of the IRA equation is that the number of interfaces in the system must be minimized. The primary means to minimize the quantitiy of interfaces is to use serial rather than parallel interfaces. This leaves two choices for the design of the interfaces. The first choice could be to attempt to replace the typical parallel interface, that might be sixty-four bits wide, with a serial interface capable of the same performance. This decision would require extremely fast circuitry for serialization logic and large amounts of other circuitry to interface this serial logic to the system's bus. This action is not compatible with equation #1.

The second design possibility is to design the system into sections that require low bandwidths of communications between each section. This design choice allows the use of commonly available components such as

UARTS which are used to serialize and deserialize data communication channels. These MOS VLSI components are easy to interface to the system bus and they support up to eight channels each. These facts make them ideal in meeting the criteria of equation #1, the use of technology that has few failures as shorts on the system bus, and few components are required to implement the function.

Equations #2 through #5 mandate the use of redundant signal interconnections. The redundancy factor (RDF) can reduce the significance of many faults in the interface. A single level of redundancy will minimize the failure rate contributions of equations #2 through #5 by a factor of a thousand while increasing the contribution of equation #1 by only two times.

The actual failure rate for the interface components is not usually under the control of the design engineer. Therefore the failure rate terms IBIAFR,IIBAFR,IIBBFR, and IBIBFR used in equations #1 through #6 provide little guidance in designing new computer architectures, except for the obvious statement of "use the most reliable components available". But equations #2 through #6 do specify the critical path term CP which is under control of the designer.

There are two approaches to making an interface (CP) non critical. The first method is to insure that the functionality on the other side of the interface is not critical to the operation of the system. Typically this would be done by partioning the system into small enough pieces so that the loss of an element is not overly detrimental to the system. The second approach is to use redundancy at a higher level than specifed by the RDF term in the IRA equation. If the system is designed so that all of the elements (BIA,IBA,IBB,BIB) of an interface are replicated, then a given interface could be considered non critical because it is provided with a backup. This backup interface is difficult to realize in a manner that does not contradict the IRA equation and philosophy. Merely installing a second interface in parallel with the first will not yield a design where CP is equal to zero. The bus connection portions of each BIA interface element and the control and decision logic used to choose between them merely create a more complex and less reliable BIA bus connection circuit. If a suitable design can be found, then the CP term will reduce the most of the IRA equation to zero.

To summerize the previous discussion, the IRA equation suggests a system design with a minimum number of low bandwidth serial interfaces to modules that are not too important to the system. The interfaces should be made redundant or non critical. A reasonable design would be a central core processor with low bandwith links to modules that provide the input and output functions for the system. This is much like the Octopus system described in chapter 3.

The Octopus design may be a good start as the basis of the IRA design but it does not meet some of the IRA criteria for optimum reliability. Its primary weakness is the lack of redundancy for the serial interfaces with the I/O modules, and the lack of redundancy for the video interface with the displays. The Octopus design already uses MOS UARTS which are a reliable means of implementing the BIA interface element but simply using multiple channels of UARTS per interface will not yield non-critical interfaces.

The duplication of the system bus with which the BIA element connects would provide the means to insure the complete independence of the redundant interfaces. But the only way to provide duplicate system buses without incurring the cost and handicap of new internal interfaces is to provide the duplicate system bus through the implementation of a duplicate (separate) system.

A system could be built with two octopus main processor boards, operating in parallel, communicating with a quantity of peripheral processors. Each processor will monitor the serial communication channels of the peripheral processors at the same time. The peripheral processor would just provide the usual single channel communications interface to the main processor, to save cost. The main processors would receive in a "Party line" manner, such as in an RS-485 circuit. The RS-485 receivers on the main processors can be effectively isolated from each other with resistors. Carbon composition resistors have failure rates that are less than one hundredth of that of an integrated circuit. The RS-485 transmitters on the main processors may be "wired-or" together at the peripheral processors since they have Tri-State outputs. A better method would be to provide two receivers on the peripheral processors to provide improved isolation between the transmitters on the main processors. Figure 5.1 shows the block diagram of the proposed Dual Octopus System. Through hardware, software, or mechanical means, one processor will be defined as the primary processor and the other as the secondary (or standby) processor. The primary processor will initiate all communications with the peripheral processors. The main processors will work the received data in parallel. They would be running identical software and they may not "know" of each other's existence. They would generate identical outputs to the CRT displays, peripherals, and other devices. Only the "active" processor would be connected to the "real world".

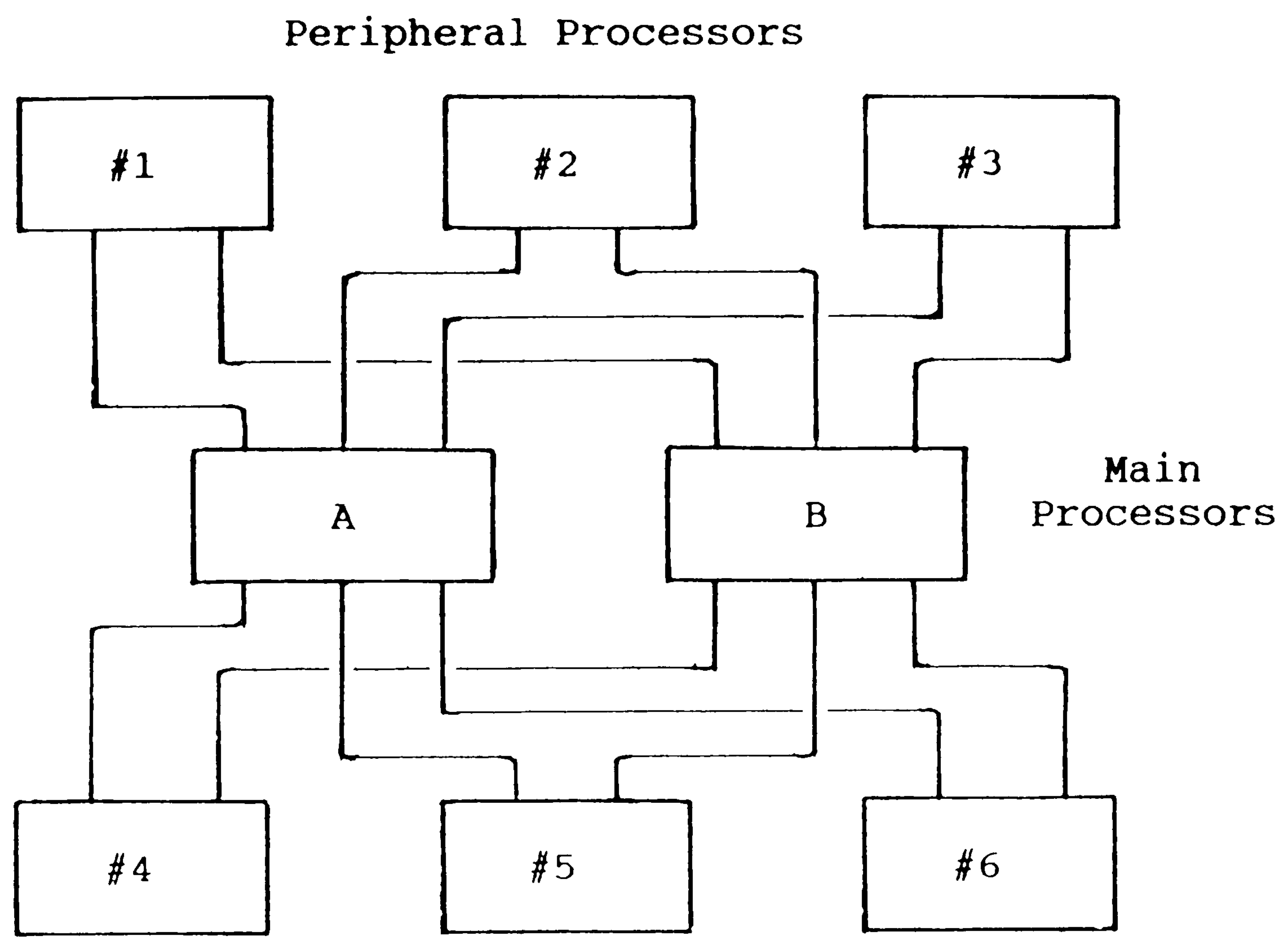


Fig.5.1. Proposed Dual Octopus System

The determination of which processor is the Primary or active processor must be made with reliable circuitry. This circuitry will be very critical to the mission MTBF because its failure could disable both processors. The simplest means is to let the operator decide when a critical fault has occured and flip a switch that will select the outputs of the secondary processor. This backup processor is what is often called a "Hot Standby" backup because it is always running when the primary processor is operating. The hot standby operation allows both processors to maintain the same data bases and insure that they will behave in a similar manner to future events. A more sophisticated means of processor selection could include some voting logic to "throw the switch" automatically.

The decision circuitry used to select the "active" processor becomes an interface subject to the IRA equation! It would appear that we traded the problem of isolating the communications interfaces from each other with the problem of isolating the systems from the voting logic. It may seem like there is no way to avoid the problem of interdependency of interfaces in designing truely redundant interfaces as discussed in the IRA interface element definition. But the other approach to be discussed next relieves us of this difficulty and achieves the maximum possible reliability.

An alternate method for determining which processor is the active processor is not to make that decision on a global level. Instead of switching off the inputs and outputs to an entire processor with a system wide logic circuit, each peripheral processor will decide, by itself, if it believes the main processor is operating properly. If it decides the processor it is communicating with is faulty, it can select to listen to the other main processor. If a particular peripheral board makes a wrong decision, at worst, only that board's functionality is lost to the system. Under normal conditions, both processors are operating, so the peripheral processor will continue to receive commands and supply data to the system.

This decoupling of the voting logic from the main processors highlights a "window of opportunity" allowed by the IRA equation. The IRA equation promotes, with lower calculated failure rates (CP=0), systems that are built with virtual interfaces instead of physical interfaces. The fact that both main processors of a dual octopus system are operating in parallel with no direct interconnection, allows the complete isolation of each other. Their only link to each other is through the commonality of the data they receieve from the outside world. This commonality of data becomes the virtual interface between the two main processors. The peripheral processors are independently evaluating the responses from the main processors and deciding which is worth listing to. The only requirement the design of these peripheral processors must meet to satisfy the IRA's CP term for non criticality is that each peripheral processor does not become so important to the system that its failure will be critical.

The IRA equation appears to suggest that distributed systems provide the best opportunity for the most reliable real time computer system designs. The complete independence between modules must be maintained, merely designing redundancy or modularity into a system will not usually result in a more reliable system.

The design of the Dual Octopus System can be identical with the regular octopus system described in appendix E with the addition of another main processor board. The component count for a dual octopus system is 240 integrated circuits. Implementing the decision making circuitry with the peripheral processors will add no extra cost to the system. The shorted failure mode is rare for bipolar interface ICs which are typically used as RS-485 interface drivers. Therefore, the main processor drivers may be wired or together with little risk of a fault disabling the communications links to the peripheral processors. This eliminates the need for dual receivers on the peripheral processors.

The resultant design has no critical interfaces. The service MTBF works out to as 13,888 hours. The system reliability with no failures in both main processor boards for 2,000 hours is 0.997413. The effective mission MTBF is 772,136 hours or 88 years. If minor failures are tolerable in both main processor boards, then the effective mission MTBF is 1,604 years.

The Dual Octopus System provides the greatest functionality per integrated circuit of any of the model systems. Compared to the Complex bus design, the dual octopus system uses twelve percent less components while providing three hundred and sixty times the mission reliabiliy (over a resonable period of time).

The interface reliability analysis technique with its unique attention to component failure modes has guided the design of the Dual Octopus System. The conventional reliability analysis techniques do not provide any assistance toward the design of realtime computer system architectures.

The increasing component complexity in VLSI devices, the dual octopus system will continue to offer improved performance/price ratios as compared to the other architectures. Most VLSI ICs are limited by the number of pins available in their packages. By definition, the parallel bus buffer or transceiver IC is pin limited, so there is little chance of improving the functionality of such devices. Serial communication ICs are not as constrained by pinout limitations. Parallel interface ICs' primary benefit to system design is their ability to provide high bandwidth resource (primarily memory) expansion, but VLSI memories have reduced this need because large amounts can now be placed onto a single PCB.

Chapter 6

CONCLUSIONS

The standard bus architecture that dominates most Realtime Control System designs is a reasonable design that combines low cost with reasonable reliability. It is possible for a designer to design a more costly and yet less reliable system with out much difficulty.

This thesis has analyzed (for reliability) four multiprocessor bus architectures drawn from various real time control applications. It has also developed a new procedure to carry out such analysis by examining only the system interfaces. This procedure summerized by the Interface Reliability Analysis equation (IRA) is quick, painless and gives results fairly close to the long and tedious conventional analysis. The new Interface Reliability Analysis equation takes into account parallelism, commonality of bus, and other practical limitations of real time control systems. But its most important benefit is its help in designing more reliable systems. The implementation of a given architecture can make or break a design. A design that is spread over many small printed circuit boards will have more internal interfaces and decreased reliability according to IRA. The partitioning of a given implementation across different boards can affect what is considered a critical component, and therefore affecting the mission MTBF. IRA also suggests that real interfaces should be replaced with virtual interfaces and parallel interfaces with serial.

A highly reliable multiprocessor architecture called the "Dual Octopus System" is designed with the help of IRA and it shows that for an additional expenditure of ninety ICs or one thousand dollars, one can improve the system reliability eighty fold.

Other factors that affect the system reliability are the operating temperature and the component reliability. A reduction of a system's ambient temperature by ten degrees C. can double or triple the system's reliability. A temperature reduction of twenty to thirty degrees, can produce larger improvements in system reliability than the extremes in system architecture described in this paper!

The use of proper screening programs for integrated circuits can reduce the expected failure rates for ICs by a factor of at least five as compared to commercial grade integrated circuits. A combination of temperature reduction and component screening can improve the MTBF of a system by twenty times!

The combined improvements of component screening, temperature reduction, careful system design, and redundant critical logic can improve the expected mission system critical MTBF by Three Thousand-Five Hundred times!

APPENDIX A

EQUATIONS FOR COMPONENT RELIABILITY PREDICTION

The Military Handbook " Reliability Prediction of Electronic Equipment" published by the U.S Department of Defense has equations for predicting the reliability of most types of electronic components. The equation for predicting the failure rate for gold plated board to board connectors, at a given temperature, is:

Connector failure rate = Base failure rate x (E x P x K)

Where E represents Environmental conditions

P represents the number of Pins in the connector

K represents number of connector Mating cycles

Assuming a ground based benign environment (E = 3.4) with expected connector mating cycles limited to once per 2,000 hours, a 96 pin connector (such as a VME connector) would have a failure rate of once per eighteen million hours (assuming a base failure rate of 0.00047). This is about six times more reliable than an integrated circuit.
With the same environmental conditions, a 24 pin socket for an integrated circuit would have an expected failure rate of once per 87 million hours. This is twenty five times better than the integrated circuit that will be plugged into the socket.

Good quality interconnections, like passive components such as resistors and ceramic capacitors, do not figure prominently in the expected system reliability.

The MIL-HDBK-217D gives the following equation for predicting the reliability of an integrated circuit:

Failure rate = Q x [C1 x T x V + ((C2+C3) x E)] x L

Where Q represents Device Quality factor

T represents Device temperature acceleration factor

V represents device voltage derating factor

E represents the application environment factor

L represents the device learning curve factor

C1, C2 represent the device complexity factors

C3 represents the package complexity factor

The quality factor describes the benefits of different levels of component screening such as Class S of MIL-M-38510 which is used for many space applications, or Class B of MIL-STD-883 which is used for many military applications. Class S level quality factor is rated at 0.5. Class B, B-0, B-1, B-2 quality levels have quality factors of 1.0, 2.0, 3.0, and 6.5 respectively. Industrial quality components that have been burned in for 160 hours at 125 degrees C have a quality factor of 17.5 and is considered to be at a quality level of D. The typical commercial product with no screening is called level D-1 and has a quality factor of 35.0. The difference in screening levels can affect the calculated system reliability by seventy fold!

The learning factor is either 1.0 or 10.0 depending if the device process is mature or experimental. The application factor describes if the system is ground-based, fixed, mobile, or airborne. This factor considers vibration, temperature cycling, and thermal shock. Except for extreme environments, this factor typically ranges from 2.5 to 6.0. The voltage factor for logic devices is 1.0.

The component complexity factors can be calculated from the number of gates, NG, in the IC as

For Bipolar SSI: C1 = 7.48 x (10 E-4) x (NG ^0.654),
C2 = 2.19 x (10 E-4) x (NG ^0.364).

For CMOS SSI: C1 = 2.17 x (10 E-3_ x (NG ^0.357),
C2 = 3.11 x (10 E-4) x (NG ^0.178).

For MOS LSI: C1 = 1.75 x (10 E-3) x (NG ^0.4),
C2 = 2.52 x (10 E-4) x (NG ^0.226).

For a typical 50 gate bipolar SSI device, C1 would be 0.0097 and C2 would be 0.0009. A five thousand gate LSI device will have C1 = 0.053 and C2 = 0.0017. The complexity of the integrated circuit does not have a strong affect on the device reliability.

The package complexity factor C3 is described by the following equations:

Hermetic DIPs: $C3 = 9.0 \times (10\ E{-5}) \times (NP \wedge 1.51)$

Nonhermetic Dips: $C3 = 2.0 \times (10\ E{-4}) \times (NP \wedge 1.23)$

Where NP is the number of pins.

For 24 pin devices, both hermetic and nonhermetic packages have the same value for C3, 0.010. The package type and size are minor factors in device reliability.

The temperature factor is described by the following equation: $T = 0.1 \exp [-A ((1/(Tj+273)) - (1/298))]$,

where Tj = Junction temperature,

A = 5000 to 6300 for Bipolar, and

A = 7500 to 10,500 for CMOS

The calculated temperature factor for some different technologies and temperatures are shown in Table TA1.

The temperature of the device has a strong affect on the expected device reliability. For bipolar devices, the failure rate doubles and for CMOS, triples for every ten degree C rise in junction temperature.

An integrated circuit screened to 883B levels and used in a typical industrial control environment has the following calculated reliability:

Failure rate = 6.5x[(1.6x.01x1.0)+(.001 +.01)x2.5]x1

Failure rate = 0.283 failures per million hours.

This calculated failure rate is very close to the observed failure rates for ICs as observed by RAC.

The only parameters for device reliability that the system designer has some control of are device screening and the ambient temperature at which the ICs operate.

Table TA1. Temperature factor as a function of temperature and technology

Temperature	LSTTL	NMOS	CMOS
25 C.	0.1	0.1	0.1
35 C.	0.2	0.28	0.31
45 C.	0.38	0.71	0.90
55 C.	0.71	1.7	2.5
65 C.	1.3	4.0	6.3
75 C.	2.2	8.7	15.0

APPENDIX B

COST OF INTEGRATED CIRCUIT SCREENING PROGRAMS

Many IC manufacturers and independent test houses offer burn in and test programs for integrated circuits. The vendors offer a variety of different screening levels depending on the cost/reliability constraints of the customer. Most test houses offer a low cost burn-in program that roughly approximates the military class D screening level. Some test houses offer more expensive screening that approximates military 883B level quality. The actual burn in and test programs vary from vendor to vendor because of cost or capabilities. Table AB1 lists the pricing details of some common ICs with different levels of screening.

The T.I. "3", the Fairchild "QR", the National "B+" and "A+", and the RCA "X" screening programs approximate a military "D" quality level which is about twice as good as an untested commercial grade part. The RCA "F3", the National "/883", and the T.I."54 series" screening are "B" quality level programs that have quality levels thirty five times better than the unscreened product. The level "D" screening adds, on an average, eleven cents to the base price of each component. The "B" level screening programs increase the cost, on the listed components, by

Table AB1. Effect of reliability screening IC price.

Texas Instruments

SN74LS00N	$0.25	SN74LS138N	$0.38
SN74LS00N3	$0.38	SN74LS138N3	$0.52
SN54LS00J	$0.83	SN54LS138J	$1.36
SN74LS240N	$0.52	SN74LS299N	$1.34
SN74LS240N3	$0.72	SN74LS299N3	$1.54
SN54LS240J	$2.72	SN54LS299J	$4.45
SN74LS373N	$0.52		
SN74LS373N3	$0.72		
SN54LS373J	$3.44		

FAIRCHILD

74AC00PC	$0.32	74F109PC	$0.40
74AC00PCQR	$0.42	74F109PCQR	$0.50
74F541PC	$2.14	74F574PC	$2.78
74F541PCQR	$2.28	74F574PCQR	$2.93

NATIONAL SEMICONDUCTOR

CD4001BCN	$0.33	CD4031BCN	$2.77
CD4001BCN/B+	$0.37	CD4031BCN/B+	$2.81
CD4001BCN/A+	$0.45	CD4031BCN/A+	$2.89
CD4001BMJ/883	$1.50	CD4031BMJ/883	$9.30

RCA SOLID STATE PRODUCTS

CD74HCT00E	$0.33	CD74HCT245E	$1.22
CD74HCT00EX	$0.53	CD74HCT245EX	$1.42
CD54HCT00F	$0.64	CD54HCT245F	$3.18
CD54HCT00F3	$1.74	CD54HCT245F3A	$9.57
CD54HCT00F3A	$1.71		
CD74HCT373E	$1.16		
CD74HCT373EX	$1.35		
CD54HCT373F	$2.48		
CD54HCT373F3	$7.25		
CD54HCT373F3A	$7.18		

three hundred and twenty-four percent! If the reliability of the quality levels quoted in MIL-HDBK-217D are to be believed, then the large additional cost of "B" level screening buys a very large increase in reliability. Unfortunately, most commercial applications for real time computers can not absorb a tripling in system component costs.

APPENDIX C

SAMPLE REAL TIME CONTROL SYSTEMS

This appendix lists the chip count, board area, the number of connectors, and the cost for the printed circuit boards of some realtime industrial control systems produced by Bridgeport-Textron and North American Drager, as well as PCBs produced by other firms. This list will provide valuable costing information. The costs indicated represent the cost to manufacture or buy at OEM discounts. A brief description of each system will be provided as insight into the actual implementation of a real time control system.

The first system to be analyzed is the Narkomed 3 (NM3), an anesthesia monitoring system implemented with eleven Z80 microprocessors. The NM3 has two CRT displays, six patient monitors for such measurments as breathing pressure, volume, gas analysis, blood pressure, and blood oxygen saturation, an internal communications controller, and an external communications controller. The NM3 has a microprocessor allocated for each function and they communicate over several eight bit communication buses. The NM3 is a distributed Peer processing system. It is similar to the communications bus design example described in this thesis and its cost is detailed in Table AC1.

Table AC1. Cost analysis of NM3 system based on 11 Z80 microprocessors

BOARD	SIZE (si)	CHIP-COUNT	CONNECTORS	COST
Monitor CPU	49.5	28	5	$115
Monitor Interface	25.0	8	3	$105
Display	10.5	2	2	$ 60
Each Monitor	85.0	38	10	$280
Total Monitors (6)	510.0	228	60	$1,680
CRT Controller	49.5	35	5	$150
Total CRTs (3)	148.5	105	15	$450
Internal Communications Controller	72.3	43	16	$269
External Communications Controller	66.0	43	10	$437
Interfaces	45.75	25	13	$292
Backplanes (2)	45.0	0	10	$60
Total	887.55	444	124	$3,188

```
Cost/Chip      = $7.18
S.I./Chip      = 1.99
% Display I/O = 26%
% Machine I/O = 16%
% Total   I/O = 42% of Total #IC
```

NOTE: This information was obtained from internal N.A.D. documents and was used with N.A.D. approval.

Analysis of the R2E4, a four axis real time control system for milling machines produced by Bridgeport-Textron is provided in Table AC2. The R2E4 is implemented with two MC68000 microprocessors that are built into a Versabus backplane but they communicate with each other over a serial communications channel. Each MC68000 processor also communicates with single chip microcontrollers over serial links. The system features a CRT display for the

---

Table AC2. Cost analysis of the R2E4 system with two 68000 microprocessors

BOARD	SIZE (si)	CHIP-COUNT	CONNECTORS	COST
CPU - user	130.5	129	2	$1,200
CRT controller	130.5	108	2	$700
User Interface	78.0	15	4	$200
CPU - motors	130.5	72	15	$1,000
Machine interface	128.0	18	12	$300
Backplane	52.0	0	10	$400
Power Up control	32.0	3	5	$100
Total	681.5	345	50	$3,900

Cost/IC = $11.30
S.I./IC = 1.97
% Display I/O = 32%
% Machine I/O = 24%
% Total I/O = 56% of Total #IC

NOTE: This information was obtained from internal Bridgeport-Textron documents and was used with Bridgeport approval.

---

operator and closed loop DC servo motor control of the machine's axes of motion. One MC68000 controls the axis drives while the other handles the operator interface.

One could also use the VME bus boards directly available from an OEM manufacturers to build real time industrial control systems. Pricing structure of such boards is indicated in table AC3.

Table AC3. Cost data for the OEM boards.

BOARD	SIZE (si)	CHIP-COUNT	CONNECTORS	COST
VME CPU (Plessey #68-25)	57.5	110	2	$1,481
VME ROM Memory (DY-4 #DVME-505)	57.5	84	2	$928
VME CRT Controller (Eltec #OPAC)	57.5	69	7	$2,170
VME Serial I/O (Plessey #PMESIO-3)	57.5	78	3	$840
VME Analog I/O (Xycom XVME-540)	57.5	60	4	$1,260
VME Backplane	50.0	0	12	$350
Total	337.5	401	30	$7,029

```
  Cost/IC       = $17.52
  S.I./IC       = 0.84
% Display I/O = 17%
% Machine I/O = 34%
% Total   I/O = 67% of Total #IC
```

Note: The VME board prices include a typical discount of 30% available to high volume users.

The combined average cost and size statistics for the NM3, R2E4, and the OEM VME systems are:

Cost/IC		= $11.86
S.I./IC		= 1.60
% Display	I/O	= 26%
% Machine	I/O	= 16%
% Total	I/O	= 42% of Total #IC

APPENDIX D

MICROPROCESSOR PERFORMANCE COMPARISONS

Determining the relative computing power of microprocessors that span different architectures and generations and are made by different manufacturers is an inexact science. While it is possible to contruct benchmark programs for specific applications, any general statements are hard to make because of the varied applications in which microprocessors are used. However, an attempt must be made to provide a means to judge the relative cost of different system architectures.

Table AD1 was made with a few assumptions. The processors are placed in the chart according to the typical parts used in typical systems. After a part is released for production by a manufacturer, newer, faster parts will be produced in the following years. Sometimes specially selected high speed parts are available from the manufacturer, these only blur the line between succeeding generations of processors and so do not appear on this performance comparison. The comparison assumes assembly language or "C" programming because many high level language compilers can have huge disparities in the execution speed of their generated output code. This chart was constructed with data from a variety of sources [20 - 33]. Differences of ten to one for execution speed are common for a set of compilers for a given language (such as "C") for a given processor (such as the 68000). The system configurations and resources available to a processor can help or hinder its performance relative to its competitors.

Table AD1. Comparison of various processors

Performance	Processor Type
1.0	6800, 6502 (1 mhz), 8085 (2 mhz)
3.0	6809 (2 mhz), Z80 (4 mhz)
3.5	8088 (5 mhz)
4.5	Z8000 (4 mhz)
6.0	8086 (8 mhz)
9.0	32032 (10 mhz)
10.0	68000, 80286 (10 mhz)
11.0	Z8000, 68010 (10 mhz)
22.0	Vax 11/780
30.0	80386 (16 mhz)
35.0	68020 (16 mhz)
70.0 estimated	80486,68030 (30 mhz)

Table AD1 assumes a typical workload that involves mostly nonarithmetic operations.

If a large arithmetic work load is expected, the results of the chart must be shifted in favor of the 32 bit microprocessors. Using the software library for the Z80 that was produced by Pro-Log Corporation, a 4 mhz Z80 will perform as follows:

Divide	(32 bit by 16 bit)	=	410 microseconds
Multiply	(32 bit by 32 bit)	=	4,772 microseconds
Divide	(32 bit by 32 bit)	=	3,931 microseconds

The 68000 (10 mhz) will perform the 32 bit by 16 bit divide in 12.2 microseconds which is 33 times faster than the Z80. The 68020 (16 mhz) will perform the 32 bit multiply in 2.75 microseconds which is 1,735 times faster than the Z80! The 68020 will perform the 32 bit divide in 5.625 microseconds which is 698 times faster than the Z80.

APPENDIX E

MODEL SYSTEMS' DESIGN DETAILS

The representative systems for the Standard Bus, the Complex Bus, the Communication Bus, and the Octopus System are designed with the same generation of integrated circuits. The memory chips are 256K bits in size. The read/write (RAM) memory chips are organized as 32k bytes by 8 bits and are static, not requiring logic to perform refresh operations. The processors' program memory is based on EPROM technology and is also organized as 32Kx8. The use of byte wide memories allows the designer to interchange EPROM with pin compatible RAM.

The video refresh memory is made up of 64K x 4 dynamic RAMs with specialized logic to aid the design of video systems. These specialized memories are called video RAMs (VRAMs). Most modern video controllers are designed to interface with these new devices. VRAMs incorporate shift registers used to shift out video information while allowing the processor to access the memory array in a random access manner. The VRAMs are considered dual port memories.

The majority of the control logic used in these systems is designed with PLDs (programmable logic devices) to reduce total chip counts. The use of PLDs also helps to preserve design freedom by allowing modifications to be made to the hardware design late in the design phase of the project. It is assumed in this thesis that all bus interface chips are eight bits per package.

Standard Bus Example

The design for the standard bus system is based on the Versabus architecture that was promoted by Motorola Inc.. The Versabus was the forerunner of the VME bus that is promoted by Signetics, Motorola, and others. The versabus board size is large (120 square inches) which is an aid in reducing the number of boards needed to build a system.

The system is divided into two boards, a processor board, and an I/O board. The two boards are connected with a 32 bit address bus, a 32 bit data bus, and 57 control signals, most of which are not used in the model under discussion. The chip count in the design situation being analysed here is only 152 integrated circuits. The serial interfaces are implemented on the processor board to balance the component counts between the two boards. The serial interfaces on the processor board also aid the development and testing of the processor board by allowing processor board to function as a stand alone entity and still communicate with other test equipment.

The standard bus system devotes forty percent of the system to input and output functions. The interface between the two boards costs more than twenty-two percent of the system's resources. The component list for the standard bus system under discussion is provided in tables AE1 and AE2.

Table AE1. PARTS for STANDARD BUS

Processor Board

1. (1) MC68020 - 32 bit Microprocessor
2. (1) 25 mhz crystal oscillator - processor clock
3. (16) 27C256 - 32K x 8 Eprom, program storage
4. (16) TC55257P - 32k x 8 static RAM, data storage
5. (5) GAL20V8 - PLD - Control signal, memory decode
6. (4) GAL20V8 - PLD - Memory control logic
7. (1) GAL20V8 - PLD - Bus time out logic
8. (1) 68901 - Interrupt controller, system timers
9. (2) GAL20V8 - PLD - Interrupt control
10. (1) RTC62421 - Real time clock
11. (1) 16 mhz crystal oscillator for system timing needs
12. (1) 74F191 - counter to provide lower rate clocks
13. (1) DS1232 - Power up reset generator
14. (1) 74ALS541 Data bus buffer for RTC,68901, CLD-180
15. (1) CL-CD180 - Octal Uart for serial communications
16. (1) 9.8304 Mhz crystal oscillator for baudrate clocks
17. (2) 26LS31 - RS-422 drivers for serial ports
18. (2) 26LS33 - RS-422 receivers for serial ports

19. (3) GAL20V8 - PLD - Bus buffer control
20. (4) 74F545 - 8 bit Bidirectional Transceiver, data bus
21. (4) 74F545 - 8 bit Transceiverr, address bus
22. (2) 74F541 - 8 bit Buffer, basic bus control signals
23. (1) 74F541 - 8 bit Buffer, DMA control signals
24. (2) 74F541 - 8 bit Buffer, Interupt control signals

----

76 ICs

Table AE2. Parts for Standard Bus Input/Output Board

1. (1) 95C60 - QPDM - Video Controller
2. (16) uPD41264 - 64K x 4 Video RAM
3. (4) 74F676 - 16 bit high speed video shift registers
4. (1) 57.83Mhz crystal oscillator for 800x600 video
5. (1) 74AS2645 - 10 bit buffer, video RAM address
6. (3) GAL16V8 - PLD, video logic and buffers
7. (5) 74ALS541 - 8 bit buffer for digital inputs, bus
8. (8) MCT6 - Dual optocouplers to isolate digital inputs
9. (5) 74ALS574 - 8 bit register for digital outputs
10. (2) ULN2823 - 8 bit high power output buffers
11. (1) ADC1005 - 10 bit ADC, system test, analog inputs
12. (2) 74HCT4067 - 16 input Analog multiplexer
13. (5) LT1014 - Quad Op Amps for analog
14. (1) X2001 - 128 x 8, NOVRAM, for calibration data
15. (1) SAA1099 - Sound generator for audio alarms
16. (1) LM386 - audio amplifier
17. (7) GAL20V8 - PLD, address and control decode
18. (4) 74F545 - 8 bit Bidirectional Transceiver, data bus
19. (4) 74F545 - 8 bit transceiver, address bus
20. (2) 74F541 - 8 bit Buffer, basic bus control signals
21. (1) 74F541 - 8 bit Buffer, DMA control signals
22. (2) 74F541 - 8 bit Buffer, Interupt control signals

----

76 ICs

The MC68020 microprocessor is a 32 bit machine with a perfomance of approximately 3.5 million instructions per second (MIPS) when operated at 25 Mhz. The large number of PLDs used for the control and decode logic is required by the necessity to operate at high speed with fewer levels of logic.

The 68901 is an integrated peripheral IC that includes four timers that can be used to provide system clocks used by the software. The 68901 also provides interrupt control logic. The RTC62421 is a time of day clock that also keeps track of days, months, and years. It is powered by a battery or super capacitor (1 Farad) to provide non volatile time keeping for the system. The TOD clock is important for real time control systems that keep track of system run time for warrenty needs.

The CL-CD180 is a VLSI device that has eight independent UARTS. Each UART has its own baud rate generator and hardware hand-shaking control signals (if needed). The chosen hardware interface for the serial ports is RS-422 because of its superior noise immunity as compared to RS-232.

The 74F541s and 74F545s are eight bit buffers used to provide the signal drive levels needed for reliable communication between the system's boards over the backplane PCB.

The Input/Output board is divided into three sections, the video controller, the general I/O, and the bus interface. The video logic is implemented with a AMD 95C60 video coprocessor and the video RAMs.

The general I/O has a mixture of parallel digital I/O that has optically isolated inputs for increased noise immunity, and buffered outputs capable of driving industrial loads such as solenoids and relays.

The X2001 is a nonvolatile memory made up of RAM and EEPROM that shadow each other. On powerup, data is copied from EEPROM to RAM automatically. The contents of RAM can be copied into EEPROM with a single command signal. The NOVRAM is used to store calibration data, user and system configuration information.

Complex Bus Example

The design example for the Complex Bus System is based on the extended VME bus specification that includes the VMX bus as a local or private bus used to access private resources. The VME bus features a 32 bit data bus, a 32 bit address bus, and 43 control signals to access global resources. The VMX bus features a 32 bit data bus, a 24 bit address bus that is multiplexed onto 12 lines, and 8 control signals.

The design example for the complex bus being discussed here is divided into five boards. There are two identical processor boards that may or may not execute the same software. There are two I/O boards, one for each processor, that communicate with their processor over their private VMX buses. These boards are described as being identical but they may have different configurations of I/O hardware as needed by the application. Table AE3, AE4, and AE5 give the detailed component lists for this architecture.

Table AE3. Parts for Complex Bus Processor Board

(one of two)

1. (1) MC68020 - 32 bit Microprocessor
2. (1) 12.5 mhz crystal oscillator - processor clock
3. (12) 27C256 - 32K x 8 Eprom, program storage
4. (12) TC55257P - 32k x 8 static RAM, data storage
5. (1) 68901 - Interrupt controller, system timers
6. (5) GAL20V8 - PLD - Control signal, memory decode
7. (2) GAL20V8 - PLD - Memory control logic
8. (2) GAL20V8 - PLD - Interrupt control
9. (1) GAL20V8 - PLD - Bus time out logic
10. (1) 16 mhz crystal oscillator for system timing needs
11. (1) 74F191 - counter to provide lower rate clocks
12. (1) DS1232 - Power up reset generator
13. (1) RTC62421 - Real time clock

VMX (private bus interface)

14. (1) GAL20V8 - PLD - Bus buffer control
15. (4) 74F545 - 8 bit Transceiver, 32 bit data bus
16. (4) 74F541 - 8 bit Buffer, multiplexed address bus
17. (2) 74F541 - 8 bit Buffer, basic bus control signals

VME global bus interface

18. (2) GAL20V8 - PLD - Bus buffer control
19. (4) 74F545 - 8 bit Transceiver, 32 bit data bus
20. (4) 74F545 - 8 bit transceiver, 32 bit address bus
21. (2) 74F541 - 8 bit Buffer, basic bus control signals
22. (2) 74F541 - 8 bit Buffer, DMA control signals
23. (2) 74F541 - 8 bit Buffer, Interrupt control signals

68 ICs

Table AE4. Parts for the Complex Bus CRT Board

1. (6) GAL20V8 - PLD, address and control decode
2. (4) GAL20V8 - PLD, processor contention control
3. (2) CY7C412 - FIFO, processor contention control
4. (1) DS1232 - Dead processor time out
5. (1) GAL16V8 - PLD, dead processor timeout logic
6. (1) 95C60 - QPDM - Video Controller
7. (16) uPD41264 - 64K x 4 Video RAM
8. (4) 74F676 - 16 bit high speed video shift registers
9. (1) 57.83Mhz crystal oscillator for 800x600 video
10. (1) 74AS2645 - 10 bit buffer, video RAM address
11. (5) GAL16V8 - PLD, video logic and buffers
12. (4) 27C256-12 - 32Kx8 Eprom, Common CRT Displays
13. (4) TC55257P - 32k x 8 RAM, shared data storage
14. (4) 74F545 - 8 bit Bidirectional Transceiver, data bus
15. (4) 74F545 - 8 bit transceiver, address bus
16. (2) 74F541 - 8 bit Buffer, basic bus control signals
17. (2) 74F541 - 8 bit Buffer, DMA control signals
18. (2) 74F541 - 8 bit Buffer, Interupt control signals

64 ICs

Table AE5. Parts for the Complex Bus I/O BOARD

1. (2) 74ALS541 - 8 bit buffer for digital inputs
2. (4) MCT6 - Dual optocoupler, to isolate digital inputs
2. (3) 74ALS574 - 8 bit register for digital outputs
3. (2) ULN2823 - 8 bit high power output buffers
4. (1) ADC1005 - 10 bit Analog to digital converter
5. (1) 74HCT4067 - 16 input Analog multiplexer
6. (3) LT1014 - Quad Op Amps for analog inputs
7. (1) SAA1099 - Sound generator for audio alarms
8. (1) LM386 - audio amplifier for audio alarms
9. (2) 68681 - Dual Uarts for serial communications
10. (1) 3.6864 Mhz crystal oscillator for baudrate clocks
11. (1) 26LS31 - RS-422 drivers for serial ports
12. (1) 26LS33 - RS-422 receivers for serial ports
13. (1) X2001 - 128 x 8 NOVRAM, for calibration data
14. (7) GAL20V8 - PLD, address and control decode
15. (4) 74F545 - 8 bit Bidirectional Transceiver, data bus
16. (4) 74F541 - 8 bit Buffer, address bus
17. (1) 74F541 - 8 bit Buffer, bus control signals

39 ICs

The processors share a common video board that also provides some common program (Eprom) and data (Ram) memory. The RAM memory serves as a means of communication between the two processors. The EPROM allows either processor to provide video displays without paying for the memory twice. Because the resources on the video board are shared between two processors, there is a requirement to provide a means to resolve conflicts if the processors desire to access a resource at the same time. The QPDM video controller is actually a video co-processor that perform complex tasks such as "fill bounded region". The QPDM requires a block of data to perform an operation, it would get confused if two processors wrote data or commands to it at the same time.

The design example for the complex bus has a component count of 270 integrated circuits. The I/O circuitry represents 39% of the system total. The bus interface circuitry consumes 33% of the total system.

The processor board for the complex system is very similar to the processor board of the standard bus. To make room for the VMX bus interface, the serial interfaces are moved off the processor card. This lack of serial ports on the processor card makes the complex bus cpu board more difficult to design and test because now one needs to have an operational backplane and an I/O card to communicate with the processor board.

Some cards of the complex bus are simple. For example, the CRT board features only a VME interface and the I/O boards only have VMX bus interfaces.

To reduce the component cost, the UARTs are implemented with 68681 dual UARTs. There is sufficent PCB real estate to absorb the larger component count on the I/O card.

Communications Bus Example

The design of the communications bus architecture is based on a patient monitoring system designed and built by North American Drager. The system is composed of four sections, the I/O modules, the display controllers, the internal communications controller (ICC) and the external communications controller (ECC).

The I/O modules are self contained units that can monitor and/or control subsections of the system. The I/O modules communicate with the ICC over an eight bit data bus that is controlled with only eight control signals. The I/O modules are periodically polled by the internal communications controller for data and status, and commands for new modes of operation are issued by the central bus controller as needed. The I/O modules are indentical to each other but the actual I/O circuitry depends on the application. The I/O modules are equipped with serial interfaces if there are internal sensors or modules that communicate with serial links. Detailed parts lists for the design under consideration are presented in tables AE6 - AE9.

Table AE6. Parts for Communication Bus

		Internal Communication Controller
1.	(1)	Z80A - 8 bit microprocessor
2.	(1)	4 mhz crystal oscillator
3.	(1)	27C256 - 32K x 8 Eprom - program storage
4.	(1)	TC55257P - 32k x 8 static RAM, data storage
5.	(1)	X2001 - 128 x 8 NOVRAM, calibration data
6.	(5)	GAL20V8 - PLD, Address decoding, bus(s) control
7.	(1)	Z8430A - General purpose counter/timer
8.	(1)	SAA1099 - Sound generator for audio alarms
9.	(1)	LM386.- audio amplifier
10.	(1)	RTC62421 - Real time clock
11.	(1)	DS1232 - Power up reset/ watchdog timer
		Interface to I/O Processors
12.	(1)	CY7C412 - FIFO, processor command buffer
13.	(2)	75ALS160 - 8 bit transceiver, data and control
14.	(1)	74ALS990 - 8 bit latch, bus interface
15.	(1)	74ALS244 - 8 bit buffer, bus interface
		Interface to CRT Processor
16.	(2)	74ALS245 - 8 bit transceiver, data and control
17.	(1)	74ALS244 - 8 bit buffer, status from CRTs
		Interface to Serial Interface Processor
18.	(1)	- Dual port mem, interface to serial I/F
19.	(2)	74ALS245 - 8 bit transceiver, data and control
20.	(2)	74ALS244 - 8 bit bfr, Address bus, dual port mem
	28	ICs

Table AE7. Parts for the CRT Controller (one of two)

```
 1. (1) Z80A - 8 bit microprocessor
 2. (1) Z8430A - General purpose counter/timer
 3. (1) 27C256 - 32K x 8 Eprom - program storage
 4. (1) TC55257P - 32k x 8 static RAM, data storage
 5. (1) 4 mhz crystal oscillator
 6. (2) GAL20V8 - PLD, Address/control decoding
 7. (1) TMS34061 - Video timing and memory controller
 8. (2) uPD41264 - Video RAM
 9. (1) GAL16V8 - PLD, video timing logic
10. (1) GAL20V8 - PLD, video control logic
11. (1) 74F199 - Video shift register
12. (1) 57.3 mhz crystal oscillator, video timing
13. (1) CY7C412 - FIFO, processor command buffer
14. (2) 75ALS160 - 8 bit transceiver, data and control
15. (1) 74ALS990 - 8 bit latch, bus interface
16. (1) DS1232 - Power up reset/ watchdog timer
  ----------
     19  ICs
```

Table AE8. Parts for the Input/Output Module (one of six)

```
 1. (1) Z80A - 8 bit microprocessor
 2. (1) Z8430A - General purpose counter/timer
 3. (1) 27C256 - 32K x 8 Eprom - program storage
 4. (1) TC55257P - 32k x 8 static RAM, data storage
 5. (1) 4 mhz crystal oscillator
 6. (2) GAL20V8 - PLD, Address/control decoding
 7. (1) GAL16V8 - PLD, Bus interface control
 7. (2) 75ALS160 - 8 bit transceiver, data and control
 8. (2) 74ALS990 - 8 bit latch, bus interface
 9. (1) 74ALS244 - 8 bit buffer, bus interface
10. (2) 74ALS244 - 8 bit buffer, digital inputs
11. (4) MCT6 - Dual optocoupler, to isolate digital inputs
12. (1) 74ALS990 - 8 bit latch, digital output
13. (1) ULN2823 - 8 bit high power output buffers
14. (1) ADC1005 - 10 bit Analog to digital converter
15. (1) 74HC4051 - 8 input analog multiplexer
16. (2) LT1014 - Quad Op Amps for analog
17. (1) DS1232 - Power up reset/ watchdog timer
    -----------
     26  ICs
```

Table AE9. Parts for External Serial Communications Controller

1.	(1)	Z80A - 8 bit microprocessor
2.	(1)	Z8430A - General purpose counter/timer
3.	(1)	27C256 - 32K x 8 Eprom - program storage
4.	(1)	TC55257P - 32k x 8 static RAM, data storage
5.	(1)	4 mhz crystal oscillator
6.	(2)	GAL20V8 - PLD, Address/control decoding
7.	(2)	Z8530 - Dual Uart
8.	(1)	26LS31 - RS-422 Driver
9.	(1)	26LS33 - RS-422 Receiver
10.	(1)	GAL16V8 - PLD, dual port memory control
11.	(2)	74ALS245 - 8 bit transceiver, data and control
12.	(2)	74ALS244 - 8 bit bfr, Address bus, dual port mem
13.	(1)	DS1232 - Power up reset/ watchdog timer
	17 ICs	

The internal communications controller is the hub of the system. It controls the I/O bus that links the I/O modules to the system. It controls the display bus that links the CRT controllers to the system. It has an interface with the external communications controller to send and receive data to the outside world. The primary purpose of the ICC is to gather data from the I/O modules for display on the CRTs.

The CRT controllers are connected to the ICC with the eight bit display bus. The CRT controllers are passive devices that display what is available on the display bus; the data flows one way, out to the displays.

The ECC is interfaced with the ICC with a dual port memory. The data is exchanged in packets of messages. The ECC is responsible for translating external protocols into a format that is understood by the ICC and vice a versa.

The system modules are implemented with Z80s, they are 8 bit microprocessors. To approximate a 32 bit cpu requires about eleven Z80s. The design example uses 10 Z80s which will work only if the tasks can be properly partitioned.

The communication bus design example's component count of integrated circuits is 222. The I/O circuitry represents 40% of the system. The interface circuitry consumes only 28% of the of the total system.

The communications design example may be "under-powered" with respect to the other design examples. The ICC is the logical module to be responsible for guideing the operation of the system. The ICC is specified with an eight bit processor to minimize the system component count. The ICC is heavily burdened with the task of running the system communication channels. The ICC has little time leftover to interpret the system behavior. Another module, the System Interpreter Module (SIM) is generally required for systems that are expected to interact with their environment instead of just reporting on the environment. The SIM is also connected to the ICC through a shared memory port.

The functionality of the SIM could be included into the ICC if the ICC were implemented with a more powerful processor. This would then reduce the stature of the other processor modules to "Slaves" as the ICC became the de facto "Master" of the system. To maintain the distributed peer processing system architecture, each of the processing modules may have to be scaled up in size and performance. A simple conversion to sixteen bit processors would increase the system cost by approximately twenty percent due to the doubling of the memory costs because of the wider data bus, and the additional width of the communication interfaces.

Octopus System Example

The main processor board of the Octopus system is similar to one in the standard bus design example. The primary change was the deletion of the bus interface logic and the addition of the video display logic. In many systems, the highest bandwidth requirements are between the processor and the video memory. By placing the video logic on the same board as the processor, the other I/O bandwidth requirements should be low enough to be supported by a serial link between the I/O device and the processor. If another I/O task also requires a high bandwidth link to the processor, then that device should be given its own processor to support that I/O task in such a way that the bandwidth requirements between the two processors can be reduced.

This design features half a dozen I/O modules based on a single chip microcontroller that can communicate with the main processor over RS-422 serial links. The 68HC811 microcontroller contains its own EEPROM, RAM, UART, ADC, powerup reset logic, and parallel I/O. The micro-controller can preprocess the data into a form that is more manageable for the main processor.

The octopus system example has a component count of 153 integrated circuits as shown in Tables AE10 and AE11. The I/O circuitry represents 58 percent of the system total. The interface logic is only 8% of the system.

The peripheral processors do not add significantly to the net overall processing power of the system. The single chip microcontrollers are about as powerful as the first generation microprocessors. They can perform preprocessing of data for the main processor but the main processor will spend additional resources to support the communication routines that are used to communicate with the peripheral processors.

Table AE10. Parts for Octopus System processor board

Processor core

1. (1) MC68020 - 32 bit Microprocessor
2. (1) 25 mhz crystal oscillator - processor clock
3. (16) 27C256 - 32K x 8 Eprom - program storage
4. (16) TC55257P - 32k x 8 static RAM, data storage
5. (5) GAL20V8 - PLD - Control signal, memory decode
6. (4) GAL20V8 - PLD - Memory control logic
7. (2) GAL20V8 - PLD - Interrupt control
8. (1) 68901 - Interrupt controller, system timers
9. (1) GAL20V8 - PLD - Bus time out logic
10. (1) RTC62421 - Real time clock
11. (1) DS1232 - Power up reset generator

Video controller

12. (1) 95C60 - QPDM - Video Controller
13. (16) uPD41264 - 64K x 4 Video RAM
14. (4) 74F676 - 16 bit high speed shift registers, video
15. (1) 57.83Mhz crystal oscillator for 800x600 video
16. (3) GAL16V8 - PLD, video logic and buffers
17. (1) 74AS2645 - 10 bit buffer, video RAM address

Serial interface

18. (2) CL-CD180 - Octal Uart for serial communications
19. (2) 9.8304 Mhz crystal oscillator for baudrate clocks
20. (4) 26LS31 - RS-422 drivers for serial ports
21. (4) 26LS33 - RS-422 receivers for serial ports

----

87 ICs

Table AE11. Parts for Peripheral Processor Board

(one of six)

1. (1) 68HC811A2 - 8 bit single chip microcontroller
2. (1) 7.3728 mhz crystal oscillator, processor clock
3. (1) NMC9346E - serial EEPROM for calibration data
4. (1) DS8921 - RS422 Driver and Receiver
5. (1) 74HC4051 - 8 input analog multiplexer
6. (4) MCT6 - Dual optocoupler, to isolate digital inputs
7. (1) ULN2823 - 8 bit high power output buffers
8. (1) LT1014 - Quad Op Amps for analog

----

11 ICs

The primary software benefit of the peripheral processors for the main processor will be the reduction of "Bit Banging I/O". The microcontrollers can make the I/O devices appear to be more standardized and idealized for the main processor. This would allow the main system software to be programmed by personnel who are not educated in hardware design.

REFERENCES

1. S. Bavuso, J. Dugan, K. Trivedi, E. Rothmann, and W. Smith, "Analysis of Typical Fault-Tolerant Architectures using HARP", IEEE Transactions on Reliability, Vol.R-36, No.2, pp.176-185, June 1987.

2. B. Johnson and J. Aylor, "Reliability & Safety Analysis of a Fault-Tolerant Controller", IEEE Transactions on Reliability, Vol.R-35, No.4, pp. 355-362, October 1986.

3. H. Hecht and H. Dussault, "Correlated Failures in Fault-Tolerant Computers", IEEE Transactions on Reliability, Vol.R-36, No.2, pp.171-175, June 1987.

4. R.T. Anderson, "Reliability Design Handbook", Reliability Analysis Center, Rome Air Development Center, Griffiss AFB, N.Y., March 1976.

5. USA Department of Defense, "Military Handbook - Reliability Prediction of Electronic Equipment - Mil-HDBK-217D", April 1979.

6. Intel Corporation, "Components Quality/Reliability Handbook" Santa Clara, CA., 1985.

7. Mitsubishi Electric Corporation, "Mitsubishi Semiconductor Reliability Hand Book", Tokyo, Japan, 1982.

8. K. Dey, "Practical Statistical Analysis for the Reliability Engineer", Reliability Analysis Center, Rome Air Development Center, Griffiss AFB, N.Y., 1983.

9. T. Green, "A Review of IC Fabrication Design and Assembly Defects Manifested as Field Failures in Air Force Avionic Equipment", IEEE - 26th Annual Proceedings Reliability Physics 1988, pp.226-229, April 1988.

10. M. Priore, "IC Quality Grades: Impact on System Reliability and Life Cycle Cost, Reliability Analysis Center, Rome Air Development Center, Griffiss AFB, N.Y., 1985.

11. S. Stockman and D. Rash,"Microcircuit Device Reliability Trend Analysis 1985", Reliability Analysis Center, Rome Air Development Center, Griffiss AFB, NY.

12. J. Kuhl and S. Reddy, "Fault-Tolerance Considerations in Large, Multiple Processor Systems", Computer, Vol.19, No.3, pp.56-67, March 1986.

13. D. Siewiorek, "Architecture of Fault-Tolerant Computers", Computer, Vol.17, No.8, pp.9-18, August 1984.

14. O. Serlin, "Fault-Tolerant Systems in Commercial Applications", Computer, Vol.17, No.8, pp.19-39, August 1984.

15. H. Ihara and K. Mori, "Autonomous Decentralized Computer Control Systems", Computer, Vol.17, No.8, pp.57-66, August 1984.

16. A. Avizienis and J. Kelly, "Fault Tolerance by Design Diversity: Concepts and Experiments", Computer, Vol.17, No. 8, pp.67-80, August 1984.

17. E. Irland, "Assuring Quality and Reliability of Complex Electronic Systems: Hardware and Software", Proceedings of the IEEE, pp.5-18, Jan. 1988.

18. S. Yates and R. Williams, "A Fault-Tolerant Multiprocessor Controller for Magnetic Bearings", IEEE Micro, pp.6-17, August 1988.

19. R. Negrini, M. Sami, and R. Stefanelli, "Fault Tolerance Techniques for Array Structures Used in Supercomputing", Tutorial: Fault-Tolerant Computing, IEEE Computer Society Press, Washington DC., pp.403-412, 1987.

20. M. Linton, "Benchmarking Engineering Workstations", IEEE Design & Test, pp.25-30, June 1986.

21. T. Cooper, W. Bell, F. Lin, and N. Rasmussen, "A Benchmark Comparisons of 32-bit Microprocessors", IEEE Micro, pp. 53-57, August 1986.

22. R. Grappel and J. Hemenway, "A tale of four uPs: Benchmarks quantify performance", EDN, Vol.26, No.7, pp.179-265, April 1981.

23. J. Gilbreath, "A High-Level Language Benchmark", Byte, Vol.6, No.9, pp.180-198, Sept. 1981.

24. G. Williams, "Benchmarking the Intel 8086 and 8088", Byte, Vol.8, No.7, pp.147-162, July 1983.

25. G. Williams, "A Closer Look at the IBM Personal Computer", Byte, Vol.7, No.1, pp.51-56, Jan. 1982.

26. Pro-Log Corp., "Pro-Log User's Software Library", April, 1981.

27. Advanced Micro Devices, Inc., "AmZ8000 User's Manual", Vol.7, pp.(2-4)-(2-5), 1982.

28. Intel Corp., "The 8086 Family User's Manual", p.2-3, October 1979.

29. Motorola Inc., MC68000 16-Bit Microprocessor User's Manual", p.1-10, 1980.

30. Motorola Inc., "MC68000 vs. IAPX86 Benchmark Performance", Motorola #BR150, May 1986.

31. Motorola Inc., "Motorola MC68020 Becnhmark Report", Motorola #BR322, May 1986.

32. Zilog Inc., "Benchmark Report Z8000 vs 8086 vs 68000", Microprocessor Applications Reference Book, Vol.2, pp.(4-9)-(4-19), 1983.

33. B. Kris, "A Comparison of the 11/23, Z8000 and 68000 processors", Internal Bridgeport-Textron Report, Feb. 1980.

34. Texas Instruments Inc., "Domestic Suggested Resale Pricing Guide", Third Quarter 1988.

35. RCA Solid State Products Inc., "Industrial Market Price Schedule", 1986.

36. Fairchild Inc., "Suggested Resale Pricing", May 1987.

37. National Semiconductor Inc., "OEM Resale Schedule", 1985.

VITA

Bryan Kris was born in New York City on July 25, 1954 to Mr. Robert E. Kris and Mrs. Helen Socol. He attended Lehigh University and received a B.S.E.E. in 1976. From 1976 to 1979, he was employed by Burroughs Corporation as a design engineer with the Large Systems Group. Responsibilities included the hardware design of the arithmetic processing unit in a large mainframe computer and the design of a custom integrated circuit used in the arithmetic and addressing units. The integrated circuit performed radix 3 operations used for error checking and address calculations.

Bryan joined Bridgeport-Textron in 1979 as a design engineer developing microprocessor based industrial control systems. He became the Manager of CNC Development and produced the R2E3 and the R2E4 control systems. He developed a novel Time Multiplexed Processor Bus for creating high performance multiple processor systems and received the United States Patent #4,630,193.

Bryan joined North American Drager, a manufacturer of anesthesia machines and medical electronics, in 1986 as Manager of Electrical Engineering. Responsibilities include the design and development of electronic patient monitoring equipment used in the hospital operating room environment.