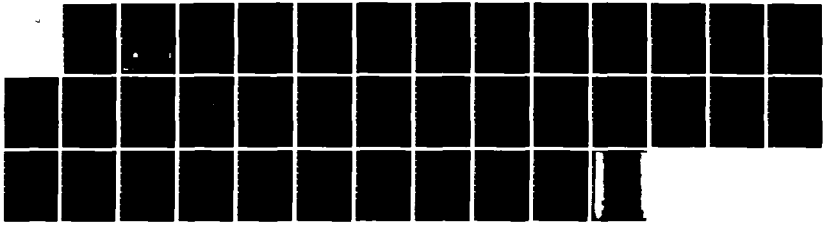
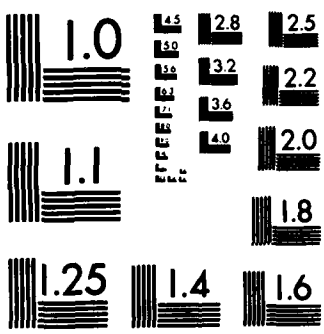
AD-A142 707 A SUMMARY OF FAULT TOLERANT COMPUTING RESEARCH AT CARNEGIE-MELLON UNIVERS..(U) CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE D P SIEWIOREK

1/1

UNCLASSIFIED MAY 84 CMU-CS-84-123 N00014-77-C-0103 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

CMU-CS-84-123

(2)

A Summary of Fault Tolerant Computing Research

at

Carnegie-Mellon University

Daniel P. Siewiorek

Departments of Computer Science

and

Electrical and Computer Engineering

DEPARTMENT of COMPUTER SCIENCE

Carnegie-Mellon University

CMU-CS-84-123

A Summary of Fault Tolerant Computing Research at Carnegie-Mellon University

Daniel P. Siewiorek
Departments of Computer Science
and
Electrical and Computer Engineering

This work has had multiple sources of funding including: Digital Equipment Corporation, Ballistic Missile Defense Advanced Technology Center, Contract No. DASG-60-80-C-0057, and the Office of Naval Research, Contract No. N00014-77-C-0103.

Table of Contents

List of Figures

List of Tables

ABSTRACT

Spurred by increased system complexity, critical application areas, and increased personnel costs, fault tolerant computing systems are rapidly becoming desirable if not necessary. This report summarizes the accomplishments of the past five years in: data collection on hard and transient faults from a variety of systems including Cm* (a 50 processor multiprocessor), PDP-10 timesharing systems, VAX-11 timesharing systems, C.mmp (a 16 processor multiprocessor), and C.vmp (a triplicated multiprocessor); modeling; experimentation; tools and techniques; and architecture.

Fault Tolerant Computing at Carnegie-Mellon University

Over the past five years substantial progress has been made in the design of reliable computer systems at Carnegie-Mellon University. This report is a summary of research accomplishments over those last five years.

1. Definitions

Prior to presenting results, some common terminology has to be established. By *failure* we mean a physical change to hardware. A *fault* is the deviation of a logic function from its design value. A fault may be either *hard* or *transient*. Hard faults can be either *permanent* or *intermittent* and may be repaired by the replacement of a physical component. Transients, on the other hand, may be due to environmental factors (static electricity, noisey power supplies, cosmic rays, etc.) or design errors. Transients cause no permanent damage to hardware. They may not be repaired by physically replacing a component with an identical component. Finally, an *error* is a manifestation of a fault by an incorrect value.

A function that we will frequently refer to is the Weibull. The Weibull reliability is expressed as:

$$R(t) = e^{-(\lambda t)^{\alpha}} \tag{1}$$

where λ is the scale parameter and α is the shape parameter. Note that for $\alpha = 1$ equation (1) reduces to the familiar exponential:

$$R(t) = e^{-(\lambda t)} \tag{2}$$

When $\alpha < 1$ the Weibull function exhibits a failure rate (also called hazard function) that decreases with time.

2. Typical Cycle

There are five stages in the design of a reliable system: data collection, modeling, experiments, tools/techniques, and architecture. First, data must be collected on system failures. This data includes the manifestations of failures as well as statistical values such as the mean or average time to failure. Once the data has been collected and analyzed, a mathematical model that describes the various fault phenomena can be developed. Experiments can be designed and conducted that prove the predictability of the model as well as further verification of the model. Next, software tools are developed to assist the modeling of complex systems. A set of redundancy techniques[1] is developed, each of which circumvents a class of observed faults. The tools must be capable of analyzing these techniques, both in isolation and in combination with other techniques. Finally, an architecture can be designed using the tools and techniques to make design decisions.

---

[1] By redundancy we mean extra time and/or hardware beyond the theoretical minimum.

The architecture will tolerate the fault and error manifestations assembled during the data collection phase.

Our research has followed this typical cycle twice: once relating to hard failures and the second time relating to transient faults. The cycle has almost been completed with respect to hard failures and we will discuss this work first.

3. Hard Failures

3.1. Data Collection

Data on hard failures was collected from Cm*, a 50 processor, LSI-11 based, experimental system built at Carnegie-Mellon University. Table 3-1, taken from [Siewiorek et al., 1978a] depicts this data. Several different module types were utilized in Cm*. The chip count for each module is tabulated followed by the number of modules of this type in the system, followed by the total number of hours these modules were utilized, followed by the total number of failures, followed by the Mean Time To Failure if the data followed an exponential distribution.[2] The first question to answer was whether the data fit an exponential or a Weibull function. Statistical tests applied to the data indicated with a high confidence level that the data represented a Weibull with shape parameter $\alpha = 1$ (that is, the exponential).

The second question to answer was, "What model can be used for the scale parameter (λ, usually referred to as the failure rate)?" At the time of our study the typical method for calculating λ was to use the Military Handbook 217B [U.S. 1976]. The model is as follows:

$$\lambda = \pi_q \pi_l (c_1 \pi_e + c_2 \pi_t) \tag{3}$$

where π_l is the learning factor with values 1 or 10; π_q is the quality factor with values ranging from 1 to 150; π_e is an environmental factor with values ranging from 0.2 to 10; π_t, the temperature factor with typical values ranging from 0.1 to 100; and c_1 and c_2, complexity factors that are functions of the number of bits or gates in the chip. At the time, there was considerable concern over the MIL 217B recommendation of π_q for commercial components as well as the prediction for λ for large scale integrated and MOS circuits. According to MIL 217B, $\pi_q = 150$ for commercial grade components. Experience indicated that $\pi_q = 16$ produced a more accurate fit to observe data. $\pi_q = 16$ corresponded to screening class C. While commercial components undergo screening, the screening tests are not as exhaustive as for full class C rating. However, the environment that commercial components experience usually is not as harsh as the full military

---

[2] Although several other sources of hard failure data were consulted, none of the sources contained the detailed information required to do a statistical analysis. This information would include the number of power-on hours, duty cycle, ambient temperature, etc. In addition, the data was usually from high temperature accelerated life testing, which requires a translation (using an exponential function) from hours at high temperature to hours at ambient temperature. The sources consulted included component manufacturers data (such as Signetics and Intel) and the Reliability Analysis Center (RAC) at Rome Air Development Command (RADC), Griffiss Air Force Base, Rome, NY. The data collected for Cm* was documented and submitted to RAC, RADC.

Module	Complexity (Chips)	# of Modules	Total Time (Hours)	Total Failures	MTBF (Hours)
Kbus	138	3	36696	8	4587
Pmap	106	3	37416	12	3118
Mmicro	116	6	68328	4	17082
Mdata	142	3	37080	2	18540
Linc	116	3	22608	0	-
LSI-11	68	14	163200	10	16320
Slocal	126	10	120720	5	24144
4K memory	56	21	260568	5	52003.6
16K memory	104	10	122280	5	24456
Slu	28	17	223248	5	44649.6
Power board	6	16	195456	3	65152
Refresh	14	16	162912	0	-

Table 3-1: Data Collection on Hard Failures

environment. Since almost a factor of 10 in failure rate is the difference between class C and commercial class, it was important to establish the actual value for π_q.

For LSI MOS devices, the available data indicated that the MIL 217B model was a factor of 16 to 64 pessimistic. Since the MIL 217B model was published in 1974 and was probably developed on 1972 data, one might speculate that LSI technology had not stabilized in time for creation of the model. One could speculate further that the complexity factor could be modified with time since as the process matures more complex components are feasible. If we use the rule of thumb that memory doubles in complexity every one to one-and-a-half years and we want the state of the art portion of the curve in 1977 (when we did our study) to correspond to its position in 1972, then the complexity axis (number of bits) should be divided by $2^4 = 16$. Table 3-2 summarizes the observed data and the model that statistical tests determined as the best fit. Only four possible models (out of tens of models that might have been suggested) were considered. These included $\pi_q = 16$, 150 and derating factors of 1, 16.[3] Table 3-2 indicates that $\pi_q = 16$ and a derating factor of 16 was generally the best fit to our observed data. Our calibration of hard failure models continues with the recently announced MIL 217D model [U.S. 1982].

[3] A complexity derating factor of 16 means the number of gates or bits on a chip was divided by 16 prior to using the MIL 217B model. For example, a 16K RAM was treated as 1K RAM for modeling purposes.

3.2. Modeling

The modeling of non-redundant computer structures has centered around the MIL 217 model described above. Modeling of redundant structures has centered around multiprocessors such as Pluribus, C.mmp, Cm*, etc. [Siewiorek 1978b].

3.3. Tools/Techniques

It is easy to envision a reliability design cycle. One starts with an architecture and a set of reliability techniques. The current state of the architecture is modeled and evaluated (See Figure 3-1). Weak points in the design are identified and suitable reliability techniques are selected and applied to the architecture. The design evaluation cycle is repeated until the overall design goals are met.

As an example, consider a 32 bit minicomputer such as the VAX-11. It is not unusual for contemporary computer designs to be dominated by ROM and RAM chips. Indeed, frequently 50% or more of the failure rate can be attributed to memory chips. A program called LAMBDA [Elkind 1983] helps to automate the evaluation process. Figure 3-2 depicts the LAMBDA input for a VAX class machine.

Module	Observed Failure Rate (per 10^6 hours)	Best Fit Modification To MIL-217B	Predicted Failure Rate With Modified MIL-217B
Kbus	218	Q = 150/16	413
Pmap	320.7	Q = 150/16	333.7
Mmicro	58.5	Q = 16/16	26.6
Mdata	53.9	Q = 16/16	35.4
Linc	-		
LSI-11	61.3	Q = 16/16	29.9
Slocal	41.4	Q = 16	1.8
4K memory	19.1	Q = 16/16	23.1
16K memory	40.9	Q = 16/16	74.1
Slu	22.4	Q = 150/16	43.9
Power Board	15.3	Q = 150/16	9.1
Refresh			

Table 3-2: Comparison of Modified MIL-HDBK-217B Model with Experimental Data

The output of LAMBDA is depicted in Figure 3-3. As we can see, over 51% of the failure rate is due to ROMs and RAMs. From the relative failure rate distribution we can pick an area to apply a redundancy technique. For example, the control store accounts for the largest percentage of failure (almost 30%) and of this more than 94% is attributed to the ROM and RAM. Figure 3-4 depicts the impact on failure rate when using error correcting code on the control store. Now the ROM and RAM have been reduced from 51% to

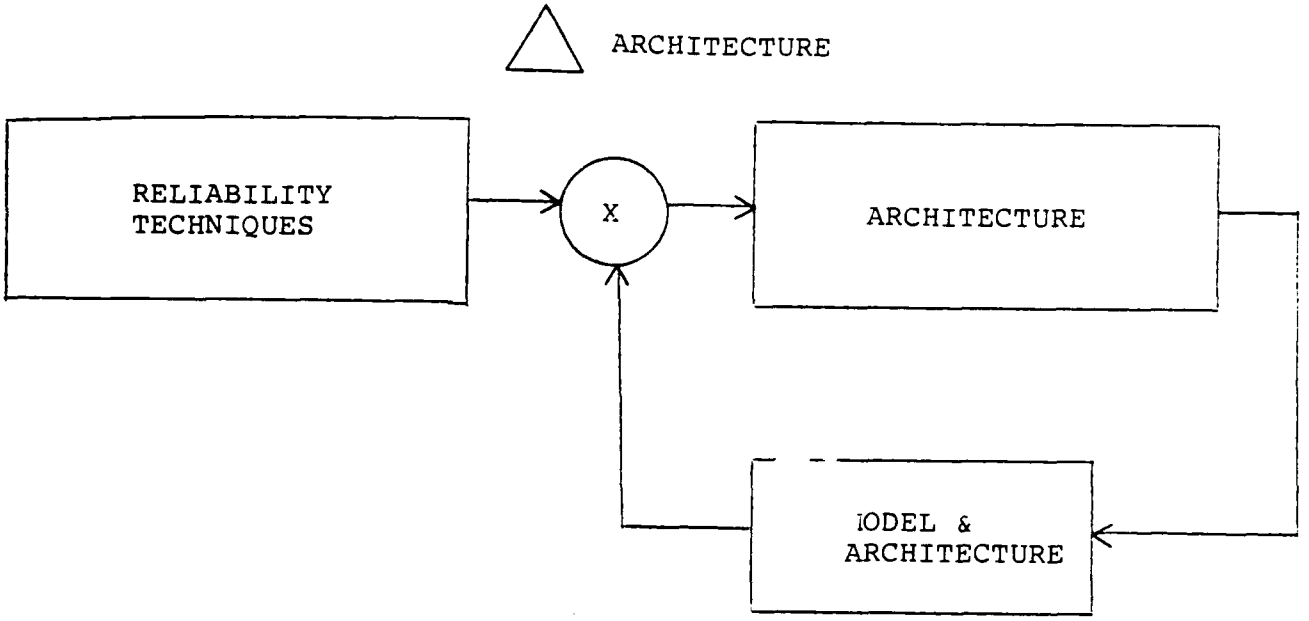


Figure 3-1: Reliability Design Cycle

32% of the failure rate and the control store from almost 30% of the failure rate to 3%. A companion program, called SEC [Elkind and Siewiorek 1980], calculates the Mean Time to Failure of a fault tolerant memory given the memory organization and failure rate per bit. By simply applying error correcting code to various portions of memory it is possible to more than double Mean Time To Failure in this example.

Indeed, it is possible to develop a reliability space such as depicted in Figure 3-5. The vertical axis plots Mean Time To Failure while the horizontal axis depicts the number of chips in a design. The space is that of a PDP-8 using replication and coding techniques. Eight possible designs were illustrated covering a factor of 4 in Mean Time To Failure and over a factor of 2 in cost. Several interesting features can be noted. The first is the phenomena of diminishing returns. Path B1 to B2 illustrates that for approximately a 30% increase in cost less than a 5% increase in Mean Time To Failure results. As a matter of fact, the point of diminishing returns can be carried so far that the inclusion of extra redundancy actually decreases system reliability (as depicted on path B1 to A2).

Models can be developed for individual techniques upon individual portions of the system such that a designer need only know values of some physical constants in order to decide whether the redundancy technique is worthwhile. For example, consider the nomagraph of Figure 3-6. The figure depicts the design space for error correcting code memory. The vertical axis is the log of normalized Mean Time To Failure, the

```
[VAX
  [Data.Path.Module
    [Misc
     4,7400; 1,7403; 3,7404; 1,7405; 3,7410; 1,7420; 1,7422; 1,74112;
     2,74153; 1,74157; 1,74194; 8,74151; 8,74280; 1,74138; 1,7408;
     2,7437; 1,74240; 1,74241; 8,74374; 2,74165; 1,74260; 4,74374]
    [ROM.AND.RAM
     4,3101; 32,3101; 1,3601; 1,MK2048; 2,MR4096; 7,MK8192]
    [GATE.ARRAY
     22,GA400]]
  [Memory.Interface.Cache
    [Misc
     1,7400; 2,7404; 1,7410; 1,7420; 3,7464; 3,74153; 2,74158;
     17,74280; 4,74257; 1,7451; 1,7486; 1,74182; 1,7402; 1,7408;
     1,7437; 5,74241; 5,74374; 2,74165; 2,74260; 1,7430; 1,74244;
     1,7411; 2,74175; 1,7432]
    [ROM.AND.RAM
     20,93415; 50,93425]
    [GATE.ARRAY
     24.GA400]]
[Unibus.Interface
    [Misc
     2.DM8881; 2,7400; 2,74123; 3,7400; 1,7403; 3,7404; 1,7405;
     2,7410; 3,7420; 4,7474; 1,74174; 4,74175; 2,DM8640; 2,74280;
     6,DM8641; 1,74138; 2,7451; 1,7402; 4,74253; 2,7400; 1,7420;
     1,7422; 1,7430; 1,7451; 1,7483; 1,74273; 1,7432; 1,74241;
     7,74374; 1,74165; 1,74260; 3,74393; 1,74132; 6,74244; 1,UART;
     3,MC14040]
    [ROM.AND.RAM
     17,93415]
    [GATE.ARRAY
     13.GA400]]
  [Control.Store
    [Misc
     1,74374; 1,7474; 1,74280; 1,74138; 1,7432; 1,7443; 1,74240;
     9,7437; 4,7444]
    [ROM.AND.RAM
     120,MR4096]]
  [Memory.Controller
    [Misc
     13,7400; 1,7403; 7,7404; 6,7410; 1,7411; 6,7420; 2,7464;
     1,7465; 7,7474; 2,7474; 7,74112; 1,74140; 2,74153; 3,74157;
     6,74158; 1,74174; 1,DM8640; 1,74138; 1,74139; 1,7451; 1,74133;
     1,7486; 6,7402; 1,7408; 2,7437; 1,7404; 1,74375; 4,74373;
     11,74374; 2,74260; 5,7430; 3,74393]
    [ROM.AND.RAM
     4,MR16384; 15,3601]
    [GATE.ARRAY
     7,GA400]]]
```

Figure 3-2: Example LAMBDA Input for a VAX-Class Minicomputer

horizontal axis is the log of the ratio of the failure rate of the support electronics to the failure rate per bit of the memory. The support electronics is the error encoding, decoding, memory controller, and bus interface

26 January 1982

REL. LSI= 16.000 ROM= 16.000 RAM= 16.000

E = 1.000 Q = 16.000 L = 1.000 T = 40.000

MODULE	PERCENTAGE
VAX	100.000
DATA.PATH.MODULE	16.187
MISC	42.643
ROM.AND.RAM	27.007
GATE.ARRAY	30.271
MEMORY.INTERFACE.CACHE	25.038
MISC	30.003
ROM.AND.RAM	48.648
GATE.ARRAY	21.349
UNIBUS.INTERFACE	14.158
MISC	58.633
ROM.AND.RAM	20.905
GATE.ARRAY	20.462
CONTROL.STORE	29.908
MISC	5.657
ROM.AND.RAM	94.343
MEMORY.CONTROLLER	14.717
MISC	64.169
ROM.AND.RAM	25.237
GATE.ARRAY	10.594

# of chips = 662.000 # of gates = 33361.000 # of bits = 732416.000

SUMMARY ROLLUP BY COMPONENT TYPE

TYPE	# of CHIPS	PERCENTAGE
SSI	180.000	10.824
MSI	142.000	22.809
LSI	67.000	14.914
ROM	150.000	35.179
RAM	123.000	16.274
MOS	.000	.000
BIP	662.000	100.000

Figure 3-3: LAMBDA Output for a VAX Class Computer

logic. Two memory sizes are depicted: 16 Kwords and 64 Kwords. There is a non-redundant memory model, whole chip failure mode model, and single bit failure mode model. [Elkind and Siewiorek 1980]. The 45° line depicts the place where the failure rate of the support electronics dominates the failure rate of the memory cells. For contemporary logic the log of λ_s/λ_b is approximately 5. Thus, we see that under the single bit failure mode assumption there is no advantage to increasing redundancy on the memory array. The support electronics already dominates the failure rate of the memory. Even under worse case assumptions

26 January 1982

LSI= 16.000 ROM= 16.000 RAM= 16.000

E = 1.000 Q = 16.000 L = 1.000 T = 40.000

MODULE	PERCENTAGE
VAX	100.000
DATA.PATH.MODULE	22.342
MISC	42.643
ROM.AND.RAM	27.087
GATE.ARRAY	30.271
MEMORY.INTERFACE.CACHE	34.558
MISC	30.003
ROM.AND.RAM	48.648
GATE.ARRAY	21.349
UNIBUS.INTERFACE	19.531
MISC	58.633
ROM.AND.RAM	20.905
GATE.ARRAY	20.462
CONTROL.STORE	3.257
MISC	71.689
GATE.ARRAY	28.311
MEMORY.CONTROLLER	20.312
MISC	64.169
ROM.AND.RAM	25.237
GATE.ARRAY	10.594

# of chips = 545.000 # of gates = 34561.000 # of bits = 240896.000

SUMMARY ROLLUP BY COMPONENT TYPE

TYPE	# of CHIPS	PERCENTAGE
SSI	180.000	14.940
MSI	142.000	31.481
LSI	70.000	21.507
ROM	30.000	9.611
RAM	123.000	22.461
MOS	.000	.000
BIP	545.000	100.000

Figure 3-4: LAMBDA Output Assuming Error Correcting Code Applied to the Control Store

that a failure knocks out a complete memory chip (the whole chip failure model) the support electronics still dominates the failure rate, and, indeed, if the memory chips are reliable enough (corresponding to log λ_s/λ_b of 6 or more), a non-redundant memory would be preferable.

It is our contention that enough fault tolerant techniques have been discovered that there exists a spectrum from which the designer can choose. For example, we have developed a taxonomy that divides techniques

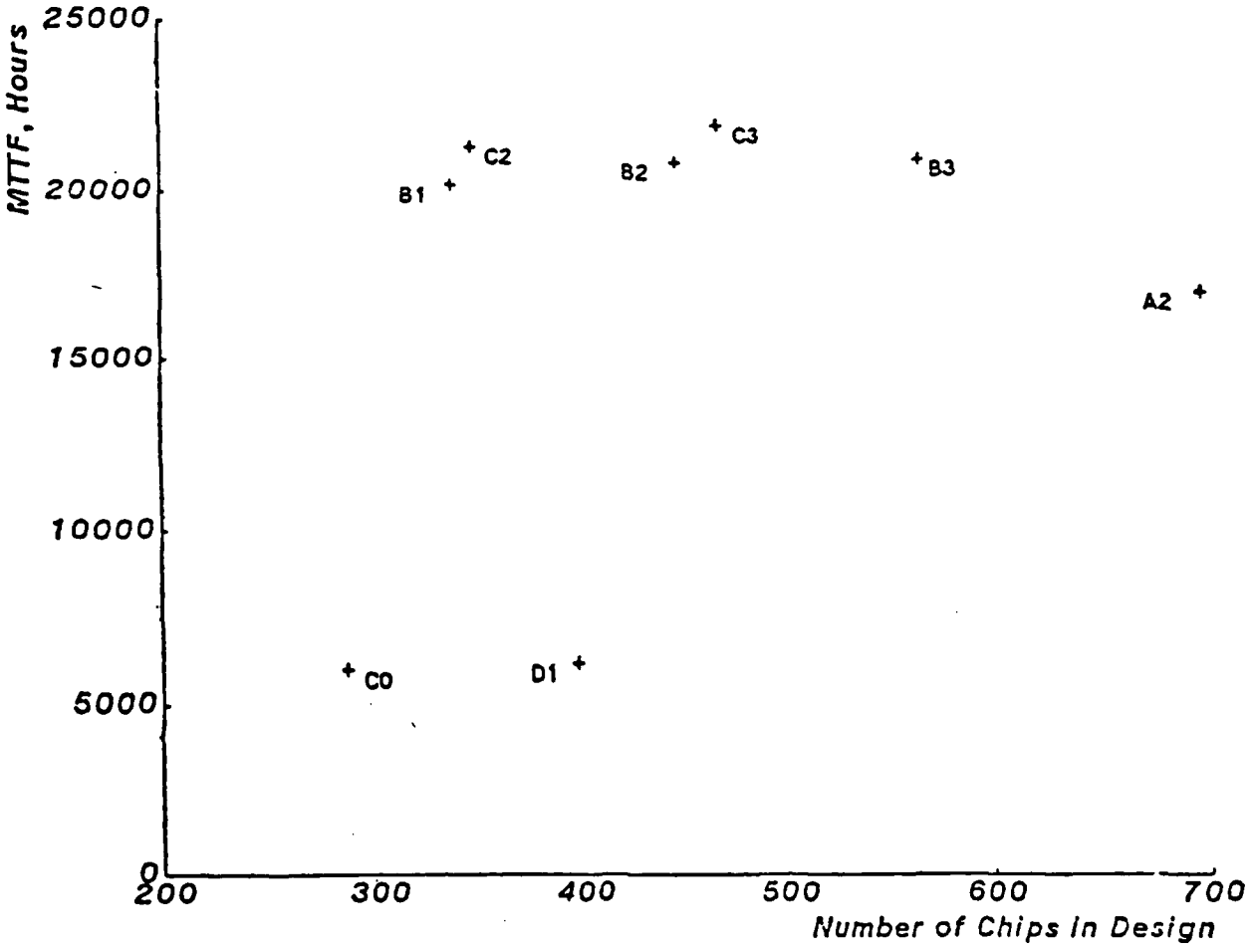


Figure 3-5: The Reliability Design Space for a PDP-8
Depicting MTTF as a Function of Chip Count

into on-line/off-line detection/correction. The designer determines whether off-line detection/off-line correction, on-line detection/off-line correction, or on-line detection/on-line correction is desired and what can be paid in cost and performance. The designer then enters the catalogue according to detection/correction cost. The designer then picks out techniques in that region which apply to the design at hand (e.g., memory, random, logic, etc.). We have compiled a large catalogue [Elkind 1982]. The catalogue is organized according to the taxonomy and provides reliability formulas and cost evaluations for each technique.

A companion program termed ADVISER [Kini 1981] [Kini and Siewiorek 1982] has been written. ADVISER takes as input a description of a computer system at the PMS (processor, memory, switch) level

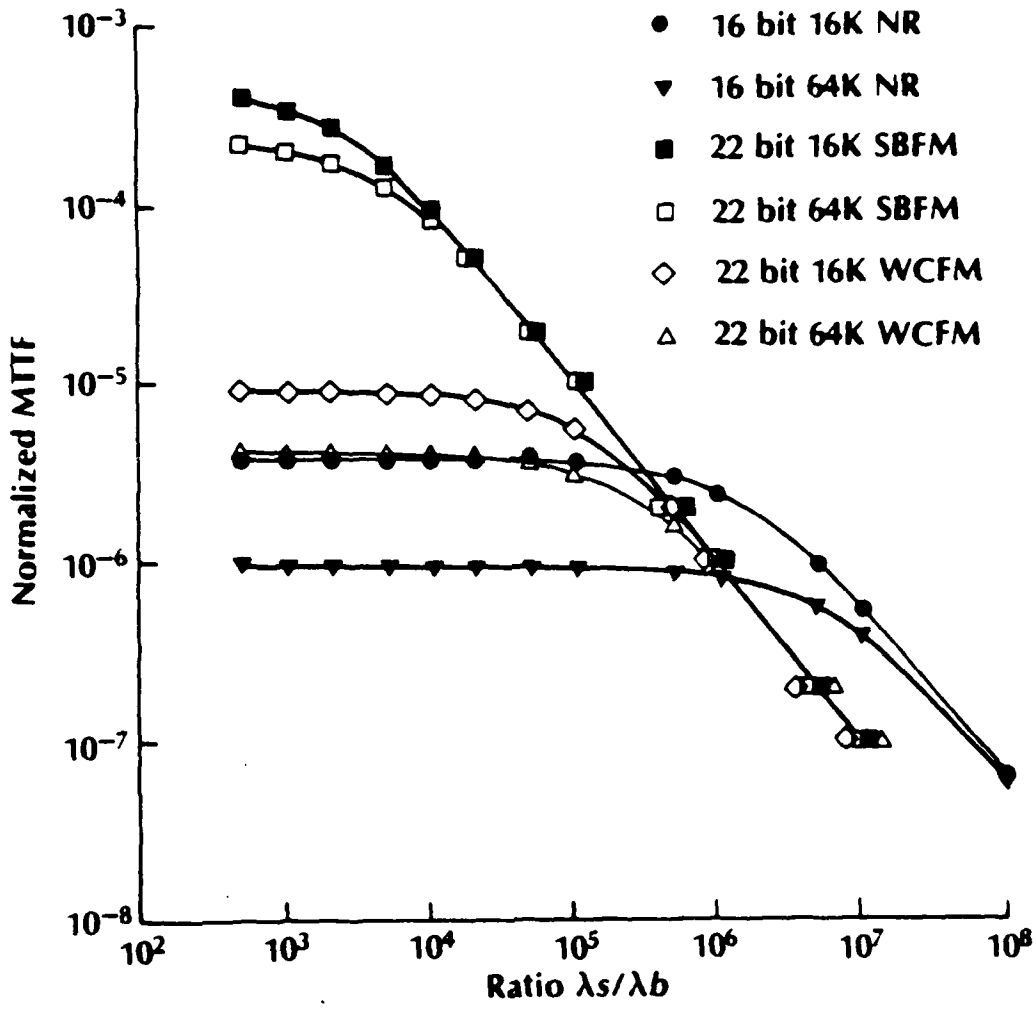


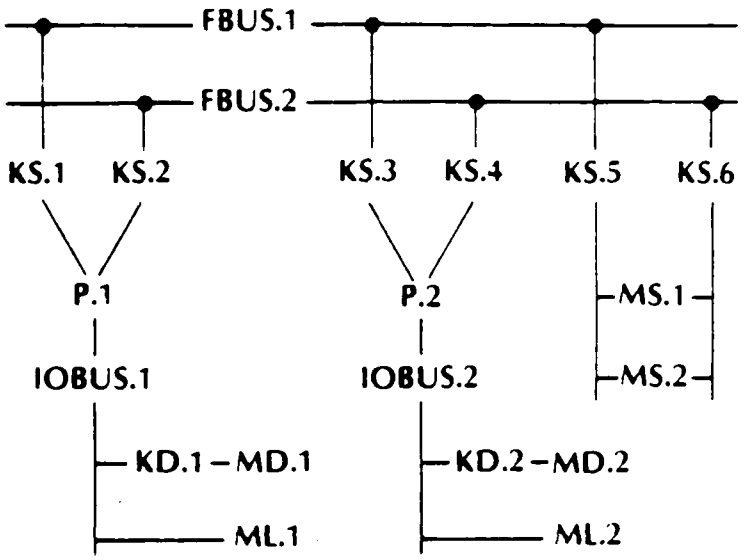
Figure 3-6: MTTF normalized with respect to technology, i.e., as a function of (λ_s/λ_b), for error correcting code memory.

and user constraints and produces a symbolic reliability equation for the system as output. Figure 3-7 depicts a Tandem-like organization that might be described to ADVISER. The K's are controllers, the Mp's are primary memories, the Ms's are secondary memories (disks), the S's are switches, and the P's are processors. A user requirement might consist of requiring one processor, one primary memory, and one secondary memory to work. ADVISER would then use graph techniques to derive the symbolic reliability equation. ADVISER would also fill in missing information. For example, if P1 and Ms1 were to fulfill the system requirements, ADVISER would realize that S and either K13 or K14 would also have to work.[4] Thus

[4] ADVISER would also deduce that alternate paths existed through K12 or K11 to K21 or K22 through P2, through S, through K23 or K24, finally to Ms1.

ADVISER can be used to design a global structure of a redundant system while LAMBDA can be used to design a structure of individual P's, M's, S's, K's, etc. Indeed, one form of ADVISER's output is an executable program where each symbol is treated as a subroutine call that evaluates the reliability of the PMS component for the value of time that has passed as its argument.

PMS DIAGRAM:



Key: P = processor, FBUS = fast bus,
KS = fast bus interface, MS = shared memory,
IOBUS = processor bus, ML = local memory,
MD = disk memory, KD = disk controller.

REQUIREMENTS EXPRESSION:

1 of P and 1 of ML and 1 of MD and 1 of MS

Figure 3-7: A PMS Structure Described to ADVISER

Another tool that has been developed is a simulator. ISP (Instruction set processor) was initially defined as a language for describing computer instruction sets [Bell and Newell 1971] and has since been expanded to describe arbitrary digital systems from the functional level down to and including the gate level [Barbacci et al. 1977] [Barbacci 1981]. The ISP compiler and companion simulator are in use by over 80 government, university, and industrial organizations. The simulator has the ability to insert faults into memories, registers,

and control logic. These faults can be either permanent, intermittent, or transient. The ISP simulator and fault inserter has been used by Bendix to explore, at the gate level, the fault tolerant properties of the SIFT computer designed by SRI [Wensley et al. 1978].

Another topic of interest is the generation of tests [Siewiorek and Lai 1981]. A program has been written which generates functional. diagnostics automatically from an ISP-like description of a digital system [Lai 1981], [Lai and Siewiorek 1983]. To calibrate the quality of the automatically generated functional diagnostics, a comparison was made between the manufacturer's diagnostics and the automatically generated diagnostics for a PDP-8. Almost 1500 faults were inserted into a PDP-8 description upon which the respective diagnostics were simulated. The results were that the automatically generated diagnostics had a higher detection percentage (98.5% vs. 95.5%) and required a factor of 20 less instruction executions.

3.4. Architecture

As a result of this research two architectures for reliable systems have been built. Figure 3-8 depicts C.vmp (computer, voted multiprocessor) and its connections with the CMU environment. C.vmp employs off the shelf components with little or no modifications to achieve hard and transient faults survivability [Canepa, Clark and Siewiorek 1977]. Experience indicates that C.vmp is about six times more reliable for transient faults than a single LSI-11 system [Siewiorek et al. 1978a]. C.vmp executes an unmodified operating system. In addition to a voting mode, the bus level voter also allows a non-replicated device (such as a console terminal) to broadcast results to all three processors or allows the system to divide up into three independent computers intercommunicating through parallel interfaces, i.e., links. C.vmp can switch between independent mode and voting mode operation, thus allowing the user to dynamically trade performance for reliability.

C.vmp has been in operation for over four years. Performance degradation due to the voter has been both theoretically predicted and experimentally measured [Siewiorek and McConnel 1978]. The voter reduces system level performance by about 15%. The voter design has been generalized to include both asynchronous and synchronous bus protocols [McConnel and Siewiorek 1981].

In the last year a statistics board has been added to the C.vmp design. The statistics board compares the three buses for disagreements and snapshots the contents of all three buses (including a unique time stamp) into a shift register when a disagreement is noticed. We are currently collecting data on the nature and duration of transients on C.vmp. Indeed, C.vmp has become a transient meter whereby we can measure the sources and effects of transient errors much in the same way that a voltmeter can measure sources and magnitudes of voltages [Shombert 1981].

The generalized voter design was implemented in a VLSI chip by Sandia Laboratory [McConnel and

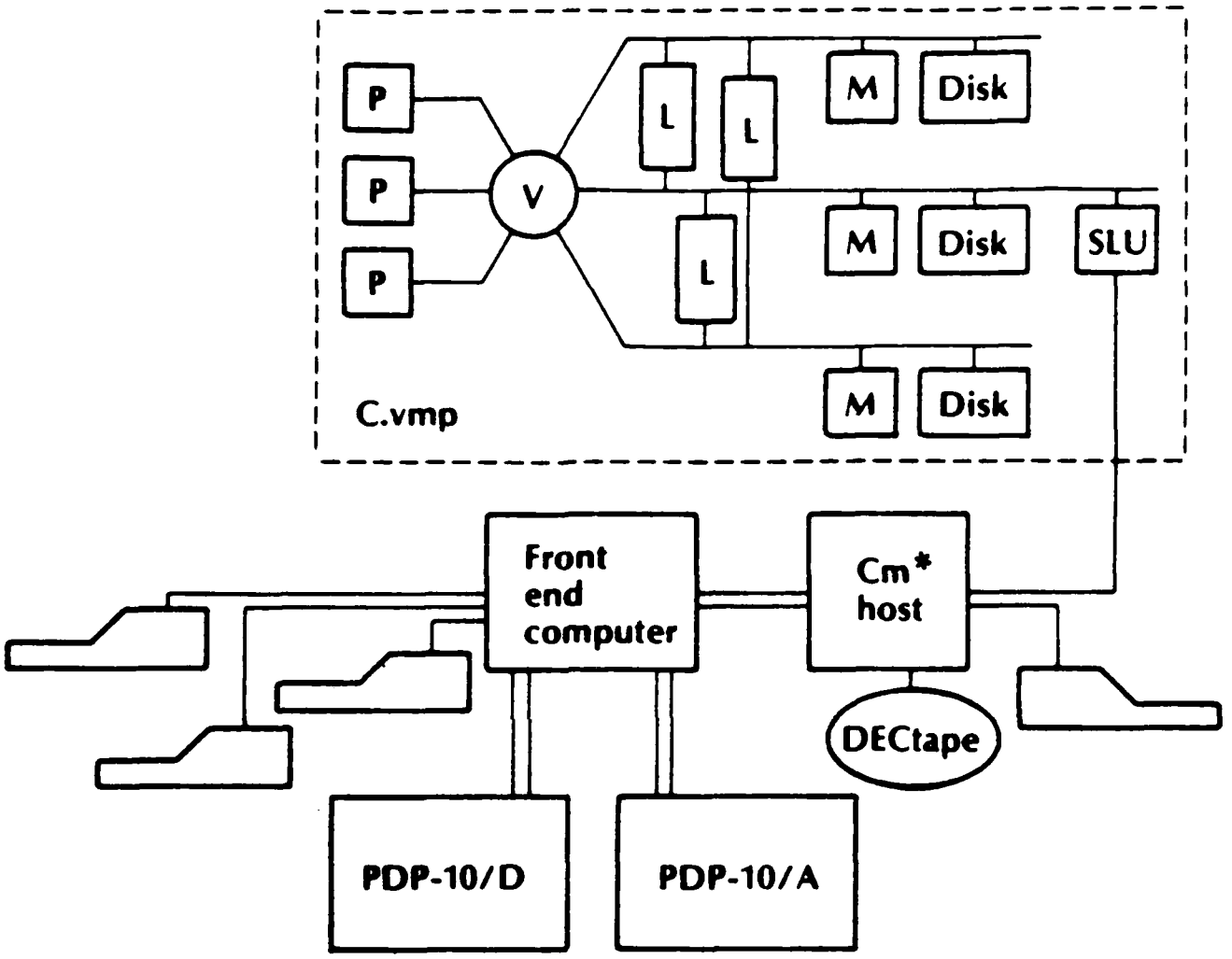
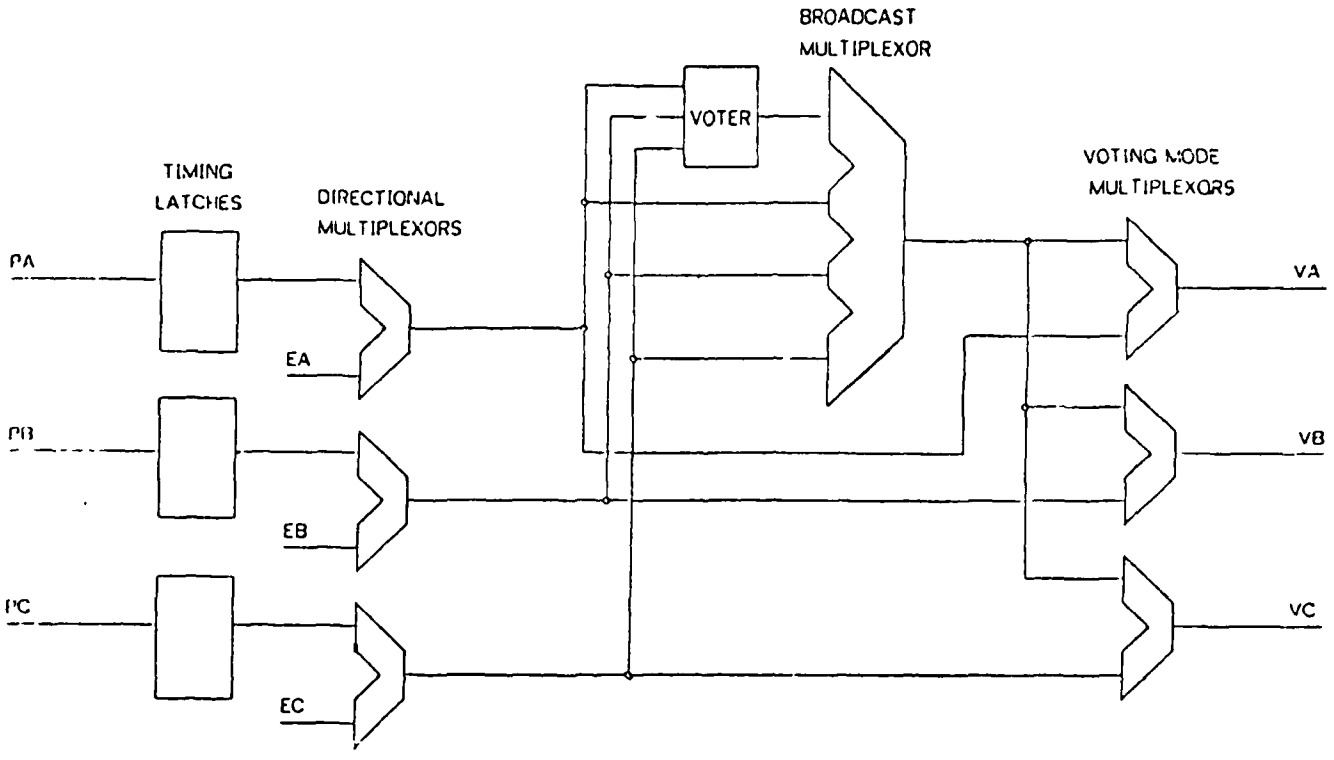


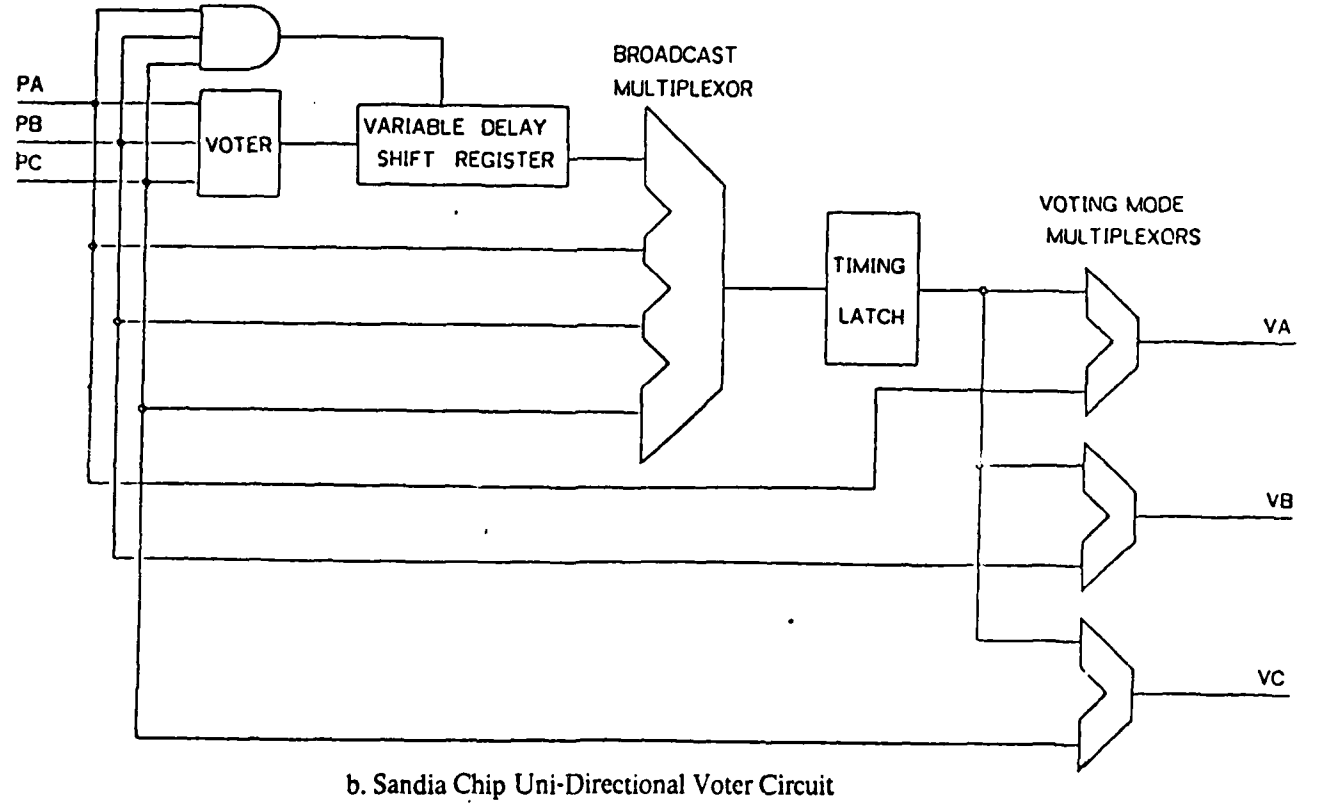
Figure 3-8: C.vmp Configuration and Connection to CMU Facilities

Siewiorek 1980]. Figure 3.9 depicts the data (bi-directional) and control (uni-directional) designs. Each chip has two data and two control voters so that approximately 10 chips are required to replace over 120 chips used in the C.vmp design.

The goal of C.fast (computer, fault tolerant and self testing) was to incorporate fault tolerant and self testing design concepts into a VLSI implementation of a microprocessor. The Fairchild F8 instruction set was selected as the test vehicle. The fault tolerant and self testing techniques were designed to be transparent to the F8 instruction set. Thus, all existing F8 software could be executed on C.fast. C.fast incorporates parity on internal data buses, registers, and PLA's. Three major machine registers have redundant copies so that internally detected errors can be recovered from by attempting macroinstruction retry. Duplication and retry



a. Sandia Chip Bi-Directional Voter Circuit



b. Sandia Chip Uni-Directional Voter Circuit

Figure 3-9: Sandia Chip Voter Circuit

are employed at the chip level to recover from external faults or internal faults that are not detected by other error detection mechanisms. C.fast employs a control bus that is externally controllable and observable. This control bus is the major means for efficiently testing the system [Tsao et al. 1981, 1982].

4. Transient Faults

Substantial progress has also been made in the area of understanding hard failures. However, transient faults are a much harder problem. Once a hard failure has occurred, it is possible to uniquely isolate the faulty component. On the other hand, by the time a transient fault manifests itself (perhaps by a system software crash) all traces of its nature and location are long gone. Hence, the work to date has focused very heavily on data collection and modeling of transient faults.

4.1. Data Collection

We have been collecting data from four timesharing systems, an experimental multiprocessor, and an experimental fault tolerant system. These systems range in size from microprocessors to large ECL mainframes. The method of detecting transient induced errors varied widely. For the PDP-10 timesharing systems internally detected errors are reported in a system error log file. For the experimental multiprocessor, Cm*, a program was written that would automatically load diagnostics into idle processors, initiate the diagnostics, and periodically poll diagnostics as to their state. For the triply redundant C.vmp, a manually generated crash log was kept. Table 4-1 summarizes the Mean Time To Hard Failure and the Mean Time Between Transient Errors for these various systems. Transient faults are seen to be approximately 20 times more prevalent than hard failures [McConnel, Siewiorek, Tsao 1979].

System Technology	Detection Mechanism	MTBE per Processor	MTTF per Processor	MTBE/MTTF
CMUA PDP-10, ECL	Parity	44 hrs.	800-1600 hrs.	0.03-0.06
Cm* LSI-11, NMOS	Diagnostics	128 hrs.	4200 hrs.	0.03
C.vmp TMR LSI-11	Crash	97-328 hrs.	4900 hrs.	0.02-0.07
Telettra, TTL	Mismatch	80-170 hrs.	1300 hrs.	0.06-0.13
1M x 37 RAM, MOS	(Parity)	106 hrs.	1450 hrs.	0.07

Table 4-1: Ratios of Transient to Permanent Errors

The gross attributes of observed transients were recorded [Siewiorek et al. 1978a]. Data from C.mmp (Computer, multi-miniprocessor) illustrated that the manifestation of transient faults is a long way from the

traditional permanent fault models of stuck-at-1 and stuck- at-0.[5] C.mmp exhibited approximately one hardware caused crash per processor every 300 hours. Autodiagnostics from Cm* indicated approximately 250 hours between diagnostic detected errors. In addition, Cm* autodiagnostics indicated that approximately one sixth of the errors occurred in more than one processor at the same time. Thus, if one would assume that these common mode failures would cause a triplicated processor to crash, one would expect C.vmp to crash approximately 6 x 250 = 1500 hours. This compares favorably with the observed 1200 hours Mean Time To Crash in C.vmp.

4.2. Modeling

The next question to ask is, "What mathematical model fits the observed data?" The first step was to analyze the interarrival time of transient errors. Figure 4-1 depicts the probability of system crash as a function of time since the last system crash. Note that the probability of crash (failure rate) decreases with time. Figure 4-2 transforms the data to axes upon which the Weibull function appears as a straight line. The shape parameter α can be estimated by the slope of the line and the scale parameter λ can be estimated from α and the Y-intercept. Table 4-2 depicts data from several systems and the important thing to note is that shape parameter for all the systems is well below 1.0. That is, a decreasing failure rate function (and not an exponential) is the best fit to the data. Figure 4-3 displays the two dimensional 90% confidence intervals (called consonant sets) for the α and λ parameters. None of the consonant sets, denoted by capital letters encompasses the corresponding maximum liklihood estimators, denoted by lower case letters, assuming an exponential function.

Since experimental data supports the decreasing failure rate model, a natural question is, "How far can you stray if you assume an exponential with constant failure rate instead of a decreasing failure rate function?" Figure 4-4 depicts the difference in reliability as a function of time between an exponential and a Weibull function with the shape parameters as indicated. Reliability differences of up to .25 are illustrated. Since the reliability function can only range between 0 and 1, this error is indeed substantial [McConnel 1981] [Castillo, McConnel, Siewiorek 1982].

Another area of modeling involves the relationship between system load and system error rate. Figure 4-5 depicts the number of blocks accessed on a disk as a function of time of day. Figure 4-6 depicts the number of failures on the same disk per day. There is a strong correlation between the two graphs. Software has been developed to analyze system error log entries and to statistically sample system load. [Castillo 1981] and [Castillo and Siewiorek 1982] generate a model of system reliability that predicts failures involving hardware

[5] Examples included incorrect timeout indications, incorrect number of arguments pushed or popped onto a stack, loss of interrupts, and incorrect selection of a register from a register file.

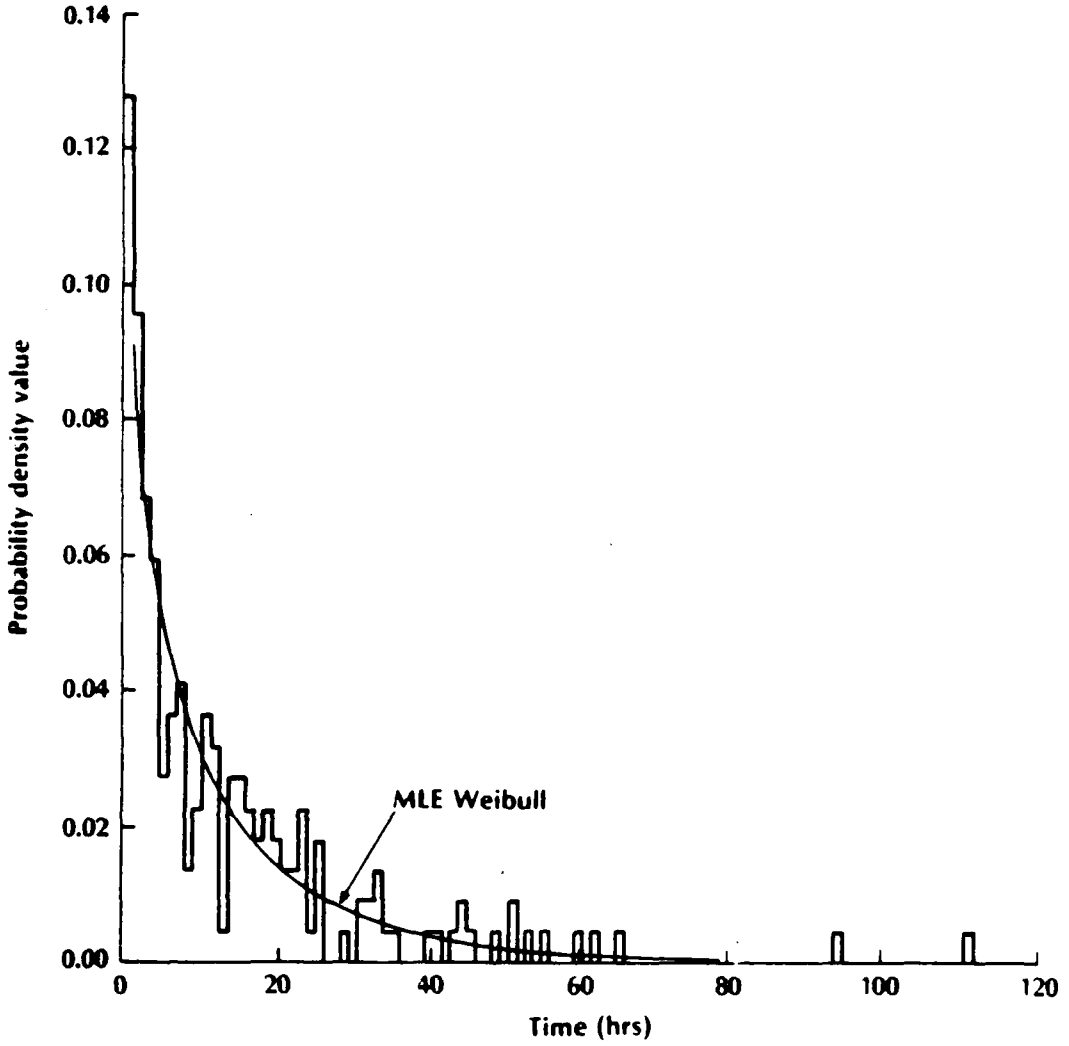


Figure 4-1: Distribution of TOPSC System Reloads

and software errors. Starting with first principles[6] a model called cyclostationary is derived. This model is depicted along with the exponential and Weibull in Table 4-3. The cyclostationary model is an excellent match to the measured data. It also exhibits the property of a decreasing failure rate. Statistical tests demonstrate that for the cost of some modeling accuracy, the Weibull function is a reasonable approximation to the cyclostationary model with the advantage of less mathematical complexity. Figure 4-7 depicts one application of the cyclostationary model. The relative percentage of system crashes due to hardware and software are plotted as a function of time of day. The band represents the interval of uncertainty.

[6] It is assumed that system has two modes of operation, user and kernel. The probability of being in kernel mode is a random event with measurable statistics. A second random event is system fault. It is assumed that the system is much more susceptible to crashing if the fault occurs while in kernel mode than if the fault occurs in user mode. Thus, a doubly stochastic process is set up between the probability of being in kernel mode and the probability of system fault.

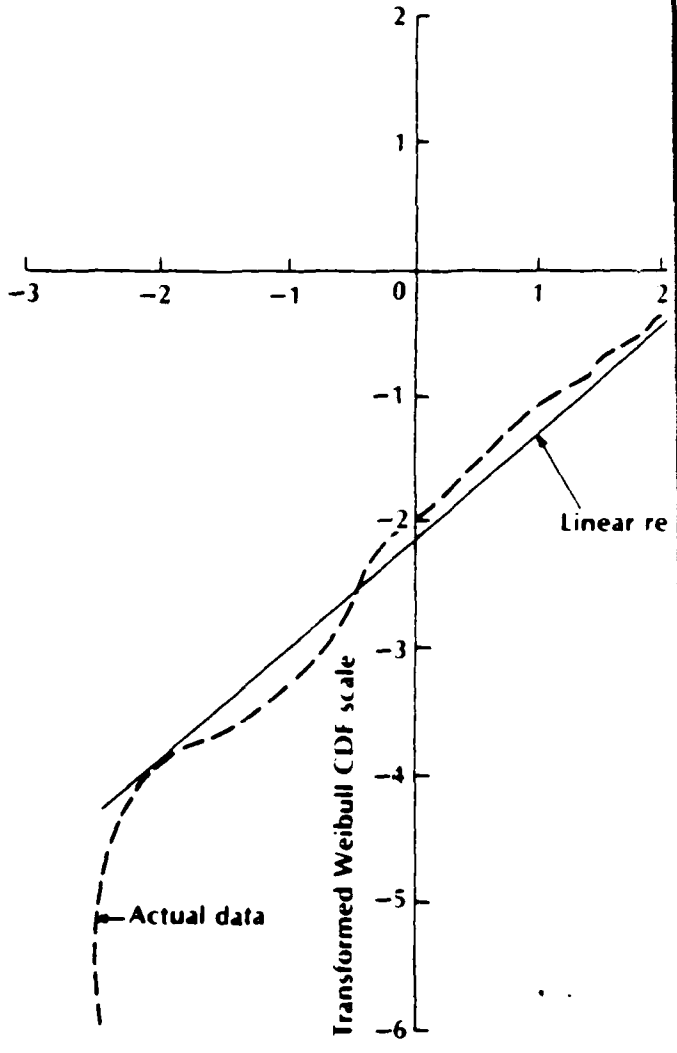


Figure 4-2: Weibull Plot of TOPSC System

	TOPSC Reload	PDP-10 Reload	PDP-10 Parity
Time (hrs)	2646	8576	8596
Errors	195	636	74
Interarrivals	196	640	78
μ	13.5	13.4	110.2
σ	16.5	24.6	244.9
α (Linear)	0.864	0.684	0.500
α' (MLE)	0.826	0.639	.481
λ (Linear)	0.0843	0.109	0.0206
λ' (MLE)	0.0826	0.106	0.0203

Table 4-2: Statistics for Transient Err

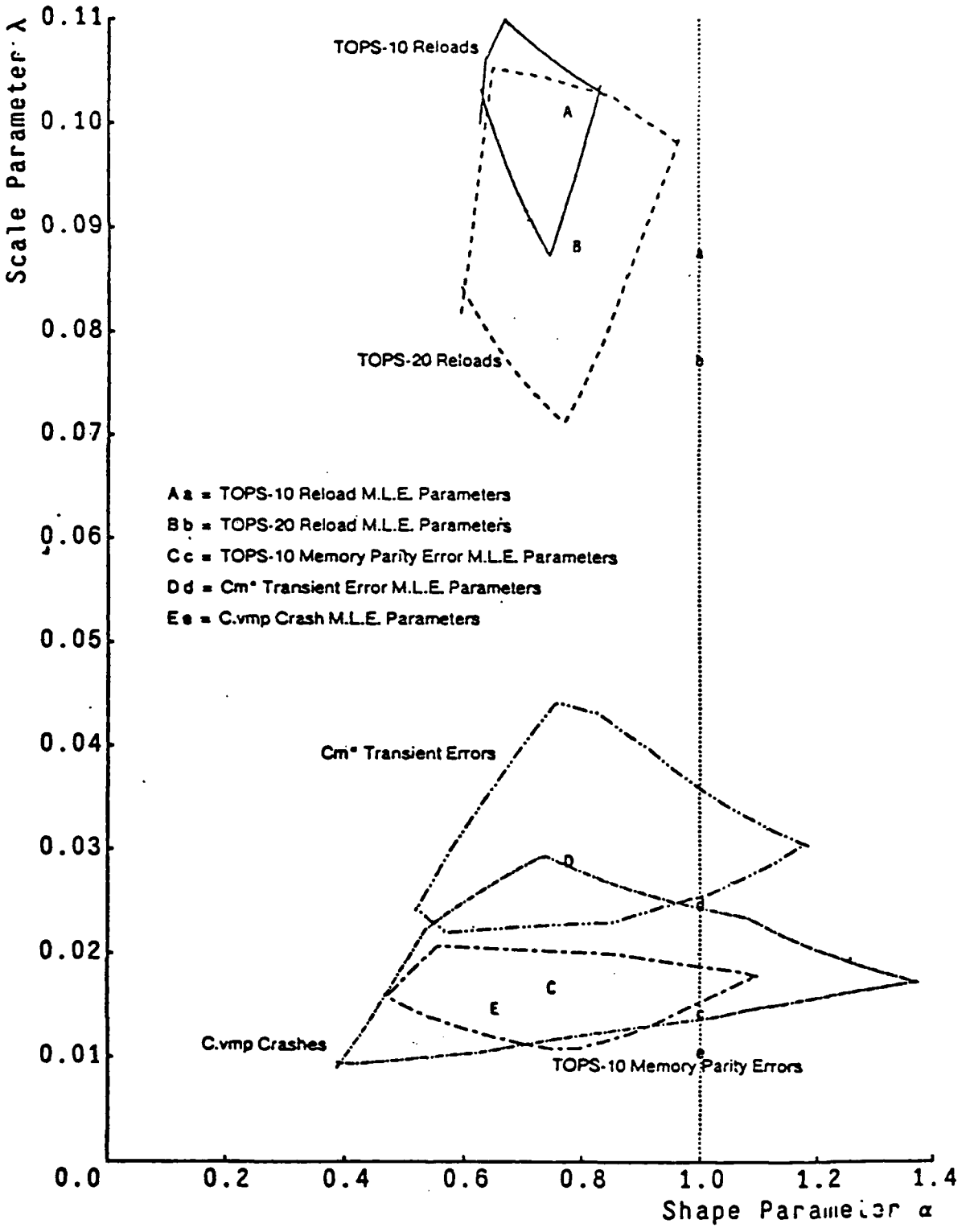


Figure 4-3: 90% Consonance Sets for Weibull Parameters

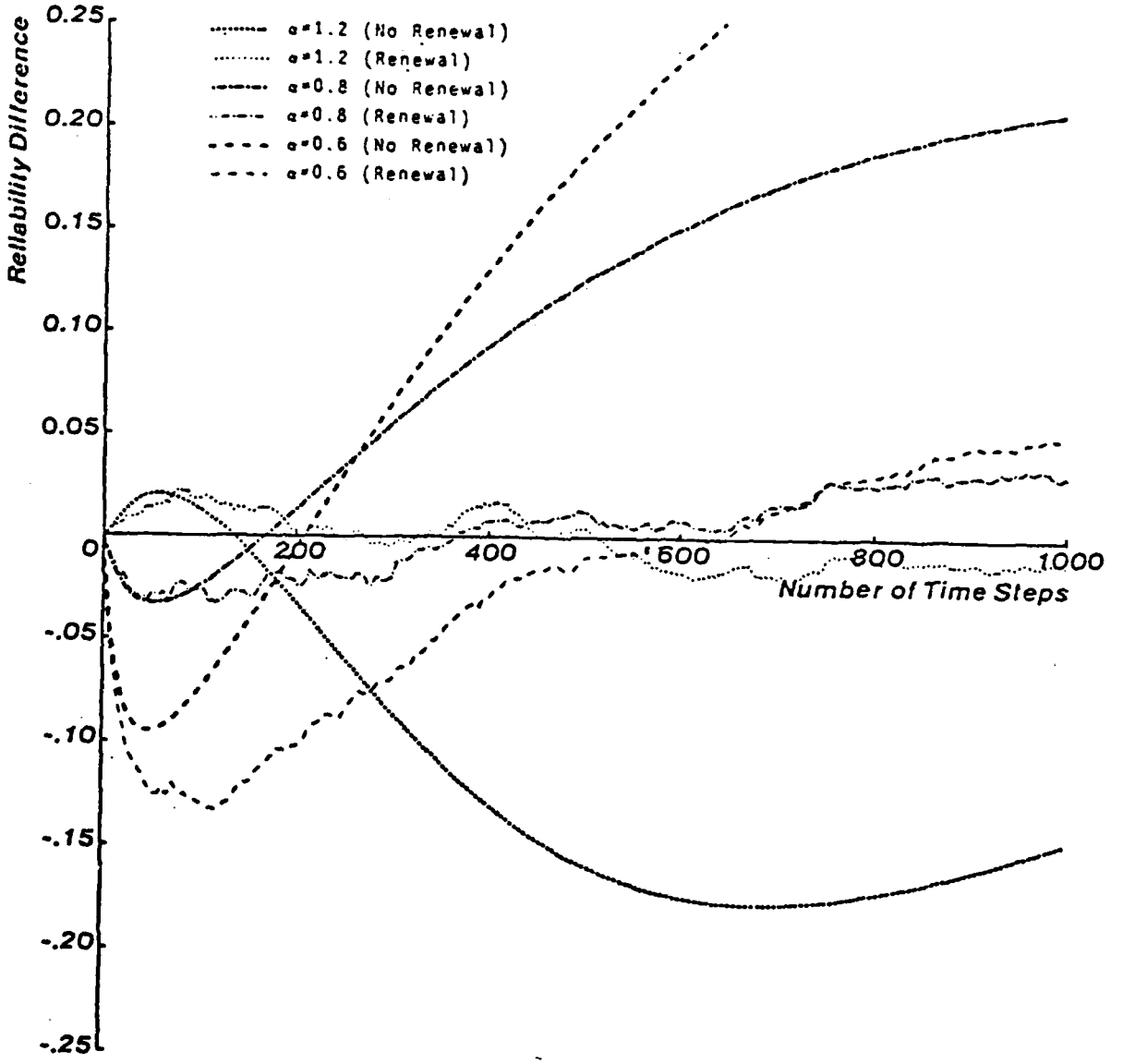


Figure 4-4: Reliability Difference of Duplex System Relative to $\alpha_e = 1.0$

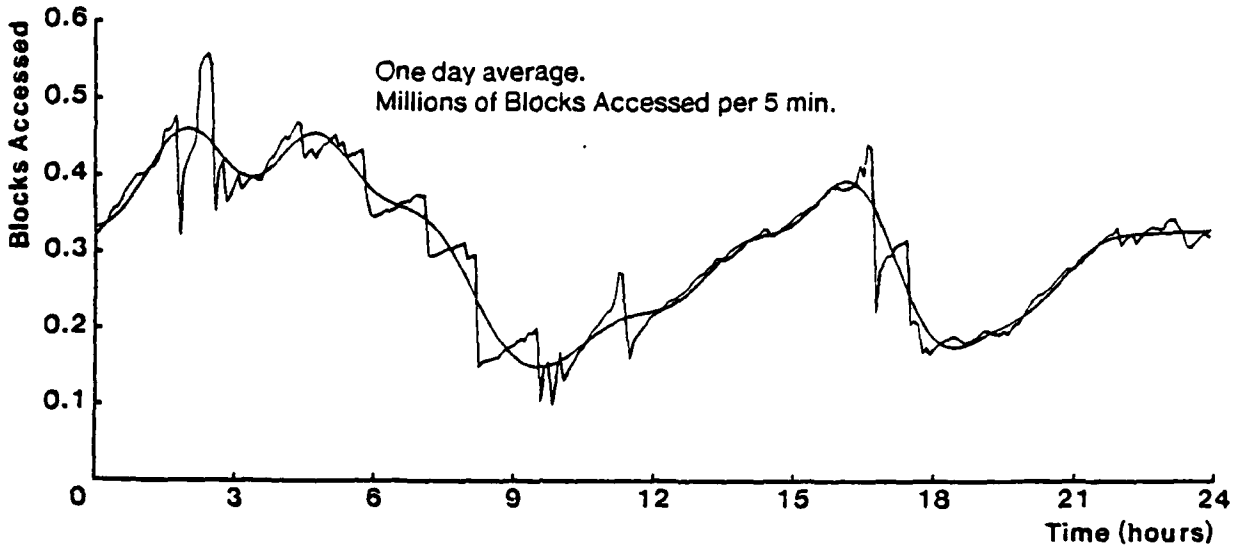


Figure 4-5: Average number of blocks accessed in the file system as a function of time of day.

A natural extension of our work with system error logs is to analyze log entries to determine trends. It may be possible to identify weak or failing components prior to their catastrophic failure [Tsao and Siewiorek 1983]. This information could be provided to an operating system so that reconfiguration could be effected prior to any recoverable errors.

4.3. Experiments

A major limitation to our modeling effort is the fact that contemporary operating systems lack sufficient instrumentation to track and analyze events just prior to a system error. Cm* is a 50 processor system for which two operating systems exist. A synthetic workload generator has been built [Singh 1981] that allows the novice user to convert the data flow graph representing the resource demands of an application into a synthetic workload. A corresponding status monitor has also been built. Hence, the user can ask questions such as, "What percentage of time are processors idle for this application?" That user inquiry creates commands to turn on sensing probes within the operating system. Events relating to the answer of that question (e.g., scheduling and termination of processes on processors, etc.) are reported back to the status monitor. Any system event, performance or reliability related, can be reported and observed. The capability

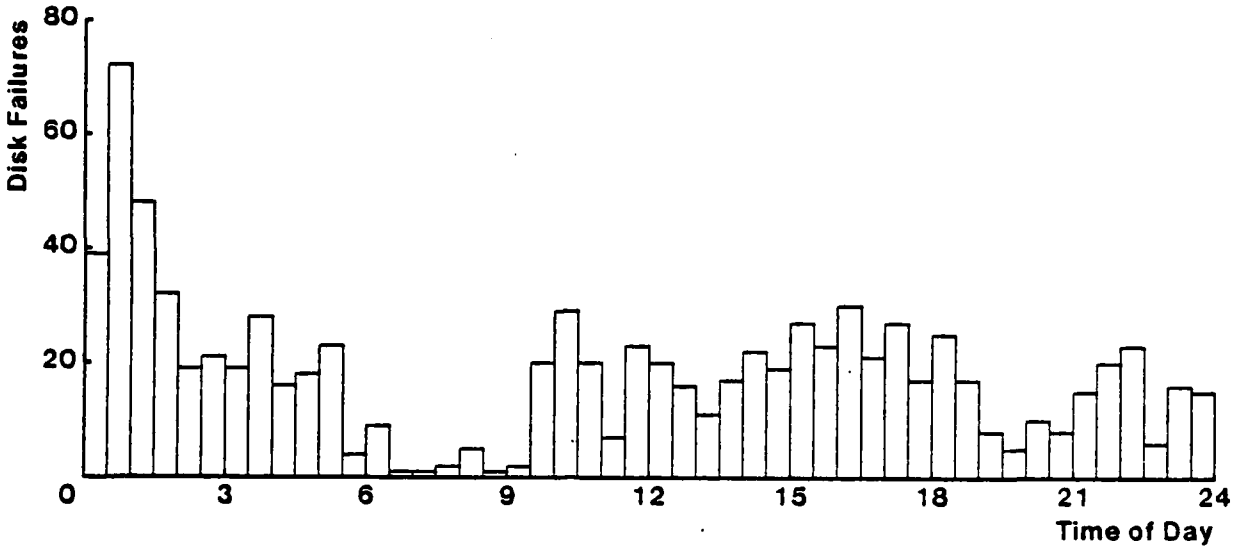


Figure 4-6: Disks failures as a function of time of day.

now exists to design and monitor experiments that vary load and observe the impact on system failure events [Segall et al. 1982].

4.4. Tools/Techniques

The error log analysis software packages that were derived to support the modeling effort are in routine use. These packages extract events from the error log and perform various statistical tests upon the data. Software packages also exist for extracting parameters for the cyclostationary models.

An error logger for the Unix operating system on VAX-11/780 and VAX-11/750 systems has been written [Markuson 1981]. This will allow us to collect data on Unix based processors.

4.5. Architecture

C.vmp has already demonstrated its ability to tolerate transient failures. Work is in progress to make Cm* and its operating systems more transient tolerant.

Exponential

$$R_e(\tau) = e^{\lambda_e \tau}$$

$$h_e(\tau) = \lambda_e$$

Weibull

$$R_w(\tau) = e^{-(\lambda_w \tau)^{\alpha_w}}$$

$$h_w(\tau) = \frac{\alpha_w \lambda_w}{(\lambda_w t)^{1-\alpha_w}}$$

Periodic

$$R_p(\tau) = e^{-\lambda_p \tau} e^{-F_p u(\tau)}$$

$$h_p(\tau) = \left[\lambda_p + F_p \frac{\partial u(\tau)}{\partial \tau t}\right]$$

Cyclostationary

$$R_c(\tau) = e^{-(\lambda_c + \sigma_{c1} + \sigma_{c2})\tau - \frac{\sigma_{c1}}{\beta_1}[1 - e^{-\beta_1 \tau}] - \frac{\sigma_{c2}}{\beta_2}[1 - e^{-\beta_2 \tau}] + \ln \phi(t)}$$

$$h_c(\tau) = \lambda_c - \sigma_{c1}[1 - e^{-\beta_1 \tau}] - \sigma_{c2}[1 - e^{-\beta_2 \tau}] - \frac{1}{\phi(t)} \frac{\partial \phi(t)}{\partial t}$$

Simplified Cyclostationary

$$R_m(\tau) = e^{-(\alpha_m - \gamma_m)\tau - \frac{\gamma_m}{\beta_m}[1 - e^{-\beta_m \tau}]}$$

$$h_m(\tau) = \alpha_m - \gamma_m[1 - e^{-\beta_m \tau}]$$

Table 4-3: Reliability and Hazard functions of the five compared models.

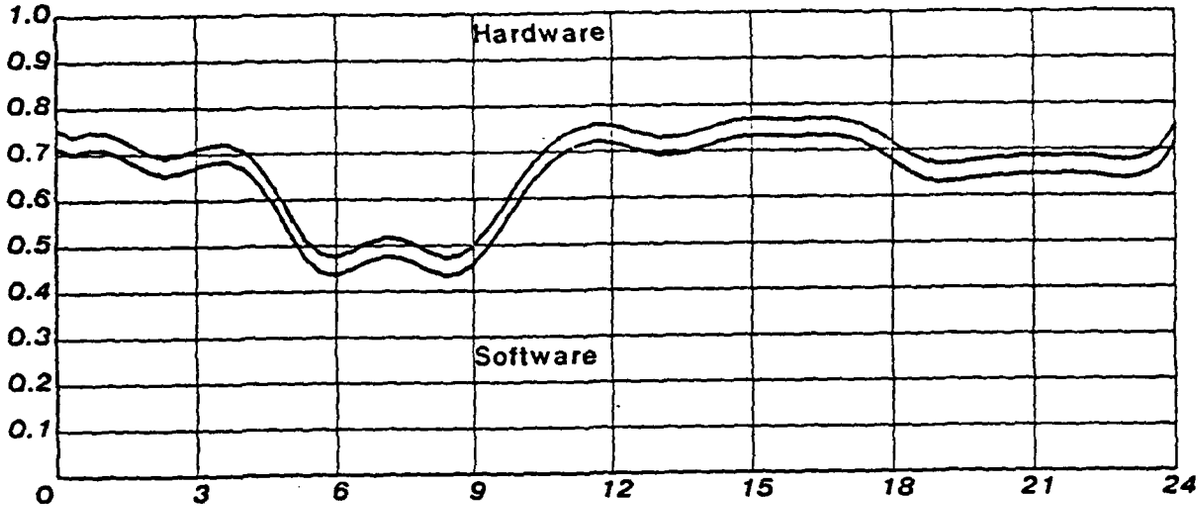


Figure 4-7: Probability that a crash is due to software or hardware as a function of the time of day.

5. Summary

We feel we have made substantial progress towards a designer's handbook for tolerating faults and evaluating alternative designs. Indeed, in the next one or two years, we feel we can build a Reliable Systems Designer's Workbench whereby a designer can interactively construct a cost effective computing organization. The concepts we have developed are contained in a textbook [Siewiorek and Swarz 1982].

6. References

[Barbacci et al. 1977] Barbacci, M., G. Barnes, R. Cattell, D. P. Siewiorek, "The ISPS Computer Description Language," Department of Computer Science Technical Report, Carnegie-Mellon University, 1977.

[Barbacci 1981] Barbacci, M., "Instruction Set Processor Specifications (ISPS): The Notation and Its Application," *IEEE Transactions on Computers*, vol. C-30, no. 1, January 1981, 24-40.

[Bell and Newell 1971] Bell, C. G. and A. Newell, *Computer Structures: Readings and Examples*, McGraw-Hill Book Co., New York, 1971.

[Canepa, Clark and Siewiorek 1977] Canepa, M., S. Clark and D. P. Siewiorek, "C.vmp: The Architecture and Implementation of a Fault- Tolerant Multiprocessor," *Proceedings of the FTCS-7*, Los Angeles, CA, 1977. Also Department of Computer Science Technical Report, Carnegie-Mellon University, December 1976.

[Castillo 1981] Castillo, X., "A Compatible Hardware/Software Reliability Prediction Model," Ph.D. Thesis, Department of Electrical Engineering, July 1981. Also Department of Computer Science Technical Report, Carnegie-Mellon University.

[Castillo, McConnel and Siewiorek 1982] Castillo, X., S. R. McConnel, D. P. Siewiorek, "Derivation and Calibration of a Transient Error Reliability Model," *IEEE Transactions on Computers*, vol. C-31, no. 7, July 1982, 658-671.

[Castillo and Siewiorek 1982] Castillo, X. and D. P. Siewiorek, "A Workload Dependent Software Reliability Prediction Model," *12th International Fault Tolerant Computing Symposium [FTCS-12]*, Santa Monica, CA, June 21-24, 1982.

[Elkind and Siewiorek 1980] Elkind, S. and D. P. Siewiorek, "Reliability and Performance of Error-Correcting Memory and Register Arrays," *IEEE Transactions on Computers*, vol. 29, no. 10, October 1980, 920-927.

[Elkind 1982] Elkind, S., "Reliability and Availability Techniques for Computer System," Chapter 3 in *The Theory and Practice of Reliable System Design*, D. P. Siewiorek and R. Swarz, Digital Press, Bedford, MA, 1982.

[Elkind 1983] Elkind, S. A., "LAMBDA User Manual," Departments of Electrical Engineering and

Computer Science, 1982.

[Kini 1981] Kini, V., "Automatic Generation of Reliability Functions for Processor-Memory-Switch Structures," Ph.D. Thesis, Department of Electrical Engineering, Carnegie-Mellon University, February 1981.

[Kini and Siewiorek 1982] Kini, V. and D. P. Siewiorek, "Automatic Generation of Symbolic Reliability Functions for Processor-Memory-Switch Structures," Special Issue on Reliable and Fault Tolerant Computing, *IEEE Transactions on Computers*, August 1982.

[Lai 1981] Lai, K. W., "Functional Testing of Digital Systems," Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, 1981.

[Lai and Siewiorek 1983] Lai, K. W. and D. P. Siewiorek, "Functional Testing of Digital Systems," *ACM IEEE 20th Design Automation Conference*, Miami Beach, FL, June 27-29, 1983.

[Markuson 1981] Markuson, D., "A VAX-11 Error Logger Under the Unix Operating System," Master's Thesis, Department of Electrical Engineering, Carnegie-Mellon University, 1981.

[McConnel 1981] McConnel, S. R., "Analysis and Modeling of Transient Errors in Digital Computers," Ph.D. Thesis, Department of Electrical Engineering, Carnegie-Mellon University, June 1981. Also Department of Computer Science Technical Report, Carnegie-Mellon University, 1981.

[McConnel, Siewiorek and Tsao 1979] McConnel, S. R., D. P. Siewiorek and M. M. Tsao, "Transient Error Data Analysis," Department of Computer Science Technical Report, Carnegie-Mellon University, May 1979. Also in *Proceedings 1979 International Symposium on Fault-Tolerant Computing*, Madison, WI, June 1979.

[McConnel and Siewiorek 1980] McConnel, S. R. and D. P. Siewiorek, "The CMU Voter Chip," Department of Computer Science Technical Report, Carnegie-Mellon University, 1980.

[McConnel and Siewiorek 1981] McConnel, S. R. and D. P. Siewiorek, "Synchronization and Voting," *IEEE Transactions on Computers*, vol. C-30, no. 2, February 1981, 161-164.

[Segall et al. 1982] Segall, Z., A. Singh, R. Snodgrass, A. Jones, and D. P. Siewiorek, "An Integrated Instrumentation Environment for Multiprocessors," Departments of Computer Science and Electrical Engineering, Carnegie-Mellon University, January 1982. Also in *IEEE Transactions on Computers*, vol. C-32, no. 1, January 1983, 4-14.

[Shombert 1981] Shombert, L., "The C.vmp Statistics Board Experiment," Master's Project, Department of Electrical Engineering, Carnegie-Mellon University, 1981.

[Siewiorek et al. 1978a] Siewiorek, D. P., V. Kini, H. Mashburn, S. McConnel and M. Tsao, "A Case Study of C.mmp, Cm*, and C.vmp: Part I - Experiences with Fault Tolerance in Multiprocessor Systems," *Proceedings of the IEEE*, vol. 66, no. 10, October 1978, 1178-1199.

[Siewiorek et al. 1978b] Siewiorek, D. P., V. Kini, R. Joobbani and H. Bellis, "A Case Study of C.mmp, Cm*, and C.vmp: Part II - Predicting and Calibrating Reliability of Multiprocessor Systems," *Proceedings of the IEEE*, vol. 66, no. 10, October 1978, 1200-1220.

[Siewiorek and Lai 1981] Siewiorek, D. P. and K. W. Lai, "Testing of Digital Systems," *Proceedings of IEEE*, vol. 69, no. 10, October 1981, 1321-1333.

[Siewiorek and McConnel 1978] Siewiorek, D. P. and S. R. McConnel, "C.vmp: The Implementation, Performance, and Reliability of a Fault- Tolerant Multiprocessor," *Proceed...gs of the Third USA-Japan Computer Conference*, October 1978.

[Siewiorek and Swarz 1982] Siewiorek, D. P. and R. Swarz, *The Theory and Practice of Reliable System Design*, Digital Press, Bedford, MA, 1982.

[Singh 1981] Singh, A., "Pegasus: A Controllable, Interactive, Workload Generator for Multiprocessors," Master's Thesis, Department of Electrical Engineering, Carnegie-Mellon University, December 1981. Also Department of Computer Science Technical Report, Carnegie-Mellon University, 1982.

[Tsao 1981] Tsao, M. M., "Transient Error and Fault Prediction," Ph.D. Thesis Proposal, Department of Electrical Engineering, Carnegie-Mellon University, January 1981.

[Tsao et al. 1981] Tsao, M. M., A. W. Wilson, R. C. McGarity, C-J. Tseng, and D. P. Siewiorek, "C.fast: A Fault Tolerant and Self Testing Microprocessor," *Proceedings of the Conference on VLSI Systems and Computations*, Pittsburgh, PA, October 1981.

[Tsao et al. 1982] Tsao, M. M., A. W. Wilson, R. C. McGarity, C-J. Tseng, and D. P. Siewiorek, "C.fast: A Single Chip Fault Tolerant Microprocessor," *12th International Fault Tolerant Computing Symposium (FTCS-12)*, Santa Monica, CA, June 21-24, 1982.

[Tsao and Siewiorek 1983] Tsao, M. M. and D. P. Siewiorek, "Trend Analysis on System Error Files," *13th*

International Fault Tolerant Computing Symposium (FTCS-13), Milan, Italy, June 1983.

[U.S. 1976] U.S. Department of Defense, *Military Standardization Handbook: Reliability Prediction of Electronic Equipment*, MIL-STD- HDBK-217B, Notice 1, 1976.

[U.S. 1982] U.S. Department of Defense, *Military Standardization Handbook: Reliability Prediction of Electronic Equipment*, MIL-STD- HDBK-217D, January 1982.

[Wensley et al. 1978] Wensley, J. H., L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliar-Smith, R. E. Shotak, and C. B. Weinstock, "SIFT: The Design and Analysis of a Fault-Tolerant Computer for Aircraft Control," *Proceedings IEEE*, vol. 66, no. 10, 1978, 1240-1255.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CMU-CS-84-123	2. GOVT ACCESSION NO. A142707 — 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): A SUMMARY OF FAULT TOLERANT COMPUTING RESEARCH AT CARNEGIE-MELLON UNIVERSITY	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Daniel P Siewiorek	8. CONTRACT OR GRANT NUMBER(s): N00014-77-C-0103
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA 15213	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Arlington, VA 22217	12. REPORT DATE: May 1984
	13. NUMBER OF PAGES: 34
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)