PARTITIONING OF DIGITAL SYSTEMS

Final Report

K. C. Mei, W. M. vanCleemput, M. Blout,

D. L. Hanson, T. S. Payne, J. Savir, L. K. Scheffer

TECHNICAL REPORT 165

April 1979

Computer Systems Laboratory
Stanford University
Stanford, California 94305

ABSTRACT

We consider the effects of implementation technology on the life cycle cost of large digital systems. Models are developed for both life cycle cost and reliability of the Navy's large shipboard systems.

A portion of a large processor is partitioned as an example into several technologies by both manual and automatic methods. Finally, a testability measure is developed and applied to the resulting partitions and conclusions are drawn concerning the attributes of various implementation technologies.

Key words: Testibility, automatic partitioning, design automation.

Partitioning of Digital Systems

Final Report

K.C. Mei
W.M. vanCleemput
M. Blount
D.L. Hanson
T.S. Payne
J. Savir
L.K. Soheffer

Technical Report 165

April 1979

Computer Systems Laboratory
Stanford University
Stanford, California 94305

This research was supported by the
U.S. Naval Underwater Systems Center
under contract number
N00140-78-c-8401

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

The aim of this study is to develop concepts and tools for understanding the influence of partitioning on the life-cycle cost of a system.

Throughout this study three types of boards will be considered as examples to illustrate the concepts being developed. These three board types are being used by the U.S. Navy for various types of equipment. The types considered are:

1. Type 1A: A small PC card with space for up to 8 IC's and a single 40-pin connector,

2. Type 2A: A PC card with space for up to 18 IC's and a single 100-pin connector.

3. Type 5X: A PC card with space for up to 55 IC's and two connectors: a 100-pin connector for the bark plane connection and a 30-pin test point connector to be used fur diagnostic purposes only.

Chapter 2 describes the extensions to the Design Automation system at Stanford to calculate the following life-cycle-cost parameters for a digital design specified hierarchically by means of the SCALD [MW78] design system: estimated procurement cost, heat dissipation, weight and volume, reliability, and system testability.

This program accesses the hierarchical data base containing the description of a system, including its partitioning and apply the appropriate algorithm to calculate the parameter for the total system in a bottom-up fashion.

In chapter 3, the influence of technology on the partioning problem is discussed. As an example, a major portion of the S-l computer system [MW78] is partitioned into boards of type 5X and into masterslice integrated circuits with up to 1000 gates per IC.

Chapter 4 addresses the problems associated with automat ed algorithms for partitioning digital systems.

In chapter 5, the influence of partitioning on a system's testability is discussed.

Chapter 2

DEVELOPMENT OF LIFE-CYCLE-COST-FACTOR EVALUATION TOOLS

2 . LIFE CYCLE COST MODEL

2.1.1 Introduction

The life-cycle-cost model used in this study is similar to that used in MIL_HDBK-246. Emphasis has been placed on parameters that are a function of the partitioning of the system.

The total life cycle cost CT is broken into two major categories : Acquisition cost, CA and Lifetime Support cost, CS.

CT= CA + cs [2.1]

The acquisition cost consists of the following major components:

1. cost of research, logical/physical design (CRD)

2. cost of fabrication (CFAB)

3. cost of installation (CINST)

4. cost of documentation (CDOC)

5. cost of supporting equipment (e.g. tools, test equipment) (CSUPP)

6. cost of facilities (CFACIL)

Costs (3-6) are relatively independent of the partitioning and technology. R/D cost (CRD) is affected by the partitioning and by the implementation technology (e.g. standard components versus custom integrated circuits). The cost of diagnostic test generation differs due to the difference in board complexities and the different number of board types. cost of fabrication is the total cost of equipment to be procured.

2.1.2 Acquisition Cost

The fraction of the acquisition cost which is relevant to the partitioning problem can be expressed by the following relationship:

$$CA = CD(1) + \mathrm{SUM}_{i=1,\dots,n(1)} [NE(i) * CM(i)] \qquad \textbf{12.21}$$

where CD(1) = Design cost for a design with partition 1;

NE(i) = The number of units of type i to be procured;

CM(i) = Unit cost of module type i;

n(1) = The number of module types for partition 1;

In MIL-HDBK-246, an example is given comparing the design cost of a SEM module design (1A or 2A) to a non-standard design such as 5X. It is postulated that the design cost of the SEM module implementation is very small, while the design cost for the non-standard design is very high. It is clear that the logic design cost of each of the implementations is roughly the same. However, the physical (packaging) design cost can be much higher for non-standard packaging technology.

2.1.3 Life-Support Cost

· · Life support cost is mor2 difficult to model due to various logistics organizations involved. Only the partitioning sensitive cost parameters will be given here:

cs = [n(1)*CLI] + [n(1)*CSC*NL] + Cspares + Crepair [2.3]

where CL I = Cost of introducing a line item into the supply system

CSC = shelf cost per year of maintaining a line item in the supply system

NL = planned operational life of the equipment

Cspares = cost of spares required over the system's life-time

Crepair = cost of all repairs made during the system's lifetime.

In order to estimate the cost of spares, the number of spare modules for each type must be derived from reliability calculations. For a given module type j, Pj(nj) is **defined** as the probability that nj spares of module type j will be sufficient over the planned life, NL, of the equipment.

Assuming independence of failure among modules and a constant failure rate for individual modules, then it can be shown that Pj(nj) is given by the Poisson distribution function as follows:

$$Pj(nj) = P(wj<nj) = \sum_{wj=0...nj} [e^{(-\theta j)} * \theta j^{wj}] / wj! \qquad [2.4]$$

where wj = total number of failures of the jth module during period NL;

θj = expected number of failures of module type j during period NL.

θj may be computed from the relationship:

$$\theta j = Lj * NL * NE(j) \qquad [2.5]$$

where Lj= failure rate for the jth module.

The adequacy of the equipment spares is the product of Pj(nj) over all modules and it must be greater than some minimum acceptable value A, or

$$\prod_{j=1...n(l)} P_j(n_j) > A \qquad [2.6]$$

$$C_{spares} = \sum_{j=1...n(l)} n_j * CM(j) \qquad [2.7]$$

In an ideal repair situation (the Perfect repair model), the faulty modules are isolated and then replaced by spares. No working (not faulty) modules are replaced. This requires near perfect diagnostic routines and/or undesirably long MTTR's. The other approach replaces all suspected modules at once (the unnecessary repair model). Many good modules will go through unnecessary repair procedures. Current research is focussing on modelling diagnostic resolution because it is the key factor in calculating repair costs.

2.1.4 MTTR Constraint

The time required to perform maintenance may be quantified by TMF and TMD, where:

TMF=field maintenance time

TMD=repair depot maintenance time

Each maintenace period may be broken into four activities: prepare, isolate, replacement(repair) and checkout. Mean-time-to-repair is simply the expected time to complete these activities,i.e.

$$MTTR = E(TM) \qquad [2.8]$$

In the repair depot, the time to isolate the failed component within a module is an exponential function of the module circuit complexity, while the time to prepare, replace and check out is relatively constant.

$$MTTRD = T1 + T2 * \sum_{i=1,...n(1)} [e ** (K1*Pi)] \qquad [2.9]$$

where: T1, T2 = constant overhead

K1 = model constant

Pi = ith module complexity measure (e.g. pin count)

In the field, the time to isolate the failed module is a function of system complexity, module complexity and the diagnostic resolution. Hence

$$MTTRF = T3 + T4 * \sum_{i=1,...n(1)} [e ** (K2 + K1*Pi/D(i))] \qquad [2.10]$$

where T3, T4 = constant overhead

K2 = model constant due to system complexity

D(i) = diagnostic resolution of module i (0<D<1)

The following system design constraint must be met:

$MTTRF < MTTRF\ max$ 12.111

$MTTRD < MTTRD\ mas$ 12.121

The cost of repair in LCCF is :

$Crepair = Labor\ Cost * (MTTRF + MTTRD) * NE * NL * FRsys$ t2.131

Estimating MTTR serves two purposes. One is to compute repair cost. The other is to estimate the system availability, AV.

$AV = MTBF / (MTBF + MTTR)$ [2.14]

During the mission time, the field availablilty AVF must meet certain minimum requirements:

$AVF = MTBF / (MTBF + MTTRF) > AVFmin$ [2.15]

$MTBF = 1 / FRsys$

2.1.5 Repair Cost Comparison Analysis

The following analysis provides a repair cost model for a 10gic syst3m using modular logic boards, Its purpose is to furnish a convenient tool to make repair cost comparisons between systems implemented with different board sizes.

When a failure is detected, depending on the diagnostic resolution of the diagnostic routines, two different repair procedures may take place:

1. perfect repair - only the faulty board is replaced

2. random repair - some number of cards are replaced even though only one card is faulty.

Let m be the number of boards per some arbitrary function such that it is the smallest entity that the diagnostics can verify. When such function failure occurs, the number of boards to be replaced will be a random variable from 1 to m. Hence, the expected value is $(m+1)/2$.

Let K be the ratio between the large board area and small board area. If a function takes A large boards, it will take at least K*A small boards. The cost of each board is assumed to be proportional to the board area (this is a questionable assumption made in [MIL-HDBK-246]).

The ratio of the failure rates is also K. In [MIL-HNDBK-246], a higher failure rate is credited to the large boards due to the lack of high-quality testing at the present time. Furthermore, no connector/backplane faults are accounted for here. Questionable small boards are assumed to be not repairable. Large boards are are assumed to be repaired at a cost of 50% of the original board cost. Therefore, each large board that has been replaced has a salvaged value of 50%. The following table highlights the comparison between SEM-IA and 5X boards.

	small board (SEM)	large board (5X)
number of boards per functi0n	6*A	A
average board cost ($)	100	600
mean number of boards to be replaced/failure	(6A+1)/2	(A+1)/2
(cost of new boards)	(6A+1)*100/2	(A+1)*600/2
repair cost of the faulty board ($)	0	300
repaired boards value	0	(A+1)*600/2
net cost per failure	(6A+1)*100/2	300
repair ($)	= 300A + 50	

From this simple comparison, the net cost per each failure assuming random repair strongly favor the large board. Note that we assume that both systems meet the MTTR and Availability constraints. Labor cost for repair task is not included.

2.2 RELIABILITY MODEL

2.2.1 Introduction

In this subsection, the current version of the model is presented. The goal of the model is to try to determine, and to quantify if possible, the difference in the reliability of two system implementations that differ only in the size of the circuit board.

An important assumption is that an attempt is made to repair every fault that is detected. A failure in a module will not necessarily cause a system crash if that module is a member of a set of redundant modules. However, if a failed redundant module is not repaired, the system is left vulnerable to a crash if other redundant modules of the set fail. Thus, it is reasonable to assume that an attempt is made to repair all detected faults.

The failure rate of the system, FRsys, is defined to be the sum of the failure rates of all of the modules in the system. The mean-time- betraeen-f ai lures of the system, MTBFsys, *is* given by 1/FRsys. If the system is used for a mission of time duration T, then the average number of failures that will occur during the mission is T/MTBFsys = T*FRsys.

In the following paragraphs, we will attempt to determine the difference in the system failure rates of two implementations of a system design. If the efficiency of the fault diagnosis and repair procedures are the same for both implementations, then the implementation with the lower failure rate is more likely to successfully complete the mission without experiencing a system crash.

The two board sizes will be called small (S) and large (L). The ratio of the area of the large board to the area of the small board is KA.

The basic elements of the system will be taken to be electronic devices (ICs, resistors, capacitors, etc.), circuit boards and connectors. The failure rate of the system is the sum of the failure rates of all of the basic elements. The system failure rate is broken down into three components, devices (D), boards (B) and connectors (C).

$$FRsys = FR\text{-}D + FR\text{-}B + FR\text{-}C \qquad [2.16]$$

2.2.2 Boards

We will indicate the parameters for a large board implementation by a subscript (1) and those for a small board implementation by a subscript (2). Let NB(2) be the number of small boards that are in a small-board implementation. It is assumed that the packaging, density is the same for the small boards and large boards. The number of large boards in the large-board implementation will then be:

$$NB(1) = NB(2)/KA \quad [2.17]$$

Let FRb(2) be the failure rate of a small board. If the failure rate of the board is proportional to the area of the board, then the failure rate of a large board is:

$$FRb(1) = KA * FRb(2) \quad [2.18]$$

The total failure rate for the boards in a small- and large-board implemented system are FR-B(2) and FR-B(1), respectively, where

$$FR\text{-}B(2) = NB(2) * FRb(2) \quad [2.19]$$

$$FR\text{-}B(1) = KA*FRb(2)*NB(2)/KA = NB(2)*FRb(2) = FR\text{-}B(2) \quad 12.201$$

This result should be obvious since we assume that the board failure rate is a function of board area and both systems have equal board area.

2.2.3 Connectors and Backplane

There is a well known rule, called Rent's Rule, that gives a relationship between the number of connector pins used per board and the number of chips per board. The rule is as follows:

NCP = NPPC * (NCPB ** r) [2.21]

where

NCPB: maximum number of components per board.

NPPC: number of pins per component.

NCP: number of connector pins for a given board

The parameter r is usually in the range [.60, .70]. If the chip density for boards is proportional to the board area, then on the average,

KA = NCPB(1) / NCPB(2) [2.22]

Then NCP(1) / NCP(2) = KA ** r [2.23]

Let KC be the average number of connector pins used on a small board. The failure rates of the connectors in a small- and large-board implemented system are:

FR-C(2) = NB(2)*KC*FRc(2) [2.24]

FR-C(1) = (KA**r)*KC*NB(2)*FRc(2)/KA [2.25]

respectively, where FRc(2) is the failure rate of a single connection. Furthermore, the backplane failure rate is a function of backplane area, which in turn is a function of the number of connector pins. For simplicity, FRc(2) should include connector failure rate and backplane failure rate.

The assumption is made here that both implementations are not limited by connector capacity limitations; this assumption may not always be realistic.

2.2.4 Devices

The device failure rate is almost implementation-independent. The primary difference is that the small-board implementation requires more driver chips to drive the board-to-board signals. In a small-board implementation, the board-to-board signals are more numerous and require a higher drive capability (due to higher line capacitances and fanout) than do similar signals in a large-board implementation. It has been observed that the reliability of an electronic device decreases as its junction temperatures increases. Thus, the small-board implementation requires a larger number of a device-type that are relatively less reliable than most other devices.

Let FR-D(1) be the failure rate of the devices in a large-board implementation. Then

FR-D(2) = (1 + KD) * FR-D(1) [2.26]

where $0 < KD < 1$, is the failure rate of the devices in a small-board implementation. KD represents the fractional increase in the overall device failure rate that is attributable to the extra driver chips.

At this point, we want to compare the system failure rates FRsys(2) (small-board) and FRsys(1) (large-board).

```
  FRdiff = FRsys(2) -FRsys(1)
         = FR-B(2) - FR-B(1) + FR-C(2) - FR-C(1)
                                   + FR-D(2) - FR-D(1)
         =  [NB(2)*FRb(2) - NB(2)*FRb(2)]
          + [NB(2)*KC*FRc(2)*( 1 - (KA**r)/KA) ]
          + [FR-D(1)*(1 + KD - 1)]                        [2.27]
```

The first term vanishes. The others are positive quantities because

[1 - (KA**r)/KA] > 0 if r<1, and KD>0.

Thus, we conclude that system with large cards has better reliability than that of the system with small cards,

FRsys(2) > FRsys(1) [2.28]

The difference in failure ra-te is contributed by the additional devices and connectors required in a small card system.

2.3 CONCLUSIONS

In the previous sections, a LCCF model for digital systems was developed. This model differs from the similar- one used in [MIL-HDBK-246] particularly because the emphasis has been placed on the factors that are a function of the partitioning of a design.

Although it is very difficult to draw conclusions without having accurate data, a few observations can be made: the aquisition cost is a relatively small portion of the life cycle cost. This cost is higher for larger boards, mainly due to the higher cost of physical design and to the decresing probability that a given design will be used again.

The support cost model centers around the f rcquency of repair, the number of spares required and the MTTR constraint . These in turn depend on the reliability of the systen and on the quality of the diagnostic procedures (diagnostic resolution>. A comparison was made between system with large and small boards. It was concluded that a design with larger boards is more reliable due to the reduced need for of f-board drivers and for connectors. Contrary to [MIL-HDBK-246 I we conclude that the repair cost for larger boar-d systems can be smaller.

Diagnostic resolution is defined as the probability of a functional diagnosis procedure to isolate a failure to a replaceable unit (a board). Larger boards have definite advantages over smaller ones in this regard. Since this measure involves both hardware design as diagnostic programming evaluation, much more research is needed in this area.

Chapter 3
THE INFLUENCE OF TECHNOLOGY ON THE PARTITIONING PROBLEM

3.1 INTRODUCTION

In order to evaluate the influence of technology it is necessary to isolate several important parameters:

1. The set of primitive modules to be used. This indicates the level of integration used in the design.

2. The nature of the technology employed (TTL, ECL, MOS, etc.)

3. The nature of a design (control logic, arithmetic unit, memory, etc.)

4. The nature of the packaging philosophy employed (e.g. printed circuit board size).

In order to study the influence of the level of integration on the partitioning problem an existing design for which a multilevel functional partitioning is available was studied. This system is the Stanford-l mark 1 processor [MW78]. This machine has been built at Lawrence Livermore Laboratory. It is a 36-bit , architecture with a speed of about 5 Mips, implemented using Motorola MECL-10K technology- The design consists of about 5000 integrated circuits,

The reason for choosing this example for an initial test is the existence of a machine-readable design specification, which includes a functional partitioning at several levels. Some of th Design Automation tools available at Stanford [Va77, VA78] were adapted for this study.

Various parts of the Stanford-l processor can be used to provide some insight into the influence of the nature of the design: (random) control logic vs. regular logic.

3.2 DESIGN TOOLS AND EXAMPLES USED

3.2.1 SCALD software and S-l

The SCALD system was obtained from Lawrence Livermore Lab [MW78] and installed on the IBM 370/168. The complete design of the S-l processor was obtained from Lawrence Livermore Lab and installed on the IBM 370/168.

3.2.2 Manual Partitioning Aids

The SCALD system allows only partitioning of a design into boards i.e. one level of physical hierarchy. Software has been completed for using f-our levels four levels of physical hierarchy i.e. chips, boards, racks, cabinets. A program has been written to convert the SCALD output into a list of signal nets for each chip, board, rack and cabinet.

3.3 INFLUENCE OF TECHNOLOGY

3.3.1 Introduction

Two different manual partitions of the EBOX of the S-1 system were carried out. The EBOX is one of two major subsystems of the S-1 processor; the second subsystem is the IBOX. The EGOS contains approximately 1600 MECL-10K circuit packages. The first will partition the system into 1000 gate custom LSI circuits. The second will partition the EBOX into boards of type 5X. In addition to the original design of the S-1, which contains 500 IC's per board, this provides three different manual partitions.

In the next chapter, an attempt to partition the EBOX by means of an automatic partitioning algorithm into boards of type 1A, 2A and 5X will be described.

3.3.2 Impact of LSI on system Partitioning

In order to dicuss the effects on system partitioning of an IC technology, it is necessary to discuss briefly the nature of the limitations of the ICs themselves. There are four primary limitations: gate count, pin count, heat dissipation, and drive capability. These limitations will be discussed, and the tradeoffs identified.

The first limitation is that of gate count. A high speed technology, such as ECL, is usually limited by the heat that can be dissipated by the chip. To obtain the high performance, each gate must dissipate a certain amount of power. In a technology of this type, as chips get larger and larger, the problem of heat dissipation is mot-e serious than thi! increased cost of the die.

In NMOS and CMOS technologies, on the other hand, cost is a more fundamental limitation. It is easily possible to produce chips so large that they are impossible to build economically, but that would work well if they were built. Here cost dictates the maximum gate count that may be used.

In all IC technologies, the maximum gate count per chip is rapidly increasing (by roughly a factor of 1.5 per year).

In 1979, these limits stood at about 1000 gates/chip for high performance, and at about 10000 gates/chip for high density logic.

The second IC limitation is pinout. The IC must communicate with the rest of the system, and it must do this through the pins of the package in which it is mounted. The number of pins required varies according to Rent's rule [2.21]. This is an empirically observed relationship, and not an ironclad law, but it does hold over a wide range of systems. There are exceptions, and they will be discussed later.

High speed designs cannot afford to spend time multiplexing different data over the same pins. Furthermore, more pins must be devoted to grounds, power supplies, and clocks to insure signal integrity.

In practice pinouts of IC's for high performance systems tend to be close to the largest available limit. New packages with larger pinouts are not easily introduced because the tooling costs are enormous. The largest packages that are widely used commercially are the 64 pin dual inline package, the 68 pin JEDEC chip carrier, and the 64 pin Intel-311 package [Ma79]. Recognizing the need for larger packages in the future, however, JEDEC standards are available for up to 156 pin packages, some of which have already been constructed for use in military systems [C179]. Further progress in this direction is inevitabie.

There are two common cases rrhere exceptions to Rent's rule are observed. The first exception occurs in the case of memories and other highly regular structures, such as PLAs. In these cases, the number of pins may only grow as the log of the number of gates. Thus a 64K random access memory, probably the most complex part in production at this time, requires only 16 pins. Furthermore, the same rule holds true for collections of these devices; a two dimensional array of these devices can be wired on a 2 sided PC board, with no denser wiring being required as the array grows larger. As digital systems grow larger, increasingly large sections of systems are composed of these regular structures, since they are easy to design and test. The S-1 processor, for example, is designed around its micro-code memories, which are regular arrays.

The second exception to Rent's rule may occur as a complete system is put onto a single chip. A typical microcomputer, for example, requires 40 pins. however, many of these are taken up by internal system functions, such as address and data busses, that are not visible to the user. If these functions are moved onto the processor chip, then these pins are available for user functions. Commercial examples of this type of effect are the Mostek 3870 or the Zilog Z-S. There are only 8 required pins on these processors, leaving 32 whose use is defined by the internal programming. In this example, the addition of the RAM and ROM to the original processor vastly decreased the required number of pins, rather than increasing them as Rent's rule would predict.

The third problem faced by an IC designer is heat. Reliability demands that the maximum chip temperature be kept below a certain limit, and the package has to dissipate the heat developed by IC without allouing the temperature to rise too far. This is a major problem with high performance technologies, that limits the number of gates that can be placed on a chip, as discussed earlier.

New designs for packages help to alleviate this problem somewhat. The new Intel-3M package [Ma79 I, and some of the implementations of the JEDEC chip carriers allow for increased heat sinking ability. This comes at the expense of decreased packing density (in the vertical direction).

Another problem faced by the chip designer is drive capability. This is a problem in both the high performance area, which requires controlled impedance environments, and in the high density areas, where the problem is the translation of the high impedance, low power signals found inside the chip to the low impedance, high power signals used for chip to chip communication.

In the high performance area, precious pins must often be dedicated for extra grounds, because of the impedance of the wire bonds and the speed of the transitions. In both the high performance and the high density cases, the drivers can use a significant fraction of the power of a chip, and are relatively slow compared to the rest of the circuitry. This problem for MOS technologies is dicussed in detail in [MC79]. The conclusion is that as MOS technologies move to higher densities, the output drivers will increase in absolute speed, but decrease in speed relative to the internal gates. Hence the communication problem between two MOS chips will get worse.

3.3.3 Effects of LSI on Design Criteria

When designing a new system, the decision has to be made whether or not to use LSI type technology. The decision will be based on the following factors: cost, performance, development time, reliability, testability, and repairability. Each of these factors is influenced strongly by the use of LSI, and each in turn affects the system partitioning.

The first, and often foremost factor is cost. Many different costs are affected by th2 decision to us2 LSI, am0ng then are development costs, production costs, and repair costs. Therefore life cycle cost is strongly affected by the decision to use LSI.

Development costs are, in general, higher with LSI than with MSI, in terms of both direct and indirect costs. There are two najor methodologies that are possible for a large system; either standard components may be interconnected to perform the desired function, or new components may be designed, and then interconnected. Traditional PC board design falls in the first category, and design with LSI falls into the second.

The first alternative, a PC card with conventional ICs, is very well understood, and many tools are available to help with the design. PC boards, even multi- layer ones, are comparatively easy to change, and hence easy to debug. Alternative forms of construction, such as wire-wrap, are available that provide similiar performance and are even easier to change, although they require more space.

The second alternative involves designing IC components, which is a much more difficult problem. The tools necessary to mechanize this process are still in research stages, and much of the work must be done by hand. It is extremely difficult to build breadboards that will give useful information about the performance of the final system, so almost all of the design must be done with the aid of simulators. Once a breadboard system is designed, and masks fabricated, then modifications to that system are difficult at best. Any errors must wait until the next turnaround to be corrected, For these reasons, the cost of developing a system composed of LSI is much higher than the cost of developing a system of interconnected standard components.

Once the system has been designed, then the cost of production is the next major cost to be considered. Here LSI, because of its smaller parts count, has a significant edge at high volumes. PC-type technologies have the edge at very low volumes.

Once the system is in the field, the major cost is the repair and maintenance cost. A LSI system may be either easier or harder to troubleshoot and repair than a MSI type system. This is discussed more fully under repairability.

The reliability of a LSI system is almost always better than that of an equivalent MSI system. One of the primary causes of failures in any system is interconnections, whether in the form of wire bonds, IC sockets, or PC connectors. An integrated system minimizes these problems. It will consume less power, and hence help the reliability of the power and cooling systems.

It is in repairability, however, that a LSI system may far outdo other implementations. The ideal self-diagnosing system would have an indicator light on each board, that goes on when there is a problem, while the system continues to perform normally. During the next preventive maintenance period, the board can be replaced. Memory subsystems attain this level of diagnosability now, thanks to the use of error-correcting codes. In principle the techniques exist to extend this type of performance to other, less regular, portions of digital systems. The problem, however, is that thes2 techniques, such as triple modular redundancy, require a large overhead in terms of extra logic. LSI may make it possible to implement this extra logic necessary without raising the cost of the system by an equivalent amount.

Performance is another area where LSI has an advantage. In high speed logic, the interchip delays are often the performance limiting factor. LSI can reduce the number of these delays to a minimum. LSI implementations consume considerably less power per gate function, since most of the gates are only driving other gates on the same chip. Weight and volume are also reduced; not only from the smaller size of the circuits themselves, but from the smaller power supplies and cooling apparatus that is required. Hence the proper use of LSI can result in a system that is higher in performance, smaller in size, and more reliable than other implementations.

There *is* one unique feature of LSI that must also be considered when making partitioning decisions. This is the pin count problem. LSI devices have high pin counts, and an aggregation of them will have still higher i/o pin requiremerits, as discussed in the section on the limitations of IC technology.

Each of these factors influences the partitioning of the system in different ways. To get the advantages of LSI, the number and length of interconnections should be minimized. This improves both reliability and performance. The performance is improved by several factors; each interconnection requires power to drive it, and will delay its signal somewhat.

3.3.4 Implications of LSI for Memory Subsystems

One specific case that must be considered is memory. In n-early any computer system built today, the main memory will be made of LSI components. Furthermore, modern system architectures may require other memory subsystems. In the S-1 system, for example, ther2 are 3 micro-code stores in addition to the main memory. Ilemory structures are ideally suited for LSI design, for they require pin counts that grow only with the log of the number of bits in the memory. With the addition of a small amount of hardware a memory can at least discover and report its own errors. A generalization of this technique, error correcting codes allows improved reliability and diagnosability, with relatively small over-head. When these codes are used, the memory usually gives advance notice of any degradation and repair may be post-poned until a more convenient time. The exact failing compo-nent in the memory array can be pinpointed easily. Since me-mory is such a common structure, and since such powerful techniques are available for improving its reliability, it deserves a closer study. How can LSI be used to fabricate the require=! memory, and what effect does it have on system partitioning?

The nos t economical method of constructing a main memory involves large arrays of memory chips. This is true for sev-eral reasons: to minimize the overshoot and ringing on the data and address lines, the line inductance must be kept low. This implies that the wires from the line drivers to the chips should be as short as possible. Line drivers tend to carry large amounts of current and for this reason they are unreliable. They also consume comparatively large am-ounts of power. Therefore, designers like to minimize the number of line drivers in a system. The only way to get short lines from the drivers to the memory chips and minim-ize the number of driver chips is to have large arrays of mem0ry chips with driver chips at the periphery. Nearly all commercial semiconductor memory systems are built this way. Clearly, to minimize component count, which wi 11 tend to mininize cost and increase reliability, a system must have the ability to support large, dense arrays of memory chips. In the construction of main memories, this characteristic will not change as LSI memory chips get larger, since the prinary constraint on main memory is cost, which is roughly independent of chip complexity. Small special purpose memories, such as control stores, will not need to be implemented as arrays, however, when the LSI densities get higher.

Unfortunately, there is a direct trade-off between memory size, in chips, and the ability to use error correcting codes. In order to be immune to single memory chip failures, the single error correcting codes require that no more than 1 bit of any word be stored in a given chip. This way, if any chip fails completely, there will be at most one error in the resulting word, and this can be corrected by the error correcting code. Multiple bit error correcting codes are known, but they are much harder to implement and involve considerably higher overhead. This implies that the memory wi 11 always have at least as many chips in a memory as there are bits in a word. A minimum size memory, therefore, will not have fewer chips as the scale of LSI increases. Instead the number of chips will remain the sane, and the capacity of the memory will grow.

For memory, the conclusions may be summarized as follows: main memories, which already have good reliability and repairability, will get larger as LSI technology advances. They will not get physically smaller past a certain limit, since they need a minimum chip count to obtain the error detection and correction desired. Auxilliary memories, on the other hand, will continue to shrink. These memories will not get the benefit of error correcting codes, at least on the chip level, and hence will not be immune to chip failures. However, these memories will have drastically reduced size and power dissipation.

3.4 <u>RESULTS</u>

3.4.1 <u>S-l System Partitioning into Masterslice IC's</u>

To examine the result of IC technology upon the partitioning of a large digital system, we partitioned the EBOX of the S-l into LSI chips. The assumptions that were made about the chips are appropriate to 1979 high performance technology.

We assumed 1000 gates/chip and 150 signal pins/chip as appropriate limits for a performance oriented technology in 1979. This type of density has been demonstrated for high performance ECL [Pr79]. The method that was used to partition the processor into LSI chips was rather simple: each chip in the existing design was assigned to one of the LSI chips.

It was not clear that this method would be adequate since the S-l was not designed with this type of technology in mind. In the large wire-wrap board technology used, the designers of the S-l were faced with entirely different tradeoffs than designers working with IC technologies. The S-l was designed to minimize chip count, and little attention was paid to the problems of interconnections, especially within a board. The strategy used was to supply a surplus of interconnection points, so that simple pin allocation algorithms would work well [MW78]. In an IC technology, we expected the opposite to be true, that is, that in a parallel machine like the S-l, interconnections would be the limiting factor, not gate count.

Somewhat suprisingly, the partitioning worked fairly well. It was not always possible to meet the goal of 150 pins/chip, but in all cases it was easy to see how this goal cou3.d be met if the circuitry was designed with an LSI implementation in mind, without changing the architecture.

As expected, the most important limitation on the amount of logic that could be put on one chip was the I/O pin count. The limit on the number of gates to a chip was seldom reached, and then on1y in the more regular parts of the machine.

The effects on reliability and testability of this approach are summarized elsewhere in the report.

3.4.2 Partitioning a Masterslice Design onto PC Cards

In the specific case of the 1A, 2A, and 5X boards, which one is best suited for new systems implemented with LSI? The 1A board is not acceptable, for it does not have enough pins for even one LSI device. Commercial micro- processor makers have already found that 40 pins is not nearly enough, and the new generation processors, in a wide variety of proccesses, all have more than 40 pins. This is true of both high density and high performance logic families.

The 2A and 5X boards both have the same number of pins, and so the maximum number of gates that could be put on either board, if space was not a limiting factor, is the sane. According to Rent's rule, it will only be possible to put 2-3 64 pin packages on one 100 pin board, This small number of chips can be put on a 2A board, and so a 2A board will be most appropriate for random or semi-random logic implemented in LSI.

Memories and other structured logic forms, however, do not obey Rent's rule. It is possible to put as many memory chips as space allows on a card, without exceeding 100 pins. It is highly advantageous to do this, since memories typically have a fixed overhead/card in terms of their peripheral components, such as line drivers, on card power regulators, and error correction and detection circuitry. Thus for the regular parts of the system, the 5X cards will prove superior. As systems become more complicated, there will be a tendency for more portions of the system to assume a regular form.

Therefore, the best card of the three is the 5X type, for general purpose use in a LSI system. If the system consists primarily of random logic, then the 2A may be a better choice. None of these boards, however, is optimum for LSI, for none of them have have enough pins. If the 5X boards are used, then many boards will be only partially full, for there will not be enough I/O pins to allow any more logic to be placed on the board. This space could be used, however, for adding logic to increase a system's testability. The 2A board does not suffer so much from this problem, but it does not allow for the implementation of arrays of chips. Thus it will suffer in performance.

What is needed is a board large enough to allow the efficient implementation of arrays of chips, with a large enough pin count to allow the same board to be filled with random logic. Such a board, with current packaging technology, would be roughly the same size as a 5X board, but with about 300 pins. This is enough pins to support about 16 chips of 1000 gate/chip LSI.

3.4.3 Partitioning of a Conventional Design onto PC Cards

Th2 manual partition of the EBOX onto 5X boards was carried out using a few basic groundrules. These rules were established in an effort to constrain the resulting partition to be a realistic sample of the use of the 5X board technology. Several partitioning techniques were used within the structure of the groundrules to arrive at the final result. The outcome of the project then represents a reasonable attempt to utilize the 5X boards to implement a portion of a large high performance digital computer. Although the entire S-l processor was not partitioned, the EBOX contains logic representative of both functional data and address paths and random control logic. The EBOX also includes a sizable control store memory to provide a representative RAM array portion of the design.

3.4.3.1 Groundrules and Techniques

The groundrules were chosen such that the partition produced uould represent an actual functional design *using* the 5X boards. The primary constraining groundrule was of course that the configuration of the 5X board be followed. That is, a maximum of 55 integrated circut packages and a maximum of 100 signal I/O pins. In addition, an attempt was made to construct a partition which used the minimum number of boards and the minimum number different board types.

Since the S-l design utilized Motorola's MECL 10K logic family, th2 partition was implemented taking into account the wiring rules for ECL logic. These rules have to do with maintaining the transmission line integrity of the nets which interconnect the logic elements. The main requirement imposed by these rules was that of avoiding stub connections on nets that cross from one card to another. This means that signals which cross card boundaries should be sourced on one card and only have loads on the source card or exactly one destination card.

A final groundrule was to leave the original S-f design intact and therefore the partition must contain all logic elements and all connections of the actual S-l EEOX. This rule was established partially to maintain a commonality among the various partitions being carried out and partially to insure that the resulting partition represented a truely functional design. Since the *S-l* design was a hierarchical specification of the function of the processor, the decision to follow the design exactly should not *impose* any undue constriants upon the partitioning effort.

Two major techniques were employed during the partitioning effort. The first was to utilize the functional breakdown of the design as it was created to define the physical boundaries of the partition. The second was to extract bit slice collections of the logic. This later technique has the advantage that multiple uses of a single board type are a direct result. These two approaches, however, turned out to be almost diametrically opposed in their respective goals. The problem is that the S-l design is based upon a 36-bit word and therefore contains a large number of 36-bit data paths. Consequently.. a functional block with one input data path and one output data path tends to exhaust the I/O pins available on a 5X board. The solution to this problem is of course to bit slice several functional blocks and place them together one the same board. This solution, however, produces boards which are less functionally defined and so forfeits the diagnosability and understandability of a true functional partition.

In addition to the above techniques, the partition was carried out in several iterations. This allowed the first attempts to focus on global information and thus create a general partition strategy which was as functional and logical as possible. Later iterations were then used to move around pieces of logic and fine tune the partition to meet the technology requirements.

3.4.3.2 Partition Results

The outcome of the partition effort of the EBOX onto 5X boards can be summarized as follows:

Number of Boards	45
Number of Board Types	22
Average Number of Chips/Board	37
Average Number of I/O Connections / Board	90

These results were obtained after several iterations and with two notable exceptions to the groundrules.

The first exception has to do with assumed changes to the original design. In many places throughout the EBOX, diagnostic multiplexors were incorporated to allow for the 0b-servation of various signals and data paths. Most of these multiplexors select 1 of 36 signals in a data path to be sent to the console for display. Since the partition that was obtained contains many bit slice type boards, the multi-plewors of the original design were spread across several boards. This causes unnecessary use of signal I/O pins, since the multiplexors are cut into pieces. All of the signals to be selected for vision purposes on a given board could just as well be selected by one whole multiplexor rather than several partial multiplexors. Consequently, it was assumed that appropriate design changes could be generated that would not affect the functional behavior of the processor but would decrease the number of I/O pins required by some of the bit slice boards.

The second exception has to do with the wiring rules for ECL logic. Specifically, the daisy-chaining rule was violated in some instances in order to decrease the number of I/O pins required on some boards. The violation occurs when a control signal generated on one board is needed by two or more bit sliced boards. Rather than use multiple I/O pins on the source board according to the rules, only one copy of the signal was sent and it was daisy-chained to the destination boards.

The primary limitation encountered during this project was the number of signal I/O pins allowed on each board. Hence, the partitioning effort was essentially driven by the attempt to utilize as many logic packages per board as possible within the I/O constraint. On a few rare occasions, a portion of the design was encountered which was essentially self-contained. These blocks of logic tended to fill a board with logic packages and not exhaust the available I/O pins. The RAM chip arrays are the primary examples of this situation.

3.4.3.3 Conclusions

The EBOX seems to have been a good typical example of the logic of a digital system. This is based on the empirically established guideline for the relationship between the number of signal I/O pins on a board and the number of circuit packages on that board, known as Rent's Rule. Solving this equation 12.21 I for r with NCP equal to 100 and NPPC equal to 14 and NCPB equal to 37 yields r=0.66. This is in strong agreement with the previous empirical data.

Partitioning of the EBOX was done with a strong emphasis on bit slicing because of the I/O constraints. Unfortunately, the design did not lend itself well to bit slciing. In several instances, the liberal use of control signals, that is a very horizontal control structure, precluded the use of the desired bit slice because too many I/O pins were required for the control signals. This horizontal control structure approach is good for generality and for ease of modification, but too many control signals severely hinder the bit slicing technique. There were also cases where the clarity of the control design interfered with an optimal bit slicing arrangement. One such case is the structure of the shifter multiplexor in the shift box. The design was structured around a simple binary encoding of the shift amount. This means that the first stage of the shifter required all the bits that were a multiple of 16 to be in the same slice. Since the word size is 36 bits, this amounts to needing every **fourth** bit on the same board. Consequently, the first stage of the shifter could not be included in the bit slice which contained its inputs, even though there was ample chip space on the shift box boards. The conclusion here is that, in general, the EBOX design does not lend itself to the bit slicing technique of partitioning.

The original design of the S-I processor was carried out assuming a packaging technology which allowed 500 circuit packages on each board and essentially an unlimited number of I/O pins. In short, **this means** that little or no thought **needed to** be given to the partitionability of the design at **design** time. However, partitioning the design onto a more restrictive packaging technology becomes a much more difficult problem. Although partitioning may be successfully completed, the final result does not make optimum use of the capabilities of the packaging technology. In conclusion, the constraint5 of the target packaging technology must be considered as part of the initial design constraints in order to attain an optimum system design.

Chapter 4

A COMPARISON OF MANUAL AND AUTOMATED PARTITIONING

4.1 SURVEY OF PREVIOUS WORK ON PARTITIONING

There are two basically different approaches to obtaining an optimal system partitioning.

The first one considers the problem from the point of view of synthesis whereby one tries to optimize some objective function. Host work in this area has dealt with relatively small systems and/or with simple objective functions.

A second approach is to analyze the properties of a given design as a function of the partitioning. This partitioning is usually obtained manually (this is the approach used almost exclusively in industrial practice).

The problem of partitioning a digital system is usually formulated as follows: given a set of components C(i), i=1,...,NC and a set of signal nets N(j), j=1,...,NS, partition this design into a minimum number of subsets (partition elements) subject to the following constraints:

1. the size of every partition element should not exceed KCPP components.

2. the number of signal nets, connecting the partition element to the rest of the design should not exceed NSPP.

Often slightly different and more complicated objective functions can be minimized, depending on the exact nature of the problem. In [La62] an algorithm is discussed that minimizes the number of connect ions between partition elements. In [LL69] the objective function is to minimize the delay in a digital system, subject to constraints of maximum area and maximum number of external pins per partit-ion element. In [KL70], the authors describe an algorithm for partitioning a graph with weighted edges into subgraphs such that the sum of the weights of all the cut edges is minimized, subject to a maximum size for each subgraph. In [RO71] a heuristic algorithm is discussed for partitioning a circuit subject to the usual constraints of maximum number Of components and maximum number of external signals per partition element . This algorithm allows duplication of components if this results in a smaller number of modules. For a simple example, a gate redundancy of 21% leads to an optimal result. The paper does not discuss the influence of this redundancy on the testability of a design. In [RW72] the ALMS partitioning system is described. This system, developed at IBM, results in partitions that are comparable to manually obtained partitions, but does so in a more cost-effective manner. The ALMS system consists of two major programs: a group generation program (GGP) that forms clusters of components and a group allocation program (GAP) that assigns these groups to partition elements. In [CG74] the problem of partitioning a system subject to minimal life-cycle cost is explored. In this paper the following assumptions are made: the life-cycle cost is substantially higher (10 to 100 times) than the initial procurement cost. Maintenance is performed using a discard at failure strategy. A major factor in the logistics cost is the number and variety of spare modules. In order for a system to be adequately maintainable there must be a high probability that a spare module will be available when a failure occurs during a prescribed mission time. An important problem associated with such an approach is the need for adequate fault diagnosis in order to pinpoint the failed module. However, in practice it is often difficult to achieve this accuracy in fault location and as a result many good modules are also discarded, resulting in increased maintainance cost and in a reduced probability that the system will remain operational for a given mission time. The algorithm proposed in [CJ74] tries to estimate and minimize total life-cycle costs subject to the following constraints for every partition element: maximum area, maximum number of external signals, maximum heat dissipation, maximum failure rate.

A more extensive survey of the partitioning problem can be found in [Ha72] and in [Ko72].

One of the constraints that often limits the optimality of a partition is the maximum number of external pins allowed for every partition element. Experimental evidence, collected at IBM, leads to Rent's rule [2.21]:

NCP = NPPC * (NCPB ** r)

where NCP = number of external signals per partition element

NPPC = number of pins per component

NCBP = number of components per partition element

r = 0.6.....0.8

The following table illustrates this rule:

NCPP	NCPP ** 0.66
2	1.53
5	2.89
10	4.57
20	7. 22
50	13.22
100	20.89

200	33.01
500	60.44
1000	95.50
2000	150.90
5000	276.26
10000	436.52

In [Hi70] the relationship between the number of external pins and the size of a partition element is studied in more detail. In the worst case the number of external signals is obviously equal to the total number of signals in that partition element. The number of external signals is reduced by sharing of pins and by burying of signals. In [Ra70] a more refined version of Rent's rule is presented. It is clear that Rent's rule is a heuristic tool that can be effective in certain cases. This rule cannot always be applied correctly. A simple illustration of this is the existence of microprocessors with equivalent gate counts far exceeding 1000 and with only 40 - 64 external connections. It is obvious that intelligent functional partitioning of a design can greatly improve on the external pin requirements suggested by Rent's rule.

4.2 AUTOMATED PARTITIONING

4.2.1 Introduction

Although for small designs it is possible to obtain automated partitioning results that are comparable to manual ones [RU72] it is not clear that the same results can be obtained for larger designs.

Therefore, as an extension to the previous section, some autonnted partitioning algorithms were implemented and integrated into the Stanford Design Automation System [Va78]. This capability was used to compare the results of these methods with manually generated functional partitions for reasonably large designs.

4.2.2 Sequential Partitioning Algorithm

The simplest of these algorithms is the sequential constructive algorithm [Ko72]. This algorithm assigns components one at a time to a board until that hoard's constraints are reached. It then starts the next board and continues until all the components are placed.

This algorithm requires a complete logic description of the circuit and both a connector capacity constraint and a board area constraint. It attempts to minimize the number of boards. The sequential constructive algorithm in detail is:

(STEP 1) Select an initial (seed) component.

(STEP 2) Assign this component to the board.

(STEP 3) Update the external connection list. This list contains all the nets that connect to both a component on the board and a component not on the board.

(STEP 4) Update the next possible component set. These components are the components that are directly connected to components already on the board or simply the unplaced components that connect to the nets in the external net list.

(STEP 5) Select a component from the possible component set. This selection should be based on the minimization of some function of the number 0f connections its placement would add and the amount of space on the board it will use. This function should account for integrated circuits with mulitple sections and connections that become internal to a board when the last component that connects to a given net is placed on the same board. Only components that will not cause the constraints to be exceeded can be selected.

Steps 2-5 are repeated until there are no more components left in the next possible component set whose placement will not violate one of the constraints.

(STEP 6) If the next possible component set is empty and the constraints have not been reached then select any unplaced component that will not cause the constraints to be exceeded then go to step 2. This is the case where all the components on this board connect only to components already placed on other boards and not to any unplaced components.

(STEP 7) The constraints have been reached so this board is complete. Go to step 1 to start the next boa'rd.

This sequential constructive algorithm is straightforward to implement, runs in a fixed amount of time, and always results in a feasible partition. Unfortunately since the placement decisions are based on only a very small part of the information available the resulting partitions are far from optimal. A more careful selection of seed components may improve the results. Allowing interactive selection of seed components may be a possible improvement. A two level look ahead may also help. This would involve basing the component selection decision not only on the effects of the components placement but also on the effects of the possible placement of the unplaced components it connects to, This two level look-ahead would require a much more complicated algorithm and result in at least an order of magnitude increase in execution time.

4.2.3 Parallel Constructive Partitioning Algorithm

The parallel constructive algorithm [Ko72] attempts to utilize more of the available information. This algorithm constructs boards in a parallel manner so the decision on placing any given component is made based on information on all the boards in the system instead of just one board. This algorithm requires a complete logic description of the circuit and both a board connection constraint and a components per board constraint. It also attempts to minimize the number of boards. The parallel constructive algorithm in detail is:

(STEP 1) Decide on a target number N of boards. This can either be done by the user or by the algorithm using some function. For example, a function of the number of total components and connections could be used.

(STEP 2) Select a seed component for each of the N boards by:

(a) Pick some unplaced component.

(b) Place this component on the board.

(c) Locate all the components that are connected to a component on the board.

(d) If n0ne of these components are already placed on a board and constraints are not exceeded then place these components on the board.

(e) Repeat steps (c) and (d) until either an already placed component is reached or constraints are exceeded.

(f) Repeat steps (a) through (e) until all the boards are seeded.

(STEP 3) Assign the remaining unplaced components to the boards by:

(a) calculate the costs, both space and connection, of placing each remaining component on each board.

(b) Select one component for each board that can be added at the least cost and still meets the constraints.

(c) Place these components on their respective boards.

(d) Repeat steps (a) through (c) until all the components are placed or until no components can be placed without exceeding the constraints.

If all the components have been placed then a feasible partition has been found. If all the components have not been placed then the number of boards (STEP 1) can be increased and the algorithm repeated or the remaining components can be placed on more boards either by hand or by repeating this or any other partitioning algorithm.

The quality and even the completion of a feasible partition depends heavily on the original decision on the board number. It may take several iterations of the algorithm to get a "best" or even successful feasible partition.

The parallel constructive algorithm is straightforward to implement and each iteration takes a fixed amount of time.

4.2.4 Eigenvector Method

W. E. Donath and A. J. Hoffman [DH72] have devised another constructive algorithm based on eigenvectors of connection matrices, They use these eigenvectors to calculate a distance measure between each pair of components. These distance measures are then used to decide which components should be grouped together and which comp0nents should be seperated onto different boards.

The first step in this algorithm is the construction of a connection matrix. This matrix has an entry for each pair of components; therefore it is of order n x n where n is the number of components. The value of each entry i,j in the matrix is related to how heavily the components i and j are mutually connected. If they are not directly connected the entry is zero. Donath and Hofman define these entries as follows:

$$C_{ij} = \underset{n\ N_{ij}}{\text{SUM}}\ G_{ij}(n)$$

where N_{ij} is the set of nets that connect to both components i and j and $G_{ij}(n)$ is defined as:

$$G_{ij}(n) = \begin{cases} \dfrac{4fn}{|n|} & \text{if } n \text{ is even} \\[2ex] \dfrac{4fn}{|n| - 1} & \text{if } n \text{ is odd} \end{cases}$$

fn is a constant associated with the net, usually one if the net is internal to the system and one half if the net is external to the system.

The second step is to construct a degree matrix. This matrix also is of order n x n and has entries defined as follows:

$$D_{ij} = 0 \quad \text{if } i=j$$

$$D_{ij} = C_{ij} \quad \text{if } i \neg= j$$

The third step is to construct a trace zero rnatrix. This matrix can be any matrix of order n x n and entries such that:

$$U_{ij} = 0 \quad \text{if } i=j$$

and

$$U_{ii} = 0$$

The fourth step is the computation of the eigenvalues Lr and eigenvectors Xr of the matrix C-D+U.

The fifth step is the selection of a target number of boards k. This can be given by the user or calculated in some manner.

The sixth step is to define $Yk=L1+L2+L3+...+Lk$ where L1...Lk are the k largest eigenvalues. The trace zero matrix should b2 varied and steps four and six repeated to minimize Yk.

The seventh step is to construct a distance matrix using the k eigenvectors that correspond to the k largest eigenvalues. It has order n x n and each entry is calculated as follows:

$$Dij = (Xri-Xrj)$$

Small distances (Dij values) imply that components i and j are strongly connected and favor being placed on the sane board.

Donath and Hoffman suggest the us2 of the following method to assign components to boards.

(STEP 1) Assign each component to a unique single element group.

(STEP 2) Sort all edges, connnc t ions between groups, in order of increasing distance.

(STEP 3) Make a complete pass through this sorted list and for each edge (i, j) combine groups i and j if their combination will not violate any constraints. These groups must be combined in such a manner that each new group can be split back into the groups it was made up of.

(STEP 4) Repeat steps 2 and 3 until no more groups can be combined.

(STEP 5) Select the biggest k groups, where k is the target number of boards and call these groups the basis set. The remaining groups form the excess set.

(STEP 6) Pick the largest group in the excess set.

(STEP 7) Merge it with the basis group that will yeald the smallest increase in pin count and does riot violate any constraints.

(STEP 8) If no merge is possible then divide this smallest group into the two groups it was made from. If the group contains only one component so no division is possible then the partitioning should be terminated as unsuccessful.

(STEP 9) Repeat steps 6 through 8 until all the excess groups are combined with the basis groups.

If the algorithm terminates as unsuccessful then the target number of boards can be revised upward and the algorithm repeated. A suggested alternative is to calculate the distance matrix using this algorithm and then do the actual component assignment with one of the other algorithms using the distance matrix as an aid in component placement decisions.

This algorithm is much more difficult to implement than the other constructive algorithms.

4.2.5 Iterative Improvement Algorithms

A family of non-constructive algorithms also exists. These algorithms are the iterative improvement algorithms. These algorithms take an already partitioned circuit and try to improve this partition by moving and/or interchanging components. The initial partitioning can be done manually, by using a constructive algorithm, or by some randon technique. These algorithms can be as simple as a random pairwise interchange or as elaborate as the interactive iterative technique of Hanan, Mennone, and Wolff [HM74b]. The pairwise interchange algorithm is implemented as follows:

(STEP 1) Pick two components, one on each of two boards.

(STEP 2) Compare the two components and if interchanging them will improve the partitioning then interchange them.

(STEP 3) Repeat steps 1 and 2 until some termination condition is met. Possible termination conditions include time limits, number of interchange limits, and a minimum number of sequential comparisons that result in no interchange.

Since for any circuit of practical size an exhaustive pairwise interchange can take an almost unlimited amount of tine, selection of the sequence of pairs to be interchanged is very important. Since there is no really good way to select these sequences of pairs automatically Hanan, Mennone, and Wolff leave this selection to the user. Their interactive iterative algorithm is implemented as follows:

(STEP 1) Generate an initial partition using a parallel constructive algorithm.

(STEP 2) Remove some components from some boards interactively. Removals are restricted to the last component or components placed on each board by the parallel constructive algorithm. This insures that the physical constraints of the boards are not violated.

(STEP 3) Reassign the removed components to boards in one of two ways. Either assign them by user command or by the use of the last part of the parallel constructive algorithm.

The interactive iterative algorithm can be implemented in a straightforward manner and results in a feasible partition. It is limited by the fact that it is not entirely automatic and does require user input to be successful.

4.2.6 Automatic Partitioning Effects

If the only success criteria considered is the minimization of board number and the only constraints are a connection limit and a component per board limit then some form of these algorithms can be used to generate feasible partitions with a good success rate. Unfortunately these algorithms do not take into accOunt 0ther very important success criteria such as system reliability, system testability, and circuit board repeatability.

Although the system reliability depends more on the original design and on the constraints than on the actual partitioning some things can be done in a partitioning algorithm to improve reliability. The probability of a system failure in a given time interval, is defined as follows:

$$FRsys = FR\text{-}D + FR\text{-}B + FR\text{-}C$$

Wher2:

FR-D is the component failure rate.

FR-B is the board failure rate.

FR-C is the connector/backplane failure rate,

The component failure rates depend upon the number and type of components used, as given in the original design, and on the component operating temperatures. The partitiong algorithm can improve the component reliability by insuring that heat generating components are not clustered together creating high operating temperatures. The algorithm can be modified so that component placement decisions are based on mutual power dissipation as well as interconnectivity. Since minimizing the number of boards and connections also tends to minimize the board, connection, and backplane failure rates automatic partitioning algorithms for the most part adequately handle reliability constraints.

Testability also depends on the original constraints but partitioning has a wide ranging influence on testability. Goundan [GH76] has developed an algorithm that uses fault class isolation and fault class splitting to increase testability. His algorithm attempts to find a partition in which the maximum number of boards to which any fault is resolvable is minimized. The algorithm first locates all equivalent faults. All components with the same equivalent fault are combined into a set called a gate cluster. These gate clusters are treated as 'super' components and partitioned using some other algorithm. Fault splitting techniques are then used to enhance the testability of the resulting partition.

A third inadequately considered constraint is that of circuit board repeatability. Repair costs as well as initial' system costs depend heavily on the number of circuit board types used in a system. None of these algorithms attempt in any way to generate or use standard types of boards.

4.2.7 Results

Results: The S-l EBOX REGISTER FILE was automatially partitioned into 11 25-IC boards as compared to 8 boards obtained by manual partitioning. The eleventh board had only 3 modules on it.

The connection limits were reached well before the space limits were reached in the automatic partitioning so changing the space limit to 55 IC's and even changing the space and connection weighting factors had no significant effect on the partitioning.

The results of partitioning the Ebox of the S-l system with memory removed (all connections to memory sections are treated as external) are shown in Table 4. 1 for the serial algorithm.

# IC's Board	# Boards	max. Connections to any Board	average # Connections
10	96	101	68
20	47	174	124
40	24	325	205
80	12	524	376
160	6	804	603
320	3	1230	1119
460	2	1228	1228

Table 4.1

· The results of partitioning the Ebox of the S-l system with only part of the memory removed are shown in Table 4.2 for the serial algorithm.

# IC's per Board			# of	max. # Conn. to any Board		Average # Conn. Board
lmt	av	max	Boards	limit	actual	
8	5	8	20 1	40	4 0	38
18	15	18	69	100	100	91
55	16	37	65	100	100	98

Table 4.2

Board Size	# Boards	Average # Connections
10	96	70
20	43	122
40	24	199
30	12	335
160	6	531
320	3	904

Table 4.3

		Manual		Sequential		Pat-al lel	
brd size	pin limit	pin av	ic av	pin av	ic av	pin av	ic av
18	100	87	17	91	15	92	15
55	100	94	33	93	17	93	17

Table 4.4

Table 4.3 shows the average number of boards and connector pins in function of the board size. Table 4.4 compares board types 2A and 5X for both manual and automatic partitioning approaches.

4.3 CONCLUSIONS

All the practical automatic partitioning algorithms available at the present time are heuristic in nature. These algorithms span a wide range of complexity and effectiveness. They can be resonably effective when the only constraints are component per board limits and connection limits and when the objective is the minimization of the nunber of boards. Unfortunately their effectiveness when the criteria includes such things as board repeatability and testability is so low that their practical use is limited.

Chapter 5

THE INFLUENCE OF PARTITIONING ON SYSTEM TESTABILITY

5.1 INTRODUCTION

The partitioning of a system can affect the ability of that system to remain operational over the given mission time. Two different partitions will have different failure rat es when the failure rates of the boards and connector pins are included. Also, each partition will have a different sparing requirement. Any restriction on the number of spares that can be included during a mission effectively limits the number and nature of the repairs (through replacement of faulty boards> that can be made. We will develop a model for predicting the mission survival probability of a system given data on its partition. Using this model, we can compare the mission survival probability of two partitions of a system (i.e for the same system implemen-ted with two different printed circuit board sizes).

A fundamental question is how does partitioning affect the testability of a circuit? To provide insight and data to help answer that question, the following was studied: given specific testing and diagnosis procedures, one can develop heuristic and algorithmic procedures for measuring the detectability and diagnosability of the tests. The analysis procedure can be applied to partitions of a given system that result from the use of different partitioning algorithms and board sizes. The goal of this study will be to note and explain the conditions that lead to the higher levels of detectability and diagnosability.

5.2 COVERAGE AND SURVIVABILITY

In most digital systems, procedures called fault recovery procedures are provided to aid in system recovery from faults. Examples of fault recovery procedures are fault detection testing, fault diagnosis and replacement of' faulty field-replaceable-units. Faults that are successfully handled by the fault recovery procedures result in minimal, if any, system downtime. Other faults can cause severe and lengthy system outages. In such cases, a craftsman or field engineer and/or external maintenance equipment must be used to bring the system up. This situation can often result in widespread replacement of suspected circuits, which will be called an anomalous repair condition.

Let COV = the probability that a fault is successfully handled by the fault recovery procedures

1 - COV = the probability that a fault results in an anomalous repair condition.

It can be shown that under certain reasonable assumptions, the **mean time between** anomalous repair situations is:

MTTR = 1/((1-COV)*FRsys) [5.1]

The important item to **note** is that a smaller system failure rate increases the mean time between anomalous repair conditions.

Increasing COV will also increase the system's mission survivability. The way to increase the value of COV is to improve the fault recovery procedures. Later we will examine the effect of circuit board size on the quality of the fault recovery procedures. However, before this next step can be carried out, it is necessary to **have** knowledge of the fault. detection and diagnosis procedures **that** are used on the class of systems that are of concern. Some fault recovery procedures can be insensitive to circuit board size, while other others can be highly sensitive to board size.

5.3 TESTABILITY MEASUREMENT

5.3.1 Introduction

Some of the parameters that influence testability are:

1. Test Point Ratio (TPR)

2. Number of levels of combinational logic

3. Number of levels of sequential logic

4. Ability to synchronize the sequential logic to a given state.

If we assume that the design is not supposed to change when going to a different partitioning, the only parameter which is partition-dependent is the TPR (defined as the ratio between the number 0f pins available at the connector and the total number of signals on the board.

However, if we allow the design to change (without affecting it functionally) we can increase the TPR. One way to do this is use multiplexers to observe the unreachable lines within a board.

5.3.2 Test Point Ratio

In the previous section, a simple model for testability based on the test point ratio (TPR) was discussed.

$$TPR = NOS / TNS \qquad [5.2]$$

where NOS = number of observable signals.

TNS = total number of signals.

A worst-case estimate for TNS assumes that every signal connects exactly two pins. Then TNS can be derived as follows:

$$TNS = (NCP + NCMP * NPPC) / 2 \qquad [5.3]$$

where NCP = number of connector pins

NCMP = number of components per board

NPPC = number of active pins per component.

A reasonable estimate for NPPC is 12. Both NCP and NCMP depend on the actual board characteristics. The following table gives TPR for each of the three board types considered in this study.

Type	NCP	NCMP	TNS	NOS	TPR
1A	40	6	112	40	0.351
2A	100	18	316	100	0.316
5X	100	55	760	130	0.184

Table 5.1

5.3.3 Diagnostic Resolution

In many large systems diagnosis is performed using functional diagnostics. The result of such a incomplete diagnostic procedure is to isolate the fault to a functional entit):. Let a function F be implemented from n low-level components. Then the number of boards over which the function is spread lies between n and n/NCPB, where NCPB is the maximum number of components per board. Hence, the probability of isolating a failure through functional diagnostics is larger for a large-board implementation.

In calculating the MTTR constraints, it is desirable to achieve only identifying and replacing the failed board. . Unnecessary swapping of suspected boards not only requires more sparing but it also increases the MTTR. We can define diagnostic resolution as the probability of isolating a failure to a single board by runnning a set of functional diagnostics. Diagnostic resolution can be used as a quantitative measure of the quality of the partitioning. Diagnostic resolution is a function of the hardware design as well as of the diagnostic program. One possible way of comparing the diagnostic resolution of two systems would be to perform a functional fault simulation of the design alternatives.

5.4 TESTABILITY EVALUATION PROGRAM

5.4.1 Introduction

To evaluate the effects of different board sizes on system testability some method of measuring testability is required. Therefore a testability estimation program was implemented and integrated into the Stanford Design Automation System. This capability was used to study the effects of partitioning and board size on testability.

5.4.2 Testability Estimation Algorithm

The algorithm used in the testability estimation program was based on an algorithm developed by Stephenson and Grason [St76] to provide a numerical estimate of the ease of generating fault detection tests for register-transfer level circuit designs. Since the testability of a circuit is a function of how easy it is to observe and control its signal lines, Stephenson and Grason defined the four following values. The observability of a given signal is a measure of how easy it is to determine what the logic value of that signal is. It is a real number between 0 and 1 with larger values for more observable signals. The controllability of a given signal is a measure of how easy it is to control what the logic value of that signal is. Likewise it is a real number between 0 and 1 with larger values for more controllable signals. The Observability Transfer Factor is a value defined for each module type. It has a real value between 0 and 1 and is used for the calculation of the observabilities of the input signals to a module given the observabilities of the module's output signals. Similarly the Controllability Transfer Factor is a value defined for each module type. It has a *real* value between 0 and 1 and is used for the calculation of the controllabilities of the output signals of the module given the controllabilities of the inputs to the module.

Eef ore executing the algorithm both Observability and Controllability Transfer Factors for all the components were calculated by hand and stored in a library. These factors were calculated using the method developed by Stephenson and Grason [St76]. The first section of the algorithm calculates an Observability Transfer Factor for each board as follows:

1. Assign a value of 1 to the observability value of each output signal on the board. These are the signals that originate on this board and connect to another board.

2. Assign a value of 0 to the observability value of the remaining signals on the board.

3. Select a component on the board.

4. Calculate the average of the observability values of the component's outputs.

5. Multiply this average by the component's Observability Transfer Factor.

6. Assign this value to each of the component's input signal observabilities where it is larger than that signal's observability value.

7. Repeat steps 3 to 6 until all the components on the board have been selected.

8. Repeat steps 3 to 7 until the total change in all of the input observabilities is less than some constant.

9. Calculate the average observablity value of the signals that are inputs to the board. This value is the board Observability Transfer Factor.

The next step is to calculate the board Controllability Transfer Factors in the following manner:

1. Assign a value of 1 to the controllability value of each input signal on the board.

2. Assign a value of 0 to the controllability value of the remaining signals on the board.

3. Select a component on the board.

4. Calculate the average of the controllability values of the component's inputs.

5. Multiply this average by the component's Controllability Transfer Factor.

6. Assign this value to the component's output signal controllabilities.

7. Repeat steps 3 to 6 until all components on the board have been selected.

3. Repeat steps 3 to 7 until the total change in all of the output controllabilities is less than some constant.

9. Calculate the average observablity value of the signals that are inputs to the board. This value is the board Observability Transfer Factor.

The system Observability and Controllability Transfer Factors are now calculated using the above steps with the boards treated as components and the system treated as a board. The system testability estimate is equal to the square root of the product of the system Observability and Controllability Transfer Factors.

5.4.3 Addition of Extra Testpoints

It is possible to increase the testability of a circuit by adding extra connection pins to a board for use for testing purposes only. These test points increase both the observability and the controllability of the circuit. The effects of these extra text points were incorporated into the testability estimatiOn program by making the following changes:

1. The controllabilities and observabilities of the signals on each board were calculated as before.

2. Given n tes tpoints then the n signals with the lowest controllabilities and observnbilities were selected.

3. The algorithm is now repeated with these signals treated as board outputs in the observabil i ty calculation and as board inputs in the controllability calculations.

5.4.4 Addition of Extra Logic to Enhance Testability

In many systems there are boards where all the space on the board has not been fully utilized due to connection constraints. This extra space on the board can be used for added logic to increase system testability. This is usually done by adding multiplexers or some other logic to make signals more observable and controllable. Since generation of Several actual redesigns of a example for evaluation was impractical the simplifing assumption that increased logic was equivalent to adding a proportional number of testpoints was made. The testability estimation program was enhanced so extra testability logic could be considered by adding steps where the extra space on each board was calculated and transulated to an equivalent number of test points.

5.4.5 Results

Several partitions of the Sl-EBOX were generated manually and by using an automatic partitioning algorithm discribed in a previous chapter. The results of the evaluation of these partitions using the testability estimation program are summarized below.

IC LIM	PIN LIM	PART METH	TST PTS	EXTRA LOGIC	BRD TEST	SYS OBS	SYS CTRL	SYS TEST
55	100	MAN	0	NO	.30	.074	.143	.105
55	100	MAN	40	NO	.33	.114	.154	.132
55	100	MAN	0	YES	.39	.140	.154	.147
55	100	MAN	40	YES	.39	.140	.154	.147
55	100	AUTO	0	NO	.37	.069	,184	.114
55	100	AUTO	40	NO	.39	.037	.183	,129
55	100	AUTO	0	YES	.38	.086	.183	,127
55	100	AUTO	40	YES	.39	.088	.188	.129
18	100	AUTO	0	NO	.36	,063	.184	.107
13	100	AUTO	0	YES	.38	.OGS	.186	.114
8	40	AUTO	0	NO	.43	.059	.133	.OS8

Table 5.2

5.5 CONCLUSIONS

From the results obtained in the previous section, it is clear that a manualy partitioned design results in a higher degree of testability. It is also clear that larger board sizes yield a higher degree of testability using the Stephenson-Grason measure.

The partitioning and testability study can serve as the foundation for future studies. Built-in testing using self-checking circuits [Wa73], scan in/scan out of internal test points [Wi73],[Ei77], and signature analysis [Pa76 I, [HP76] are examples of design techniques that result in improved system testability and diagnosability. The reliability evaluation model might be extended to include models of module failure and the effects of periodic maintenance. Some work has been done on expressing the testability of a circuit in terms of certain physical characteristics like fan-out, logic depth and feedback paths [St76 I. This work might be extended to a comparative analysis of the testability of nodules or boards that result from various partitioning algorithms.

REFERENCES

[CJ74] Caponecchi, A.J. and Jensen, P.A., "A Partitioning Technique for Obtaining Solutions to the Modularization Problem," Naval Research Logistics Quarterly, vol. 21, no. 1, pp. 13-40, 1974.

[C179] Clark, R.J. et al "Microelectronics Packaging Trends," Electronic Packaging and Production, January 1979, p. 89

[DH72] Donath, W.E. and Hoffman, A.J., "Algorithms for Partitioning of Graphs and Computer Logic Based on Eigenvectors of Connection Matrices", I.B.M. Tech. Discl. Bul. Vol 15, No 3, (Aug 1972).

[DH73] Donath, W.E. and Hoffman, A.J., "Lower Bounds for the Partitioning of Graphs," IBM J. of Research and Development, vol. 17, no. 5, pp. 420-425, Sept. 1973.

[Ei77] Eichelberger, E.B., T. Williams, "A Logic Design Structure for LSI Testing" Proc. 14th Design Automation Conf., New Orleans, La., June 1977, pp. 462-468.

[GH76] Goundan, A. and Hayes, J.P., "Partitioning Logic Circuits to Maximize Fault Resolution", Proc. 13THa 1 Design Conference (June 1976).

[Ha72 1 Hanan, M., "The Partitioning Problem," Int. Seminar on Design Automation of Digital Systems, Weizman Institute of Science, Dec. 1972.

[HM74a] Hanan, M.; Mennone, A. and Wolff, P.K., "An Interactive Man-Machine Approach to the Computer Logic Partitioning Problem,", Proc. 1 1th Design Automation Workshop, Denver, June 1374, pp. 70-81.

[HM74b] Hanan, M.; Mennone, A. and Wolff, P.K., "Iterative Interactive Technique for Logic Partitioning," IBM J. of Research and Development, vol. 12, no. 4, pp. 328-337, July 1974.

[Hi70] Hitchcock, R.B., "Partitioning of Logic Graphs: A Theoretical Analysis of Pin Reduction," Proc. 7th Design Automation Workshop, San Francisco, June 1970, pp. 54-63,

[HP76I "Designer's Guide to Signature Analysis," Application Note 22, Hewlett-Packard Corp., Palo Alto, Ca.

[Ko72] Kodres, U.R. "Partitioning and Card Selection," Ch. 4 of M.A. Breuer, "Design Automation of Digital Systems," Prentice Hall, 1972.

[KL70] Kernighan, B.W. and Lin, S., "An Efficient Heuristic Procedure far Partitioning Graphs," Bell System Tech. J., vol. 49, no. 2, pp. 291-307, Feb. 1970.

[LS71] Landman, B.S. and Russo, R.L., "On a Pin Versus Block Relationship for Partitions of Logic Graphs," IBM Research Rept. RC 3088, Oct. 1970; also: IEEE Trans. Computers, vol. C-20, no. 12, pp. 1469-1479, Dec. 1971.

[La62] Lawler, E.L., "Electronic Assemblies with a Minimal Number of Interconnections," IEEE Trans. Electronic Computers, vol. EC-II, pp. 86-88, Feb. 1962.

ILL691 Lawler, E.L.; Levitt, K.N. and Turner, J., "Module Clustering to Minimize Delay in Digital Networks," IEEE Trans. Computers, vol. C-18, no. 1, pp. 47-57, Jan. 1969.

[Ma79] Markstein, H. "Chip Carrier Update," Electronics Packaging and Production, April 1979, pp. 79ff.

[Mc78] McWilliams, R. "A Color Graphic Terminal," Stanford University, Digital Systems Laboratory, to be published.

[MW78] McWilliams, T. and Widdoes, L.C. "The Stanford-l Multiprocessor," Stanford University, Department of Computer Science, 1978.

[MC79] Mead, C. and Conway, L. "Introducation to VLSI Sys terns," Academic Press, 1979

[Pa76] Parker, K.P., "Compact Testing: Testing with COmpressed Data," Digest 6th Int. Symp. on Fazult Tolerant Computing, June 1976, pp. 93-99.

[Pr79] Prioste, J. et al "Functional Array Eases Custom ECL Design, "Electronics, 15 February 1979, p. 113

[Ra69] Radke,C.E., "A Justification of and an Improvement on a Useful Rule for Predicting Circuit to Pin Ratios," Proc. 6th Design Automation Workshop, Miami Beach, June 1969, pp. 257-267.

[Ru72] Russo,R.L., "On the Trade-Off Between Logic Performance and Circuit-to-Pin Ratio for LSI," IEEE Trans. Computers, vol. C-21, no. 2, pp. 147-153, Feh. 1972.

[RO71] Russo,R.L.; Oden,P.H. and Wolff,P.K., "A Heuristic Procedure for the Partitioning and Mapping of Computer Logic Graphs," IEEE Trans. Computers, vol. C-20, no. 12, pp. 1455-1462, Dec. 1971.

[RW72] Russo,R.L. and Wolff,P.K.,"A Ccmputer-Based Design Approach to Partitioning and Mapping of Computer Logic Graphs," Proc. IEEE, vol. GO, no. 1, pp. 28-34, Jan. 1972.

[St76] Stephenson,J., J . Grason, "A Testability Measure for Register Transfer Level Digital Circuits," Digest 6th Int. symp. on Fault Tolerant Computing, June 1976, pp. 101-107.

[St70] Stone,H.S., "An Algorithm for Modular Partitioning,", J. ACM, vol. 17, no. 1, pp. 132-135, Jan. 1970.

[Va77] vanCleemput,W.M. "An Hierarchical Language for the Structural Description of Digital Systems," Proc. 14th Design Automation Conf., New Orleans, June 1977, pp. 377-385.

[Va78] vanCleemput,W.M. "A Structured Design Automation Environment for Digital Systems," Digest IEEE COMPCON, San Francisco, Feb. 1973, pp. 139-142.

[vaN] vanCleemput,W.M.,"Computer Aided Design of Digital Systems - A Bibliography". Computer Science Press, 1976-1979 (4 volumes).

[Wa73 I Wakerly,J.F., "Lo;:-cost Error Detection Techniques for Small Computers," Tech. Report 51, Digital Systems Laboratory, Stanford University, Stanford, Ca., December 1973.

[Wi73] Williams, M.J.Y., J.B. Angell, "Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logic," IEEE Trans. on Computers, vol. c-22, 1973, pp. 46-60.

Appendix A

LIST OF ABBREVIATIONS USED

AV: system availability

AVF: field availability

CA: Acquisition cost

CD(l): Design cost for a design with partition l

CDOC: cost of documentation

CFACIL: cost of facilities

CFAB: cost of fabrication

CINST: cost of installation

cLI = Cost of introducing a line item into the supply system

CM(i): Unit cost of module type i

cov : the probability that a fault is successfully
handled by the fault recovery procedure5

CRD: cost of research, logical/physical design

Crepair: cost of all repairs made during the system's lifetime.

CS: Lifetime Support cost

csc: shelf cost/year of maintaining item in the supply system

Cspares: cost of spares required *over* the system's life-time.

CSUPP: cost of supporting equipment (e.g. tools, test equipment)

CT: Total life cycle cost of a system

D(i): diagnostic resolution of module i $(0<D<1)$

FRb: Failure Rate of a board

FR-B: Failure Rate of boards in a system

FR-C: Failure Rate of connectors in a system

FRc: Failure Rate of a connector

FR-D: Failure Rate of Devices in a system

FRsys: system failure rate

Lj: failure rate for the jth module

MTBF: Mean Time Between Failures

MTTR: Mean Time To Repair

MTTRD: Mean Time To Repair in Depot

MTTRF: Mean Tine To Repair in the Field

NCP: number of connector pins for a given board

NCPB: maximum number of components per board .

NE(i) : the number of units of type i to be procured

NPPC: number of pins per component.

n(l): the number of module types for partition l

NL: planned operational life of the equipment (in years)

Pi: ith module complexity measure (e.g. pin count)

Pj(nj): probabi 1 ity that nj spares of module type j will be
sufficient over the planned life of the equipment,

θj: expected number of failures of module type j during period NL

TMF: field maintenance time

TMD: repair depot maintenance time

TPR: Test Point Ratio

wj: total number of failures of the jth module during perioci NL