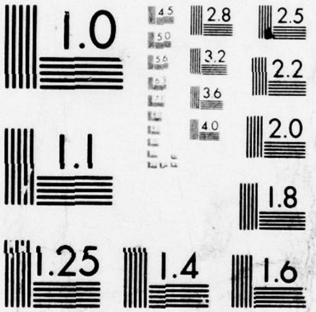
AD-A073 375 ALFRED P SLOAN SCHOOL OF MANAGEMENT CAMBRIDGE MA CEN--ETC F/G 9/2
THE IMS DATA STORAGE HIERARCHY - DSH-1.(U)
AUG 79 C LAM, S E MADNICK N00039-78-G-0160
UNCLASSIFIED CISR-M010-7908-02 NL

1 OF 1

AD
A073375

END
DATE
FILMED
10-79
DDC



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

(12)

MENS ET MANUS

1861

Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

Basic Ordering Agreement No. N00039-78-G-0160

Task No. 003

Internal Report No. M010-7908-02

Technical Report No. 2

The IMS Data Storage Hierarchy - DSH-1

Chat-Yu Lam

Stuart E. Madnick

August 1979

Principal Investigator:

Professor Stuart E. Madnick

Prepared for:

Naval Electronics Systems Command

Washington, D.C.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

9

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER Technical Report ~~No.~~ 2	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 The IMS Data Storage Hierarchy - DSH-1		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER M010-7908-02 ✓
7. AUTHOR(s) 10 Chat-Yu Lam Stuart E. Madnick		8. CONTRACT OR GRANT NUMBER(s) 15 N00039-78-G-0160-001 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS Center for Information Systems Research M.I.T. Sloan School of Management, Cambridge, MA 02139 ✓		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 12 44p.
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE 11 Aug~~ust 1~~979
		13. NUMBER OF PAGES 43
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

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

14 CISR-M010-7908-02,
CISR-TR-2

18. SUPPLEMENTARY NOTES

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

database mangement, database computer, multiple processor system, hierarchical system, storage hierarchy, automatic data repair

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

409 590 JOB

20. Abstract

To achieve high performance, large capacity, and high availability, the IMS data base computer takes the approach of implementing the logical information management functions by means of a pipeline of parallel microprocessors and makes use of a storage hierarchy with distributed control for the physical storage and manipulation of very large databases.

The data storage hierarchy (DSH) of IMS provides a very large byte addressable virtual address space accessible by the large number of processors that implement the logical information management functions of IMS. A highly parallel structure is used by the DSH to support asynchronous processing of a large number of data requests in order to attain very high throughput. Use of multiple block sizes across the storage levels and use of efficient data movement algorithms contribute to the high performance of DSH. The use of distributed control as well as multiple data redundancy gives DSH the capability to tolerate hardware failures without suffering data loss or availability of service. This report describes the design objectives and the structure of a general data storage hierarchy (DSH-1). Research issues in DSH-1 are also discussed.

Preface

The Center for Information Systems Research (CISR) is a research center of the M.I.T. Sloan School of Management. It consists of a group of management information systems specialists including : faculty members, full-time research staff, and student research assistants. The Center's general research thrust is to devise better means for designing, implementing, and maintaining application software, information systems, and decision support systems.

Within the context of the research effort sponsored by the Naval Electronics Systems Command under contract N00039-78-G-0160, CISR has proposed to conduct basic research on the *Intelligent Memory System* (IMS). The IMS is a high performance, high availability information management system for supporting future Command, Communication and Control Systems.

Current advances in LSI and Multi-Chip Integration technology offer the potential for development of modular multi-processor building blocks for information management, as well as for intelligent memory controllers. Advances in information management technologies have made it possible to hierarchically organize the information management functions so as to facilitate pipeline and parallel processing. The IMS attempts to integrate the above hardware and software advances. In the IMS, all the information management functions are decomposed into a functional hierarchy. Each level of the functional hierarchy is implemented using modular multi-processor building blocks. An automatic storage hierarchy is used by the IMS for storage and retrieval of very large databases. Each level of the storage hierarchy is implemented using modular multi-processor controllers and their associate? storage devices.

The proposed research described in Contract N00039-78-G-0160 focuses on the concept development, architectural design and evaluation of the IMS storage hierarchy. Specific research tasks to be accomplished are : (1) design of a general structure of the IMS storage hierarchy, (2) design of a revised structure of the IMS storage hierarchy, (3) develop algorithms for the IMS storage hierarchy, (4) performance evaluation of the IMS storage hierarchy.

Technical Report No. 1 introduces the concepts of IMS and its research directions. This report discusses the concepts of data storage hierarchies from a practical point of view. A design of DSH-1, the data storage hierarchy of the IMS database computer, is described.

ABSTRACT

To achieve high performance, large capacity, and high availability, the IMS data base computer takes the approach of implementing the logical information management functions by means of a pipeline of parallel microprocessors and makes use of a storage hierarchy with distributed control for the physical storage and manipulation of very large databases.

The data storage hierarchy (DSH) of IMS provides a very large byte addressable virtual address space accessible by the large number of processors that implement the logical information management functions of IMS. A highly parallel structure is used by the DSH to support asynchronous processing of a large number of data requests in order to attain very high throughput. Use of multiple block sizes across the storage levels and use of efficient data movement algorithms contribute to the high performance of DSH. The use of distributed control as well as multiple data redundancy gives DSH the capability to tolerate hardware failures without suffering data loss or availability of service. This report describes the design objectives and the structure of a general data storage hierarchy (DSH-1). Research issues in DSH-1 are also discussed.

TABLE OF CONTENTS

ABSTRACT i

Section . . . page

I. INTRODUCTION 1

II. DESIGN OBJECTIVES OF DSH-1 5

- Virtual Address Space 6
- Very Large Address Space 6
- Permanent Data Storage 7
- Support Multiple Processors 8
- Generalized Multi-level Storage System 8
- Direct Inter-level Data Transfer 9
- High performance 9
- Modularity 10
- Modularity 10
- Low Cost 11

III. GENERAL STRUCTURE OF DSH-1 13

- The DSH-1 Interface 15
- The Highest Performance Storage Level - L(1) . . 18
- A Typical Storage Level - L(i) 20
- Read and Write Operations 22

IV. RESEARCH ISSUES 28

- Support of Read and Write Operations 29
  - Multiple Cache Consistency 29
  - Bus Communication Protocols 30
  - Multiple Directory Update 31
  - Multiple Resource Arbitration 31
  - Transaction Handling 31
- Multiple Data Redundancy Properties 32
- Automatic Data Repair Algorithms 32
- Performance Evaluation 33

V. SUMMARY 34

REFERENCES 35

Section I

INTRODUCTION

The need for efficient storage and processing of very large data bases coupled with advances in Very Large Scale Integration (VLSI) technology have motivated various proposals to develop specialized computers dedicated to database processing. Four basic approaches to develop specialized data base processors are discussed in (Lam and Madnick, 1979a; Lam and Madnick, 1979c). These approaches are : (1) new instructions through microprogramming, (2) intelligent controllers, (3) dedicated conventional computers for data base processing, and (4) specialized data base computers.

IMS is a specialized data base computer. One of its objectives is to provide significant performance improvement over conventional database processors. For example, it is aimed at supporting transaction rates up to 10,000 transactions per second. Another objective of IMS is to support very large databases in a cost effective fashion. For example, it is designed to support data bases in excess of a trillion bytes of online data. A third objective of IMS is to provide a highly available system even in the event of individual component failures. IMS is designed from the start to be able to maintain continuous operation without data loss even in the event of such failures.

To meet these objectives, IMS makes use of the techniques of hierarchical decomposition to structure the logical information management functions as a hierarchy of modules. Each level of this functional hierarchy is implemented by multiple microprocessors. Thus concurrent processing of transactions is obtained through pipeline and parallel operations of the functional hierarchy. The technique of hierarchical decomposition is also applied to organize the storage subsystem to obtain a modular storage hierarchy capable of supporting the storage requirements of the functional hierarchy. The conceptual organization of IMS is illustrated in Figure I.1.

Research activities on the decomposition of the information management functions into hierarchical levels and the development of a modular storage hierarchy are currently being carried out. On the functional decomposition, a technique is being developed to obtain an optimal functional decomposition. This particular approach is called Systematic Design Methodology (Huff and Madnick, 1978).

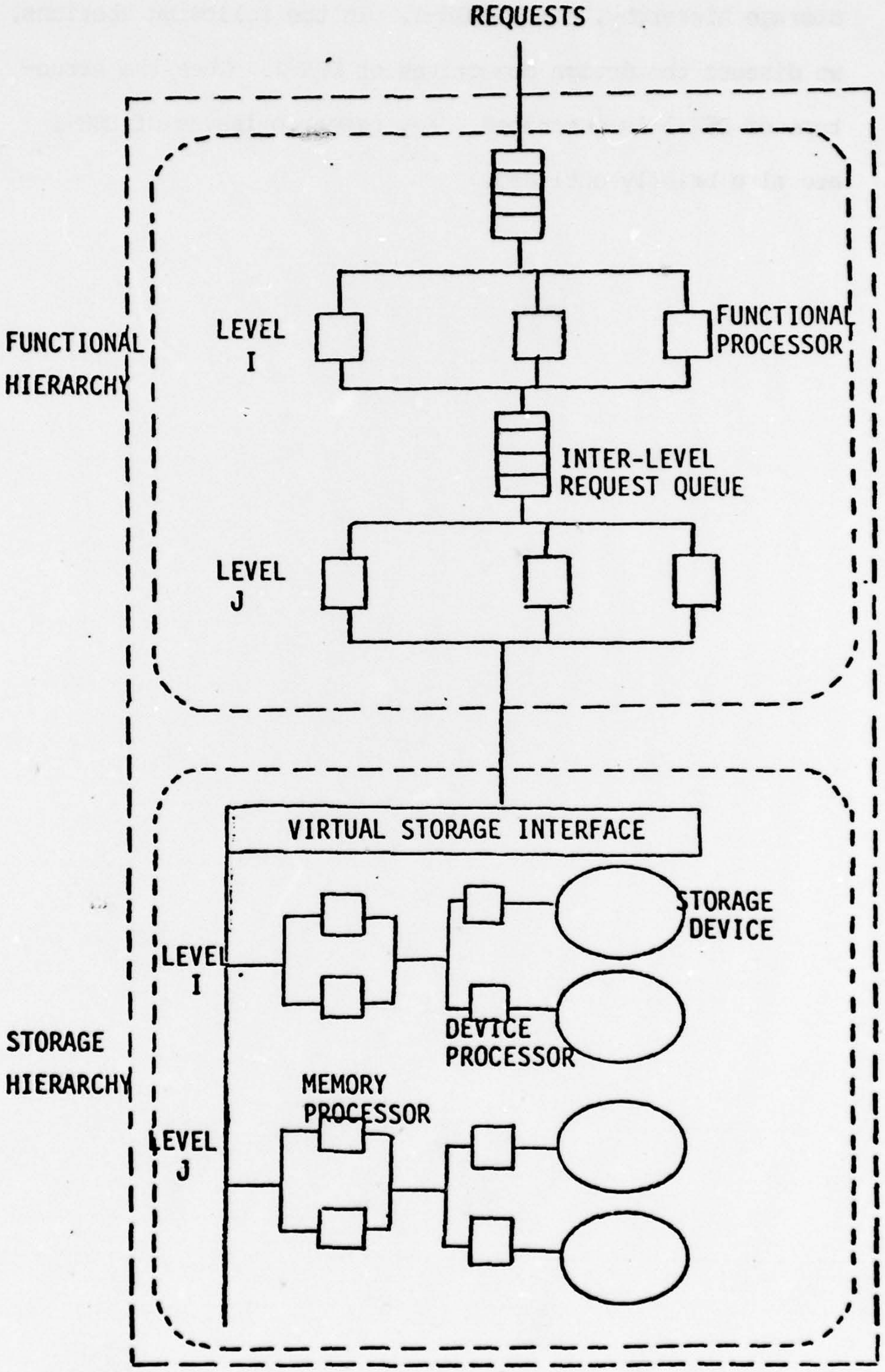


Figure I.1 The INFOPLEX conceptual structure

This report describes a general structure of the IMS data storage hierarchy, called DSH-1. In the following sections, we discuss the design objectives of DSH-1. Then the structure of DSH-1 is described. Key research issues of DSH-1 are also briefly outlined.

Section II

DESIGN OBJECTIVES OF DSH-1

In reviewing the currently available and anticipated storage device technologies we notice that economically viable low cost, high capacity storage devices have high access time while high performance storage devices have high cost. Storage hierarchy systems have been developed to take advantage of locality of references (Denning, 1970) to obtain high performance storage systems with high capacity and low overall cost by using a range of storage devices with different cost/performance characteristics arranged in a hierarchy. The effective capacity and cost of the storage hierarchy are close to those of the slower storage devices. As long as there is a high probability of accessing the high performance storage devices, the effective access time to the storage hierarchy will be close to that of the high performance devices.

There are a large number of practical storage hierarchy systems today. However, the functionality provided by each is quite different and often falls short of our expectations (for use as the storage subsystem of the IMS data base computer). In the following, we discuss the underlying design goals of DSH-1. The structure of DSH-1 will be described in a following section.

2.1 VIRTUAL ADDRESS SPACE

DSH-1 provides a virtual address space for data storage. Every data item in DSH-1 is byte addressable using a generalized virtual address. A key advantage of a virtual address space is that an user (a processor) of DSH-1 is relieved of all physical device concerns. In fact, the processor accessing DSH-1 is not aware of how the virtual address space is implemented. This latter characteristics is quite unique since most current virtual memory systems are simulated, at least partially, by software executed by the processor, e.g., the IBM OS/VS system (Scherr, 1973).

2.2 VERY LARGE ADDRESS SPACE

Early virtual memory systems were developed primarily for program storage, hence their address spaces were quite limited, e.g., in the order of one million bytes. The MULTICS (Greenberg and Webber, 1975) virtual memory and the IBM System/38 (Datamation, 1978a; Soltis and Hoffman, 1979) logical storage were developed for program as well as data file storage. These systems support a large virtual address space. However, the size of an individual data file in MULTICS is limited to 2**18 bytes and that in System/38 is limited to 2**24 bytes. Though these are very large address spaces, it is expected that future systems will require online storage capacities that are much larger. DSH-1 uses a 64-bit virtual address. Each byte is directly addressable, hence there is virtually no limit on the size of a logical entity such as a data file.

2.3 PERMANENT DATA STORAGE

Accesses to permanent data is performed by special software routines and a special I/O processor in most virtual memory systems (e.g., IBM's OS/VS). The I/O processor brings the data into the virtual memory and writes the data back to permanent storage when the data is updated. Systems like MULTICS and System/38 provide a permanent virtual data storage. Any data in virtual memory is also in permanent storage. DSH-1 also provides a permanent virtual data storage. Special data integrity schemes are used to ensure that as soon as a processor completes a write operation to a virtual location, the effect of the write becomes permanent even in the event of a power failure.

2.4 SUPPORT MULTIPLE PROCESSORS

Most current virtual memory systems have been limited to supporting 2 or 3 processors. It is necessary that DSH-1 support a large number of processors due to the requirements for high performance and high availability to be discussed below. All these processors share the same virtual data address space. Appropriate synchronization and protection schemes are used to ensure data integrity and security.

2.5 GENERALIZED MULTI-LEVEL STORAGE SYSTEM

To provide a large capacity storage subsystem with low cost and high performance, a spectrum of storage devices arranged in a hierarchy is used. Previous storage hierarchy systems have been specially designed for a specific 2 or 3 levels hierarchy (e.g., cache and main memory, or main memory and secondary storage device). In these cases, it is extremely difficult to add or remove a storage level. DSH-1 is designed to incorporate any type of storage device and support reconfiguration of storage levels. This characteristic is particularly important in responding to new device technologies.

2.6 DIRECT INTER-LEVEL DATA TRANSFER

In most current storage hierarchy systems, data movement among storage levels is performed indirectly. For example, to move data from drum to disk in the MULTICS system, data is read from drum into main memory by the processor which then writes the data from main memory to disk. Recent developments in storage systems make it possible to decentralize the control of data movement between storage devices and incorporate it into intelligent controllers for the storage devices. For example, the IBM 3850 Mass Storage (Johnson, 1975) uses an intelligent controller to handle data transfer between mass storage and disks, making the 3850 appear as a very large number of virtual disks. DSH-1 incorporates intelligent controllers at each storage level to implement the algorithms for data movement among the storage levels. Special algorithms are developed to facilitate efficient broadcasting of data from a storage level to all other storage levels as well as movement of data between adjacent storage levels.

2.7 HIGH PERFORMANCE

To support the data requirements of the functional processors in IMS, DSH-1 is designed to handle a large number of requests simultaneously. The operation of DSH-1 is highly parallel and asychronous. Thus, many requests may be in different stages of completion at various storage levels of DSH-1. Each processor accesses DSH-1 through a data cache where the most frequently used data items are stored.

2.8 MODULARITY

High availability of DSH-1 is a result of a combination of the design strategy used, hardware commonality, and special algorithms. Key design strategies in DSH-1 include the use of distributed controls and simple bus structures, both of which contribute to the high availability of DSH-1. Multiple identical hardware components are used in parallel to provide high performance and to ensure that no single component is critical to system operation. Integrated into the design are certain algorithms that exploit the structure of DSH-1 to allow data redundancy and perform automatic data repair in the event of component failure, thus diminishing the dangers of multiple failures. These mechanisms are further discussed after the structure of DSH-1 is described.

2.9 MODULARITY

DSH-1 is modular at several levels. This provides considerable flexibility in system structuring. The number of processors to be supported by DSH-1 can be varied. The number of storage levels and the type of storage devices can be chosen to meet the particular capacity and performance requirements. All the storage levels have very similar structures and the same algorithm is used by the intelligent controllers at each storage level.

Flexibility in system structuring is extended in DSH-1 to allow for dynamic system reconfiguration. For example, a defective storage device or storage level can be amputated without loss of system availability.

An example of a system that incorporates modularity as a key design goal is the PLURIBUS (Katsuki et. al., 1978) system. In PLURIBUS, the basic building block is a bus module. The number of components on a bus module as well as the number of bus modules can be easily varied to meet different system requirements.

2.10 LOW COST

A storage hierarchy is the lowest cost configuration to meet the requirement of providing a large storage capacity with high performance. DSH-1 also makes use of common hardware modules as the intelligent controllers at each storage level, thus reducing hardware development cost. The modularity features of DSH-1 discussed above also facilitate system upgrading with minimum cost.

Commonality of hardware modules and flexibility of system upgrade have been employed in many computer systems as an effective approach to reduce cost. However, these techniques are rarelv applied to storage hierarchy systems. DSH-1 is a step in this direction.

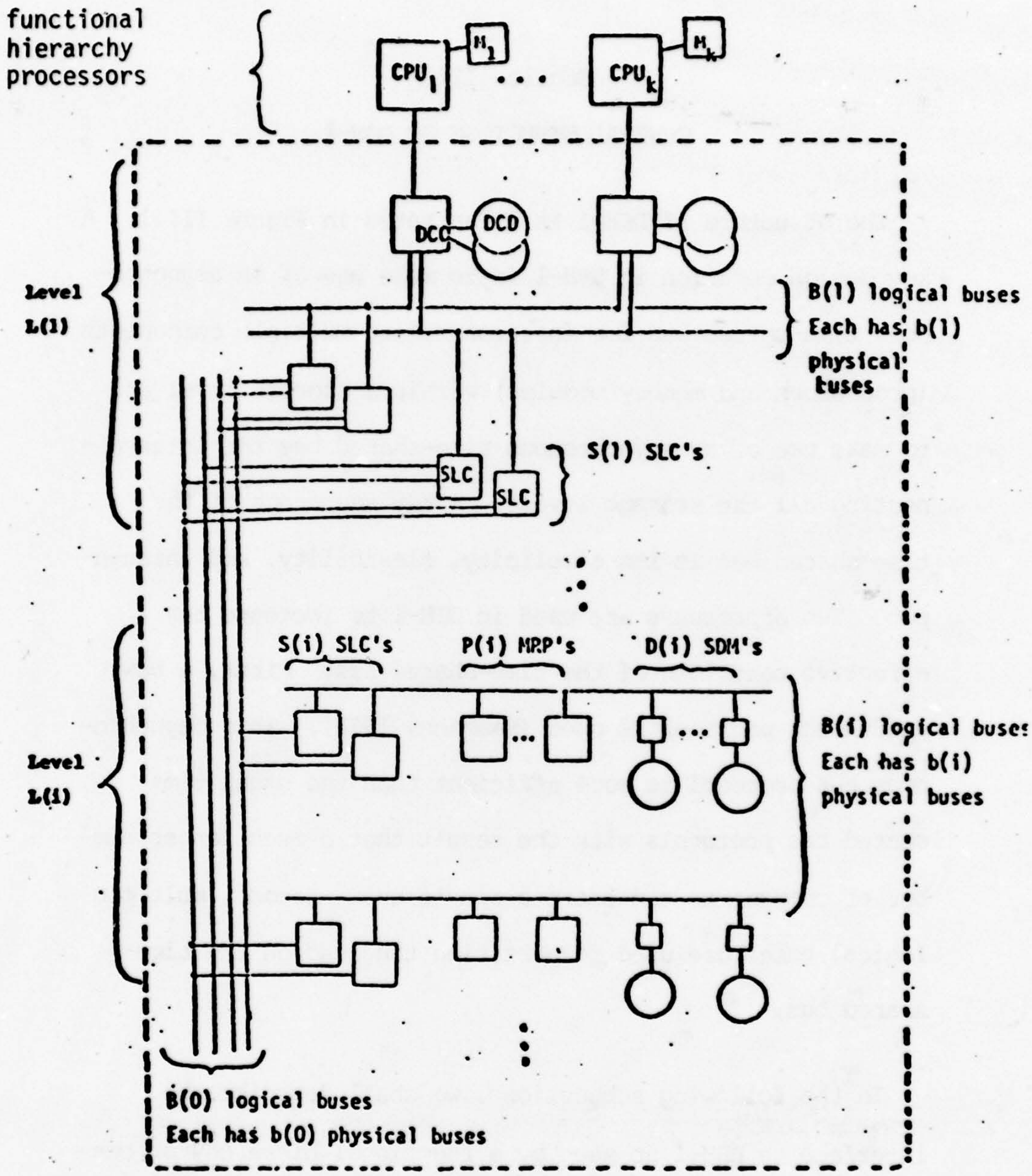
Advances in storage device and processor technologies provide great potentials for development of very effective data storage hierarchies that incorporate the above characteristics. In the next section, we describe a general structure of such a system as a basis for further research into this important subject.

Section III

GENERAL STRUCTURE OF DSH-1

The structure of DSH-1 is illustrated in Figure III.1. A key design decision in DSH-1 is to make use of an asynchronous time-shared bus for interconnecting multiple components (processors and memory modules) within a storage level and to make use of an asynchronous time-shared bus for interconnecting all the storage levels. A key advantage of the time-shared bus is its simplicity, flexibility, and throughput. Two approaches are used in DSH-1 to increase the effective bandwidth of the time-shared bus. First, a new pended-bus protocol is used (Haagens, 1978). This asynchronous bus protocol is more efficient than the usual time-shared bus protocols with the result that a much larger number of components can share a single bus. Second, multiple logical buses are used to partition the load on the time-shared bus.

In the following subsections, we shall describe the interface to DSH-1 as seen by a functional hierarchy processor (see Figure I.1). Then the structure of DSH-1 is described by examining its highest performance storage level and then a typical storage level. Finally, the approaches


functional hierarchy processors
CPU_1
M_1
...
CPU_k
M_k
DCC
DCD
Level L(1)
B(1) logical buses Each has b(1) physical buses
SLC
SLC
S(1) SLC's
S(i) SLC's
P(i) MRP's
D(i) SDM's
...
...
B(i) logical buses Each has b(i) physical buses
Level L(i)
B(0) logical buses
Each has b(0) physical buses

System Structure

component	abbreviation	quantity
data cache controller	DCC	(k one per memory port)
data cache duplex	DCD	(2 per DCC)
storage level controller	SLC	(S(i))
memory request processor	MRP	(P(i))
storage device module	SDM	(D(i))

Figure III.1 DSH-1 System Structure

that DSH-1 utilizes in handling read and write operations are then described.

3.1 THE DSH-1 INTERFACE

To the functional hierarchy processors connected to DSH-1, DSH-1 appears as a large multi-port main memory. There are K memory ports, hence K processors can simultaneously access DSH-1.

The functional processors use a 2**V (V=64) byte virtual address space. The instructions for each functional hierarchy processor are stored in a separate 2**I byte program memory. The program memories are not part of DSH-1. Thus, 2**I bytes of the processor's address space is mapped by the program memories, leaving 2**V-2**I bytes of data memory to be managed by DSH-1. This is depicted in Figures III.2(a) and III.2(b).

Each processor has multiple register sets to support efficient multiprogramming. Some of the more important registers for interfacing with DSH-1 are : (1) a V-bit Memory Address Register (MAR) for holding the virtual address, (2) a Memory Buffer Register (MBR) for storing the data read from DSH-1 and to be written into DSH-1, (3) a Memory Operation Register (MOR) indicates the particular operation to be performed by DSH-1, (4) an Operation Status

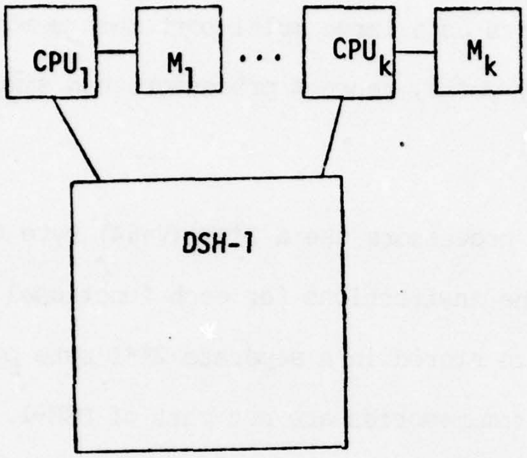


Figure III.2(a) DSH-1 Interface

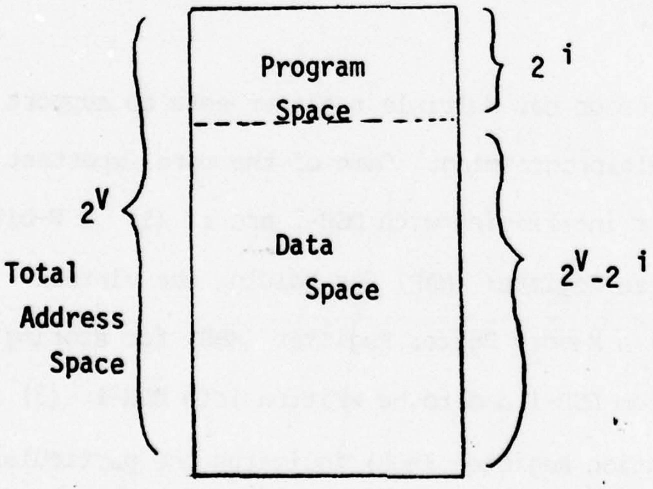


Figure III.2(b) DSH-1 Address Space

Register (OSR) which indicates the result of a operation performed by DSH-1, and (5) a Process Identifier Register (PIR) which contains the Process Identifier (PID) of the process that is currently using the processor.

A number of memory operations are possible. The key ones are the read and write operations and the primitives for locking a data item (such as those supporting the Test-and-Set type of operations).

All read and write operations to DSH-1 are performed in the highest performance storage level, L(1). If a referenced data item is not in L(1), it is brought up to L(1) from a lower storage level via a read-through (Madnick, 1979) operation. The effect of an update to a data item in L(1) is propagated down to the lower storage levels via a number of store-behind (Madnick, 1979) operations. The read-through and store-behind operations are further discussed in a later section. Here we focus on the asychronous operation of the processor-DSH interface.

In a read operation, two results can occur depending on the state of DSH-1. First, if the requested data is already in L(1), the MBR is filled with the data bytes starting at location (MAR) and the processor continues with the next operation. Alternatively, the addressed data may not be available in L(1). In this case, the processor is interrupted, the OSR is set to indicate that it may take a while for the read operation to complete, and the processor is switched to another process. Eventually, the addressed data is retrieved from a lower level and copied into L(1) from a lower storage level. When this is completed, the processor is notified of the completion of the original read operation.

Similarly, a write operation may result in two possible responses from DSH-1. First, if the data to be updated is already in L(1), the bytes in MBR are written to the virtual address locations starting at (MAR), and the processor continues with the next operation. Second, a delay similar to the read operation may occur (when the data to be updated is not in L(1)), while DSH-1 retrieves the data from a lower storage level.

This concludes a brief description of the asynchronous DSH-1 interface, as seen by a functional hierarchy processor. Next, we examine the detailed operation of DSH-1.

3.2 THE HIGHEST PERFORMANCE STORAGE LEVEL - L(1)

There are h storage levels in DSH-1, labelled L(1), L(2), L(3), ..., L(h). L(1) is the highest performance storage level. L(i) denotes a typical storage level. The structure of all storage levels are the same, except for level L(1) which has a unique structure.

A distinction must be made between the concept of a physical bus and a logical bus. The former refers to the actual hardware that implements communications among levels and within a level. A logical bus may be implemented using one or more physical buses. Logical buses represent a partitioning, based upon the virtual address referenced, of the physical buses.

Referring to Figure III.1, L(1) consists of K memory ports and S(1) storage level controllers (SLC's) on each of B(1) logical local buses (i.e., S(1)*B(1) SLC's in total for this level). Each memory port consists of a data cache controller (DCC) and a data cache duplex (DCD). A DCC interfaces with the functional hierarchy processor that is connected to the memory port. A DCC also performs mapping of a virtual address generated by the processor to a physical address in the DCD. Another function of DCC is to interface with other DCC's (e.g., to maintain data cache consistency), and with SLC's on the logical bus (for communications with other storage levels).

At L(1), a SLC accepts requests to lower storage levels from the DCC's and forwards them to a SLC at the next lower storage level. When the responses to these requests are ready, the SLC accepts them and sends them back to the appropriate DCC's. The SLC's also couple the local buses to the global buses. In essence, the SLC serves as a gateway between levels and they contend among themselves for use of the communication media, the logical buses.

At L(1), there are B(1) logical local buses. Each logical local bus consists of b(1) physical buses. Each logical bus handles a partition of the addresses. For example, if two logical buses were used, one might handle all odd numbered data blocks and the other would handle all the even numbered data blocks.

DSH-1 has B(Ø) logical global buses. Each logical global bus consists of b(Ø) global physical buses. The use of address partitioning increases the effective bus bandwidth. The use of multiple physical buses for each logical bus enhances reliability and performance.

3.3 A TYPICAL STORAGE LEVEL - L(I)

A typical storage level, L(i), is divided into B(i) address partitions. Each address partition consists of S(i) SLC's, P(i) memory request processors (MRP's), and D(i) storage device modules (SDM's), all sharing a logical bus. A logical bus consists of b(i) physical buses.

An SLC is the communication gateway between the MRP's/SDM's of its level and the other storage levels.

An MRP performs the address mapping function. It contains a directory of all the data maintained in the address partition. Using this directory, an MRP can quickly determine if a virtual address corresponds to any data in the address partition, and if so, what the real address is for the data. This real address can be used by the corresponding SDM to retrieve the data. Since each MRP contains a copy of this directory, updates to the directory have to be handled with care, so that all the MRP's see a consistent copy of the directory.

An SDM performs the actual reading and writing of data. It also communicates with the MRP's and the SLC's.

The SLC's, MRP's, and SDM's cooperate to handle a memory request. An SLC communicates with other storage levels and passes requests to an MRP to perform the address translation. The appropriate SDM is then initiated to read or write the data. The response is then sent to another SLC at another storage level.

3.4 READ AND WRITE OPERATIONS

As discussed earlier, a processor at the DSH-1 interface that has issued a read or write request may find the request delayed if the data is not ir the cache and it is necessary to retrieve it from a lower storage level.

DSH-1 uses the read-through strategy for reading data from lower storage levels. Figure III.3 illustrates the read-through operation for a storage hierarchy with three storage levels, L(1), L(2), and L(3). Suppose the block size of L(1) is b, the block size of L(2) is 2b, and the block size of L(3) is 4b. Consider a request to read a data item 'a'. The data item 'a' is assumed to be within a L(1) block boundary (if a data item crosses block boundaries, multiple read requests will be used). The request is propagated down the storage levels until the data item 'a' is found in a storage level. Suppose 'a' is found in L(3). A block of size 2b containing 'a' is broadcast to L(2) and L(1) simultaneously. L(2) will accept the entire block and place it in a storage device. L(1) will only accept a block of size b that contains 'a' and place it in its storage device. The data block containing item 'a' is simultaneously returned to the functional hierarchy processor.

One reason for using the read-through strategy is, that as a result of the read-through operation, a small locality of the addressed data item is found in the highest

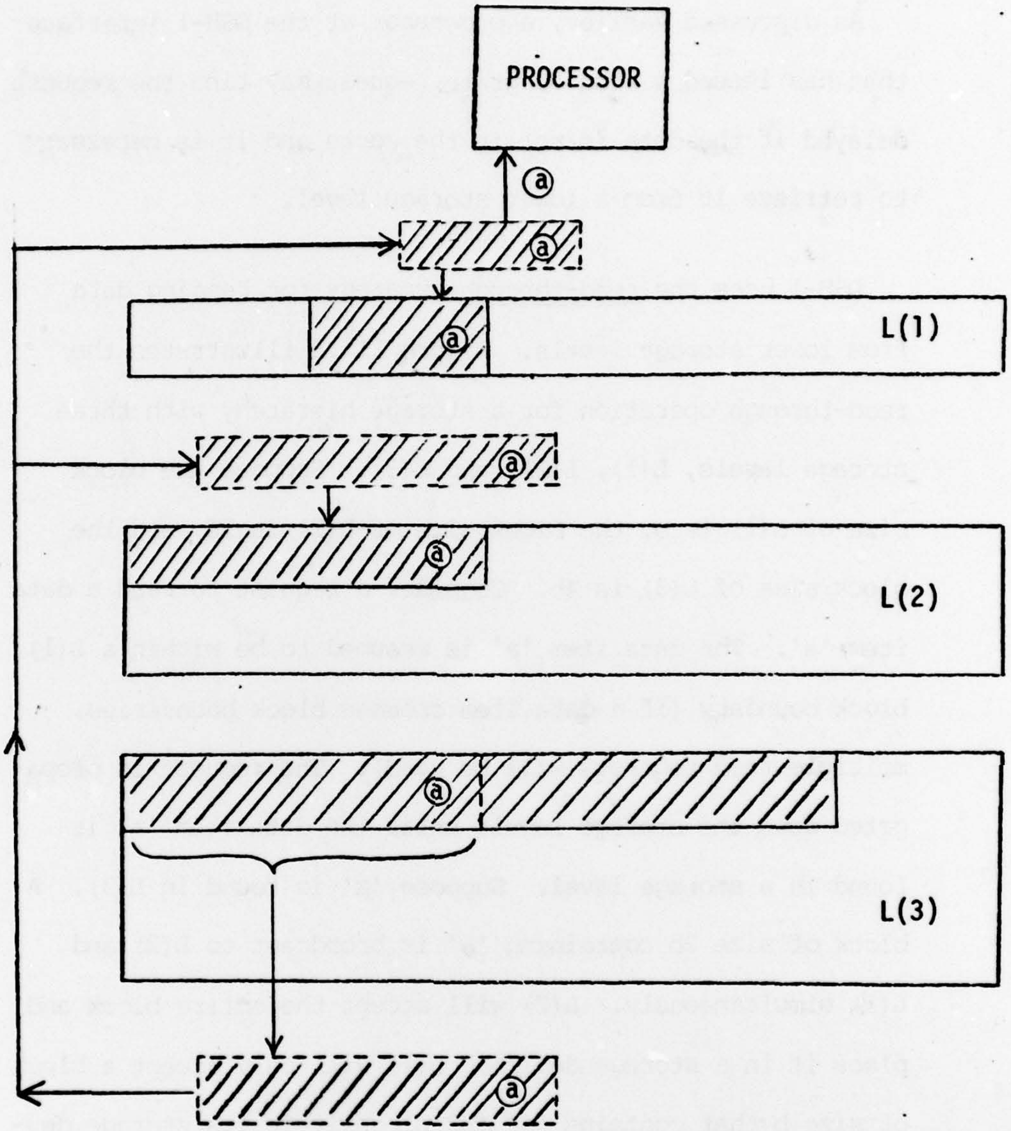


Figure III.3 Read-Through Operation

performance storage level, and a larger locality of the addressed data item is found in the next storage level, and so on. Thus, subsequent references to the locality can be handled efficiently. Another reason for using the read-through strategy is that, as a result of the read-through operation, copies of the addressed data item are found in several different storage levels, thus providing multiple data redundancy for the data item to protect against any storage level failure.

DSH-1 uses the two-level-store-behind strategy for writing data. This strategy is also motivated by performance and reliability. Using this strategy, a write operation does not require an immediate store-through operation, which will reduce performance. At least two copies of the written data are always maintained at all times using this strategy, thus reliability is enhanced.

In a write operation, we shall assume that the data item to be written is already in L(1) (This can be realized by reading the data item into L(1) before the write operation). The data item is written to the data cache duplex, and the processor is notified of the completion of the write operation. The block in L(1) that has just been updated is marked with a count of 2. A store-behind operation is next generated by the data cache controller and sent to the next lower storage level. This is illustrated in Figure III.4(a).

When a store-behind operation is received in L(2), the addressed data is written, and marked with a count of 2. An acknowledgement is sent to the next upper storage level, L(1), and a store-behind operation is sent to the next lower storage level, L(3). When an acknowledgement is received at L(1), the counter for the addressed data item is decremented by 1 resulting in a count of 1. This is illustrated in Figure III.4(b).

The store-behind is handled in L(3) by updating the appropriate data block. An acknowledgement is sent to L(2). At L(2), the corresponding block counter is decremented by 1 resulting in a count of 1. The acknowledgement is also forwarded to L(1). At L(1), the corresponding block counter is decremented by 1 which now becomes Ø, hence the block is eligible for replacement. This is illustrated in Figure III.4(c).

Thus we see that the two-level store-behind strategy maintains at least two copies of the written data at all times. Furthermore, lower storage levels are updated at slack periods of system operation, thus enhancing performance.

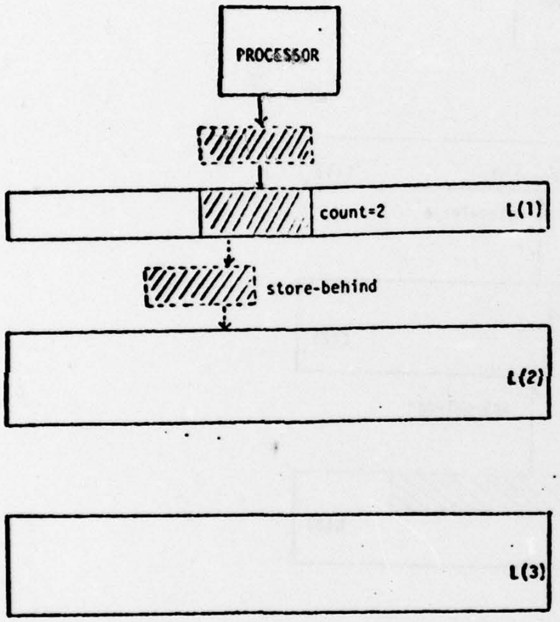


Figure III.4(a) store-behind operation

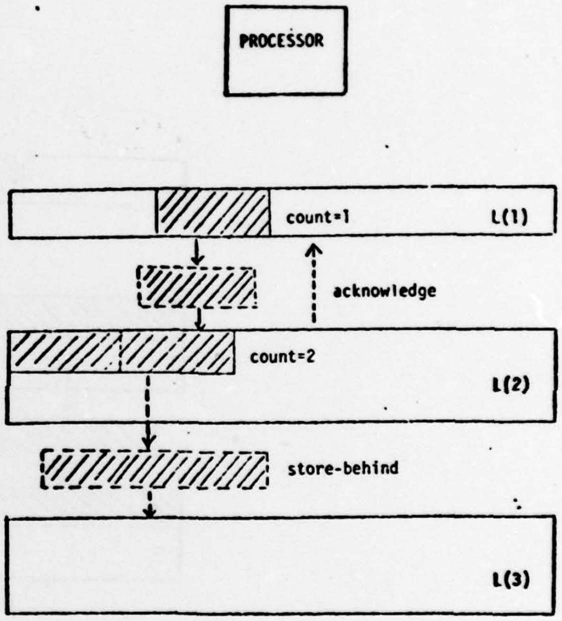


Figure III.4(b, store-behind operation

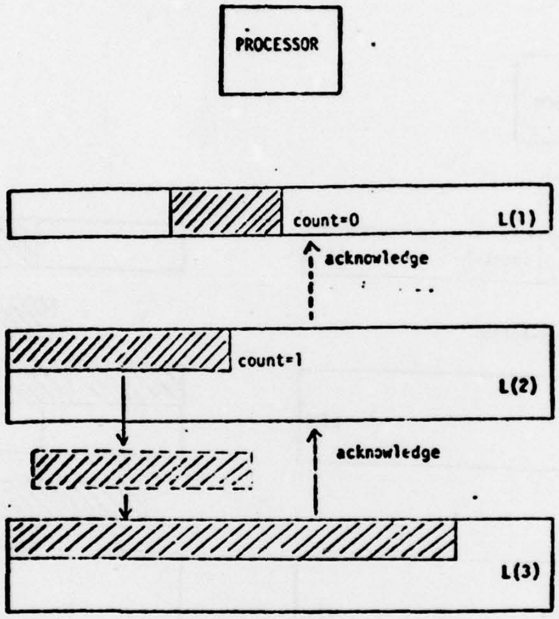


Figure III.4(c) store-behird operation

Section IV

RESEARCH ISSUES

The previous section describes the general structure of DSH-1 and the strategies for reading and writing data. From this general structure, a number of interesting alternative configurations can be obtained. For example, if all the data caches are taken away, L(1) becomes a level with only the SLC's for communicating the requests from the processors to the lower storage levels and for obtaining responses from these lower storage levels. This configuration eliminates the data consistency problems associated with multiple data caches.

If we let the number of logical buses be equal to one, we obtain the configuration without address partitioning.

Another intersting configuration is when there is only one MRP and one SDM on a given logical bus. This configuration eliminates the need for multiple identical directory updates.

Thus, by varying the design parameters of DSH-1, a large number of alternative configurations with quite different characteristics can be obtained. The general structure is a valuable vehicle for investigating various design issues. Some of the key issues that are being investigated are briefly introduced in the following sectionꞅ.

4.1 SUPPORT OF READ AND WRITE OPEIATIONS

Key problems in supporting the read and write operations in DSH-1 include : (1) data consistency in multiple data caches, (2) protocols for communicating over the shared bus, (3) algorithms for updating the redundant directories, (4) algorithms for arbitrating among usage of identical resources, such as buses, SLC's and MRP's, and (5) specifying the various steps (transactions) that have to be accomplished to handle the read and write operations.

4.1.1 Multiple Cache Consistency

As illustrated in Figure III.1, each DSH-1 memory port is a data cache directly addressable by the processor at the port. It is possible then, that a data item may be in several different data caches at the same time. When the data item gets updated by a processor, other processors may reference an inconsistent copy of the data item. The multiple cache consistency problem and possible solutions are discussed in (Tang, 1976; Censier and Feautrier, 1978).

Three basic approaches are being explored in resolving this problem in DSH-1. The first approach is to send a purge request to all other data caches whenever a processor updates data in its cache. The second approach is to maintain status information about the data cache contents. Whenever there is an update to a data item, this status information is consulted and purge requests are sent only to those caches that contain the data item being changed. The third approach is to make use of knowledge of how the data in DSH-1 is to be used so that the inconsistency problem can be avoided. For example, knowledge about the interlocking scheme used to ensure safe data sharing may be used to avoid unecessary purge requests to other caches.

4.1.2 Bus Communication Protocols

In DSH-1, the buses may be used for point-to-point communication as well as for broadcast type of communications. It is necessary to ensure that messages are sent and received correctly. For example, L(i) broadcast data to the upper levels and one or more of these levels may not be able to accomodate the data to be received, possibly due to the lack of buffer space. Communications protocols to handle these situations are important.

4.1.3 Multiple Directory Update

Each MRP contains a directory of all the data in the SDM's on the same bus. Multiple requests may be handled by the MRP's. When a MRP updates its directory, other MRP's may still reference the old copy of the directory. This is similar but not identical to the multiple cache consistency problem discussed above. It is necessary to maintain consistency of the MRP directory states.

4.1.4 Multiple Resource Arbitration

Multiple identical resources (e.g., buses, MRP's, and SLC's) are used in DSH-1 to provide parallel processing while at the same time providing redundancy against failure. A request for a resource can be satisfied by any one of the resources. An arbitration scheme is required to control the assignment of resource.

4.1.5 Transaction Handling

A read or a write request may go through a number of asynchronous steps through a number of storage levels to completion. A complication to these transactions is that for high throughput, a request (or response) may be divided into a number of messages when the request (or response) is being transported within the hierarchy. Thus, a request (or response) may have to be assembled, which may take an amount of time dependent on the traffic within DSH-1. Partial requests (responses) at a storage level require special handling.

4.2 MULTIPLE DATA REDUNDANCY PROPERTIES

As a result of the read-through operation, several copies of a referenced data item exists in the DSH-1 storage levels. The two-level store-behind operation also maintains at least two copies of any updated data item in DSH-1. In (Lam and Madnick, 1979b), several important properties associated with multiple data redundancy of storage hierarchies are analyzed in some detail. It is found that with proper choices of the sizes of the storage levels, and using certain classes of read-through and replacement algorithms, it is possible to assert that a storage level contains all the data items in the upper storage levels. We would like these properties to be preserved in DSH-1.

4.3 AUTOMATIC DATA REPAIR ALGORITHMS

One of the benefits of maintaining redundant data in DSH-1 is that lost data due to component failures can be reconstructed on a spare component from a copy of the lost data. By using automatic data repair in DSH-1 the probability of multiple data loss can be reduced.

Two classes of automatic data repair algorithms are possible. One strategy is to make use of the multiple data redundancy properties of DSH-1 and to reconstruct the lost data from its copy in a different storage level. The othet approach is to maintain duplicate copies of the data item within a storage level and to reconstruct the lost data from its copy in the same storage level. The latter approach is particularly attractive for low performance devices such as mass storage.

4.4 PERFORMANCE EVALUATION

A key issue in the DSH-1 design is predicting its performance. In order to accomplish this, a simplified design of DSH-1 and its algorithms is being developed. A simulation model can then be developed for this design. Various basic performance statistics can then be obtained under various load assumptions. This experiment will provide insights and directions for further design and performance evaluation efforts.

Section V

SUMMARY

The IMS storage hierarchy is a high performance high availability virtual memory data storage hierarchy with distributed controls for data movement and address translation. It is designed specifically to provide a very large permanent virtual address space to support multiple functional hierarchy processors.

A general structure of DSH-1, the IMS storage hierarchy has been described in this paper. This general structure can be used to derive a large number of alternative configurations which can be used to explore various algorithms for data storage hierarchy systems.

A number of important research issues associated with DSH-1 are outlined. These issues are currently under further investigation.

REFERENCES

(Censier and Feautrier, 1978) : Censier, L.M., and Feautrier, P., 'A New Solution to Coherence Problems in Multicache Systems', IEEE Transactions on Computers, Vol. C-27, No. 12(December, 1978), 1112-1118.

(Considine and Myers, 1977) : Considine. J.P., and Myers, J.J., 'MARC: MVS Archival Storage and Recovery Program', IBMSJ, 1977, 378-397.

(Datamation, 1978a) : Datamation, December, 1978, 230.

(Datamation, 1978b) : Datamation, June, 1978, 254.

(Denning, 1970) : Denning, P.J. 'Virtual Memory'. ACM Computing Surveys, 2, 3 (September 1970), 153-190.

(Gentile and Lucas, 1971) : Gentile, R.B., and Lucas, J.R. Jr., 'The TABLON Mass Storage Network', SJCC, 1971, 345-356.

(Gravina, 1978) : Gravina, C.M., 'National Westminster Bank Mass Storage Archive', IBMSJ, Vol. 17, No. 4, 1978, 344-358.

(Greenberg and Webber, 1975) : Greenberg, B.S., and Webber, S.H. 'MULTICS Multilevel Paging Hierarchy'. IEEE INTERCON, 1975.

(Johnson, 1975) : Johnson, C. 'IBM 3850 - Mass Storage System'. AFIPS Conference Proceedings, 44, (1975), 509-514.

(Haagens,1978) : Haagens, R.B., 'A Bus Structure for Multi-Microprocessing', MS Thesis, MIT Department of Electrical Engineering and Computer Sciences, January, 1978.

(Huff and Madnick, 1978) : Huff, S.L., and Madnick, S.E., 'An Approach to Constructing Functional Requirement Statements for System Architectural Design', M.I.T. Sloan School of Management, CISR Internal Report No. P010-7806-06, (June 1978).

(Katsuki et. al.,1978) : Katsuki, D., Elsam, E.S., Mann, W.F., Roberts, E.S., Robinson, J.G., Skowronski, F.S., and Wolf, E.W., 'Pluribus - An Operational Fault-Tolerant Multiprocessor', Proceedings of the IEEE, Vol. 66, No. 10, (October 1978), 1146-1159.

(Lam and Madnick, 1979a) : Lam, C.Y., and Madnick, S.E., 'INFOPLEX Data Base Computer Architecture - Concepts and Directions', MIT Sloan School Working Paper No. 1046-79 (also as CISR Working Paper No. 41), 1979.

(Lam and Madnick, 1979b) : Lam, C.Y., and Madnick, S.E., 'Properties of Storage Hierarchy Systems with Multiple Page Sizes and Redundant Data', MIT Sloan School Working Paper No. 1047-79 (also as CISR Working Paper No. 42), 1979.

(Lam and Madnick, 1979c) : Lam, C.Y., and Madnick, S.E., 'The Intelligent Memory System Architecture - Research Directions', MIT Sloan School of Management Internal Working Paper No. M010-7908-01, August 1979.

(Liptay, 1968) : Liptay, J.S., 'Structural Aspects of the System 360/85 II. The Cache', IBMSJ, Vol. 7, 1968, 15.

(Madnick, 1979) : Madnick, S.E., 'The INFOPLEX Database Computer: Concepts and Directions', IEEE COMPCON, February 26, 1979, 168-176.

(Scherr, 1973) : Scherr, A.L., 'Functional Structure of IBM Virtual Storage Operating Systems Part II : OS/VS2-2 Concepts and Philosophies', IBMSJ, 12, 4 (1973), 382-400.

(Soltis and Hoffman, 1979) : Soltis, F.G., and Hoffman, R.L., 'Design Considerations for the IBM System/38', IEEE Spring COMPCON, 1979, 132-137.

(Tang, 1976) : Tang, C. K., 'Cache System Design in the Tightly Coupled Multiprocessing System', NCC, 1976, 749-753.