WSI MEMORY SYSTEM: ADDRESS-SELECTION APPROACH

By

GANG HWA LEE

Bachelor of Science

Seoul National University

Seoul, Korea

1981

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1992

Thesis
1992
L478w

WSI MEMORY SYSTEM: ADDRESS-SELECTION APPROACH

Thesis Approved:

Jong J. Lee
Thesis Adviser

Louis G. Johnson

Richard L. Cummins

Thomas C. Collins
Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to give my sincere appreciation to Dr. Jong J. Lee for his continuous advice and guidance as a thesis adviser throughout my graduate program. Also many thanks go to Dr. Richard L. Cummins and Dr. Louis G. Johnson for advising me as committee members. Without their help, this study would not have been completed well.

I give my sincere love to my wife, MyoungJin, for her continuous encouragement and support. I also give my love to our most precious son, HanJae, for his endurance during my studies.

Great appreciation should go to our parents for their continuous support and never-ending love. Many thanks also go to our beloved brother and sister and nephews.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1

Objective 2

II. WSI MEMORY SYSTEM 4

WSI Problems 4
Yield Problem 4
Power-dissipation 5
Worn-out Failure 5
WSI Memory Configuration 6
Path-growth Method 6
Address-selection Method 8

III. WSI MEMORY ORGANIZATION 12

Virtual Memory System 12
WSI Memory System Analysis 14
Line Delay 14
WSI Memory Model 17
Interleaved Memory Structure 19

IV. WSI MEMORY DESIGN 24

Module Design 24
Module Size Decision 24
Address Identification Method 24
Timing Diagram of WSI Memory 28
Refresh Hand-shaking 28
Wafer Design 33
Wafer I/O Selector 33
Design Factors 33
Bus Redundancy 35

Chapter Page

Controller Design 37
Address generator 37
Controller Buffer 37
Address Translation Mapping 39
Multi-wafer Configuration 39

V. SIMPLE WSI DESIGN 42

Central Refresh Control 42

VI. CONCLUSIONS 46

REFERENCES 47

APPENDIXES 50

APPENDIX A – THE EFFECT OF PAGE SIZE UPON AVERAGE ACCESS TIME 51

APPENDIX B – THE BANDWIDTH AS A FUNCTION OF NUMBER OF BANKS 54

LIST OF FIGURES

Figure Page

1. The Path-growth Method 7
2. An Example of a Critically Passed Path 9
3. The Address-selection Method 10
4. Logical to Physical Address Translation 13
5. RC Model and 3 Driving Methods of Interconnection Line 15
6. WSI Memory Model and Worst-case Time Delay 18
7. The Bandwidth according to the l, m changes 21
8. A Possible Way for WSI Memory Interleaving 23
9. An Address Identification and the Data Sequence 26
10. The Use of a 2-bit Control Tag 27
11. The Timing Diagram of an Address Identification WSI 29
12. The Hand-shaking of a WSI Memory 30
13. The Block Diagram of the WSI Memory Module 32
14. Two structures of the WSI I/O Selector 34
15. The Bus Control Module and the Block Diagram 36
16. A two-point Redundant Bus 38
17. The Block Diagram of the WSI Controller 40
18. A Multi-wafer Configuration of the WSI Memory 41

Figure Page

19. A Control Logic including the Refresh Tag 44

20. A Schematic Diagram of a R/W/Refresh Address Mapping 45

21. The Effect of Page Size upon Mean Access Time 53

22. The State-graph of (a,d)=(2,4) 56

CHAPTER I

INTRODUCTION

A hierarchical memory system usually consists of a cache memory in a central processing unit, a main memory and magnetic hard-disk memories. Usually, the data transfer between main and hard-disk memory is based on the block transfer like paged and/or segmented memory system. Wafer-Scale Integrated (WSI) semiconductor memories can be used as high-capacity memory devices between the main and the hard-disk memories. Because WSI memories are semiconductor devices, they can be a few hundred times faster than hard-disk memories. So if we use WSI memories in a hierarchical memory system, the overall memory access time can be greatly reduced.

A WSI system has many advantages over the conventional Printed Circuit Board (PCB) approach; a WSI system is more reliable, less expensive and faster because of the implementation of the whole system in a wafer [1]-[2]. However, a WSI system has a yield problem because the integration of a whole system in a wafer has higher probability of containing faulty modules. Therefore in a WSI system, we need some techniques, which are called the WSI configuration methods, to select good modules and link them together.

As examples of the WSI configuration method, "fault-tolerant" approaches have been introduced by Manning [3] and Aubusson and Catt [4]. Both methods use some kind of intelligent mechanisms to select the next neighboring good modules when the chain of good modules is growing. But those methods may have problems to reconfigure the chain when the chain is broken and to grow multiple chains in one WSI wafer.

On the other hand, Chesley [5] suggested the so-called "addressable non-redundant approach," which uses an address translation table to use only the good modules in a WSI memory. This method requires a pre-sorting test of faulty modules but has less problems in reconfiguration and multi-bank selection if contents of the address translation table are altered accordingly. So, in this thesis, based upon Chesley's suggestions, an actual WSI memory will be designed.

Objective

The objective of this thesis is to design a WSI memory system. If we use existing memory configuration techniques to design a WSI memory system, we may have several problems. There are too many chip-select lines, address and data lines, uneven delays to access the memory modules at different positions, and bus-failure problems. To reduce the number of chip-select lines, a novel address-selection technique named address identification method is introduced. In address identification method, each memory module can be accessed randomly but every word within each memory module can be accessed sequentially with the assumption that WSI memory is used in the paged memory system. To reduce the number of address and data lines, multiplexed address and data lines are used. To cope with the refresh request and the different time delays to access memory modules, self-refresh techniques [6]-[8] and asynchronous techniques [9]-[12] are used. To cope with bus failure problems, a bus-replacememnt mechanism is suggested.

Chapter II describes the WSI problems and the WSI memory configuration methods. Chapter III describes some topics on the memory system organization. To analyze the WSI memory organization, the effects of the page size in virtual memory systems and interleaved memory structures are analyzed. Also, the speed of WSI memory system is studied with a delay line model and WSI memory organization problems are illustrated. Chapter IV consists of the WSI memory module design, the wafer design, and the wafer controller design. The module design contains the address identification method, the data and control sequence, and the refresh handling problems. The wafer design deals with an I/O bank multiplexer and a bus repair mechanism. Finally, controller functions are described. Chapter V suggests a rather simple WSI module design when we adopt a central refresh control.

CHAPTER II

WSI MEMORY SYSTEM

WSI Problems

Yield Problem

When we use sliced semiconductor chips to design a certain system, we need the pad-layout of each chip to connect the input and output terminals of the internal circuit of the chip to the lead frame of the package. A WSI system, which does not have any sliced chips on a wafer, can eliminate all the pad-layouts of all modules (which are chips if sliced) except the pad-layout of the input and output terminals of a WSI wafer. In a WSI wafer, there are always flawed modules due to a variety of defects and we can not remove those faulty modules from a wafer. The difficulties of a WSI implementation arise from the existence of those defects. Even worse, those defects are unpredictable and are randomly dispersed throughout the entire wafer. So, a WSI wafer can never be free of some defects [13].

The cost of packaging by a WSI approach is about a half of that by a PCB approach [1]. But if each WSI wafer does not have enough numbers of good modules, we need to use more WSI wafers for the required total capacity of WSI systems. To keep the cost of WSI systems lower than that of PCB systems, the method to improve a WSI yield should be considered first.

To maintain an acceptable yield of a WSI wafer, we can use redundancies for some functional units. But a redundant unit may be faulty also. Preparing triple redundancies can overcome the redundancy-failure problems somewhat, but too many redundant units reduce the available wafer area for actual functional units. In a WSI memory system, because all the modules have identical structures, the yield improvement of a WSI memory is accomplished by selecting as many good modules as possible from a WSI wafer.

Power-dissipation

The disadvantage of a WSI system, and in some cases the most troublesome problem, is the power-dissipation problem. Although each module dissipates relatively little power, an entire wafer may dissipate enormous power. For instance, 100 8080A microprocessors integrated in NMOS technology could dissipate about 100W at a 1-megahertz clock rate [14]. That amount of power-dissipation needs a special package to remove the heat produced, and such packages with the cooling system are very expensive. Also, large power-dissipation causes physical stresses to a WSI wafer. Hence, the method to reduce power-dissipation should be considered either in semiconductor device physics or circuit technology.

Worn-out Failure

Any two WSI wafers can never be same because all the defects are located at different places with different amounts. So, in some applications, it is very difficult to extract proper failure distributions of entire wafers when WSI wafers are mass-produced. Also, in a WSI wafer, we can hardly screen out modules which will fail soon because each module will not be sliced. As a result, a WSI wafer has a higher failure rate than sliced chips. From this point of view, when we have a worn-out failure (which means a failure during the operation after some time has passed), we need to reconfigure a WSI wafer to use it again.

WSI Memory Configuration

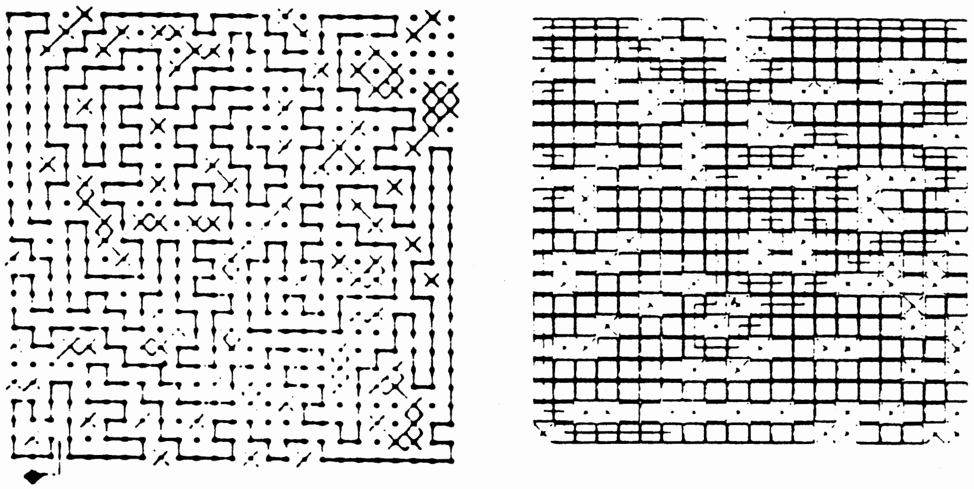
In a memory system, all the modules are identical. Only the external control circuitry is added to those identical modules; therefore in a WSI memory, it is a great advantage that the existence of some faulty modules will not cause a complete WSI failure, but only reduce the capacity of integration. Suppose that, in a 20-Mbyte WSI memory, 8 Mbytes of memory modules are faulty, then the remaining 12 Mbytes of memory in a wafer can be used. A CMOS DRAM is the best memory structure to achieve high integration density and low power-dissipation. Therefore, we choose the CMOS DRAM module as the basic building block of a WSI memory.

A recent WSI approach by Anamartic Ltd. holds about 200 1-Mbit DRAMs in a 6-in wafer [14]. A "Wafer Stack" (so-named by Anarmatic) has a maximum of $200 - \mu s$ access time. That is 200 times faster than conventional hard-disk memories. Wafer Stack uses a spiral array method and two wafers can form as much as 40 Mbytes.

Path-growth Method

Typical WSI configuration methods are represented by Manning [3]'s Cellular array method (see Figure 1a) and Aubusson [4]'s Spiral array method (see Figure 1b). These methods aim at the fault-tolerance in configuring large-scale systems without a pre-sorting test. The path is automatically grown by intelligently choosing the next neighboring good modules. In this thesis, those methods will be called the "path-growth" method. Figure 1 shows the way that the path-growth method achieves a long serial chain.

But the path-growth method has difficulty in reconfiguration when worn-out failures occur and lacks the flexibility to get multiple path-growing. Even though the path-growth method tests every module and reconfigure the wafer whenever



(a) Cellular Array (b) Spiral Array

Figure 1. The Path-growth Method

system is turned on, there may not exist an alternative reconfigurable path. As an example, in Figure 2, let us assume that a WSI memory has a critically-passed long serial path. In other words, a path grows through one critical position which is surrounded by faulty blocks of upper and lower positions or left and right positions. In case that critically-passed position turns out to be a worn-out failure after a moderately short time (probability of mortality is high in the early stage), the entire array path can be broken at that point. This reduces the reconfiguration flexibility greatly and there may not be any alternative chain at all.

Address-selection Method

In addition to the path-growth method, Chesley [5] suggested the so-called "Addressable non-redundant approach." This method suggests using the well-known virtual memory concept [15]-[17]. So, instead of excluding bad modules in the path-growth method, the bad modules are not accessed by prohibiting the address from being broadcast to those faulty modules. In this thesis, Chesley's method will be called the "address-selection" method. Figure 3 shows the addressable non-redundant approach. Even though the address-selection method needs a pre-sorting test of the faulty modules, this approach will be used in this thesis because it has better reconfiguration ability. Virtual fault-tolerance is achieved by using a WSI memory with the virtual memory concept because we can use every good module in spite of the existence of bad modules.

A WSI memory system can be used as a main or a secondary memory in a memory hierarchy. But in our approach, a WSI memory system is assumed to be used as an intermediate memory in front of hard disk memories. A combination of the address translation table and the wafer (memory) controller fools a processor as if the processor accesses continuous addresses.

In summary, the address-selection method will be used in this thesis. In

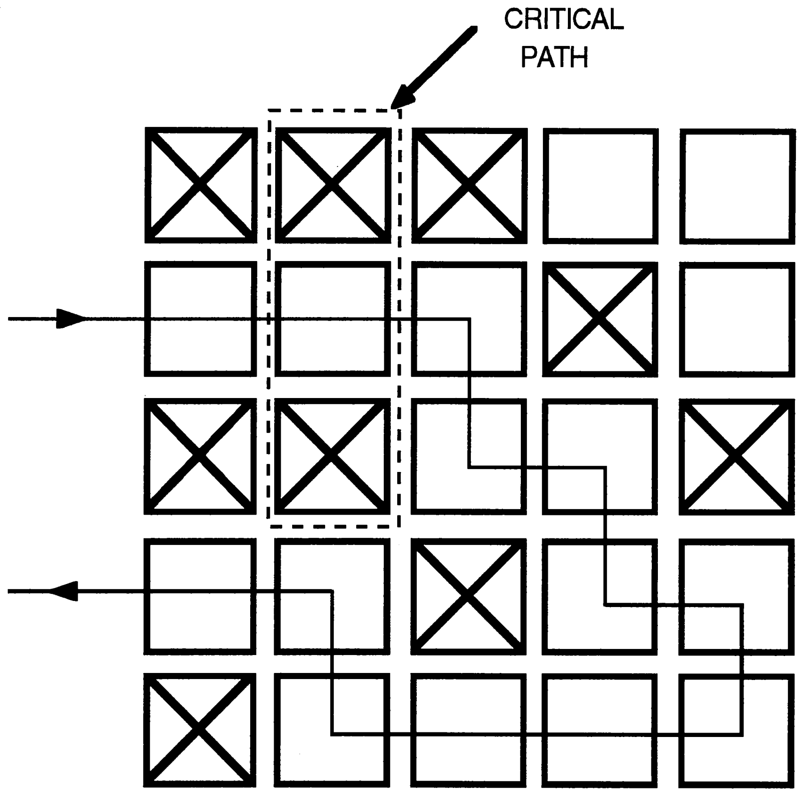


Figure 2. An Example of a Critically Passed Path

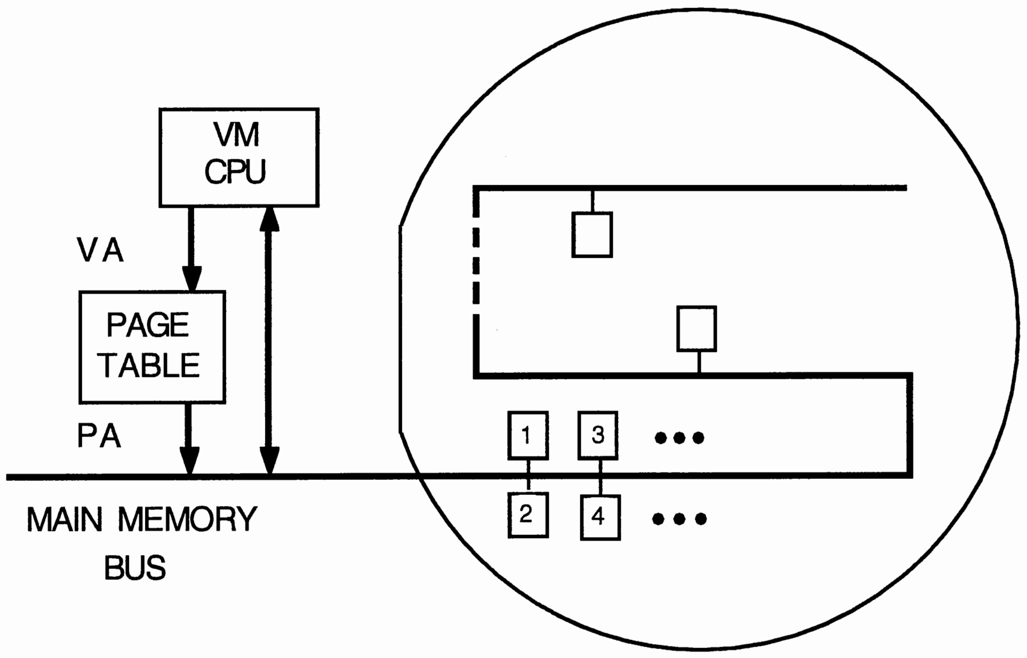


Figure 3. The Address-selection Method

the next chapter, we will discuss the memory and the WSI memory organization problems to find the necessary structures for the WSI memory design.

CHAPTER III

WSI MEMORY ORGANIZATION

In this chapter, a virtual memory system is studied first, and then a typical WSI memory system is analyzed by the time-delay and the memory access time.

Virtual Memory System

The virtual memory concept [15]-[17] is widely used in computer systems. The importance of this approach is that a logical address space is translated to a physical address space by the address translation table in a hierarchical memory organization. In address translation, the Address Translation Table (ATT) contains the necessary mappings. Figure 4 shows the concept of this arrangement. The ATT holds the page map tables and segment map tables. The segment selector indicates the position of the segment descriptor in segment tables. The segment descriptor contains the base segment address of segment frames. The page descriptor contains the actual page frame base address.

The translation is performed as follows:

(1) The virtual address is searched for in the cache. If the address in the cache matches, the corresponding physical address in the cache is used.

(2) If the virtual address misses the cache, but the page or segment is in the primary memory, then the physical address is transferred by the translation tables.

(3) If the virtual address misses the primary memory, then the page associated with the physical address should be transferred from the memory in the next hierarchy. In this case, the translation table must also be altered accordingly.

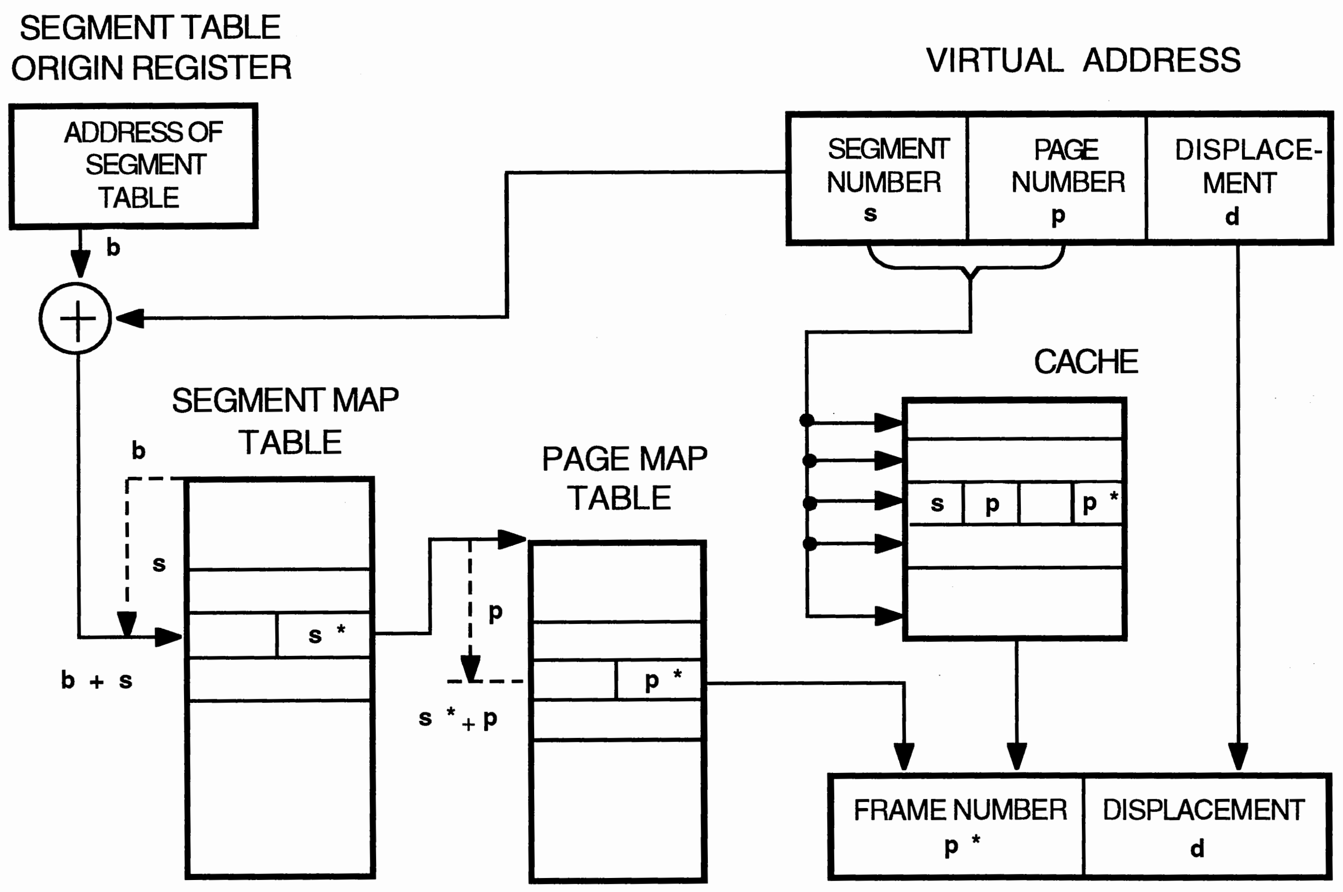


Figure 4. Logical to Physical Address Translation

In the paged and segmented virtual memory system, the memory access time depends upon the page size (see Appendix A for the result). When we arrange the page size small, the mean access time of the sequential page search is also reduced [18], which means that a small page size is preferable. But when considering the page fault ratio (miss ratio), a small page size may give a high fault ratio [17], [19]. Therefore, the optimal decision should be made with a compromise of the page size. This topic can be another important research area.

WSI Memory System Analysis

Line Delay

As signal lines run across a part of an entire wafer, a propagation delay becomes an important factor. Especially in a long interconnection line and with a small feature size, the interconnection resistance and capacitance may not be electrically negligible.

As an example, a $4cm$ Al line which is $1\mu m$ wide and $0.3\mu m$ thick and with $0.5\mu m$ thick insulator over the substrate produces 4000Ω resistance and $8.09pF$ capacitance, so the RC delays are as much as $32nsec$ [20]. This example shows that in WSI applications, the line delay can be far longer than the gate delay and be a considerable portion of the memory cycle.

A long interconnection line can be represented by several RC sections as shown in Figure 5(a).

The response at node V_k is given by [21]

$$\begin{aligned} C\frac{dV_k}{dt} &= (I_{k-1} - I_k) \\ &= \frac{(V_{k-1} - V_k)}{R} - \frac{(V_k - V_{k+1})}{R}. \end{aligned} \tag{3.1}$$

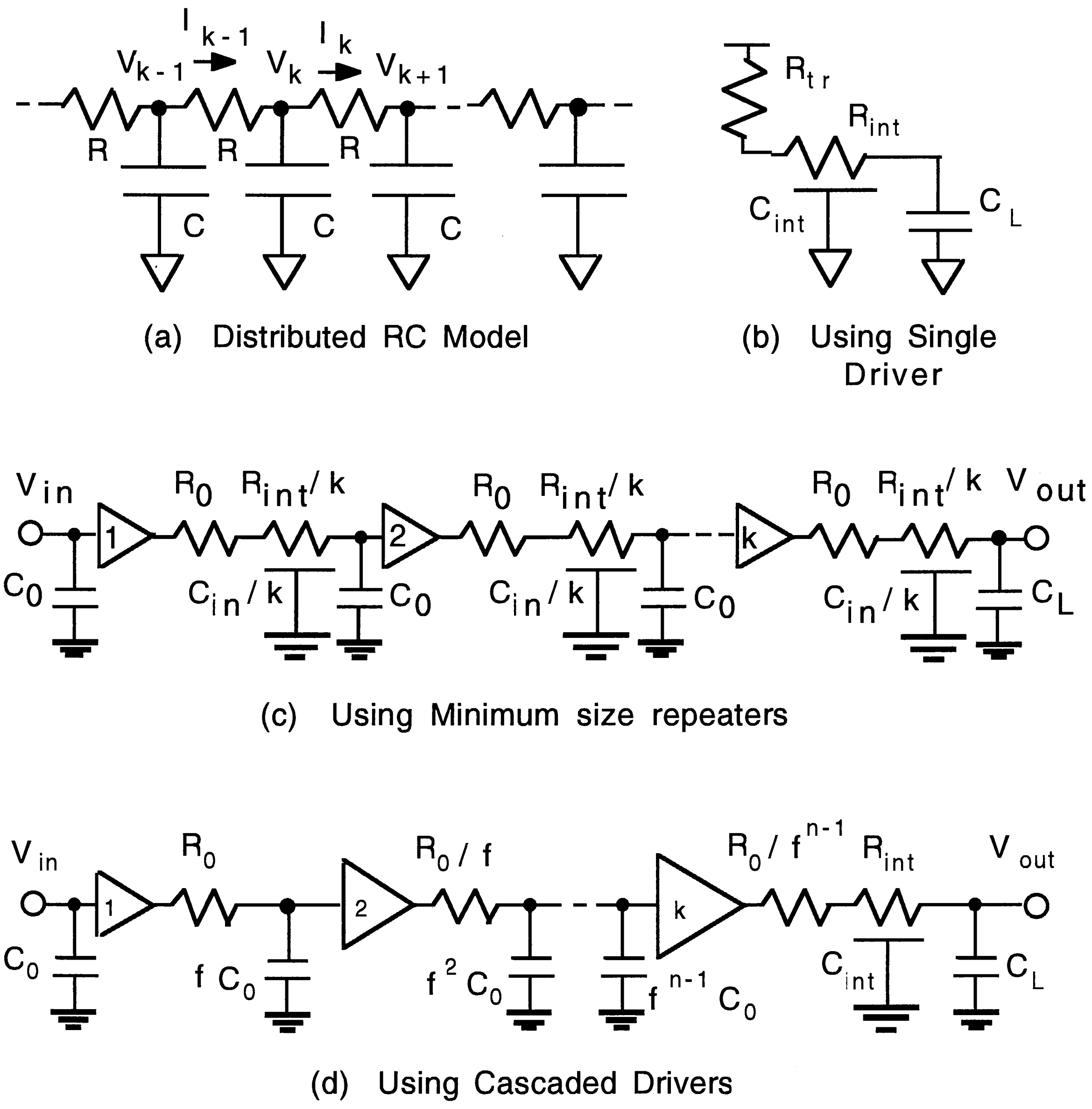


Figure 5. RC Model and 3 Driving Methods of Interconnection Line

When k becomes very large, Equation (3.1) becomes a differential equation:

$$rc\frac{dV}{dt} = \frac{d^2V}{dx^2}, \tag{3.2}$$

where x = distane from input, r = resistance per unit length and c = capacitance per unit length.

The propagation time t_l over wire of length l is

$$t_l = Kl^2, \tag{3.3}$$

where K is a constant [21], [22]. Equation (3.3) shows the dependence of t_l upon the term l^2.

Alternatively, when we assume that the driver has the on-resistance of R_{tr}, the interconnection resistance of R_{int}, the distributed capacitance of C_{int} and the load of C_L, as shown in Figure 5(b). Then the delay time is well approximated by [20]

$$T = (2.3R_{tr} + R_{int})C_{int}. \tag{3.4}$$

If we assume $R_{tr} << R_{int}$, Equation (3.4) is in agreement with Equation (3.3) because both R_{int} and C_{int} increases linearly with the length.

When we use appropriate repeaters in the interconnection line as shown in Figure 5(c), the propagation delay is given by [20]

$$T = k(2.3R_0 + \frac{R_{int}}{k})(\frac{C_{int}}{k} + C_0), \tag{3.5}$$

where k is the number repeaters used and R_0 and C_0 is the output resistance and input capacitance of the inverter. With optimal k,

$$T = (\sqrt{2.3R_0C_{int}} + \sqrt{R_{int}C_0})^2. \tag{3.6}$$

If $R_0C_{int} >> R_{int}C_0$, the delay is simplified to

$$T \approx 2.3R_0C_{int}. \tag{3.7}$$

Equation (3.7) shows the linear dependence of the delay upon the line length because C_{int} is proportional to l.

When we use cascaded drivers as shown in Figure 5(d), the delay is given by [20]

$$T = 2.3eR_0 C_0 \; ln\left(\frac{C_{int}}{C_0}\right) + R_{int}C_{int}. \tag{3.8}$$

If R_{int} is small and $C_{int}R_0$ is dominant, Equation (3.8) shows the logarithmic dependence of the delay upon line length l.

WSI Memory Model

The delay problem discussed in the previous subsection tells us that a long bus has a longer delay than a short bus. Moreover, the bus and power-line failure problems are also significant in a long bus structure.

In modeling a WSI memory system, the major difference between a conventional memory system and a WSI memory system is the existence of the faulty modules in a WSI wafer. For example, the modeling of a WSI memory is based on the following assumptions:

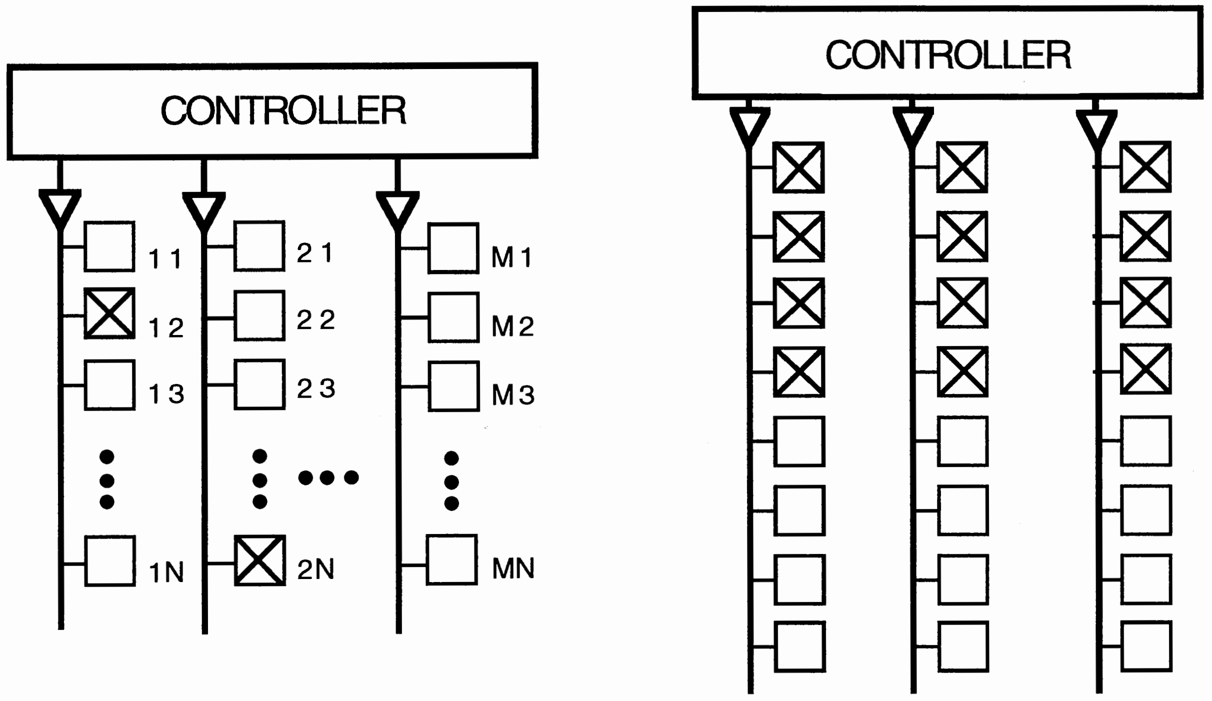
(1) Each memory cycle is composed of an Address Cycle (AC) and a Data Cycle (DC).

(2) Interconnection line has appropriate repeaters and the time delay (τ) to access memory module is assumed linearly dependent on the module position.

(3) The process time of the controller and the delay time of driver and repeaters are ignored.

(4) A WSI memory has M memory banks and each memory bank has N modules and each module has p words. Also, there may be several faulty modules assumed to be present in each bank.

According to the assumptions, Figure 6(a) illustrates the model of the WSI



(a) WSI Memory Model (b) Worst-case Time Delay

Figure 6. WSI Memory Model and Worst-case Time Delay

memory. In that drawing, WSI memory has several banks and each bank has the same number of memory modules. The box with 'X' mark indicates the faulty module.

Figure 6(b) shows an example that certain WSI memory has the worst-case delays. In that structure, M=3, N=8 and module yield equals 50%. Then the average delay to access memory module in one bank , $t_d = (5+6+7+8)\tau/4 = 6.5\tau$.

But in a WSI memory, it is hard to predict the total delay because every WSI memory will have random faulty modules. So we need some techniques to overcome unpredictable time delays. In the next chapter, an asynchronous module accessing method will be used.

From the result of the discussions about line delay, it is very desirable that a WSI wafer has a multi-bank structure rather than single long-bank structure. This implies that we have made a right choice by selecting the address-selection method instead of the path-growth method because the former can easily get a multi-bank structure.

Interleaved Memory Structure

To speed up the effective memory access, the interleaved memory structure is used [17]. This structure increases the total memory bandwidth by interleaving several memory modules which are accessed successively.

Previous works show that, in an interleaved memory system, the effective bandwidth (BW) depends on the average number of concurrently active modules [23], [24]. They use simplified analytic models to show that the BW is proportional to the number of interleaved memory modules. In a multi-processor computer system, the system performance also depends on the number of the interleaved memory modules even though some effects of memory conflict are included [25], [26]. But those models neglect any line-delays to simplify the analysis. Also it is assumed that a memory system is synchronized and every interleaved memory module is independent. But in a WSI memory, those assumptions are far from a reality due to the existence of faulty modules. The analytic model of an interleaved WSI memory is hard to set up because the WSI wafer has random faulty modules. But if we neglect the existence of the faulty modules in the wafer, the modeling of a WSI memory is exactly same as that which is presented by F. Briggs and E. Davidson [26].

When a memory system is interleaved in M ways and each memory bank has N modules, this organization will be expressed as an (M,N) memory organization. The system performance is characterized by the memory bandwidth.

We assume that each memory cycle is composed of an Address Cycle (AC) and a Data Cycle (DC). Also, each memory bank is independent and there is no distinction between the read and write cycle. According to the result (see Appendix B) [26], the memory bandwidth with (AC,DC) $= (a,d) = (2,4)$ and (M,N) $= (l,m)$ is

$$BW = l \times P_A(2,4) = \frac{(lm)^2}{(lm)^2 + lm^2 + 2lm - m + 1}, \qquad (3.9)$$

which shows that BW is higher when the number of memory banks, l, becomes large. Figure 7 shows that BW is also increasing according to the number of bank l, even though the total number of modules, lm, is fixed in a WSI memory.

There are two methods of memory interleaving [27]. The high-order interleaving has better module efficiency, but a sequential accumulation of the delays discourages the use of the high-order interleaving. On the other hand, the low-order interleaving accesses memory faster in a parallel or pipelined processor system than the high-order interleaving. Moreover, in configuring the multi-wafer system, the low-order interleaving is encouraging because it does not have a delay accumulation.

In a WSI memory, the module efficiency is determined by the bank which has

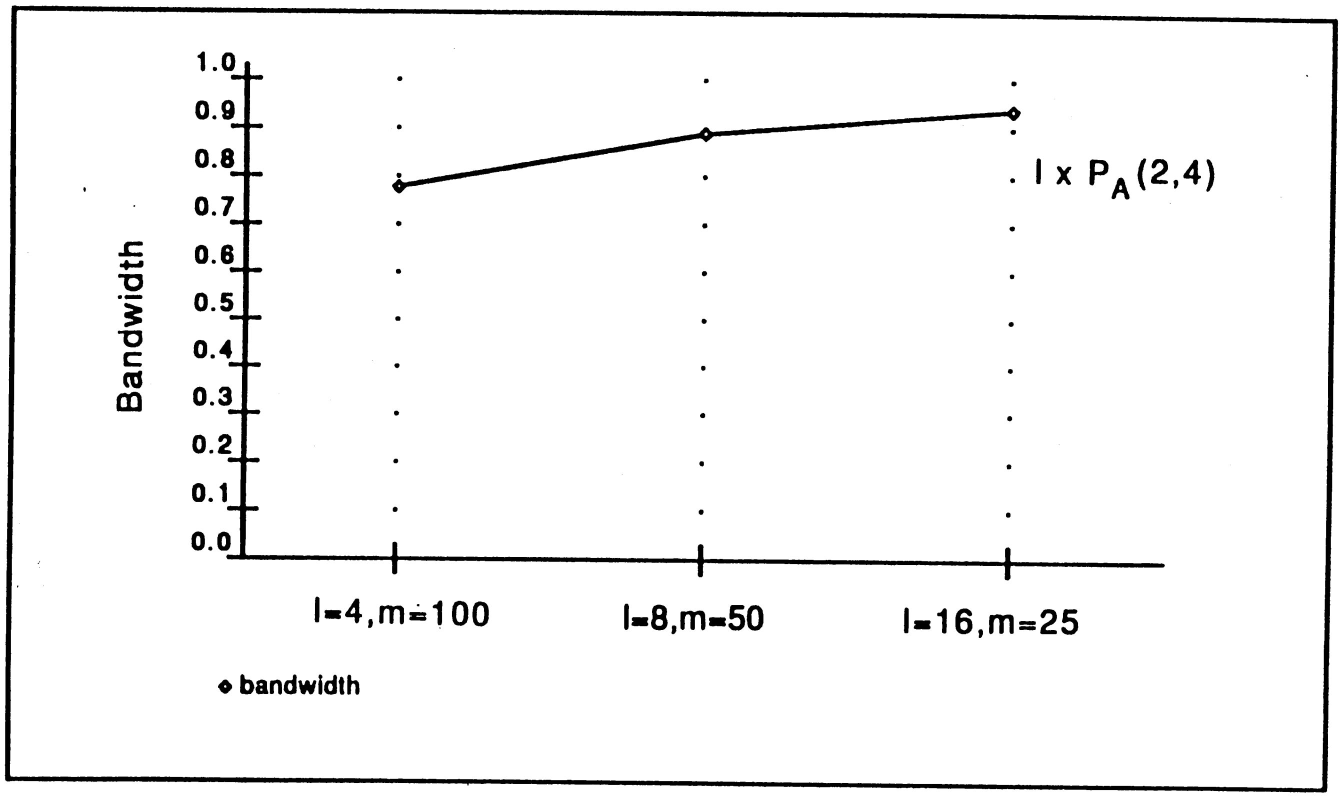


Figure 7. The Bandwidth according to the l,m changes

the least number of good modules. Figure 8 shows one possible way of WSI memory interleaving.

In WSI memory interleaving, we define the following:
The module yield is defined as

$$Y_M = \frac{total\ number\ of\ good\ modules}{total\ number\ of\ modules}.$$

The module efficiency is defined as

$$\eta = \frac{total\ number\ of\ used\ good\ modules}{total\ number\ of\ good\ modules}.$$

As a result, the system yield is

$$Y_S = \eta Y_M = \frac{total\ number\ of\ used\ good\ modules}{total\ number\ of\ modules}.$$

In Figure 8, $Y_M = \frac{28}{40} = 0.70$, $\eta = \frac{20}{28} = 0.71$ and $Y_S = 0.50$. In configuring a multi-wafer system, the memory bank which has excessively many faulty modules close to the WSI controller or which has exceptionally few working modules may be discarded in order to get a high module efficiency.

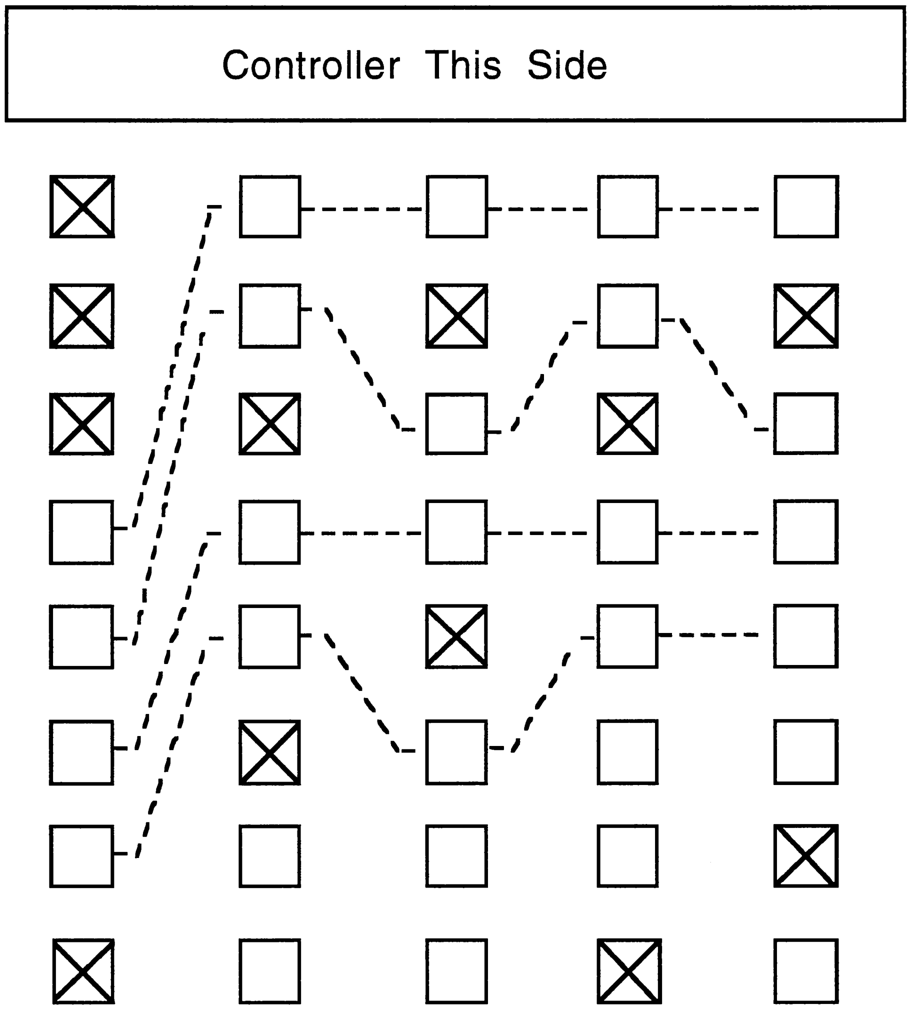


Figure 8. A Possible Way for WSI Memory Interleaving

CHAPTER IV

WSI MEMORY DESIGN

Module Design

Module Size Decision

If we increase the size of a memory module, the area of bus and control circuitry in a wafer can be reduced. However, each module failure means greater loss in the total memory capacity of a wafer. On the other hand, a small module size gives quite an opposite result.

The decision of a module size is dependent upon the organization of the system. If we use a paged memory system, it will be convenient that the address size of each module is equal to the page size. A typical page size is between 256 and 2048 bytes, while segments can be 64 Kbytes or more [16]. In our WSI memory, the module size can be decided as 32 Kbytes if we use an 8-bit address bus when a 4-bank WSI wafer is used for the capacity of 20 Mbytes. But if we use a 16-bit address bus, there are no restrictions on choosing the module size because we can select one out of 2^{16} modules in each bank.

Address Identification Method

A new WSI addressing method, in this thesis, is called the address identification method. To access the designated memory module, each memory module has an address IDentification (ID) comparator, an ADDR/DATA MUX (Selector) and an address counter, as in Figure 9.

Each memory module has its own address ID tag which is pre-programmed permanently in the address ID comparator. When an incoming address from the WSI controller through the ADDR/DATA bus matches the address in the address ID comparator, the address counter of the matched module will be reset and ready to proceed.

The distinction between address and data sent from the controller is performed by checking the control tag (bits) of the incoming data. Figure 10 shows the usage of a control tag. If the control high bit is '1', then the ADDR/DATA selector is switched to the address ID comparator and this stage is called the address sequence. If the control high bit is '0', the data bus is connected to the data buffer of the memory module and this is called the data sequence. After the address sequence, successive data are transferred via the ADDR/DATA bus with the control high bit '0'.

In addition to the control high bit, the control low bit is used. To provide the necessary clock for the address counter, the control low bit has the alternating values of '0'→'1'→'0'→ ⋯. As a result, this control bit acts like a locally-synchronized clock. Then the address counter incremented for a sequential word access of each page-sized module.

When we use a two-bit control tag, the control low bit can also be used as a READ/WRITE tag during the address sequence. The '1' value can be a write tag and '0' value can be a read tag.

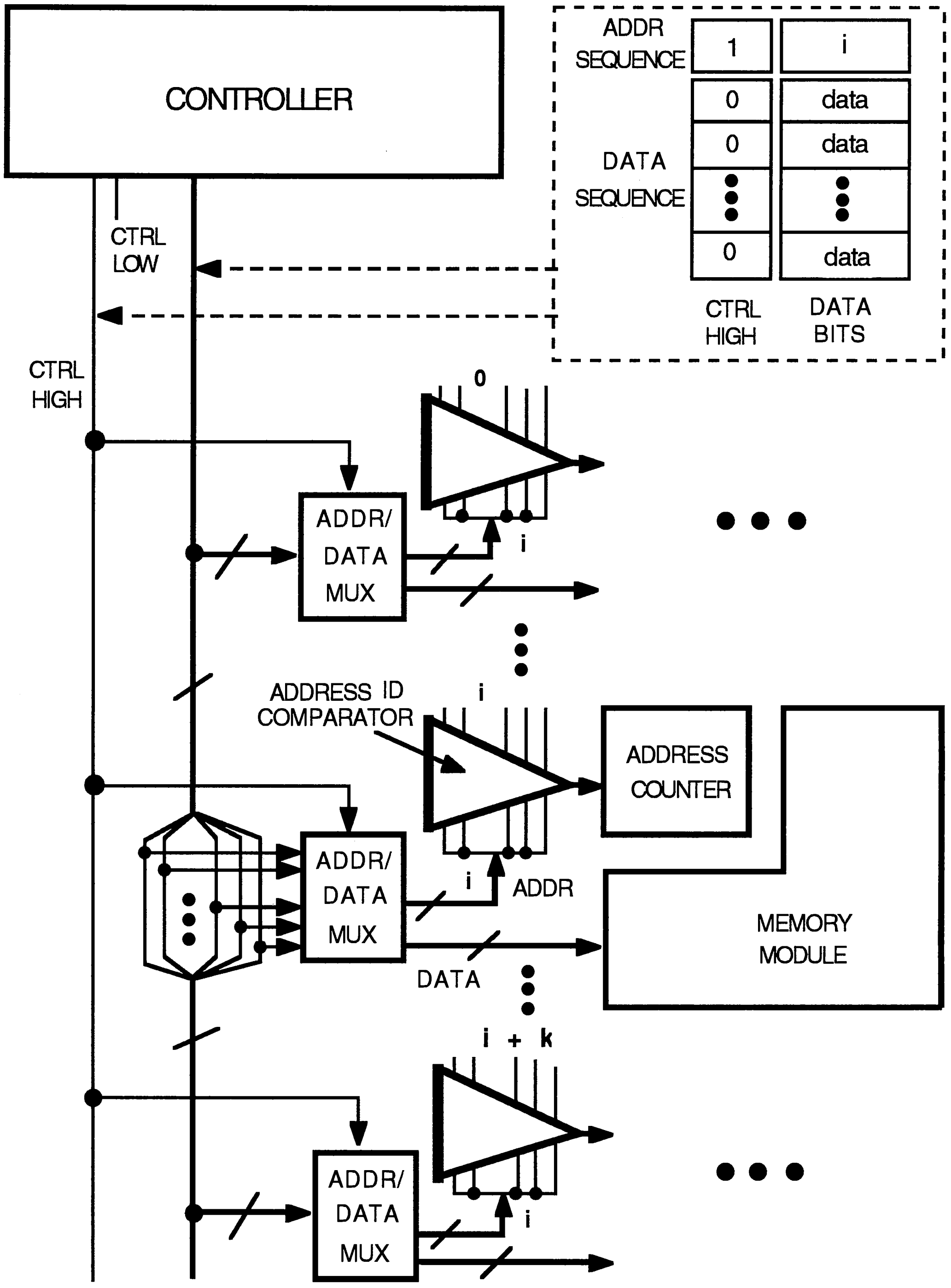


Figure 9. An Address Identification and the Data Sequence

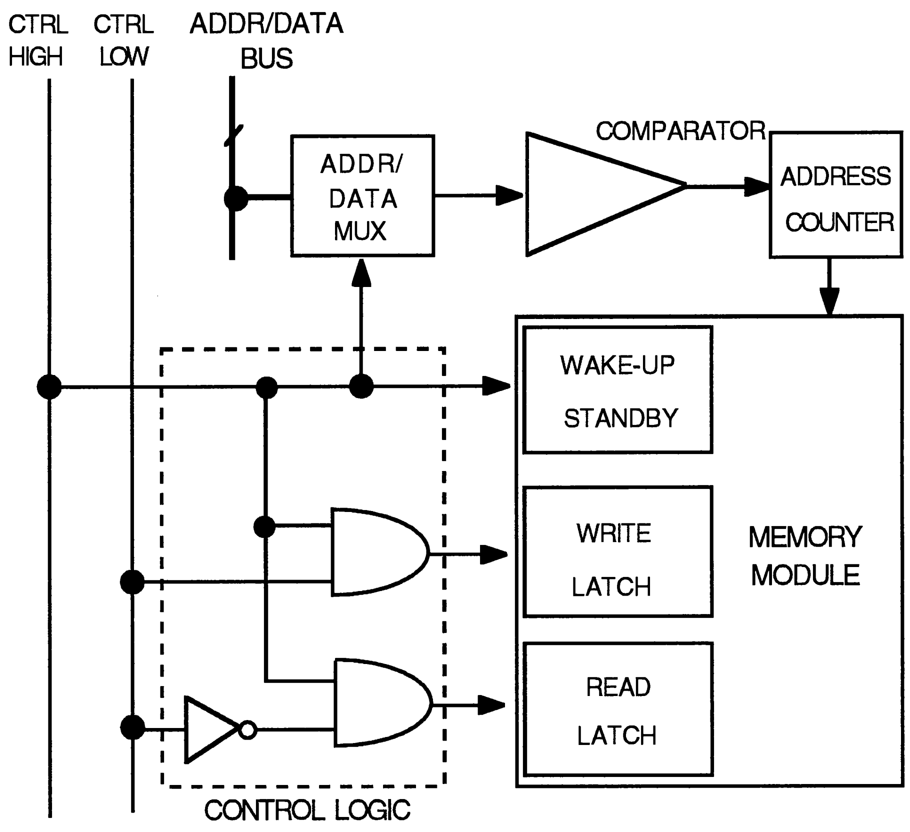


Figure 10. The Use of a Two-bit Control Tag

Timing Diagram of WSI Memory

The timing diagram of the address identification WSI memory is shown in Figure 11.

In the address sequence, the controller sends the address ID tag with control high bit '1'. The transition of the control high bit lets the ADDR/DATA Selector switched to the address counter. According to the address ID tag, the ADDR/DATA Selector lets the address ID comparator have a '1' value if the address is identified. Then the address counter is reset until the address sequence terminates.

Meanwhile, the control low bit designates the following data sequence as a memory read or write cycle. After some predefined time interval, t_{pf}, the controller sends the actual data to the memory module. Then the data sequence will start immediately.

In the data sequence, the control low bit now provides the clock of the address counter. With this clock, the address counter generates a sequence of addresses to access a page in the module. Each memory cycle repeats with the internal timing control until all the page cycles are terminated. In a module READ cycle, the internal control tag generator sets and resets the control bits alternatively in the same manner as the controller did in a module WRITE cycle.

Refresh Hand-shaking

Each memory module has an automatic self-refresh ability, but the contention between refresh and memory access is often a problem. A refresh cycle usually has a higher priority. So, there should be a method to solve this contention problem. One common way is to use a hand-shaking method. There are two cases in which we need the hand-shaking between the WSI controller and the memory module as in Figure 12.

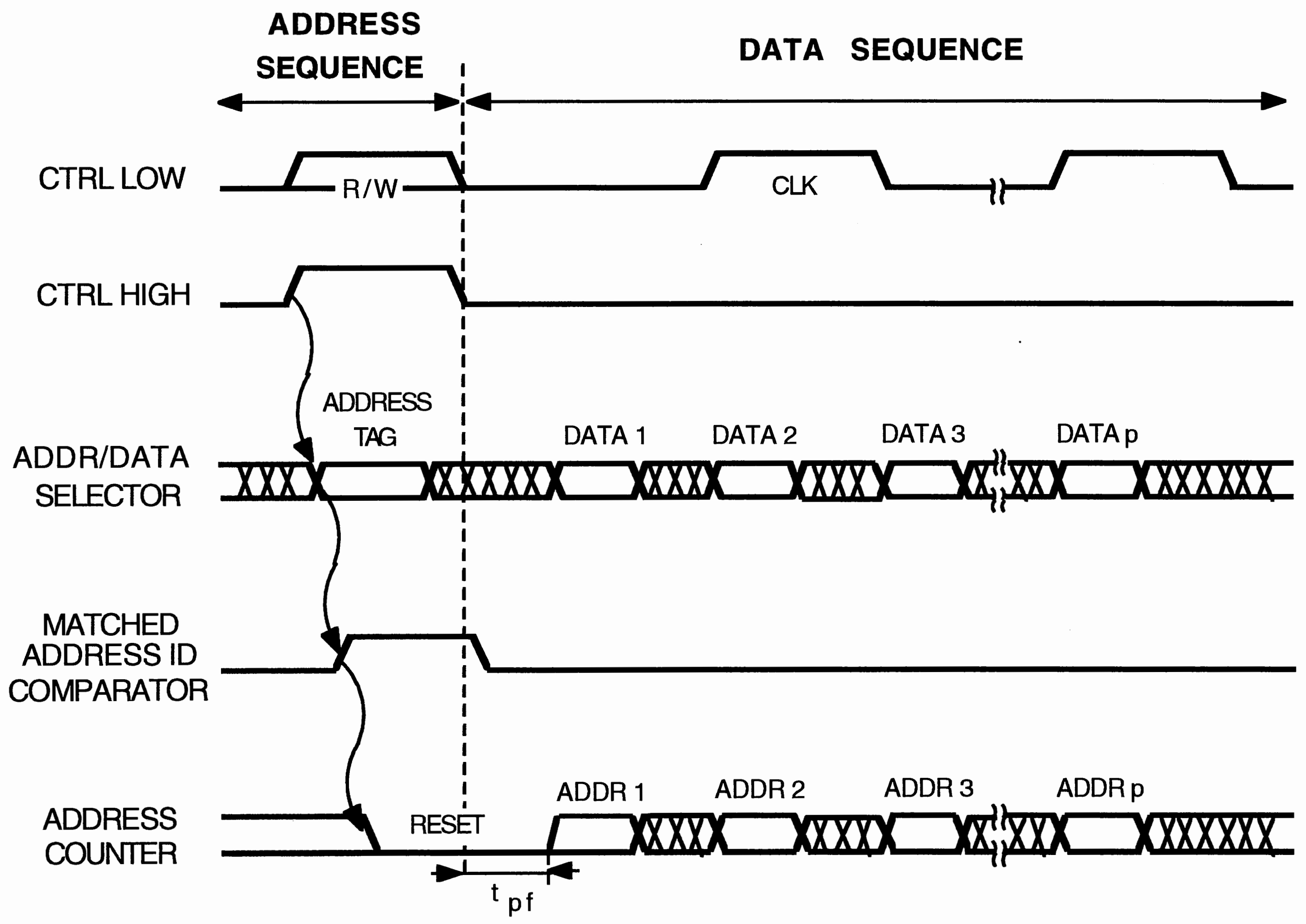


Figure 11. The Timing Diagram of an Address Identification WSI

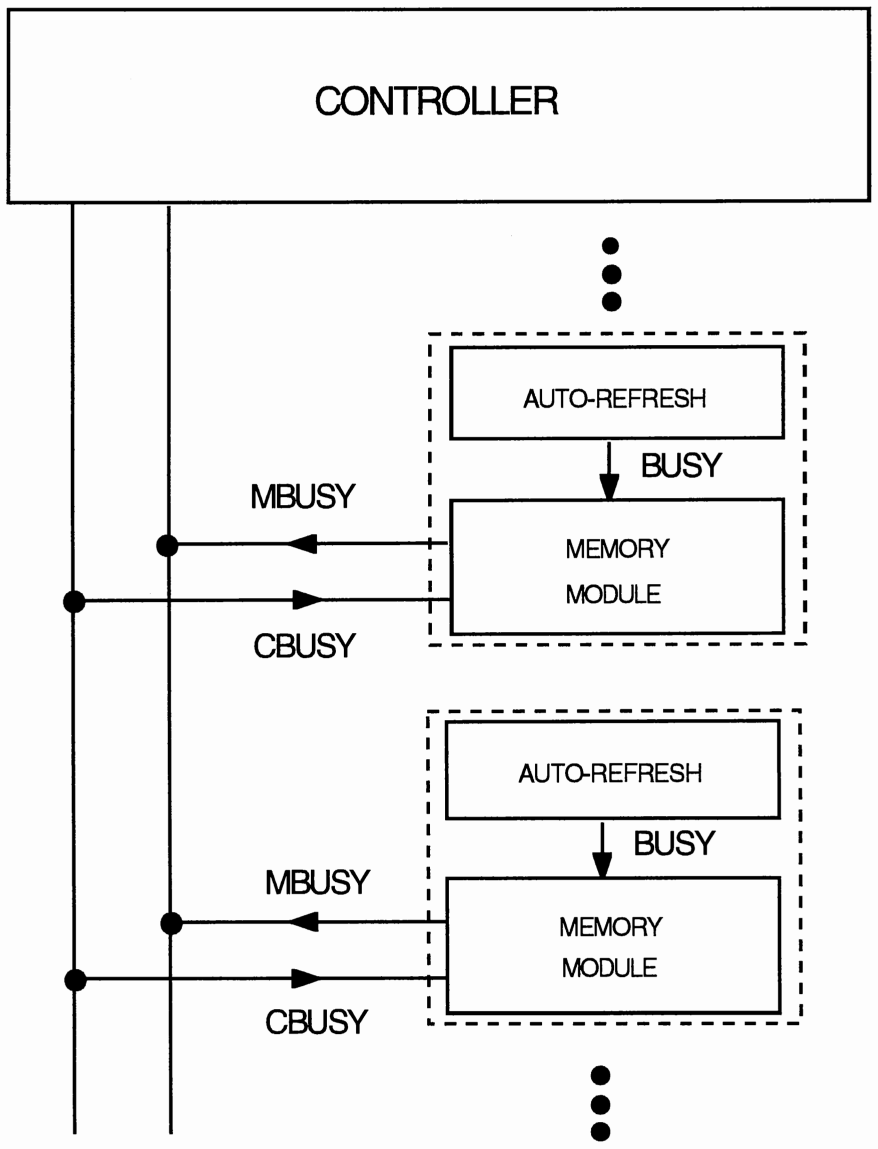


Figure 12. The Hand-shaking of a WSI Memory

The first case is the busy acknowledgement from the memory module to the controller. This signal is necessary when a module needs refreshing in a module WRITE cycle. This signal will hereafter be called the Module BUSY (MBUSY) signal. When a module enters into a self-refresh mode, the refresh controller sends a BUSY signal to the hand-shaking controller in the memory module. If a module is accessed by the ID tag at this moment, the hand-shaking controller sends an MBUSY signal to the controller. Once the MBUSY signal is activated from a certain bank, the controller stops sending data through that bank. Then an actual refresh cycle should be started after detecting no changes in the incoming data to allow for the bus delay. This arrangement prevents the loss of data during an MBUSY acknowledgement. If an MBUSY signal is inactivated after a module refresh, the controller sends the interrupted data to the module again. If a module is not accessed during self-refresh, the module does not send an MBUSY signal to the controller and refresh is done internally by read-precharge-write steps. This refresh is called the hidden module refresh.

The second BUSY acknowledgement is from the controller to the memory module. This signal is necessary when the controller is busy doing another job during a module READ cycle. This is called the Controller BUSY (CBUSY) signal. In this module READ cycle, a memory module sends data to the memory controller. Therefore, if there is a refresh request or the CBUSY signal from the controller, the module simply interrupts the data transfer and accomplishes refresh if needed. If we assume that the controller is never interrupted in a module READ cycle, the CBUSY signal will not be necessary.

To culminate the discussions to this point, a block diagram of a WSI memory module is presented in Figure 13.

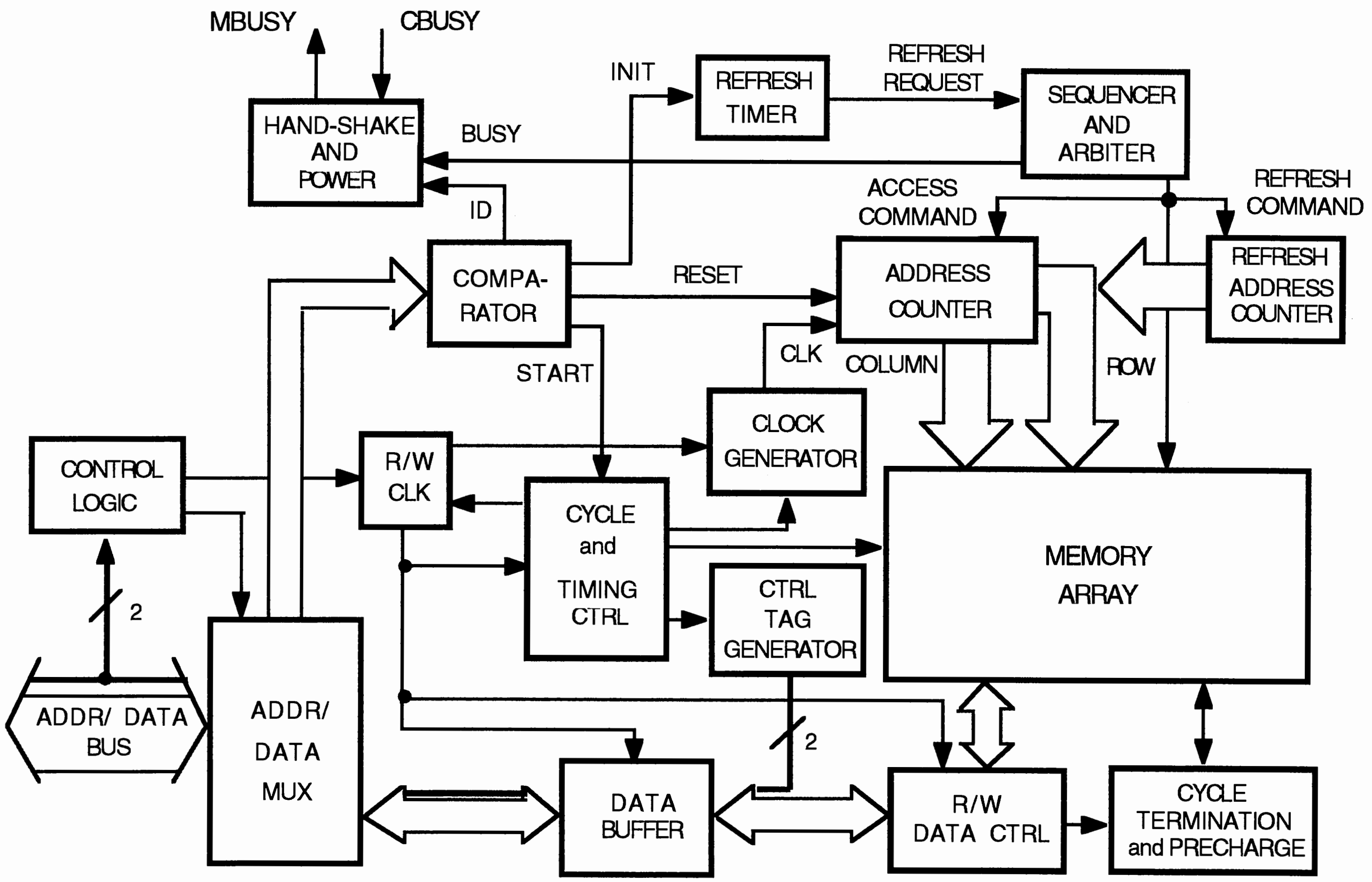


Figure 13. The Block Diagram of the WSI Memory Module

Wafer Design

Wafer design includes the layout of modules, bus lines, power lines, and input and output peripheral circuitry. Besides, in a multi-bank WSI, key issues are the bus I/O multiplexing method and bus failure protection.

Wafer I/O Selector

To see the effects of the number of I/O bus selectors on the memory performance, let us compare two structures of a 4-bank WSI memory example in Figure 14. In those structures, the bus multiplexer input is the decoded page address from the controller to each bank.

The first structure, in Figure 14 (a), is the 1-I/O and 4-bank system. This structure reduces the number of input and output pads, but one multiplexer may be a bottleneck in the system performance, and a wafer can only be a single access memory system.

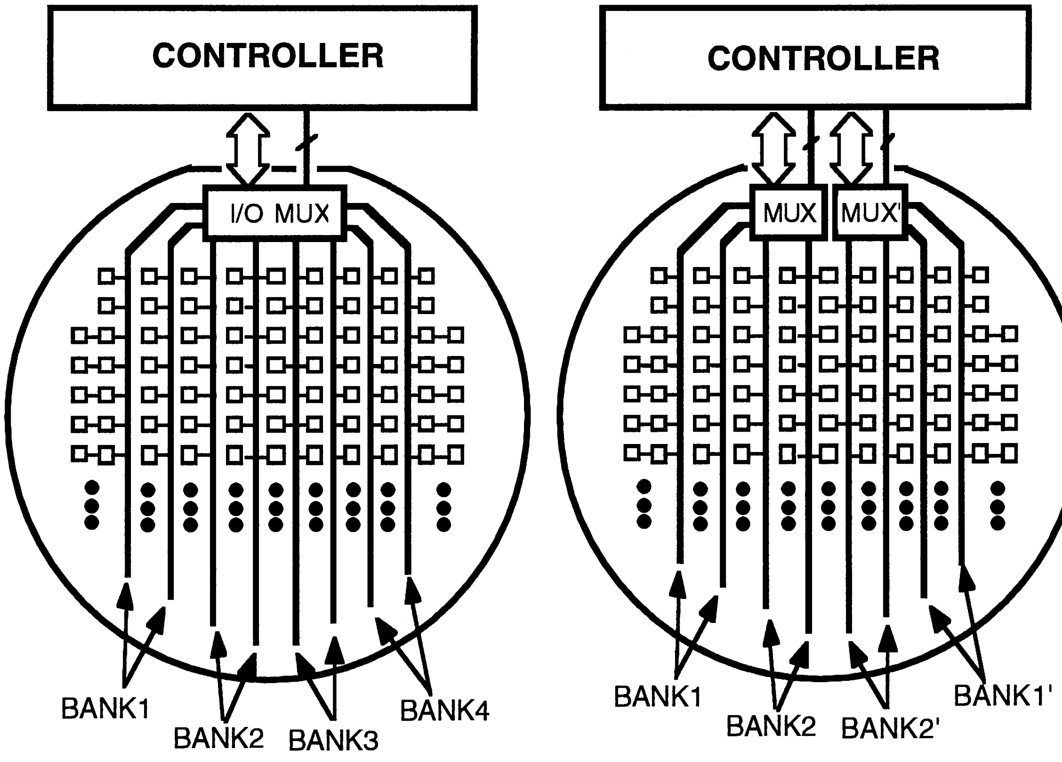
The next and more preferable structure, in Figure 14 (b), uses double 1-I/O and 2-bank structure. Each I/O has its own multiplexer and each multiplexer drives two memory banks.

Design Factors

The first factor to be considered is the bus failure problem. In a WSI wafer, bus lines run through entire memory modules, so any one bus failure causes critical results. To prevent this result, a redundant bus will be used.

The second problem is the signal delay. In a WSI system, many signal lines are expected to connect the modules placed widely apart. So, an equi-distance layout of signal lines from input pad to functional blocks is preferred.

The next problem is power-line failure. Once entire power lines are tied to



(a) 1-I/O 4-Bank Structure

(b) Double 1-I/O 2-Bank Structure

Figure 14. Two Structures of the WSI I/O Selector

every module permanently, a local spot power failure can make the entire wafer useless. Unfortunately, preparing a fusable link [2] for a power line is not easy because it will produce significant voltage drops. A possible solution is to provide seperate power blocks for each of the memory banks.

Bus Redundancy

Because WSI bus lines are very critical, we need redundant buses. The problem is how to provide a bus switching scheme for bus redundancies. Redundant bus lines can be connected to every bus line via switching elements. If some bus lines turn out to be failures, switching elements interchange the failed bus line with one of the redundant bus lines.

The implementation of switching elements needs some underlying technology: one method is using a fusible link technology [2]. But this method requires the repairs on the surface of the wafer, which may cause other defects during the repairs. Other method is to use a programmable logic (like PROM) to select redundant buses; this method, however, needs additional control function and program to change the logic of bus replacements.

Figure 15 illustrates a novel method of selecting the redundant bus lines. In this method, we use the bus control module which has its own address tag like other memory modules and has switching elements for bus replacements. The bus control module consists of the address ID comparator, the bus controller and the bus switching elements. The WSI controller accesses the bus control module during the bus selection procedure in a test program. During that procedure, the wafer controller first sends the address tag of the bus control module. If the bus control module is matched by that address tag, then the WSI controller reads preset '1' data from the bus presetter. If all the received data are '1', then all the bus lines work correctly. If some of the data are '0', those bus lines should be replaced.

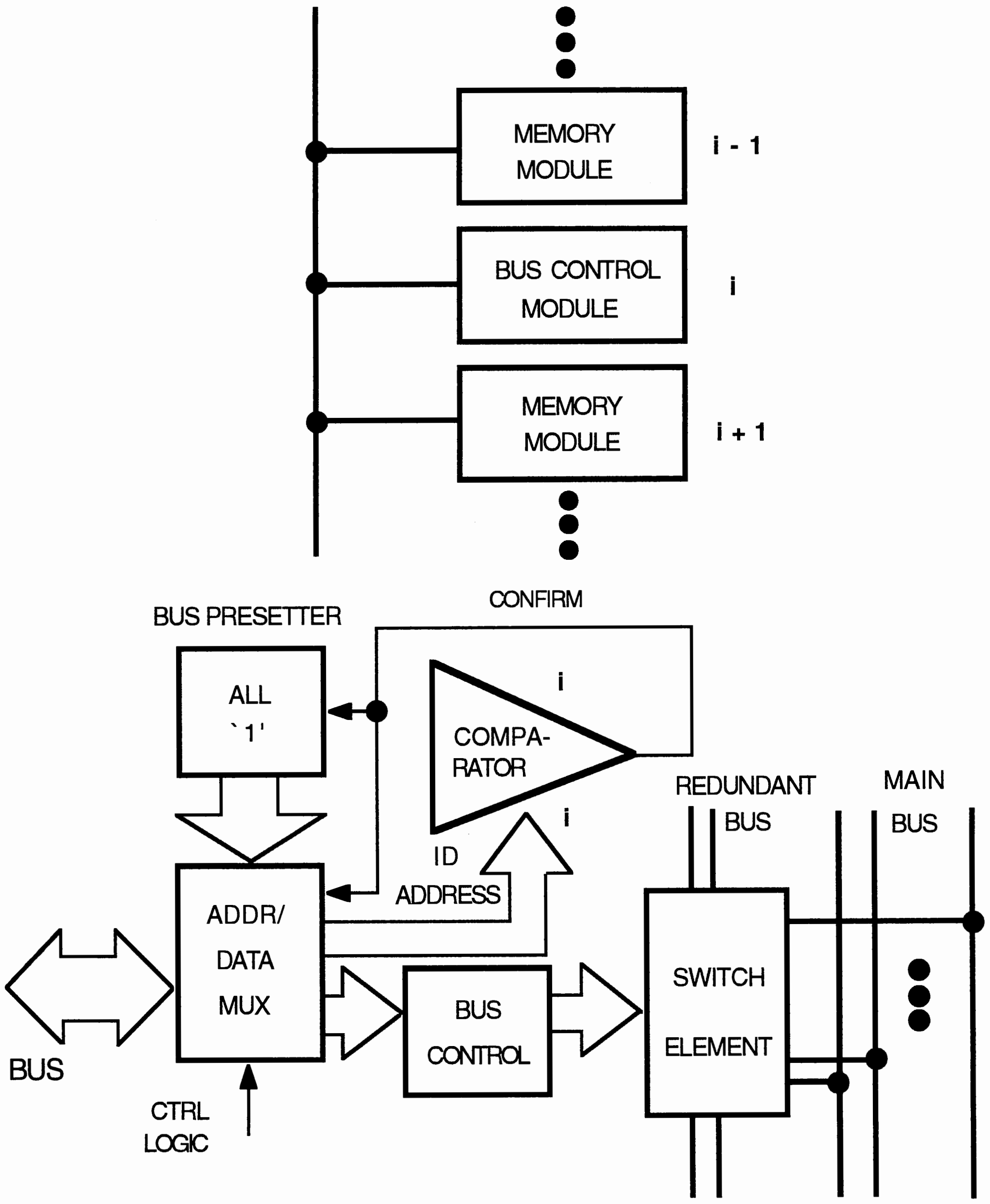


Figure 15. The Bus Control Module and the Block Diagram

According to the received data, the controller decides which bus lines need to be replaced. Then the controller sends an encoded number which indicates faulty bus lines via live bus lines. Finally, the bus controller decodes those numbers and selects switching elements accordingly.

A two-point bus redundancy, as shown in Figure 16, can cover any bus open or short without causing any bus loop. Multiple bus failures can also be repaired by providing more redundancies.

Controller Design

Address Generator

The WSI memory controller should supply the necessary signals to memory modules. The lower bits of the WSI physical address is generated by the address counter in the memory module. For example, if a module size is 32 Kbytes, the address counter represents the lower 15 bits of the WSI physical address. The higher portion of the address corresponds to the page address of each module, so the controller needs the address generator for these page addresses. An external processor or an address generator in the WSI controller can generate these page addresses. Also, the controller should send an address of the memory module with the control tags. The controller has the same address counter as that of the memory module to receive data in a module READ cycle.

Controller Buffer

Due to the signal delay, we used an asynchronous data transfer in our WSI memory. So, we need to provide a buffer to the controller which is the same size as the memory module. The use of the controller buffer in a WSI memory increases the access time of the memory cycle somewhat. But an adequate timing control,

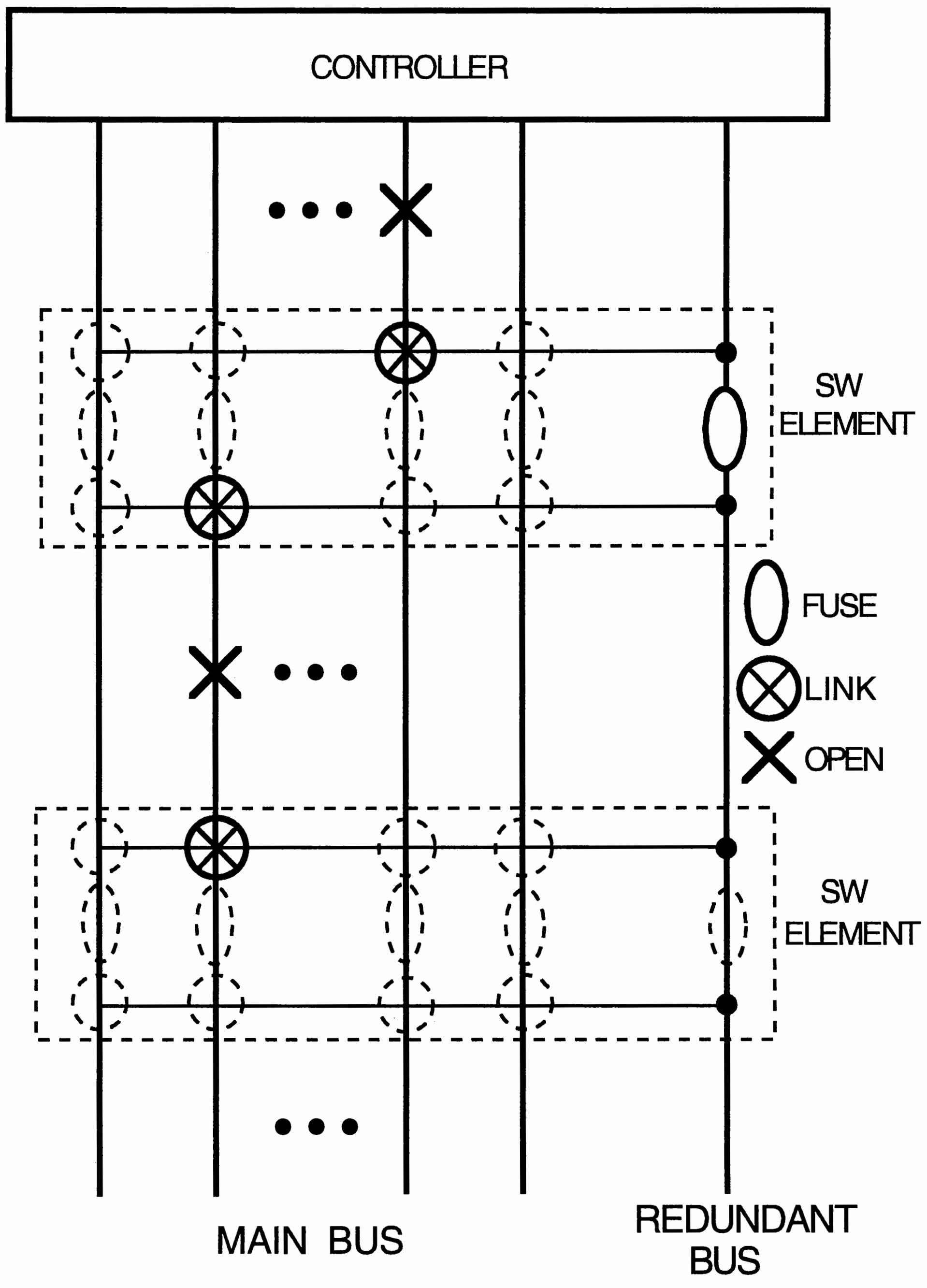


Figure 16. A Two-point Redundant Bus

which is similar to that of the pipelined structure [17], will reduce the access time. When we use a 2-I/O wafer, the WSI memory controller also needs two buffers.

Address Translation Mapping

A WSI memory will not have continuous physical addresses due to the flawed modules. The controller translates the logical page address of the processor to the physical page address of the WSI memory module. This translation is done by the Address Translation Table. If a WSI memory is used already in the virtual memory spaces, the system Address Translation Table can include these mappings.

The input and output signals of the WSI memory controller are similar to those of the conventional memory controller. There are the READ/WRITE signal, the system CLOCK, and the REQUEST signal from the processor. The necessary signals between memory modules were discussed throughout this chapter.

Figure 17 shows the block diagram of a WSI memory controller.

Multi-wafer Configuration

The extension to a multi-wafer configuration is quite easy if we prepare a proper controller arrangement. But a multi-wafer controller should be designed carefully so as not to degradate system performances. Figure 18 shows the multi-wafer memory configuration.

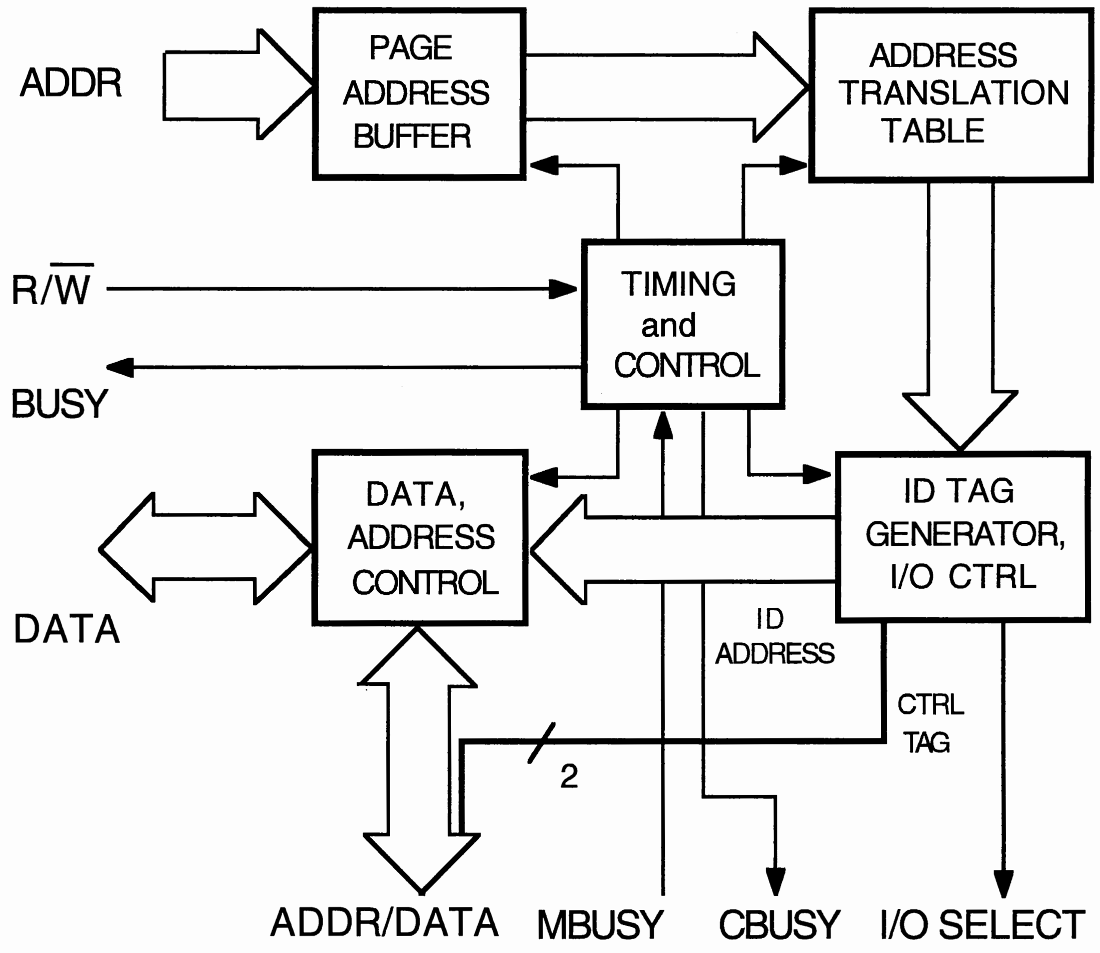


Figure 17. The Block Diagram of the WSI Controller

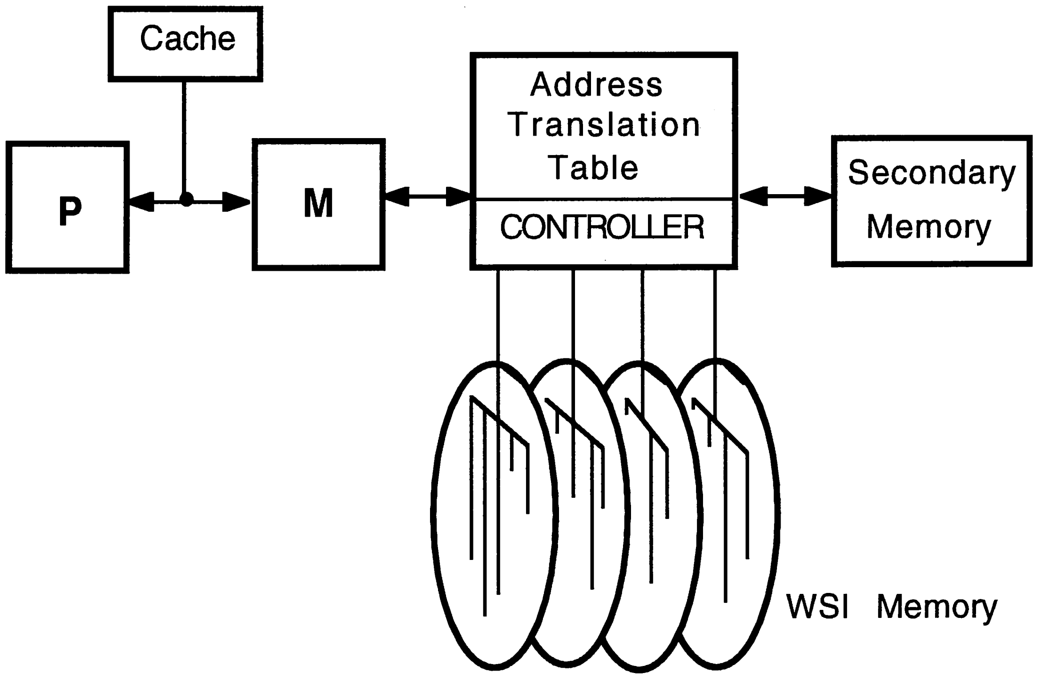


Figure 18. A Multi-wafer Configuration of the WSI Memory

CHAPTER V

SIMPLE WSI DESIGN

The WSI memory introduced in Chapter IV is based upon the adoption of two techniques: a self-refreshing of memory modules and an asynchronous data transfer between the memory module and the WSI memory controller. In this chapter, some brief suggestions about a simple WSI memory design are to be discussed. In the previous chapter, we discussed the mechanism of addressing and data transfer. Due to the asynchronous data access and the random refresh request, our WSI memory has a rather complex hand-shaking mechanism. So, the removal of hand-shaking will results in a simple WSI memory which uses the same address identification method as that of the WSI memory designed in the previous chapter.

When we use the following assumptions, the design task of a WSI memory is quite a bit simpler:

(1) Each memory module does not refresh by itself.

(2) The memory refresh is done by a centralized refresh control from the WSI controller.

Central Refresh Control

Originally, self-refresh concepts were developed to reduce the burden on the processor involved with refresh handling. The continuous time-keeping of the memory refreshing prohibits the processor from operating to its full capability. The self-refresh method can provide the elimination of time-keeping from the processor because the self-refresh counter in the memory module keeps its own refresh-timing.

If we use the self-refresh in each memory module, the memory controller cannot predict the exact refresh time of each memory module. Therefore, the refresh cycle can interrupt the memory cycle at any time. In other words, a refresh cycle can interrupt in the midst of a module page-sized data transfer. As a result, the MBUSY and the CBUSY acknowledgement are necessary.

One way to solve this problem is by the elimination of entire acknowledgements due to the refresh interruption. This approach is using a central refresh assignment from the WSI memory controller. If the WSI memory controller distributes the timing of the refresh for every module, there is no need for a BUSY acknowledgement. The WSI memory controller initiates a refresh cycle by sending the address tag and the refresh tag. When the controller assigns a refresh task to a certain module, that module is either in a READ or WRITE cycle, or is at rest. If the module is in a WRITE cycle, the WSI memory controller can skip the refresh assignment. If the module is in a READ cycle, the WSI memory controller should prohibit the READ cycle until the refresh cycle is assigned first.

In this approach, there should be a slight modification of the control logic in the previous chapter, as shown in Figure 19. Because we need the refresh information, '00' value of the control tag can be assigned for the refresh tag. Consequently, the control tag logic is modified to have the exclusive-ORed control bits instead of the control high bit only.

The advantage of central refresh control is that only a good module can be refreshed by the WSI memory controller. Figure 20 illustrates that, according to the good module address mapping, the contoller simply accomplishes READ/WRITE/REFRESH cycles.

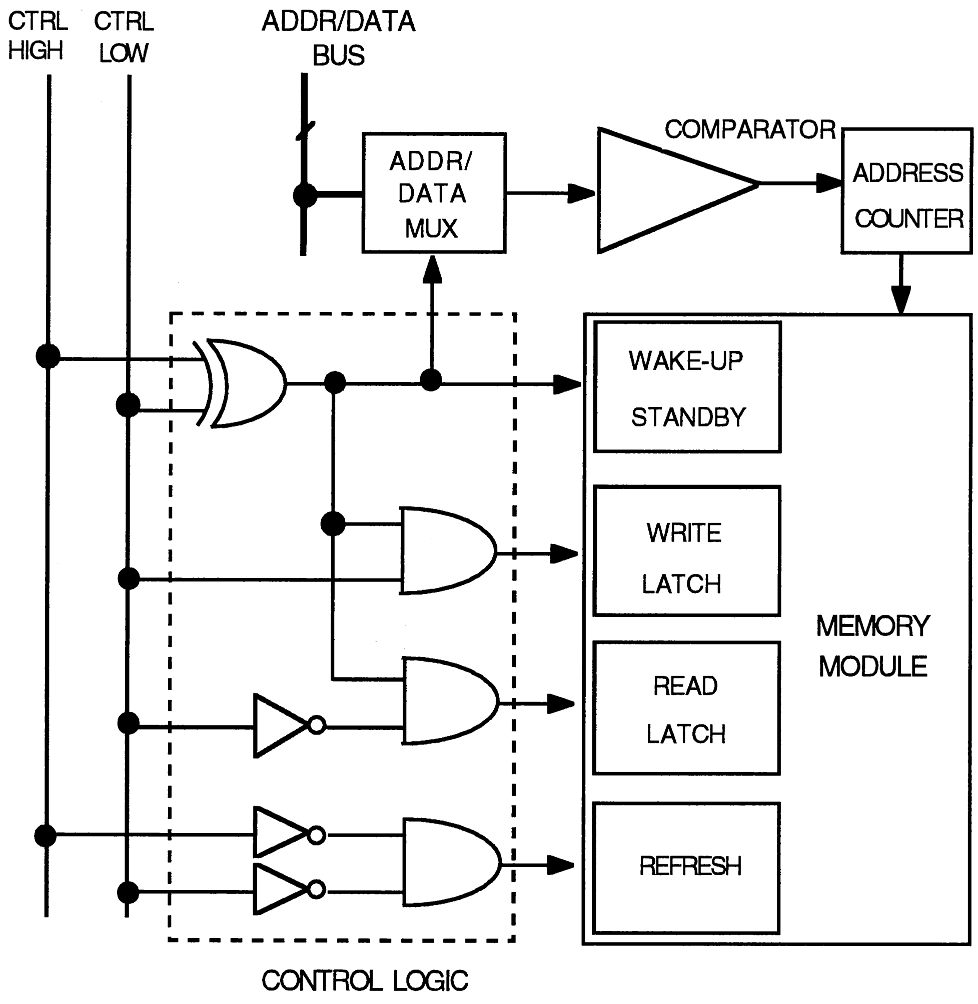


Figure 19. A Control Logic including the Refresh Tag

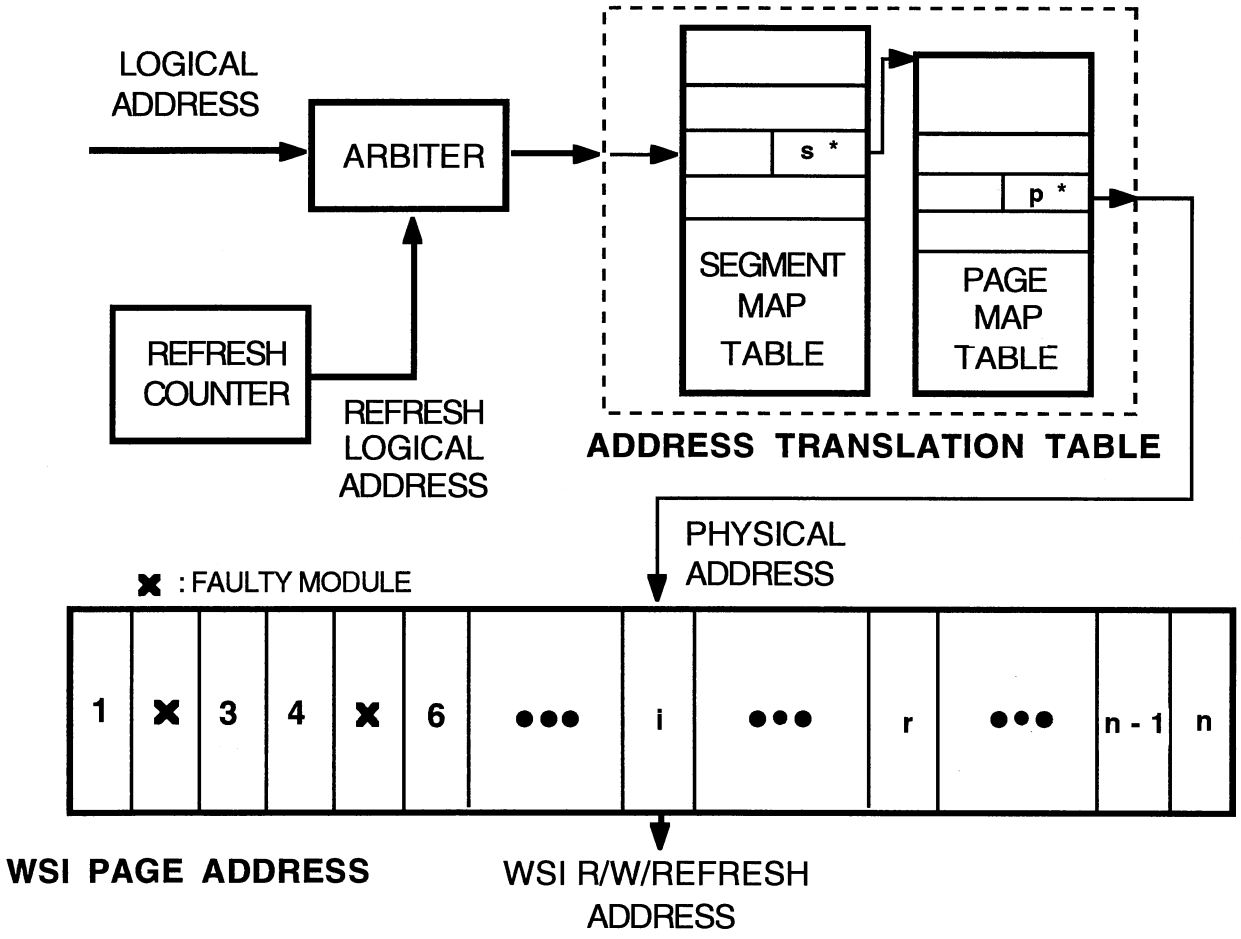


Figure 20. A Schematic Diagram of the R/W/Refresh Address Mapping

CHAPTER VI

CONCLUSIONS

Although the path-growth method provides a long serial chain of good modules, it may have difficulties in reconfiguring the chain and in growing the multiple paths. On the other hand, the address-selection method gives much flexibility in those matters.

The uneven path delay and the existence of flawed modules make the WSI memory have a multi-bank architecture. A time-delay analysis shows that a multi-bank structure is better than a one-bank structure and that an asynchronous data transfer is preferable.

By using an address identification method, the conventional addressing problem is solved. The Address/Data multiplexed bus reduces the total number of bus lines. Due to the asynchronous data transfer and the refresh requirement, the hand-shaking between the memory module and the WSI controller is necessary. A control tag can handle the addressing and the memory READ, WRITE cycle identification. The bus failure in a WSI memory may cause the entire wafer useless. A two-point bus selective switching scheme can repair either open or short bus failures. By including the Address Translation Table in the WSI memory controller, a multi-wafer configuration can be used in the virtual memory architecture. The central refresh control scheme can solve the asynchronous hand-shaking problem in the WSI data transfer and allows us to design a simple WSI memory.

A circuit-level implementation and the actual layout issues remain for future studies.

REFERENCES

[1] C. C. Bernard, "Wafer-scale faces pessimism," *Electronics Week,* pp.49-53, Apr.1, 1985.

[2] J. F. McDonald, *et al.*, "The trials of wafer-scale integration," *IEEE Spectrum,* pp.32-39, Oct. 1984.

[3] F. B. Manning, "An Approach to Highly Integrated, Computer-Maintained Cellular Arrays," *IRE Trans. on Comput.,* vol.C-26, pp.536-552, June 1977.

[4] R. C. Aubusson and I. Catt, "Wafer-Scale Integraton: A Fault-tolerant Procedure," *IJSSC,* vol.SC-13, pp.339-344, June 1978.

[5] G. Chesley, "Addressable WSI: A Non-redundant Approach," *Computer Arch. News,* vol.15, pp.73-80, March 1987.

[6] M. Taniguchi, *et al.*, "Fully Boosted 64K Dynamic RAM with Automatic and Self-Refresh," *IJSSC,* vol.SC-16, pp.492-498, Oct. 1981.

[7] E. A. Reese, *et al.*, "A 4K x 8 Dynamic RAM with Self-Refresh," *IJSSC,* vol.SC-16, pp.479-487, Oct. 1981.

[8] R. I. Kung, *et al.*, "An 8K x 8 Dynamic RAM with Self-Refresh," *IJSSC,* vol.SC-17, pp.863-871, Oct. 1982.

[9] W. W. Plummer, "Asynchronous Arbiters," *IEEE Trans. on Comput.,* vol.C-21, pp.37-42, Jan. 1972.

[10] R. M. Keller, "Towards a Theory of Universal Speed-Independent Modules," *IEEE Trans. on Comput.,* vol.C-23, pp.21-33, Jan. 1974.

[11] L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits,* Chap.6, Addison-Wesley, 1985.

[12] C. D. Marcos, "Globally asynchronous Locally synchronous systems," *Ph.D. Dissertation,* Stanford Univ., 1985.

[13] J. C. Harden and N. R. Straden II, "Architectural Yield Optimization for WSI," *IEEE Trans. on Comput.,* vol.37, pp.88-110, Jan. 1988.

[14] L. Curren, "Wafer-Scale Integration Arrives In 'Disk Form'," *Electronics Design,* pp.51-54, Oct.26, 1989.

[15] T. Kilburn, *et al.*, "One-level Storage System," *IRE Trans. on Elec. Comput.,* pp.223-235, Apr. 1962.

[16] B. Furht and V. Milutinović, "A Survey of Microprocessor Architectures for Memory Management," *Computer,* pp.48-67, March 1987.

[17] J. P. Hayes, *Computer Architecture and Organization,* McGraw-Hill, Chap.5, 1988.

[18] J.E. Shemer and G. A. Shippey, "Statistical Analysis of Paged and Segmented Computer Systems," *IEEE Trans. on Elec. Comput.,* vol.EC-15, pp.855-863, Dec. 1966.

[19] A.V. Pohm and T. A. Smay, "Computer Memory Systems," *IEEE Computer,* pp.93-110, Oct. 1981.

[20] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI,* Addison-Wesley, Chap.5-6, 1990.

[21] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design,* Addison-Wesley, Chap.4, 1988.

[22] C. A. Mead and L. A. Conway, *Introduction to VLSI Systems,* Addison-Wesley, Reading, 1980.

[23] D. E. Knuth and G. S. Rao, "Activity in an Interleaved Memory," *IEEE Trans. on Comput.,* vol.C-24, pp.943-944, Sept. 1975.

[24] C. V. Ravi, "On the Bandwidth and Interference in Interleaved Memory Systems," *IEEE Trans. on Comput.,* vol.C-21, pp.899-901, Aug. 1972.

[25] K. V. Sastry and R. Y. Kain, "On the Performance of Certain Multiprocessor Computer Organizations," *IEEE Trans. on Comput.,* vol.C-24, pp.1066-1073, Nov. 1975.

[26] F. A. Briggs and E. S. Davidson, "Organization of Semiconductor Memories for Parallel-Pipelined Processors," *IEEE Trans. on Comput.,* vol.C-26, pp.162-169, Feb. 1977.

[27] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing,* McGraw-Hill, Chap.2, 1985.

[28] R. L. Petritz, "Current Status of Large Scale Integration Technology," *IJSSC,* vol.SC-2, pp.130-147, Dec. 1967.

[29] Y. Ueoka, *et al.*, "A Defect-tolerant design for Full-Wafer Memory LSI," *IJSSC,* vol.SC-19, pp.319-324, June 1984.

[30] Y. Egawa, *et al.*, "A 1-Mbit Full-Wafer Mos RAM," *IJSSC,* vol.SC-15, pp.677-685, August 1980.

[31] Y. Kitano, *et al.*, "A 4-Mbit Full-Wafer ROM," *IJSSC,* vol.SC-15, pp.686-693, June 1984.

[32] Technological Readout, "MEMORIES: One-Megabit EPROMs Invade Disk Territory," *Electronics Week,* pp.52-54, Apr.22, 1985.

[33] Y. Uchida, *et al.*, "A Low Power Resistive Load 64 Kbit CMOS RAM," *IJSSC,* vol.SC-17, pp.804-809, Oct. 1982.

[34] K. Ochii, *et al.*, "An Ultra Power 8K x 8-bit full CMOS RAM with a Six-Transistor Cell," *IJSSC,* vol.SC-17, pp.798-803, Oct. 1982.

[35] S. T. Flannagan, *et al.*, "Two 13-ns 64K CMOS SRAM's with Very Low Active Power and Improved Asynchronous Circuit Techniques," *IJSSC,* vol.SC-21, pp.692-703, Oct. 1986.

[36] K. C. Hardee and R. Sud, "A Fault-Tolerant 30 ns/375mW 16K x 1 NMOS Static RAM," *IJSSC,* vol.SC-16, pp.435-443, Oct. 1981.

[37] M. Yoshimoto, *et al.*, "A Divided Word-line Structure in the Static RAM and Its Application to a 64K Full CMOS RAM," *IJSSC,* vol.SC-18, Oct. 1983.

[38] R. Adams and G. Scavone, "Design a DRAM controller from the top down," *EDN,* pp.183-188, Apr.27, 1989.

[39] I. T. Ho, "Analysis of Transmission Lines on Integrated-Circuit Chips," *IJSSC,* vol.SC-2, pp.201-208, Dec. 1967.

APPENDIXES

APPENDIX A

THE EFFECT OF PAGE SIZE UPON AVERAGE ACCESS TIME

In the paged, segment virtual memory system, the addressing space is represented by a triple (S,P,W), where S is a segment address, P is a page address and W is a word address. The performance of a paged, segment memory system is related to the behavior of the segment program and (S,P,W) organization.

The analytic model of software and page accessing performance is presented in reference [18]. Using the terminology used in that reference, suppose that each segment S_i is subdivided into "page units," $\{U_1; U_2; \cdots\}_i$. Each page unit, U_k, is characterized by the number of pages it contains and the number of pages currently in its core. By definition, "the probability that a page referenced within U_k is among the last x distinct pages previously referenced by a procedure segment S_0" is the "Page Reference Distribution Function" $F_k(x,p)$, where p is the total number of pages in U_k referenced by S_0. When we assume "Random Sequential Segment" [18],

$$F_k(x,p) = \begin{cases} 0, & \text{if } x = 0 \\ 1 - \frac{1}{v}(1 - \frac{x}{p}), & \text{if } x > 0, \end{cases} \qquad (A.1)$$

where v = word size in each page.

In equation (A.1), the term $(1 - \frac{x}{p})$ means the probability that pages were not referenced after x pages of random search among p pages in U_k. When we assume a sequential word reference within a page boundary, the referencing of one of v words results in referencing entire pages. So, the term $\frac{1}{v}(1 - \frac{x}{p})$ indicates that some pages were not referenced after x pages of search. Hence, equation (A.1) is the distribution function of the referenced pages up to x pages random search with v words in each page.

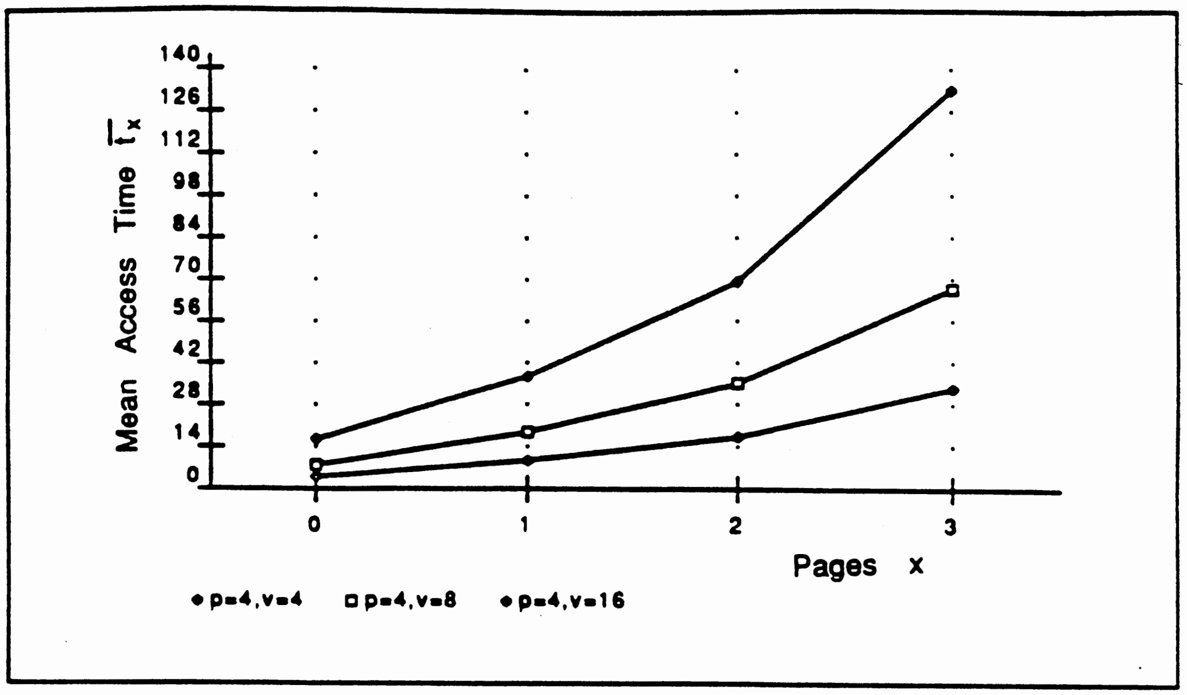
When we define the transition probability λ_x, paging unit U_k has the status from the population of x pages to $x+1$ pages with μ pages at a time during unit time Δt,

$$\lambda_x = \mu\{1 - F(x,p)\}\Delta t. \tag{A.2}$$

Then, the mean time to access x pages within this paging unit is given:

$$\overline{t_x} = \frac{1}{\lambda_0} + \frac{1}{\lambda_1} + \cdots + \frac{1}{\lambda_{x-1}}. \tag{A.3}$$

According to equation (A.3), the comparison between (v=4), (v=8) and (v=16) word size is made when (p=4) and (p=8), respectively. Figure 21 illustrates the effect of page word size upon the average access time. This result shows that when we arrange the page size small, the mean access time of x pages is also reduced, which means that a small page size is preferable.



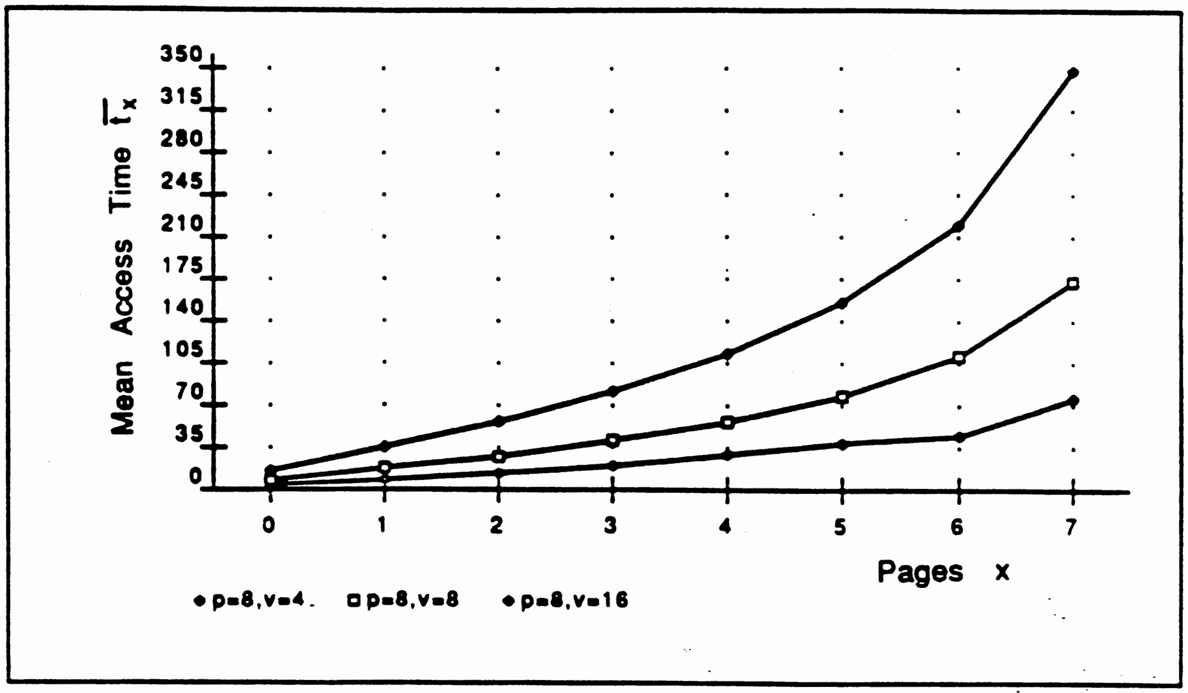


Figure 21. The Effect of Page Size to Mean Access Time

APPENDIX B

THE BANDWIDTH AS A FUNCTION OF NUMBER OF BANKS

When a memory system is interleaved in M ways and each memory line has N modules, this organization will be expressed as an (M,N) memory organization. The system performance will be analyzed by the memory bandwidth.

We assume that each memory cycle is composed of an Address Cycle (AC) and a Data Cycle (DC) of (AC,DC) $= (a, d)$. Also, each memory line is independent and there is no distinction between the READ and WRITE cycles. Following the concept and the terminology of reference [26], a discrete Markov model for a pipelined single processor with (M, N) memory organization is derived.

Definition B.1: A module state $\mu(t)$ at time t is

$$\mu(t) = \begin{cases} 0, & \text{if module is not busy} \\ r \in \{1, 2, \cdots, d-1\}, & \text{if module is busy,} \end{cases}$$

where r indicates that the current module was accepted how many Time Units (TU) ago.

Definition B.2: A line state $\lambda(t)$ at time t is the unordered set of all nonzero module states for all modules on the line currently being tested.

Because any two modules on the same line can not be accessed at the same time, any two elements in a line state should have a difference of at least a TUs.

Definition B.3: If there exists $r \in \lambda(t)$ such that $0 < r < a$, then $\lambda(t)$ is in a busy line state; otherwise it is in an idle line state.

Definition B.4: A line state is an accepted state if $\lambda(t)$ contains the element of $r = 1$.

Once a module is addressed, a line state goes into an accepted state on the very next TU. If the line is not accepted, we call it a nonaccepted state. According to this notation, there can only be two state transitions: from an accepted state to a nonaccepted state, or vice versa.

Result B.1: If no requests are accepted at time t by a line in state $\lambda(t)$, the next line state is

$$\lambda(t+1) = \{x \,\big|\, x - 1 \in \lambda(t) \textit{ and } x < d\}.$$

When a line is busy or when there is no request, a request can not be accepted.

Result B.2: If a request is accepted at time t, the next line state is

$$\lambda(t+1) = \{1\} \cup \{x \,\big|\, x - 1 \in \lambda(t) \textit{ and } x < d\}.$$

When a line is idle or there is more than one request, a request can be accepted.

When we use the memory organization (M,N) $=$ (l, m) and memory cycle (AC,DC) $=$ (a, d) $=$ (2,4), the Markov state graph of a single line is derived as in Figure 22 [26], where node with * is an accepted line state.

According to the state-graph of Figure 22, the state probability at time $t + \Delta t$ is described in terms of the state probability at time t:

$$\begin{bmatrix} \lambda_0(t+\Delta t) \\ \lambda_1(t+\Delta t) \\ \lambda_2(t+\Delta t) \\ \lambda_3(t+\Delta t) \end{bmatrix} =$$

$$\begin{bmatrix} 1 - b\Delta t + b\Delta t - a\Delta t & 0 & 0 & d\Delta t \\ a\Delta t & 1 - \Delta t & c\Delta t & c\Delta t \\ 0 & \Delta t & 1 - c\Delta t - d\Delta t & 0 \\ 0 & 0 & d\Delta t & 1 - c\Delta t - d\Delta t \end{bmatrix} \begin{bmatrix} \lambda_0(t) \\ \lambda_1(t) \\ \lambda_2(t) \\ \lambda_3(t) \end{bmatrix}$$

$$+\mathbf{o}(\Delta t), \qquad (B.1)$$

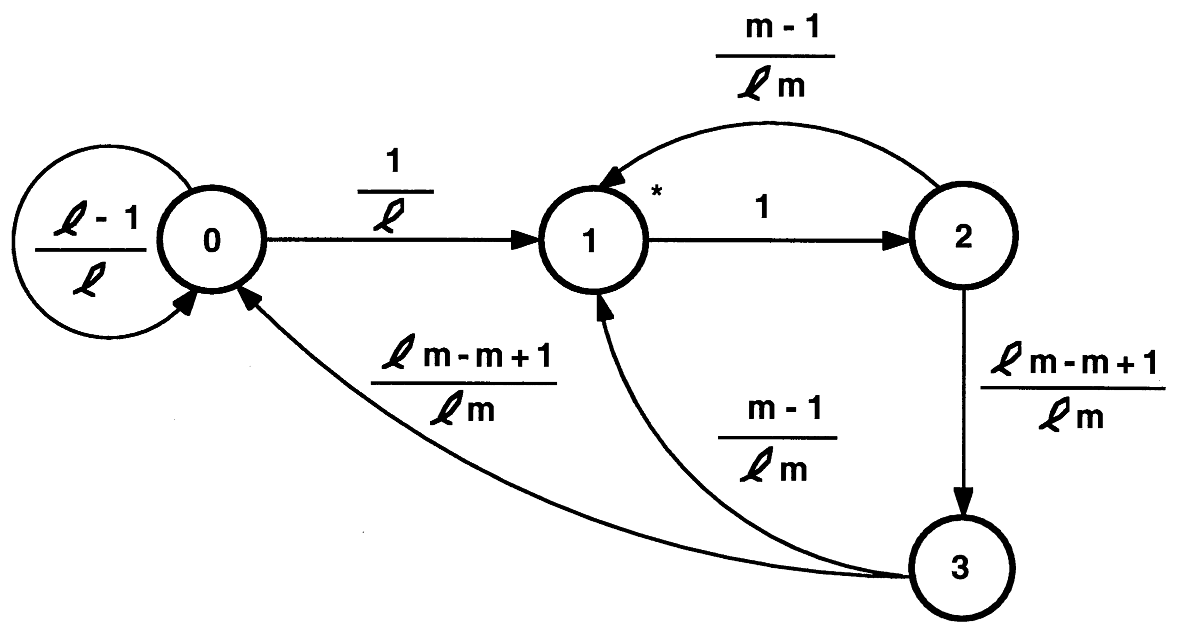


Figure 22. The State-graph of (a,d)=(2,4)

where $a = \frac{1}{l}$, $b = \frac{l-1}{l}$, $c = \frac{m-1}{lm}$, $d = \frac{lm-m+1}{lm}$ and $\mathbf{o}(\Delta t)$ denotes a quantity of order less than Δt.

When we rearrange equation (B.1), we get

$$\begin{bmatrix} \lambda_0(t+\Delta t) - \lambda_0(t) \\ \lambda_1(t+\Delta t) - \lambda_1(t) \\ \lambda_2(t+\Delta t) - \lambda_2(t) \\ \lambda_3(t+\Delta t) - \lambda_3(t) \end{bmatrix} = \mathbf{A} \begin{bmatrix} \lambda_0(t) \\ \lambda_1(t) \\ \lambda_2(t) \\ \lambda_3(t) \end{bmatrix} \Delta(t) + \mathbf{o}(t), \qquad (B.2)$$

where

$$\mathbf{A} = \begin{bmatrix} -a & 0 & 0 & d \\ a & -1 & c & c \\ 0 & 1 & -(c+d) & 0 \\ 0 & 0 & d & -(c+d) \end{bmatrix}. \qquad (B.3)$$

Dividing both sides by Δt and neglecting $\frac{\mathbf{o}(\Delta t)}{\Delta t}$ term, we obtain

$$\dot{\lambda}(t) = \mathbf{A}\lambda(t), \qquad (B.4)$$

where $\mathbf{A}$ is called the generator of the Markov chain.

Recall that

$$c + d = \frac{m-1}{lm} + \frac{lm-m+1}{lm} = 1.$$

Thus,

$$\mathbf{A} = \begin{bmatrix} -a & 0 & 0 & d \\ a & -1 & c & c \\ 0 & 1 & -1 & 0 \\ 0 & 0 & d & -1 \end{bmatrix}.$$

To obtain the steady-state solution, we set the time-variation in equation (B.4) to zero. So $\mathbf{A}\lambda = \mathbf{0}$ gives

$$-a\lambda_0 + d\lambda_3 = 0, \qquad (B.5)$$

$$a\lambda_0 - \lambda_1 + c\lambda_2 + d\lambda_3 = 0, \qquad (B.6)$$

$$\lambda_1 - \lambda_2 = 0, \qquad (B.7)$$

$$d\lambda_2 - \lambda_3 = 0. \qquad (B.8)$$

From equations (B.5), (B.7) and (B.8),

$$\begin{cases} \lambda_2 = \lambda_1 \\ \lambda_3 = d\lambda_2 = d\lambda_1 \\ \lambda_0 = \frac{d}{a}\lambda_3 = \frac{d^2}{a}\lambda_1 . \end{cases} \tag{B.9}$$

Invoking $\lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 = 1$, we get

$$\lambda_1 = \frac{1}{2 + d + \frac{d^2}{a}} = \frac{a}{d^2 + ad + 2a}. \tag{B.10}$$

The probability of acceptance, $P_A(a, d)$, is the probability that a certain line is accepted by the request to that line. By substituting $a = \frac{1}{l}$, $d = \frac{lm - m + 1}{lm}$ into equation (B.10) and rearranging, we have the accepted line state λ_1

$$\lambda_1 = P_A(2, 4) = \frac{lm^2}{(lm)^2 + lm^2 + 2lm - m + 1}. \tag{B.11}$$

The memory bandwidth, BW, can be expressed as l times the probability of single line acceptance because every line is independent.

$$BW = l \times P_A(2, 4) = \frac{(lm)^2}{(lm)^2 + lm^2 + 2lm - m + 1}. \tag{B.12}$$

Equation (B.12) tells that BW is encreasing with larger l.

VITA

Gang Hwa Lee

Candidate for the Degree of

Master of Science

Thesis: WSI MEMORY SYSTEM: ADDRESS-SELECTION APPROACH

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Daegu, Korea, March 22, 1959, the son of KeeChoon and KoJi Lee; married MyoungJin Lee in 1987; HanJae, son, was born in 1989.

Education: Graduated from KyoungBuk Senior High School, Daegu, Korea, in Febuary, 1977; received Bachelor of Science Degree in Electrical Engineering from Seoul National University at Seoul, Korea, in Febuary, 1981; completed requirements for the Master of Science Degree at Oklahoma State University in July, 1992.

Professional Experience: Research Engineer, GoldStar Electronics Co. Ltd., Jan., 1984, to May, 1988; manager of GoldStar Semiconductor Research Lab. at Seoul, Korea, June, 1988, to August, 1989; Research Assistant, Department of Electrical Engineering, Oklahoma State University, September, 1990, to March, 1991.