# **Module-III**

1. Discuss bus arbitration and its types in multiprocessor systems.
2. Illustrate sequential and weak consistency models.
3. Discuss any two mapping techniques.
4. Explain with diagram Backplane bus Specification
5. Explain the following terms associated with cache and memory architecture
6. Low order memory interleaving
7. Atomic v/s non atomic memory
8. Physical address cache vs virtual address cache
9. Memory bandwidth and fault tolerance

**Chapter 5 Bus, Cache and Shared Memory**

## 5.1 Bus Systems

* System bus of a computer operates on contention basis.
* Several active devices such as processors may request use of the bus at the same time.
* Only one of them can be granted access to bus at a time
* The Effective bandwidth available to each processor is inversely proportional to the number of processors contending for the bus.
* For this reason, most bus-based commercial multiprocessors have been small in size.
* The simplicity and low cost of a bus system made it attractive in building small multiprocessors ranging from 4 to 16 processors.

### 5.1.1 Backplane Bus Specification

* A backplane bus interconnects processors, data storage and peripheral devices in a tightly coupled hardware. bus without disturbing the internal activities of all the devices attached to the bus.
* The system bus must be designed to allow communication between devices on the devices on the
* Timing protocols must be established to arbitrate among multiple requests. Operational rules must be set to ensure orderly data transfers on the bus.
* Signal lines on the backplane are often functionally grouped into several buses as shown in Fig 5.1. Various functional boards are plugged into slots on the backplane. Each slot is provided with one or more connectors for inserting the boards as demonstrated by the vertical arrows.

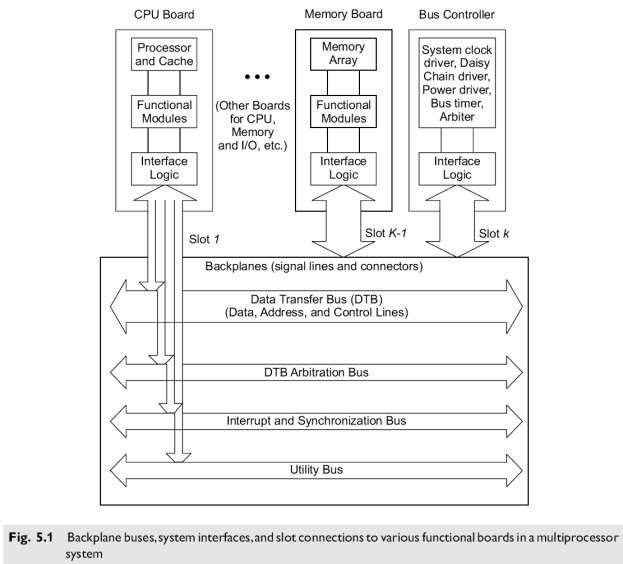
#### Data Transfer Bus (DTB)

* Data address and control lines form the data transfer bus (DTB) in VME bus.
* Address lines broadcast data and device address

– Proportional to log of address space size

* Data lines proportional to memory word length
* Control lines specify read/write, timing, and bus error conditions

#### Bus Arbitration and Control



* The process of assigning control of the DTB to a requester is called arbitration. Dedicated lines are reserved to coordinate the arbitration process among several requesters.
* The requester is called a master, and the receiving end is called a slave.
* Interrupt lines are used to handle interrupts, which are often prioritized. Dedicated lines may be used to synchronize parallel activities among the processor modules.
* Utility lines include signals that provide periodic timing (clocking) and coordinate the power-up and power-down sequences of the system.
* The backplane is made of signal lines and connectors.
* A special bus controller board is used to house the backplane control logic, such as the system clock driver, arbiter, bus timer, and power driver.

**Functional Modules** A functional module is a collection of electronic circuitry that resides on one functional board (Fig.

5.1) and works to achieve special bus control functions.

Special functional modules are introduced below:

* **Arbiter** is a functional module that accepts bus requests from the requester module and grants control of the DTB to one requester at a time.
* **Bus timer** measures the time each data transfer takes on the DTB and terminates the DTB cycle if a transfer takes too long.
* **Interrupter** module generates an interrupt request and provides status/ID information when an interrupt handler module requests it.
* **Location monitor** is a functional module that monitors data transfers over the DTB. A power monitor watches the status of the power source and signals when power becomes unstable.
* **System clock driver** is a module that provides a clock timing signal on the utility bus. In addition, board interface logic is needed to match the signal line impedance, the propagation time, and termination values between the backplane and the plug-in boards.

#### Physical Limitations

plugged into a single backplane.

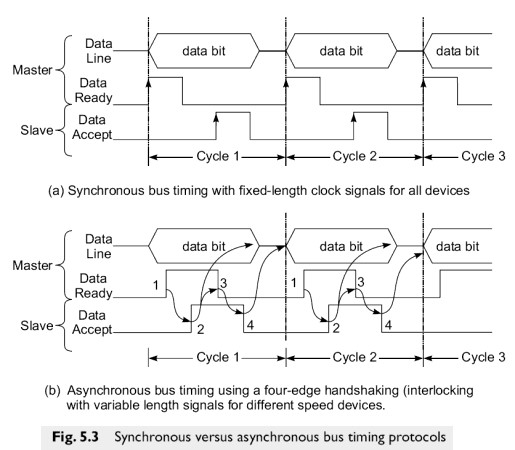
* Due to electrical, mechanical, and packaging limitations, only a limited number of boards can be
* Multiple backplane buses can be mounted on the same backplane chassis.
* The bus system is difficult to scale, mainly limited by packaging constraints.

### Timing Protocols

* Timing protocols are needed to synchronize master and slave operations.
* Figure 5.2 shows a typical timing sequence when information is transferred over a bus from a source to a destination.
* Most bus timing protocols implement such a sequence.

## Synchronous Timing

* All bus transaction steps take place at fixed clock edges as shown in Fig. 5.3a.
* The clock signals are broadcast to all potential masters and slaves.
* Clock cycle time determined by slowest device on bus
* Once the data becomes stabilized on the data lines, the master uses Data-ready pulse to initiate the transfer
* The Slave uses Data-accept pulse to signal completion of the information transfer.
* Simple, less circuitry, suitable for devices with relatively the same speed.



## Asynchronous Timing

* Based on handshaking or interlocking mechanism as shown in Fig. 5.3b.
* No fixed clock cycle is needed.
* The rising edge (1) of the data-ready signal from the master trioggers the rising (2) of the dataaccept signal from the slave.
* The second signal triggers the falling (3) of the data-ready clock and removal of data from the bus.
* The third signal triggers the trailing edge (4) of the data accept clock.
* This four-edge handshaking (interlocking) process is repeated until all the data is transferred. **Advantages:** Provides freedom of variable length clock signals for different speed devices
* No response time restrictions
* More flexible

**Disadvantage**: More complex and costly

## 5.1.3

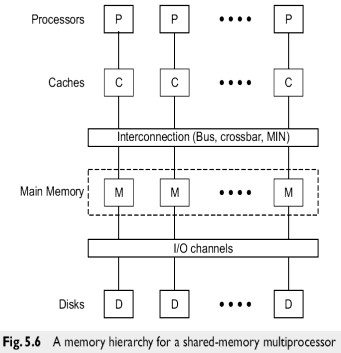
## 5.2 Cache Memory Organizations

Cache memory is the fast memory that lies between registers and RAM in memory hierarchy. It holds

recently used data and/or instructions.

### 5.2.1 Cache Addressing Models

* Most multiprocessor systems use private caches for each processor as shown in Fig. 5.6
* Have an interconnection network between caches and main memory • Caches can be addressed using either a Physical Address or Virtual Address.
* Two different cache design models are:
  + Physical address cache
  + Virtual address cache



### Physical address cache

* When cache is addressed by physical address it is called physical address cache. The cache is indexed and tagged with physical address.
* Cache lookup must occur after address translation in TLB or MMU. No aliasing is allowed so

that the address is always uniquely translated without confusion.

* After cache miss, load a block from main memory
* Use either write-back or write-through policy

**Advantages:**

* No cache flushing on a context switch
* No aliasing problem thus fewer cache bugs in OS kernel.

# • Simplistic design

• Requires little intervention from OS kernel

**Disadvantages:**

Slowdown in accessing the cache until the MMU/TLB finishes translating the address

## Virtual Address caches

* When a cache is indexed or tagged with virtual address it is called virtual address cache.
* In this model both cache and MMU translation or validation are done in parallel.
* The physical address generated by the MMU can be saved in tags for later write back but is not used during the cache lookup operations.

**Advantages:**

* do address translation only on a cache miss
* faster for hits because no address translation

# • More efficient access to cache

**Disadvantages:**

* Cache flushing on a context switch (example : local data segments will get an erroneous hit for virtual addresses already cached after changing virtual address space, if no cache flushing).
* Aliasing problem (several different virtual addresses cannot span the same physical addresses without being duplicated in cache).

## The Aliasing Problem

* The major problem associated with a virtual address cache is aliasing.
* Different logically addressed data have the same index/tag in the cache
* Confusion if two or more processors access the same physical cache location
* Flush cache when aliasing occurs, but leads to slowdown
* Apply special tagging with a process key or with a physical address

## 5.2.2

## The ultimate performance depends upon cache access patterns, organization, and management policy

* Blocks in caches are called **block frames**, and blocks in main memory are called **blocks**

### Direct Mapping Cache

* Direct mapping of **n/m = 2s-r** memory blocks to one block frame in the cache
* Placement is by using modulo-m function. Block Bj is mapped to block frame Bi

**Bj**  **Bi if i=j mod m**

* There is a unique block frame Bi that each B j can load into.
* There is no way to implement a block replacement policy.
* This Direct mapping is very rigid but is the simplest cache organization to implement.

The memory address is divided into 3 fields:

* The lower w bits specify the word offset within each block.
* The upper s bits specify the block address in main memory – The leftmost (s-r) bits specify the tag to be matched

The block field (r bits) is used to implement the (modulo-m) placement, where **m=2r**

Once the block **Bi**  is uniquely identified by this field, the tag associated with the addressed block is compared with the tag in the memory address.

**Advantages**

* Simple hardware
* No page replacement policy
* Lower cost
* Higher speed

**Disadvantages**

* Rigid mapping
* Poorer hit ratio
* Prohibits parallel virtual address translation
* Use larger cache size with more block frames to avoid contention

### Sector Mapping Cache

* Partition both the cache and main memory into fixed size sectors. Then use fully associative search ie., each sector can be placed in any of the available sector frames.
* The memory requests are destined for blocks, not for sectors.
* This can be filtered out by comparing the sector tag in the memory address with all sector tags using a fully associative search.
* If a matched sector frame is found (a cache hit), the block field is used to locate the desired block within the sector frame.
* If a cache miss occurs, the missing block is fetched from the main memory and brought into a

congruent block frame in available sector.

* That is the *i*th block in a sector must be placed into the ith block frame in a destined sector frame.
* Attach a valid bit to each block frame to indicate whether the block is valid or invalid.
* When the contents of the block frame are replaced from a new sector, the remaining block frames in the same sector are marked invalid. Only the block frames from the most recently referenced sector are marked valid for reference.

**Advantages:**

* Flexible to implement various block replacement algorithms
* Economical to perform a fully associative search a limited number of sector tags.
* Sector partitioning offers more freedom in grouping cache lines at both ends of the mapping.

# 

.

### 5.3 Shared Memory Organizations

Memory interleaving provides a higher bandwidth for pipelined access of continuous memory locations.

Methods for allocating and deallocating main memory to multiple user programs are considered for optimizing memory utilization.

#### 5.3.1 Interleaved Memory Organization

* In order to close up the speed gap between the CPU/cache and main memory built with RAM modules, an *interleaving* technique is presented below which allows pipelined access of the parallel memory modules.
* The memory design goal is to broaden the *effective memory bandwidth* so that more memory words
* The ultimate purpose is to match the memory bandwidth with the bus bandwidth and with the can be accessed per unit time.

processor bandwidth.

##### Memory Interleaving

* The main memory is built with multiple modules.
* These memory modules are connected to a system bus or a switching network to which other resources such as processors or I/O devices are also connected.
* Once presented with a memory address, each memory module returns with one word per cycle.
* It is possible to present different addresses to different memory modules so that parallel access of

multiple words can be done simultaneously or in a pipelined fashion.

Consider a main memory formed with **m = 2a**memory modules, each containing **w = 2b**words of memory cells. The total memory capacity is **m.w = 2a+b** words.

These memory words are assigned linear addresses. Different ways of assigning linear addresses result in different memory organizations.

Besides random access, the main memory is often block-accessed at consecutive addresses.

Figure 5.15 shows two address formats for memory interleaving.

* Low-order interleaving
* High-order interleaving

##### Low-order interleaving

* Low-order interleaving spreads contiguous memory locations across the  **m** modules horizontally (Fig. 5.15a). • This implies that the low-order **a**bits of the memory address are used to identify the memory

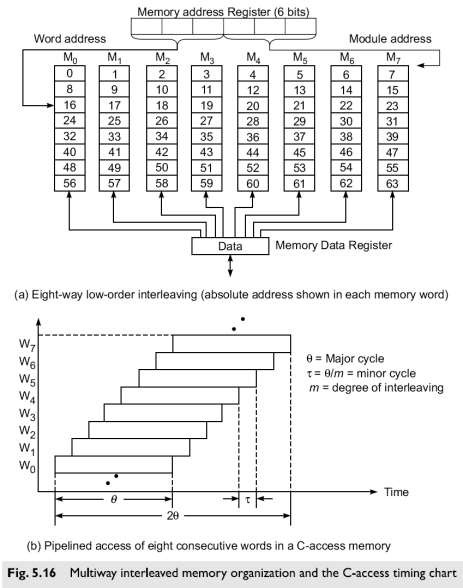
module.

* The high-order **b**bits are the word addresses (displacement) within each module.
* Note that the same word address is applied to all memory modules simultaneously. A module address decoder is used to distribute module addresses.

##### High-order interleaving

* High-order interleaving uses the high-order **a** bits as the module address and the low-order **b**bits as the word address within each module (Fig. 5.15b).
* Contiguous memory locations are thus assigned to the same memory module. In each memory cycle, only one word is accessed from each module.
* Thus the high-order interleaving cannot support block access of contiguous locations.

##### Pipelined Memory Access



* Access of the **m**memory modules can be overlapped in a pipelined fashion.
* For this purpose, the memory cycle (called the *major cycle)* is subdivided into **m** minor cycles.
* An eight-way interleaved memory (with m=8 and w=8 and thus a=b=3) is shown in Fig. 5.16a.
* Let be the major cycle and the minor cycle. These two cycle times are related as follows:

 **=** **/m**

m=degree of interleaving

=total time to complete access of one word

=actual time to produce one word

Total block access time is **2**

Effective access time of each word is 

* The timing of the pipelined access of the 8 contiguous memory words is shown in Fig. 5.16b. • This type of concurrent access of contiguous words has been called a C-access memory scheme.

### 5.3.3 Memory Allocation Schemes

#### Virtual Memory

* **Purpose**: Allows multiple software processes to use main memory simultaneously.
* **Memory Manager**: Handles the swapping of processes in and out of memory, monitors available memory, and decides which processes to keep or remove.

#### Allocation Policies

* **Memory Swapping**: Moving data blocks between different memory levels.
* **Nonpreemptive Allocation**:
  + Swaps out some allocated processes if memory is full.
  + Easier to implement, but less efficient.
* **Preemptive Allocation**:
  + Can interrupt and swap out an executing process.
  + More complex and expensive but more flexible.
* **Local Allocation**:
  + Focuses on the working set of the process causing a fault.
  + Used by most computers.
* **Global Allocation**:
  + Considers the working sets of all processes when deciding which to swap.

#### Swapping Systems

* **Process-Level Swapping**: Swaps entire processes.
* **Swap Device**: Part of the disk used for temporary storage of swapped data.
* **Swap Space**: Disk portion reserved for swapping.
* **Types**: Systems may swap entire processes or just necessary pages.

#### Swapping in UNIX

* **Triggers for Swapping**:
  + Creating a child process.
  + Increasing process address space.
  + Increased stack space demand.
  + Returning previously swapped out processes.
* **Special Process 0**: Handles swapping.

#### Demand Paging Systems

* **Function**: Transfers only the needed pages between main memory and swap device.
* **Benefits**:
  + Allows process address space to exceed physical memory size.
  + Dynamically accommodates many processes.

#### Working Sets

* **Definition**: Set of pages a process used in the last 'n' memory references.
* **Usage**: Keeps only the working sets of active processes in memory.

### 5.4

##### 