MULTIPROCESSORS

- Characteristics of Multiprocessors
- Interconnection Structures
- Interprocessor Arbitration
- Interprocessor Communication and Synchronization
- Cache Coherence
Characteristics of Multiprocessor systems

A multiprocessor system is an interconnection of two or more CPU’s with memory and input-output equipment.

Multiprocessors system are classified as multiple instruction stream, multiple data stream systems (MIMD).

There exists a distinction between multiprocessor and multicomputers that though both support concurrent operations. In multicomputers several autonomous computers are connected through a network and they may or may not communicate but in a multiprocessor system there is a single OS Control that provides interaction between processors and all the components of the system to cooperate in the solution of the problem.

VLSI circuit technology has reduced the cost of the computers to such a low Level that the concept of applying multiple processors to meet system performance requirements has become an attractive design possibility.
Benefits of Multiprocessing:

1. Multiprocessing increases the reliability of the system so that a failure or error in one part has limited effect on the rest of the system. If a fault causes one processor to fail, a second processor can be assigned to perform the functions of the disabled one.

2. Improved System performance. System derives high performance from the fact that computations can proceed in parallel in one of the two ways:
   a) Multiple independent jobs can be made to operate in parallel.
   b) A single job can be partitioned into multiple parallel tasks. This can be achieved in two ways:
      • The user explicitly declares that the tasks of the program be executed in parallel
      • The compiler provided with multiprocessor s/w that can automatically detect parallelism in program. Actually it checks for Data dependency.
COUPLING OF PROCESSORS

Tightly Coupled System/Shared Memory
- Tasks and/or processors communicate in a highly synchronized fashion.
- Communicates through a common global shared memory.
- Shared memory system. This doesn’t preclude each processor from having its own local memory (cache memory).

Loosely Coupled System/Distributed Memory
- Tasks or processors do not communicate in a synchronized fashion.
- Communicates by message passing packets consisting of an address, the data content, and some error detection code.
- Overhead for data exchange is high.
- Distributed memory system.

Loosely coupled systems are more efficient when the interaction between tasks is minimal, whereas tightly coupled system can tolerate a higher degree of interaction between tasks.
Granularity of Parallelism

Coarse-grain
- A task is broken into a handful of pieces, each of which is executed by a powerful processor
- Processors may be heterogeneous
- Computation/communication ratio is very high

Medium-grain
- Tens to few thousands of pieces
- Processors typically run the same code
- Computation/communication ratio is often hundreds or more

Fine-grain
- Thousands to perhaps millions of small pieces, executed by very small, simple processors or through pipelines
- Processors typically have instructions broadcasted to them
- Compute/communicate ratio often near unity
MEMORY

Shared (Global) Memory
- A Global Memory Space accessible by all processors
- Processors may also have some local memory

Distributed (Local, Message-Passing) Memory
- All memory units are associated with processors
- To retrieve information from another processor's memory a message must be sent there

Uniform Memory
- All processors take the same time to reach all memory locations

Nonuniform (NUMA) Memory
- Memory access is not uniform
Shared Memory Multiprocessors

Characteristics

All processors have equally direct access to one large memory address space

Limitations

Memory access latency; Hot spot problem

Interconnection Network

Buses, Multistage IN, Crossbar Switch
MESSAGE-PASSING MULTIPROCESSORS

Characteristics:
- Interconnected computers
- Each processor has its own memory, and communicate via message-passing

Limitations:
- Communication overhead; Hard to programming
The interconnection between the components of a multiprocessor system can have different physical configurations depending on the number of transfer paths that are available between the processors and memory in a shared memory system and among the processing elements in a loosely coupled system.

Some of the schemes are as:
- Time-Shared Common Bus
- Multiport Memory
- Crossbar Switch
- Multistage Switching Network
- Hypercube System

**Time shared common Bus**

All processors (and memory) are connected to a common bus or busses
- Memory access is fairly uniform, but not very scalable
BUS

- A collection of signal lines that carry module-to-module communication
- Data highways connecting several digital system elements

Operations of Bus

M3 wishes to communicate with S5

[1] M3 sends signals (address) on the bus that causes S5 to respond
[2] M3 sends data to S5 or S5 sends data to M3 (determined by the command line)

Master Device: Device that initiates and controls the communication
Slave Device: Responding device

Multiple-master buses
- Bus conflict
- Need bus arbitration
SYSTEM BUS STRUCTURE FOR MULTIPROCESSORS

SYSTEM BUS

Local Bus

Common Shared Memory

System Bus Controller

CPU

IOP

Local Memory

System Bus Controller

CPU

IOP

Local Memory

Local Bus

Local Bus
**MULTIPOINT MEMORY**

Multiport Memory Module
- Each port serves a CPU

Memory Module Control Logic
- Each memory module has control logic
- Resolve memory module conflicts Fixed priority among CPUs

Advantages
- Multiple paths -> high transfer rate

Disadvantages
- Memory control logic
- Large number of cables and connections
CROSSBAR SWITCH

- Each switch point has control logic to set up the transfer path between a processor and a memory.

- It also resolves the multiple requests for access to the same memory on the predetermined priority basis.

- Though this organization supports simultaneous transfers from all memory modules because there is a separate path associated with each module. The H/w required to implement the switch can become quite large and complex.

Block Diagram of Crossbar Switch

Memory Module

- Data
- Address
- R/W
- Memory enable

Multiplexers and arbitration logic

- Data, address, and control from CPU 1
- Data, address, and control from CPU 2
- Data, address, and control from CPU 3
- Data, address, and control from CPU 4

Memory modules

CPU1

CPU2

CPU3

CPU4
MULTISTAGE SWITCHING NETWORK

Interstage Switch

A connected to 0

B connected to 0

A connected to 1

B connected to 1
MULTISTAGE INTERCONNECTION NETWORK

Binary Tree with 2 x 2 Switches

Some requests cannot be satisfied simultaneously.
For example, if P1 is connected to 000 through 001, P2 can be connected to only one of the destinations i.e., 100 through 111.

8x8 Omega Switching Network
n-dimensional hypercube (binary n-cube)

- $p = 2^n$
- processors are conceptually on the corners of a $n$-dimensional hypercube, and each is directly connected to the $n$ neighboring nodes
- Degree = $n$
- Routing Procedure: source 010, destination 001
  Ex-or :011. So data is transmitted on y axis and then on Z axis i.e. 010 to 000 and then 000 to 001

One-cube

Two-cube

Three-cube
INTERPROCESSOR ARBITRATION

- Only one of CPU, IOP, and Memory can be granted to use the bus at a time
- Arbitration mechanism is needed to handle multiple requests to the shared resources to resolve multiple contention.

SYSTEM BUS:
A bus that connects the major components such as CPU’s, IOP’s and memory

A typical System bus consists of 100 signal lines divided into three functional groups: data, address and control lines. In addition there are power distribution lines to the components.

<table>
<thead>
<tr>
<th>e.g. IEEE standard 796 bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 86 lines</td>
</tr>
<tr>
<td>Data: 16 (multiple of 8)</td>
</tr>
<tr>
<td>Address: 24</td>
</tr>
<tr>
<td>Control: 26</td>
</tr>
<tr>
<td>Power: 20</td>
</tr>
</tbody>
</table>
SYNCHRONOUS & ASYNCHRONOUS DATA TRANSFER

Synchronous Bus
Each data item is transferred over a time slice known to both source and destination unit
- Common clock source
- Or separate clock and synchronization signal is transmitted periodically to synchronize the clocks in the system

Asynchronous Bus

* Each data item is transferred by *Handshake* mechanism
  - Unit that transmits the data transmits a control signal that indicates the presence of data
  - Unit that receiving the data responds with another control signal to acknowledge the receipt of the data

* Strobe pulse - supplied by one of the units to indicate to the other unit when the data transfer has to occur
### BUS SIGNALS

**Bus signal allocation**
- address
- data
- control
- arbitration
- interrupt
- timing
- power, ground

#### IEEE Standard 796 Multibus Signals

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Signal(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data and address</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data lines (16 lines)</td>
<td></td>
<td>DATA0 - DATA15</td>
</tr>
<tr>
<td>Address lines (24 lines)</td>
<td></td>
<td>ADRS0 - ADRS23</td>
</tr>
<tr>
<td><strong>Data transfer</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory read</td>
<td></td>
<td>MRDC</td>
</tr>
<tr>
<td>Memory write</td>
<td></td>
<td>MWTC</td>
</tr>
<tr>
<td>IO read</td>
<td></td>
<td>IORC</td>
</tr>
<tr>
<td>IO write</td>
<td></td>
<td>IOWC</td>
</tr>
<tr>
<td>Transfer acknowledge</td>
<td></td>
<td>TACK (XACK)</td>
</tr>
<tr>
<td><strong>Interrupt control</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt request</td>
<td></td>
<td>INT0 - INT7</td>
</tr>
<tr>
<td>interrupt acknowledge</td>
<td></td>
<td>INTA</td>
</tr>
</tbody>
</table>
IEEE Standard 796 Multibus Signals (Cont’d)

**Miscellaneous control**
- Master clock: CCLK
- System initialization: INIT
- Byte high enable: BHEN
- Memory inhibit (2 lines): INH1 - INH2
- Bus lock: LOCK

**Bus arbitration**
- Bus request: BREQ
- Common bus request: CBRQ
- Bus busy: BUSY
- Bus clock: BCLK
- Bus priority in: BPRN
- Bus priority out: BPRO

**Power and ground (20 lines)**
INTERPROCESSOR ARBITRATION STATIC ARBITRATION

Serial Arbitration Procedure

Parallel Arbitration Procedure
INTERPROCESSOR ARBITRATION DYNAMIC ARBITRATION

Priorities of the units can be dynamically changeable while the system is in operation

**Time Slice**
Fixed length time slice is given sequentially to each processor, round-robin fashion

**Polling**
Unit address polling - Bus controller advances the address to identify the requesting unit. When processor that requires the access recognizes its address, it activates the bus busy line and then accesses the bus. After a number of bus cycles, the polling continues by choosing a different processor.

**LRU**
The least recently used algorithm gives the highest priority to the requesting device that has not used bus for the longest interval.

**FIFO**
The first come first serve scheme requests are served in the order received. The bus controller here maintains a queue data structure.

**Rotating Daisy Chain**
Conventional Daisy Chain - Highest priority to the nearest unit to the bus controller

Rotating Daisy Chain – The PO output of the last device is connected to the PI of the first one. Highest priority to the unit that is nearest to the unit that has most recently accessed the bus (it becomes the bus controller)
Interprocessor Communication

Sending Processor

Communication Area
- Mark
- Receiver(s)
- Message

Receiving Processor

Receiving Processor

Receiving Processor

Receiving Processor

Sending Processor

Instruction

Shared Memory

Interrupt

Computer Organization
INTERPROCESSOR SYNCHRONIZATION

Synchronization
Communication of control information between processors
- To enforce the correct sequence of processes
- To ensure mutually exclusive access to shared writable data

Hardware Implementation

Mutual Exclusion with a Semaphore

Mutual Exclusion
- One processor to exclude or lock out access to shared resource by other processors when it is in a Critical Section
- Critical Section is a program sequence that, once begun, must complete execution before another processor accesses the same shared resource

Semaphore
- A binary variable
- 1: A processor is executing a critical section, that not available to other processors
  0: Available to any requesting processor
- Software controlled Flag that is stored in memory that all processors can be access
### SEMAPHORE

#### Testing and Setting the Semaphore

- Avoid two or more processors test or set the same semaphore
- May cause two or more processors enter the same critical section at the same time
- Must be implemented with an indivisible operation

\[
\begin{align*}
R &\gets M[\text{SEM}] \quad / \text{ Test semaphore } / \\
M[\text{SEM}] &\gets 1 \quad / \text{ Set semaphore } /
\end{align*}
\]

These are being done while *locked*, so that other processors cannot test and set while current processor is being executing these instructions.

If R=1, another processor is executing the critical section, the processor executed this instruction does not access the shared memory.

If R=0, available for access, set the semaphore to 1 and access

The last instruction in the program must clear the semaphore.
Cache Coherence

Caches are Coherent

Cache Incoherency in Write Through Policy

Cache Incoherency in Write Back Policy

Computer Organization

Computer Architectures Lab
MAINTAINING CACHE COHERENCY

Shared Cache
- Disallow private cache
- Access time delay

Software Approaches
* Read-Only Data are Cacheable
  - Private Cache is for Read-Only data
  - Shared Writable Data are not cacheable
  - Compiler tags data as cacheable and noncacheable
  - Degrade performance due to software overhead

* Centralized Global Table
  - Status of each memory block is maintained in CGT: RO(Read-Only); RW(Read and Write)
  - All caches can have copies of RO blocks
  - Only one cache can have a copy of RW block

Hardware Approaches
* Snoopy Cache Controller
  - Cache Controllers monitor all the bus requests from CPUs and IOPs
  - All caches attached to the bus monitor the write operations
  - When a word in a cache is written, memory is also updated (write through)
  - Local snoopy controllers in all other caches check their memory to determine if they have a copy of that word; If they have, that location is marked invalid (future reference to this location causes cache miss)