Chapter-9

Introduction to parallel Processing

Parallelism in Uniprocessor system

A computer system achieves parallelism when it performs two or more tasks simultaneously. A system that processes two different instructions simultaneously could be considered to perform parallel processing. If a system that performs different operation on same instruction is not considered as parallel processing.

Parallelism in uniprocessor is achieved by instruction pipelining. By overlapping the fetching, decoding, executions of instructions, a RISC or CISC processors with an instruction pipeline exhibits parallelism. Arithmetic pipelining is another example of parallelism in uniprocessor. IOP is another example of parallelism in uniprocessor where IOP performs data transfer whereas CPU executes instruction. DMA when operating in transparent mode supports parallelism.

Characteristics of Multiprocessor

A multiprocessor system is an interconnection of two or more CPUs with memory and input-output equipment. Multiprocessor can mean either central processing unit (CPU) or an input-output processor (IOP). For multiprocessor system IOP must have computational facility like CPU in a single CPU system. Multiprocessor are classified as (MIMD) multiple instruction stream, multiple data stream.

A multiprocessor system is controlled by one operating system that provides interaction between processor and all the components of the system co-operate in the solution of a problem. Multiprocessing improves the reliability of the system so that a failure error in one part has a limited effect on the rest of the system. If a fault occurs in one process the other can be assigned to perform that task. Multiprocessor organization improved the system performance. Here computation can proceed in parallel in one of two ways:

1. Multiple independent jobs can make to operate in parallel.
2. A single job can be partitioned into multiple parallel tasks.

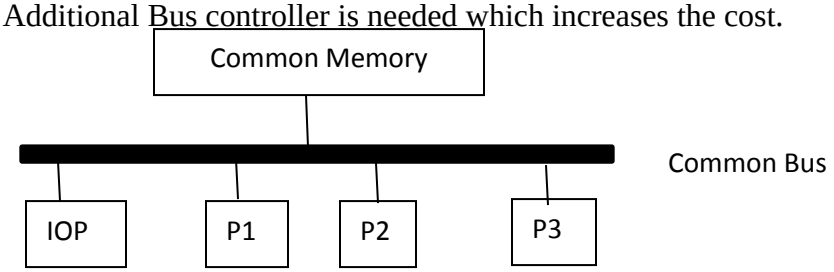
Multiprocessor can improve performance by decomposing a program into parallel executable tasks. It can be achieved in two ways:

1. User can explicitly define that certain task of program be executed in parallel.
2. Provide a compiler with multiprocessor software that can detect parallelism in user's program.

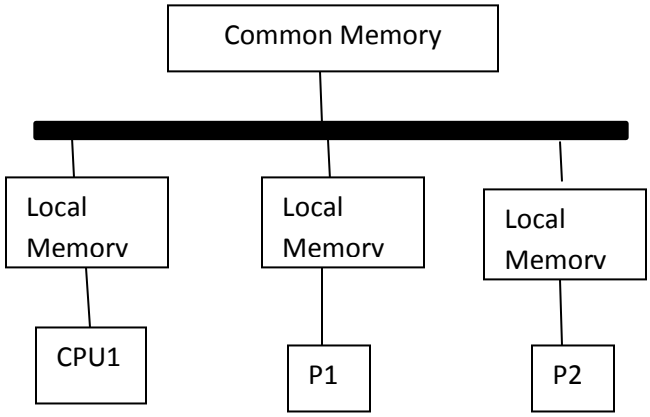
Enslow's Classification

Multiprocessors are classified according to memory organization:

1. Tightly Coupled- Multiprocessor system with common shared memory is classified as shared memory or tightly coupled multiprocessor. Most commercial tightly coupled multiprocessor provides cache memory with each CPU. In addition, there is a common memory that all CPU can access. All processors are interconnected through common bus. Bus controller detects data congestion in common bus. No local memory only common memory is shared by all processors.
   Advantage: Number of bus is less. Number of connection path is less.
   Disadvantage: Additional Bus controller is needed which increases the cost.



2. Loosely Coupled- Multiprocessor system with private local memory distributed or loosely coupled system i.e. each processor has its local memory. The processors are tied together by a switch scheme designed to route information from one processor to another through a message passing scheme. Here program and data transfer are in the form of packets. The packets are address to a specific processor or taken by first available processor. This system is efficient where interaction between tasks is minimal whereas in tightly coupled system interaction between tasks is high.



Flynn's Classification

1. SISD (Single instruction single data): SISD machine consists of single CPU executing individual instructions on individual data values. It provides internal parallelism by use of pipelining but then also it is classified as SISD.

2. MIMD (Multiple instructions multiple data): Multiprocessors or multi-computers are usually MIMD machines. The processors execute multiple instructions simultaneously. Thus each processor must include its control memory. the processors can work on same task or totally different tasks. MIMD computers are better for specific tasks depending upon topology and architecture.
3. SIMD (Single instruction multiple data): SIMD machine executes a single instruction on multiple data values simultaneously using many processors. Since one instruction proceeds at a given time, it is not necessary to fetch and decode instruction. Instead a single control unit handles this task for all processors within the SIMD computers.
4. MISD (Multiple instruction single data): MISD does not exist practically to implement. It is included in taxonomy.

Inter connection Structures

The interconnection between the different components of the multiprocessor system has different configurations. Depending on communication path between multiprocessor it is divided into following types.

1. Time shared common bus
2. Multiport memory
3. Crossbar switch
4. Multistage switching n/w
5. Hypercube system

Time shared common bus

A common bus multiprocessor system consists of a number of processor connected through a common path to a memory unit. Only one processor can communicate with the memory at any given time. Transfer operations are conducted by the processor that is in control of the bus to initiate transfer. A command is issued to inform the destination unit what operation is to be performed. The receiving unit recognizes its address in the bus and responds to the control signals from the sender. After then transfer is initiated. Here transfer conflicts may occur as one common bus is shared by all processors. These conflicts can be resolved by incorporating a bus controller that establishes priorities among the requesting units.

In this system from single bus only one processor operates at a time. All other processor is busy with internal operations or must be idle waiting for the bus. Thus overall transfer rate is limited by the speed of single path. Thus dual bus structure can be implemented. Here we have number of local buses. Each connected with its local memory. Each local bus is connected to CPU, IOP. A system bus controller links with common system bus. here only one processor can communicate with common memory and all other processor through the system bus at any time.

Multiport memory

A multiport memory system employs separate buses between each memory module and each CPU. Each processor bus I connected to each memory module. A processor bus consists of the address, data and control lines required to communicate with memory. The module must have internal control logic to determine which port will have access to memory at any given time. Memory access conflicts are resolved by assigning fixed priorities to each memory port. Here priority is assigned by their physical port. CPU1 have priority over CPU1, CPU2 have over CPU3 and so on.

Advantage- High transfer rate that can be achieved because of the multiple paths between processor and memory.

Crossbar Switch

The crossbar switch organizations consist of a number of cross points that are placed at intersections between processor bus and memory module paths. Each switch (cross points) determines the path from a processor to memory module. It has control logic to set up the transfer between a processor and memory. It examines the address that is placed in the bus to determine whether its particular module is being addressed. It also resolves multiple requests for access to same memory module in a predetermined priority basis.

This organization supports simultaneous transfers from all memory modules because there is a separate path associated with each module.

Multistage switching Network

The basic component of a multistage network is a two-input, two-output interchange switch. In the figure A and B are two inputs and outputs are 0 & 1. There are control signals associated with the switch that establishes interconnection between input and output terminals. A & B can have access to both the terminals but one at a time i.e. if both requests the same output terminal, one of them is blocked and the other gets connected.

Using this 2*2 switch block a multistage network is built to communicate between a number of sources and destinations.

Here two processor p1 and p2 are connected to eight memory modules marked in binary from 000 through 111. The path from source to destination is determined from the binary bits to the destination number.

Many different topologies have been proposed for multistage switching network to control processor memory communication in tightly coupled and communication between processing elements in a loosely coupled system. Omega network is one such topology. In this case there is exactly one path from each source to any particular destination. A particular request is initiated in the switching network by the source which sends a 3-bit pattern representing the destination number. Each level examines a different bit to determine the 2*2 switch setting. Level1 examines MSB, Level2 examines the middle bit and Level 3 examines the LSB. When request arrives on either input of the 2*2 switch, it is routed to s 0 upper outputs if the selected bit is 0 or to the lower output if the bit is 1.

Hypercube interconnection

The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of $N=2^n$ processor interconnected in an n-dimensional binary cube. Each processor forms a node of the cube. Each processor has direct communication path to n other neighbor processors. These paths correspond to edges of the cube. There are 2^n, n-bit address that can be assigned to processor. Each processor address differs from its neighbors by one bit position.

Here one cube has 2 nodes and a single path. Two-cube has 4 nodes interconnected as a square. A three-cube structure has eight nodes interconnected as a hypercube.

Routing message through an n-cube structure may take from one to n links from a source node to a destination node. For e.g. in a 3-cube 000 can directly communicate with 001. To with 011 two links (000-001, 001-011 or 000-010,010-011). A routing procedure can be developed by computing the exclusive-or of the source node with the destination node address.

Cache Coherence

In a shared memory multiprocessor system, all the processors share a common memory. Each processor may have a local memory, part or all of which may be cache. Having separate cache minimizes the average access time. The same information resides in caches or main memory. To execute memory operation correctly, the multiple copies must be kept identical. This requirement creates cache coherence problem. Let us see from example

Fig: Cache configuration after a load on x.

Here X is laoaded in three processors P1,P2,P3 thus they are also loaded in their private caches. X contains a value of 52. If one of the processors performs a store to X, the copies in the caches become inconsistent. A load by the other processor will not return the latest value.

Fig: write through cache policy

Fig: Write Back cache policy

Solution for cache coherence problem

Software Solutions:

1. Disallow private caches for each processor and have a shared cache memory. This method violates the closeness of CPU to cache and increases average memory access time.
2. Only non-shared and read only data to be stored in cache. The items are called cacheable. Shared writable data are non-cacheable. The compiler must tag data as cacheable and non-cacheable and the system hardware only stores cacheable data. This restricts the type of data stored in caches and introduces extra software overhead that may degrade performance.

Hardware Solutions:

In hardware solutions the cache controller is specially design to allow it to monitor all bus requests from CPU and IOPs. All caches attaché to the bus constantly monitor the network for possible write operations. Depending upon the method used, they must then either update or

invalidate their own cache copies when a match is detected. The bus controller that monitors this operation is called snoopy cache controller. Using write through policy, snoopy controller watch the bus for memory store operations. When a word in cache is updated, the corresponding main memory is also updated. The local snoopy controller in all other cache checks their memory to determine if they have a copy of word that has been overwritten. If a copy exists, that local is marked as invalid.

Vector processing

A conventional computer cannot solve the entire computational problem. This problem requires a vast number of computations that will take conventional computer days or even weeks to complete. These problems can be formulated in terms of vectors and matrices that lend themselves to vector processing. Computers with vector processing capabilities are in demand in special applications. Following are the specialized area of vector processing:

- Long-range weather forecasting
- Petroleum explorations
- Seismic data analysis
- Medical diagnosis
- Aerodynamics and space flight simulations
- Artificial intelligence and expert systems
- Mapping and human genome
- Image processing

Vector operations

Scientific problem that requires arithmetic operations on large arrays of numbers are formulated as vectors and matrices of floating-point numbers. A vector is an ordered set of one-dimensional array of data items. A vector V of n length is represented as a row vector $V=[V_1\ V_2\ V_3\ \ldots\ldots\ V_n]$. It may be represented as a column vector if a data items are listed in a column. Operations on vectors must be broken down into single computations with subscripted variables. The element V_i of vector V is written as V(I) and the index I refers to a memory address or register where the number is stored. Lets examine the difference between conventional scalar processor and vector processor

```
        DO 20 I=1,100
20      C (I)=B(I)+A(I)
```

Above example is Fortran DO loop.

This is a program for adding two vectors A and B of length 100 to produce a vector C.

```
        Initialize I=0
20      Read A(I)
        Read B(I)
        Store C(I)= A(I)+B(I)
        Increment I=I+1
        If I<= 100 go to 20
        Continue
```

This constitutes a program loop that reads a pair of operands from arrays A and B and performs a floating-point addition. The loop control variables are then updated and the steps repeat 100 times.

A computer capable of vector processing eliminates the overhead of associated with the time it takes to fetch and execute the instructions in the program loop. It allows operations to be specifies with a single vector instruction of the form

C (1:100) = A (1:100) + B (1:100)

The vector includes initial address of the operands, the length of the vectors, and the operation to be performed, all in one instruction. It is in the form of three address format.

Operation Code	Base address Source1	Base address Source2	Base address Destination	Vector length

Fig: Instruction format for vector

Memory interleaving

Pipeline and vector processors often require simultaneous access to memory from two or more source. In such case instead of using two buses, memory can be partitioned into a number of modules connected to a common memory address and data buses. A memory module is a memory array with its address and data registers. The address register receives information from a common address bus and data registers communicate with bidirectional data bus. The two least significant bit of the address can be used to distinguish between the modules. The modular system permits one module to initiate a memory access while other module are in the process of reading and writing a word.

The advantage of modular memory is that it allows the use of a technique called interleaving. In an interleaving memory system, different sets of addressing are assigned to different memory modules. A modular memory is useful in systems with pipeline and vector processing. A vector processor that uses n-way interleaved memory can fetch n operands from n different modules. A CPU with instruction pipeline can take advantage of multiple memory modules so that each segment in the pipeline can access memory independent of memory access from other segment.

Array processors

An array processor is a processor that performs computations on large arrays of data. The term is used to refer two different types of processors. An attached array processor is an auxiliary processor attached to general purpose computer. It is used to improve the performance of the host computer in specific computation tasks. An SIMD array processor is a processor that has a single-instruction multiple-data organization. It manipulates vector instruction by means of many functional units responding to a common instruction.

Attached Array Processor

An attached array processor is designed as a peripheral for a conventional host computer, and its purpose is to enhance the performance of the computer by providing vector processing for complex scientific applications. It achieves high performance by means of parallel processing with multiple functional units. The array processors can be programmed by the user to accommodate a variety of complex arithmetic problems.

SIMD Array processor

An SIMD array processor is a computer with multiple processing units operating in parallel. The processing units are synchronized to perform the same operation under the control of a common control unit, thus providing a single instruction stream, multiple data stream. It contains a set of identical processing elements each having a local memory. Each processor element (PE) includes ALU, a floating point arithmetic unit and working registers. The master control unit controls the operations in the processor elements.

Vector instructions are broadcasted to all processor elements simultaneously. E.g. the vector addition C=A+B. The master control unit first stores the ith components a_i and b_i of A and B in local memory M_i for i= 1, 2, 3, ….., n. It then broadcasts floating point add instruction $c_i = a_i + b_i$ to all PEs. The components of c_i are stored in fixed locations in local memory.

Introduction to Multithreaded Architecture

Multithreading allows a high degree of instruction-parallelism without increasing circuit complexity or power consumption. The instruction stream is divided into several smaller streams known as threads. These threads can be executed in parallel. The variety of specific multithreading designs realized in both commercial systems and experimental system is

vast. Multithreading increases utilization of a single core by using thread-level as well as instruction-level parallelism.

A thread is concerned with scheduling and execution where as a process is concerned with scheduling/execution and resource ownership. The multiple threads with in a process shares the same resources. Implicit multithreading refers to the concurrent execution of multiple threads extracted form a single sequential program. These implicit threads can be defined either statically by the compiler or dynamically by hardware.

For explicit multithreading processor must provide a separate program counter for each thread of execution to be executed concurrently. The designs differ in the amount and type of additional hardware used to support concurrent thread execution. There are four principal approaches to multithreading:

- Interleaved multithreading: This is also known as fine-grained multithreading. The processor deals with two or more thread contexts at a time, switching from one thread to another at each clock cycle. If a thread is blocked due to data dependencies or memory latencies, that thread is skipped and a ready thread is executed.
- Blocked multithreading: This is also known as coarse-grained multithreading. The instructions of a thread are executed successively until an event occurs that may cause delay, such as cache miss. This event switches thread to another.
- Simultaneous multithreading: Instructions are simultaneously issued from multiple threads to the execution units of a superscalar processor. This combines the wide superscalar instruction issue capability with the use of multiple threads contexts.
- Chip multiprocessing: Here the entire processor is replicated on a single chip and each processor handles separate threads. This is referred to as multicore.