High Performance Computing

University questions with solution

Q1) Explain the basic working principle of VLIW processor. (6 marks)

The following points are basic working principle of VLIW processor.

- The aim of VLIW processor is to speed up the computation by utilizing instruction level parallelism.
- Hardware core is same as a superscalar processor with more than one execution units.
- A word length is considered from 52 bits to 1 Kbits.
- All the operations executed in lock-step mode.
- It depends on the compiler for scheduling dependency free program code and searching parallelism

Q2) Explain SIMD, MIMD and **SIMT architecture. (6 marks)**

- SIMD:

1. In Single Instruction Multiple Data Stream, one instruction works on multiple data streams simultaneously with the help of processing elements (PE).
2. The structure of SIMD is as:

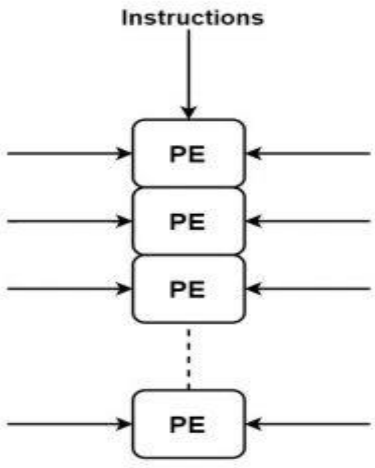


Fig. SIMD architecture

3. The single control unit is used to send a different instruction to different PE. SIMD is used to read instruction by program counter after it decodes them and send the signals to processing elements.

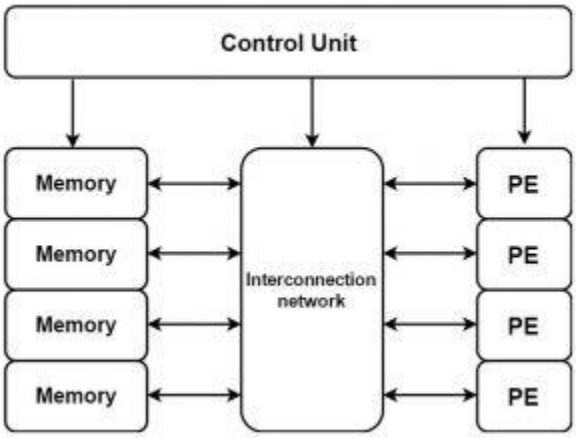


Fig. Processor array

4. All units are connected to each other with an interconnection network which also manages the data transmission. Memory is used to take data. The no of the data path depends upon the number of PE.

- MIMD

1. In Multiple instructions multiple data stream, multiple instructions are executed on multiple data streams simultaneously. It can execute multiple programs at same time.
2. It is used in parallel machines. In MIMD, every processor fetches its own instruction and work on its own data.
3. Processing elements execute different programs at a time. Architectures of MIMD are complex but they provide better performance.
4. For communication between PE, explicit synchronization and protocols are requires for identification.
5. In MIMD, each processing elements stores its own copy of the program. MIMD architecture is divided into shared memory and distributed memory MIMD architecture.

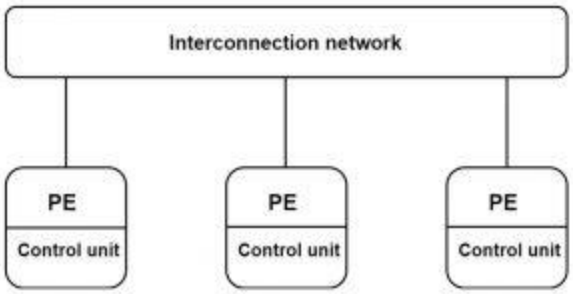


Fig. MIMD architecture

- SIMT:

1. The execution model of Single instruction multiple threads (SIMT) has been implemented on GPUs and GPGPU.
2. It is created with the combination of software and hardware side support.
3. At software side, CUDA and Open CL provide support for achieving data parallelism while at hardware side, for the goal of achieving energy efficiency, it provides a way of converting scalar instructions into vector style SIMD processing.

Q3) Write a short note on data flow model. (4 marks)

- Data flow model is based on the graphical representation of programs where arcs represent data dependencies and nodes represent operations.
- Properties of data flow model are:
  a) Asynchronous executions: If all operands of the instruction are available then only it can execute which represent data flow model is Asynchronous in nature.
  b) No sequencing: There is no need to execute the instruction in a specific order.
  c) Dataflow representation: It allows the use of different forms of parallel program execution without the help of tools.
- Data flow model are divided into the static and dynamic data flow. In static data flow, at most one instance of a node is allowed for firing. When all input token is available for node and no token present in node output arcs then only execution of node can be done.

- In dynamic data flow, during execution, at the same time, a different instance of a node can be activated.
- By using tags which present with every token we can differentiate the different instances of the node and can identify the context.

Q4) Explain Granularity, Concurrency and dependency graph. (5 marks)

- Granularity:

1. The number of tasks into which a problem is decomposed determines its granularity.
2. Coarse granularity: When a system is divided into a smaller number of large parts then it is coarse granularity. It refers smaller no of sub-tasks.
3. Fine granularity: When a system is divided into a large number of small parts then it is fine granularity.
4. Total three values are used to specify granularity they are: fine, medium, coarse.
5. There are three characteristics of the algorithm used for determination of granularity they are: a) the structure of the problem. b) Size of the problem. c) The number of processors available.

- Concurrency:

1. A number of tasks executed in parallel is known as concurrency. Degree of concurrency is used in algorithm design. Maximum the degree of concurrency, the maximum number of the concurrent task can be executed simultaneously at the same time.
2. The average degree of concurrency is the average number of tasks which can be processed in parallel.
3. Two important aspects of concurrency are the ability to deal and responding external events which happen in any order and need to response these events in required time.

- Dependency graph

1. It is a set of items and item to item dependencies which is generally used to verify that at any point the given work is equally distributed across all processes or not.
2. Task dependency graph also contains nodes and edges where nodes represent tasks and edge represent dependency.
3. We can show the dependency graph in different ways.

Q5) What are the characteristic of tasks and interactions. (5 marks)

Let discuss characteristic of tasks first. Characteristic of tasks are:

1) Static and dynamic task generation.
2) Task size.
3) Knowledge of task sizes.
4) Data size.

Let discuss it one by one in detail:
1) Static and dynamic task generation:
In static task generation, the algorithm which is performing work is already known in advance on a priority basis. First, the tasks are defined before execution of algorithm and algorithm follows a certain order in execution. In dynamic task generation, the task is generated as we perform the computation. Before execution of the algorithm, performing tasks are not available.

2) Task size: Size of the task is the time required for its completion. Tasks are divided into uniform and non-uniform tasks. Uniform tasks take the same amount of time in mapping while non-uniform tasks mapping time varies in different schemes.

3) Knowledge of task sizes: If we already know the size of the task then we can use it for mapping of tasks to processes.

4) Data size: The required data should be available while mapping of the task to processes. The overhead can be reduced when the size of data and its memory location is available.

Characteristic of interactions:

1) Static vs. dynamic interaction: in static interaction, tasks, and their interaction are known priori while in the dynamic interaction the timing and interacting tasks cannot determined priori.

2) Regular vs. Irregular: In regular interaction which has a spatial structure which we can utilize for better efficient implementation while in irregular interaction has no any well-defined structure.
Example: Regular- Image dithering
Irregular- Space matrix-vector multiplication.

3) Read-only vs. read-write: In read-only, tasks needs read-only access to shared data while in read-write tasks require read as well as write on shared data.

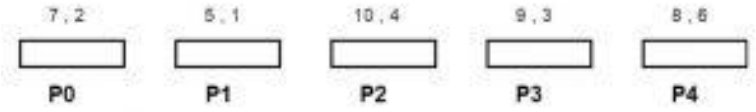
4) One-way vs. two-way: In this, an interaction may be one-way interaction or two-way interaction. In one-way interaction, it can be initiated and accomplished by any one of the interacting tasks while two-way interaction needs involvement of both tasks.

Q6) Explain randomized block distribution and hierarchical mappings. (6 marks)

- Randomized block distribution:

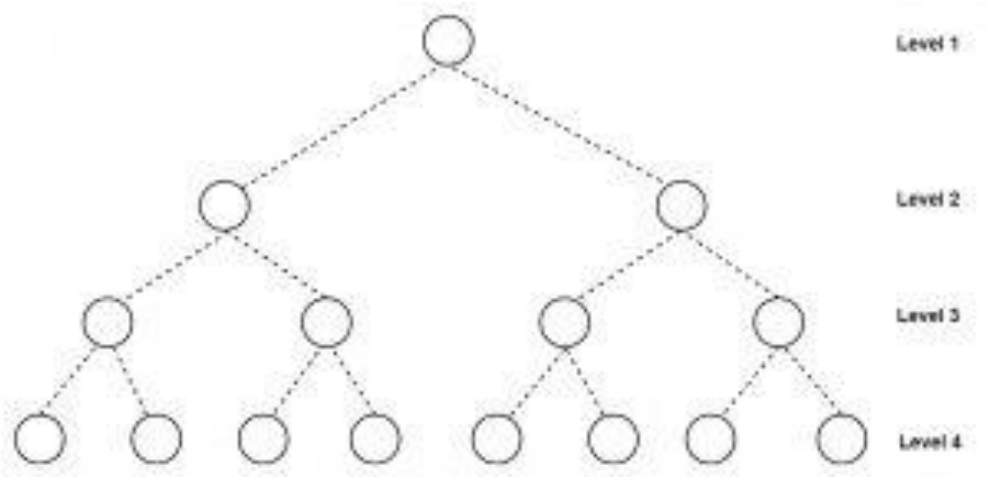
1. In this, an array is dividing into more numbers of blocks than the total number of processes currently active.
2. We can say it is similar to block cyclic distribution. It handles Blocks distribution uniformly and random distribution of blocks is done.
3. Suppose in one-dimensional block distribution, a block size is 2 and 5 processes are available and a vector contains 10 elements then these elements are distributed randomly on each block.

A={1,2,3,4,5,6,7,8,9,10}
The random distribution of elements:
A={7,2,5,1,10,4,9,3,8,6}
Mapping of blocks:



- Hierarchical mappings

1. For binary tree task dependency graph, we can use Hierarchical mapping. For example.



2. The figure represents a binary tree with 4 levels. At the top, root task is divided into two sub-tasks further these level 2 tasks are assigned to four processes. At the bottom level, tasks are mapped with processes. The mapping used at this level is one to one mapping.

Q7) Write a note on IBM Cell Broadband Engine (CBE). (4 marks)

- It is a heterogeneous multicore chip. Basically, CBE is a combination of eight SPE (Synergistic processing element) and one PPE (PowerPC processing elements).
- IBM cell processor is a combination of one 64 bit dual-threaded IBM PowerPC processor and 8 numbers of synergistic processing elements.
- Chip also contains a high-speed memory controller and bandwidth elements interconnect bus, memory and I/O interfaces.

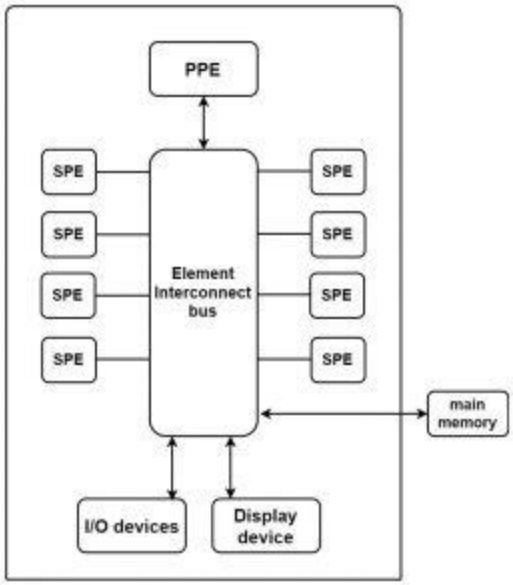


Fig. Cell process architecture

- PPE is a 64 bits core which provides full double precision FMA data path. PPE is complex while the SPE is less complex than PPE. SPE contains 128 *128 register, DMA unit and branch unit etc.
- The main purpose of SPE is to iterate simple operations which required for processing the multimedia data.
- Cell is created by IBM with join collaboration of both Sony and Toshiba.

Q8) What are principles of message passing programming? (4 marks)

- Message passing programs generally are written using:
  a) Asynchronous paradigm: All concurrent tasks make use of asynchronous paradigm which helps in implementation of parallel algorithms. In asynchronous paradigm, Programs are non-deterministic in behaviour because of race condition.
  b) Loosely synchronous programs: In this, a subset of tasks synchronizes to perform interactions and in between these, tasks execute asynchronously. Many parallel algorithms are implemented using loosely synchronous programs.
- Numbers of sequential programming paradigms are considered together for message passing paradigm.
- Here processors have their own memory. Programs modules are designed in such a way that they can run in each processor.
- Communication is needed if modules are running in different processors which occur through messages.

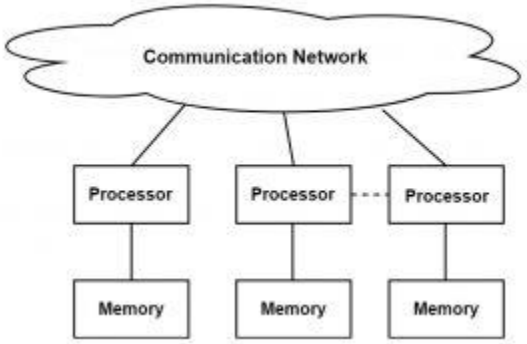


Fig. a system based on message passing paradigm

- In this, memory is not shareable thus processors cannot access each other's memory. When there is data movement from one module to another then message transfer occurs.
- For providing information required for such message transfer, each processor needs to communicate with each other.
- Every process has permission for performing computation on its own data in local phase while for synchronization and communication in a processor, the global phase is allowed but computation in any form is not allowed in global phase.

Q9) Write a note on: Topologies and Embedding. (6 marks)

- We can represent process topology by graphs. 1, 2, 3D topologies are used in message passing programs which are denoted as cartesian topologies.
- Processes are connected with their neighbour in the virtual grid and they can be recognized by cartesian coordinates.
- Function for creating Cartesian topology is:
  int MPI_cart_create (MPI_Comm comm_old,int ndims, int *dims,int *periods,int recorder,MPI_Comm *comm_cart). First, it takes all the processes into old communicator then creates a new communicator having ndims dimensions. Through this, now we can identify every processor in new cartesian topology.
- MPI sees processes are arranged in linear order while parallel programs in high dimension communicate naturally. We can find out such mapping by interaction pattern and topologies of a machine.
- MPI enables programmers to assemble processors in logical n-dimension meshes.
- Topology coordinates are used to identify every process in cartesian topology. By using ranks in MPI we can specify source and destination of processes.
- To convert ranks to Cartesian coordinate and vice versa, MPI provides some routines to do so.
- MPI_Cart_shift function in MPI is used to calculate the rank of a source and destination processes.

Q10) Explain Non-blocking communications using MPI. (4 marks)

- Sender and receiver operations do not block the involved processes till their completion, for this to be carried out we use non-blocking communication.
- First sender process invokes send() but it does not wait until completion of send operation and continue towards the next task.
- Non-blocking send returns control to calling process so that other parts of the system can take control. After starting send() operation sender process needs to handle send complete call. When sending returns then sender continue its other tasks while in this period send perform carry operation on data.

- Sender process verifies that data has been copied from send buffer or not, this task is handled by sending complete call.
- MPI_Isend() and MPI_Irecv() function are used in non-blocking which start their operation but immediately returns control before task completion. I stand for initiate in above functions.
- MPI_Test() is used for checking completion status and MPI_Wait() is used for giving waiting status until the operation completes its execution.

Q11) Write a short note on blocking message passing operations. (6 marks)

Some methods are used to send/receive primitives. Let discuss those methods in detail:

a) Blocking Non-buffered send/receive:
1) In this, handshaking is used for handling the communication between sender and receiver which occur before message transfer. Until matching receiver operation not found the operation is not completed.
2) First sender process sends the request to a receiver for communication then receiver gives reply to the sender process. When sender process gets the reply, it immediately starts the message transfer. Since we are not using any buffering mechanism in message transfer, therefore, this process is known as non-buffered communication.

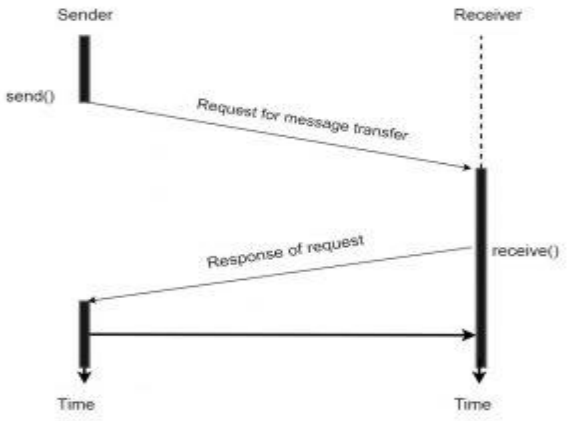


Fig. handshaking of blocking non-buffered send/receive

3) Consider a situation where receiver primitive starts execution before sending primitive thus receiver has to wait for the request from the sender. In that period receiver process goes into the idle state without doing any tasks. This problem of idling overheads when occur then it is minimized because both sender and receiver are ready for exchanging information.

b) Blocking buffered send/receive:

1) In this, the problem of overhead is removed by the use of buffer at both sender and receiver processes.
2) When the sender invokes send primitive operation, a message to be transferred is copied from sender address space to its buffer. After copying, sender resumes its execution.
3) At the receiver side, receiver process checks whether receive buffer contains any message or not. This checking takes place with the help of interrupt and polling mechanism.

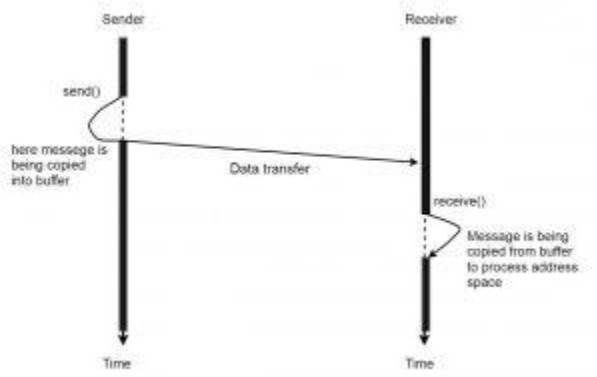


Fig. blocking buffer transfer

4) When the message is available in receiver buffer then an interrupt is generated and informed to the receiver then receiver copy that message to receiver process.
5) Small programs are used in polling for checking the availability of message in the buffer in short interval of time.