Inter processor arbitration

Computer systems need buses to facilitate the transfer of information between their various components. There is a dispute in the system bus that connects the CPU, Input-Output processor, and memory (main components of a computer). Only one between the CPU, Input-Output processor, and memory gets the grant to use the bus simultaneously. Hence, an appropriate priority resolving mechanism is required to decide which processor should get control of the bus. Therefore, a mechanism is needed to handle multiple requests for the bus, known as Inter processor Arbitration.

Arbitration Techniques

Following are the techniques of arbitration:

Static Arbitration Techniques

In this technique, the priority assigned is fixed. It has two types. They are serial arbitration procedures and parallel arbitration logic.

Serial Arbitration

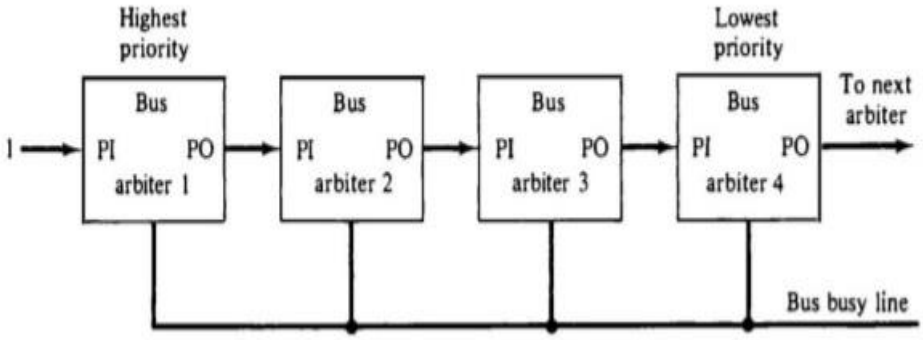

Fig: Serial (daisy-chain) arbitration

It is also known as Daisy chain Arbitration. It is obtained by the daisy-chain connection of bus arbitration circuits. The scheme got the term from the structure of the grant line, which chains through each device from the higher to lowest priority. The highest priority device will pass the grant line to the lower priority device only if it does not want it. Then the priority is forwarded to the next in the sequence. All devices use the same line for bus requests. If a busy bus line returns to its idle state, the most high-priority arbiter enables the busy line, and its corresponding processor can then run the required bus transfer.

Advantage

i) It is a simple design.

ii) Less number control lines are used.

Disadvantage

i) Priority depends on the physical location of the device

ii) Propagation delay due to serially granting of bus

iii) Failure of one of the devices may fail the entire system

iv) Cannot assure fairness- a low priority device may be locked out indefinitely

Parallel Arbitration

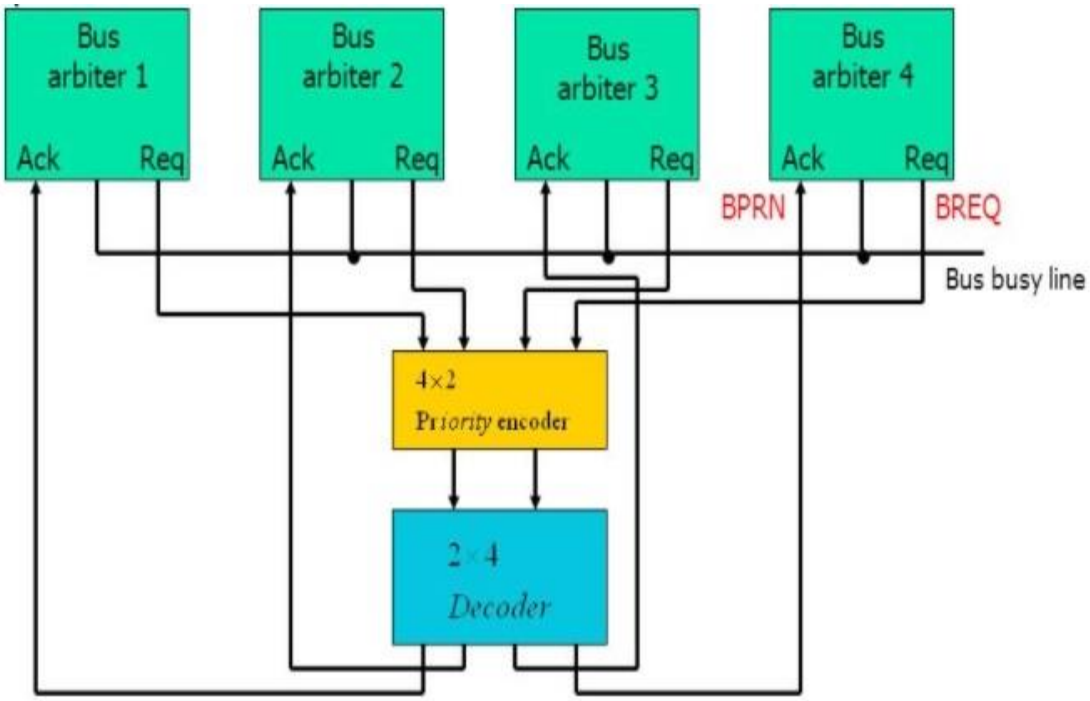

Fig: Parallel arbitration

It uses an external priority encoder and decoder. Each bus arbiter has a bus request output line and a bus acknowledge input line. Each arbiter enables request lines when its processor is requesting the system bus. The one with the highest priority determined by the output of the decoder gets access to the bus.

Dynamic Arbitration Techniques

Serial and Parallel bus arbitration are static since the priorities assigned are fixed. In dynamic arbitration, priorities of the system change while the system is in operation.

The various algorithms used are:-

Time Slice

It allocates a fixed-length time slice of bus time offered sequentially to each process in a round-robin fashion. The service provided by each system component and the system is independent of its location near the bus.

Polling

In polling, the controller generates the addresses for the devices. The number of address lines needed depends upon the number of connected devices to the system. The controller generates a sequence of device addresses in response to the bus request. When the requesting device recognizes its address, it activates the busy bus line and uses the bus. After several bus cycles, the polling process continues by choosing a different processor. The polling sequence usually is programmable, and as a result, the selection priority can be altered under program control.

Least Recently Used (LRU)

This algorithm provides the highest priority for a device, requesting that it has not used the bus for too long. The priorities are adjusted after several bus cycles according to the LRU algorithm.

FIFO (First IN First Out)

Requests are served in the order received in the FIFO scheme. The bus controller establishes a queue as per the request of the incoming bus to use this algorithm.

Each processor has to wait for its bus usage time as per first in and first-out (FIFO).

Rotating Daisy chain

It is a dynamic extension of the daisy chain algorithm. There is no central bus controller in this scheme, and the priority line is connected from the priority out of the last device back to the priority-in of the first device in a closed-loop.

Each arbiter priority for a given bus cycle is determined by its position along the bus priority line from the arbiter whose processor is currently controlling the bus. Once an arbiter releases the bus, it has the lowest priority.

Interprocessor Communication and Synchronization

In general, Interprocess Communication is a mechanism provided by the operating system (or OS). The various processors in a multiprocessor system should be provided with a facility for communicating with each other. The working of this mechanism is to provide communications between several processes.

In short, intercommunication allows a process to let another process know that some event has occurred or data is to be transferred from one process to another.

This article will help you understand Interprocessor Communication and Synchronization, methods to provide synchronization, approaches used for inter-process communication, and the importance of Inter-process communication.

What is Interprocessor Communication?

Interprocessor communication is used for interchanging useful information among various regions in one or more processes (or programs). This communication could involve letting another process know that some event has occurred or the transferring of data from one process to another.

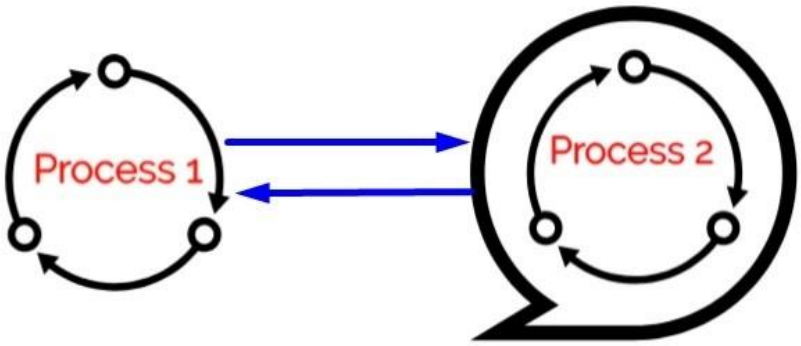

Fig: Interprocess communication illustration

Synchronization in Interprocessor Communication

Synchronization is an essential part of interprocess communication. It refers to a case where the data used to communicate between processors is control information. It is either given by the interprocess control mechanism or handled by the communicating processes.

It is required to maintain the correct sequence of processes and to make sure equal access to shared writable data.

Multiprocessor systems have various mechanisms for the synchronization of resources. Below are some methods to provide synchronization are as follows −

- **Mutual Exclusion**
- **Semaphore**
- **Barrier**
- **Spinlock**
- **Mutual Exclusion**

Mutual Exclusion requires that only a single process thread can enter the critical section one at a time. This also helps synchronize and prevents the race condition by creating a stable state.

Semaphore

Semaphore is a type of variable that generally controls the access to the shared resources by several processes. Further, Semaphore is divided into two types as follows:

Binary Semaphore

A binary semaphore is limited to zero or one. It could be used to control access to one resource. In particular, it can be used to force the same release of an important category in the user code.

Counting Semaphore

Counting semaphore may take any integer value. It could be used to control access to resources having many instances.

Barrier

A barrier (as evident by its name) does not allow an individual process to proceed unless all the processes do not reach it. Many parallel languages use it, and collective routines impose barriers.

Spinlock

As evident by its name, a spinlock is a type of lock that prevents processes from operating any function unless it is available. The processes which are trying to acquire the spinlock wait in a loop while checking if the lock is available or not. This is also known as busy waiting because the process is not doing any helpful operation even though it is active.