5 Synchronous Computations

[Weightage(17%): Approx. 11-12 Marks out of 70 Marks]

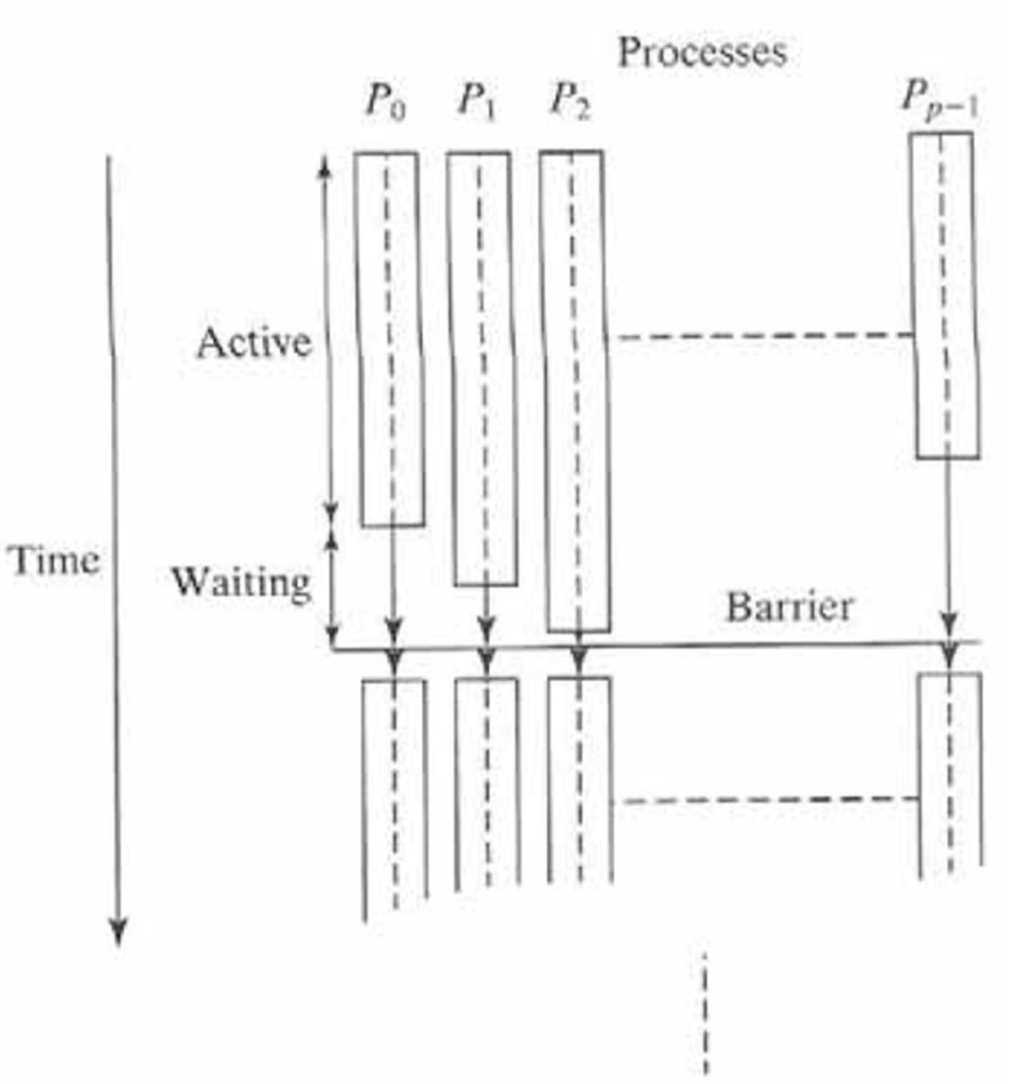
- Synchronization
- Synchronized Computations
- Synchronous Iteration Program Examples
- Partially Synchronous Methods

5.1 Synchronization

- Group of separate computations that must at times wait for each other before proceeding (thereby becoming synchronize)
- All the processes are synchronized at regular points.
- Generally, same operation is applied to a set of data points (E.g. SIMD)

Barrier

- As every process completes the task given to it at different time, each process must wait until all the processes have reached a particular reference point, so that they can exchange data and then continue from a known state together. This mechanism is called a **barrier**.



Library Call approach for a Barrier

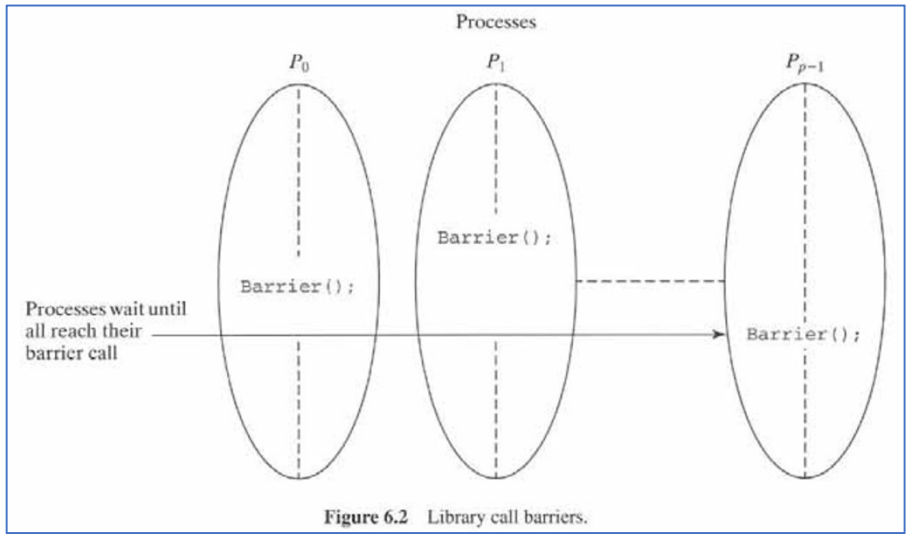


Figure 6.2 Library call barriers.

Centralized Counter Implementation in Barrier (Linear Barrier)

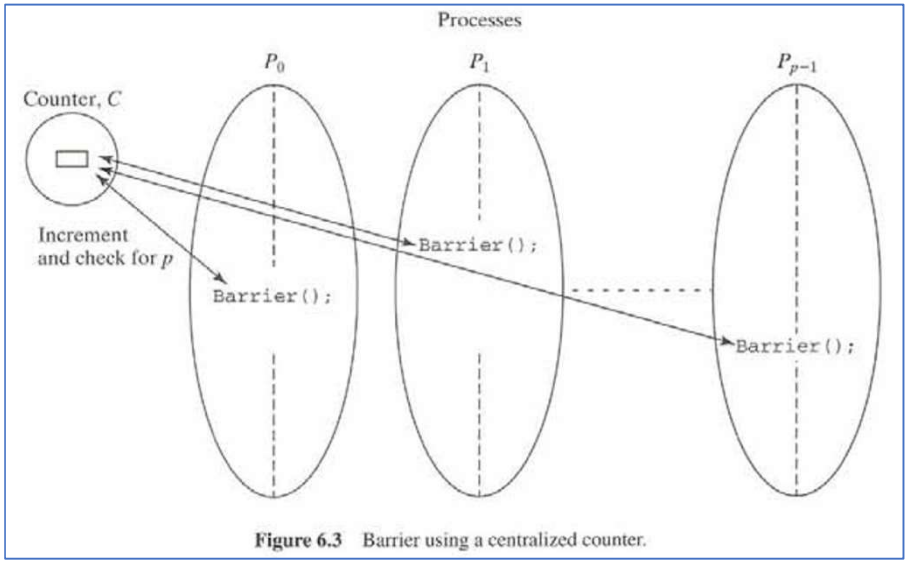
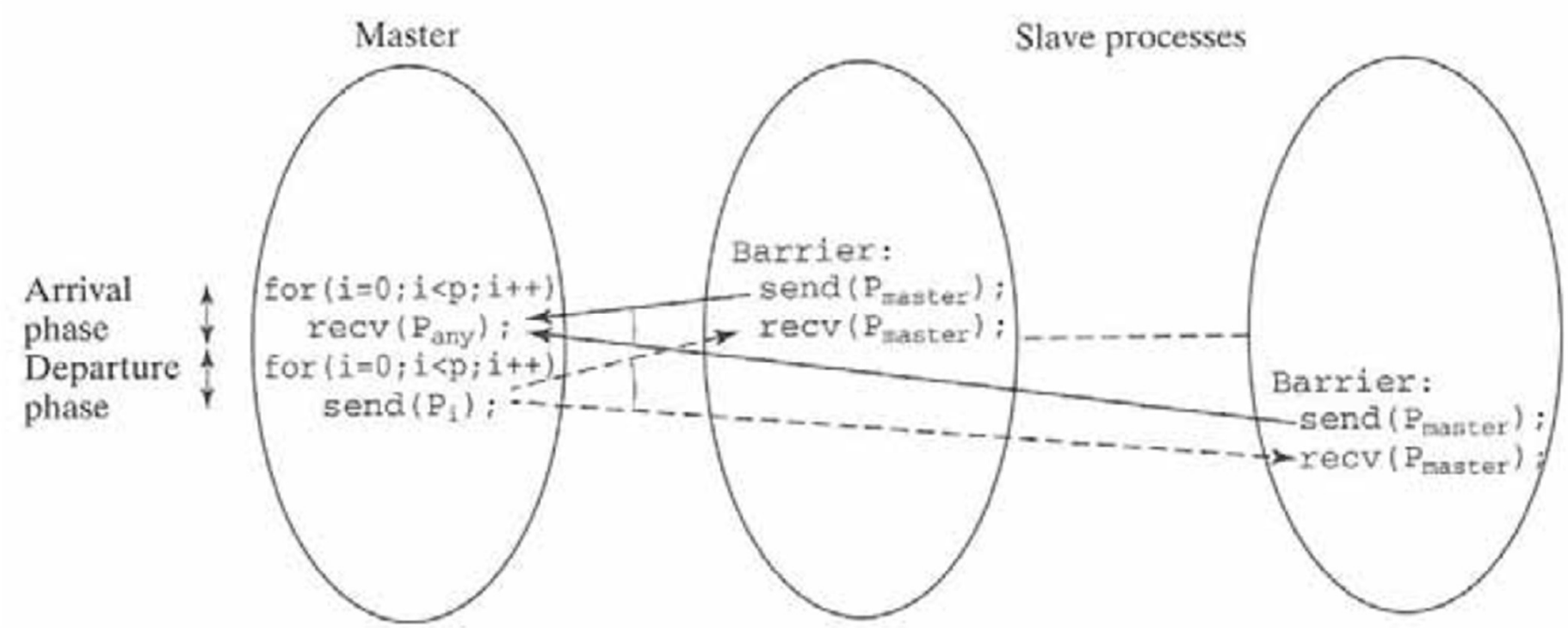


Figure 6.3 Barrier using a centralized counter.

A single counter is used to count the number of processes reaching the barrier.

Counter Implementation in a Message-Passing System



The master process maintains the barrier counter (`i`) and it counts the messages received from "slave" processes when they reach their barrier (during the arrival phase) and releases slave processes (in the departure phase).

Barrier Implementation using Tree construction

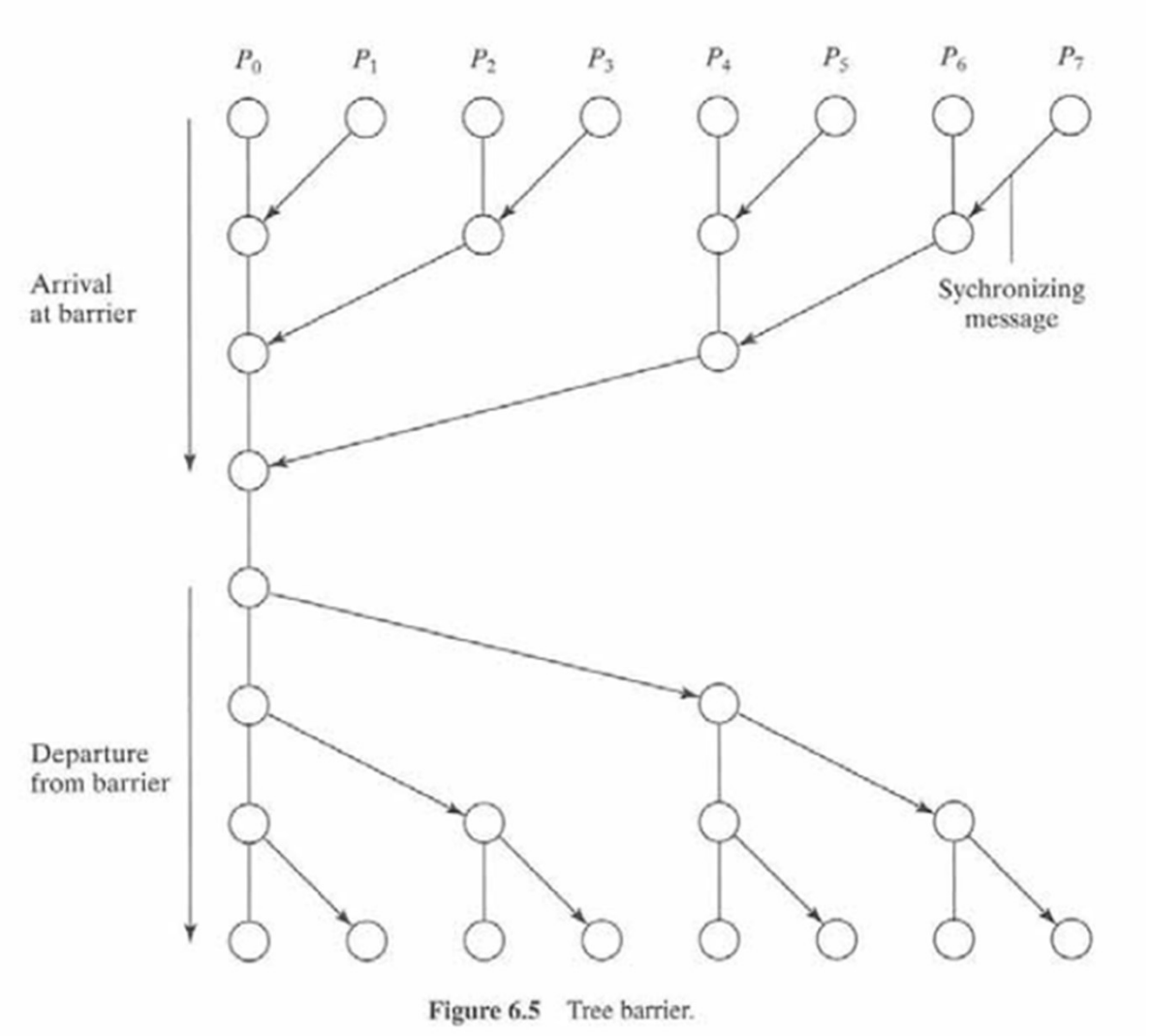


Figure 6.5 Tree barrier.

Barrier Implementation using Butterfly construction

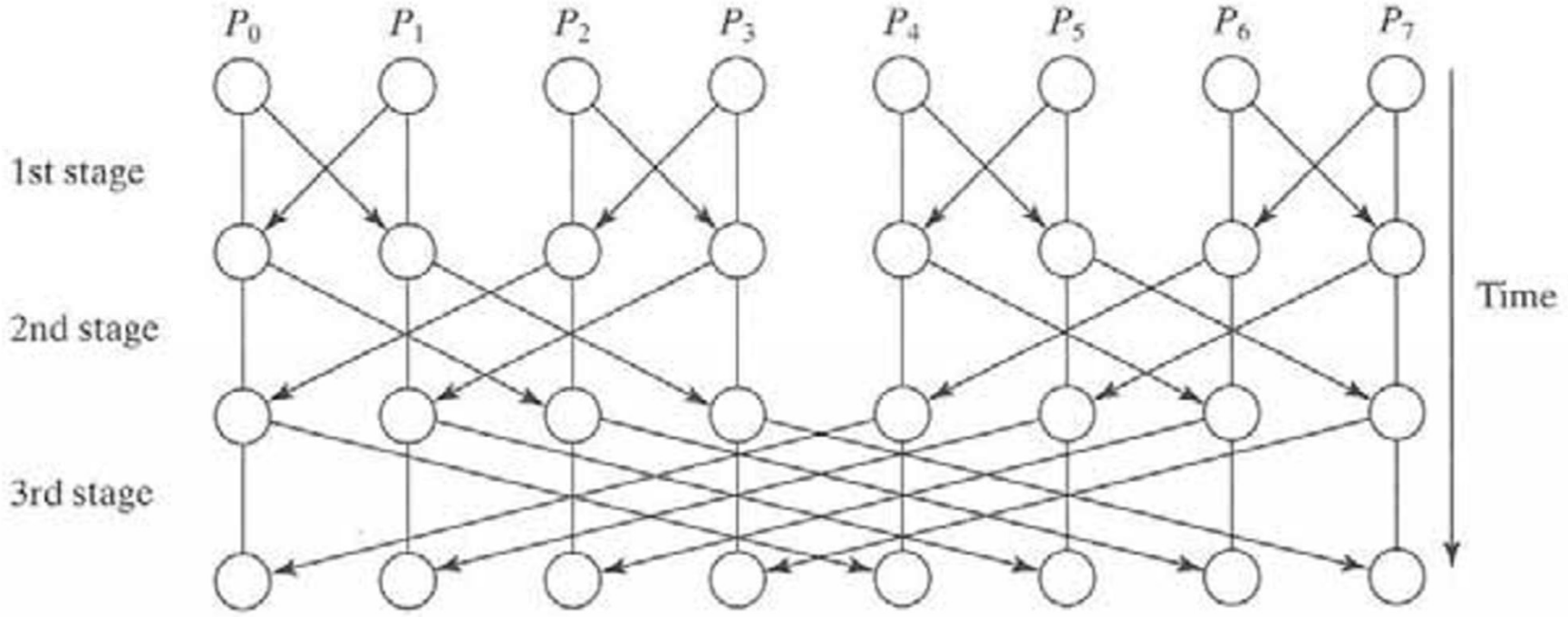
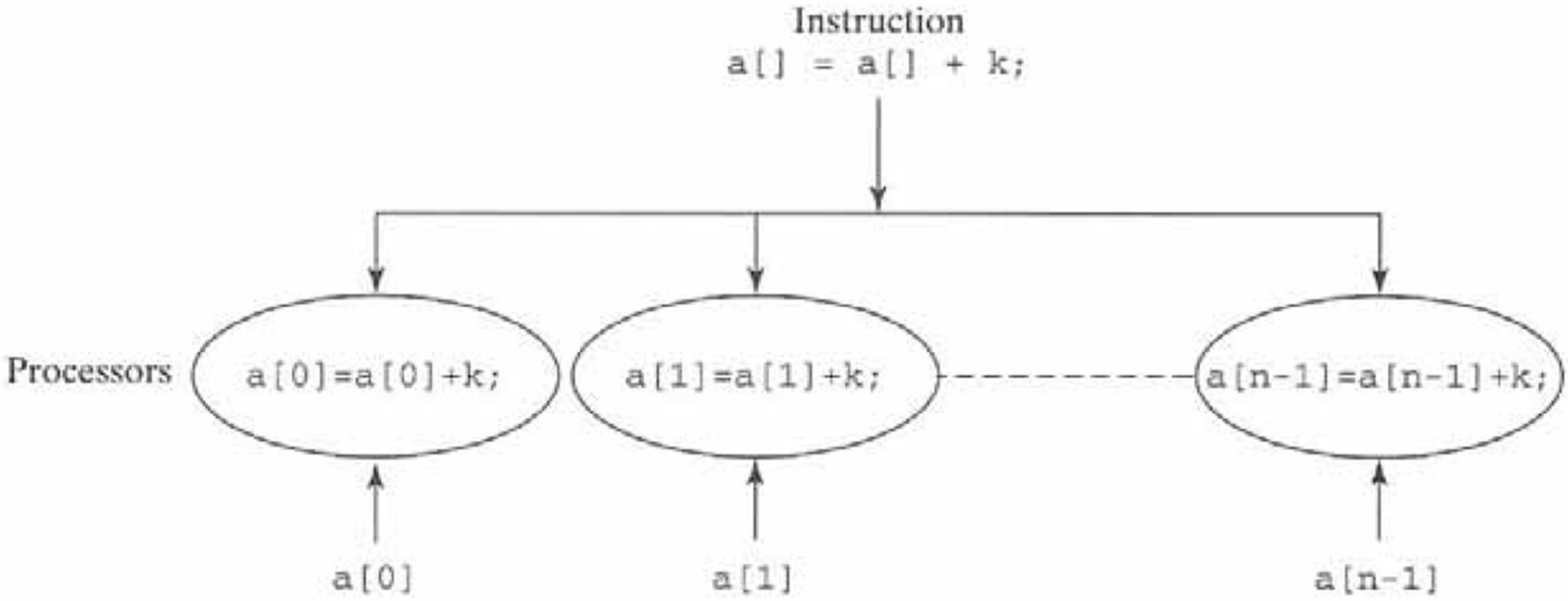


Figure 6.6 Butterfly construction.

Deadlock

- When a pair of processes send and receive from each other, deadlock may occur.
- Deadlock will occur using synchronous routines if both processes perform the send first and will wait for matching receives that are never reached.
- Solution:
- Arrange for one process to receive first and then send AND the other process to send first and then receive.
- For instance, even-numbered processes can be arranged to perform their sends first and the odd-numbered processes to perform their received first.

5.2 Synchronized Computations: Data Parallel Computations (E.g. SIMD)



Prefix Sum Problem

- Data parallel method of adding all the partial sums of 16 numbers
- It has a multiple treelike construction and computes the partial sums in the locations x[i] (0 <= i < 16)
- First, 15 (16-1) additions occur in which x[i-1] is added to x[i] for 1<=i<16.
- Then, 14 (16-2)additions occur in which x[i-2] is added to x[i] for 2<=i<16.
- Then, 12 (16-4)additions occur in which x[i-4] is added to x[i] for 4<=i<16.

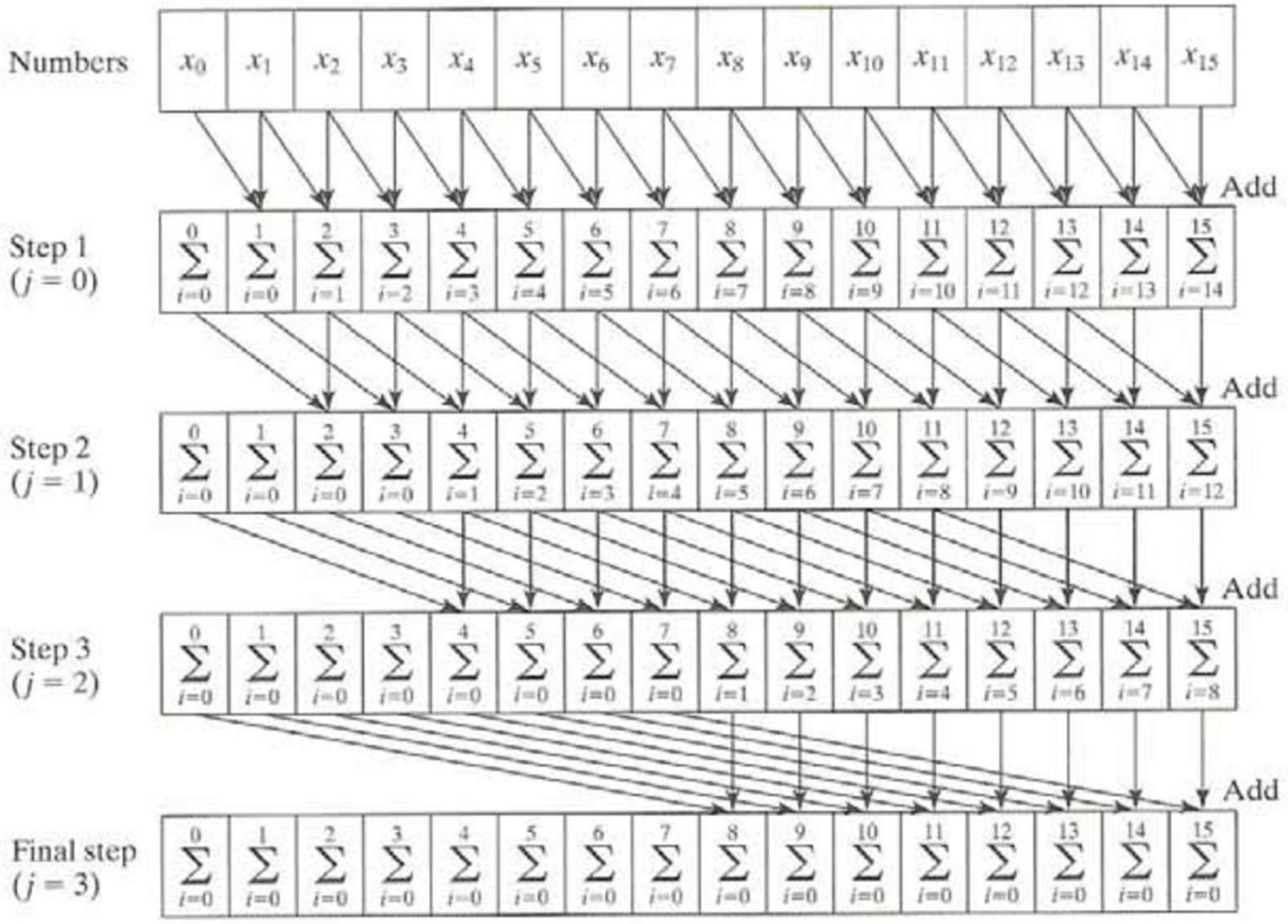


Figure 6.8 Data parallel prefix sum operation.

5.3 Synchronous Iteration Program Examples

$$a_{n-1,0}x_0 + a_{n-1,1}x_1 + a_{n-1,2}x_2 \ldots\ldots\ldots + a_{n-1,n-1}x_{n-1} = b_{n-1}$$

$$\vdots$$

$$a_{2,0}x_0 + a_{2,1}x_1 + a_{2,2}x_2 \ldots\ldots\ldots + a_{2,n-1}x_{n-1} = b_2$$

$$a_{1,0}x_0 + a_{1,1}x_1 + a_{1,2}x_2 \ldots\ldots\ldots + a_{1,n-1}x_{n-1} = b_1$$

$$a_{0,0}x_0 + a_{0,1}x_1 + a_{0,2}x_2 \ldots\ldots\ldots + a_{0,n-1}x_{n-1} = b_0$$

where unknown variable are $x_0, x_1, x_2 \ldots x_{n-1}$.
One way to solve the equation is ***Jacobi iteration*** wherein all the values of x are updated together (Alternative methods could be *Gauss-Seidel*).

5.4 Partially Synchronous Methods

- Synchronization causes significance performance degradation

Parallel Code

```
for (iteration=0; iteration<limit; iteration++)
        forall(i=1; i<n; i++)
                forall(j=1; j<n; ++j)
                        h[i][j] = 0.25 * (h[i-1][j] + h[i+1][j] + h[i][j-1] + h[i][j+1]);
```

The above parallel code is written with an assumption that the values on the right side of the computation are computed from the preceding iteration. This is a typical Jacobi iteration method which requires a global synchronization point (barrier) for processes to wait until all processes have performed their calculations.

Suppose the barrier was removed altogether, allowing processes to continue with subsequent iterations before other processes have completed their present iteration. Then the processes moving forward would use valued computed from not only the preceding iteration but from earlier iterations and not only the last iteration. The method is called an **asynchronous iterative method**.

5 Synchronous Computations

[Weightage(17%): Approx. 11-12 Marks out of 70 Marks]

1. Explain the concept barrier with diagram in synchronous computation.
2. Explain Linear Barrier or Centralized Counter Implementation in Barrier.
3. Explain with diagram, barrier implementation in a message-passing system
4. Barrier Implementation using Tree construction.
5. Explain Barrier Implementation using Butterfly construction.
6. Explain the problem of deadlock in brief.
7. Explain prefix sub problem with example.