
UNIT 1 PARALLEL ALGORITHMS

Structure	Page Nos.
1.0 Introduction	5
1.1 Objectives	6
1.2 Analysis of Parallel Algorithms	6
1.2.1 Time Complexity	
1.2.1.1 Asymptotic Notations	
1.2.2 Number of Processors	
1.2.3 Overall Cost	
1.3 Different Models of Computation	8
1.3.1 Combinational Circuits	
1.4 Parallel Random Access Machines (PRAM)	10
1.5 Interconnection Networks	10
1.6 Sorting	11
1.7 Combinational Circuit for Sorting the String	11
1.8 Merge Sort Circuit	14
1.9 Sorting Using Interconnection Networks	16
1.10 Matrix Computation	19
1.11 Concurrently Read Concurrently Write (CRCW)	20
1.12 Concurrently Read Exclusively Write (CREW)	20
1.13 Summary	21
1.14 Solutions/Answers	22
1.15 References/Further Readings	22

1.0 INTRODUCTION

An algorithm is defined as a sequence of computational steps required to accomplish a specific task. The algorithm works for a given input and will terminate in a well defined state. The basic conditions of an algorithm are: input, output, definiteness, effectiveness and finiteness. The purpose of the development an algorithm is to solve a general, well specified problem.

A concern while designing an algorithm also pertains to the kind of computer on which the algorithm would be executed. The two forms of architectures of computers are: sequential computer and parallel computer. Therefore, depending upon the architecture of the computers, we have sequential as well as parallel algorithms.

The algorithms which are executed on the sequential computers simply perform according to sequence of steps for solving a given problem. Such algorithms are known as sequential algorithms.

However, a problem can be solved after dividing it into sub-problems and those in turn are executed in parallel. Later on, the results of the solutions of these subproblems can be combined together and the final solution can be achieved. In such situations, the number of processors required would be more than one and they would be communicating with each other for producing the final output. This environment operates on the parallel computer and the special kind of algorithms called parallel algorithms are designed for these computers. The parallel algorithms depend on the kind of parallel computer they are designed for. Hence, for a given problem, there would be a need to design the different kinds of parallel algorithms depending upon the kind of parallel architecture.



A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem. This definition is broad enough to include parallel supercomputers that have hundreds or thousands of processors, networks of workstations, multiple-processor workstations, and embedded systems. The parallel computers can be represented with the help of various kinds of models such as random access machine (RAM), parallel random access machine (PRAM), Interconnection Networks etc. While designing a parallel algorithm, the computational power of various models can be analysed and compared, parallelism can be involved for a given problem on a specific model after understanding the characteristics of a model. The analysis of parallel algorithm on different models assist in determining the best model for a problem after receiving the results in terms of the time and space complexity.

In this unit, we have first discussed the various parameters for analysis of an algorithm. Thereafter, the various kinds of computational models such as combinational circuits etc. have been presented. Subsequently, a few problems have been taken up, e.g., sorting, matrix multiplication etc. and solved using parallel algorithms with the help of various parallel computational models.

1.1 OBJECTIVES

After studying this unit the learner will be able to understand about the following:

- Analysis of Parallel Algorithms;
- Different Models of Computation;
 - Combinational Circuits
 - Interconnection Networks
 - PRAM
- Sorting Computation, and
- Matrix Computation.

1.2 ANALYSIS OF PARALLEL ALGORITHMS

A generic algorithm is mainly analysed on the basis of the following parameters: the time complexity (execution time) and the space complexity (amount of space required). Usually we give much more importance to time complexity in comparison with space complexity. The subsequent section highlights the criteria of analysing the complexity of parallel algorithms. The fundamental parameters required for the analysis of parallel algorithms are as follow:

- Time Complexity
- The Total Number of Processors Required
- The Cost Involved.

1.2.1 Time Complexity

As it happens, most people who implement algorithms want to know how much of a particular resource (such as time or storage) is required for a given algorithm. The parallel architectures have been designed for improving the computation power of the various algorithms. Thus, the major concern of evaluating an algorithm is the determination of the amount of time required to execute. Usually, the time complexity is calculated on the basis of the total number of steps executed to accomplish the desired output.



The Parallel algorithms usually divide the problem into more symmetrical or asymmetrical subproblems and pass them to many processors and put the results back together at one end. The resource consumption in parallel algorithms is both processor cycles on each processor and also the communication overhead between the processors.

Thus, first in the computation step, the local processor performs an arithmetic and logic operation. Thereafter, the various processors communicate with each other for exchanging messages and/or data. Hence, the time complexity can be calculated on the basis of computational cost and communication cost involved.

The time complexity of an algorithm varies depending upon the instance of the input for a given problem. For example, the already sorted list (10,17, 19, 21, 22, 33) will consume less amount of time than the reverse order of list (33, 22, 21,19,17,10). The time complexity of an algorithm has been categorised into three forms, viz:

- i) Best Case Complexity;
- ii) Average Case Complexity; and
- iii) Worst Case Complexity.

The best case complexity is the least amount of time required by the algorithm for a given input. The average case complexity is the average running time required by the algorithm for a given input. Similarly, the worst case complexity can be defined as the maximum amount of time required by the algorithm for a given input.

Therefore, the main factors involved for analysing the time complexity depends upon the algorithm, parallel computer model and specific set of inputs. Mostly the size of the input is a function of time complexity of the algorithm. The generic notation for describing the time-complexity of any algorithm is discussed in the subsequent sections.

1.2.1.1 Asymptotic Notations

These notations are used for analysing functions. Suppose we have two functions $f(n)$ and $g(n)$ defined on real numbers,

- i) *Theta Θ Notation:* The set $\Theta(g(n))$ consists of all functions $f(n)$, for which there exist positive constants c_1, c_2 such that $f(n)$ is sandwiched between $c_1 * g(n)$ and $c_2 * g(n)$, for sufficiently large values of n . In other words,

$$\Theta(g(n)) = \{ 0 <= c_1 * g(n) <= f(n) <= c_2 * g(n) \text{ for all } n >= n_0 \}$$
- ii) *Big O Notation:* The set $O(g(n))$ consists of all functions $f(n)$, for which there exists positive constants c such that for sufficiently large values of n , we have $0 <= f(n) <= c * g(n)$. In other words,

$$O(g(n)) = \{ 0 <= f(n) <= c * g(n) \text{ for all } n >= n_0 \}$$
- iii) *Ω Notation:* The function $f(n)$ belongs to the set $\Omega(g(n))$ if there exists positive constants c such that for sufficiently large values of n , we have $0 <= c * g(n) <= f(n)$. In other words,

$$O(g(n)) = \{ 0 <= c * g(n) <= f(n) \text{ for all } n >= n_0 \}.$$

Suppose we have a function $f(n) = 4n^2 + n$, then the order of function is $O(n^2)$. The asymptotic notations provide information about the lower and upper bounds on complexity of an algorithm with the help of Ω and O notations. For example, in the sorting algorithm the lower bound is $\Omega(n \ln n)$ and upper bound is $O(n \ln n)$. However, problems like matrix multiplication have complexities like $O(n^3)$ to $O(n^{2.38})$. Algorithms



which have similar upper and lower bounds are known as optimal algorithms. Therefore, few sorting algorithms are optimal while matrix multiplication based algorithms are not.

Another method of determining the performance of a parallel algorithm can be carried out after calculating a parameter called “speedup”. Speedup can be defined as the ratio of the worst case time complexity of the fastest known sequential algorithm and the worst case running time of the parallel algorithm. Basically, speedup determines the performance improvement of parallel algorithm in comparison to sequential algorithm.

$$\text{Speedup} = \frac{\text{Worst case running time of Sequential Algorithm}}{\text{Worst case running time of Parallel Algorithm}}$$

1.2.2 Number of Processors

One of the other factors that assist in analysis of parallel algorithms is the total number of processors required to deliver a solution to a given problem. Thus, for a given input of size say n, the number of processors required by the parallel algorithm is a function of n, usually denoted by TP (n).

1.2.3 Overall Cost

Finally, the total cost of the algorithm is a product of time complexity of the parallel algorithm and the total number of processors required for computation.

$$\text{Cost} = \text{Time Complexity} * \text{Total Number of Processors}$$

The other form of defining the cost is that it specifies the total number of steps executed collectively by the n number of processors, i.e., *summation of steps*. Another term related with the analysis of the parallel algorithms is *efficiency* of the algorithm. It is defined as the ratio of the worst case running time of the best sequential algorithm and the cost of the parallel algorithm. The efficiency would be mostly less than or equal to 1. In a situation, if efficiency is greater than 1 then it means that the sequential algorithm is faster than the parallel algorithm.

$$\text{Efficiency} = \frac{\text{Worst case running time of Sequential Algorithm}}{\text{Cost of Parallel Algorithm}}$$

1.3 DIFFERENT MODELS OF COMPUTATION

There are various computational models for representing the parallel computers. In this section, we discuss various models. These models would provide a platform for the designing as well as the analysis of the parallel algorithms.

1.3.1 Combinational Circuits

Combinational Circuit is one of the models for parallel computers. In interconnection networks, various processors communicate with each other directly and do not require a shared memory in between. Basically, combinational circuit (cc) is a connected arrangement of logic gates with a set of m input lines and a set of n output lines as shown in *Figure 1*. The combinational circuits are mainly made up of various interconnected components arranged in the form known as *stages* as shown in *Figure 2*.

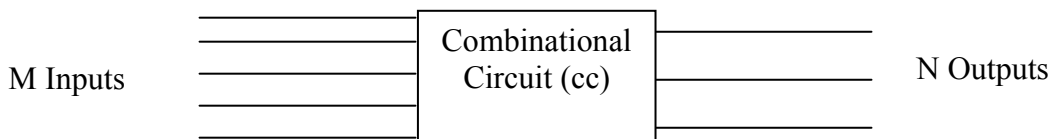


Figure 1: Combinational circuit

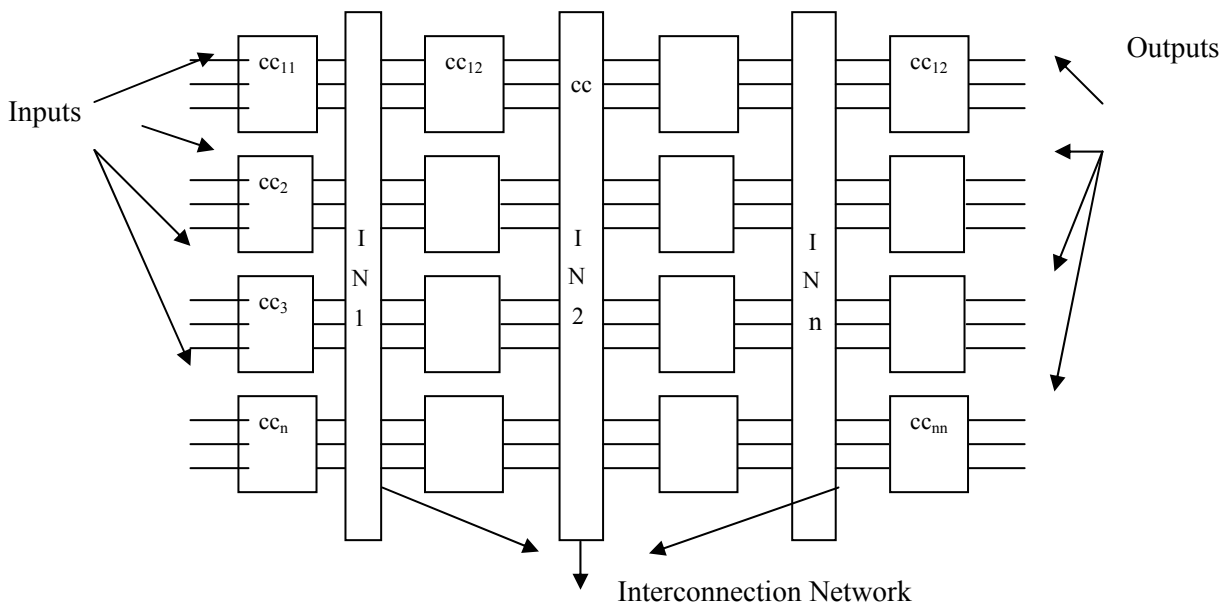


Figure 2: Detailed combinational circuit

It may be noted here that there is no feedback control employed in combinational circuits. There are few terminologies followed in the context of combinational circuits such as fan in and fan out. Fan in signifies the number of input lines attached to each device and fan out signifies the number of output lines. In *Figure 2*, the fan in is 3 and fan out is also 3. The following parameters are used for analysing a combinational circuit:

- 1) *Depth*: It means that the total number of stages used in the combinational circuit starting from the input lines to the output lines. For example, in the depth is 4, as there are four different stages attached to a interconnection network. The other form of interpretation of depth can be that it represents the worst case time complexity of solving a problem as input is given at the initial input lines and data is transferred between various stages through the interconnection network and at the end reaches the output lines.
- 2) *Width*: It represents the total number of devices attached for a particular stage. For example in *Figure 2*, there are 4 components attached to the interconnection network. It means that the width is 4.
- 3) *Size*: It represents the total count of devices used in the complete combinational circuit. For example, in *Figure 2*, the size of combinational circuit is 16 i.e. (width * depth).



1.4 PARALLEL RANDOM ACCESS MACHINES (PRAM)

PRAM is one of the models used for designing the parallel algorithm as shown in *Figure 3*. The PRAM model contains the following components:

- i) A set of identical type of processors say $P_1, P_2, P_3 \dots P_n$.
- ii) It contains a single shared memory module being shared by all the N processors. As the processors cannot communicate with each other directly, shared memory acts as a communication medium for the processors.
- iii) In order to connect the N processor with the single shared memory, a component called Memory Access Unit (MAU) is used for accessing the shared memory.

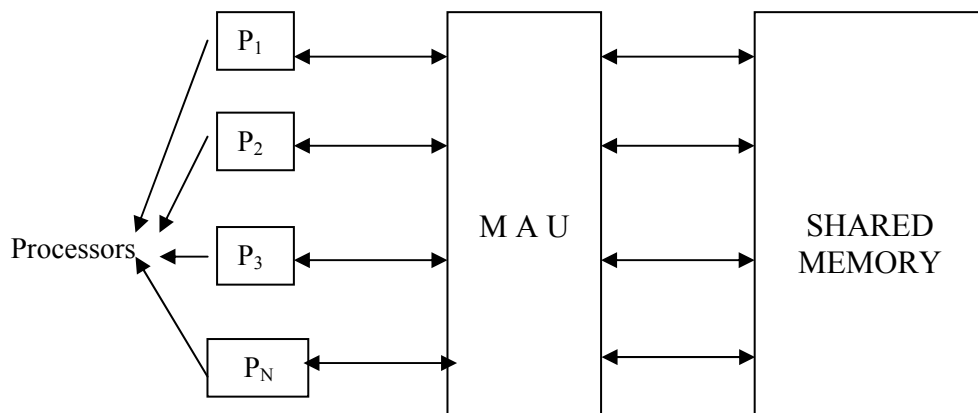


Figure 3: PRAM Model

Following steps are followed by a PRAM model while executing an algorithm:

- i) *Read phase*: First, the N processors simultaneously read data from N different memory locations of the shared memory and subsequently store the read data into its local registers.
- ii) *Compute phase*: Thereafter, these N processors perform the arithmetic or logical operation on the data stored in their local registers.
- iii) *Write phase*: Finally, the N processors parallel write the computed values from their local registers into the N memory locations of the shared memory.

1.5 INTERCONNECTION NETWORKS

As in PRAM, there was no direct communication medium between the processors, thus another model known as interconnection networks have been designed. In the interconnection networks, the N processors can communicate with each other through direct links. In the interconnection networks, each processor has an independent local memory.



☞ **Check Your Progress 1**

- 1) Which of the following model of computation requires a shared memory?
 - 1) PRAM
 - 2) RAM
 - 3) Interconnection Networks
 - 4) Combinational Circuits

- 2) Which of the following model of computation has direct link between processors?
 - 1) PRAM
 - 2) RAM
 - 3) Interconnection Networks
 - 4) Combinational Circuits

- 3) What does the term width depth in combinational circuits mean?
 - 1) Cost
 - 2) Running Time
 - 3) Maximum number of components in a given stage
 - 4) Total Number of stages

4) Explain the concept of analysis of parallel algorithms.

.....

.....

.....

.....

1.6 SORTING

The term sorting means arranging elements of a given set of elements, in a specific order i.e., ascending order / descending order / alphabetic order etc. Therefore, sorting is one of the interesting problems encountered in computations for a given data. In the current section, we would be discussing the various kinds of sorting algorithms for different computational models of parallel computers.

The formal representation of sorting problem is as explained: Given a string of m numbers, say $X = x_1, x_2, x_3, x_4, \dots, x_m$ and the order of the elements of the string X is initially arbitrary. The solution of the problem is to rearrange the elements of the string X such that the resultant sequence is in an ascending order.

Let us use the combinational circuits for sorting the string.

1.7 COMBINATIONAL CIRCUIT FOR SORTING THE STRING

Each input line of the combinational circuit represents an individual element of the string say x_i and each output line results in the form of a sorted list. In order to achieve the above mentioned task, a comparator is employed for the processing.

Each comparator has two input lines, say a and b , and similarly two output lines, say c and d . Each comparator provides two outputs i.e., c provides maximum of a and b (**max**



(a, b)) and d provides minimum of a and b (**min (a, b)**) in comparator InC and DeC it is opposite, as shown in *Figure 4 and 5*.

In general, there are two types of comparators, often known as increasing comparators and decreasing comparators denoted by + BM(n) and - BM(n) where n denotes the number of input lines and output lines of the comparator. The depth of + BM(n) and - BM(n) is log n. These comparators are employed for constructing the circuit of sorting.

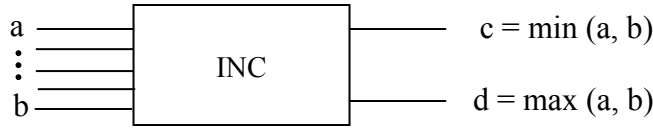


Figure 4 (a) Increasing Comparator, for 2 inputs

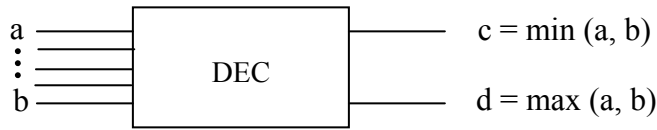


Figure 4 (b) Decreasing Comparator, for 2 inputs

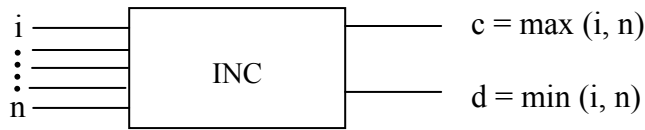


Figure 5 (a): Increasing Comparator, for n inputs

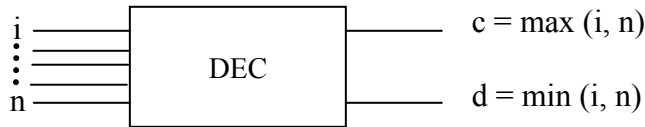


Figure 5 (b): Decreasing Comparator, for n inputs

Now, let us assume a famous sequence known as bitonic sequence and sort out the elements using a combinational circuit consisting of a set of comparators. The property of bitonic sequence is as follows:

Consider a sequence $X = x_0, x_1, x_2, x_3, x_4, \dots, x_{n-1}$ such that condition 1: either $x_0, x_1, x_2, x_3, x_4, \dots, x_i$ is a monotonically increasing sequence and $x_{i+1}, x_{i+2}, \dots, x_{n-1}$ is a monotonically decreasing sequence or condition 2: there exists a cyclic shift of the sequence $x_0, x_1, x_2, x_3, x_4, \dots, x_{n-1}$ such that the resulting sequence satisfies the condition 1.

Let us take a few examples of bitonic sequence:

B1= 4,7,8,9,11,6,3,2,1 is bitonic sequence

B2= 12,15,17,18,19,11,8,7,6,4,5 is bitonic sequence

In order to provide a solution to such a bitonic sequence, various stages of comparators are required. The basic approach followed for solving such a problem is as given:



Let us say we have a bitonic sequence $X = x_0, x_1, x_2, x_3, x_4, \dots, x_{n-1}$ with the property that first $n/2$ elements are monotonically increasing elements are monotonically increasing like $x_0 < x_1 < x_2 < x_3 < x_4 \dots < x_{n/2-1}$ and other numbers are monotonically decreasing as $x_{n/2} > x_{n/2+1} > \dots > x_{n-1}$. Thereafter, these patterns are compared with the help of a comparator as follows:

$$Y = \min(x_0, x_{n/2}), \min(x_1, x_{n/2+1}), \min(x_2, x_{n/2+2}), \dots, \min(x_{n/2-1}, x_{n-1})$$

$$Z = \max(x_0, x_{n/2}), \max(x_1, x_{n/2+1}), \max(x_2, x_{n/2+2}), \dots, \max(x_{n/2-1}, x_{n-1})$$

After the above discussed iteration, the two new bitonic sequences are generated and the method is known as bitonic split. Thereafter, a recursive operation on these two bitonic sequences is processed until we are able to achieve the sorted list of elements. The exact algorithm for sorting the bitonic sequence is as follows:

Sort_Bitonic (X)

// Input: N Numbers following the bitonic property
 // Output: Sorted List of numbers

- 1) The Sequence, i.e. X is transferred on the input lines of the combinational circuit which consists of various set of comparators.
- 2) The sequence X is splitted into two sub-bitonic sequences say, Y and Z, with the help of a method called bitonic split.
- 3) Recursively execute the bitonic split on the sub sequences, i.e. Y and Z, until the size of subsequence reaches to a level of 1.
- 4) The sorted sequence is achieved after this stage on the output lines.

With the help of a diagram to illustrate the concept of sorting using the comparators.

Example 1: Consider a unsorted list having the element values as {3,5,8,9,10,12,14,20,95,90,60,40,35,23,18,0}

This list is to be sorted in ascending order. To sort this list, in the first stage comparators of order 2 (i.e. having 2 input and 2 output) will be used. Similarly, 2nd stage will consist of 4, input comparators, 3rd stage 8 input comparator and 4th stage a 16 input comparator.

Let us take an example with the help of a diagram to illustrate the concept of sorting using the comparators (see Figure 6).

3	+BM(2)	3	+BM(4)	3	+BM(8)	3	+BM(16)	0	
5		5		5		5		5	3
8	-BM(2)	9	8	8		8		5	
9		8	9	9		9		8	
10	+BM(2)	10	-BM(4)	20	-BM(8)	10		+BM(16)	9
12		12		14		12			12
14	-BM(2)	20	12	14		14			12
20		14	10	20		20			14
95	+BM(2)	90	+BM(4)	40	-BM(8)	95	+BM(16)		18
90		95		60		90			90
60	-BM(2)	60	90	60		60			23
40		40	95	40		40			35
35	+BM(2)	23	-BM(4)	35	-BM(8)	35		+BM(16)	40
23		35		23		23			23
18	-BM(2)	18	18	18		18			90
0		0	0	0		0			95

Figure 6: Sorting using Combinational Circuit



Analysis of Sort_Bitonic(X)

The bitonic sorting network requires $\log n$ number of stages for performing the task of sorting the numbers. The first $n-1$ stages of the circuit are able to sort two $n/2$ numbers and the final stage uses a +BM (n) comparator having the depth of $\log n$. As running time of the sorting is dependent upon the total depth of the circuit, therefore it can be depicted as:

$$D(n) = D(n/2) + \log n$$

Solving the above mentioned recurrence relation

$$D(n) = (\log^2 n + \log n)/2 = O(\log^2 n)$$

Thus, the complexity of solving a sorting algorithm using a combinational circuit is $O(\log^2 n)$.

Another famous sorting algorithm known as merge sort based algorithm can also be depicted / solved with the help of combinational circuit. The basic working of merge sort algorithm is discussed in the next section

1.8 MERGE SORT CIRCUIT

First, divide the given sequence of n numbers into two parts, each consisting of $n/2$ numbers. Thereafter, recursively split the sequence into two parts until each number acts as an independent sequence. Consequently, the independent numbers are first sorted and recursively merged until a sorted sequence of n numbers is not achieved.

In order to perform the above-mentioned task, there will be two kinds of circuits which would be used in the following manner: the first one for sorting and another one for merging the sorted list of numbers.

Let us discuss the sorting circuit for merge sort algorithm. The sorting Circuit.

Odd-Even Merging Circuit

Let us firstly illustrate the concept of merging two sorted sequences using a odd-even merging circuit. The working of a merging circuit is as follows:

- 1) Let there be two sorted sequences $A=(a_1, a_2, a_3, a_4, \dots, a_m)$ and $B=(b_1, b_2, b_3, b_4, \dots, b_m)$ which are required to be merged.
- 2) With the help of a merging circuit ($m/2, m/2$), merge the odd indexed numbers of the two sub sequences i.e. $(a_1, a_3, a_5, \dots, a_{m-1})$ and $(b_1, b_3, b_5, \dots, b_{m-1})$ and thus resulting in sorted sequence $(c_1, c_2, c_3, \dots, c_m)$.
- 3) Thereafter, with the help of a merging circuit ($m/2, m/2$), merge the even indexed numbers of the two sub sequences i.e. $(a_2, a_4, a_6, \dots, a_m)$ and $(b_2, b_4, b_6, \dots, b_m)$ and thus resulting in sorted sequence $(d_1, d_2, d_3, \dots, d_m)$.
- 4) The final output sequence $O=(o_1, o_2, o_3, \dots, o_{2m})$ is achieved in the following manner: $o_1 = a_1$ and $o_{2m} = b_m$. In general the formula is as given below: $o_{2i} = \min(a_{i+1}, b_i)$ and $o_{2i+1} = \max(a_{i+1}, b_i)$ for $i=1, 2, 3, 4, \dots, m-1$.

Now, let us take an example for merging the two sorted sequences of length 4, i.e., $A=(a_1, a_2, a_3, a_4)$ and $B=(b_1, b_2, b_3, b_4)$. Suppose the numbers of the sequence are $A=(4, 6, 9, 10)$ and $B=(2, 7, 8, 12)$. The circuit of merging the two given sequences is illustrated in *Figure 7*.

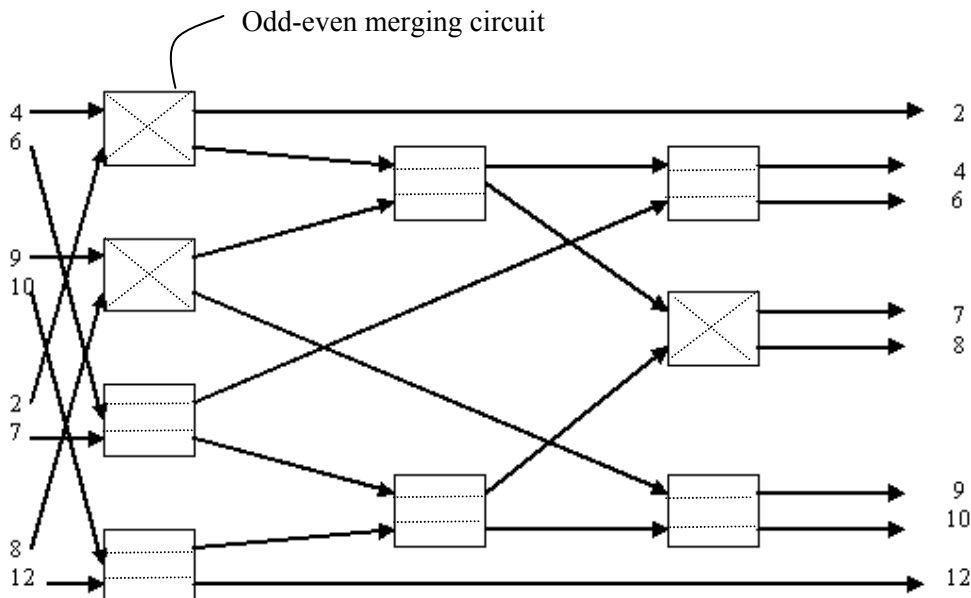


Figure 7: Merging Circuit

Sorting Circuit along with Odd-Even Merging Circuit

As we already know, the merge sort algorithm requires two circuits, i.e. one for merging and another for sorting the sequences. Therefore, the sorting circuit has been derived from the above-discussed merging circuit. The basic steps followed by the circuit are highlighted below:

- i) The given input sequence of length n is divided into two sub-sequences of length $n/2$ each.
- ii) The two sub sequences are recursively sorted.
- iii) The two sorted sub sequences are merged $(n/2, n/2)$ using a merging circuit in order to finally get the sorted sequence of length n .

Now, let us take an example for sorting the n numbers say 4,2,10,12 8,7,6,9. The circuit of sorting + merging given sequence is illustrated in *Figure 8*.

Analysis of Merge Sort

- i) The width of the sorting + merging circuit is equal to the maximum number of devices required in a stage is $O(n/2)$. As in the above figure the maximum number of devices for a given stage is 4 which is $8/2$ or $n/2$.
- ii) The circuit contains two sorting circuits for sorting sequences of length $n/2$ and thereafter one merging circuit for merging of the two sorted sub sequences (see stage 4th in the above figure). Let the functions T_s and T_m denote the time complexity of sorting and merging in terms of its depth.

The T_s can be calculated as follows:

$$T_s(n) = T_s(n/2) + T_m(n/2)$$

$$T_s(n) = T_s(n/2) + \log(n/2) ,$$

Therefore, $T_s (n)$ is equal to $O(\log^2 n)$.

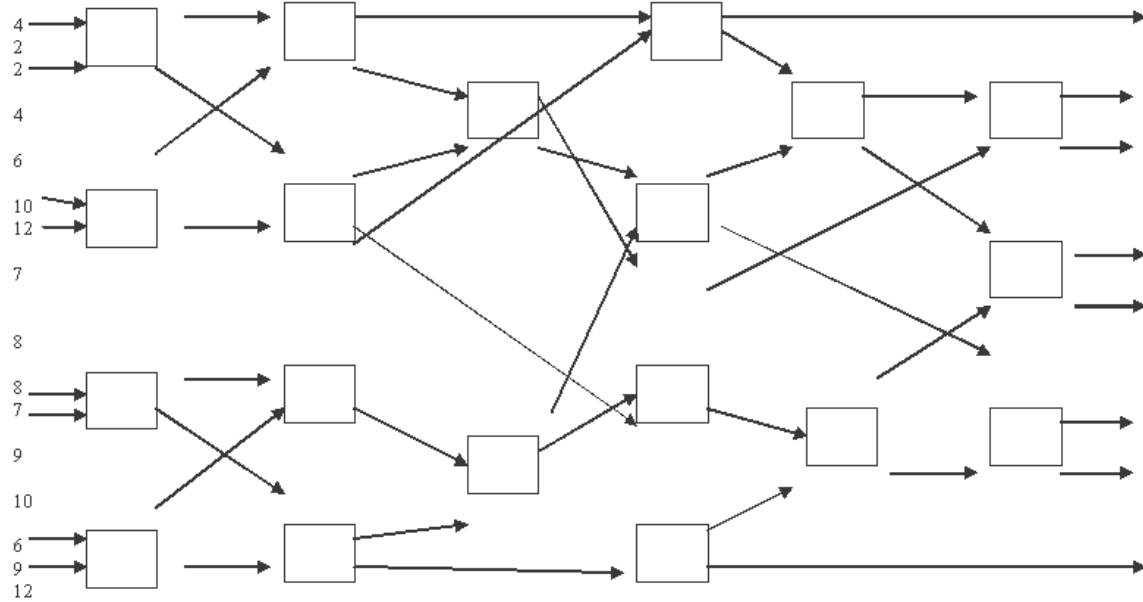


Figure 8: Sorting + Merging Circuit

1.9 SORTING USING INTERCONNECTION NETWORKS

The combinational circuits use the comparators for comparing the numbers and storing them on the basis of minimum and maximum functions. Similarly, in the interconnection networks the two processors perform the computation of minimum and maximum functions in the following way:

Let us consider there are two processors p_i and p_j . Each of these processors has been given as input an element of the sequence, say e_i and e_j . Now, the processor p_i sends the element e_i to p_j and consequently processor p_j sends e_j to p_i . Thereafter, processor p_i calculates the minimum of e_i and e_j i.e., $\min(e_i, e_j)$ and processor p_j calculates the maximum of e_i and e_j , i.e. $\max(e_i, e_j)$. The above method is known as compare-exchange and it has been depicted in the Figure 9.

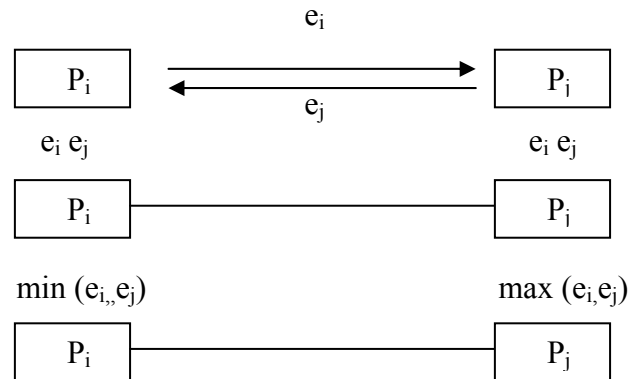


Figure 9: Illustration of Exchange-cum-Comparison in interconnection networks

The sorting problem selected is bubble sort and the interconnection network can be depicted as n processors interconnected with each other in the form of a linear array as



shown in *Figure 10*. The technique adopted for solving the bubble sort is known as odd-even transposition. Assume an input sequence is $B=(b_1, b_2, b_3, b_4, \dots, b_n)$ and each number is assigned to a specific processor. In the odd-even transposition, the sorting is performed with the help of two phases called odd phase and even phase. In the odd phase, the elements stored in $(p_1, p_2), (p_3, p_4), (p_5, p_6), \dots, (p_{n-1}, p_n)$ are compared according to the Figure and subsequently exchanged if required i.e. if they are out of order. In the even phase, the elements stored in $(p_2, p_3), (p_4, p_5), (p_6, p_7), \dots, (p_{n-2}, p_{n-1})$ are compared according to the Figure and subsequently exchanged if required, i.e. if they are out of order. Remember, in the even phase the elements stored in p_1 and p_n are not compared and exchanged. The total number of phases required for sorting the numbers is n i.e. $n/2$ odd phases and $n/2$ even phases. The algorithmic representation of the above discussed odd-even transposition is given below:

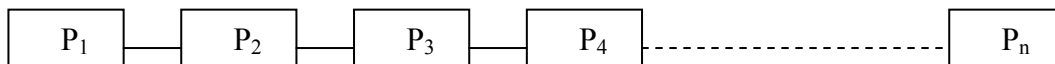


Figure 10: Interconnection network in the form of a Linear Array

Algorithm: Odd-Even Transposition

//Input: N numbers that are in the unsorted form
 //Assume that element b_i is assigned to p_i

```

for I=1 to N
{
    If (I%2 != 0) //i.e Odd phase
    {
        For j=1,3,5,7,.....2*n/2-1
        {
            Apply compare-exchange( $P_j, P_{j+1}$ ) //Operation is performed in parallel
        }
    }
    else // Even phase
    {
        For j=2,4,6,8,.....2*(n-1)/2-1
        {
            Apply compare-exchange( $P_j, P_{j+1}$ ) //Operation is performed in parallel
        }
    }
    I++
}
    
```

Let us take an example and illustrate the odd-even transposition algorithm (see Figure 11).

Analysis

The above algorithm requires one ‘for loop’ starting from $I=1$ to N , i.e. N times and for each value of I , one ‘for loop’ of J is executed in parallel. Therefore, the time complexity of the algorithm is $O(n)$ as there are total n phases and each phase performs either odd or even transposition in $O(1)$ time.

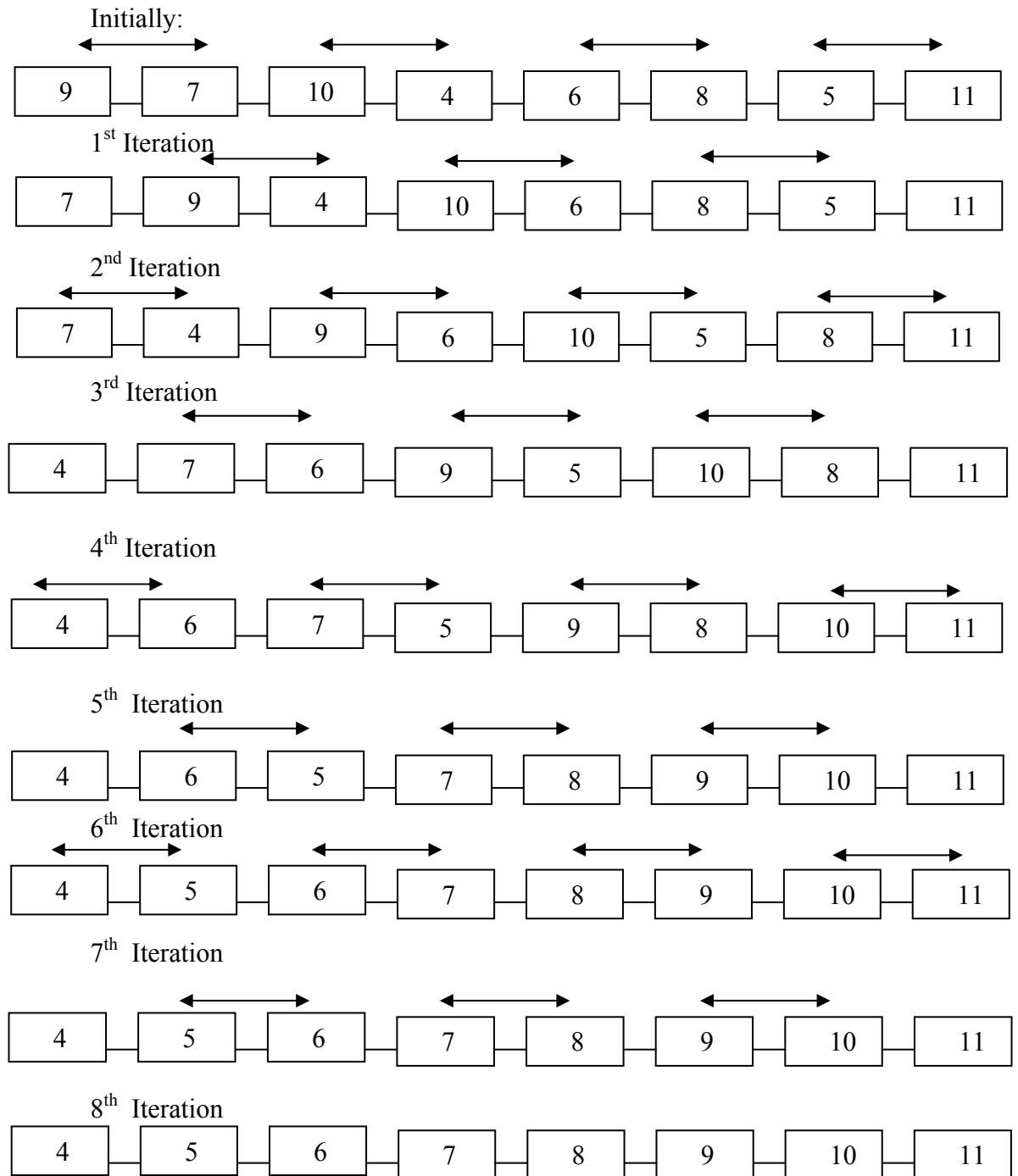


Figure 11: Example

☞ **Check Your Progress 2**

1) Which circuit has a time complexity of $O(n)$ for sorting the n numbers?

- 1) Sort-Merge Circuit
- 2) Interconnection Networks
- 3) Combinational Circuits
- 4) None of the above



2) Which circuit has a time complexity of $O(\log n^2)$ for sorting the n numbers?

- 1) PRAM
- 2) Interconnection Networks
- 3) Combinational Circuits
- 4) Both 1 and 3.

3) Explain the concept of sorting in the combinational circuits

.....

.....

.....

.....

1.10 MATRIX COMPUTATION

In the subsequent section, we shall discuss the algorithms for solving the matrix multiplication problem using the parallel models.

Matrix Multiplication Problem

Let there be two matrices, $M1$ and $M2$ of sizes $a \times b$ and $b \times c$ respectively. The product of $M1 \times M2$ is a matrix O of size $a \times c$.

The values of elements stored in the matrix O are according to the following formulae:

$$O_{ij} = \text{Summation } x \text{ of } (M1_{ix} * M2_{xj}) \text{ } x=1 \text{ to } b, \text{ where } 1 < i < a \text{ and } 1 < j < c.$$

Remember, for multiplying a matrix $M1$ with another matrix $M2$, the number of columns in $M1$ must equal number of rows in $M2$. The well known matrix multiplication algorithm on sequential computers takes $O(n^3)$ running time. For multiplication of two 2×2 , matrices, the algorithm requires 8 multiplication operations and 4 addition operations.

Another algorithm called Strassen algorithm has been devised which requires 7 multiplication operations and 14 addition and subtraction operations. The time complexity of Strassen's algorithm is $O(n^{2.81})$. The basic sequential algorithm is discussed below:

Algorithm: Matrix Multiplication

Input// Two Matrices $M1$ and $M2$

```

For I=1 to n
  For j=1 to n
  {
    Oij = 0;
    For k=1 to n
      Oij = Oij + M1ik * M2kj
    End For
  }
End For
End For
    
```

Now, let us modify the above discussed matrix multiplication algorithm according to parallel computation models.



1.11 CONCURRENTLY READ CONCURRENTLY WRITE (CRCW)

It is one of the models based on PRAM. In this model, the processors access the memory locations concurrently for reading as well as writing operations. In the algorithm, which uses CRCW model of computation, n^3 number of processors are employed. Whenever a concurrent write operation is performed on a specific memory location, say m , then there are chances of occurrence of a conflict. Therefore, the write conflicts i.e. (WR, RW, WW) have been resolved in the following manner. In a situation when more than one processor tries to write on the same memory location, the value stored in the memory location is always the sum of the values computed by the various processors.

Algorithm Matrix Multiplication using CRCW

```

Input// Two Matrices M1 and M2
For I=1 to n      //Operation performed in PARALLEL
  For j=1 to n    //Operation performed in PARALLEL
    For k=1 to n  //Operation performed in PARALLEL
      Oij = 0;
      Oij = M1ik * M2kj
    End For
  End For
End For

```

The complexity of CRCW based algorithm is $O(1)$.

1.12 CONCURRENTLY READ EXCLUSIVELY WRITE (CREW)

It is one of the models based on PRAM. In this model, the processors access the memory location concurrently for reading while exclusively for writing operations. In the algorithm which uses CREW model of computation, n^2 number of processors have been attached in the form of a two dimensional array of size $n \times n$.

Algorithm Matrix Multiplication using CREW

```

Input// Two Matrices M1 and M2

For I=1 to n      //Operation performed in PARALLEL
  For j=1 to n    //Operation performed in PARALLEL
    {
      Oij = 0;
      For k=1 to n
        Oij = Oij + M1ik * M2kj
      End For
    }
  End For
End For

```

The complexity of CREW based algorithm is $O(n)$.



☞ **Check Your Progress 3**

- 1) Which of the models has a complexity of $O(n)$ for matrix multiplication?
 - 1) RAM
 - 2) Interconnection Networks
 - 3) CRCW
 - 4) CREW
- 2) Which of the models has a complexity of $O(1)$ for matrix multiplication?
 - 1) RAM
 - 2) Interconnection Networks
 - 3) CRCW
 - 4) CREW
- 3) Explain the algorithm for matrix multiplication in sequential circuits.

.....

.....

.....

.....

1.13 SUMMARY

In this chapter, we have discussed the various topics pertaining to the art of writing parallel algorithms for various parallel computation models in order to improve the efficiency of a number of numerical as well as non-numerical problem types. In order to evaluate the complexity of parallel algorithms there are mainly three important parameters, which are involved i.e., 1) Time Complexity, 2) Total Number of Processors Required, and 3) Total Cost Involved. Consequently, we have discussed the various computation models for parallel computers, e.g. combinational circuits, interconnection networks, PRAM etc. A combinational circuit can be defined as an arrangement of logic gates with a set of m input lines and a set of n output lines. In the interconnection networks, the N processors can communicate with each other through direct links. In the interconnection networks, each processor has an independent local memory. In the PRAM, it contains n processors, a single shared memory module being shared by all the N processors and which also acts as a communication medium for the processors. In order to connect the N processors with the single shared memory, a component called Memory Access Unit (MAU) is used for accessing the shared memory. Subsequently, we have discussed and applied these models on few numerical problems like sorting and matrix multiplication. In case of sorting, initially a combinational circuit was used for sorting. A bitonic sequence was given as an input to a combinational circuit consisting of a set of comparators interconnected with each other. The complexity of sorting using Combinational Circuit is $O(\log^2 n)$. Another famous sorting algorithm known as merge sort based algorithm can also be depicted / solved with the help of the combinational circuit. The complexity of merge-sort using Combinational Circuit is $O(\log^2 n)$. The interconnection network can be used for solving the sorting problem known as bubble sort. The interconnection network can be depicted as n processors interconnected with each other in the form of a linear array. The complexity of bubble sort using interconnection network is $O(n)$. The well known matrix multiplication algorithm on sequential computers take $O(n^3)$ running time and strassen algorithm take $O(n^{2.81})$. In the present study, we have discussed two models based on PRAM for solving the matrix multiplication problem. In CRCW model, the processors access the memory location concurrently for reading as well as for writing operation. In the algorithm which uses CRCW model of computation, n^3 number of processors are employed. The complexity of CRCW based algorithm is $O(1)$. In CREW model, the processors access the memory



location concurrently for reading while exclusively for writing operation. In the algorithm which uses CREW model of computation, n^2 number of processors have been attached in the form of a two dimensional array of size $n \times n$. The complexity of CREW based algorithm is $O(n)$.

1.14 SOLUTIONS/ANSWERS

☞ Check Your Progress 1

1: 1

2: 3

3: 3

4: The fundamental parameters required for the analysis of parallel algorithms are as under:

1. Time Complexity;
2. The Total Number of Processors Required; and
3. The Cost Involved.

☞ Check Your Progress 2

1: 2

2: 4

3: Each input line of the combinational circuit represents an individual element of the string say, X_i , and each output line results in the form of a sorted list. In order to achieve the above-mentioned task, a comparator is employed for processing.

☞ Check Your Progress 3

1: 4

2: 3

3: The values of elements stored in matrix O are according to the following formulae:

$$O_{ij} = \text{Summation of } (M1_{ix} * M2_{xj}) \quad x=1 \text{ to } b, \text{ where } 1 < i < a \text{ and } 1 < j < c$$

1.15 REFERENCES/FURTHER READINGS

- 1) Cormen T. H., *Introduction to Algorithms*, Second Edition, Prentice Hall of India, 2002.
- 2) Rajaraman V. and Siva Ram Murthy C. *Parallel Computers - Architecture and Programming*, Second Edition, Prentice Hall of India, 2002.
- 3) Xavier C. and Iyengar S. S. *Introduction to Parallel Algorithm*.