Parallel Algorithm for Adding Long Integers

Sanjeev Gangwar[1], Prashant Kumar Yadav[2]
Department of Computer Application[1] , Department of Computer Science & Engineering[2]
VBS Purvanchal University,Jaunpur, India

***Abstract*:In the new era of computing, people want to spend less time for any type of computation. In this paper, we are going to present a new algorithm for addition in parallel environment for PRAM model. Hence we are using more than one processor for performing our addition. The number of processor used depends upon the maximum number of digits given as input to the algorithm. In this algorithm, first of all we store a given number in a one dimensional array. Hence for each number we use individual one dimensional array. For each address of array, we allot an individual processor for addition. Hence the number of processor is equal to the size of largest array. The results of every processor is stored in a new array called "Resulting array", on the last step of algorithm, we use an extra processor for adding each elements of resulting array .In this way, we can easily perform a largest calculation of addition in single unit of time. This algorithm takes a very less time, and space. Hence it is very optimal algorithm for parallel addition. We will implement the idea of algorithm is c programming language which provides better environment to implement our idea.**

***Keywords*: Parallel processing, PRAM model, CREW, SIMD architecture.**

I. INTRODUCTION

The proposed paper is designed for the larger and typical addition in parallel environment. The intended audiences are those people who want to spend very less time for calculation and can spend more memory. The idea behind this algorithm is as given below: The designed algorithm uses N processors, depends upon the largest array size. The one dimensional array is used for storing a given input number, where the elements of array represents the digits, present in given input number. The single and individual array is used for storing a single number; hence the number of array is equal to the number of given input numbers. For adding the elements of i_{th} address space of each and every, array, we use a single processor.

II. PARALLEL COMPUTING

Parallel computing is a method for solving the longest calculation. In this more than one processor are used and worked concurrently. A parallel computer is a multiple processor computer that is capable of parallel processing. Parallel processing is information processing that emphasize the concurrent manipulation of data elements belonging to on ore more processes solving a single problem. The advantage of parallel processing over serial execution is to speed up the processing capability and increase throughput. Parallel processing is established by distributing the data among the multiple functional units.

A. PRAM model

To properly design a parallel algorithm, alternative model of computation underlying the parallel computer is required. The extension of RAM (Random access Machine)model for sequential algorithm is referred as PRAM (Parallel Random Access Machine) model. This PRAM model used for parallel computations is described as follows:

- A PRAM (Parallel Random Access Machine) model consists of a control unit, Global memory and unbounded set of processors with their own private memory.
- Every processor has a unique index which is used to enable or disable the processor or identify with memory location it access.
- All communication is performed by a shared memory.
- All active processors execute identical instruction.
- Processors have read, compute, and write phases that happen synchronously.
- PRAM computer's instruction will execute in 3-phase cycle:
  - ✓ Read
  - ✓ Local computation
  - ✓ Write
- PRAM computation starts with a single active processing element and the input stored in global memory. During every step of computation an active, enabled processor(which readsinput value from single private or global memory location) performs a single RAM operation and output is written into one local or global memory location.
- On the other hand, during a computation process, a processor have a capability to activate another

processor on their need. All active, enabled processors must execute the some instruction, on different memory locations. The computation process ends with the halt of last active processor.

- Cost of PRAM computation =parallel time complexity * number of processors used. where parallel time complexity= time elapsed in executing all the instruction simultaneously.

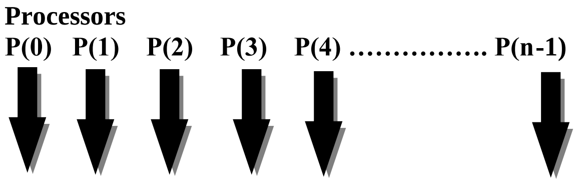
B. Classification of PRAM

The ability to perform operations of PRAM depends on the accessing method to the shared memory location. PRAMs are classified based on Read/ Write abilities. According to read or write conflicts i.e., when two or more processorsare used to read or write to the same global memory location, four memory update are possible:

- **Exclusive Read (ER).** Allow at most one processor to read from any memory location .This is most restrictive policy.
- **Exclusive Write (EW).**Allow at most one processor to write into a memory location at a time.
- **Concurrent Read(CR).** Allow multiple processors to read the some information from the some memory location at a time.
- **Concurrent Write (CW) .**Allow simultaneous write to the same memory location .In order to avoid confusion, some policy must be setup to resolve the write conflicts.

III. ALGORITHM FOR SLAVE PROCESSORS

```
SUM (CREW, PRAM)
Initial Condition:
List of n>=1 element stored in one dimensional  array.
Final Condition: Sum of element stored in RA[i].
Begin
Spawn ($P_0$,$P_1$,$P_{2,.........,}$$P_{n-1}$)
For all $P_i$ where (i=0 to n-1)
do
{
RA[i] A[i]+B[i]+C[i]+…………+N[i]
}
end
endfor
end
```



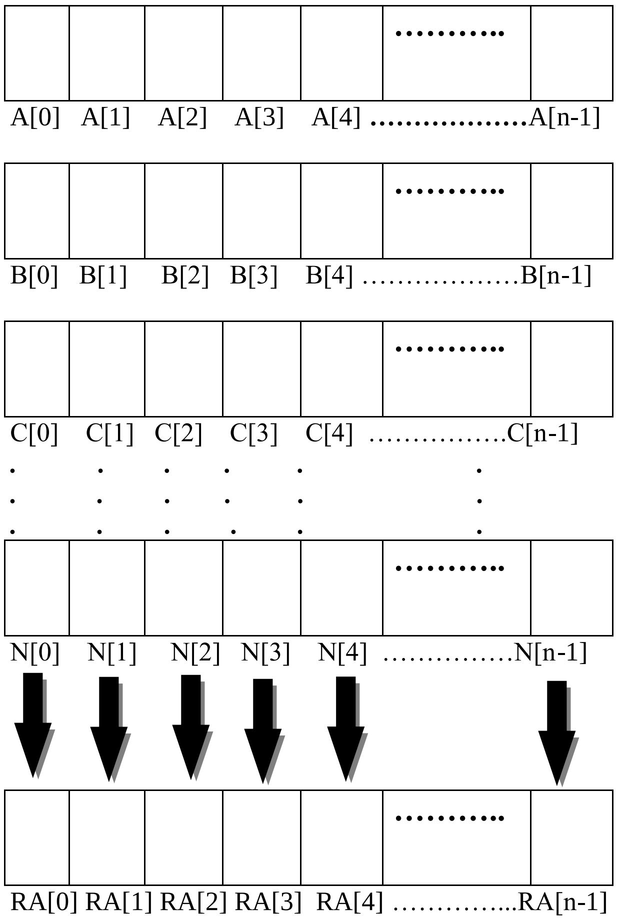
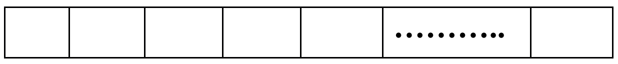

A[0] A[1] A[2] A[3] A[4] ………………A[n-1]
B[0] B[1] B[2] B[3] B[4] ………………B[n-1]
C[0] C[1] C[2] C[3] C[4] …………….C[n-1]
N[0] N[1] N[2] N[3] N[4] ……………N[n-1]
RA[0] RA[1] RA[2] RA[3] RA[4] …………...RA[n-1]

Figure 1 Structure of working of Slave Processors.

A. *Algorithm for resulting processor*

```
SUM(EREW, PRAM)
Initial Condition: List of n >=1element stored in A[0……n-1]
Final Condition: Sum of element stored in A[0]
Global Variables: n, A[0…….n-1],j.
Begin
Spawn ($P_0$,$P_1$,$P_{2,.........}$P $_{n/2-1}$)
For all Pi where 0<= i <=n/2-1
do
For j=0 to [logn]-1
do
If i modulo$2_j$ =0 and 2i+$2_j$ < n then
A[2i] A[2i] + A[2i+$2_j$]
Endif
Endfor
Endfor
End
```



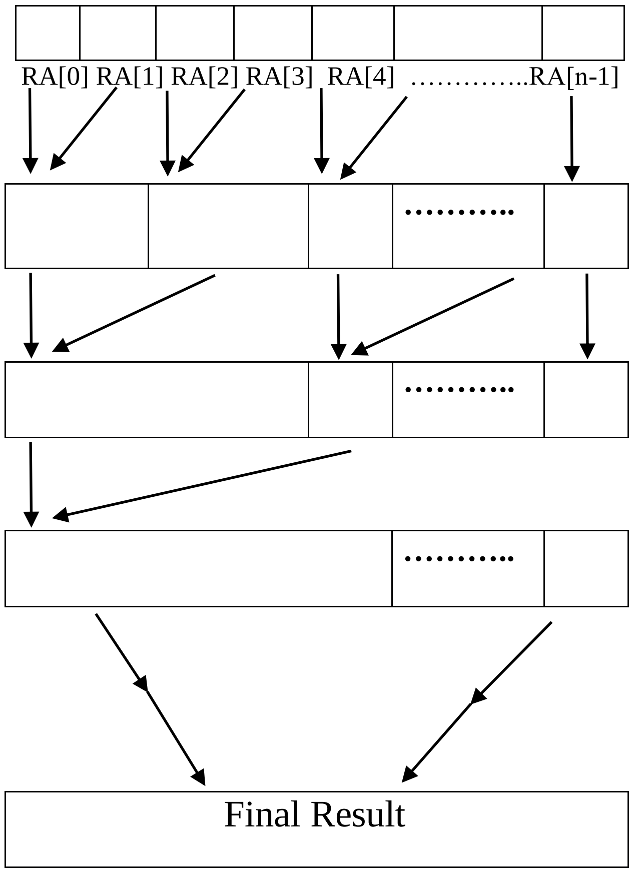


Figure 2: Structure for working of Resulting Processor

B. *Time complexity*

The time complexity for the algorithm of slave processors is order of n. and the time complexity of resulting processor algorithm is order of log(n).Hence the overall time complexity of these algorithms is order of (n + log(n)).

C. *Cost optimality*

Hear, if we calculate the cost of solving a problem on a single processing element then it will be the execution time of the fastest known sequential algorithm. Now if we talk about parallel computing, then cost optimal parallel algorithm is an algorithm for which the cost have the same complexity class as an optimal sequential algorithm.

IV. CONCLUSION

In this paper we have represented a new way of adding long digit numbers with the use of sequential and parallel both algorithms. Here in sequential algorithm, more than processors are used to perform addition operation. The result of each processor is stored in new Resulting array. After this, At second step, the element of Resulting array are given as input to parallel addition algorithm.

V. FUTURE SCOPE

This paper is proposed to represent the addition algorithm for long digit numbers using both sequential and parallel approach. but the time complexity of proposed algorithm is very high, so we can work in future to minimize the time complexity of the algorithm using PRAM additional algorithms and we can also work on this to minimize the number of processors working at a time. Hence we can also work on it to minimize the cost of this algorithm.

REFERENCES

[1] Parallel Computing, "Michel .J.Queen" Pearson .
[2] Parallel Algorithms "S.J. Akl " TMH.
[3] Design and analysis of algorithms, "J.D.Ullman" Pearson.
[4] Sequential and parallel algorithm for the addition of big integer number "Youssef Bassil, Aziz Barbar" *International Journal of Computational Science Vol.4 No.1 pp.52-69, 2010.*
[5] B. fagin fast addition of large integers. IEEE transactions on computer.
[5] Lidong Zhou "Algorithm Concept" http://www.cs.cornell.edu/home/ldzhou/algo.pdf