SOCD Sort on Centralized Diamond Architecture

Kamal Jadidy Aval, Masumeh Damrudi*
Department of Computer Science, Firoozkooh Branch, Islamic Azad University, Firoozkooh, Iran.

* Corresponding author. Email: m.damrudi@gmail.com

Manuscript submitted November 10, 2014; accepted September 8, 2015.
doi: 10.17706/ijcce.2016.5.4.246-252

Abstract: Parallel sorting is a technique which researchers have studied from the time parallelism was proposed as a way of making fast algorithms. Different parallel sorting techniques on different architectures have been studied for many years. Sorting is one of the most important operations in different algorithms. This paper presents another sort on Centralized Diamond architecture which sorts unsorted lists of length L. The theoretical analysis of the algorithm specifies that the algorithm needs $o(log\, n)$ time complexity on PRAM EREW.

Key words: Sorting, centralized diamond, SIMD, EREW PRAM.

1. Introduction

Parallelizing can save time and solve very large problems in computational tasks. Parallelism provides concurrency which is used to solve problems such as time consuming sort. Some sorting algorithms are given in [1], [2]. Parallel architectures are made up of multiple processors and memory modules that ate connected together using some interconnection networks [3]. Interconnection network is an important core of digital parallel processing systems [4]. The definitions of Diamond are first presented in [5]. There are various parallel algorithms on Diamond architecture with different specifications in [6]-[9].

Centralized Diamond is an SIMD architecture which has four subsection including EREW, ERCW, CREW, and CRCW [10]. There are lots of processors that provide a set of Single Instruction Multiple Data (SIMD) instructions including the VMX instruction set or the SSE instruction set. Such processors can do multiple operations on multiple blocks of data in parallel to speedup CPU intensive parts of applications which includes a wide variety of applications. Availability of data parallelism degree in one instruction is an obvious advantage of the SIMD [11]. Centralized Diamond has $7/4n$+1 processor elements. The sorting algorithm presented in this paper is on unsorted lists of input data with length of L.

In this algorithm, the k-merge algorithm which sorts k sorted list with $O(\log n)$ on EREW PRAM [12] is employed. The PRAM is used widely for parallel computation. It was developed to do parallel computing operations like the RAM does for sequential computing and ignores the communication issues [13].

The rest of the article is organized as following. In the next section, interconnection networks are summarized. After that, the proposed algorithm is presented. Then, the time complexity and the cost of the proposed algorithm are discussed. Finally the conclusion is drawn.

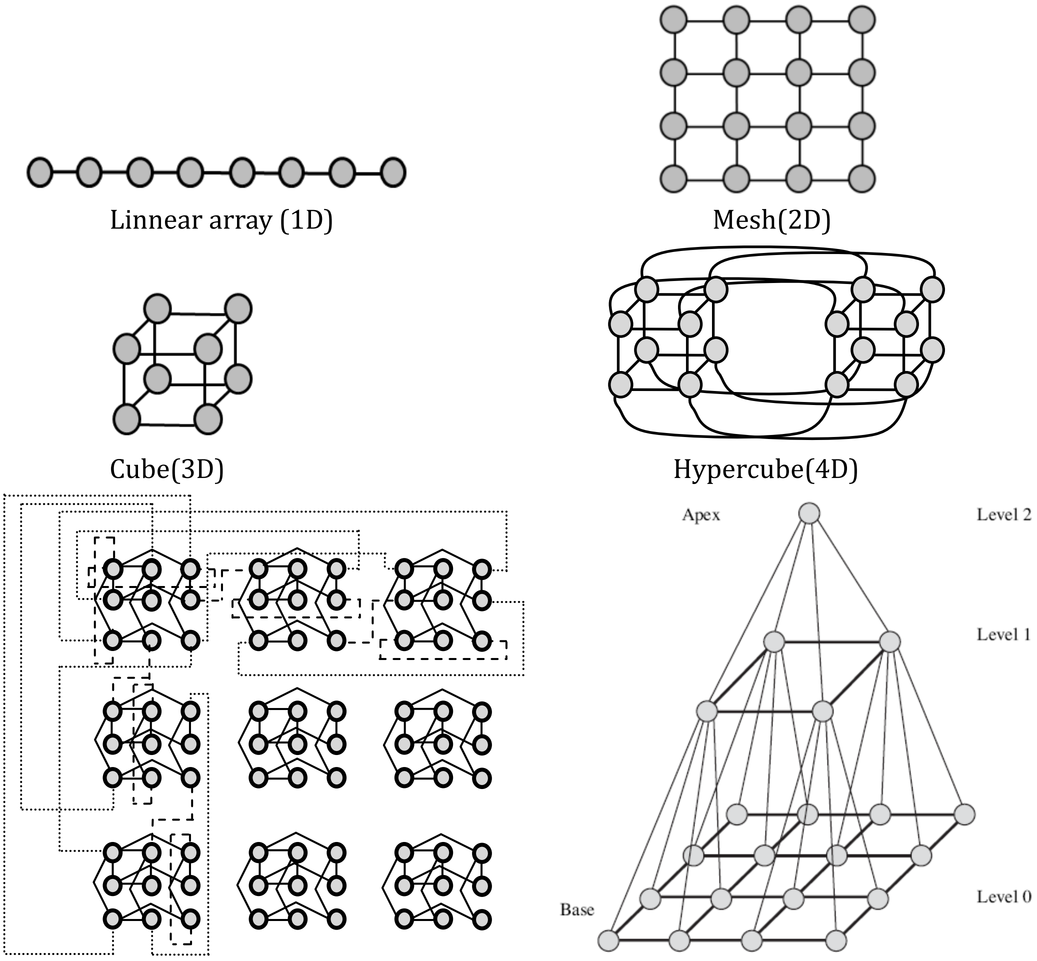
2. Interconnection Networks

Interconnection networks are divided into static and dynamic. Dynamic interconnection networks are also called indirect [14]. In dynamic interconnections, the connections are set up on the fly according to the needs. Based on the interconnection scheme, dynamic interconnections are divided into switch-based and bus-based.

There are different kinds of static interconnection networks including completely connected network (CCN) which is expensive due to many interconnections and limited connected network (LCN). The latter is divided into one-dimensional (1D), two-dimensional (2D), three-dimensional (3D), four-dimensional (4D), and hybrid. As opposed to the dynamic INs, the static interconnection networks have fixed connections.

In CCNs, all PEs have one direct path to all other PEs. Therefore, they are expensive because of the links which are in need. The links' count in a CCN with N PEs is $N(N\text{-}1)/2$, which is $O(N^2)$ [3]. In these interconnection networks, the diameter is one and the degree of this topology is $O(N)$ [3], [15].

In LCNs, linear array is the simplest one–dimensional limited interconnection network. Each PE is connected to its two neighbors. The common 2D interconnection networks include mesh, tree, ring, fat tree, ILLIAC, and torus. The 3D interconnection networks are designed in the form of cube. Cube is one of the most practical interconnection networks among existing interconnection networks. The most well-known 4D interconnection networks is Hypercube topology because of its properties such as strong connectivity, topological symmetry, regularity, and recursive constructions. Hybrid topologies such as mesh of trees and pyramid are combination of the other basic topologies including meshes and trees. Centralized Diamond is a hybrid interconnection network which has appropriate connections between processor elements (PEs). It is composed of tree and ring interconnection network. An example of different types of limited interconnection networks is drawn in Fig. 1.



Linnear array (1D)

Mesh(2D)

Cube(3D)

Hypercube(4D)

Multi Mesh of Trees (Hybrid) [16]

Pyramid(Hybrid)

Fig. 1. Examples of static interconnection network topologies.

One of the most important factors with high impact on IN performance is topology [17]. While there are different architectures with different specifications, researchers choose the topology which is more appropriate for their algorithm considering the specifications of the architecture.

3. Sorting on Centralized Diamond Architecture

Assume that N is the number of processor elements, and L is the length of each unsorted list which is given to the third level of Centralized Diamond. Only one layer of the processor elements of the architecture are dealing with the data at each step. The steps of doing the sort are described in the following.

Step 1: Each PE gets its unsorted list with length of L as it is shown in Fig. 2. This figure is drawn for N=29 and n=16. The length of unsorted lists is considered to be three. The number of processor elements is 7/4n+1. This architecture has four levels where level three has n PEs, level two has n/2 PEs, and level one has n/4 PEs. Finally, level zero there has only one processor element which is called centralized PE. In the third level, each PE has direct connectivity to the processor elements next to it. The relation is as following:

$$\begin{array}{ll} x >= 3n/4 & \\ \Pi_{+1}(x) = x+1 & \text{where } x \bmod 2 \neq 0 \end{array}$$

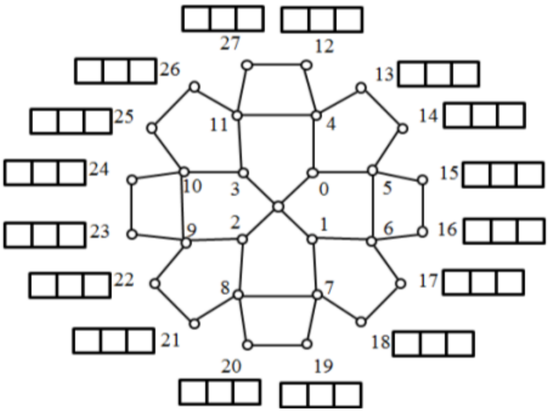


Fig. 2. Entrance of unsorted lists with length L=3 to the third level.

Each PE has adequate registers to store data elements. In addition, each PE has registers for sorting data elements in it. Therefore, each PE in the third level has 2L registers. Each PE in the third level sorts its data elements using parallel merge sort with $O(\log L)$ time complexity where $L << n$. This operation is illustrated in Fig. 3 considering $L = 7$.

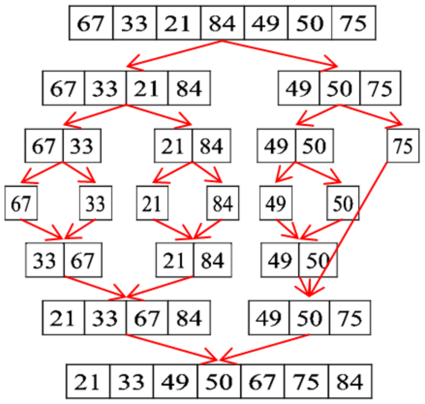


Fig. 3. Parallel merge sort in the PEs of the third level.

Step 2: At the beginning of this step, each PE in the third level has a sorted list. Using k-merge sorted list from [12] with $O(\log n)$ time complexity, the data elements of third level are sorted in the second level of the architecture. In the second level, each processor element is connected directly to the processor element next to it. The relation is as following:

$$\begin{aligned}&n/4 \le x \le 3n/4\\ &\Pi_{+1}(x) = x+1\\ &\Pi_{+n/4}(x) = 2x+n/4\\ &\Pi_{+n/4+1}(x) = 2x+n/4+1\end{aligned} \qquad \text{where } x \bmod 2 \ne 0$$

Fig. 4 shows a small part of the whole architecture to perform this operation including PEs 18, 19, and 7.

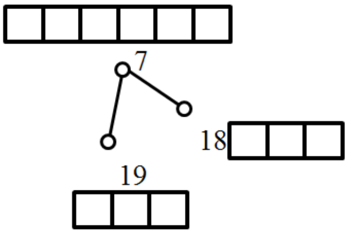


Fig. 4. Merging sorted list of level three using k-merge sort.

Step 3: Each PE in the second level has a sorted list with length of $2L$. The data elements of second level are merged with the k-merge sorted list algorithm and the sorted data are located in the first level. In the first level, each PE is connected directly to the centralized node. The equations are as following:

$$\begin{aligned}&x < n/4\\ &\Pi_{+n/4}(x) = 2x+n/4\\ &\Pi_{+n/4+1}(x) = 2x+n/4+1\end{aligned}$$

To simplify the understanding, a part of the architecture is presented in Fig. 5 including PEs, 16, 17, 18, 19, 6, 7, and 1 which performs the operation of this step.

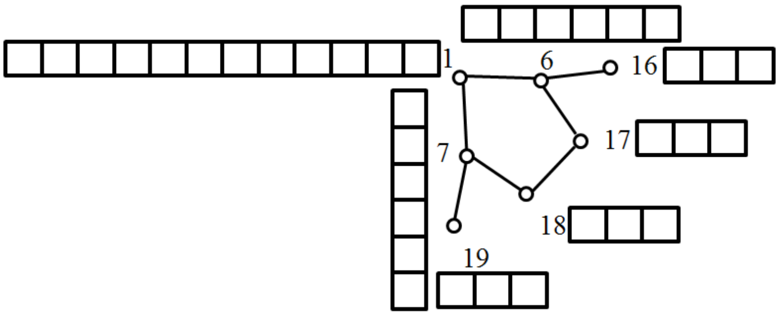


Fig. 5. Merging sorted list of level two using k-merge sort.

Step 4: The centralized PE merge sorted data elements of the first layer using k-merge sort which needs $O(\log n)$. This level has only one processor element. Length of each list is $4L$. The final list in the centralized PE is the sorted list. The length of the list for this example is $16L$.

The flowchart of the proposed algorithm is presented in Fig. 6. Following the steps for unsorted input lists, the sorted list will be achieved.

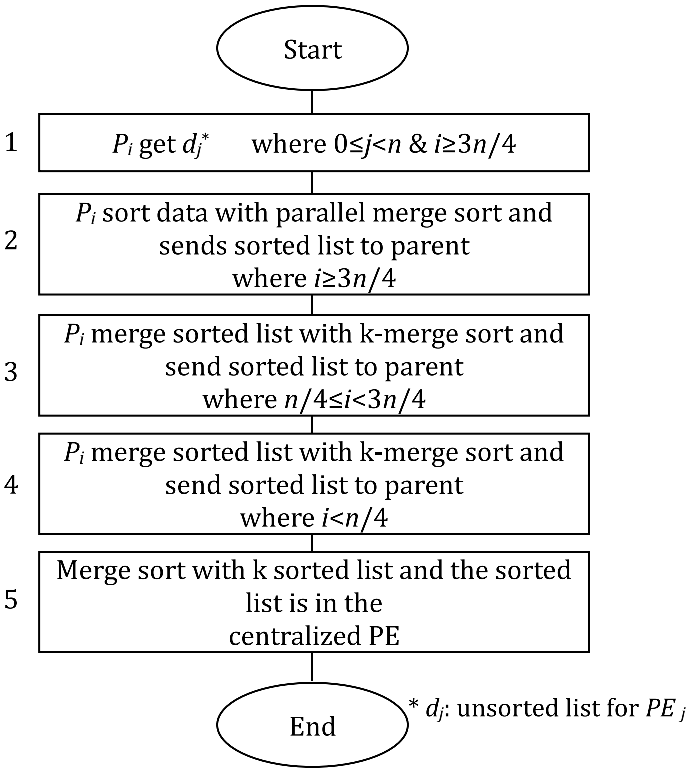

Start
1
P_i get d_j^* where $0 \le j < n$ & $i \ge 3n/4$
2
P_i sort data with parallel merge sort and sends sorted list to parent where $i \ge 3n/4$
3
P_i merge sorted list with k-merge sort and send sorted list to parent where $n/4 \le i < 3n/4$
4
P_i merge sorted list with k-merge sort and send sorted list to parent where $i < n/4$
5
Merge sort with k sorted list and the sorted list is in the centralized PE
End
* d_j: unsorted list for PE_j

Fig. 6. Flowchart of the sorting algorithm.

4. Time Complexity and Cost

In the proposed sorting algorithm and from the flowchart in Fig. 6, step one has a time complexity of $O(\log L)$ and each of steps 2, 3, 4 and 5 has a time complexity of $O(\log n)$. The overall time complexity is $O(\log L) + O(\log n) + O(\log n) + O(\log n)$ where $L << n$. Therefore, the total execution time is $O(\log n)$. The number of processor elements that are used for execution of the algorithm is called p_n. If the processor elements are equal to the number of data elements, all the data can be fed into the system simultaneously. In this algorithm number of data elements are more than PEs, which can be calculated using $p_n = O\left(\frac{7}{4}n+1\right) = O(n) = O(N)$. The time from the start of the first processor element to the end of the last processor element for executing the algorithm is the parallel time. This is the number of basic operations for the longest path. The parallel time in this algorithm is $t_n = \mathrm{O}(\log n)$. The total time which the processor elements use to execute the algorithm is the total cost. If the parallel time is $t(n)$ and the number of processors is $p(n)$, then the total cost will be $C(n) = t(n) \times p(n)$. The total cost is $c_n = \mathrm{O}(n \log n)$.

5. Conclusion

We proposed a sorting algorithm on unsorted input lists using Centralized Diamond architecture. The type of interconnections of this architecture affects the results of sorting in terms of speed and time complexity. The Centralized Diamond architecture is composed of ring and tree which has suitable specifications such as logarithmic diameter. The proposed sorting algorithm needs $O(\log n)$ time complexity. This algorithm has an appropriate order. The architecture is heterogeneous, which means the capabilities of PEs in the leaves are different from the PEs in other levels. This fact does not affect the synchronization of processing because the processor elements are the same at each level, therefore they finish their operation at the same time.

Acknowledgment

This research was supported by Islamic Azad University, Firoozkooh Branch. The authors would like to thank reviewers of this paper for their valuable comments and suggestions that improve the presentation of this paper.

References

[1] Damrudi, M., Salehnamadi, M. R., & Aval, K. J. (2008). ROE sorting on ILLIAC array processor. *Proceedings of the WSEAS International Conference on Applied Computing Conference* (pp. 326-330).

[2] Damrudi, M., & Aval, K. J. (2012). Generalized approach to SOCD sorting on centralized diamond architecture. *Indian Journal of Science and Technology, 5(9)*, 3288-3291.

[3] El-Rewini, H., & El-Barr, M. A. (2005). *Advanced Computer Architecture and Parallel Processing*. USA: John Wiley & Sons Publishing.

[4] Abdullah, M., Abuelrub, E., & Mahafzah, B. (2011). The chained-cubic tree interconnection network. *International Arab Journal of Information Technology, 8(3)*, 334-343.

[5] Damrudi, M., & Aval, K. J. (2009). Diamond architecture with NOD sorting. *Proceedings of IEEE Youth Conference on Information, Computing and Telecommunication* (pp. 431-434).

[6] Damrudi, M., Aval, K. J., & Asadollahi, H. (2010). Centralized diamond architecture, a parallel approach. *Proceedings of the 2nd International Conference on Information and Multimedia Technology* (p. 200).

[7] Damrudi, M., & Aval, K. J. (2010). A new parallel sorting on diamond architecture. *Proceedings of the 4th Conference on European Computing* (pp. 284-287).

[8] Damrudi, M., & Aval, K. J. (2011). Searching data using centralized diamond architecture. *Journal of Communication and Computer, 8(9),* 807-811.

[9] Damrudi, M., & Aval, K. J. (2011). Sorting data elements by SOCD using centralized diamond architecture. *Computer Technology and Application, 2(5),* 374-377.

[10] IBM Corporation. (2007). *SIMD Math Library API Reference Manual*.

[11] Inoue, H., Moriyama, T., Komatsu, H., & Nakatani, T. (2007). AA-sort: A new parallel sorting algorithm for multi-core SIMD processors. *Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques, IEEE Computer Society* (pp. 189-198).

[12] Hayashi T., Nakano K., & Olariu S. (1998). Work-time optimal algorithms on the PRAM*. IEEE Transactions on Parallel and Distributed Systems, 9(3),* 298–302.

[13] Miller, R., & Boxer, L. (2005). *Algorithms Sequential and Parallel: A Unified Approach* (2nd ed.). Hingham, Massachusetts: Computer Engineering Series.

[14] Pimentel, A. (2002). Introduction to parallel architecture — 1st syllabus for the architecture part of the course Introduction Parallel Computing & Architecture. University of Amsterdam.

[15] Culler, D., Singh, J. P., & Gupta, A. (1997). *Parallel Computer Architecture: A Hardware / Software Approach*. *Computer Architecture and Design.* USA: Morgan Kaufmann Publishers.

[16] Rakesh, N., & Chanderwal, N. (2010). K-K sorting on the multi-mesh of trees. *Proceedings of 2nd International Conference on Computer and Automation Engineering*. Singapore: IEEE Computer Society.

[17] Hoseiny-Farahabady, M., Imani, N., & Sarbazi-Azad, H. (2008). Some topological and combinatorial properties of WK-recursive mesh and WK-pyramid interconnection networks*. Journal of Systems Architecture*, *54(10)*, 967-976.

Kamal Jadidy Aval has received his B.Sc. and M.Sc. degrees in computer science (software engineering) from Iran University of Science and Technology and Islamic Azad University, Tehran South branch, Tehran, Iran in 2002 and 2004, respectively. He will receive his PhD degree from Universiti Teknologi Malaysia (UTM), JB, Malaysia in 2015.

He is a lecturer in the Islamic Azad University, Firoozkooh branch since 2006. He has more than 20 papers in various journals and conferences. His research interests are image processing, evolutionary algorithms, WSNs, and MANets.

Masumeh Damrudi has received her B.Sc. and M.Sc. in computer science (software engineering) from Islamic Azad University, Tehran South branch, Tehran, Iran in 2001 and 2004 respectively. She has received her PhD degree from Universiti Teknologi Malaysia (UTM), JB, Malaysia in 2013.

She is a lecturer in the Islamic Azad University, Firoozkooh branch since 2006. She has more than 20 papers in various journals and conferences. Her research interests are parallel processing, cryptography and image processing.