TriBA – A Novel Scalable Architecture for High performance Parallel Computing Applications

HAROON-UR-RASHID, SHI FENG, JI WEIXING, QIAO BAOJUN
Department of Computer Science and Engineering
Beijing Institute of Technology
Beijing 100081,
P.R.CHINA

Abstract: - This paper presents *TriBA* - a new idea in multiprocessor architectures. It is believed to be a high performance parallel computing architecture. The root of this idea is based on the concept that "Complex problems can be decomposed to three relatively independent sub-problems, which are data Processing, data Management and data Communication". Triplet Based Architecture (TriBA) is a real scalable architecture, featuring fractal nature for computers, is proposed in this paper. *TriBA* is a new solution for computer architecture, which is suitable for sophisticated embedded applications with multiple concurrent processing centers. The characteristics of this architecture are its great modularity, flexibility and scalability to meet the real-time signal processing demands in future telecommunication and multimedia systems.

Keywords: multiprocessor architecture, Von Neumann architecture, Sierpinski gasket, hierarchical interconnection, message based communication, complex embedded applications.

1 Introduction

TriBA is a new idea in multiprocessor architectures. It is believed to be a high performance parallel computing architecture. The root of this idea is based on the concept that "Complex problems can be decomposed to three relatively independent sub-problems, which are *data Processing*, *data Management* and *data Communication*". Further they can be divided into smaller ones, which are simpler. This procedure repeats, until we think that each sub-problem is an atomic problem or some single individual can handle it. From the view of computer, these small tasks, which cannot be further subdivided, are called instructions. Instructions are composed of operation (op-code) and operand. Semantics of instructions indicate where and how to load and store operands. The basic Von Neumann computer runs these instruction sequences to solve complex problems. Not only the computer instructions have this characteristic, but also Von Neumann computer itself consists of data processing, storage and Input/Output modules. If we think that small task can constitute large task and Von Neumann architecture is the basic (low hierarchical) computer architecture, and a new large-scale architecture, which is powerful to solve large-scale problems, can be set up by three Von Neumann architectures. This constitution way follows bottom up style and uses three smaller architectures to compose a more powerful and larger scale computer systems.

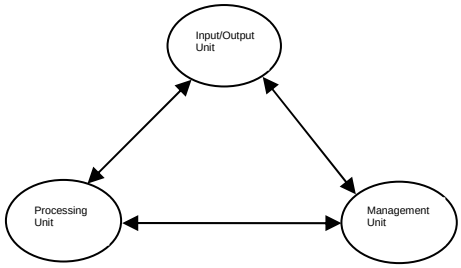

Fig. 1: Illustrates the basic cell in TriBA architecture.

The basic idea of TriBA is based on the similar concept of hierarchical structure which is self similar and fractal in nature. Section 2 elaborates this a bit further. Rest of the paper is organized as follows: In section 3 and 4 we briefly describe Intercommunication Network and the Execution Model for TriBA. Sections 5 compare TriBA with a typical 2D mesh of processors. Section 6 analyzes the speedup and efficiency of TriBA, which proves it a real scalable architecture for high performance

parallel applications. Finally section 7 describes what we intend to do for TriBA in future.

2 Fractal Architecture

Figure 2 we can see that the structure is self-similar. In fact, this is a fractal object called Sierpinski Gasket. Fractal geometry theory states, Sierpinski Gasket, can be produced by Iterator Function System (IFS). The existence of fractal structure is the result of long-term evolution of the natural world as a natural phenomenon. And widespread availability of such structure narrates that it has unparalleled superiority in information diffuseness and energy transmission [1].

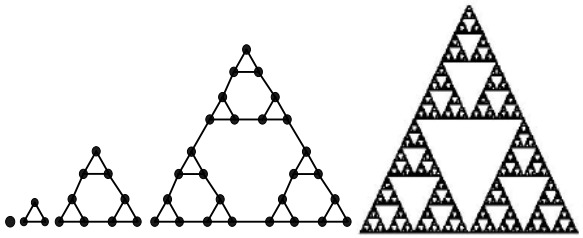

Fig. 2: Sierpinski gasket and pyramid

Object-oriented methodology is a cognitive methodology and tries to enable computer thinking and cognizing in the way of human. The abstract and stratification principles advocated by object-oriented methodology are the keys to solving complex problems, in which way we view the decomposition of complex problems. In object-oriented methodology, each object is considered as an independent entity and large object is the aggregation of small ones [2]. Objects contain attribute data, the operations used to manipulate data and message is the only way that different object communicates with each other. Thereby, we can use object-oriented methodology to model complex problems, and running object-oriented programs solves complex problems [3].

Studies show that the speed of solving problem will be greatly accelerated, if the question structure matches the system communication structure. The fractal nature of TriBA architecture exactly tallies with the question structure mentioned previously. The interconnection of TriBA manifests great similarity to the hierarchal structure of object systems, so that the software and computer system achieve a certain degree of structural unity. Consequently, this can facilitate the efficient execution of object-oriented programs and increase the pace of resolving problems.

"Complexity takes the form of hierarchy and hierarchical systems evolve faster than nonhierarchical ones" [H. A. Simon '78]. A hierarchy is a recursive partition of a system into subsystems and a general theory of complex system must refer to a theory of hierarchy. Software is such a complex artificial thing that discovering hierarchy and making use of hierarchy are principles to analyze and construct systems. Therefore, the TriBA computer architecture embodies the concept of hierarchicy and narrows the gap between problem structure and system structure. Thus it has the potential to become a high-performance parallel computing architecture.

3 TriBA Intercommunication Network

TriBA is a logical network. Nodes on TriBA cannot only be a simple cell but also a main board, a computer, etc. So that TriBA gives a uniform interconnection among the units in the core of multi-core CPU, main boards, and computers.

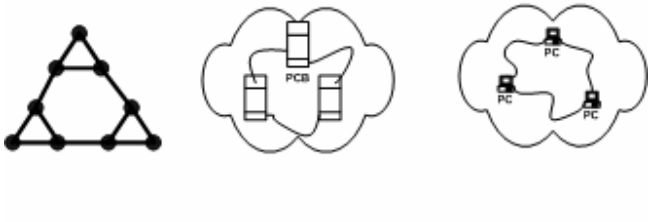

Fig. 3: Uniform Interconnect for TriBA

In addition TriBA interconnection has some convincing features like: 1 connection required per node for the lowest layer complete-connect, while a conventional 2D grid needs 1.25 connections for complete-connect. Nodes on TriBA can be coded directly for N-to-N, multi-cast, group-cast, that means TriBA makes routing implement easy. ID of node on TriBA is simply coded by 2 bit for each layer. It can also tag the groups formed by three lower nodes Direct networks have become a popular architecture for constructing massively parallel computers because they scale well [4]. Many experimental and commercial parallel computers [5], [6] exploit direct networks for low latency, high bandwidth interprocessor communication. A newly introduced class of networks called the Hierarchical interconnection networks (HIN's) which employ

multiple levels of explicitly defined connections to link disjoint clusters of nodes.

Triple-based hierarchical interconnection network (THIN) is not only a new kind of direct networks but also a kind of HIN's. Efficient routing algorithm is very essential to the performance of the interconnection network and the parallel computing system [7].

The constructing process of THIN is: based on level 1 THIN, replacing every node with lower level THIN to structure a higher level THIN, reiterating this process, we can get any higher level THIN, illustrated in figure:4 [7].

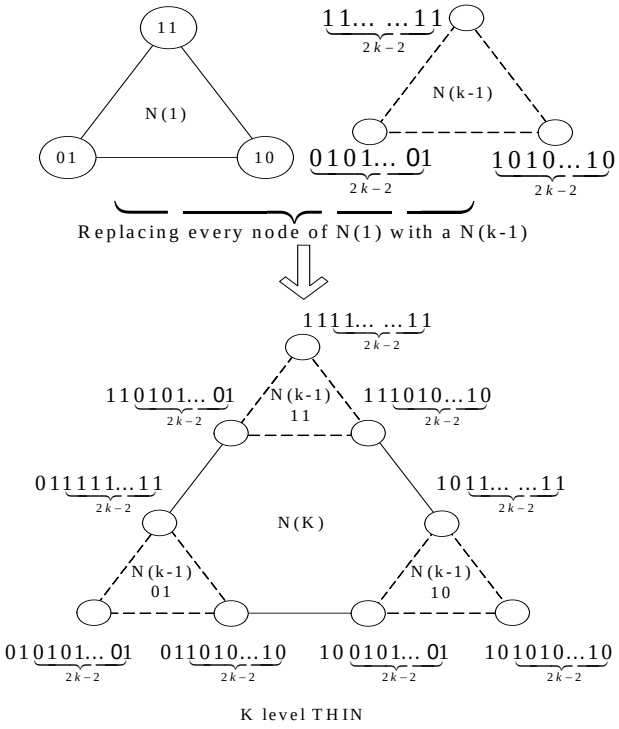

Fig. 4: Construction of Level-K THIN [7]

It is apparent that the routing information for communication incorporates ID of target node, which provides us that IDs in fact comprise of a distributed routing table.

4 Execution Model for TriBA

The logical view of a machine supporting the message-passing paradigm consists of *p* processes, each with its own exclusive address space. There are two implications of a partitioned addresses space. First, each data element must belong to one of the partitions of the space; hence, data must be explicitly partitioned and placed. This adds complexity to programming, but encourages locality of access that is critical for achieving high performance, since processes can access its local data much faster than non-local on such architectures. The second implication is that all interactions (read only or read/write) require cooperation of two processes – the process that has the data and the process that wants to access the data. This requirement for cooperation adds great deal of complexity [10].

The message-passing programming paradigm requires that the parallelism is coded explicitly by the programmer i.e., the programmer is responsible for analyzing the underlying serial algorithm/application and identifying ways by which the programmer can decompose the computations and extract concurrency. Message passing program can often achieve very high performance and scale to large number of processes.

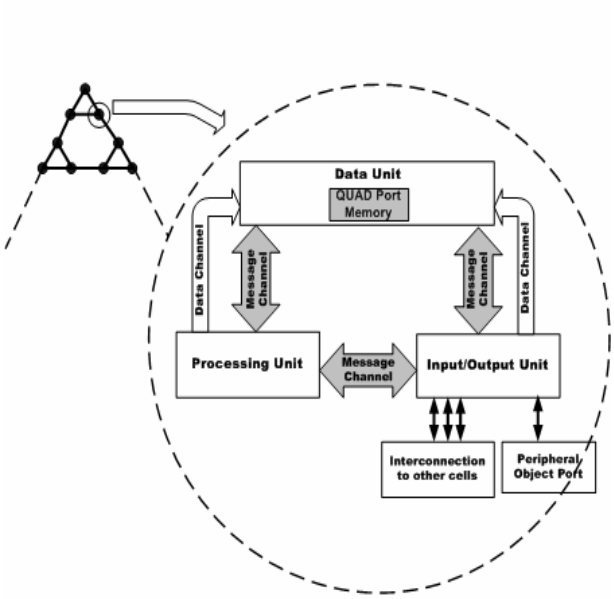

Fig. 5: Internal view of a cell in TriBA

The execution model of TriBA's architecture, as shown in Figure 5 clearly reflects message channels as well as data channels separately. Message passing between processing unit and interface unit, as shown in figure 6, is via a separate channel MC-PI, whereas MC-PD and MC-DI is used for message passing between Proc. Unit and Dat. Unit and between Dat. Unit and Inter Unit respectively. It is worth noting that Data Channels exist between Dat. Unit and Proc. Unit and Inter. Unit only for obvious reasons.

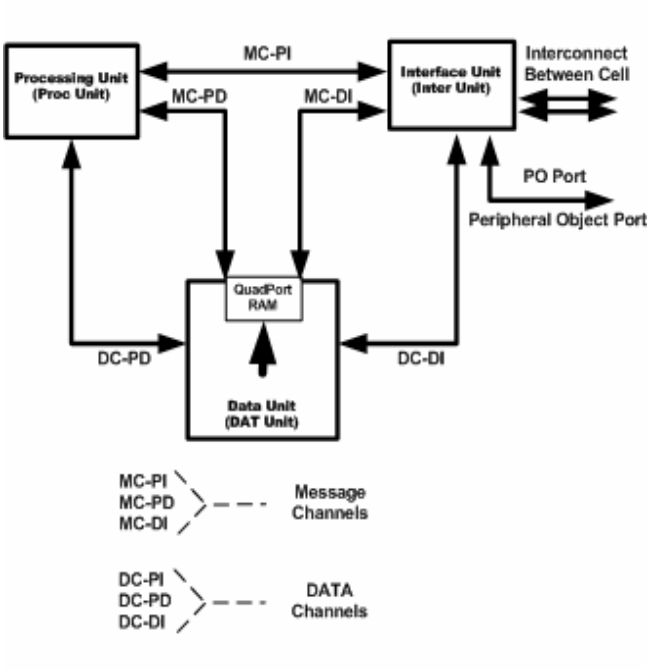

Fig. 6: Message based communication in TriBA

Figure 6 gives a clearer picture of the message-passing paradigm in TriBA not within one cell but also between cells

Direct Message	Send on Message Channel, Suitable for short massage high priority level)
Indirect Message	Will be inserted into message que in DatUnit, suitable for longer messages of low priority
Express Message	Sent on message channel, suitable for short messages with high priority level
Ordinary Message	Will be inserted into message que in DatUnit, suitable for longer messages with low priority level

Table. 1: Types of messages for TriBA

The types of messages used in TriBA's model are listed in Table 1. The messages are categorized as high priority short messages and low priority longer messages. DatUnit holds a message queue where the second class of messages i.e., indirect and ordinary messages are placed. In addition, to this these messages can be categorized as communication messages and mail messages. Communication Messages : are like phones and correspondences with no additional data, whereas, Mail Message : are those like parcel post with additional data.

5 Performance Evaluation of TriBA

The distinguishing advantage of TriBA is its low communication cost as described in [7]. In this paper two important metrics are analyzed i.e., the speed up and efficiency as shown in Figure. 7 and 8 respectively, representing TriBA's comparison to a typical 2D mesh processor structure. However, it is worth mentioning here that problem size has been taken constant in this analysis. A couple of examples are appended in this paper to elaborate the idea mentioned above.

Figure 7 give the speedup comparison of TriBA with a typical 2D mesh structure showing prominence of superior performance of TriBA with increment in the number of processors. Similarly, Figure 8 depict the efficiency relation between two structures boosting the concept of utilizing TriBA as compared to typical 2D mesh structure.

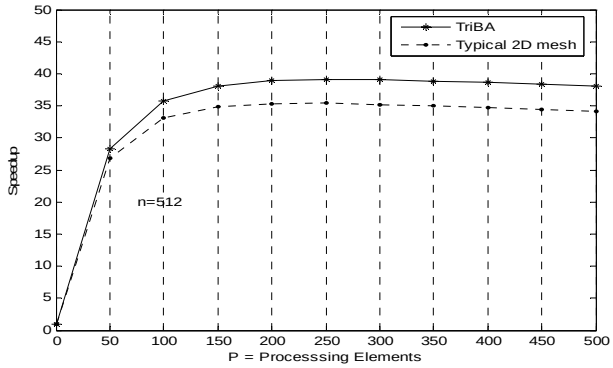

Fig. 7: Speedup of TriBA compared to 2D mesh

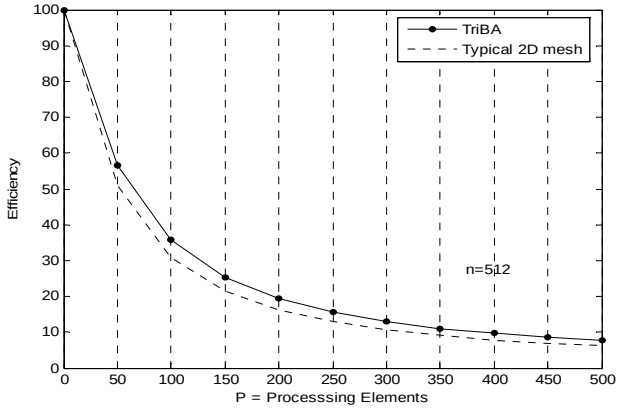

Fig. 8: Efficiency of TriBA compared to 2D mesh

6 Scalability Analysis of TriBA

Scalability has been become an important consideration in parallel algorithm and machine designs. It is used in practice as a property that describes the demand for appropriate changes in performance with adjustments in system size. When evaluating a parallel system, we are interested in its performance gain by parallelizing a given application over a sequential implementation. Speedup '*S*' is defined as the ratio of the time taken to solve a problem on a single processing element to the time required to solve the same problem on a parallel computer with identical processing elements measure. Speedup can never exceed the number of processing elements, *p*.

6.1 Example: Cost of adding' n' numbers

As the number of processing elements decreases by a factor of *n/p*, the computations at each process element increases by a factor of *n/p*. Very often programs are designed and tested for smaller problems on fewer processing elements. However, the real problems are much larger, and the machines contain larger number of processing elements. Their performance and correctness of programs is much more difficult to establish based on scaled-down systems. In this section we evaluate the scalability of parallel architecture TriBA, using analytical tools.

Consider the problem of adding *n* numbers on *p* processing elements. Assuming unit time for adding two numbers, the first phase (local summation) of the algorithm takes roughly *n/p* time. The second phase involves *log p* steps with a communication and an addition at each *2 log p* [10]. Therefore, we can drive parallel time, speedup, and efficiency as;

$$Parallel\ Execution\ time = \frac{n}{p} + 2\log p \qquad (1)$$

$$Speedup = \frac{n}{\frac{n}{p} + 2\log p} \qquad (2)$$

$$Efficiency = \frac{1}{1 + \frac{2p\log p}{n}} \qquad (3)$$

These expressions are used to calculate the speedup and efficiency for any pair of *n* and *p*. Figure 9 shows the Speedup versus *p* curves for a few different values of *n* and *p*. Table:2 shows the corresponding efficiencies. Efficiency plot can also be seen in Figure 10. Figure:9 shows that the speedup tends to saturate and efficiency drops as a consequence of ***Amdahl's law***.

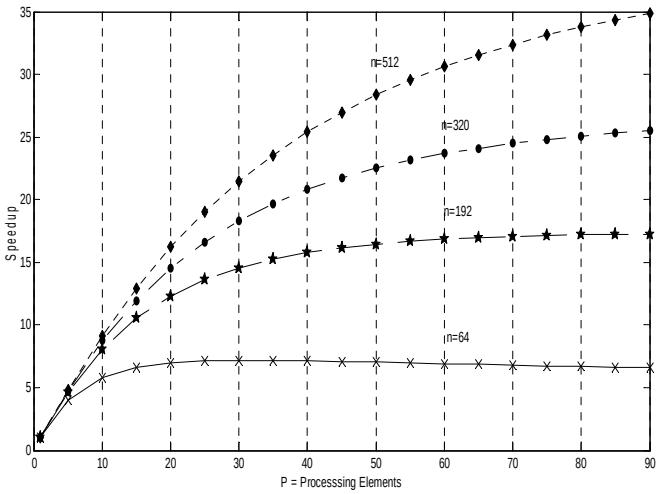

Fig. 9: Speedup versus the number of processing elements for adding *n* numbers.

We investigate the effect of increasing problem size keeping the processing elements constant. In cases, where overhead function grows sub linearly with respect to problem size, we see efficiency increases if the problem size is increased keeping the number of processing elements constant. For such algorithms it is possible to keep the efficiency fixed by increasing both the size of the problem and the number of processing elements simultaneously.

N	P=1	p=3	p=6	p=9	P=15	p=27	p=81
64	*1.00*	***0.90***	*0.74*	*0.61*	*0.44*	*0.26*	*0.08*
192	*1.00*	*0.96*	***0.89***	*0.82*	*0.70*	*0.51*	*0.21*
320	*1.00*	*0.97*	*0.93*	***0.89***	*0.79*	*0.64*	*0.31*
512	*1.00*	*0.98*	*0.95*	*0.92*	***0.88***	*0.74*	*0.42*

Table: 2 Efficiency as a function of *n* and *p*

This can be verified by results in Table: 2 i.e., the efficiency of adding 64 numbers using 3 (TriBA-architecture) processing elements is 90%. If the number of processing elements is increased to 6 and the size of the problem is scaled up to add 192 numbers, the efficiency nearly remains 90%. Increasing *p* to 15 and 27 and *n* to 320 and 512 respectively, results in approximately the same

efficiency figure of 90%. This proves TriBA architecture has full capability of being scalable parallel systems. Scalability reflects parallel system's ability to utilize increasing processing resources effectively.

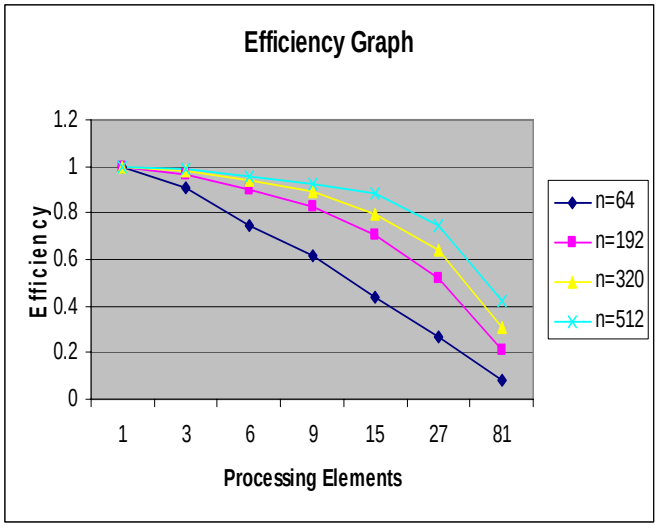

Fig. 10: Efficiency as function of *n* and *p* for adding *n* numbers on *p* processing elements

7 Conclusion

TriBA architecture is though a new idea in high performance computing systems, but is full capable of being a real scalable architecture which is suitable for sophisticated embedded applications with multiple concurrent processing centers. The goal of this research effort is to move a step forward in support of TriBA as it is believed to be a new solution for complex embedded applications as well as real time applications. However, we still need to investigate many aspects of TriBA like the hardware implementation of its characteristic object oriented methodology etc.

References

[1] Arthur D. Hall, III, "*The Fractal Architecture of the Systems Engineering Method*" IEEE transactions on systems, man, and cybernetics—part c: applications and reviews, vol. 28, no. 4, November 1998.

[2] P. N. Green, M. D. Edwards, "*An Object Oriented Design Method for reconfigurable computing systems*" IEEE Proc –Design Automation and test in Europe Conference and Exhibition 2000., pages 692-696, March 2000.

[3] P. N. Green, M. D. Edwards, "*Object Oriented development method for reconfigurable embedded systems*" IEE Proc -Comput. Digit Tech, Vol. 147 No. 3, May 2000.

[4] Lionel M. Ni and Philip K. McKinley, "*A Survey of Wormhole Routing Techniques in Direct Networks*", IEEE Computer, Vol. 26, No. 2, February 1993, pp. 62-76.

[5] Michael D. Noakes, Deborah A. Wallach and William J. Dally, "*The J-Machine Multicomputer: An Architectural Evaluation*", Proceedings of the 20th International Symposium on Computer Architecture, May 1993, pp. 2-13.

[6] Anant Agarwal, Ricardo Bianchini and David Chaiken et al, "*The MIT Alewife Machine: Architecture and Performance*", Proceedings of the 22nd International Symposium on Computer Architecture, June 1995, pp. 224- 235.

[7] Qiao Baojun, Shi Feng, and Ji Weixing, "*A New Routing Algorithm in Triple-based Hierarchical Interconnection Network*" Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06), 2006, 725-728.

[8] Shi Feng, Ji Wei-xing, Qiao Bao-jun, Liu Bin, "*A New Non Von Neumann Architecture TriBA*" Transactions of Beijing Institute of Technology, Vol. 26, No.10, Oct. 2006, pp. 847-849.

[9] Weiwei Lin, Changgeng Guo, Deyu Qi, Yuehong Chen, and Zhang Zhili, "*Implementations of Grid-Based Distributed Parallel Computing*", Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06), 2006,

[10] Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar, "*Introduction to Parallel Computing*", 2nd Ed., HZ Books, 2003.

[11] Julian M. Bass, "*Proposals toward an integrated design environment for Complex Embedded Systems*", Parallel and Distributed Processing, PDP '98. Proceedings of the Sixth Euromicro Workshop, 1998.

[12] Morris D., D.G. Evans, P.N. Green, and C.J. Theaker, "*Object oriented computer system engineering*", Springer Verlag, Berlin,1996.