THE ASIAN BULLETIN OF BIG DATA MANAGMENT
Vol. 4. Issue 1 (2024)
https://doi.org/10.62019/abbdm.v4i1.110

Leveraging Scratchpad Memory in a Hierarchical Architecture for Multicore

Kavita Tabbassum, Shahnawaz Farhan Khahro, Saima Shaikh, Farah Naveen Issani, Suhni Abbasi, Hina Chandio *

Chronicle

Article history

Received: February 12, 2024
Received in the revised format: Feb 24, 2024
Accepted: Feb 26, 2024
Available online: March 1, 2024

Kavita Tabbassum, Saima Shaikh, Farah Naveen Issani, Suhni Abbasi and Hina Chandio are currently affiliated with the Department of Information Technology Center, Sindh Agricultural University Tandojam, 70060, Pakistan.
Email: kavita@sau.edu.pk
Email: ss2kcs@gmail.com
Email: Farahnaveenissani@gmail.com
Email: suhni.abbasi@sau.edu.pk
Email: hinashafi@sau.edu.pk

Shahnawaz Farhan Khahro, is currently affiliated with the Energy Department, Govt. of Sindh.
Email: shahnawazfarhan@gmail.com

***Corresponding Author**

Abstract

This paper proposes a novel architecture for multi-core processors, tailored for high-performance parallel computing. The architecture is founded on the innovative notion that complex problems can be decomposed into three relatively independent sub-problems: data processing, data management, and data communication. It features a grid of small, programmable processing units intricately connected to their three neighbouring units, forming a physically scalable and fractal environment. With flexibility, modularity, and scalability as focal points, this architecture aims to address the anticipated real-time signal processing demands in future telecommunication and multimedia systems. One notable aspect of the proposed architecture is its direct support for object-oriented features at the hardware level. Adopting a hybrid approach with Scratchpad Memory (SPM) combined with Cache in the on-chip memory hierarchy enhances performance and adaptability for sophisticated multi-core applications. The study highlights SPM management and introduces a dynamic data management framework. Unlike traditional SPM allocation methods relying on compiler or profiling knowledge, the proposed approach leverages random sampling and probability theory to predict hot-access data during runtime. This dynamic memory access pattern guides SPM allocation, ensuring optimal utilization of SPM's advantages in access speed and energy consumption, complemented by hardware support from DataUnit. This paper presents a paradigm shift in multi-core processor architecture, offering advanced features to meet the evolving requirements of parallel computing, making it well-suited for future telecommunication and multimedia systems.

Keywords: Scratchpad memory, Dynamic memory management, Multi-core, CMP Set.

INTRODUCTION

The objective of this paper is to introduce a innovative architecture tailored for multi-core processors, aimed at enabling high-performance parallel computing. The proposed architecture is significant due to its innovative approach to addressing the complex challenges inherent in parallel computing tasks. Traditional multi-core architectures often struggle to efficiently manage data processing, data management, and data communication simultaneously. This paper seeks to overcome these challenges by presenting a novel architecture that decomposes complex problems into three relatively independent sub-problems. By providing a grid of small, programmable processing units intricately connected to their neighboring units, this architecture offers scalability and

flexibility crucial for meeting the demands of modern computing applications. Moreover, the proposed architecture directly supports object-oriented features at the hardware level, enhancing performance and adaptability for sophisticated multi-core applications. By combining Scratchpad Memory (SPM) with Cache in the on-chip memory hierarchy, the architecture achieves optimal utilization of resources, paving the way for efficient parallel computing. It is a new type of architecture proposed based on the development trend of architecture in recent years. This paper mainly introduces several characteristics and focuses on the SPM-based hierarchical memory architecture based on the on-chip SPM management mechanism. In summary, the objective of this paper is to introduce an advanced multi-core processor architecture capable of addressing the evolving requirements of parallel computing. Its significance lies in its ability to offer unprecedented flexibility, modularity, and scalability, making it well-suited for future telecommunication and multimedia systems.

LITERATURE REVIEW

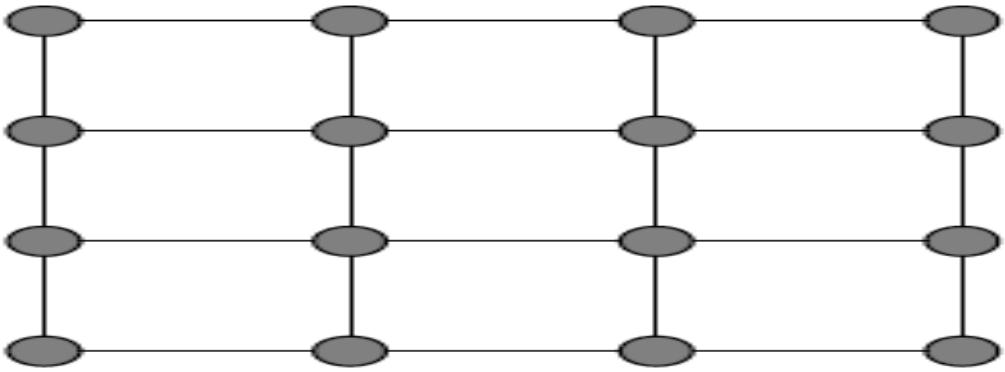
The proposed architecture can find application in various real-world scenarios where high-performance parallel computing is essential. Here are some specific examples: In telecommunications systems, in telecommunications, real-time signal processing is crucial for tasks like signal modulation, demodulation, and error correction. The proposed architecture's ability to handle complex data processing tasks with high efficiency makes it suitable for telecommunication systems requiring low-latency and high-throughput processing. Multimedia Processing: Multimedia applications such as video encoding, decoding, and image processing demand substantial computational resources. The architecture's support for parallel computing enables efficient handling of these tasks, leading to faster processing times and improved multimedia quality. Scientific Computing: Scientific simulations and computations often involve processing large datasets and running complex algorithms.

The proposed architecture's scalability and flexibility make it well-suited for scientific computing applications, allowing researchers to perform simulations and analyses more efficiently. Data Analytics: With the proliferation of big data, there is a growing need for high-performance computing architectures to analyze large datasets quickly. The proposed architecture's parallel processing capabilities can accelerate data analytics tasks, enabling businesses to extract valuable insights from their data in a timely manner. Artificial Intelligence and Machine Learning: AI and machine learning algorithms often require intensive computational resources to train models and process large datasets. The proposed architecture's support for parallel computing can significantly speed up the training process and improve the performance of AI applications.

Overall, the proposed architecture's versatility and efficiency make it suitable for a wide range of real-world applications that require high-performance parallel computing capabilities. The 2D-Mesh stands out as the most mature on-chip network structure, as illustrated in Figure 2.1 *(Deng et al., 2009*). Its proven track record and suitability make it a preferred interconnection method for on-chip network topology *(Egger, Lee, & Shin, 2008)*. Low-dimensional networks like the 2D-Mesh offer certain advantages, including reduced communication delays and blocking probabilities compared to high-dimensional networks *(*Deng, Ji, & Shi, 2009; Takase, Tomiyama, & Takada, 2010*)*.

Consequently, in multi-core chip design, low-dimensional networks are frequently chosen for interconnecting processing components (*Gauthier, Ishihara, & Takada, 2010)*. Multi-core interconnect structures have been a focal point in research, driven by the shift from Chip Multiprocessors (CMPs) to multi-core and many-core configurations. While the bus structure has historically been popular due to its simplicity and ease of implementation (Marongiu & Benini, 2010), it is now facing limitations in bandwidth and signal integration as technology advances. This has led to a shift towards network-based interconnection methods, known as Network-on-Chip (NoC) architectures.

The 2D-Mesh, depicted in Figure 2.1, has emerged as a mature and suitable choice for on-chip network topology *(Zuo, Qi, & Jiaxing, 2008; Zuo, Qi, & Jiaxin, 2009; Liu, Ji, Li, et al., 2009)*. By enhancing communication bandwidth and efficiency among nodes within basic groups, internal interconnections greatly improve communication efficiency. However, the full interconnection of nodes within each basic group of the 2D-Mesh can incur high costs, as depicted in Fig. 2.2. To address scalability challenges, the size of full interconnections is gradually reduced in upper-level interconnections, leading to diminished scalability with increasing core count.



Figur1.
2D-Mesh Structure

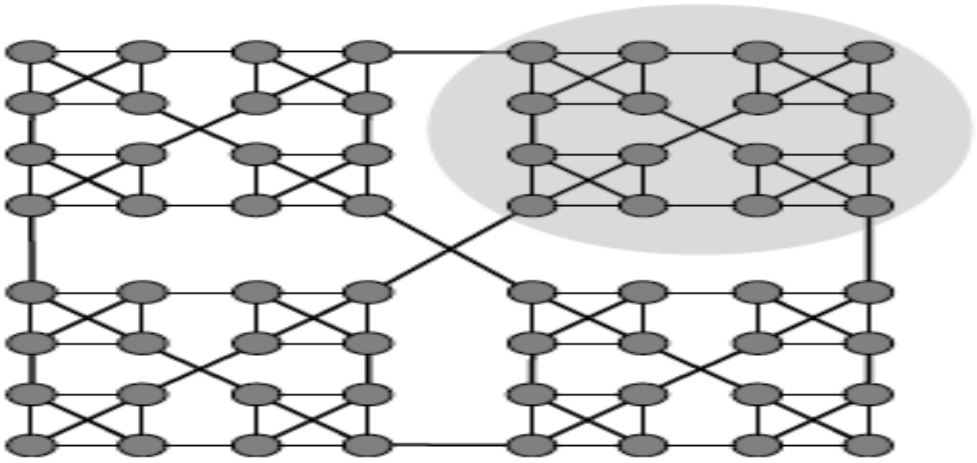


Figure 2.
2D-Mesh Structure Full

$$N_{k+1} = \bigcup_{i=1}^{3} F_i(N_k) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

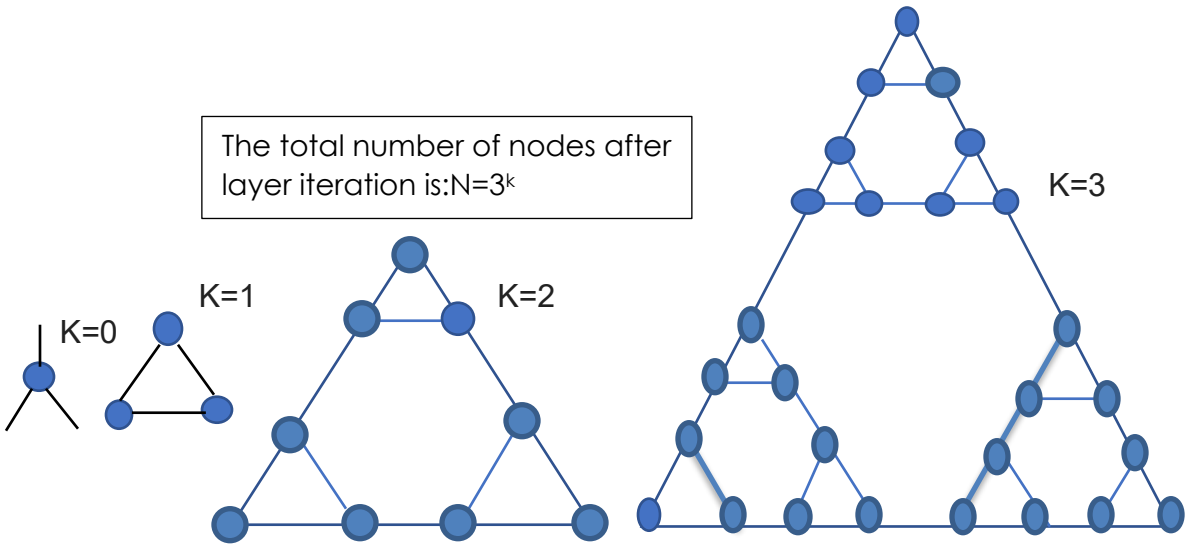


Figure 3.
Schematic diagram of iterative network at k=0, 1, 2, and 3

FORMAL MODEL OF NETWORK

Proposed architecture adopts a basic three-layered on-chip network structure to solve the problem of full interconnection between the underlying nodes. The structure has the advantages of simple topology, low degree of nodes, and obvious hierarchy. The topological structure of the three-layered on-chip network is shown in Figure 2.3. It is assumed that the iterative function family of the structure is 123{,,}IFSFFF, and the 1-layer interconnection network is regarded as the network after 1 iteration (1)N. , () Nk represents the k-layer network obtained after k iterations, then the interconnection network construction process can be expressed as *(Takase, Tomiyama, & Takada, 2010; Catthoor et al., 2018)*.

The network topology is a low-level, fully interconnected structure in which the basic building blocks are a group of three nodes. The fully interconnected structure is called a super node. By replacing the nodes in the topology with super nodes, a larger base triple topology can be obtained. Figure 2.3 shows the 1-node, 3-node, 9-node, and 27-node base three topology structures obtained by the basic components after 0, 1, 2, and 3 replacements.

Features of the Interconnect Network

The static metrics of the general network mainly include: the degree of the network, the total number of links, the width of the section and the diameter of the network. From the perspective of VLSI layout and routing, low-dimensional networks have advantages in implementation compared to high-dimensional networks. Because low-dimensional networks have a certain online width, smaller communication delays and blocking probabilities can be obtained than high-dimensional networks *(Wasly, 2018; Alvarez et*

al., 2015). Therefore, in the design of multi-core chips, low-dimensional networks are often used for interconnection between processing components. Table 2.1 shows the comparison of several common interconnected network structures in terms of node degree, total number of links, cross-sectional width, and network diameter, where N represents the number of network nodes.

According to the calculation formula in the above table, it can be concluded that as the number of nodes increases, the changes of various indicators on the network on the chip. It is easy to see that the base three network nodes are the smallest among the nodes of all networks, which also reduces the required entries for the implementation of routers in the base three network. In Table 2.1, several metrics are compared for different interconnected network structures, including degree, total number of links, split width, and network diameter. These metrics are essential for evaluating the performance and efficiency of on-chip network structures.

Degree: The degree of a network refers to the number of connections or edges each node has. In the context of on-chip networks, a lower degree indicates a less complex topology with fewer connections per node. This metric is crucial because a lower degree typically implies simpler routing algorithms, reduced hardware complexity, and potentially lower power consumption.

Total Number of Links: This metric represents the overall number of connections or links within the network. A lower total number of links implies a more efficient use of resources and potentially lower hardware costs. It also affects the complexity of routing algorithms and the overall scalability of the network.

Split Width: The split width refers to the number of links crossing a cut in the network. In on-chip networks, split width is crucial for determining the efficiency of communication between different sections or modules of the chip. A smaller split width implies faster communication and reduced latency between components.

Network Diameter: The network diameter is the maximum distance or number of hops between any pair of nodes in the network. A smaller network diameter indicates shorter communication paths and lower latency, which are critical for real-time applications and overall system performance.

These metrics are essential for evaluating the suitability of on-chip network structures for multi-core processors. Lower values for degree, total number of links, split width, and network diameter generally indicate more efficient and scalable network architectures, which are crucial for meeting the performance requirements of modern computing applications. Therefore, comparing these metrics helps researchers and designers choose the most appropriate network structure for their specific use case, considering factors such as performance, power consumption, and cost.

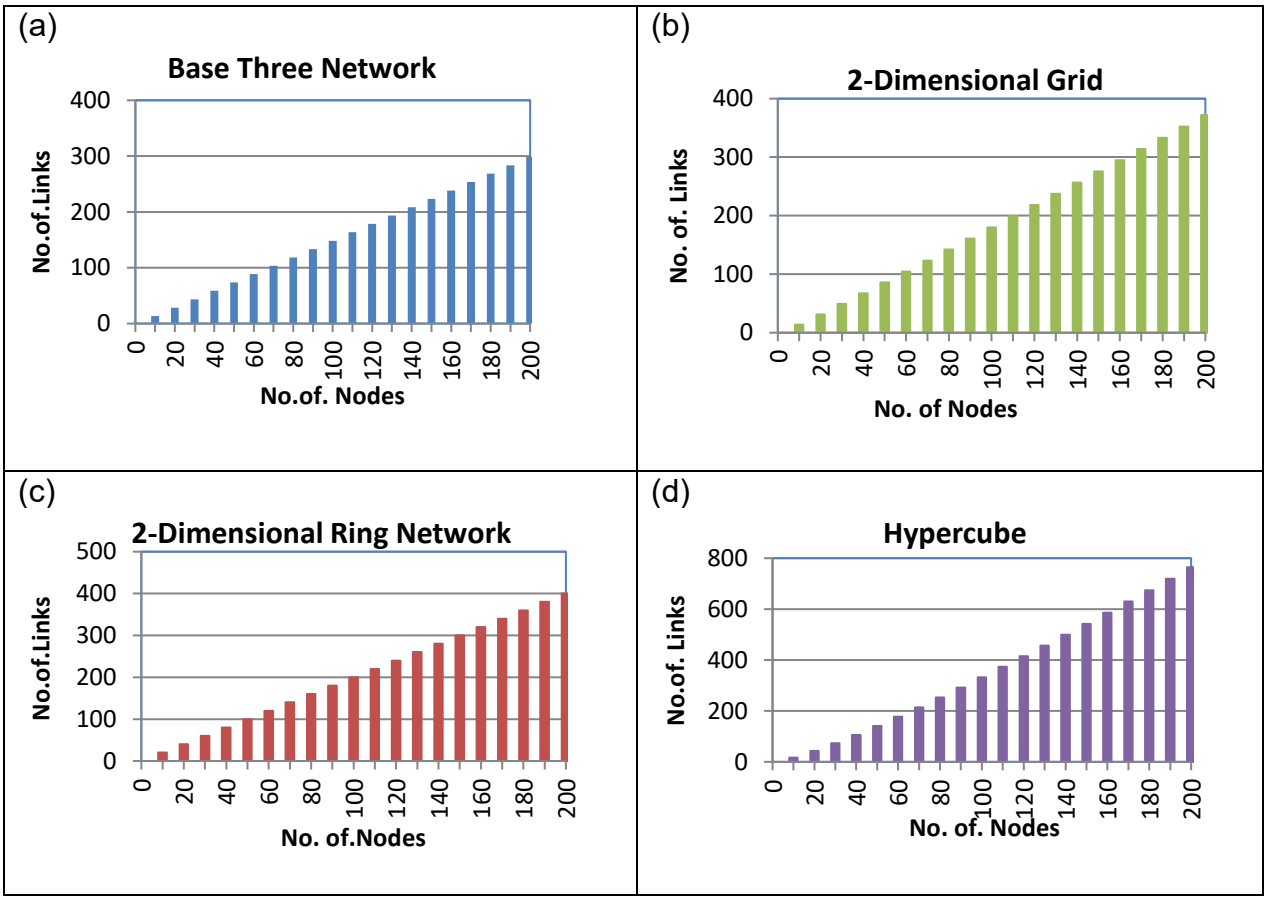
Table 1.
Comparison of Static Metrics for Several Interconnected Networks

Network Type	**Interconnect Network Static Metric**			
	Degree	Total number of links	Split Width	Network Diameter
Base three Network	3	3(N-1)/2	$\log_3 N$	$2^{\log_3 N} - 1$
Two-Dimensional Grid	4	2 (N - $\sqrt{N}$)	$\sqrt{N}$	2 ($\sqrt{N} - 1$)

Two-Dimensional Ring Network	4	2N	$2\sqrt{N}$	2 ($\sqrt{N}$ / 2)
Hypercube	$\log_2 N$	(N × $\log_2$ N) / 2	N/2	$\log_2 N$

Figure 2.4(a)(b)(c)(d) and (e) and Figure 2.5(a)(b)(c)(d) and (e) show the trends in the number of links and network diameters of various networks as the number of nodes produces. As can be seen from Figure 2.4, in the case of the same number of nodes, the number of links in the base three network is the smallest among all network topologies, and the number of links directly affects the complexity of the layout and routing in hardware implementation. Hardware overhead. The comparison of the network diameters in Figure 2.5 shows that the hypercube has the smallest network diameter when the number of nodes is the same, but because the node degree of the hypercube grows logarithmically with the number of nodes (2logN), it is not suitable for building on-chip. It is easy to see that in the case of a small number of nodes (N <140), the network diameter of the base three network is smaller than that of the two-dimensional grid. As the network scale expands, the network diameter exceeds the network diameter of the two-dimensional grid.

ProcUnit



Simplified Description: Think of ProcUnit as the brain of the processor. It's responsible for processing data and executing tasks, much like how your brain processes information and performs actions.Concrete Example: Imagine ProcUnit as a chef in a kitchen. The chef receives orders (data) and performs tasks like chopping vegetables, cooking dishes, and serving meals. Similarly, ProcUnit receives data and performs operations like calculations, logic tasks, and data manipulation. ProcUnit corresponds to the function or method in the object and is mainly responsible for processing the data. Its basic functions are similar to those of a normal CPU, used to run normal instructions or to manipulate the behavior of objects by processing messages between objects. In addition to the functions of the arithmetic logic unit and control unit in the usual processor, ProcUnit can also directly support some of the more commonly used object-oriented instructions *(Egger, Lee, & Shin, 2008)*. ProcUnit achieves the purpose of changing the state of an object by running a method in the object to modify the data corresponding to the object.

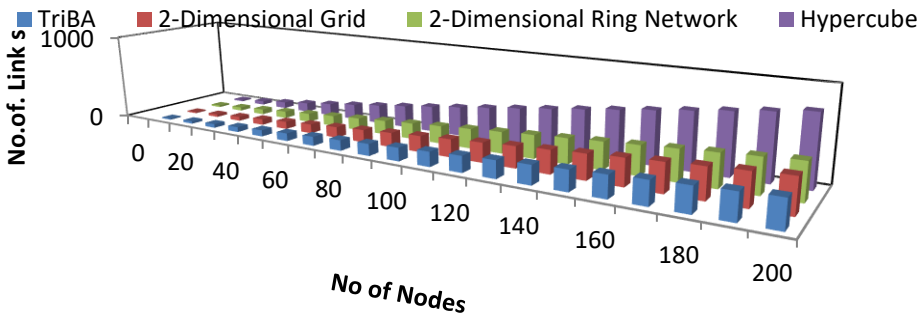
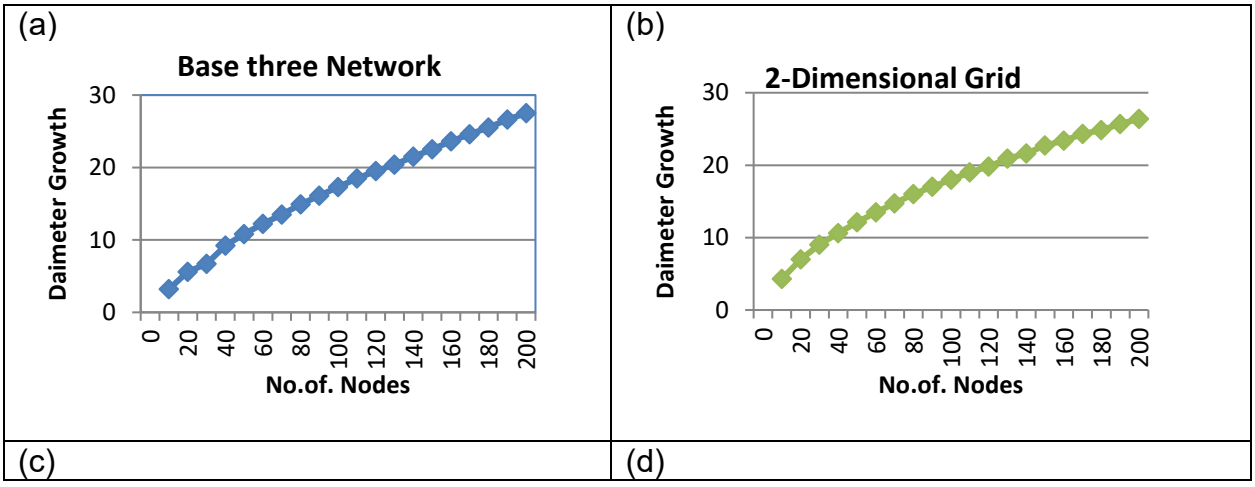


Figure 4.
Trend of Link Numbers in Network over Time



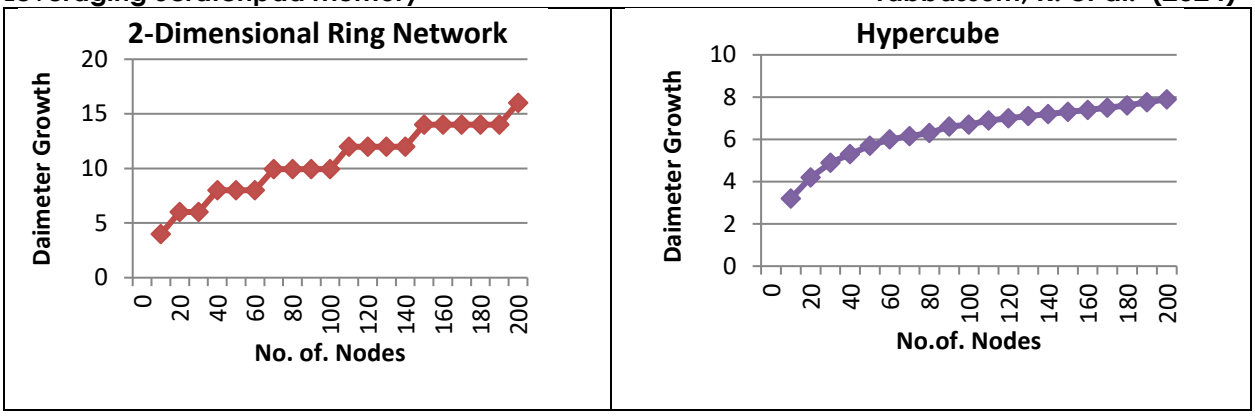


Figure 5.
Trend of Network Diameter in the Proposed Network over Time

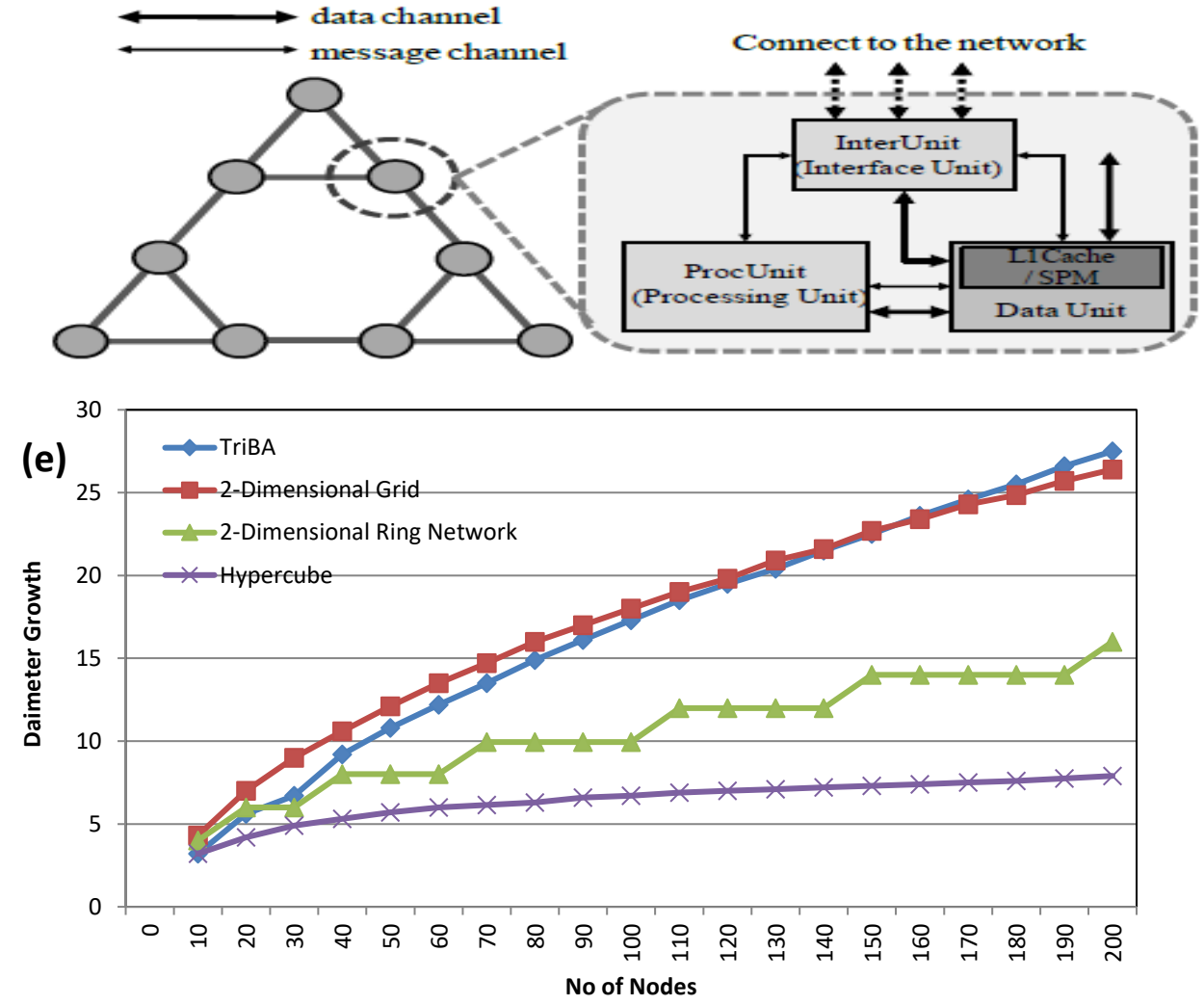


Figure 6.
Schematic Diagram of the Internal Structure of a Single Node.

ProcUnit and DataUnit are fundamental components of our proposed architecture for multi-core processors. Let's break down their descriptions into simpler terms:

DATA UNIT

Simplified Description: DataUnit is like the memory manager of the processor. It handles the storage and retrieval of data, ensuring that information is accessible when needed. Concrete Example: Think of DataUnit as a librarian in a library. The librarian manages books (data) by organizing them on shelves, keeping track of their locations, and retrieving them when requested. Similarly, DataUnit manages data by allocating memory space, organizing data storage, and facilitating data access for the processor. By simplifying ProcUnit and DataUnit using these analogies, it becomes easier to understand their roles and how they contribute to the overall functioning of our multi-core processor architecture. DataUnit is mainly responsible for the memory management of data. Its basic functions can be summarized as follows:

(1) Provide an efficient object access mechanism. In the object-oriented processor, the object replaces the traditional data into the individual stored in the memory, so it is necessary to design a special hardware and memory protocol to support the access of the object. In the object store, each object exists in the form of a contiguous space in memory. When accessing an object, an object reference is needed to identify the access object, and the data field contained in the object is usually added by the object reference plus the object data offset. The form is accessed. Corresponding to the virtual memory technology used in the traditional process-oriented processor, in the object-oriented processor, the mapping between the object reference and the physical address needs to be implemented by establishing an object table. DataUnit is used within a single Cell to map such virtual objects to physical addresses. In the work of the students before the group, the object memory technology has been discussed in depth (*Takase, Tomiyama, & Takada, 2010; Deng, Ji, Li, Zuo, & Shi, 2011)*.

(2) Dynamic Management of SPM. The program needs to dynamically apply for the memory space of the corresponding size during the running process, and release the space after a certain time. In proposed architecture, each processing core has a certain amount of on-chip SPM resources *(Tabbassum, Talpur, Narejo, & Laghari, 2019)*. The allocation and release of local SPM space is also one of the main functions of DataUnit. DataUnit's support for SPM allocation is mainly reflected in the hardware support of virtual memory management and SPM runtime management. In terms of virtual memory management, it supports a variety of granular page sizes. You can adjust the granularity of data blocks involved in SPM dynamic management as needed, and update the virtual and real address mapping according to the change of physical addresses before and after SPM allocation. Address redirection; in terms of SPM runtime management, DataUnit mainly provides corresponding hardware support to complete efficient SPM runtime dynamic management.

In order to achieve efficient support for the runtime SPM adaptive algorithm, the corresponding hardware components are added to the DataUnit to assist the runtime system for efficient SPM management. For example, in the random sampling SPM allocation algorithm, the Memory Reference Sampling Unit (MRSU) is integrated into the

DataUnit to sample the core data; in the SPM management strategy based on the access count, the access accumulator is added in the DataUnit. (ACC) Counts memory accesses. This method of dynamic management of SPM through software and hardware coordination ensures high runtime efficiency. This paper attempts to use the hardware and software synergy method to dynamically manage the application and release of SPM. By using the hardware provided by DataUnit, the core working set is predicted at runtime and the SPM is managed efficiently. Provide the function of sending and receiving messages. DataUnit itself can also be considered as having a pumping Objects like functions can provide a certain amount of buffer and support data transmission and reception *(Gao, 2014; Javaid, Zafar, Awais, & Shah, 2017)*. When the InterUnit needs to forward data to the data store in the local Cell, the DataUnit is responsible for receiving the data and assisting in the completion of the data memory function; when the data in the DataUnit needs to be forwarded through the InterUnit, it will also provide the local cache to be sent data. Perform a temporary memory to be sent by InterUnit at the appropriate time.

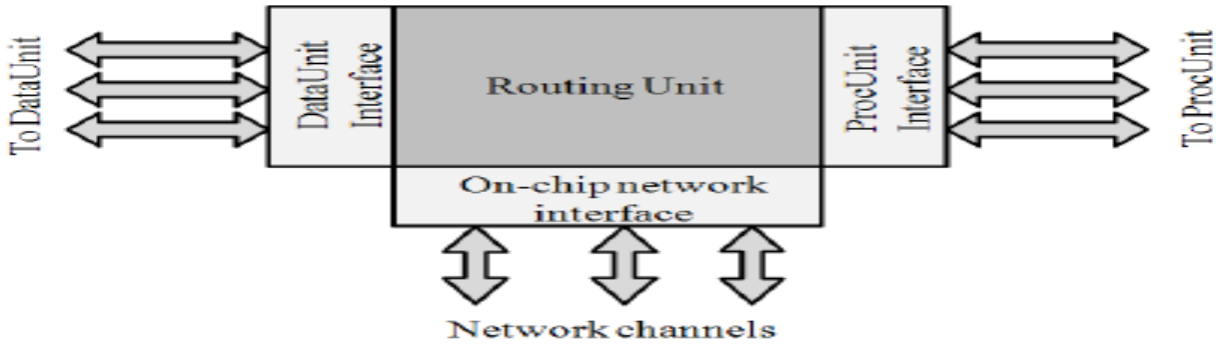


Figure 7.
Inter-Unit Structure Diagram

InterUnit

As shown in Figure 2.7, InterUnit is mainly responsible for the interaction and forwarding of data or messages. The objects interact through messages, and the interaction and synchronization between the various processing nodes is also controlled by the message as a carrier. InterUnit consists of routing component, DataUnit interface, ProcUnit interface and network interface. InterUnit is used to connect DataUnit, ProcUnit and other on-chip network interfaces for processing cores to enable data and message interaction between different nodes. Inside a single processing core, InterUnit also acts as a channel between the ProcUnit and the DataUnit . It is responsible for sending data or messages from the DataUnit to the ProcUnit for processing, and writing the results back to the DataUnit to complete the related operations of the object.

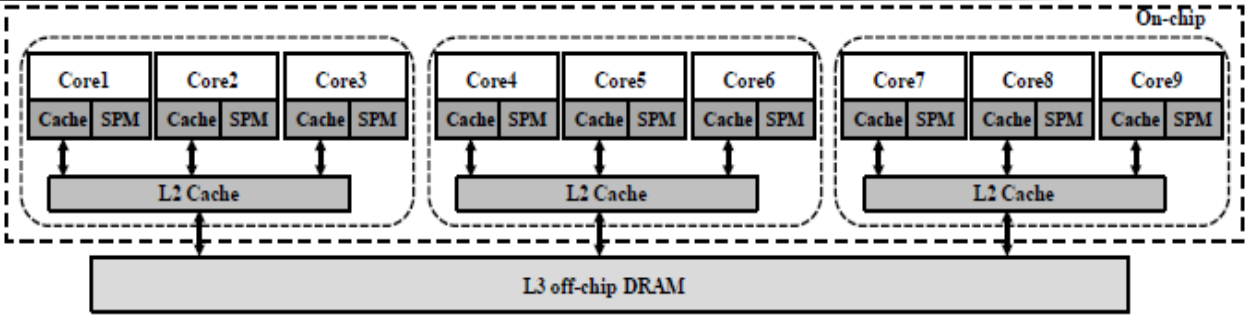


Figure 8.
Hierarchical Group Shared Memory Design

CONCLUSION

This paper introduces the basic characteristics of the base three multi-core architecture, including its unique base three network topology, kernel micro-architecture and hierarchical memory architecture. These characteristics make the base three multi-core architecture different from the existing traditional architecture, and become a special architecture with the underlying full interconnection, the number of links between the layers decreasing step by step, and the locality of the network operation is very obvious.

DECLARATIONS

Acknowledgement: We appreciate the generous support from all the supervisors and their different affiliations.
Funding: No funding body in the public, private, or nonprofit sectors provided a particular grant for this research.
Availability of data and material: In the approach, the data sources for the variables are stated.
Authors' contributions: Each author participated equally to the creation of this work.
Conflicts of Interests: The authors declare no conflict of interest.
Consent to Participate: Yes
Consent for publication and Ethical approval: Because this study does not include human or animal data, ethical approval is not required for publication. All authors have given their consent.

REFERENCES

Alvarez, L., Vilanova, L., Moret´o, M., Casas, M., Gonz`alez, M., & Martorell, X., et al. (2015). Coherence Protocol for Transparent Management of Scratchpad Memories in Shared Memory Many-core Architectures. In Proceedings of the 42nd Annual International Symposium on Computer Architecture, ISCA '15 (pp. 720–732). ACM.

Anzt, H., Hahn, T., Heuveline, V., & Rocker, B. (2010). GPU Accelerated Scientific Computing: Evaluation of the NVIDIA Fermi Architecture; Elementary Kernels and Linear Solvers. EMCL Preprint Series.

Catthoor, F., Hartmann, M., Gomez, J. I., Tenllado, C., Xydis, S., Rodrigo, J. S., ... & Soudris, D. (2018). Memory Structure Comprising Scratchpad Memory. United States patent application US 15/726,749.

Deng, N., Ji, W., & Shi, F. (2009). A novel adaptive scratchpad memory management strategy. In The 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (pp. 236–241).

Deng, N., Ji, W., Li, J., Zuo, Q., & Shi, F. (2011). Core Working Set Based Scratchpad Memory Management. IEICE Transactions on Information and Systems, 94(2), 274–285.

Egger, B., Lee, J., & Shin, H. (2008). Dynamic scratchpad memory management for code in portable systems with an MMU. Transactions on Embedded Computing Systems, 7(2), 1–38.

Egger, B., Lee, J., & Shin, H. (2008). Scratchpad memory management in a multitasking environment. In EMSOFT (pp. 265–274).

Gao, Y. (2014). Automated Scratchpad Mapping and Allocation for Embedded Processors. Ph.D. Thesis, University of South Carolina - Columbia.

Gauthier, L., Ishihara, T., & Takada, H. (2010). Stack Frames Placement in Scratch-Pad Memory for Energy Reduction of Multi-task Applications. In the Workshop on Synthesis and System Integration of Mixed Technologies (pp. 171-176).

Javaid, Q., Zafar, A., Awais, M., & Shah, M. A. (2017). Cache Memory: An Analysis on Replacement Algorithms and Optimization Techniques. Mehran University Research Journal of Engineering & Technology, 36(4), 10.

Liu, M., Ji, W., Li, J., et al. (2009). Storage Architecture for an On-chip Multi-core Processor. In 12th Euromicro Conference on Digital System Design: Architecture, Methods and Tools (pp. 263-270).

Marongiu, A., & Benini, L. (2010). An OpenMP Compiler for Efficient Use of Distributed Scratchpad Memory in MPSoCs. IEEE Transactions on Computers.

Singh, A. K., Geetha, K., & Ramasubramanian, N. V. (2016). Efficient Utilization of Shared Caches in Multicore. Arab Journal of Science and Engineering, 41, 5169–5179.

Suhendra, V., Roychoudhury, A., & Mitra, T. (2008). Scratchpad allocation for concurrent embedded software. In CODES + ISSS (pp. 37–42).

Takase, H., Tomiyama, H., & Takada, H. (2010). Partitioning and allocation of scratch-pad memory for priority-based preemptive multi-task systems. In Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2010 (pp. 1124–1129). European Design and Automation Association, Leuven.

Tabbassum, K., Talpur, S., Narejo, S., & Laghari, N. (2019). Management of Scratchpad Memory Using Programming Techniques. Mehran University Research Journal of Engineering & Technology, 38(2), 305-312.

Wasly, S. (2018). Scratchpad Memory Management for Multicore Real-Time Embedded Systems.

Zuo, W., Qi, Z., & Jiaxin, L. (2009). Triplet-based topology for on-chip networks. WSEAS Transactions on Computers, 8(3), 516-525.

Zuo, W., Qi, Z., & Jiaxing, L. (2008). Traffic Analysis for Triplet-Based Networks Based on Full-System Simulation. In International Symposium on Information Science and Engineering (pp. 121-124).