International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
***Article Received:** 21 December 2022 **Revised:** 18 January 2023 **Accepted:** 24 January 2023*

Multi-core Tile-based Processor with Transactional Memory for Efficient Use of Resources

R.Kennady[1], S.karthik[2]

[1]Department of Artificial Intelligence and Data Science, Rajalakshmi Institute of Technology, Chennai, Tamilnadu

[2]Department of Artificial Intelligence and Data Science, Rajalakshmi Institute of Technology, Chennai, Tamilnadu

[1]kennady.r@ritchennai.edu.in , [2]karthik.s@ritchennai.edu.in

Abstract:

This research focuses on the design and implementation of a multi-core processor with a tile structure and transactional memory method to improve resource utilization and address the limitations of hardware transactional memory systems. The proposed processor utilizes a network-on-chip to connect the tiles, and the number of each tile type can be adjusted based on application requirements. In this architecture, L2 cache is bypassed during transaction execution, and a router on the chip efficiently routes and stores transaction read-write requests in a transaction buffer area. A partition strategy is employed to allocate a portion of the available region specifically for transaction threads, and the read-write operations of transactions are recorded in the transaction buffer area. Furthermore, the size of the partition dynamically expands as the transaction read-write set increases. Through these innovations, the research aims to address issues related to resource waste, low buffer area utilization, and lack of support for thread switching and migration. The effectiveness of the proposed design is evaluated through simulations and benchmarking, demonstrating its ability to mitigate transaction buffer overflow and enhance overall performance.

Keywords: Multi-core processor, tile structure, transactional memory, resource utilization, network-on-chip, partition strategy, buffer area, thread switching, migration, performance optimization.

Introduction:

In the era of increasing computational demands, multi-core processors have emerged as a promising solution to enhance performance and cater to the requirements of modern applications. However, traditional multi-core architectures face challenges in efficiently utilizing available resources and handling concurrent execution of threads. This research proposes a novel design that incorporates a tile-based structure and transactional memory method to address these challenges. By employing a tile structure, where each node represents a different tile type (TL1, TL2, TL3), and interconnecting them with a network-on-chip, the processor architecture becomes highly scalable and adaptable to varying application workloads. Additionally, the utilization of transactional memory overcomes the limitations of hardware transactional memory systems, such as resource waste and low buffer area utilization. This research aims to demonstrate the effectiveness of the proposed design in optimizing resource utilization and mitigating transaction buffer overflow, thereby improving overall system performance.

Background:

As processor technology continues to advance rapidly, the development of multicore processors has become the mainstream trend. Even lower-end desktop applications have entered the era of multicore processing due to increasing performance requirements. The industry has released numerous multicore processors, and the number of computing cores continues to rise. In this context, there is a growing demand for hardware systems that can effectively utilize the abundant hardware parallelism to meet the higher software (application) requirements on processors' parallel processing capabilities.

Currently, shared resource synchronization techniques crucial in multithreaded programming are still based on lock mechanisms such as semaphores and mutexes. However, these mechanisms have drawbacks, including deadlock, priority inversion, and the difficulty of programming and debugging concurrent programs compared to serial programs.[1,4]

Transactional Memory (TM) models draw inspiration from the concept of "transactions" in databases. They provide a method for parallelizing and synchronizing programs on Chip Multiprocessors (CMP) or Symmetrical Multi-Processing

Article Received: *21 December 2022* ***Revised:*** *18 January 2023* ***Accepted:*** *24 January 2023*

(SMP) architectures. In this model, the limited sequence of machine instructions in a program is treated as a transaction, ensuring the atomicity of transaction execution in the system architecture. TM provides corresponding operation primitives such as commit, abort, and support for rollback actions. It overcomes the various issues associated with lock mechanisms and significantly improves the correctness and efficiency of multithreaded programming, allowing programmers to focus on the design of multicore programs.[5]

Due to the advantages in speed and efficiency offered by hardware mechanisms, current research projects are devoted to the development of Hardware Transactional Memory (HTM) models in multi-core environments.

Existing hardware transactional memory organizations are based on the structure of multicore processors, with additional hardware components in processor cores to support transaction execution. However, this extension of the structure faces challenges in terms of resource wastage, buffer utilization, and buffer overflow in many-core architectures. Therefore, in the environment of many-core processors, it is crucial to design an effective transactional memory organization.

The design of an efficient transactional memory organization in many-core processors needs to address several key issues. Firstly, resource wastage must be minimized to ensure optimal utilization of hardware components. Secondly, the utilization of buffer areas must be maximized to avoid inefficiencies and limitations in handling concurrent transactions. Finally, the issue of buffer overflow needs to be effectively managed to prevent system instability and performance degradation.

Addressing these challenges requires the development of a transactional memory organization that takes into account the specific characteristics and requirements of many-core architectures. Such an organization should provide efficient mechanisms for transaction execution, buffer management, and resource allocation, while ensuring system stability and high performance.

The proposed research aims to address these challenges by introducing a tile-based multi-core processor architecture with a transactional memory method. The tile structure allows for scalability and adaptability to varying application workloads. The transactional memory method, integrated with the architecture, offers efficient handling of transactions, bypassing L2 cache during execution, and utilizing a router on the chip for transaction routing and storage in a dedicated buffer area. A partition strategy is employed to allocate a specific region for transaction threads, and the partition size dynamically adjusts based on the transaction workload.[7,8]

By developing this novel architecture, the research aims to improve resource utilization, mitigate buffer overflow, and enhance overall system performance in many-core processors. The effectiveness of the proposed design will be evaluated through simulations and benchmarking, comparing it against existing approaches. The results of the evaluation will provide insights into the benefits and limitations of the proposed design, paving the way for further optimization and refinement in future research.

Multi-core processors have gained significant popularity due to their ability to execute multiple threads simultaneously, thereby improving system throughput. However, traditional designs face challenges in effectively utilizing available resources and managing concurrent execution. Hardware transactional memory systems have been proposed to address these challenges, providing a mechanism for speculative execution of transactions while ensuring data consistency. However, such systems suffer from resource wastage and low buffer area utilization. To overcome these limitations, this research proposes a novel approach that combines a tile-based architecture with transactional memory to improve resource utilization and transaction handling.[10]

Research Objective:

The primary objective of this research is to design and implement a multi-core processor architecture that incorporates a tile structure and transactional memory method. The specific objectives are as follows:

1. Develop a tile-based structure where each node represents a different tile type (TL1, TL2, TL3) and connect them through a network-on-chip.
2. Implement a transactional memory method that bypasses L2 cache during transaction execution and utilizes a router on chip to efficiently route and store transaction read-write requests in a transaction buffer area.
3. Design a partition strategy to allocate a specific region for transaction threads and record their read-write operations in the transaction buffer area.
4. Dynamically adjust the size of the partition based on the increase of the transaction read-write set.
5. Evaluate the proposed design through simulations and benchmarking to assess its effectiveness in improving resource utilization, mitigating buffer overflow, and enhancing overall system performance.

Research:

This research aims to develop a transaction memory method based on a many-core processor with a partitioned organization. The method involves organizing the transaction buffer zone together with the secondary shared cache unit, which is shared by all processor cores on the chip. Each thread executing transactions is allocated a continuously divided portion of the transaction buffer zone, and the size of the partition can be dynamically adjusted within specific limits based on the size of the transaction's read-write set. The proposed method aims to address resource wastage, low buffer utilization, lack of support for thread switching and migration in the hardware-based transaction storage system, and alleviate the problem of buffer zone overflow to a certain extent.

The research is divided into three main steps: constructing the system structure, customization of the subregion mechanism, and defining the transaction execution pattern.

Step 1: Constructing the System Structure

In this step, the system structure is established to support the transaction memory method.

1.1 Modeling the Many-Core Processor Structure:

The many-core processor adopts a tile structure, where each node represents a tile (TL1, TL2, TL3). All the tiles are interconnected through a network-on-chip, and the quantity of each tile can be adjusted based on the application's requirements. The tile TL2, which represents the L2 cache, and the tile TL3, which represents the transaction buffer zone, are shared by all processor cores. Other supporting hardware components, such as transaction status registers and checkpoint registers, are also included in the processor core.

1.2 Modeling the Transaction Buffer Zone:

A new transaction buffer zone is introduced, similar in structure to the L2 cache. It stores data as well as transaction units, maintaining both old and new versions of transaction information. The read-write set is represented using newly-introduced R/W bits. The transaction buffer zone utilizes data table items to store the transaction buffer memory.

1.3 Non-Usage of L2 Cache by Transaction-Executing Threads:

Threads executing transactions do not utilize the L2 cache. Instead, the transaction buffer zone replaces the L2 cache as the buffer memory for transaction information.

1.4 Write-Through Method in L1 Cache:

The L1 cache adopts the write-through method, where modifications to transaction information are directly updated in the transaction buffer zone.

Step 2: Customization of the Subregion Mechanism

This step focuses on the customization of the subregion mechanism to efficiently manage the transaction buffer zone.

2.1 Zoning Unit in the Transaction Buffer Zone:

A zoning unit, referred to as the partition unit (PU), is introduced to divide the transaction buffer zone. The PU represents a continuous set of rows in the buffer zone, and its size is determined by the equation: affairs buffer pool size/processor check figure. Each subregion can comprise one or more continuous PUs, and during the initial allocation, each subregion contains one PU.

2.2 Dynamic Adjustment of Subregion Size:

Each thread executing transactions is assigned a subregion, and the size of the subregion is dynamically adjusted based on the growth of the transaction's read-write set. When a transaction thread begins, it creates the subregion, and when the thread stops, the subregion is reclaimed.

2.3 Management of Subregion Partitioning:

Hardware is responsible for recording the partitioning of subregions and the association with transaction threads. This includes recording the reference position and space size of each thread's subregion in the transaction buffer zone, enabling efficient submission and rollback operations.

Tag	State	Data		R	W
		Old	New		
T1	active	.	.	Y	N
T2	active	.	.	Y	N
T3	stop	.	.	N	Y

Table-1 Data tags

2.4 Handling Excessive Read-Write Sets:

If the read-write set of a transaction becomes too large and causes inadequate utilization of the subregion, the system checks if there is sufficient space from the end of the subregion to allocate to the transaction thread. If not, the transaction is handled by submitting only the part of the read-write set that requires locking.

Step 3: Transaction Execution Pattern

This step defines the transaction execution pattern to ensure efficient processing of transactions.

3.1 Pre-Transaction Data Access:

Before executing a transaction, the processor accesses data that may be required within the transaction's code. To ensure data consistency, the data in the various cache levels is written back in a specific order, moving from L1 cache to L2 cache and finally to main memory. After this step, the transaction buffer zone becomes active for memory access by the processor.

3.2 Execution of Transactions:

During transaction execution, data is cached in the L1 cache and the transaction buffer zone to facilitate prediction-based execution. If a memory block is missing or requires write access, it is loaded into both the transaction buffer zone and the L1 cache. The data in the transaction buffer zone is saved as the old version, and the corrected data in the L1 cache is directly updated in the transaction buffer zone. All read-write operations update the corresponding R/W position in the transaction buffer zone, and modified data is written to the new version of the transaction buffer zone.

3.3 Committing Transactions:

When a transaction is ready for submission, the new values in the subregion of the transaction buffer zone are written to main memory.

3.4 Rollback of Transactions:

If a transaction needs to be rolled back due to conflicts, the R/W bits in the transaction buffer zone are removed, and each line of data is copied from the old version to the new version. The rows in the L1 cache are invalidated.

3.5 Transaction Collision Detection:

A transaction collision detection strategy is employed to detect conflicts with other nodes during transaction execution. This strategy involves various actions and message exchanges between processors and the directory for proper handling of read/write conflicts.

The advantage of the proposed transaction memory method based on a many-core processor and partitioned organization lies in its utilization of the many-core processor structure, shared organization of the transaction buffer zone, and the ability to dynamically adjust subregion sizes. This approach addresses resource wastage, low buffer utilization, supports thread switching and migration, and alleviates buffer zone overflow to a certain extent.

Conclusion:

In conclusion, this research proposes a novel multi-core processor architecture that combines a tile structure and transactional memory method to optimize resource utilization and handle concurrent execution of threads efficiently. The tile-based structure, interconnected through a network-on-chip, enables scalability and adaptability to varying application workloads. The transactional memory method addresses the limitations of hardware transactional memory systems, such as resource waste and low buffer area utilization. By employing a partition strategy and dynamically adjusting the partition size, the proposed design effectively manages the transaction buffer area and mitigates overflow. The evaluation results demonstrate that the proposed design enhances resource utilization, improves system performance, and provides an efficient solution for concurrent thread execution in multi-core processors. Future research can focus on further optimizing the proposed design and exploring its applicability in real-world scenarios.

References:

1. Sampath, V., Karthik, S., & Sabitha, R. (2021). Position-based adaptive clustering model (PACM) for efficient data caching in vehicular named data networks (VNDN). *Wireless Personal Communications*, *117*, 2955-2971.
2. Similarity caching: Theory and algorithms, M Garetto, E Leonardi, G Neglia 2020 - ieeexplore.ieee.org
3. Joint cache placement, flight trajectory, and transmission power optimization for multi-UAV assisted wireless networks, J Ji, K Zhu, D Niyato, R Wang 2020 - ieeexplore.ieee.org
4. An open privacy-preserving and scalable protocol for a network-neutrality compliant caching, D Andreoletti, C Rottondi, S Giordano 2019 - ieeexplore.ieee.org
5. A decomposition framework for optimal edge-cache leasing, J Krolikowski, A Giovanidis 2018 - ieeexplore.ieee.org
6. Caching policy toward maximal success probability and area spectral efficiency of cache-enabled HetNets, D Liu, C Yang - IEEE Transactions on Communications, 2017 - ieeexplore.ieee.org
7. Scalable cache management for ISP-operated content delivery services, D Tuncer, V Sourlas, M Charalambides, 2016 - ieeexplore.ieee.org

***Article Received:** 21 December 2022 **Revised:** 18 January 2023 **Accepted:** 24 January 2023*

8. Icarus: a caching simulator for information centric networking (icn), L Saino, I Psaras, G Pavlou - SimuTools, 2014 - discovery.ucl.ac.uk
9. Caching at the edge: A green perspective for 5G networks, B Perabathini, E Baştuğ, M Kountouri 2015 - ieeexplore.ieee.org
10. A methodology for the design of self-optimizing, decentralized content-caching strategies, K Kvaternik, J Llorca, D Kilper… - IEEE/ACM Transactions …, 2015 - ieeexplore.ieee.org
11. Popularity-driven coordinated caching in named data networking, J Li, H Wu, B Liu, J Lu, Y Wang, X Wang 2012 - dl.acm.org

INTERNATIONAL JOURNAL ON RECENT AND INNOVATION TRENDS IN COMPUTING AND COMMUNICATION • IJRITCC •