Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 6, No. 2, 2023

Analysis of Energy Consumption Optimization for Shared Cache Conflicts

Mingquan Zhang, Junwei Hu, Zihao Wu

School of Control and Computer Engineering, Engineering Research Center of Intelligent Computing for Complex Energy Systems, Ministry of Education, North China Electric Power University, Baoding 071003, China

Abstract: Multi-core shared cache conflict constraint technology for low energy consumption and low latency can be widely applied in various industries such as intelligent automobiles, aviation, aerospace, medical, industrial control, etc., and has important application backgrounds. This paper analyzes the energy consumption optimization method for multi-core shared cache conflict constraints, providing important reference for energy consumption optimization research under shared resource conflicts and promoting the development of multi-core industries.

Keywords: Shared Cache; Partitioning; Multi-core.

1. Introduction

In recent years, With the successive introduction of technological plans such as the Internet of Things, smart cities, and deep space exploration, many fields such as intelligent transportation, aerospace, industrial control, signal processing, telemedicine, smart homes, and digital consumer electronics have been establishing high-end real-time systems based on embedded multi-core processors [1]. However, the intensification of embedded multi-core dependencies, as well as shared cache storage conflicts and bus contention, make hard real-time task WCET (Worst Case Execution Time) timing analysis not only dependent on its worst-case execution path, but also on the conflict situation of shared cache used by other collaborative tasks. Therefore, the current commercial embedded multi-core shared cache management technology is difficult to provide effective real-time guarantee for hard real-time tasks.

For high-end real-time applications powered by batteries, multi-core shared cache conflict delay analysis needs to be studied in combination with energy consumption optimization. According to the relationship between embedded multi-core power consumption and performance (as shown in Figure 1, released by ARM), low energy technology is one of the important development directions in the future. By studying a multi-core shared cache conflict constraint optimization model for low energy consumption and low latency, while ensuring the effective real-time performance of hard real-time tasks, further 'pulling down' the right end of the Power curve; And enhance the bus service capability for non-hard real-time tasks. At present, the conflict constraint technology for multi-core shared cache is far from meeting the requirements of low energy consumption and low latency. Overall, the research on conflict constraint optimization for multi-core shared cache faces the main challenges:

a) How to reduce the Bank conflict delay caused by multi-core and multi-cache partitions sharing the same Bank in the process of low-power shared cache partitioning, and achieve the minimization of Bank conflict delay;

b) On the basis of a), how to establish a multi-core shared cache conflict constraint optimization model for low energy consumption and low latency, in order to minimize the energy consumption of on-chip cache.

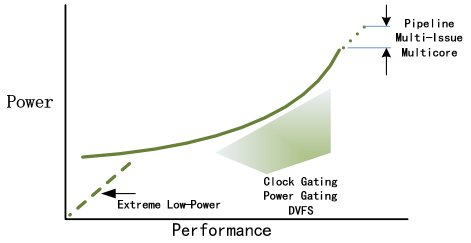

Figure 1. Relationship between power consumption and performance of embedded chip

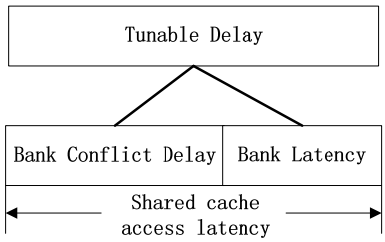

Figure 2. Classification of access latency for shared cache

Unlike traditional multi-core shared cache conflict constraint techniques, in order to reduce the Bank conflict delay generated when multi-core and multi cache partitions share the same Bank (as shown in Figure 2), we establish a reasonable mapping relationship between cache energy-saving partitions and shared cache Banks by parsing the interweaving semantics of multi-core tasks and real-time bus scheduling semantics, guided by the cache energy-saving partition capacity required by hard real-time tasks and the information obtained from parsing semantics, Implement the Bank conflict delay minimization algorithm. In the process of constructing a multi-core shared cache conflict constraint optimization model, under the condition that the hard real-time task WCET satisfies its timing constraints, the Bank conflict delay minimization algorithm is applied to minimize the Bank conflict delay, close inactive cache units, and use corresponding energy-saving mechanisms to minimize the energy consumption of on-chip cache. Therefore, achieving these requirements requires a significant amount of research work.

In summary, the multi-core shared cache conflict constraint

technology studied for low energy consumption and low latency can be widely applied in various industries such as intelligent automobiles, aviation, aerospace, medical, industrial control, etc., and has important application backgrounds. The multi-core shared cache conflict constraint optimization model studied simultaneously will also become an important reference for related work, with significant scientific research significance.

2. Dynamic Analysis of Low Energy Consumption of Shared Caches

This article studies the conflict constraint optimization technology for low energy consumption multi-core shared cache. The purpose is to study the Bank conflict delay minimization algorithm based on cache energy-saving partitioning, and then explore the multi-core shared cache conflict constraint optimization model and implementation algorithm based on this. The aim is to minimize the energy consumption of on-chip shared cache and meet the high-end real-time application needs of battery power supply. Regarding this overall goal, relevant research work can be summarized into two key issues.

2.1. Low Energy Shared Cache Partitioning

The embedded multi-core shared cache partitioning technology [2] is the research foundation for multi-core shared cache conflict constraints in the WCET computing mode, and it is also the core technology for eliminating "storage conflicts" and ensuring the effective real-time performance of hard real-time tasks. Unlike shared cache partitioning techniques based on fairness and throughput [3], embedded multi-core shared cache partitioning algorithms mainly determine the required cache partition capacity for tasks by using utility functions. They can generally be divided into two main methods: traditional shared cache partitioning and shared cache partitioning that supports energy conservation.

The traditional shared cache partitioning methods mainly include three types: Bank based shared cache partitioning, group based shared cache partitioning, and path (column) based shared cache partitioning. Although the Bank based shared cache partitioning method can eliminate Bank conflicts, it is only suitable for application scenarios where the total number of shared cache partitions required for hard real-time tasks is not greater than the total number of available Banks and the same Bank is not shared by multi-cache partitions, which is difficult to meet the needs of most high-end real-time application scenarios. Group based shared cache partitioning is currently rarely used due to its own limitations. Based on the column (path) partitioning method, its replacement strategy is changed to only occur in the assigned paths; Reference [4] implemented a path partitioning method based on the Utility function, which can dynamically obtain Utility information using ATD (Auxiliary Tag Directory) and DSS (Dynamic Set Sampling) techniques.

In recent years, scholars have increasingly focused on supporting energy-saving shared cache partitioning. In this type of method, multi-core shared cache is constrained to support energy-saving mechanisms [5], which mainly include changing the cache capacity and correlation by enabling or not enabling certain paths, using gate Vdd technology to not enable unused cache rows, and using drowsy cache energy-saving technology to turn data cache rows into low voltage mode to achieve the dual effect of energy saving and available cache row data. This type of energy-saving method can not only shut down smaller granularity unused cache components, but can also be achieved through static or dynamic partitioning. For example, reference [6] proposes an offline shared cache hybrid partitioning algorithm based on profile. It determines the cache partition capacity size for each hard real-time task by changing different path and group parameter values, based on the cache utility function and tolerance threshold. As shown in Figure 3, tasks T1, T2, T3, and T4 are statically partitioned according to the above rules, and energy-saving is achieved by not enabling unused paths, groups, or cache rows.

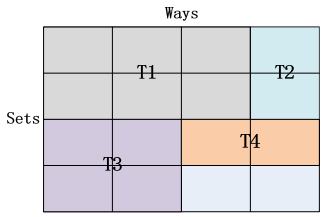

Figure 3. Shared Cache Hybrid Partition

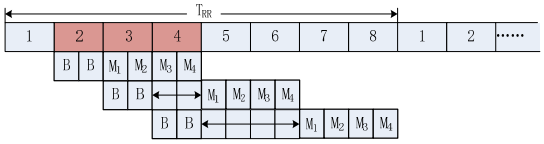

Figure 4. Bank conflict when sharing the same bank with multi-core

The shared cache partitioning algorithms mentioned above generally have a flaw: when multi-core and cache partitions share the same Bank, it can cause the Bank conflict problem shown in Figure 4. In the process of low-power shared cache partitioning, it is necessary to minimize the delay of bank conflicts and cache energy consumption. By reducing Bank conflict latency and eliminating Storage conflicts, we ensure the effective execution of hard real-time tasks, while also ensuring the energy efficiency of shared cache partitioning results.

2.2. Multi-Core Shared Cache Conflict Constraint Optimization Model

As mentioned above, the Bank conflict delay minimization algorithm based on cache energy-saving partitioning is used to eliminate or reduce the Bank conflict delay, which also provides a foundation for real-time energy-saving bus optimization based on time distance.

Taking cache energy consumption calculation as an example, the current multi-level cache energy consumption calculation methods can be roughly divided into two categories:

Single core multi-level cache energy consumption calculation is the calculation of the dynamic and static energy consumption of each level of cache used in task execution. Reference [7] proposes a dynamic reconfigurable cache structure and its energy consumption calculation method, where the dynamic energy consumption is composed of the number of cache accesses multiplied by the energy consumption of one cache access and the number of cache misses multiplied by the energy consumption of each cache miss. The static energy consumption is the static power consumption of the cache multiplied by the time of task execution. In addition, in multi-level cache energy consumption calculation, it is necessary to correctly handle

the overlapping parts of energy consumption calculation. Reference [8] proposes an energy-saving algorithm based on reconfigurable cache structure; References [9,10] proposed a low energy AH Cache design method based on runtime caching behavior.

The energy consumption calculation method for multi-core shared cache partitioning [11-13] is based on the mapping relationship between hard real-time tasks and cores, L1 cache configuration, shared cache partitioning, and other joint effects. For private cache and shared cache partitions that meet the timing constraints of hard real-time tasks, the energy consumption and total energy consumption are calculated based on the above energy consumption calculation method. For example, reference [13] proposes an energy consumption calculation method based on DCR (Dynamic Cache Reconfigure) L1 cache configuration and path cache partition configuration.

3. Conclusion

Although the above methods achieve good results under their respective assumptions, their assumptions do not match the actual situation in high-end real-time applications. For example, in high-end real-time applications, in addition to containing multi-core hard real-time tasks, there are also non hard real-time tasks; Secondly, it not only involves cache energy-saving issues, but also real-time bus energy-saving issues. On the other hand, these algorithms often only partition specific features for a certain type of shared cache, and different features have different energy-saving effects on shared cache partitioning. For example, the issue of the number of paths in cache partitions, due to the parallel search of each TAG, the more paths there are, the lower its energy-saving effect, and so on.

Starting from the semantic parsing of high-end real-time applications, the mapping relationship between tasks and processing cores can be established; Based on information such as cache energy-saving partition capacity and task interleaving semantics, establish a reasonable mapping relationship between cache energy-saving partitions and shared cache banks under the guidance of real-time bus semantic information, and implement the Bank conflict delay minimization algorithm.

Acknowledgments

This paper was financially supported by “the Fundamental Research Funds for the Central Universities (2020MS122)”.

References

[1] Jongsok Choi, Stephen D. Brown, Jason H. Anderson. From Pthreads to Multicore Hardware Systems in LegUp High-Level Synthesis for FPGAs[J]. IEEE Transactions on Very Large Scale Integration Systems, 2017, PP(99):1-14.

[2] Luís Fernando Arcaro, Karila Palma Silva, Rômulo Silva De Oliveira. On the Reliability and Tightness of GP and Exponential Models for Probabilistic WCET Estimation[J]. ACM Transactions on Design Automation of Electronic Systems, 2018, 23(3):1-27.

[3] Noemí Pérez-Macías, José- Luis Fernández-Fernández, Antonio Rúa Vieites. The impact of network ties, shared languages and shared visions on entrepreneurial intentions of online university students[J]. Studies in Higher Education, 2019(26): 1-15.

[4] M. K. Qureshi and Y. N. Patt. Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches. In MICRO, 2006, pages 1-10.

[5] Poorvi Jain, Bishnu Prasad Das. On-Chip Threshold Voltage Variability Estimation Using Reconfigurable Ring Oscillator [J]. IEEE Transactions on Semiconductor Manufacturing, 2019, 32(2):226-235.

[6] R. Reddy and P. Petrov. Cache Partitioning for Energy-Efficient and Interference-free Embedded Multitasking. ACM Transactions on Embedded Computing System, 2010, 9(3),16:1-16:35.

[7] Chuanjun Zhang, Frank Vahid and Walid Najjar. A Highly Configurable Cache Architecture for Embedded Systems. In IEEE ISCA2003,2003, pages:1-9.

[8] Jagadish B. Kotra, Haibo Zhang, Alaa R. Alameldeen. CHAMELEON: A Dynamically Reconfigurable Heterogeneous Memory System[C]// 2018 51st Annual IEEE/ ACM International Symposium on Microarchitecture (MICRO). ACM, 2018.

[9] Francesco Cavenago, Lorenzo Voli, Mauro Massari. Adaptive Hybrid System Framework for Unified Impedance and Admittance Control[J]. Journal of intelligent & robotic systems: Theory & applications, 2018, 91: 569-581.

[10] Jun Wu, Fenglei Ni, Yuanfei Zhang. Smooth transition adaptive hybrid impedance control for connector assembly[J]. Industrial Robot, 2018, 45(2): 287-299.

[11] Liu L, Li Z, Chen Y, et al. HReA: An Energy-Efficient Embedded Dynamically Reconfigurable Fabric for 13-Dwarfs Processing[J]. 2017, PP(99):1-1.

[12] Yang J, Hui G, Lei C, et al. The JTAG Circuit Design of Accomplishing Dynamic Reconfiguration[J]. 2018, 131:454-461.

[13] Weixun Wang, Prabhat Mishra and Sanjay Ranka. Dynamic Cache Reconfiguration and Partitioning for Energy Optimization in Real-Time Multi-Core Systems. In ACM DAC2011, 2011,948-953.