The Application and Research of Multi-core Processing in Avionics System

Zhan Wentao[1,2(✉)], Niu Wei[2], Yuan Ji[2], Wang Xiaobo[2], and Yang Liu[2]

[1] School of Civil Aviation, Northwestern Polytechnical University, Xi'an, China
zhanwt1984@qq.com

[2] Xi'an Aeronautics Computing Technique Research Institute, Xi'an, China

Abstract. Multi-core processing is widely used for its high performance, but in avionics system, there are some time behavior interferences caused by cache sharing and bus sharing etc. To address application aspects of multi-core processing in avionics system, this paper made tests and analyses, which showed that dual-core processor can get doubled performance in some specified situations. Gathering service experience of multi-core processing in less critical system is also a usable way towards critical system. Then this paper presents a scheduling configuration of SMP mode multi-core AVIC OS to meet the needs of mixed critical applications. This envisioned that multi-core processing will support the development of next-generation avionics systems.

Keywords: Multi-core processing · Sharing interference · Application mode

1 Introduction

Avionics system is the core component of aircraft, which is essentially a complex real-time system, that is, the system must correctly respond to the changes of external physical processes within the specified time range [1]. The system is also considered as safety critical system because its failure can cause catastrophic consequences such as loss of lives [2]. In modern aircraft avionics, more and more functions traditionally transition from federated to integrated system architectures. In federated architectures, computers connected by buses usually host one application, while in integrated architectures, such as the Integrated Modular Avionics (IMA) [3], many applications are often integrated on powerful platforms of mixed criticality. This can improve systems in several dimensions, such as weight, space, energy consumption and maintainability etc. Now, next generation of new aircraft avionic programs will integrate more new functions, including safety functions, information services and comfort features, which will greatly increase the demand of avionics processing performance. The traditional approach to improve processing performance was to increase CPU clock frequency or increase pseudo-parallel processing on code level. However, with the development of semiconductor technology based on Moore's law, this method has reached its limit. The increase of CPU frequency will lead to sharp rise in power consumption, which will bring trouble to heat dissipation. In addition, increasing CPU frequency will also introduce more and more problems

Chinese Society of Aeronautics and Astronautics: CASTYSF 2022, LNEE 972, pp. 412–419, 2023.
https://doi.org/10.1007/978-981-19-7652-0_38

caused by chip internal and external crosstalk, and chip signal integrity. As one of the ways of parallel processing, multi-core processing has become an important measure to improve processing capacity of processors in recent years. In avionics system, multi-core processing is increasingly favored by system developers because of its advantages of high performance, high integration and low power consumption.

2 Challenges of the Application of Multi-core Processor

2.1 The Architecture of Multi-core Processor

Software designer of single core processor can directly improve the performance by deploying the existing software to the CPU with higher frequency and faster speed without modification. CPU is the only shared resources in single core system. Once a task obtains the execution right of CPU, it will be able to control other hardware resources of the system. But in multi-core system, as shown in Fig. 1, resource sharing exists in multiple dimensions. Parallel tasks not only need to share multiple processing cores, but also share hardware resources such as cache, bus, memory controller and main memory. The sharing among multiple dimension leads to interdependence and interference among tasks, which leads to the high unpredictability of the overall performance and time behavior of the system, and it is difficult to guarantee the logical correctness [4].

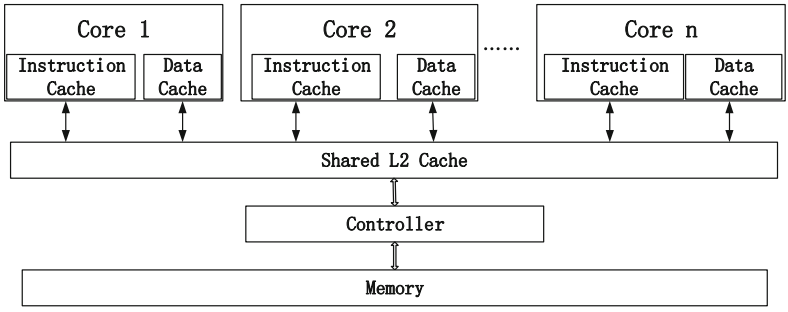

Fig. 1. Architecture of shared resource of multi-cores

2.2 The Problem of Shared Resources

Cache sharing interference. In multi-core system, a possible cache interference situation is that one core will continuously drive out cache blocks useful to another core, which will not bring significant performance improvement, but also increase the cache miss rate of the system and reduce system performance. Reference [5] shows that the shared cache interference between cores will increase the completion time of tasks by 40% compared with that running on a single core. In addition, the disordered preemption timing among cores makes it very difficult to analyze the system behavior. Inter tasks interference leads the WCET of a task not only rely on the task itself, but also affected

by other tasks executed in parallel [6]. Moreover, the interaction among tasks explodes exponentially with the increase of the number of cores.

Bus sharing interference. Each core needs to access the off chip memory through the shared bus, which will cause the interference of the shared bus and increase the response time of the task on multi-core system. The bus access request of the task does not carry the priority attribute, so the bus will be reordered according to the access of the task, which may cause the access request service time of high priority task later than that of the low priority task. The delay of task bus access request, that is, determining the bus access service order, depends not only on the bus arbitration mechanism, but also the bus access volume of parallel tasks concurrently. So how to analyze the response time considering the interference of shared bus is still a challenging task.

Software interference. Software interference only occurs when the software is running concurrently. Interference severity mainly depends on the software architecture and the way the software utilizes shared resources of the cores. The existing operating system and software applied in avionics system comply with ARINC653 standard, which has been implemented and applied on a single core platform without considering the characteristics of multi-core platform. Previously, the system software only needed to consider avoiding shared resource conflict between partitions running concurrently. Now the system software must prevent unintended coupling and must not introduce additional interference due to concurrent access to shared resources among different cores and different partitions.

3 Research Development of Multi-core Processing in Avionics

Reference [7] proposed a static analysis method to solve the impact of shared cache on WCET to a limited extent. Considering the complexity of the problem, these solutions need to introduce some assumptions to simplify the problem. Such as only analyze the behavior of instruction cache and ignore the influence of data cache, and assuming that the data can always be accessed in data cache, this method cannot be verified on the actual multi-core computing platform. These simplified assumptions also limit the availability of the proposed solution. Therefore, some scholars turn to specific design to avoid or reduce the interference between cores, for example, time division multiple access (TDMA) arbitration mechanism [8]. Rosen described a scheme of TDMA bus scheduling on multi-core computing platform [9]. All tasks are scheduled based on the off-line scheduler and the scheduler is written to the memory of bus arbiter. When the system is running, the bus access request is coordinated and controlled according to the schedule, so as to avoid bus interference. However, these studies need system customization and re-development, which induced high cost and cannot meet the needs of generalization and marketization. Especially in avionic multi-core applications, if it cannot support incremental certification, the application will be greatly restricted. As described by Corradi [10], the semiconductor industry is facing technical and economic challenges, where fewer chip suppliers can afford the investments, safety-related and avionic semiconductor market is too small for semiconductor companies to provide economically competitive safety specific solutions compliant with defined certification standards. Therefore, it is expected that multi-core devices used in safety-related and

avionics systems will have to select COTS chip. This is also in line with the goal of reducing life cycle costs for avionics system. Therefore, starting with the application mode, that is, the software architecture of the operating system, is also an optional method [11]. Multi core application modes include AMP (asymmetric multi-processing) and SMP (symmetric multi-processing) [12–14].

In AMP mode, each core is configured with an independent operating system, and the blueprint is configured to manage shared resources. The operating environment of AMP is similar to the traditional single core system, so it has good portability and is easy to inherit the software of single processor. AMP only needs to select an operating system that can provide distributed programming mode to maximize the use of multi-core. Applications on one core are allowed to communicate transparently with applications or system services on another core. In AMP mode, multiple partitions are executed in parallel, and all partitions run on their corresponding cores at the same time. When the number of processors increases, the system performance can be significantly improved.

In SMP mode, there is an operating system controls the simultaneous activities of all cores. Since these tightly coupled processors access a common memory area, they must synchronize with each other through a communication mechanism based on low latency shared memory. Since there is only one operating system, local communication between different cores is completed through process communication, which greatly improves the communication efficiency between processes. At the same time, it also allows the operating system to dynamically schedule any process to run on any core. SMP provides much more flexibility and a better load balancing than that of AMP. But The SMP software layer is more complex since it needs to protect its internal data and the internal data need to be arranged very carefully by system.

Incremental authentication is also a requirement that must be considered in multi-core integration, especially in civil avionics, traditional avionics software has completed certification and the process has been proved to be effective by the airworthiness authority. In multi-core processing system, how to inherit existing certification results and avoid providing additional interference related information to certification authority are also the key balancing factors for the system user.

4 The Promote Methods and Potential Solutions

4.1 The Test Result of AMP Mode Application

In AMP mode, each core runs its operating system instance, so the resources occupied by operating system instances increased with the number of cores. Therefore, the performance proportion of dual core is doubled in theory while the resource occupation is small, especially when there is no conflict in memory access. Based on a dual core processor platform, this paper introduces AVIC operating system conforming with ARINC653 standard, hosting the spec cpu2006 test cases which tests the integer performance and floating-point performance of the processor. These cases mainly tests the real computing performance of the processor with relatively few memory accesses. The test results are shown in Table 1 and Fig. 2. The smaller the test value, the better the performance. It can be seen that the measured performance of each core in dual core is the same as that

of single core activated, which means that dual core can indeed achieve double performance improvements to a certain extent. It should be noted that in F_433_milc test case, the performance of dual core is not double in all, which may related to the fact that the test case uses more shared storage resources.

Table 1. Dual core processor spec 2006 integer and floating point benchmark results

Test case		Dual core		Single core
		Core0	Core1	
I_401_bzip2	Integer test case	1050.5	1050.6	1001
I_429_mcf		429.9	429.9	384.1
I_456_hmmer		1459.7	1459.7	1452.2
I_458_sjeng		3681.7	3682.1	3451.4
I_462_libquantum		112.8	112.9	103.1
F_433_milc	Float test case	255.0	255.3	137.9
F_470_lbm		764.2	764.2	764.2
F_482_sphinx3		276.2	276.2	267.4

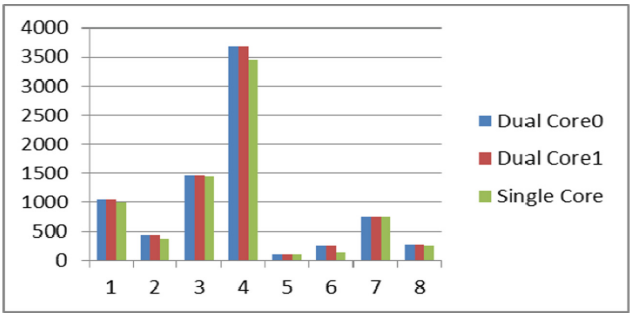

Fig. 2. Dual core processor spec 2006 integer and floating point benchmark results

4.2 Low Critical Application

Due to its good compatibility and inheritance, AMP mode is considered as the first step of multi-core processing application in avionics system. However, due to the limited number of cores and application scenarios, its performance improvement is also limited. In order to adapt more cores and significantly improve performance, a more effective way is to use SMP mode operating system with better flexibility and scalability. However, SMP mode needs to provide more additional evidence to airworthiness authority to ensure that the whole design process is approved. In fact, due to intellectual property restrictions, almost no suppliers are willing to provide detailed design documents of

processor and operating system software to prove that it complies with the certification process of DO-254 or DO-178. An alternative way is to provide service experience to certification authority and prove that there is no problems happened with the operating system and multi-core processing when in service. If the highest IDAL of multi-core processing system is below C, the authentication process does not require equipment and software to conduct interference analysis, so as to avoid additional evidence collection process. Therefore, multi-core processing system can be firstly applied in the low critical aviation system to gather in-service data and problem reports as a basis for the application of mixed criticality in avionics system.

4.3 An Applicable Schedule Configuration for SMP

In fact, future IMA platform should not only meet high performance requirements, but also meet the requirements of lower criticality and critical applications running simultaneously. These applications may also be consistent with ARINC 653 standard. The operating system configuration must support exclusive access of platform resource for the most critical applications. The operating system AVIC OS is a good example that can be configured to meet the needs of mixed critical applications, from which less critical applications can get high performance and highly critical applications can get exclusive resources for determinism and isolation. Figure 1 shows a possible configuration of the AVIC OS partition scheduler.

The configuration is assumed to be based on a quad-core CPU, the major time frame is divided into three time partition windows. To avoid interference on hardware and software level, the critical application that is running on partition 3, will only execute on core '3' during 'T3' time slice. Although it seems to waste CPU time, it ensures critical application have exclusive access to a single core and have exclusive access to entire platform resource during its allocated time window, which guaranteed highest isolation (Fig. 3).

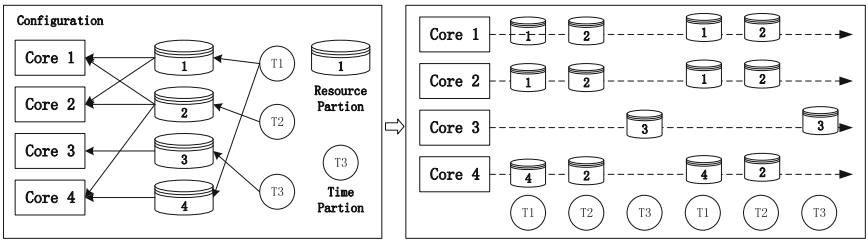

Fig. 3. A configuration using the AVIC OS Scheduler for mix-critical application

5 Conclusion

In addition to IMA platforms, the use of multi-core processing is necessary for future avionics system to meet high performance requirements. Dual-core CPUs using AMP mode are suitable for scenarios with less memory access and high computational requirements for performance improvement. Multi-core processing needs to be firstly introduced in less critical systems to gather in service experience and then be applied in critical systems with no problems. We also demonstrate a schedule configuration of multi-core platform with AVIC OS for critical applications with exclusive resources. Multi-core devices are envisioned to support the development of next-generation avionics systems.

References

1. Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications, 2nd edn. Springer-Verlag (2011)
2. J. Athavale, R.M., Paulitsch, M.: Flight safety certification implications for complex multi-core processor based avionics systems. In: Proceedings of the IEEE 25th International Symposium on On-Line Testing and Robust Syst. Design (IOLTS'19), pp. 38–39 (2019). https://doi.org/10.1109/IOLTS.2019.8854415
3. Watkins, C.B., Walter, R.: Transitioning from federated avionics architectures to integrated modular avionics. In: 26th AIAA/IEEE Digital Avionics Systems Conference (DASC) (2007)
4. Chen, G., Guan, N., Lü, M.S., Wang, Y.: State-of-the-art survey of real-time multicore system. Ruan Jian Xue Bao/J. Softw. **29**(7), 2152–2176 (in Chinese) (2018). http://www.jos.org.cn/1000-9825/5580.htm
5. Kim, H., Kandhalu, A., Rajkumar, R.: A coordinated approach for practical OS-level cache management in multi-core rea1-time systems. Real-Time Syst. **8114**, 80–89 (2013)
6. Lü, M.S., Guan, N., Wang, Y.: Survey of cache analysis for worst-case execution time estimation. Ruan Jian Xue Bao/J. Softw. **25**(2), 179−199 (in Chinese) (2014). http://www.jos.org.cn/1000-9825/4529.htm
7. Li, Y., Suhendra, V., Liang, Y., Mitra, T., Roychoudhury, A.: Timing analysis of concurrent programs running on shared cache multi-cores. In: Proceedings of the IEEE Real-Time Systems Symposium, pp. 56–57. IEEE, Washington (2009)
8. Kelter, T., Falk, H., Marwedel, P., Chattopadhyay, S., Roychoudhury, A.: Bus-aware multicore WCET analysis through TDMA offset bounds. Real-Time Syst. **6794**(29), 3–12 (2011)
9. Rosen, J., Andrei, A., Eles, P., Peng, Z.: Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip. In: Proceedings of the 28th IEEE Int'l Real-Time Systems Symposium Tucson, pp. 48–60. IEEE (2007)
10. Corradi, G.: Tools, architectures and trends on industrial all programmable heterogeneous MPSoC (KeyNote). In: Proceedings of the 29th Euromicro Conference on Real-Time Systems (ECRTS'17) (2017)
11. Fuchsen.: How to address certification for multi-core based IMA platforms: current status and potential solutions. IEEE. IEEE (2010)
12. Agrou, H., Sainrat, P., Gatti, M, et al.: Mastering the behavior of multi-core system to match avionics requirements. In: Proceedings of the 31st Digital Avionics Systems Conference, pp. 6E5-1–6E5-12. IEEE/AIAA, Williamsburg (2012)

13. Nowotsch, J., Paulitsch, M.: Leveraging multi-core computing architectures in avionics. In: Proceedings of the 9th European Dependable Computing Conference, pp. 132–143. Sibiu IEEE (2012)
14. Crespo, A., Masnano, M., Coronel, J., et al.: Multi-core partitioned systems based on hypervisor. In: Cape Town. Proceedings of the 19th IFAC World Congress, pp. 12293–12298. IFCA (2014)