

Barco Defense & Aerospace

Application of multi-core CPUs in a safety-critical context

Content

Introduction
Increasing demand for processing power
Multi-core CPUs in safety-critical applications
Resource sharing versus determinism
Mitigation strategy
Conclusion

Author

Brecht Baert
Steven Luys

INTRODUCTION

Driven by the ongoing challenge to reduce the overall operational cost and in particular the fuel consumption of commercial aviation, a lot of engineering effort is being invested in reducing the weight and power consumption of the avionics building blocks. A good example of this is the amounts of research and development effort spent on Integrated Modular Avionics (IMA) platforms that have to decrease the overall avionics processing infrastructure and the related maintenance costs.

Considering the fact that more and more functionalities are being transferred from dedicated stand-alone modules to more integrated software solutions, one can understand the constant need to create more powerful processing platforms.

A second effect is that software applications themselves are becoming increasingly complex (e.g. the Enhanced Vision System (EVS) application and the Synthetic Vision System (SVS) applications), resulting in a constant demand for greater processing power.



Figure 1: Enhanced Vision System application



Figure 2: Synthetic Vision System application

INCREASING DEMAND FOR PROCESSING POWER

Until a few years ago, extra processing power was obtained by increasing the number of transistors on the silicon chip, in accordance with Moore's law which states that "the number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years."

Over the years, the processing architectures have been miniaturized so that the speed of the core clocks could grow exponentially. Additionally, the internal structures of the devices have been parallelized resulting in a higher instruction throughput. This is called 'Instruction Level Parallelism' (ILP).

Although this trend has been going on for decades, conventional semiconductor technology finally seems to be confronting the boundaries of physics.

To further increase the performance, central processing unit manufacturers have started to multiply the number of processing cores in the devices. This is called Thread Level Parallelism (TLP).

Today, dual-core, quad-core, hexa-core and even octal-core processors have become commodities in commercial off-the-shelf computers.

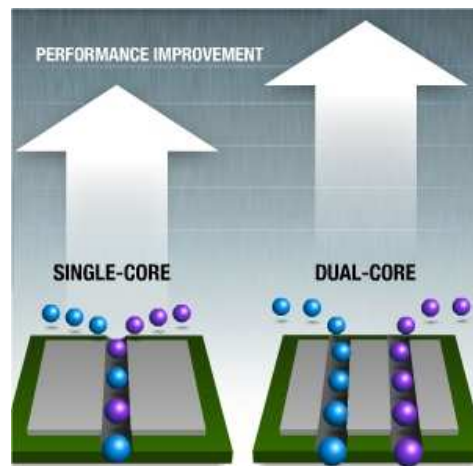


Figure 3: Enhanced Vision System application

MULTI-CORE CPUS IN SAFTETY CRITICAL APPLICATIONS

The need for more powerful processing platforms in the avionics industry - and the availability of multi-core devices in general - oblige companies developing avionics processing platforms to investigate the use of multi-core devices.

But to do this effectively, it is of utmost importance to analyze possible certification pitfalls that multi-core devices can bring with them.

From a commercial application point-of-view, a multi-core CPU is intended to parallelize the execution of instructions as much as possible. Therefore, operating system builders have introduced hypervisors with load-balancing mechanisms that can balance different threads over different cores in order to complete the scheduled tasks as fast as possible.

However, if we would apply this principle to a safety-critical application, it jeopardizes the application's most important characteristic: determinism.

RESOURCE SHARING VERSUS DETERMINISM

Unlike a traditional single-core processor, the cores inside multi-core processors have to share several resources.

Figure 4 illustrates a generic dual-core design:

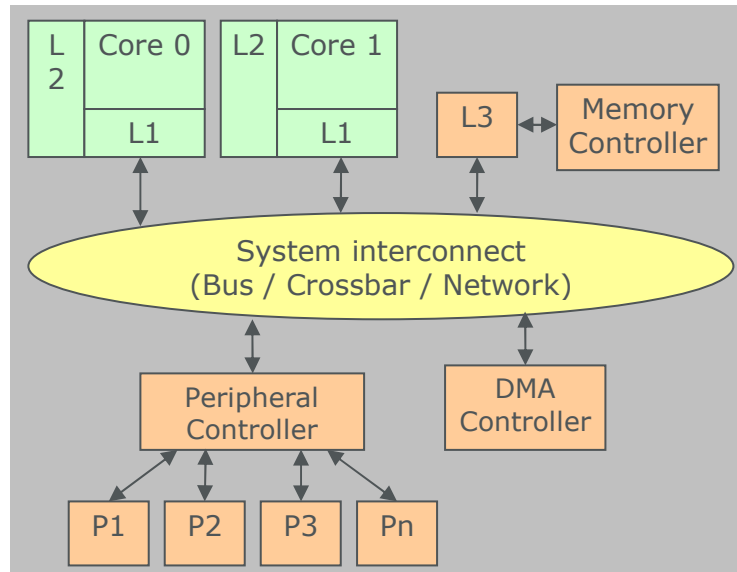


Figure 4: Enhanced Vision System application

Whereas each core usually has its own Level 1 and Level 2 caches, they often share a common Level 3 cache and a memory controller. Whether peripheral devices are embedded in the processing device or made available through a separate companion chip, they will have to be shared in one way or another as well.

Usually, a system interconnect will link the cores with the memory complex and peripheral controllers. Resource sharing increases the possibility of certain consequences that can seriously reduce the system's determinism or integrity. Possible consequences of resource sharing are:

- Latency: the time interval between an event and a consecutive response increases which decreases the overall response time of the system
- Jitter: the response time deviates from the normal response time
- Throughput: the net transfer of data to and from one core decreases
- Lockout: one core blocks a resource in such a way that the other core cannot gain access to it for a period of time
- Deadlock: two devices alternately lock each other due to badly architected resource sharing

MITIGATION STRATEGY

Using multi-core CPUs in a safety-critical context requires a strong strategy to mitigate the potential deterioration of the system's determinism.

Device selection

Selecting the proper device requires careful evaluation of architectural and processing criteria. In addition to the obvious selection criteria, such as maximum clock rate and power dissipation, architectural benefits and possible disadvantages should also be assessed. Preferred devices are for example: devices with multiple memory controllers that could be assigned to dedicated cores; devices with a very high bandwidth interconnect instead of a traditional bus architecture; and so on.

The following figures illustrate popular multi-core architectures:

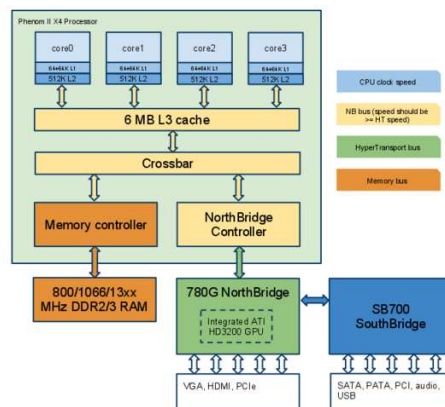


Figure 5: Intel Core i7

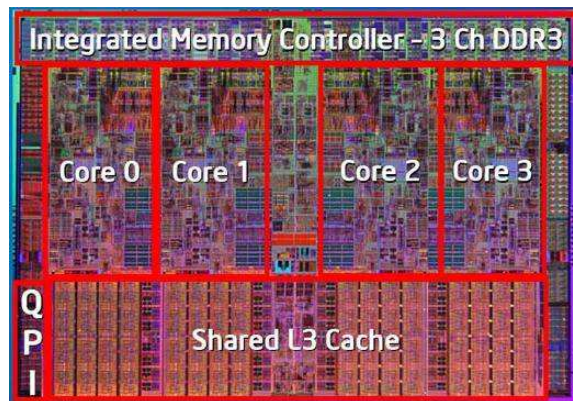


Figure 6: AMD Phenom II

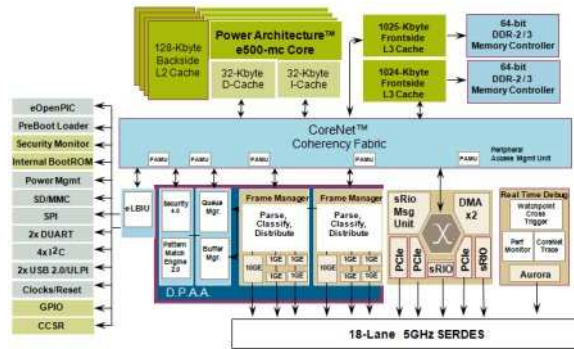


Figure 7: Freescale QorIQ P4040

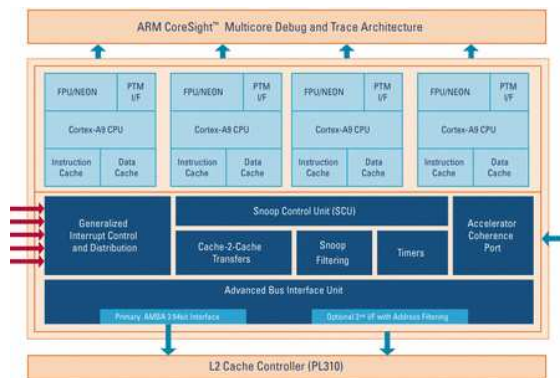


Figure 8: ARM Cortex A9 multi-core

OS usage

As stated above, the use of standard commercial OS algorithms to balance the different threads cannot be justified in a safety-critical context. In other words, Symmetric Multi-Processing (SMP) should not be used. Instead, an Asymmetric Multi-Processing (AMP) model should be established, in which each core runs its independent instantiation of the OS (or even instantiations of different operating systems). This approach guarantees that specific tasks or threads are locked to a certain processing core.

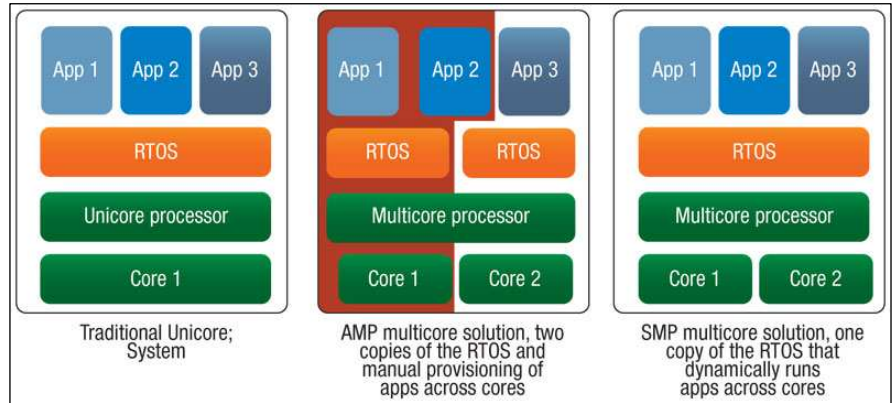


Figure 9: Enhanced Vision System application

Memory usage

Solid knowledge of how the memory is organized and the impact of concurrent accesses to the memory is an important aspect of the mitigation strategy. For shared caches, the best approach is to allocate a fixed portion to each of the cores. In the extreme, the whole cache could be assigned to one of the cores. This way, cache flushes from and to other components. When a certain memory device needs to be shared because there is only one single memory controller, one needs to know the exact impact of concurrent accesses. This includes not only a categorization of the different combinations of concurrent accesses, but also the impact on throughput for all of these combinations. Then, one can predict worst-case results with this measurement table.

Peripheral usage

The path between the cores and the peripherals can contain numerous bottlenecks. That's why it's crucial to determine which peripherals are accessible to other components, including the different cores. One example of a safe approach is to assign all peripheral access to one core, which acts as an IO server for the other cores.

Analysis of architectural design

In order to be able to predict the system's behavior, it's important to understand the operation of key parts of the device. One of those components is the bus or network, which interconnects the cores, the memory complex and the peripherals. It is important to know and understand the way in which device requests are getting scheduled inside this system in order to determine possible consequences (e.g. the maximum latency).

Inter-core communication & synchronization

Tasks running on different cores cannot always be executed totally independently of each other. Although inter-core communication is not always avoidable, one should be careful of linking execution chains running on different cores. The safest way to implement inter-core communication is by using shared memory regions. These regions should be set up in such a way that only one producer of the data is possible and one (or perhaps more) consumers.

Impact analysis

Even when all of the above measures have been taken into account, it's still possible that some areas in the design cannot be fully covered. In order to close the loop, an impact analysis should be made in which, theoretically and/or empirically, the impact of concurrent events upon each other is quantified. For example, the impact of concurrent DMA and core operations upon the system memory is quantified.

CONCLUSION

Multi-core devices can help meet the constantly growing demand for more powerful processing platforms for safety-critical applications. However, to preserve the determinism, and to assure the certifiability of the platform, one should pay special attention to the device selection, have solid knowledge of the internal operation of the device, and take design decisions that mitigate the potential deterioration of the system's determinism.

The "MOSArt" open system software platform has been developed to encapsulate this complexity and to enable software application providers and integrators to benefit from multi-core processing power, without having to mitigate all possible consequences of resource sharing.

CONTACTS

For all questions or remarks please contact:

brecht.baert@barco.com

or

steven.luys@barco.com