Parallel Programming Models for Space Systems

Exploiting the performance of current multi-core processors (e.g.NGMP) requires the usage of appropriate parallel programming models. Such a requirement is further exacerbated with the advent of next-generation many-core heterogeneous embedded architectures (e.g. Kalray MPPA), in which parallel programming becomes mandatory to exploit the massive parallel computation capabilities. OpenMP, the de-facto standard for shared memory architectures in the high performance domain, is increasingly being considered in the embedded domain. Originally focused on massively data-parallel, loop-intensive applications, the latest specification of OpenMP (version 4.5) has evolved to consider a very sophisticated tasking model supporting fine-grained and irregular parallelism. Moreover, it also incorporates new features to couple a main host processor to one or more many-core accelerators, where highly parallel code kernels can be offloaded for improved performance/watt.

Our vision is that OpenMP is an excellent choice for current and future real-time embedded systems for a twofold reason: First, it provides the abstraction level required to program parallel applications, while hiding the complexities of multi- and man-ycore architectures. Second, it facilitates the migration of real-time embedded systems from multi- core to many-core platforms. Unfortunately, OpenMP adopts a parallel execution model that differs in many aspects from the real-time execution model: The programming interface and the runtime scheduler are completely agnostic to any timing requirement that the target application may have. Moreover, current implementations of OpenMP are not designed for embedded environments in which the execution is constrained by hardware resources, operating systems (OS) or application requirements.

In the "Parallel Programming Models for Space Systems” project, we evaluated the use of OpenMP in the space domain with a twofold objective. First, to demonstrate the benefits of using OpenMP4 in terms of: performance speed-up, programmability and time analysability. Second, to identify the challenges that future implementations of OpenMP must address, by evaluating the implications of using OpenMP in terms of system resources, i.e. memory footprint, operating system services and size of application memory working set.