Design Tools Session

Resource management for embedded multi-core platforms

C. Ykman-Couvreur[1], R. Obermaisser[2], C. El Salloum[2], M. Goedecke[3], R. Zafalon[4], L. Benini[5]
[1]*IMEC, Belgium, ykman@imec.be*
[2]*TUV, Austria, {ro,sc}@vmars.tuwien.ac.at*
[3]*Infineon, Germany, Michael.Goedecke@infineon.com*
[4]*ST Microelectronics, Italy, roberto.zafalon@st.com*
[5]*UNIBO, Italy, lbenini@deis.unibo.it*

Abstract

In the context of the project GENESYS (GENERIC Embedded SYStem Platform), we have developed a Resource Management (RM) framework to facilitate the application development and to support efficient and optimal management of all platform resources. This is made possible by: (1) providing generic services to separate the application functionality from the underlying platform, (2) dynamically adapting to changing context and transparently optimizing platform resource usage, (3) supporting a holistic view of the platform resources, not only to find optimal trade-offs in application mapping, but also to detect failures and anomalies. The context can be changing for several reasons: applications are dynamically activated, users dynamically change their requirements, and physical constraints of the platform (e.g. temperature, age, battery life) are changing.

The approach followed by our RM is component-based. First, each application is considered as a set of communicating jobs, and exactly one job is mapped on one platform component. This makes the platform more robust in the presence of faulty components. This also simplifies the implementation of power shut-off techniques. Secondly, whereas the functionality of a job is fixed, there may be a number of specific algorithms or implementations with run-time tunable parameters for a given job. They allow trade-offs between quality levels and available resources. Finally, each job can consist of multiple communicating tasks. The RM manages jobs and communication between jobs, whereas each IP core manages tasks of any job mapped on it and communication between tasks.

The services provided by our RM are partitioned into several managers. The IP core manager is responsible for the interface between the RM and the IP cores of the platform. It is not visible to the application programmer. The RM transparently conforms to the local practices and policies of the platform components. The communication manager allows exchange of messages of different types and characteristics: sporadic messages for events, periodic messages for states, and streaming data. The Quality-of-Experience (QoE) manager is responsible for (re)negotiating with the user the Quality-of-Service (QoS) requirements of each active application, for selecting an utility function (e.g., energy/power consumption, performance, reliability, revenue) to be optimized by the RM, and for determining which applications should be revoked when the platform becomes overloaded. The job manager is responsible for selecting an optimal implementation for each active job, taking into account the QoS requirements chosen by the user, while optimizing the utility function and making sure that distinct jobs will be mapped on distinct components of the platform. This run-time optimization problem can be modeled as a Multiple-choice Multiple-dimension Knapsack Problem (MMKP), which is known to be NP-hard. Since state-of-the-art heuristics for real-time systems are still too slow for RM of embedded multi-core platforms, a new fast and lightweight heuristic for finding near-optimal solutions has been developed. Resource monitoring is also needed, not only to track the load of the platform resources, but also to detect any platform overload.