Incremental Mapping and Scheduling for Distributed Heterogeneous Real-Time Systems

Paul Pop, Petru Eles, Zebo Peng
Dept. of Computer and Information Science, Linköping University, Sweden
{paupo, petel, zebpe}@ida.liu.se

Extended Abstract

A characteristic of research efforts concerning the codesign of real-time systems is that researchers concentrate on the design, from scratch, of a new system optimized for a particular application. For many application areas, however, such a situation is extremely uncommon and only rarely appears in design practice. It is much more likely that one has to start from an already existing system running a certain application and the design problem is to implement new functionality (including also upgrades to the existing one) on this system. In such a context it is very important to make as few as possible modifications to the already running applications. The main reason for this is to avoid unnecessarily large design and testing times. Performing modifications on the (potentially large) existing applications increases design time and, even more, testing time (instead of only testing the newly implemented functionality, the old application, or at least a part of it, has also to be retested). However, this is not the only aspect to be considered. Such an incremental design process, in which a design is periodically upgraded with new features, is going through several iterations. Therefore, after new functionality has been implemented, the resulting system has to be structured such that additional functionality, later to be mapped, can easily be accommodated.

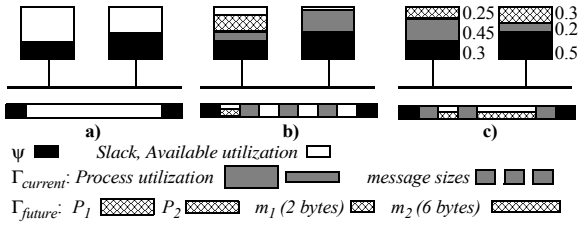
In this paper, we concentrate on scheduling and mapping of hard real-time systems which are implemented on distributed architectures. Process scheduling is based on a static priority preemptive approach while the bus communication is statically scheduled. We consider mapping and scheduling for hard real-time systems in the context of a realistic communication model. Because our focus is on hard real-time safety critical systems, communication is based on a time division multiple access (TDMA) protocol, the time-triggered protocol (TTP), as recommended for applications in areas like, for example, automotive electronics.

We have considered the design of distributed embedded systems in the context of an incremental design process as outlined above. This implies that we perform mapping and scheduling of new functionality so that certain design constraints are satisfied, and: (a) the existing applications are disturbed as little as possible, and (b) there is a good chance that new functionality can, later, be easily mapped on the resulted system.

To illustrate the role of mapping and scheduling in the context of an incremental design process, let us consider the example in the figure, where we have two processors with the same speed connected by a TTP bus. With black we represent the set of already running applications ψ while the current application $\Gamma_{current}$ to be mapped and scheduled is represented in grey and consists of two processes and three messages. To simplify the discussion, for this particular example we consider that the system is not schedulable if the *utilization factor* of any node is greater than one. We say that the processor can be "filled up" with processes until it reaches an utilization factor of one (the square depicting the processor is full). The utilization factor U_i of a process P_i is the ratio between the worst case execution time C_{P_i} of that process and its period T_i: $U_i = C_{P_i} / T_i$. The utilization factor of a node is the sum of the utilization factors of all processes mapped on that node. The processes and messages that are to be mapped on the processors are depicted as blocks. The height of a process block is equal with its utilization factor, while the length of a message block gives the size of the message. White space on a processor represents available utilization, while white space on the bus represents available slack in the schedule table.

Now, let us suppose that in the future another application, Γ_{future}, has to be mapped on the system. Γ_{future} consists of two processes and two messages represented as hashed blocks.

We can observe that the new application can be scheduled only in the third case, case c. If $\Gamma_{current}$ has been implemented as in the case b, we are not able to schedule process P_2 and message m_2 of Γ_{future}. The way our current application is mapped and scheduled will influence the likelihood of successfully mapping additional functionality on the system without being forced to modify the implementation of already running applications.

Thus, we propose a heuristic, together with the corresponding design criteria, which finds the set of old applications which have to be remapped at the same time with the new one such that the disturbance on the running system (expressed as the total cost implied by the modifications) is minimized. Once this set of applications has been determined, mapping and scheduling is performed according to the requirements (a) and (b) stated before. Supporting such a design process is of critical importance for current and future industrial practice, as the time interval between successive generations of a product is continuously decreasing, while the complexity due to increased sophistication of new functionality is growing rapidly.

The approaches have been validated through several experiments and a case study consisting of a vehicle cruise controller.