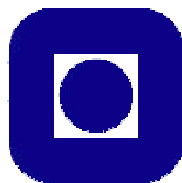


Storage Requirement Estimation and Optimization for Data Intensive Applications

by

Per Gunnar Kjeldsberg

February 22, 2001



Submitted to the
Norwegian university of Science and Technology
in partial fulfillment of the requirements for the degree
Doktor Ingeniør

Summary

Many integrated circuit systems, especially in the multi-media and telecom domains, are inherently data dominant. For this class of applications, data transfer and storage largely determine cost and performance parameters. This is the case for *chip size*, since large memories are usually needed. It is however also the situation for *performance*. Accessing the memories is in more and more cases the main bottleneck and the cache behavior is for a significant part determined by “capacity misses” which are related to size issues. Finally, it is the case for *power consumption*, since the memories and buses consume large quantities of energy and the “effective data size” influences their capacitive loading in case of RAMs and their miss-related activity in case of caches. In both cases, the power consumption per access is also heavily affected. During the system development process, the designer must hence concentrate first on exploring the data transfer and storage to produce a cost-optimized end product. At the system level, no detailed information is available regarding the size of the memories required for storing data in the alternative realizations of an application. To guide the designer and help in selecting the best solution, estimation techniques for the storage requirements are needed, very early in the system design trajectory. The simplest estimates use the maximal size of the intermediate array data as declared in the application code. This is however not representative for the effective size required for their storage during the actual execution since arrays and parts of arrays may not be alive simultaneously. An array element is alive from the moment it is written, or produced, and until it is read for the last time. This last read is said to consume the element. To achieve accurate estimates, the so-called in-place mapping opportunity generated by these non-overlapping lifetimes must be taken into account. For scalars, a relatively simple lifetime analysis suffices, but for arrays, this is extremely complex due to the huge number of signals and the often very complex interdependencies between them.

For our target classes of data dominant applications the high-level description is typically characterized by large multi-dimensional nested loops and arrays. Within the loop nests statements access the arrays using read and write operations. At the beginning of the design process, no information about the execution order of these loops is available, except what is given from the data dependencies between the statements in the code. As the process progresses, the designer makes decisions that gradually fix the ordering, until the full execution ordering is known. This execution ordering determines the order in which array elements are accessed and hence the lifetimes of the array elements. Since these lifetimes in turn influence the in-place

mapping opportunity, the storage requirements of the arrays within the loop nests is largely determined by the execution ordering. To guide the designer it is therefore essential to have storage requirement estimates that can take the available partially fixed execution ordering into account during the exploration of the implementation solution space. Previous work has either not taken execution ordering into account at all, resulting in large overestimates, or required a fully specified ordering. In the last case, a time consuming full exploration of all possible alternative orderings of the unfixed loop dimensions is needed. This is infeasible during the early system design steps where fast feedback is needed to be able to explore the huge solution space.

The storage requirement estimation methodology proposed in this doctoral thesis solves these important design problems. The methodology is divided into four steps. In the first step, a data-flow graph is generated that reflects the data dependencies in the application code. The array accesses and the dependencies between them are described using a polyhedral model. The second step places the polyhedral descriptions of the array accesses and their dependencies in a so-called common iteration space. A worst-case and best-case placement may be performed, both taking available execution ordering into account during the placement. The third step estimates the upper and lower bounds on the storage requirement of individual data dependencies in the code, taking into account the available execution ordering. As the execution ordering is gradually fixed, the upper and lower bounds on the data dependencies converge. This is a very useful and unique property of the methodology. Finally, simultaneously alive data dependencies are detected. Their combined maximal size at any time during execution equals the total storage requirement of the application. An important part of the estimation technique utilizes loop ordering guidance to estimate upper and lower bounds on dependency sizes. These guiding principles and the proof of their validity are together with the general estimation methodology important contributions of this thesis. The guiding principles can be used for high-level synthesis independently of the storage requirement estimation methodology.

The feasibility and usefulness of the methodology are substantiated using several representative application demonstrators. It is for instance shown how the designer is guided into reducing the memory size of the major arrays in the MPEG-4 Motion Estimation Kernel from 262400 to 257 memory locations. Similar results are achieved for a Cavity Detection algorithm. Applying the methodology on an Updating Singular Value Decomposition algorithm, it is also demonstrated how estimation feedback during global loop reorganization can approximately halve the application's storage requirement. Furthermore, a prototype CAD tool has been developed that includes major parts of the storage requirement estimation and optimization methodology. Using manually generated design examples the tool proved the feasibility of the techniques and in particular showed that run times on computers will be short, in the order of seconds even for substantial applications.