Optimization of memory organization and hierarchy for decreased size and power in video and image processing systems *

L.Nachtergaele, F.Catthoor, F.Balasa, F.Franssen, E.De Greef, H.Samsom, H.De Man†
IMEC, Kapeldreef 75, B3001 Leuven, Belgium
†Professor at the Katholieke Universiteit Leuven, Belgium

Abstract

Video and image processing applications deal with large amounts of data which have to be stored and transferred. As the initial system specification describing these data manipulations heavily influences the final memory organization and hierarchy, there is a clear need for exploration support. We believe that the emphasis should lie on fast but accurate estimation and on the high-level steering of the involved system transformations. In this paper, a system exploration environment, called ATOMIUM, is presented that supports these requirements. To illustrate the effectiveness of our approach, two realistic demonstrators are worked out and design results are described.

1 Introduction

For most video and image processing applications there are many ways to realize the multi-dimensional data involved in terms of a specific memory organization and memory partitioning. As reported by system designers this choice is mainly based on "cost" measures such as the number of board-level components, performance, pin count, power consumption and the area of the memory components. Currently, due to design time restrictions, the system designer has to select – on an ad-hoc basis – a single promising path in the huge decision tree from abstract specification to more refined specification. To alleviate this situation, there is a need for fast and early feedback at the algorithm level without going all the way to layout. First the design space must be sufficiently explored at a high level. Only when a limited number of promising candidates have been identified, a more thorough and accurate evaluation of area (A), time (T) and power (P) is required for the final choice (fig.1).

![Figure 1: System exploration environment for data-dominated applications: envisioned situation](image)

Design experience shows that the initial specification, especially in terms of loop organization and array indices, heavily influences the outcome of the background memory estimation or memory mapping tools (e.g. for memory allocation, memory assignment, address generation). Note that this high-level memory management stage is fully complementary to the traditional high-level synthesis step known as "register allocation/assignment" [1, 2, 3, 4, 5, 6] which deals with individual storage places for scalars in registers or register-files, after scheduling.

Manual transforming the specification during the early system-level to explore the cost measures for several alternatives is tedious and error-prone. To remove this design time bottleneck, in addition to memory mapping tools, we have developed automated steering methods for the set of system-level transformations that have the most crucial effect on the system exploration decisions. Such transformations change the loop and index organization in the initial specification models on which the subsequent memory estimation/synthesis tasks work. We also allow a non-procedural, applicative description as initial description. In this way, the system designer has to worry

---

*Research partly funded by ESPRIT Project SPRITE (2260) and Basic Research Action NANA (6632)
less how the original specification is described, while still arriving at a reasonable cost estimate for a particular figure of merit, without having to explore all the possible transformations manually. This drastically reduces the system design time.

2 Previous Work

Up to now, only very little work has been spent on the (automated) design support of these issues. In the high-level synthesis community, all effort has been spent on scalar approaches, as mentioned above, with exception to PHIDEO [7] and recently also MeSA [8]. In PHIDEO, streams are used during the memory allocation but the latter is done after scheduling and is tuned to fixed-rate video applications. In MeSA, the memory allocation cost is based both on a layout model and on the expected performance, but the possibility of signals to share common storage locations when their life times are disjoint, is neglected.

The dominant cost for data intensive applications is memory storage and data transfer. To evaluate this cost, a data dependence analysis is necessary. This has been a topic of research for a long time [9, 10, 11]. The result of the analysis is then used to maximize the fine- or coarse-grain parallelism in loop nests, or to improve data locality [12]. However, it is not yet used to automatically improve the memory mapping. Most effort on memory organization has been spent on cache coherence and cache organization for general-purpose scientific computations (see e.g. [13] for an overview), and not on more custom memory organizations as needed e.g. for video and image processing applications.

3 The ATOMIUUM environment

Our target domain consists of signal and data processing systems which deal with large amounts of data. This happens especially in multi-dimensional signal processing applications like video and image processing, which handle indexed array signals in the context of loops. This class contains many important applications like video coding, document handling, graphics, medical image archival, multi-media terminals, artificial vision.

Architecture experiments have shown that 50-80% of the area and power cost in (application-specific) architectures for real-time multi-dimensional signal processing is due to memory units, i.e. single or multi-port RAMs, pointer-addressed memories, and register files [14, 7]. Hence, we believe that the dominating factor (both for area and power) in the system-level design decisions for our target application domain is provided by the organization of the global communication and data storage, and the related algorithmic transformations. Therefore, we have concentrated ourselves mainly on the effect of system-level decisions on the access to large (background) memories and the transfer of data over long “distances” (long-term main storage). The support of this is achieved in our system-level exploration environment ATOMIUUM (see below). We have also demonstrated that for our target application domain, it is best to optimize the memory/communication related issues before the data-path and control related issues are tackled [14, 15]. Even within the constraints resulting from the memory decisions, it is then still possible to obtain a feasible solution for the data-path, and even a near-optimal one if the appropriate transformations are applied [15].

Our activities have been mostly aimed at application-specific architecture styles, but recently also predefined processors (e.g. DSP cores) are envisioned [16]. The cost functions which we currently incorporate for the storage/communication resources are both area and power oriented [17]. Due to the real-time nature of the targeted applications, the throughput is normally a constraint.

The input of the ATOMIUUM environment is a specification of the design written in a Data Flow oriented Language (called DFL) [18]. The output is a netlist of memories and address generators, combined with a transformed specification which is the input for the architecture (high-level) synthesis when custom realizations are envisioned, or for the software compilation stage in the case of predefined processors. The specific subtasks addressed within our toolbox are now discussed (see also Fig. 2) 1:

1. Transformation engine (SynGuide): allows both interactive and automated (script based) steering of language-coupled source code transformations. It includes a syntax-based check which captures most simple specification errors, and a user-

---

1 All of these tools operate on models which allow run-time complexities which are nearly independent on system parameters like the size of the loop iterators, as opposed to the scalar-based methods published in conventional high-level synthesis literature.
friendly graphical interface. The transformations are applied by identifying a piece of code and by entering the appropriate parameters for a selected transformation. The main emphasis lies on loop manipulations including both affine (loop interchange, reversal and skewing) and non-affine (e.g. loop splitting and merging) cases [19].

2. Formal verification of the loop transformations has been solved in a very general context including procedural (e.g. WHILE) loops, general affine indices (including the so-called “lattices” [20]), and data-dependent signal indices [21]. A prototype verification tool is available.

3. Loop control flow definition: extended transformation steering tool MASAI aiming at total background memory cost reduction with emphasis on transfers and size. An abstract measure for the number of transfers is used as an estimate of the power cost and a measure for the number of locations as estimate for the final area cost [22]. This tool is based on an earlier prototype [23]. The transformed behavioral description can already be used for more efficient simulation or it can be further optimized by the next two steps of this script.

4. Memory allocation/port binding: given the cycle budget the goal of this task is to allocate memory units and ports including their types from a memory library. It also includes a high-level but accurate estimation of the memory cost, under the assumption of an applicative (non-procedural) input specification. In a second stage, the multi-dimensional signals are assigned to these background memory units and ports. Again, this results in an updated flowgraph specification. A prototype tool HIMALAIA has been developed [24].

5. In-place mapping investigation of experimental methods for deciding on in-place storage of multi-dimensional signals.

In addition, several tasks are devoted to address expression optimization and realization of an efficient (multiplexed) architecture [25]. These tasks will however not be addressed in the paper. Here, the emphasis will be on the use of this methodology and its major subtasks in memory dominated designs and not on the actual CAD techniques applied to achieve this (and which have been partly published elsewhere).

4 Demonstrating applications

The results of our approach and the implemented environment will now be demonstrated on two realistic test-vehicles. The emphasis will lie on the memory/port organization and the partitioning required to realize these real-time applications for the specified throughput.

First, we will show the effect of our memory transformation tools on a medical image processing application.
We have coupled our tools to a simulation engine for the system-level applicative Silage/DFL language, with excellent results for the virtual memory usage and run-times on mid-size workstations. An example of this effect is shown in figure 3. For relatively small parameter values \( Npr \) (the number of projections to be used for reconstruction), namely \( Npr=6 \), the simulation of the original description (BEFORE) becomes infeasible due to memory shortage even with a workstation swap space of 80 Megabyte. Also for lower \( Npr \) values, the CPU times are becoming rapidly dominated by the memory swapping. After automatic optimization of the description (AFTER), the simulation run-time (on a HP715/50) and the memory requirements are drastically reduced as shown in the figure, even for large \( Npr \) values. The number of scalar signals which would have to be analyzed when the loops would be expanded as in conventional simulation paradigms is then up to several billion words. Figure 3 clearly shows a large improvement in parameter range for which simulation becomes feasible.

Figure 4: Estimated power cost for back-projector with \( Npr = 12 \). Notice that both system-level loop transformations and memory organization decisions are needed to support these steps. The loop transformations have been automatically obtained with our MASAI tool.

For this same application with realistic parameters for modern scanners (image size \( N = 512 \times 512 \); detector resolution \( Nd = 4096 \)) also the obtained results on off-chip memory area cost and power budget will now be discussed. We have obtained result for both the low accuracy (\( Npr = 12 \)) (summarized in figure 4), and the high accuracy (\( Npr = 1200 \)) (see figure 5) case [17].

Figure 5: Estimated power cost for high-performance back-projector with \( Npr = 1200 \). Notice that a different system transformation and memory organization are needed to arrive at a good result. This clearly illustrates that a parameterized specification with fixed realization directives stored in a system module library would lead to unacceptable results in terms of memory power and size.

This example shows that memory considerations do depend on context and parameters. In the high resolution case, the original DFL specification would require a single-port projection data buffer of no less than 77Mbit. This would require an enormous power budget of about 20W, which is clearly not acceptable. A better solution, which is also automatically found by our MASAI tool when starting from the initial specification, is to choose the \( \theta \) loop as outer. In that case, the main memory requirement is due to the storage of the intermediate image information \( I \) which requires two SRAMs for the partitioned even and odd data. The on-chip projection detector data buffer requires about 64Kbit, which is very acceptable and contributes only in a minor way to the system power budget. These alternatives are illustrated in
In this analysis, we have not yet incorporated the effect of the number of bits which are really active (the worst-case power budget has been assumed for the RAM's which is the only figure given on the data sheets) or the interconnect capacity between the memory chips and the ASIC for the data-path operations (which is very difficult to characterize). Taking these considerations into account would change the absolute value of the power contributions but the conclusions will remain the same, given the large relative differences for the alternative memory organizations.

In summary, we believe that the effects of both transformations and memory organization decisions on the area (reduction from 1 "virtual" bank of RAMs which would require a storage of 77Mbit to 2 banks of switched RAMs, each with about 2.25 Mbit) and the system power budget (reduction from 20W to 1.56W) are convincing for the effect of the proposed approach.

In addition, we will show the results for a memory-dominated submodule of the MPEG2 video decoder/encoder, namely a motion detection kernel. It allows to estimate the motion vector of small blocks of successive image frames. The version we consider here is commonly referred to as the "full-search full-pixel" implementation [26]. The parameters we used for our implementation are the following: image of $720 \times 576$ pixels, block size of $8 \times 8$ pixels, neighborhood size of $23 \times 23$. The pixels are 8-bit gray scale values.

The result of the memory allocation (automatically obtained with our HIMALAI tool) for several cycle budgets in the inner loop body is provided in figure 6. Note that the frame memory which is needed to store the delayed frames is not included. This large frame memory needs to be stored off-chip. What is included here are the smaller background (and partly also foreground) memories for storing the intermediate data on-chip, relatively close to the data-paths. The area figures have been obtained for an accurate RAM layout model, instantiated for a fast 1- or 2-port embedded SRAM generator in a 1.2µ CMOS technology.

It can be seen that the memory organization is heavily influenced by the allowed cycle count. For instance for 2 cycles 3 RAMs are needed, two of which have 1 R/W port and the third one has a R port and a R/W port. In contrast, if we allow 6 clock cycles - with a shorter period! - only 2 RAMs are needed, both with 1 R/W port. However, the area which is needed in the end is not really smaller. This is true because in the 2-cycle solution, the data can be efficiently distributed over several memories by our tools. These type of area-cycle trade-offs are very useful for the system architect to decide on the type of RAMs which are chosen: either slower but cheaper ones, or faster but more expensive ones. For this application, it is clearly more effective to choose 3 slower inexpensive on-chip memory modules, with other parameters for the layout and area estimation model. The latter require less area and exhibit a higher yield than the 2 memories in the 6-cycle solution using the relatively fast default SRAM generator.

Suppose the designer has manually selected a solution (e.g. with 500mm²) during memory allocation. This can easily lead to the wrong conclusion that the 6-cycle solution would be better, whereas this requires much faster and more expensive RAMs to achieve the same real-time throughput constraint. This clearly shows the usefulness of accurate estimators and near-optimal exploration tools to support these decisions.

Figure 6: Results for automated memory allocation for several cycle budgets in inner loop of a video motion detection kernel. The cross-hatched boxes show the area required for the optimal memory organization found, including the number of RAMs needed and their Read or Read/Write port requirements (arrows on top of boxes). The dashed boxes in the background represent the number of feasible solutions which have been found by the tool and their range of area costs. For instance, for the 2 cycle case, 46 feasible solutions have been identified in the large search space and these require an area cost ranging from 850 mm² to about 250 mm² (optimal case found).

5 Summary and conclusions

In this paper the system design exploration environment ATOMIUM has been presented. This tool box enables the system designer to quickly evaluate several alternatives obtained by algorithm transformations and it provides early feedback. In addition,
for the most promising algorithm candidates a good memory organization can then be explored. At the end, also the necessary addressing hardware is automatically generated. Our approach is targeted to data-dominated systems like video and image processing applications. This approach was exemplified for two realistic applications. For the first example the memory need is reduced from 77Mbit down to 4.5Mbit. The power consumption could be reduced from 20W down to 1.56W. For a second example, the result of memory allocation for several cycle budgets shows the potential for fast memory organization exploration.

References


