Available online at www.sciencedirect.com

SCIENCE DIRECT®

The Journal of Systems and Software 79 (2006) 231–245

The Journal of Systems and Software

www.elsevier.com/locate/jss

Polyhedral space generation and memory estimation from interface and memory models of real-time video systems

Benny Thörnberg [a,*], Qubo Hu [b], Martin Palkovic [c,1], Mattias O'Nils [a], Per Gunnar Kjeldsberg [b]

[a] Mid-Sweden University, Holmgatan 10, 851 70 Sundsvall, Sweden
[b] Norwegian University of Science and Technology, 7491 Trondheim, Norway
[c] IMEC, B-3001 Leuven, Belgium

Received 3 June 2004; received in revised form 18 April 2005; accepted 22 April 2005
Available online 21 June 2005

Abstract

We present a tool and a methodology for estimating the memory storage requirement for synchronous real-time video processing systems. Typically, a designer will use the feedback information from this estimation to select the most optimal execution order for software processors or space to time mapping for hardware. We propose to start from a conceptual interface and memory model that captures memory usage and data transfers. This high-level modeling is provided as an extension library of SystemC called IMEM. A common polyhedral iteration space is generated from the model, where polytopes are placed using a new placement algorithm based on simple heuristics. This algorithm will ensure maximum freedom of selecting executing order as all negative dependencies are removed to the length of zero. A demonstration is given regarding how the polytopes and dependency vectors can then be used as input to a memory storage estimation tool called STOREQ.
© 2005 Elsevier Inc. All rights reserved.

Keywords: Memory storage estimation; Polyhedral; Polytope placement; Modeling; Real-time; Video; SystemC

1. Introduction

Real-time video processing systems (RTVPS) use a very large amount of data storage and transfers. This will become a major design bottleneck for embedded systems, since memories and bus transfers will consume a large amount of power (Wuytack et al., 1998). Initial work in the area of high-level synthesis of RTVPS (Thörnberg and O'Nils, 2003) has indicated the need for memory storage estimation in order to efficiently map a model onto an architecture. STOREQ (Kjeldsberg et al., 2004) is a tool that can be used for the estimation of memory STOrage REQuirements for data intensive digital signal processing systems, namely DSP systems. STOREQ memory estimation includes selecting the most optimal processing order or space to time mapping with respect to storage requirements, which can guide the high-level synthesis to a better result. This estimation is done using a polytope model as the input (Kjeldsberg et al., 2004).

A polytope in this model is a multi-dimensional body of iteration nodes when the statements with data write or data read are executed. A statement that uses data produced by an earlier statement is data dependent on the earlier statement. These dependencies are

[*] Corresponding author. Tel.: +46 60 148917; fax: +46 60 148456.
E-mail addresses: benny.thornberg@mh.se (B. Thörnberg), qubo.hu@iet.ntnu.no (Q. Hu), martin.palkovic@imec.be (M. Palkovic), mattias.onils@mh.se (M. O'Nils), per.gunnar.kjeldsberg@iet.ntnu.no (P.G. Kjeldsberg).
[1] Tel.: +32 1628 1679.

0164-1212/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2005.04.034

A

```
Int Vr = 2;
Int Vc = 2;
Int Ce = 6;
Int Re = 3;

for(r=0; r<Re+Vr, r++)
 for(c=0; c<Ce+Vc ,c++)
 {
    if(r<Re && c<Ce)
/*S1*/  A[r,c] = input();
    if(r>=Vr  && c>=Vc)
/*S2*/  B[r,c] = f(A[r-Vr, c-Vc]);
 }
```

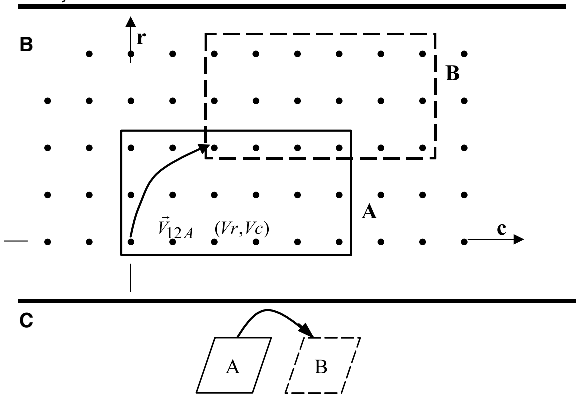

Fig. 1. Example of a common polyhedral iteration space.

represented as dependency vectors between the iteration nodes, which can be calculated from the index expressions. Fig. 1A shows an example code with a 2-dimensional loop nest. The set of iteration nodes when the statement *S1* with data array *A* write is executed is represented as polytope *A* in Fig. 1B. Polytope *B* consists of the set of iteration nodes when the statement *S2* with data array *B* write and data array *A* read is executed. We simply say there are dependencies between these two polytopes. One of them is depicted as vector $\vec{V}_{12A}$ in Fig. 1B. The dependency between these two polytopes is also depicted in Fig. 1C as a polytope dependency graph. In this directed graph, the vertices correspond to the polytopes and the single edge to the data flow dependency going from polytope *A* to *B*.

The STOREQ tool has been developed to be a part of the Data Transfer and Storage Exploration methodology from IMEC (Catthoor et al., 2002). When used in that context, a platform independent transformation step identifies the corresponding polytopes and data dependency vectors from a given application code. This step also include polytope placement into a common polyhedral iteration space (Danckaert et al., 2000). The placement algorithm used becomes complicated since both regularity and locality in data accesses are considered. Verdoolaege et al. (2003) only looks at locality and can achieve a reasonable result even for complex multimedia applications. There are also some other works in the area of polytope placement, which are referred to in Section 2. However, the complexity of all these algorithms, still necessitate a complex tool for the RTVPS we are looking at. An even simpler placement algorithm would be sufficient enough and is thus developed and implemented in this work.

This paper presents the integration of STOREQ into the IMEM development workflow (Thörnberg et al., 2002). IMEM is a library extension of SystemC (Panda, 2001) for modeling of neighborhood-oriented multi-rate synchronous real-time video processing systems. In IMEM, no loops are initially specified and thus details related to the final implementation are excluded. In addition, the specification of coarse-grained data flow dependencies is separated from the functional specification of image processing operators, which makes code pruning unnecessary. Code pruning is a source code pre-processing step at which array accesses are separated from the computational statements. Code pruning is applied on C-code or Data Flow Language in the DTSE-methodology (Catthoor et al., 1998). The integration of STOREQ requires the development of an algorithm and a tool to support the transformation of an IMEM model into a polytope model. The fact that image frame sizes and data dependencies are captured as C++ objects in IMEM, simplifies the extraction of polytopes and dependency vectors. Neighborhood-oriented RTVPS are perfectly regular in their data accesses, simplifying the polytope placement algorithm that we have developed.

This work has resulted in a new polytope placement algorithm suitable for IMEM models and implemented in the tool IMEM Projector. IMEM Projector can import an IMEM model and map this model onto a TriMedia DSP prototyping platform (Thörnberg and O'Nils, 2003). To our knowledge, no similar approach exists, where application specific, extended SystemC-based modeling is combined with the geometrical polyhedral modeling into a memory storage estimation methodology. This application specific methodology takes advantage of simplifications such as the regularity in RTVPS, the separation of data dependencies from computation and the powerful system modeling provided by SystemC. This is also the reason why we are doing polytope placement before integrating STOREQ, as opposite to other approaches. This is discussed in Section 8.3.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 describes modeling of a real-time video system in IMEM. Section 4 describes our tool IMEM Projector and motivates the existence of memory estimation in a system development workflow. Section 5 explains the polyhedral iteration space and how polytopes and dependencies can be generated from an IMEM model. The polytope placement algorithm is explained in Section 6. A real-life video system is presented in Section 7. Section 8 further discusses this work