Partitioning for DSP Software Synthesis

Ming-Yung Ko and Shuvra S. Bhattacharyya

Electrical and Computer Engineering Department, and
Institute for Advanced Computer Studies
University of Maryland, College Park, Maryland 20742, USA

Abstract. Many modern DSP processors have the ability to access multiple memory banks in parallel. Efficient compiler techniques are needed to maximize such parallel memory operations to enhance performance. On the other hand, stringent memory capacity is also an important requirement to meet, and this complicates our ability to lay out data for parallel accesses. We examine these problems, data partitioning and minimization, jointly in the context of software synthesis from dataflow representations of DSP algorithms. Moreover, we exploit specific characteristics in such dataflow representations to streamline the data partitioning process. Based on these observations on practical dataflow-based DSP benchmarks, we develop simple, efficient partitioning algorithms that come very close to optimal solutions. Our experimental results show 19.4% average improvement over traditional coloring strategies with much higher efficiency than ILP-based optimal partitioning computation. This is especially useful during design space exploration, when many candidate synthesis solutions are being evaluated iteratively.

1 Introduction

Limited memory space is an important issue in design space exploration for embedded software. An efficient strategy is necessary to fully utilize stringent storage resources. In modern DSP processors, the memory minimization problem must often be considered in conjunction with the availability of parallel memory banks, and the need to place certain groups (usually pairs) of storage blocks (program variables or arrays) into distinct banks. This paper develops techniques to perform joint data partitioning and minimization in the context of software synthesis from Synchronous Dataflow (SDF) specifications of DSP applications [10]. SDF is a high-level, domain specific programming model for DSP that is widely used in commercial DSP design tools (e.g., see [4][5]). We report on insights on program structure obtained from analysis of numerous practical SDF benchmark applications, and apply these insights to develop an efficient data partitioning algorithm that frequently achieves optimum results.

The assignment techniques that we develop consider variable-sized storage blocks as well as placement constraints for simultaneous bank accesses across pairs...
of blocks. These constraints derive from the feature of simultaneous multiple memory bank accesses provided in many modern DSP processors, such as the Motorola DSP56000, NEC µPD77016, and Analog Devices ADSP2100. These models all have dual, homogenous parallel memory banks. Memory allocation techniques that consider this architectural characteristic can employ more parallelism and therefore speed up execution. The issue is one of performing strategic data partitioning across the parallel memory banks to map simultaneously-accessible storage blocks into distinct memory banks. Such data partitioning has been researched for scalar variables and register allocation [7][8][18]. However, the impact of array size is not investigated in those papers. Furthermore, data partitioning has not been explored in conjunction with SDF-based software synthesis. The main contribution of this paper is in the development of novel data partitioning techniques for heterogeneous-sized storage blocks in the synthesis of software from SDF representations.

In this paper, we assume that the potential parallelism in data accesses is specified by a high level language, e.g., C. Programmers of the SDF actor (dataflow graph vertex) library provide possible and necessary parallel accesses in the form of language directives or pseudocode. Then the optimum bank assignment is left to software synthesis. Because of the early specifications, users can not foresee the parallelism that will be created by compiler optimization techniques, like code compaction and selection. It is neither our intention to explore such low level parallelism. From the benchmarks collected (in the form of undirected graphs), a certain structural pattern is found. The observations help in the analysis on practical applications and motivates a specialized, simple, and fast heuristic algorithm.

To describe DSP applications, dataflow models are quite often used. An application is divided into modules with data passing between modules. Modules receive input data and output results after processing. Data for module communication flows through and is stored in buffers. In dataflow semantics, buffers are allocated for every flow. In Section 4, it is demonstrated that the nature of buffers helps in optimizing parallel memory operations. SDF [5] for multirate applications is especially suitable for buffer analysis and is referenced in our discussion.

The paper is organized as follows. A brief survey of related work is in section 2. Detailed and formal descriptions of the problem are given in section 3. Some interesting observations on SDF benchmarks are presented in section 4. A specialized as well as a general case algorithm are provided in section 5. In section 6 are the experimental results and our conclusion.

2 Related Work

Due to performance concerns, embedded systems often provide heterogeneous data paths. These systems are generally composed of specialized registers, multiple memory modules, and address generators. The heterogeneity opens new research problems in compiler optimization.

One such problem is memory bank assignment. One early article of relevance on this topic is [15]. This work presents a naive alternating assignment approach. In [17], interference graphs are derived by analyzing possible dual memory accesses in