Abstract Machines for RASCs and Signal/Image Processing

L. Mulin[1,2], J. Raynolds[3], I. Grout[4], J. Ryan[4] and Q. Li[1]

[1] Department of Computer Science, College of Computing and Information
[2]Department of Physics, College of Arts and Sciences,
[3]College of Nanoscale Science and Engineering
University at Albany, SUNY, Albany, New York State, USA
[4]Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland

Abstract

In this paper, we present a novel approach to optimizing the implementation of signal processing kernels on Reconfigurable Application Specific Architectures (RASCs). RASCs expose so many lower level hardware details to the software that a formalism that abstracts and allows optimization across the hardware/software boundary is required. There exists no such monolithic abstraction to date. To address this, we are developing the **Conformal Computing**[1] paradigm as a potentially powerful methodology for developing optimizations on such kernels. A central feature of the **Conformal Computing** paradigm is the use of a single, powerful, matrix/array algebra *(A Mathematics of Arrays* (MoA), and corresponding index calculus (ψ-*calculus*)[1]) to describe formally the algorithm, array decompositions to processor/memory hierarchies, data communications, etc.

Our approach is to capture both the problem and machine architecture in one monolithic, parameterized, multi-dimensional representation. This multi-dimensional representation reflects the mapping of the algorithm to the hardware by lifting the dimensionality of the algorithm to include machine components, e.g., memory, processor, etc. We include in our representation a model of implementation cost along each machine dimension. Different mappings of problem to machine are expressed as re-shapings of the combined representation with associated costs. We can then use this representation to find not only the best mapping of problem to fixed hardware, but also the best machine configuration given problem size by minimizing the cost. In this way, we provide a uniform framework for reasoning about array-based computations by algebraically connecting the software and hardware [6] so that we can optimize across the boundary between them.

We will present the motivation for our approach, the basics behind our representation, and the theoretical framework for reasoning about performance. An integral part of our presentation is to illustrate how a *normal form* can provide ***intentional information*** to tools that instantiate the syntax and semantics needed at a particular memory level. The *normal form* provides details of the iteration space and data flow. We build upon our processor/cache designs for the FFT [3,5] and illustrate how the iteration space for cache can be ***morphed*** to the iteration space of the FPGA in the RASC. We also discuss the difficulties with communicating such intentional information to the machine and our progress made to date [2]. We will show that our analysis deterministically identifies the best mapping via our cost model.

[1] The name **Conformal Computing©** is protected. Copyright 2003, The Research Foundation of State University of New York, University at Albany.

Introduction

RASCs represent a revolutionary class of machines that provide the capability to optimize across the hardware/software boundary via a computing architecture that re-structures itself at the micro-architectural level to meet changing application requirements with optimal, or near optimal, efficiencies. Different mission requirements can be supported by changing the allocation of physical resources to abstract architectures in response to different mixes of processing types, i.e., streaming, threaded, etc. Also, reconfiguring the underlying physical hardware becomes necessary when certain algorithms, or combinations of algorithms, dominate the scientific computation, e.g. FFT. In a RASC the overall system must be analyzed to determine what portions run on the host and what portions run on the FPGA. One also can imagine a network or grid of machines such that each had multiple nodes with multiple processors with multiple levels of memory AND multiple FPGAs.

In Conformal Computing, reconfiguring the hardware to the problem and problem size relies on an abstract architecture formulated by the same algebra used for the problem (MoA). The ability to *reconfigure* an architecture in response to changing processing, application and mission requirements will reduce cost to the DoD for sensor processing systems by reducing the hardware complexity and cost, as well as development complexity and cost. Additionally, since the architecture is abstract, it is portable to many different COTS hardware realizations and processing platforms: RASCs, GRIDs, NUMAs, etc.

Realization of an abstract, portable computing architecture requires an efficient, flexible representation that efficiently maps to hardware constructs. Such a representation can be used for design space optimization, program implementation and compilation, or run-time hardware allocation and/or resource management. The representation must include the capability and state of the hardware, including performance and cost for the operations resulting from realization of a particular algorithm, i.e., memory access, arithmetic operations, inter-processor communication, and I/O. The representation must retain higher-level information pertinent to the algorithm so that changes in the abstract machine architecture can be easily propagated back to the algorithm implementation. ***Reconfiguring*** the hardware will involve transformations on this intermediate representation; any transformation performed in the course of reconfiguring must ***preserve correctness in a theoretically provable manner***.

We are currently working to develop hybrid codes that run certain operations in hardware and the balance in software on machines such as the SGI MOATB and the Cray XD1. There are numerous obstacles to overcome and there is very little documented experience in the literature on how to program such machines. A number of important issues such as memory management must be dealt with directly by the user. We report on our efforts to simplify the interface to and optimize the mapping of algorithms to RASCs. A prototyping system (SULU)[2] has been developed for enabling the prototyping of RASC algorithms without the need to access the RASC system itself. Thus algorithm design, development and debug can be undertaken and once complete, can be uploaded and run on the RASC system. This prototyping system is based on a PC (Microsoft® Windows® operating system with C-coding) and Xilinx® Inc. Spartan™-3 FPGA development board. This combined with optimisations from Conformal Computing promise to produce verified designs for both hardware and software realizations.

Approach

Our approach in defining a machine abstraction that meets the requirements outlined above is to algebraically describe the algorithm and architecture using MoA. MoA is an algebra that provides the ability to describe array expressions in terms of the shapes of their arguments and the capability for symbolic verification of n-dimensional array expressions. Many array operations have common ***access patterns*** that can be ***known a-priori***. The code to generate a given access pattern is generated by composing indices based on array shapes. The Psi Calculus is a calculus of array indexing that reduces an MoA array expression to a denotational normal form (DNF) via formal *linear/or multi-linear* transformations on the MoA expression. The DNF completely expresses the composition of indexing functions and is the semantic normal form, i.e., the form in terms of Cartesian coordinates. From the DNF, an operational normal form (ONF) is derived that expresses the index functions as "start, stop, and stride" accesses to memories. For a given architecture, the ONF can be theoretically proven to have minimal memory accesses due to the materialization of unnecessary intermediate values and temporary variables. Since the Psi Calculus has the Church-Rosser property [3], *array expressions can be proven equivalent by showing that their normal forms are the same*.

The machine architecture is incorporated into the algorithm description by lifting the dimensionality of the arguments by the dimensionality of the machine architecture. An example is shown in Figure 1. In this example, the array to be operated on is a vector (dimensionality = 1), and the abstract machine is composed of processors with cache memory (dimensionality = 2). Since the resulting, monolithic structure is a 3D array, we can use MoA and Psi Calculus to represent algebraically an algorithm's implementation (decomposition and mapping to processors and memory). Given such an implementation, we can prove the equivalence of two algorithm implementations and/or machine realizations. Reconfiguring involves a change to the abstract machine, which, in our model, changes the number and sizes of dimensions in the original representation of the problem. Such a restriction affects the DNF and consequently the ONF of expressions. Using Psi reduction rules (linear and multi-linear transformations), changes to the DNF and ONF can be guaranteed to preserve the correctness of the algorithm. Since Psi Reduction is guaranteed to produce minimal temporaries, the efficiency of the implementation is also preserved. From the ONF, the code to implement the algorithm can be methodically derived. Since array representations and transformation rules are not limited by size and dimensionality, such an approach is inherently scalable. Costs along each dimension are included in our model.

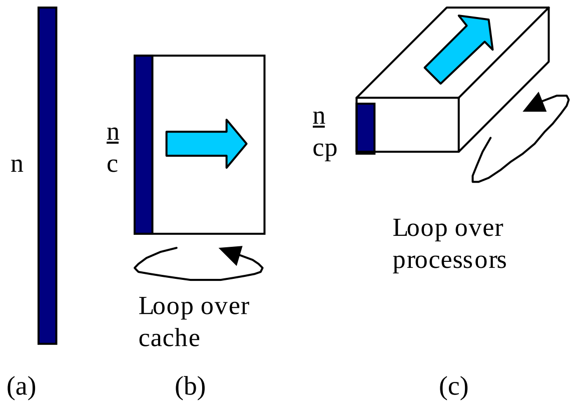

Figure 1: (a) the original vector; (b) inclusion of cache; (c) inclusion of processors.

Experiments

The morphing concepts will be demonstrated by expressing one or two representative signal processing kernels as well as the abstract machine using MoA as described above. We will demonstrate the ability of the abstract machine to express machine morphs using the MoA representation. We will show how computation and communication costs can be deterministically derived from the abstract machine model, and verify the model by comparing estimates with measured performance. Finally, we will demonstrate the mapping of the abstract machine to actual hardware via kernel implementations on representative RASC hardware. (SGI MOATB) To show the viability of our approach in modeling performance and cost for array expressions, we use a simple signal-processing kernel such as the 1-d FFT. In the talk, we will go through pictorially the MoA representation for the FFT, show the DNF and ONF, show the representation with processors and memory added, with the FPGA, and then show the costs.

Conclusions and Future Research

We have demonstrated a uniform approach to reasoning about signal/image processing using an algebraic framework and reconfigurable architectures. Our goal is to produce fast, portable, scalable, reconfigurable, mathematically correct code without requiring huge investments in software development. We have described our approach to machine abstraction, explained the theoretical framework for proving correctness, and given an example of our cost model using a COTS RASC. The representation we have designed encapsulates the machine via parameters. By varying these parameters and searching the design space, we can determine both the best decomposition and mapping to hardware and/or the best hardware environment for problem size by minimizing the costs from our machine model.

Future research is to develop the algorithms, methods and trade-off knowledge base to search the design space efficiently. The knowledge base would contain, power and area efficiency, utilization, latency and/or throughput, etc. The transformations are *linear* or *multi-linear* and thus mechanizable, although we have demonstrated them manually. Experiments to mechanize MoA and Psi Reduction were validated at MIT/LL [4]. Extensions to program block optimizations were also validated. These experiments could lead to automatic code generation from the ONF. Future research also involves identifying families of algorithm data flows to extend these types of optimizations across other disciplines, e.g., simulation science, scientific computation. We also plan to investigate other RASC architectures, more complex kernels and abstract machine models.

This work is the result of an interdisciplinary and international collaborative effort. Our team includes the coordinated efforts of computer scientists, physicists and electrical engineers.

References

1. L.R. Mullin, *A Mathematics of Arrays*, PhD Thesis, Department of Computer Science, Syracuse University, December, 1988.

2. I. Grout. L. Mullin, J. Ryan, and Q. Li, “A Desktop RASC Prototyping System”, under review ESS Conference, October, 2006.

3. L. Mullin and J. Raynolds, "Optimizing the Fast Fourier Transform over memory hierarchies for embedded digital systems: a fully in-cache algorithm", In *Proceedings of the High Performance Embedded Computing (HPEC) Workshop, MIT Lincoln Lab, September 2004, (2004).* See also (preprints): http://trr.albany.edu/documents/TR00004 and, http://trr.albany.edu/documents/TR00005

4. E. Rutledge, et. al., “Monolithic Compiler Experiments using C++ Expression Templates”, HPEC, 2002.

5. J. Raynolds and L. Mullin, "Applications of Conformal Computing techniques to problems in Computational Physics: the Fast Fourier Transform", *Comp. Phys. Comm.* **170**, 1 (2005).

6. L. Mullin, "A uniform way of reasoning about array-based computation in radar: algebraically connecting the hardware/software boundary", *Digital Signal Processing*, **15**, 466 (2005).