Malleable Architectures for Adaptive Computing
MIT9904-04

Proposal for 1999-2000 Funding

Arvind, Larry Rudolph and Srinivas Devadas

Project Overview

Existing general-purpose microprocessors and digital signal processors do not meet all the needs of intelligent personal devices, multimedia and advanced communication applications. Designing special-purpose chips for each application is typically not feasible due to expensive fabrication costs and long turnaround time. A large reconfigurable device with appropriate tools and infrastructure may be the solution. We propose a framework for the synthesis of *malleable* processors based on million-gate Field Programmable Gate Arrays (FPGAs) such as the Xilinx XC4000XV series. For each application or set of applications, an instruction set and micro-architecture are chosen to significantly improve either the performance or the power dissipation. The architecture can be synthesized for FPGA's or other implementation technologies by an architecture exploration system centered around the TRAC compiler, which takes high-level behavioral descriptions and generates synthesizable Register Transfer Language (RTL) descriptions in Verilog.

We believe that the proposed framework will stimulate architecture innovation, and result in the development of creative, new architectures for multimedia applications. The development of such a framework requires the solution of difficult problems in the areas of architecture specification, automated hardware synthesis, and software compilation.

The proposed work will also provide an opportunity to collaborate with Hiroshi Sawada, Dr. K. Shimohara and other members of the adaptive architecture group at NTT. The FPGA CAD software being developed at NTT can be evaluated using the RTL (Verilog) models of application-specific processors generated by our architecture exploration system.

1 Malleable Architectures

Multimedia applications can benefit from an Instruction Set Architecture (ISA) tailored to its specific needs, such as the word size, parallel bit vector operations, and other complex instructions. An application can also benefit from a specific micro-architecture for the chosen ISA. But perhaps the most critical shortcoming of existing architectures is in the memory organization. On-chip caches now consume 30-60% of the silicon area in general-purpose processors, and this percentage is increasing. Tailoring memory organization for specific applications, or enabling dynamic or adaptive memory management while running applications is thus becoming very important. This is especially true for multimedia applications with streaming data access patterns for which conventional set-associative caches with LRU do not work well.

We propose novel ways of controlling caches to dramatically improve their utility and power consumption. Such ***malleable*** caches are suitable for a wide range of data-intensive applications. Small performance-critical code sequences of stream-processing and real-time applications can achieve several orders of magnitude improvement with malleable caches. Applications with low information entropy can make use of the cache resources for data compaction or memoized functions to dramatically improve execution time. In addition, malleable caches permit many architectural optimizations, such as power management and adaptive prefetching.

Malleable caches are made possible through several innovations. ***Column caching*** provides control as to where items are stored in a cache and ***curious caching*** allows the cache to fetch data that has been placed on the bus. A ***flexible indexing*** technique enables parts of the cache to be used as a general-purpose associative memory. A malleable cache can act as a transducer of information flowing into and out of the processor. Finally, a ***cache pressure gauge*** allows better overall utilization of cache resources in a dynamic, on-line fashion. Each of the above mechanisms enable adaptive cache management by providing the user, compiler, or operating system finer control over cache resources, which in turn can result in dramatic speedups or stream-based and real-time applications.

In order to evaluate different cache organizations, we propose the development of specification mechanisms within ADL, our architecture description language (see below), which will allow us to implement a general-purpose cache simulator, which can evaluate malleable cache performance under real workloads. This framework will enable the development of processor architectures and efficient, adaptive memory management algorithms tailored to the needs of data-intensive multimedia applications.

Architecture Exploration Framework

We propose the development of a framework that enables architectural experimentation in programmable processor design. Architecture exploration requires the capability of evaluating and comparing multiple micro-architectures, and this in turn requires a mechanism for detailed performance feedback from a high-level processor specification. There are two major dimensions to the specification of programmable processors. The first dimension is the programmer's model of processor behavior, termed the Instruction Set Architecture (ISA) of the processor. The second dimension is the micro-architecture of the processor, which provides details on how the instruction set is implemented in hardware, e.g., pipelined or unpipelined, in-order or out-of-order execution. We are developing an Architecture Description Language (ADL) that describes *both* these dimensions, and which serves as the centerpiece of our architecture exploration framework, as illustrated in Figure 1.

An ADL description is used to drive two design trajectories. The instruction set architecture (ISA) component of the ADL description uses the ISDL language developed in earlier research, and drives the generation of software tools essential for the use and evaluation of processor architectures including a retargetable compiler and simulator. The retargetable compiler will produce optimized code that corresponds to machine instructions described in the ADL description. The simulator will serve as a functional, cycle-accurate, instruction-set simulator, and along with the compiler, it will be used to obtain accurate clock cycle counts corresponding to application code on the processor.

The micro-architecture component of the ADL description drives a hardware synthesis tool, which generates a hardware description language (e.g., Verilog) model of the processor. In order to describe the intricacies of modern processors such as superscalar functional units, ADL will use nondeterministic rules based on Term Rewriting Systems (TRS). However, ADL descriptions will be restricted to always be synthesizable, and this process will be completely mechanized. Logical synthesis from the generated hardware model will produce a gate-level implementation that can be analyzed to determine power consumption, area and maximum clock frequency. The two trajectories described above synergistically provide a mechanism for accurate performance evaluation of application code running on a given processor design. Our goal is to build a framework such that the evaluation of candidate architectures and comparisons across candidate architectures simply require that the architect write the corresponding ADL description(s). This will allow the architect to focus on the creative aspects of architectural design and experimentation; we believe this increased focus will result in the creation of novel general-purpose architectures, as well as novel application-specific processors.

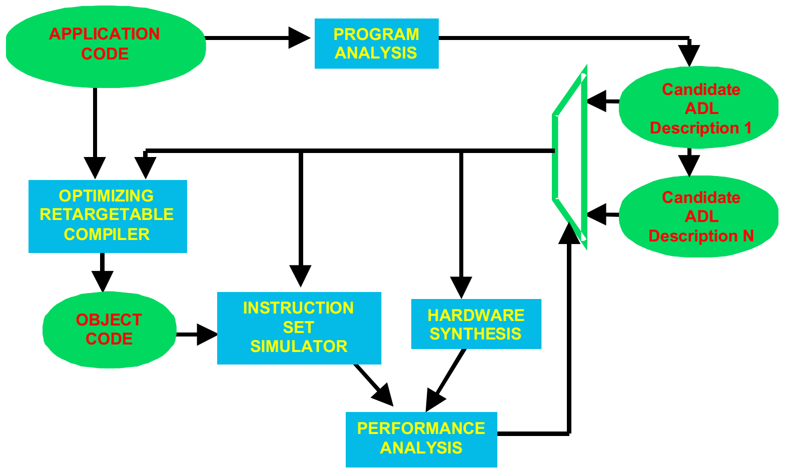
APPLICATION CODE
PROGRAM ANALYSIS
Candidate ADL Description 1
Candidate ADL Description N
OPTIMIZING RETARGETABLE COMPILER
OBJECT CODE
INSTRUCTION SET SIMULATOR
HARDWARE SYNTHESIS
PERFORMANCE ANALYSIS

Figure 1: Proposed System for Architecture Exploration

Status, Schedule and System Demonstration

July 1, 1999: A version of ADL, the architecture description language has been developed, and tools for generating assemblers, disassemblers and instruction-level simulators have been written.

December 31 1999: A prototype hardware synthesis system from ADL will be demonstrated. Compact specifications of application-specific, reconfigurable hardware in ADL will be automatically synthesized to Verilog. Using commercial FPGA synthesis tools, or NTT's synthesis software, the Verilog can be mapped to silicon.

June 2000: We will build an emulation platform for malleable caches that will allow us to evaluate adaptive cache management for caches of very large (64MB) size. We will incorporate malleable cache specification within ADL, so we can target the synthesis of both processor hardware and malleable caches.

June 2001: The system described in Figure 1 will be completed, and adaptive processors with malleable caches will be automatically generated from an ADL specification.

Collaboration with NTT Researchers

We will collaborate with Hiroshi Sawada, Dr. K. Shimohara and other members of the adaptive architecture group at NTT. The FPGA CAD software being developed at NTT can be evaluated using the hardware (Verilog) models of application-specific processors generated by the architecture exploration system. These models will be quite structured and hierarchical and the development of FPGA synthesis methods that exploit this structure is an interesting and worthwhile problem that the NTT and MIT groups can work on collaboratively. Adaptive cache management algorithms and new cache designs developed may be of interest to computer architects at NTT.