Abstract

As the importance of the development of embedded processor-based systems grows, the need for systematic methodology to rapidly explore several processor architectures for system-on-a-chip solutions also increases. In this thesis, some optimization and exploration issues that arise in different subsets of the embedded system architecture using program analysis techniques are studied.

First, several program analysis and optimization techniques to exploit newly added DSP architectural features to a basic RISC core processor are developed. By adding DSP specific features to a RISC core and developing a back-end code converter that modifies compiled assembly codes to support these added functions, a compiler-friendly programmable DSP processor is synthesized. The development of the code converter is not only much simpler than the whole compiler development, but also the technology can be adopted to other instruction-set architectures fairly easily. The compiler-driven performance comparison using DSP kernel and application benchmarks show that the code quality is significantly better than that of compilers for conventional DSP processors.

Second, using extended versions of program analysis techniques employed in the case of RISC-based DSP processor, a 2-way VLIW multimedia processor for embedded applications and its software environment is developed. Due to its combined SIMD and VLIW approach in addition to the powerful software
Abstract

environment including a high-performance VLIW compiler system featuring a code
converter, an EPS (Enhanced Pipeline Scheduling)-based scheduler, and a fast
compiled simulator, this proposed multimedia processor shows a comparable
performance in terms of the number of execution clock cycles and much smaller
code sizes when compared to the TMS320C62x.

Third, several static and dynamic methods to analyze given application programs
and determine the weight of each code block are developed for an instruction
caching scheme for multimedia applications, which combine the advantages of both
internal memory and cache by allowing the programmer to selectively prioritize
parts of cache blocks. The proposed caching scheme assigns priorities to each code
block, and tries to keep code blocks that are more important longer in the case of set
associative caches. The priority information can be specified by a programmer by
analyzing the program flow of multimedia applications, which is known to be
highly regular and predictable.

Keywords – embedded processor, systems-on-chip, architecture exploration,
optimization, software analysis, multimedia processor, digital signal processor,
data flow analysis, code converter, compiler-friendly, architecture synthesis,
performance evaluation, cache memory

Student Number – 96420-801