Compilation Methods for the Address Calculation Units of Embedded Processor Systems

CLIFFORD LIEM
Laboratoire TIMA, Institut National Polytechnique de Grenoble, 46, ave Félix Viallet,
38000 Grenoble cedex - France
and
SGS-Thomson Microelectronics, 850, rue Jean Monnet, BP 16 - 38921 Crolles, France

PIERRE PAULIN
SGS-Thomson Microelectronics, 850, rue Jean Monnet, BP 16 - 38921 Crolles cedex - France

AHMED JERRAYA
Laboratoire TIMA, Institut National Polytechnique de Grenoble, 46, ave Félix Viallet,
38000 Grenoble cedex - France

Received January 30, 1996; Accepted March 6, 1996

Abstract. An essential component of today’s embedded system is an instruction-set processor running real-time software. All variations of these core components contain at least the minimum data-flow processing capabilities, while a certain class contain specialized units for highly data-intensive operations for Digital Signal Processing (DSP). For the required level of memory interaction, the parallel executing Address Calculation Unit (ACU) is often used to tune the architecture to the memory access characteristics of the application. The design of the ACU is performance critical. In today’s typical design flow, this design task is somewhat driven by intuition as the transformation from application algorithm to architecture is complex and the exploration space is immense. Automatic utilities to aid the designer are essential; however, the key compilation techniques which map high-level language constructs onto addressing units have lagged far behind the emergence of these units. This paper presents a new retargetable approach and prototype tool for the analysis of array references and traversals for efficient use of ACUs. In addition to being an enhancement to existing compiler systems, the ArrSyn utility may be used as an aid to architecture exploration. A simple specification of the addressing resources and basic operations drives the available transformations and allows the designer to quickly evaluate the effects on speed and code size of his/her algorithm. Thus, the designer can tune the design of the ACU toward the application constraints. ArrSyn has been successfully used together with a C compiler developed for a VLIW architecture for an MPEG audio decoding application. The combination of these methods with the C compiler showed on average a 39% speedup and 29% code size reduction for a representative set of DSP benchmarks.

1. Introduction

The emergence of deep submicron technologies is increasingly inviting more and more sophisticated embedded systems on chip. A key component of these systems is the embedded core, a miniature instruction-set processor running real-time software. As one example, the SGS-Thomson (ST) Video Codec [1], [2] contains 2 embedded cores, while its successor, the future ST Integrated Video Telephone, may contain as many as 5 embedded cores! It is the flexibility of software that has been inviting the designer to use embedded cores as he is able to compensate for late specification changes, design for product evolution, and
facilitate macroblock reuse. As well, embedded cores encourage concurrent engineering practices between hardware and software design teams.

For application domains such as multimedia, the key aspect of the embedded DSP core is the ability for number crunching. As data intensive algorithms push for higher speeds and throughput, access to data memories become the limiting factor. In response to this, designers have conceived the Address Calculation Unit (ACU) (sometimes termed Address Generation Unit (AGU), Address Arithmetic Unit (AAU), or Memory Management Unit (MMU)), an arithmetic unit which works in parallel to the main Data Calculation Unit (DCU). The ACU works solely on address generation to ensure efficient retrieval and storage of data that is calculated on the DCU. In most cases, the ACU works in a post-modify (increment/decrement) fashion to ensure high speed. Pre-modify addressing is rare because this would require at least two operations to occur in the same instruction cycle, namely the address calculation, then the memory access.

Post-increment/decrement address units are present on countless general-purpose DSPs and cores, for example the SGS-Thomson D950 core [3], the Motorola 56000 and new 56800 series, the Texas Instruments TMS320C25 [4], and the Lode DSP Engine [5], to name a few. They are also common in Application Specific Instruction-Set Processors (ASIPs) used in applications such as MPEG audio [6] [8], Dolby decoding [9], and DSP for telecommunications [10]. A typical linear post-indexing ACU for DSPs is depicted in Figure 1. Through each of these examples, the design of the ACU varies in the number and function of the available registers and the operation that each supports.

Although ACUs have existed for some time, the compiler techniques for mapping high-level language constructs onto the register structures are immature. This is immediately reflected in the poor performance of today’s DSP compilers [7]. The problem of mapping array structures onto these calculation units manifests itself in two ways:

1. difficulties of dealing with special-purpose registers and connections.