Performance Improvement through Predicated Execution in VLIW Machines

Morteza Biglari-Abhari

Thesis submitted for the degree of

Doctor of Philosophy

Department of Electrical and Electronic Engineering
University of Adelaide
South Australia
5005

February, 2000

Contents

Abstract

State-of-the-art VLSI technology provides the opportunity to implement a complex microarchitecture with many functional units capable of parallel operation. To employ the machine parallelism effectively requires extracting ILP from the application programs as much as possible. Hardware techniques to extract ILP have an impact on cycle time, which may result in lower performance. To achieve higher clock rates, hardware can be used as only the execution engine and ILP extraction is transferred to the compiler. One way to achieve this is through employing the Very Long Instruction Word (VLIW) architecture to implement a wide-issue ILP processor.

One of the major obstacles to the exploitation of ILP is the uncertain change in instruction flow caused by conditional branches. Keeping multiple functional units busy most of the time requires the executing operations from multiple execution paths. Therefore, techniques should be employed to reduce the impact of conditional branches which are less predictable in general-purpose applications. Predicated or guarded execution as an architectural model reduces these limitations. Predicated execution refers to the conditional execution of an operation based on the value of a boolean source operand, referred to as the predicate. Predicated execution is exploited through a structure called the hyperblock which was developed by the IMPACT compiler group.

This thesis investigates techniques to achieve performance improvement in VLIW machines through predicated execution. We describe an experimental processor based on VLIW architecture called EVA. The back-end of an optimising compiler for EVA is implemented in order to investigate the proposed techniques. These include improving the quality of the generated code through estimating the resource usage during the code generation. This is applied at the time of generating the predicated code (through if-conversion) and at the time of prepass scheduling.

VLIW architectures traditionally have not been used extensively for general-purpose ap-

plications. One of the major reasons is lack of object code compatibility among different implementations of the same architecture. This is due to the detailed microarchitectural information employed by the compiler to generate the binary code. When the assumed microarchitectural features are changed, the previously generated code not only may suffer performance loss, but also may generate incorrect results. We propose a new approach to overcome this problem.