

09 PH
B593.



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

List of Figures	v
Abstract	x
Acknowledgements	xiv
1 Introduction	1
1.1 Contribution of the Thesis	4
1.2 Outline of the Thesis	4
2 Compiler Techniques in ILP Processing	6
2.1 Introduction	6
2.2 Control Dependency Reduction	8
2.2.1 Speculative Execution	8
2.2.2 Predicated Execution	12
2.3 Data Dependency Reduction	14
2.3.1 Register Renaming	14
2.3.2 Memory Access Disambiguation	15
2.3.3 Operand Value and Dependency Prediction	19
2.4 Static Instruction Scheduling	19
2.4.1 Acyclic Global Scheduling	20
2.4.2 Cyclic Global Scheduling	25
2.5 Hardware Techniques	28
2.6 Summary	30

3	EVA: An Experimental VLIW Architecture	32
3.1	VLIW Architectural Features	32
3.1.1	Multiple-Operation Instruction (MultiOp)	33
3.1.2	Non-Unit Assumed Latency (NUAL)	33
3.2	Traditional VLIW Shortcomings	35
3.2.1	Inefficient Memory Usage	35
3.2.2	Object-Code Incompatibility	36
3.3	Architecture of EVA	38
3.3.1	Instruction Set Architecture	38
3.3.2	Architectural Support for Speculative Execution	38
3.3.3	Architectural Support for Predicated Execution	39
3.3.4	Architectural Support for Memory Access Disambiguation	40
3.4	Implementation of EVA Architecture: An Example	40
3.5	Summary	42
4	The VLIW Compiler for the EVA	44
4.1	Compiler Structure	44
4.1.1	Motivation	44
4.1.2	General Structure	45
4.2	Predicate-Sensitive Analysis	48
4.2.1	Overview of Partition Graph Construction	51
4.3	Supporting Predicated Execution	53
4.3.1	Hyperblock Formation	53
4.4	Code Optimisation	55
4.4.1	Classical Optimisations	57
4.4.2	ILP Optimisations	59
4.4.3	Hyperblock-specific Optimisations	60
4.5	Instruction Scheduling	63
4.5.1	Dependency Representation	64
4.5.2	Resource Management	65
4.5.3	Scheduling Process	66
4.6	Register Allocation	72
4.6.1	Live Range Construction	73

4.6.2	Interference Graph Construction	74
4.6.3	Graph Colouring	76
4.7	Summary	78
5	Experimental Evaluation of the EVA Compiler	80
5.1	Simulation Techniques for ILP Processing	80
5.2	Experimental Results	84
5.2.1	Methodology	84
5.2.2	Results	87
5.2.3	Results for modified static priority calculation algorithm	89
5.2.4	Effects of Non-perfect Cache	91
5.3	Summary	93
6	Partial Path Selection for Hyperblock Formation	95
6.1	Issues in Hyperblock Formation	95
6.2	Related Work	97
6.3	A New Approach	101
6.4	Experimental Results	103
6.5	Summary	107
7	Supporting Binary Compatibility in VLIW Machines	109
7.1	Related Works	109
7.2	A New Approach	115
7.2.1	Speculative Scheduling	116
7.2.2	DVG Scheduling Algorithm	117
7.3	Comparison with Related Works	121
7.4	Experimental Results	123
7.5	Summary	127
8	Summary and Conclusion	128
8.1	Summary	128
8.2	Future Directions	129
A	A Brief Description of Libraries and Different Passes in EVA VLIW Compiler	131

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.