



Technische Universität Dresden

Approaches to Code Generation for Synchronous Transfer Architecture (STA)

Jie Guo

von der
Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieurs
(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. Eduard Jorswieck	
Gutachter:	Prof. Dr.-Ing. Gerhard Fettweis	Tag der Einreichung: 07.05.2008
	Prof. Dr. rer.nat. Rainer Leupers	Tag der Verteidigung: 25.08.2008
	Prof. Dr. habil. Uwe Assmann	

Berichte aus der Informatik

Jie Guo

**Approaches to Code Generation for
Synchronous Transfer Architecture (STA)**

Shaker Verlag
Aachen 2008

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Zugl.: Dresden, Techn. Univ., Diss., 2008

Copyright Shaker Verlag 2008

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8322-7615-7

ISSN 0945-0807

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen

Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9

Internet: www.shaker.de • E-Mail: info@shaker.de

Abstract

In order to meet the ever increasing demands for higher performance, nowadays most Digital Signal Processors (DSPs) incorporate irregular architectures. However, traditional code generation and optimization techniques often fail to achieve a satisfactory code quality for irregular architectures. One important reason for this drawback is the interdependence among code generation phases involving instruction selection, register allocation and instruction scheduling. This is the well-known phase-coupling problem.

To ease this problem, three integrated code generation methods are studied based on the Synchronous Transfer Architecture (STA). The first one aims to find the optimized code by formulating the code generation as an Integer Linear Programming (ILP) problem. The second one couples instruction selection and register allocation into one phase with a two-step register allocator. The third one combines register allocation and instruction scheduling by using greedy algorithms that are widely used for network planning and optimization. All these methods have been implemented in a Matlab compiler for a SIMD-VLIW STA-based processor. In this thesis, it will be shown that much better performance can be achieved by the integrated code generation in comparison to that obtained in the conventional way.

Besides the studies on the SIMD processor, the commercial Compiler development System (CoSy) is applied to generate assembly code for a scalar, VLIW, STA-based processor. Because of the architectural features of this processor, some adaptations for STA data transfer and addressing mode have been carried out in the code generation.

Acknowledgments

This thesis results from my work at the Vodafone Chair Mobile Communications Systems at Technische Universität Dresden.

I am particularly grateful to Professor Gerhard Fettweis. I would like to thank Gerhard for his invaluable advice and his guidance and support throughout my work. Because of his outstandingly encouraging and guiding personality, I had not only a lot of freedom to do research, but also enjoyed the productive working atmosphere in the last years.

Further, I would like to express my gratitude to Dr. Emil Matus, Dr. Boris Boesler and all the members in CATS group. I am glad and proud to be a member of this group.

Finally, I would like to thank my family and especially my husband.

Contents

List of Figures	vii
List of Tables	x
Abbreviations	xii
1 Introduction	1
1.1 Background and Problems	1
1.2 Code Generation for STA	2
1.3 Objectives of this Thesis	3
1.4 Outline	3
2 Concepts of Code Generation	5
2.1 Compilation Phases	5
2.2 Code Generation	11
2.2.1 Instruction Selection	11
2.2.2 Register Allocation and Assignment	13
2.2.3 Instruction Scheduling	15
3 Code Generation for Synchronous Transfer Architecture	20
3.1 Code Generation for Digital Signal Processors	22
3.2 Synchronous Transfer Architecture (STA)	23
3.3 Code Generation for STA	25
3.4 Integrated Code Generation	26
3.4.1 Phase Coupling Problem	26
3.4.2 Related Work	28
3.4.3 Overview of My Studies	29

4	Integrated Code Generation Using Integer Linear Programming	31
4.1	Introduction	31
4.2	Mathematical Foundations	32
4.2.1	The Theory of Linear Programming	32
4.2.2	The Theory of Integer Linear Programming	39
4.3	ILP Model for STA Code Generation	42
4.3.1	Basic Coefficients	43
4.3.2	Objective Function	47
4.3.3	Constraints Part I: Variables	48
4.3.4	Constraints Part II: STA Data Routing	49
4.3.5	Constraints Part III: Machine Resource	53
4.3.6	Constraints Part IV: Data Dependence	58
4.4	Model Analysis	60
4.4.1	Model Complexity	60
4.4.2	Solving Time Analysis	62
4.5	Experimental Results	64
4.5.1	Testing and Results	64
4.5.2	Discussion and Summary	68
5	Integrated Code Generation Using Two-Step Register allocation	72
5.1	Introduction	72
5.2	Mathematical Foundations	75
5.2.1	Graph Theory	75
5.2.2	Graph Coloring (GC) Problem	76
5.3	Local Two-Step Register Allocation	77
5.3.1	Traditional GC for Local Register Allocation	77
5.3.2	Extended GC for Local Output Register Allocation	79
5.3.3	Standard GC for General-purpose Register Allocation	86
5.3.4	Re-scheduling	86
5.4	Global Two-Step Register Allocation	87
5.4.1	GC for Global Register Allocation	87
5.4.2	Extended GC for Global Output Register Allocation	88
5.5	Experimental Results	90
5.5.1	Testing and Results	91
5.5.2	Discussion and Summary	94

6	Integrated Code Generation Using Greedy Algorithms	95
6.1	Introduction	95
6.2	Mathematical Foundations	96
6.2.1	Combinatorial Optimization	96
6.2.2	Minimum Spanning Tree (MST) Problem	96
6.2.3	Knapsack Problem	99
6.3	Code Generation Based on Greedy Algorithms	99
6.3.1	Instruction Scheduling Modeled as MST Problem	100
6.3.2	Register Allocation Modeled as Knapsack Problem	103
6.4	Experimental Results	105
6.4.1	Testing and Results	105
6.4.2	Discussion and Summary	107
7	Discussion of Three Integrated Approaches	109
7.1	Experimental Environment	109
7.1.1	Compiler Construction	109
7.1.2	SAMIRA Architecture	111
7.2	Comparison of Code Performance	112
7.2.1	Code Size	112
7.2.2	Execution Speed	113
7.2.3	Compilation Time	113
7.2.4	Register Consumption	114
7.2.5	Memory and Register Access	115
7.3	Discussion and Summary	116
8	Cosy Compiler on STA	119
8.1	Basics of CoSy	119
8.1.1	CCMIR	120
8.1.2	Engines	123
8.1.3	BEG	124
8.2	Features of SIOUX Architecture	126
8.3	The Implementation of Our Code Generator	130
8.3.1	Realization of STA Direct Data Routing	130
8.3.2	The Scheduler	136
8.3.3	The Rules	139
8.3.4	Calling Conventions	145

8.4	Experimental Results	146
8.4.1	Testing and Results	148
8.4.2	Discussion and Summary	150
9	Conclusions	151
A	Solving LP Problem Using GLPK	155
A.1	LP Problem	155
A.2	LP Model	155
A.3	Results Using GLPK	156
B	Instruction Set Architecture of the SAMIRA	158
B.1	SAMIRA Core	158
B.2	Instruction Set Architecture	158
B.2.1	Scalar Units	158
B.2.2	Vector Units	159
B.2.3	Decoder Unit	159
B.2.4	Sequencing Units	159
B.2.5	Inter-Connection Unit (ICU)	159
B.2.6	General-purpose Registers	160
C	Benchmarks For The Matlab Compiler	165
C.1	Introduction	165
C.2	FIR	165
C.3	IIR	169
C.4	LMS	171
C.5	Convolution	180
C.6	FFT	181
C.7	DCT	191
D	Instruction Set Architecture of the SIOUX	207
D.1	SIOUX Core	207
D.2	Instruction Set Architecture	207
D.2.1	Computation Units	207
D.2.2	General-purpose Registers	207
D.2.3	Addressing Units	208
D.2.4	Decoder Unit	208

D.2.5 Sequencing Units	209
D.2.6 Memory Load/Store Unit	209
Bibliography	209