

RESEARCH

Martin Kumm

Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays



Springer Vieweg

Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays

Martin Kumm

Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays

With a preface by Prof. Dr.-Ing. Peter Zipf

 Springer Vieweg

Martin Kumm
Kassel, Germany

Dissertation of Martin Kumm in the Department of Electrical Engineering and
Computer Science at the University of Kassel. Date of Disputation: October 30th, 2015

ISBN 978-3-658-13322-1 ISBN 978-3-658-13323-8 (eBook)
DOI 10.1007/978-3-658-13323-8

Library of Congress Control Number: 2016935387

Springer Vieweg

© Springer Fachmedien Wiesbaden 2016

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer Vieweg imprint is published by Springer Nature
The registered company is Springer Fachmedien Wiesbaden GmbH

Preface

As silicon technology advances, field programmable gate arrays appear to gain ground against the traditional ASIC project starts, reaching out to form the mainstream implementation basis. Their predefined structures result in an essential inefficiency, or performance gap at all relevant axes, i.e. clock frequency, power and area. Thus, highly optimised system realisations become more and more important to use this technology at its best. Microarchitectures and their adaptation to the FPGA hardware, combined with an optimal matching of model structures and FPGA structures, are two points of action where engineers can try to get optimally balanced solutions for their designs, thus fighting the performance gap towards the ASIC reference.

While microarchitecture design based on the knowledge of FPGA structures is located in the domain of traditional hardware engineering, the mapping and matching is based on EDA algorithms and thus strongly related to computer science. Algorithms and the related sophisticated tools are permanently in short supply for leading edge optimisation needs.

Martin's dissertation deals with the algorithmic optimisation of circuits for the multiplication of a variable with constants in different flavours. As this type of operations is elementary in all areas of digital signal processing and also usually on the critical path, his approaches and results are of high relevance not only by themselves but also as a direction for further research. His direct contributions are the advancement of algorithmic treatment of pipeline-based multiplier circuits using heuristics and exact optimisation algorithms, the adaptation of several algorithms to the specific conditions of field programmable gate arrays, specifically lookup-table based multipliers, ternary adders and embedded multipliers with fixed word length, and the transfer of his findings to a floating-point multiplier architecture for multiple constants.

Along with all the accompanying details, this is a large range of topics Martin presents here. It is an impressive and comprehensive achievement, convincing by its depth of discussion as well as its contributions in each of the areas.

Martin was one of my first PhD candidates. Thrown into the cultural conflict between computer science and electrical engineering he soon developed a sense for the symbiosis of both disciplines. The approach and results are symptomatic for this developing interdisciplinary area, where systems and their optimisation algorithms are developed corporately.

I found Martin's text exciting to read, as it is a comprehensive work within the ongoing discussion of algorithmic hardware optimisation. His work is supported by a long list of successful publications. It is surely a contribution worth reading.

Kassel, 22. of April, 2016

Prof. Dr.-Ing. Peter Zipf

Abstract

Digital signal processing (DSP) plays a major role in nearly any modern electronic system used in mobile communication, automotive control units, biomedical applications and high energy physics, just to name a few. A very frequent but resource intensive operation in DSP related systems is the multiplication of a variable by several constants, commonly denoted as multiple constant multiplication (MCM). It is needed, e. g., in digital filters and discrete transforms. While high performance DSP systems were traditionally realized as application specific integrated circuits (ASICs), there is an ongoing and increasing trend to use generic programmable logic ICs like field programmable gate arrays (FPGAs). However, only little attention has been paid on the optimization of MCM circuits for FPGAs.

In this thesis, FPGA-specific optimizations of MCM circuits for low resource usage but high performance are considered. Due to the large FPGA-specific routing delays, one key for high performance is pipelining. The optimization of pipelined MCM circuits is considered in the first part of the thesis. First, a method that optimally pipelines a given (not necessary optimal) MCM circuit using integer linear programming (ILP) is proposed. Then, it is shown that the direct optimization of a pipelined MCM circuit, formally defined as the pipelined MCM (PMCM) problem, is beneficial. This is done by presenting novel heuristic and optimal ILP-based methods to solve the PMCM problem. Besides this, extensions to the MCM problem with adder depth (AD) constraints and an extension for optimizing a related problem – the pipelined multiplication of a constant matrix with a vector – are given.

While these methods are independent of the target device and reduce the number of arithmetic and storage components, i. e., adders, subtracters and pipeline registers, FPGA-specific optimizations are considered in the second part of the thesis. These optimizations comprise the inclusion of look-up table (LUT)-based constant multipliers, embedded multipliers and the use of ternary (3-input) adders which can be efficiently mapped to modern FPGAs. In addition, an efficient architecture to perform MCM operations in floating point arithmetic is given.

Acknowledgements

First of all, I would like to thank my supervisor, Prof. Dr.-Ing. Peter Zipf, who gave me the opportunity to do my PhD under his supervision. He provided an excellent guidance and was always open-minded to new and exceptional ideas, leading to creative and fruitful discussions. I also thank my colleagues at the digital technology group of the University of Kassel. In special, Michael Kunz, who had many ideas how to prepare the course material and Konrad Möller, who started as student with his project work and master's thesis and is now a highly-valued colleague. Our mathematicians, Diana Fanghänel, Dörthe Janssen and Evelyn Lerche, were always open to help with math related questions. The deep understanding in discrete optimization and ILP (and more) of Diana Fanghänel helped a lot in obtaining some of the results in this work. I am grateful to all the students that contributed to parts of this thesis with their bachelor or master's thesis, project work or student job: Martin Hardieck, who did a great job in the C++ implementation of RPAG extensions for ternary adders and the constant matrix multiplication (and the combination of both) during his project work, his diploma thesis and his student job; Katharina Liebisch, for implementing MCM floating point units for reference; Jens Willkomm, who realized the ternary adder at primitive level for Xilinx FPGAs during his project work; and Marco Kleinlein, who implemented flexible VHDL code generators based on the Floating Point Core Generator (FloPoCo) library [1]. I would also like to thank our external PhD candidates Eugen Bayer and Charles-Frederic Müller for interesting discussions and collaborative work. I also appreciate the financial support of the German research council which allowed me to stay in a post-doc position in Kassel.

During the PhD, I started cooperations with Chip Hong Chang and Mathias Faust from the Nanyang Technological University, Singapore, Oscar Gustafsson and Mario Garrido from the University of Linköping, Sweden, and Uwe Meyer-Baese from the Florida State University, USA. Especially, I am very grateful to Oscar Gustafsson to invite me as guest researcher in his group. Exchanging ideas with people from the same field expanded my horizon enormously. Thanks to the ERASMUS program, two visits in

Linköping were possible and further visits are planned. Another big help was the friendly support of other researchers. Levent Aksoy was always fast in answering questions, providing filter benchmark data or even the source code of their algorithms. My thanks also go to Florent de Dinechin and all the contributors of the FloPoCo library [1] for providing their implementations as open-source [2]. Open-source in research simplifies the comparison with other work which inspired me to publish the presented algorithms as open-source, too.

Last but not least, I want to kindly thank my wife Nadine and my son Jonathan. Their love and support gave me the energy to create this thesis.

Contents

Preface	V
Abstract	VII
Acknowledgements	IX
Contents	XI
List of Figures	XIX
List of Tables	XXIII
Abbreviations	XXVII
Nomenclature	XXX
Author's Publications	XXXI
1 Introduction	1
1.1 Motivation	1
1.2 Goal and Contributions of the Thesis	2
1.3 Organization of the Thesis	3
1.4 Applications of Multiple Constant Multiplication	5
2 Background	9
2.1 Single Constant Multiplication	9
2.1.1 Binary Multiplication	9

2.1.2	Multiplication using Signed Digit Representation . . .	10
2.1.3	Multiplication using Generic Adder Graphs	11
2.2	Multiple Constant Multiplication	12
2.3	Constant Matrix Multiplication	12
2.4	Definitions	13
2.4.1	Adder Graph Representation of Constant Multiplication	14
2.4.2	Properties of the \mathcal{A} -Operation	15
2.4.3	\mathcal{A} -Sets	17
2.4.4	The MCM Problem	17
2.4.5	MCM with Adder Depth Constraints	18
2.5	Field Programmable Gate Arrays	19
2.5.1	Basic Logic Elements of Xilinx FPGAs	21
2.5.2	Basic Logic Elements of Altera FPGAs	21
2.5.3	Mapping of Adder Graphs to FPGAs	22
2.5.4	Cost Models for Pipelined Adders and Registers . . .	26
2.6	Related Work	28
2.6.1	Single Constant Multiplication	28
2.6.2	Multiple Constant Multiplication	29
2.6.3	Modified Optimization Goals in SCM and MCM Op- timization	33
2.6.4	Pipelined Multiple Constant Multiplication	33

I The Pipelined Multiple Constant Multiplication Problem 37

3 Optimal Pipelining of Precomputed Adder Graphs 39

3.1	Pipelining of Adder Graphs using Cut-Set Retiming	39
3.2	Minimized Pipeline Schedule using Integer Linear Programming	41
3.2.1	Basic ILP Formulation	42

3.2.2	Extended ILP Formulation	43
3.3	Experimental Results	44
3.3.1	Register Overhead	45
3.3.2	Slice Comparison	46
3.3.3	Device Utilization	47
3.3.4	Silicon Area	49
3.4	Conclusion	51
4	The Reduced Pipelined Adder Graph Algorithm	53
4.1	Definition of the PMCM Problem	54
4.1.1	Cost Definition	55
4.1.2	Relation of PMCM to MCM	56
4.2	The RPAG Algorithm	56
4.2.1	The Basic RPAG Algorithm	57
4.2.2	Computation of Single Predecessors	58
4.2.3	Computation of Predecessor Pairs	59
4.2.4	Evaluation of Predecessors	62
4.2.5	The Overall RPAG Algorithm	63
4.2.6	Computational Complexity	68
4.3	Lower Bounds of Pipelined MCM	71
4.4	Implementation	72
4.5	Experimental Results	73
4.5.1	Registered Operations and CPU Runtime	73
4.5.2	Influence of the Search Width Limit to the Optimization Quality	74
4.5.3	Comparison of Cost Models	74
4.5.4	Comparison Between Gain Functions	77
4.5.5	Influence of the Pipeline Stages to the Optimization Quality	79

4.5.6	Comparison with the Optimal Pipelining of Precomputed Adder Graphs	81
4.5.7	Comparison Between Aksoy's Method and RPAG	84
4.6	Conclusion	84
5	Optimally Solving MCM Related Problems using ILP	87
5.1	Related Work	87
5.2	ILP Formulations of the PMCM Problem	91
5.2.1	PMCM ILP Formulation 1	92
5.2.2	PMCM ILP Formulation 2	94
5.2.3	PMCM ILP Formulation 3	96
5.3	Extensions to Adder Depth Constraints and GPC Minimization	96
5.4	Further Extensions of the ILP Formulations	99
5.5	Analysis of Problem Sizes	100
5.5.1	Evaluation of Set Sizes of \mathcal{A}^s and \mathcal{T}^s	100
5.5.2	Relation to FIR Filters	100
5.6	Experimental Results	101
5.6.1	FIR Filter Benchmark	102
5.6.2	Benchmark Filters from Image Processing	103
5.6.3	Optimization Results with High-Level Cost Model	104
5.6.4	Optimization Results with Low-Level Cost Model	105
5.6.5	Synthesis Results	105
5.6.6	Optimization Results for Minimal Adder Depth	107
5.7	Conclusion	110
6	A Heuristic for the Constant Matrix Multiplication Problem	113
6.1	Related Work	115
6.2	RPAG extension to CMM	116
6.2.1	Adder Graph Extensions to CMM	116

6.2.2	Definition of the PCMM and CMM_{MAD}/CMM_{BAD} Problems	117
6.2.3	Computation of Predecessors	118
6.2.4	Evaluation of Predecessors	119
6.2.5	Overall Algorithm	119
6.3	Experimental Results	120
6.4	Conclusion	124
II FPGA Specific MCM Optimizations		125
7	Combining Adder Graphs with LUT-based Constant Multipliers	127
7.1	Related Work	127
7.2	LUT-based Constant Multiplication	128
7.3	LUT Minimization Techniques	129
7.4	ILP Formulation for the Combined PALG Optimization . . .	130
7.5	Experimental Results	133
7.6	Conclusion	135
8	Optimization of Hybrid Adder Graphs	139
8.1	Implementation Techniques for Large Multipliers	140
8.2	Hybrid PMCM Optimization with Embedded Multipliers . .	142
8.2.1	Depth of the Input PAG	146
8.2.2	Depth of the Output PAG	146
8.3	RPAG Modifications	146
8.4	Experimental Results	147
8.4.1	Trade-off Between Slices and DSP Resources	147
8.4.2	Reducing Embedded Multipliers in Large MCM Blocks	147
8.5	Conclusion	149

9	Floating Point Multiple Constant Multiplication	153
9.1	Constant Multiplication using Floating Point Arithmetic . . .	153
9.2	Floating Point MCM Architecture	154
9.2.1	Multiplier Block	154
9.2.2	Exponent Computation	155
9.2.3	Post-Processing	156
9.3	Experimental Results	156
9.3.1	Floating Point SCM	157
9.3.2	Floating Point MCM	158
9.4	Conclusion	161
10	Optimization of Adder Graphs with Ternary (3-Input) Adders	163
10.1	Ternary Adders on Modern FPGAs	163
10.1.1	Realization on Altera FPGAs	164
10.1.2	Realization on Xilinx FPGAs	165
10.1.3	Adder Comparison	166
10.2	Pipelined MCM with Ternary Adders	167
10.2.1	\mathcal{A} -Operation and Adder Depth	168
10.2.2	Optimization Heuristic RPAGT	168
10.2.3	Predecessor Topologies	169
10.3	Experimental Results	172
10.4	Conclusion	174
11	Conclusion and Future Work	177
11.1	Conclusion	177
11.2	Future Work	178
11.2.1	Use of Compressor Trees	178
11.2.2	Reconfigurable MCM	179
11.2.3	Optimal Ternary SCM circuits	179

A	Benchmark Coefficients	181
A.1	Two-Dimensional Filters	181
A.2	Floating-Point Filters	183
B	Computation of the \mathcal{A}_* Set	187
	Bibliography	189

List of Figures

- 1.1 Graphical representation of the relation of the main thesis chapters 3-10 4
- 1.2 Structure of an FIR filter in (a) direct form and (b) transposed form 6

- 2.1 Different adder circuits to realize a multiplication with 93 . . . 10
- 2.2 Different adder circuits to realize a multiplication with coefficients 19 and 43 13
- 2.3 Example CMM operation with constant $19 + j43$ 14
- 2.4 Adder graph realizing the multiplication with the coefficients $\{19, 43\}$ 16
- 2.5 Simplified architecture of a generic FPGA layout 20
- 2.6 Simplified BLE of Xilinx FPGA families 22
- 2.7 Simplified BLE (half ALM) of Altera Stratix III to Stratix IV FPGAs 23
- 2.8 BLE configurations for pipelined adder graphs on Xilinx Virtex 4 FPGAs 24
- 2.9 BLE configuration for an 8 bit register on Xilinx Virtex 6 FPGAs 25
- 2.10 Adder graphs realizing the multiplication with the coefficients $\{19, 43\}$ 27

- 3.1 Adder graphs for the coefficient target set $T = \{480, 512, 846, 1020\}$, (a) adder graph without pipelining, (b) pipelined adder graph after cut-set retiming (c) optimized adder graph 40
- 3.2 Resource and speed comparison of adder graph based methods 49

4.1	Different multiplier block realizations for the coefficients $\{44, 130, 172\}$	54
4.2	Predecessor graph topologies for (a)-(c) a single predecessor p , (d)-(e) a predecessor pair (p_1, p_2)	59
4.3	Decision tree of the overall RPAG algorithm for $T_{\text{odd}} = \{63, 173\}$ and $L = 1$	68
4.4	Optimization results for different N and B_c with $L = 0$	75
4.5	Optimization results for different search width limits L and $B_c = 16$ bit	76
4.6	Resulting low-level improvement when optimized with the low-level cost model compared to the high-level cost model	77
4.7	Comparison of different gain functions for the low-level cost model	78
4.8	Average improvement of registered operations when using one extra stage ($S = D_{\text{max}} + 1$) compared to $S = D_{\text{max}}$ and the best of both result for $L = 0$ and $L = 10$	80
4.9	Histogram of number of instances (out of 100) which are better when using one extra stage ($S = D_{\text{max}} + 1$) compared to $S = D_{\text{max}}$	81
4.10	Example PAGs where an increased pipeline depth is beneficial	82
4.11	Pipelined adder graph SCM example where an increased pipeline depth is beneficial	83
5.1	Example hypergraphs for target set $T = \{44, 70, 104\}$ (a) subset of the search space including the Steiner nodes 3, 7 and 9, (b) one possible solution hypertree by selecting 7 as Steiner node (c) an invalid solution containing loops	88
5.2	PAG solutions of two instances of the MIRZAEI10 benchmark	103
6.1	Different realizations of the example PCMM circuit	114
6.2	Adder graph representation of PCMM circuit in Figure 6.1(d)	117
6.3	Benchmark of random 8 bit $N \times N$ matrices (a) absolute number of adders (b) improvement of RPAG-CMM over multiple use of RPAG	121

7.1	4×12 bit signed multiplier using 4-input LUTs [3]	130
7.2	Pipelined LUT based multiplier	131
7.3	Example PAG/PALG of the highpass 5×5 instance with $B_i = 10$ bit	135
8.1	A single precision mantissa multiplier using DSP48E blocks on a Xilinx Virtex 6 FPGA	141
8.2	An optimized double precision mantissa multiplier on a Xilinx Virtex 6 FPGA [4]	143
8.3	A pipelined hybrid adder graph consisting of input adder graph, embedded multipliers ($S_i = 3$) and output adder graph ($S_o = 2$)	145
8.4	Pareto front for filter $N = 28$ of benchmark set [5]	148
8.5	Hybrid adder/multiplier graph adder graph for the $N_{uq} = 5$ benchmark filter	151
8.6	Adder graph for the $N_{uq} = 5$ benchmark filter with maximum post adder usage	152
9.1	Proposed architecture for the single precision floating point MCM operation	155
9.2	Detailed architecture of one round, normalize and exponent selection block of Figure 9.1	157
9.3	Required adders for exponents	160
10.1	Pipelined adder graphs of a constant multiplier with coefficients $T = \{7567, 20406\}$ using (a) two-input adders (b) ternary adders	164
10.2	Realization of ternary adders on (a) Altera Stratix II-V adaptive logic modules (ALMs) (b) Xilinx Virtex 5-7 slices	165
10.3	Comparison of clock frequencies of two-input and ternary adders (top) and the frequency drop from two to three inputs (bottom) realized on Virtex 6 and Stratix IV FPGAs	167
10.4	Predecessor graph topologies for (a)-(f) one, (g)-(h) two and (i) three predecessors	170

10.5 Relative improvement of registered operations of RPAGT over RPAG	173
--	-----

List of Tables

- 3.1 Optimization results for H_{cub} and $H_{\text{cub,BAD}}$ using the benchmark set of [5], without pipelining (no. of adders), CSR pipelining with ASAP scheduling and the proposed optimal pipelining (registered operations). 46
- 3.2 Synthesis results of benchmark filters from [6] on Xilinx Virtex4 for the proposed method (optimal pipelining using $H_{\text{cub,BAD}}$) compared to other methods discussed. N is the number of coefficients 48
- 3.3 Device utilization using a mean metric for the Virtex 4 device family 50
- 4.1 Complexity of the MSD evaluation in topology (d) compared to a full search 71
- 4.2 Comparison between RPAG and the optimal pipelining of pre-computed adder graphs obtained by $H_{\text{cub,BAD}}$ based on benchmark filters from [5]. Abbreviations: N_{uq} : no. of unique coefficients, reg. add: registered adder/subtractor, pure regs: registers, reg. ops: registered operations 83
- 4.3 Comparison of the proposed method for the low-level ASIC model with HCUB-DC+ILP [7] based on benchmark filters from [7]. 84
- 5.1 Number of elements in \mathcal{A}^s and upper bound of triplets in \mathcal{T}^s for pipeline stages $s = 1 \dots 4$ and bit width $B_T = 2 \dots 32$. . . 101
- 5.2 Parameters of the used benchmark filters 105
- 5.3 Optimization results for the high-level cost model in terms of the number of registered operations for RPAG (v1) [8] using R iterations, RPAG (v2) and the optimal pipelined adder graphs (best marked bold) 106

5.4	Runtime comparison of different ILP formulations of the PMCM problem with high level cost function using different solvers and a timeout (TO) of 8 h/28,800 sec	107
5.5	Optimization results for the low-level cost model in terms of the number of BLEs for RPAG (v1) [8] using R iterations, RPAG (v2) and the optimal pipelined adder graphs (best marked bold)	108
5.6	Comparison between synthesis results using the same benchmark instances as in Table 7.1 providing the actual BLEs as well as the maximum clock frequency f_{\max} (best marked bold) on a Virtex 6 FPGA	109
5.7	MCM _{MAD} benchmark with filters from Aksoy [9]	110
6.1	Comparison of different CMM methods for benchmark matrices from literature (best marked bold)	122
6.2	Synthesis results for the best reference design of Table 6.1 and the proposed method for CMM min AD. and pipelined CMM	123
7.1	Optimization results in terms of the number of BLEs for the previous methods RPAG [8] using R iterations and the optimal pipelined adder graphs	134
7.2	Synthesis results using the same benchmark instances as in Table 7.1 providing the actual BLEs as well as the maximum clock frequency f_{\max} (best marked bold) on a Virtex 6 FPGA	136
8.1	Constraints for adder depth AD_s , minimum word size B_s^{\min} , and maximum word size B_s^{\max} of the segments shown in Figure 8.3	145
8.2	Synthesis results of benchmark set [5]	148
8.3	MCM benchmark with single precision mantissas	150
8.4	MCM benchmark with double precision mantissas	150
9.1	Benchmark of floating point single constant multiplication	158
9.2	Required adders for exponents	160

- 9.3 Benchmark of floating point MCM using slices only (best marked bold) 160
- 9.4 Benchmark of floating point multiple constant multiplication using logic and embedded DSP48 blocks 161

- 10.1 Adder depth comparison for two-input and ternary adder graphs 169
- 10.2 Comparison of high-level operations between RPAG with two-input adders and the ternary adder optimization RPAGT . . 173
- 10.3 Synthesis results of MCM blocks optimized by RPAG with two-input adders and the ternary adder optimization RPAGT 174

- A.1 Convolution matrices of the benchmark filters 181
- A.2 Mantissas of the single precision benchmark filters 183
- A.3 Mantissas of the double precision benchmark filters 184