

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Designing Application-Specific Processors for Image Processing

A thesis presented in partial fulfilment of the
requirements for the degree of

Master of Science in Computer Science

Massey University, Palmerston North,
New Zealand

Aaron Bishell

2008

Abstract

Implementing a real-time image-processing algorithm on a serial processor is difficult to achieve because such a processor cannot cope with the volume of data in the low-level operations. However, a parallel implementation, required to meet timing constraints for the low-level operations, results in low resource utilisation when implementing the high-level operations. These factors suggested a combination of parallel hardware, for the low-level operations, and a serial processor, for the high-level operations, for implementing a high-level image-processing algorithm.

Several types of serial processors were available. A general-purpose processor requires an extensive instruction set to be able to execute any arbitrary algorithm resulting in a relatively complex instruction decoder and possibly extra FUs. An application-specific processor, which was considered in this research, implements enough FUs to execute a given algorithm and implements a simpler, and more efficient, instruction decoder. In addition, an algorithm's behaviour on a processor could be represented in either hardware (i.e. hardwired logic), which limits the ability to modify the algorithm behaviour of a processor, or "software" (i.e. programmable logic), which enables external sources to specify the algorithm behaviour.

This research investigated hardware- and software- controlled application-specific serial processors for the implementation of high-level image-processing algorithms and compared these against parallel hardware and general-purpose serial processors. It was found that application-specific processors are easily able to meet the timing constraints imposed by real-time high-level image processing. In addition, the software-controlled processors had additional flexibility, a performance penalty of 9.9% and 36.9% and inconclusive footprint savings (and costs) when compared to hardware-controlled processors.

Acknowledgements

Thanks go to my supervisors, Donald Bailey and Paul Lyons, for their patience and assistance in writing this thesis. It would not have been the same without their considerable knowledge in both image processing and formal writing. Their assistance has been greatly appreciated even though this may not have always been apparent.

Again, thanks to my friend Erica Jones for her friendship, support and writing assistance.

Table of Contents

1	Designing Application-Specific Serial Processors for Image Processing.....	1
2	Related Research.....	7
2.1	Hardware Description Languages	7
2.2	Architectural Design Space.....	10
2.2.1	Computer Architecture.....	11
2.2.2	Processor Architecture	13
2.3	Hardware-Software Codesign.....	20
3	Development of Hardware-Controlled Processors.....	23
3.1	Robot Soccer	25
3.2	Lens Distortion	30
3.2.1	Creating a System of Linear Equations.....	32
3.2.2	Solving the Linear Equations	34
3.2.3	Calculating the Barrel Distortion Component.....	38
3.3	Implementing Arithmetic Operations.....	39
3.4	Summary.....	47
4	Development of Software-Controlled Processors.....	49
4.1	Architecture and Controller Design.....	51
4.2	Functional Unit Partitioning	60
4.3	Tuning the Architecture.....	63
4.4	Programming.....	68
4.4.1	Automating Aspects of the Controller.....	69
4.4.2	Creating the Instruction Set Architecture.....	72
4.4.3	Logical Addressing	75
4.4.4	Automated Addressing	76
4.4.5	Exploiting Instruction-Level Parallelism.....	77
5	Evaluation.....	79
6	Conclusion and Future Work.....	85
	References.....	89
	Glossary.	95
	Appendices.....	97

List of Figures and Tables

Figure 1.1: Implementation Approaches for an Image Processing Algorithm	3
Figure 2.1: Harvard Architecture with a Single Processor	12
Figure 2.2: Flynn's taxonomy	13
Figure 2.3: Processor Architecture Taxonomy	15
Figure 2.4: TTA and VLIW Instruction Words	16
Figure 2.5: Differences between the VLIW and TTA Architectures	17
Figure 2.6: Top-Down and Bottom-Up Hardware-Software Partitioning Approaches	21
Figure 3.1: Example Frame from a Robot Soccer Video Stream	25
Figure 3.2: Plan View of a Robot	26
Figure 3.3: Complete Robot Soccer Algorithm Behaviour	29
Figure 3.4: An Image with Severe Barrel Distortion	30
Figure 3.5: Filling the Matrix ready for Gaussian Elimination	33
Figure 3.6: Algorithm for Eliminating a Column	35
Figure 3.7: Algorithm for Solving a Row	35
Figure 3.8: Algorithm for Back-Substitution	36
Figure 3.9: Algorithm for Gaussian Elimination	37
Figure 3.10: Determining the Barrel Distortion Component	38
Figure 3.11: Fixed Point Number Representation for U16.6	40
Figure 3.12: Division Algorithm	43
Figure 3.13: Arctangent Algorithm	44
Figure 3.14: Multiplication Algorithm	45
Figure 3.15: Handling Signed Arithmetic	46
Figure 4.1 CISC-sequential Architecture	51
Figure 4.2: Example Decoder	52
Figure 4.3: CISC-sequential and VLIW Instruction Words	52
Figure 4.4: VLIW Architecture	54
Figure 4.5: Instruction Prefetching for Multiple Instructions	56
Figure 4.6: Lens Distortion Instruction Word before Merging	64
Figure 4.7: Lens Distortion Instruction Word after Merging	65

Figure 4.8: Instruction Word Structure Example	69
Figure 4.9: Example Expressions Representing the Bit Ranges of Two Fields ..	70
Figure 4.10: Example Controller Logic for Passing Operand Data to an FU	70
Figure 4.11: Example Structure of Instruction Memory and Instruction Words	72
Figure 4.12: Instruction Expression Example	73
Figure 4.13: Instruction Signature for Jump	73
Figure 4.14: Assembly Code Example using Logical Addressing	75
Figure 4.15: Assembly Code Example for Automated Addressing	76
Figure 5.1: Footprint Comparison of Hardware- and Software- Controlled Processors	80
Figure 5.2: Performance Comparison of Hardware- and Software- Controlled Processors	81
Table 4.1: Instruction Prefetching Evaluation Differences	57
Table 4.2: Reductions in Footprint for Instruction Word Field Merging for Robot Soccer	64
Table 4.3: Reduction in Footprint for Instruction Word Field Merging for Lens Distortion	65
Table A.1	99
Table A.2	99
Table B.1	101
Table B.2	102
Table B.3	103
Table B.4	103