VERTIPH: A Visual Environment for Real-Time Image Processing on Hardware

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

Doctor of Philosophy
in
Computer Systems Engineering

at Massey University, Palmerston North, New Zealand.

Christopher Troy Johnston

2009
Abstract

This thesis presents VERTIPH, a visual programming language for the development of image processing algorithms on FPGA hardware. The research began with an examination of the whole design cycle, with a view to identifying requirements for implementing image processing on FPGAs. Based on this analysis, a design process was developed where a selected software algorithm is matched to a hardware architecture tailor made for its implementation. The algorithm and architecture are then transformed into an FPGA suitable design. It was found that in most cases the most efficient mapping for image processing algorithms is to use a streamed processing approach. This constrains how data is presented and requires most existing algorithms to be extensively modified. Therefore, the resultant designs are heavily streamed and pipelined.

A visual notation was developed to complement this design process, as both streaming and pipelining can be well represented by data flow visual languages. The notation has three views each of which represents and supports a different part of the design process. An architecture view gives an overview of the design's main blocks and their interconnections. A computational view represents lower-level details by representing each block by a set of computational expressions and low-level controls. This includes a novel visual representation of pipelining that simplifies latency analysis, multiphase design, priming, flushing and stalling, and the detection of sequencing errors. A scheduling view adds a state machine for high-level control of processing blocks. This extended state objects to allow for the priming and flushing of pipelined operations.

User evaluations of an implementation of the key parts of this language (the architecture view and the computational view) found that both were generally good visualisations and aided in design (especially the type interface, pipeline and control notations). The user evaluations provided several suggestions for the improvement of the language, and in particular the evaluators would have preferred to use the diagrams as a verification tool for a textual representation rather than as the primary data capture mechanism.

A cognitive dimensions analysis showed that the language scores highly for thirteen of the twenty dimensions considered, particularly those related to making details of the design clearer to the developer.
To my family
Acknowledgements

Thanks to my supervisors Donald and Paul for all the advice and encouragement that you gave me.

Thanks to my office mates for the interesting discussions, especially Kim for being a sounding board for ideas.

Thanks to my friends and flatmates for keeping me sane and making sure I got out once in a while, especially Andy, Jess, Øyvind, Reuben and Sarah.

I would like to thank my family for everything, without you I would not be the person I am today. Though you did not live to see me achieve this work thanks to Grandma and Granddad for the Lego that encouraged me into engineering and to Nana for encouraging me to think and go as far as I could. Thank you Pop for letting me build things and take them apart and for all the wood over the past years to keep me warm in winter.

To Nic and Bridget thanks for the dinners and proof reading. AJ for just being you.

To Mum and Dad thank you for all the support and encouragement with out it I would not have got this far.
# Table of Contents

1. Introduction ............................................................................................................. 1

1.1. Image Processing .................................................................................................. 2

1.2. Field Programmable Gate Arrays ....................................................................... 3

1.2.1. What are they? ............................................................................................... 3

1.2.2. Appropriate for Image Processing .................................................................. 5

1.3. Languages ............................................................................................................ 8

1.4. Thesis Statement ................................................................................................. 11

1.5. Thesis Contributions ........................................................................................... 12

2. Image processing, FPGAs and Visual languages ..................................................... 17

2.1. Image Processing Design Process ...................................................................... 17

2.1.1. Design Stages ............................................................................................... 17

2.1.2. Analysis of Example Implementations ......................................................... 26

2.1.3. Lens Distortion Correction ............................................................................ 27

2.1.4. Histogram Example ..................................................................................... 35

2.1.5. Object tracking ........................................................................................... 36

2.1.6. Connected Components .............................................................................. 44

2.1.7. Improved Algorithm ..................................................................................... 48

2.2. Insights Summary ............................................................................................... 50

2.3. Other work on Image Processing on FPGAs ....................................................... 50

2.3.1. More General FPGA Design Papers ............................................................. 54

2.4. Languages for Hardware Design ........................................................................ 56

2.4.1. Hardware Description Languages .................................................................. 56

2.4.2. High-level Languages .................................................................................. 58

2.4.3. Parallel Language Extensions ....................................................................... 59

2.4.4. Serial Language Extensions ......................................................................... 61

2.4.5. Hardware Software Co-Design Languages .................................................. 65
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.4</td>
<td>The Sequential Pipeline with Staggered Bars</td>
<td>126</td>
</tr>
<tr>
<td>5.3.5</td>
<td>The Sequential Pipeline with Detailed Bars</td>
<td>127</td>
</tr>
<tr>
<td>5.3.6</td>
<td>Data Explicit Pipeline Representation</td>
<td>128</td>
</tr>
<tr>
<td>5.3.7</td>
<td>Pipeline Control</td>
<td>130</td>
</tr>
<tr>
<td>5.4</td>
<td>Control Structures</td>
<td>132</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Dataflow and Pipeline Branching</td>
<td>136</td>
</tr>
<tr>
<td>5.5</td>
<td>Expression Editor</td>
<td>138</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Number Representation</td>
<td>139</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Filter Editor</td>
<td>140</td>
</tr>
<tr>
<td>5.6</td>
<td>Design Example</td>
<td>146</td>
</tr>
<tr>
<td>5.7</td>
<td>Summary</td>
<td>150</td>
</tr>
<tr>
<td>6</td>
<td>Scheduling View</td>
<td>153</td>
</tr>
<tr>
<td>6.1</td>
<td>Requirements</td>
<td>154</td>
</tr>
<tr>
<td>6.2</td>
<td>Graphical Representation</td>
<td>156</td>
</tr>
<tr>
<td>6.3</td>
<td>Summary</td>
<td>166</td>
</tr>
<tr>
<td>7</td>
<td>Example</td>
<td>169</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary of Example</td>
<td>177</td>
</tr>
<tr>
<td>8</td>
<td>Evaluation and Discussion of VERTIPH</td>
<td>179</td>
</tr>
<tr>
<td>8.1</td>
<td>Paper Based User Evaluation of VERTIPH</td>
<td>179</td>
</tr>
<tr>
<td>8.2</td>
<td>User Evaluation of VERTIPH</td>
<td>182</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Screening of Participants</td>
<td>183</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Introduction to VERTIPH</td>
<td>184</td>
</tr>
<tr>
<td>8.2.3</td>
<td>Evaluation Tasks</td>
<td>185</td>
</tr>
<tr>
<td>8.2.4</td>
<td>Analysis of the Questionnaires</td>
<td>186</td>
</tr>
<tr>
<td>8.2.5</td>
<td>Summary of User Evaluations</td>
<td>193</td>
</tr>
<tr>
<td>8.3</td>
<td>Cognitive Dimensions Analysis of VERTIPH</td>
<td>195</td>
</tr>
<tr>
<td>8.3.1</td>
<td>CD 1: Abstraction Gradient</td>
<td>196</td>
</tr>
<tr>
<td>8.3.2</td>
<td>CD 2: Closeness of Mapping</td>
<td>197</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1.1: Block representation of a FPGA layout........................................................4
Figure 1.2: Common parts of a block of Configurable Logic ........................................5
Figure 1.3: Image processing pyramid ......................................................................6
Figure 1.4: Temporal parallelism creates a pipeline ...................................................7
Figure 1.5: Spatial parallelism same operations on different data ................................7
Figure 1.6: A hybrid of temporal and spatial parallelism .............................................7
Figure 1.7: Logical flow of instructions in Handel-C................................................10
Figure 2.1: Hosted configuration...............................................................................21
Figure 2.2: Stand alone configuration.........................................................................21
Figure 2.3: Conversion from spatial parallelism to temporal based on a stream processing model.................................................................22
Figure 2.4: Random access processing ......................................................................22
Figure 2.5: Hybrid of stream and random access processing ....................................23
Figure 2.6: FPGA design cycle (Bailey, 2007).............................................................25
Figure 2.7: Explicit design cycle ...............................................................................26
Figure 2.8: Distorted captured image of a regular image on the left and desired image on the right ........................................................................................................28
Figure 2.9: System diagram ......................................................................................30
Figure 2.10: Pipeline for coordinate calculation .........................................................32
Figure 2.11: Image and its histogram .........................................................................35
Figure 2.12: Timing of processes and resources used for a streamed histogram function.................................................................................................................36
Figure 2.13: Block diagram of tracking system ...........................................................38
Figure 2.14: Representation within a LUT element ....................................................41
Figure 2.15: Result of colour conversion then LUT.....................................................42
Figure 2.16: A label is assigned to the current pixel based on already processed neighbours .................................................................45
Figure 2.17: Basic architecture of the single pass algorithm................................. 46
Figure 2.18: Architecture of the single pass algorithm............................................. 47
Figure 2.19: Improved algorithm block diagram...................................................... 48
Figure 2.20: Window Filter Structure....................................................................... 51
Figure 2.21: Logical flow of instructions, and time to run........................................ 60
Figure 2.22: Example of IP-Core based design, OpShop algorithm for Abingdon cross benchmark showing blocks, their parameters and their effects. (Ngan, 1992) ........................................ 67
Figure 2.23: TRAIPSE example for a three-level tree of stages constructed using face image processing operations from (Cinque et al., 2007)........................................ 70
Figure 2.24: Example of Gates notation in Labview, from Green and Petre (Green and Petre, 1992)......................................................................................................................... 76
Figure 3.1: Sequential, parallel, and pipelined arrangements of modules in a design.................. 86
Figure 3.2: Pipeline representation, for a four stage pipeline........................................ 87
Figure 4.1: Overview of the architecture view............................................................. 96
Figure 4.2: Architectural block with 2 input terminals and one output terminal ...... 96
Figure 4.3: Architecture view with blocks, terminals and wires................................. 97
Figure 4.4: Junction box ......................................................................................... 99
Figure 4.5: Type editor .......................................................................................... 100
Figure 4.6: Parts of Basic type panel....................................................................... 101
Figure 4.7: Type editor panel showing editable controls............................................ 102
Figure 4.8: Type editor panel normal view .............................................................. 102
Figure 4.9: 7 bit integer ......................................................................................... 102
Figure 4.10: Signed 7 bit integer ............................................................................ 103
Figure 4.11: Unsigned 7 bit fixed point with binary point at 4.................................. 103
Figure 4.12: Signed 7 bit fixed point with binary point at 4...................................... 103
Figure 5.1: Hierarchy of the Architectural and Computational views.......................... 107
Figure 5.2: Process representations: (a) Sequential, (b) Parallel............................... 110
Figure 5.3: Sequential flow of operations ................................................................. 112
Figure 5.4: Mixed parallel and sequential operations................................................ 113
Figure 5.5: Sequential based pipelined operations .................................................... 114
Figure 5.6: Parallel pipeline operations ..................................................................... 115
Figure 5.7: Graphical representation of the pipeline ............................................... 117
Figure 5.8: Two phase shared hardware example...................................................... 118
Figure 5.9: How can be hardware shared................................................................... 119
Figure 5.10: FPGA system architecture..................................................................... 119
Figure 5.11: Detailed FPGA view with different clock domains ......................... 120
Figure 5.12: In the Diagonal Gantt each pipeline stage is translated across and down
one space from the previous stage. ................................................................. 123
Figure 5.13: The Staggered Gantt illustrates that the pipeline is active on successive
clock pulses......................................................................................................... 124
Figure 5.14: The Staggered Gantt with conditional branching. ......................... 124
Figure 5.15: The Sequential Pipeline view uses coloured bars to show the “extent” of
each pixel. Here data arrives every clock cycle................................................ 126
Figure 5.16: The user drags the pointer one clock cycle to the right to design a
pipeline in which data arrives every second pipeline clock cycle. The coloured
regions denoting the amount of processing that occurs in a single pixel clock
cycle automatically double in width................................................................. 126
Figure 5.17: The Sequential Pipeline with Staggered Bars clarifies the relationship
between the pipeline clock and the pixel interarrival delay: The top view shows
data arriving every pipeline clock cycle, whereas in the bottom view, the control
has been dragged to the right to indicate that data arrives every second pipeline
clock cycle........................................................................................................... 127
Figure 5.18: The Staggered Sequential Pipeline with Detailed Bars shows how the
pipeline operates on successive pixels ............................................................. 128
Figure 5.19: Data explicit pipeline view for single phase operations ....................... 129
Figure 5.20: Single phase pipeline representation, real example of the Data Explicit
Pipeline representation...................................................................................... 129
Figure 5.21: Two phase explicit pipeline view ......................................................... 130
Figure 5.22: Three phase explicit pipeline view...................................................... 130
Figure 5.23: Conditional branch selection (from (Nassi and Shneiderman, 1973)).... 133
Figure 6.4: Extended state machine editor, Vertical Blanking selected. The three concurrent processors associated with the state are shown in the lower split screen.

Figure 6.5: Three pipeline controls

Figure 6.6: Whens controlling architecture nodes in connected components analysis

Figure 6.7: State machine, showing when for Connection

Figure 7.1: Root architectural view for Barrel distortion

Figure 7.2: a generic 10-bit type used for correction factor and other internal variables

Figure 7.3: Control data type

Figure 7.4: Coordinate type

Figure 7.5: Input junction box for distortion correction block

Figure 7.6: Output junction box for distortion correction block

Figure 7.7: Distortion correction computational node view

Figure 7.8: Row update expression view

Figure 7.9: Pipeline controlled by if else

Figure 8.1: Junction box view with RGB components

Figure 8.2: Trade-offs adapted from (Green, 1996)

Figure 8.3: Architectural and Computational nodes showing terminals

Figure 8.4: Type editor

Figure 8.5: A Handel-C pipeline with an error involving stages 3, 4, and 5

Figure 8.6: Graphical representation of the pipeline

Figure 8.7: VERTIPH representation of the pipeline

Figure 8.8: Nested if-else

Figure 11.1: Architectural and Computational views

Figure 11.2: Sequential, parallel, and pipelined arrangements of modules in a design

Figure 11.3 New Project
Figure 11.4 Root ......................................................................................................... 248
Figure 11.5 New Node ................................................................................................ 248
Figure 11.6 Name ....................................................................................................... 248
Figure 11.7 Pin ........................................................................................................... 248
Figure 11.8 Type ........................................................................................................ 249
Figure 11.9 Type Basic ............................................................................................. 249
Figure 11.10 Type complex ...................................................................................... 250
Figure 11.11 Mouse over .......................................................................................... 250
Figure 11.12 Connection .......................................................................................... 250
Figure 11.13 Junction Box ....................................................................................... 251
Figure 11.14 Junction Box 2 .................................................................................... 251
Figure 11.15 Connections ....................................................................................... 252
Figure 11.16 Connections ....................................................................................... 252
Figure 11.17 Menu .................................................................................................... 252
Figure 11.18 Dialog .................................................................................................. 253
Figure 11.19 Open as ............................................................................................... 253
Figure 11.20 Change ................................................................................................ 254
Figure 11.21 Comp View .......................................................................................... 254
Figure 11.22 Add Menu ........................................................................................... 255
Figure 11.23 Operations .......................................................................................... 255
Figure 11.24 Pipeline ............................................................................................... 255
Figure 11.25 Pipeline 2 ........................................................................................... 256
Figure 11.26 Pipeline two phase ............................................................................. 256
Figure 11.27 Node .................................................................................................... 256
Figure 11.28 Controls ............................................................................................... 257
Figure 11.29 Controls 2 ........................................................................................... 257
Figure 11.30 Controls 3 .......................................................................................... 258
Thesis Structure

Introduction

Image processing, FPGAs and Visual Languages

Requirements and Overview of VERTIPH

Architecture

Computational

Schedule

Example

Evaluation

Conclusions and Future work