

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.

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

2.5	Languages for Image Processing.....	65
2.6	Image Processing Visual Tools and Languages.....	66
2.7	Visual Languages.....	70
2.7.1	Visual Hardware Languages.....	74
2.7.2	Concurrent Visual Languages	74
2.7.3	Criticism and Evaluation of Visual Languages.....	75
2.8	Summary of Languages	76
3	Requirements Analysis and Overview of VERTIPH	79
3.1	Three Graphical Representations	81
3.1.1	Architecture Graphical Representation	81
3.1.2	Computational Graphical Representation	84
3.1.3	Scheduling Graphical Representation.....	88
3.2	Tying it back together.....	89
4	Architectural view	93
4.1	The Design of the Architectural View.....	95
4.2	Architecture Blocks	96
4.2.1	Terminals.....	97
4.2.2	Junction Boxes	98
4.2.3	Data Types.....	99
4.3	Summary	103
5	Computational View.....	107
5.1	Operations (expression blocks).....	109
5.2	Pipelines	111
5.2.1	Multiphase Pipeline Representation.....	119
5.3	Pipeline Visualisations	122
5.3.1	The Diagonal Gantt	123
5.3.2	The Staggered Gantt.....	124
5.3.3	The Sequential Pipeline.....	125

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

8.3.3	CD 3: Consistency	199
8.3.4	CD 4: Diffuseness / Terseness	200
8.3.5	CD 5: Error-proneness	201
8.3.6	CD 6: Hard Mental Operations (HMO)	202
8.3.7	CD 7: Hidden Dependencies	206
8.3.8	CD 8: Premature Commitment.....	207
8.3.9	CD 9: Progressive Evaluation	208
8.3.10	CD 10: Role-expressiveness	209
8.3.11	CD 11: Secondary Notation and Escape from Formalism.....	210
8.3.12	CD 12: Viscosity	210
8.3.13	CD 13: Visibility	212
8.3.14	CD 14: Juxtaposability.....	213
8.3.15	CD 15: Specificity	213
8.3.16	CD 16: Synopsie (originally “grokkiness”).....	214
8.3.17	CD 17: Free Rides.....	214
8.3.18	CD 18: Unevenness.....	215
8.3.19	CD 19: Useful Awkwardness.....	215
8.3.20	CD 20: Permissiveness	216
8.3.21	Cognitive Dimensions Summary.....	217
8.4	Discussion	221
9	The Final Countdown.....	225
9.1	Conclusions	225
9.2	Future Work	230
10	Appendix I: Paper based evaluation.....	234
10.1	VERTIPH paper based user evaluation.....	234
10.2	Question	238
10.2.1	Experience questions	238
10.2.2	Questions on VERTIPH.....	238

11	Appendix II: Evaluation of prototype.....	244
11.1	VERTIPH user evaluation questionnaire.....	244
11.2	A short introduction to VERTIPH.....	245
11.3	Getting Started	247
11.4	Activities	259
11.4.1	Activity 1	259
11.4.2	Activity 2.....	263
11.4.3	Activity 3.....	265
11.4.4	Activity 4	268
11.5	Screening Questions.....	269
11.6	Question after introduction.....	271
11.6.1	Questionnaire following evaluation tasks	272
11.6.2	Questions about the first task	272
11.6.3	Questions about the second task.....	272
11.6.4	Questions related to both tasks.....	273
12	References	280

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: TRAIPISE 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 5.24: 'F' based <code>if-else</code>	134
Figure 5.25: While or For construct, from (Nassi and Shneiderman, 1973).....	135
Figure 5.26: While loop	135
Figure 5.27: Until loop	135
Figure 5.28: Path selection with control based on multiplexors	137
Figure 5.29: Data path for a true condition	137
Figure 5.30: Data path for a false condition	137
Figure 5.31: If-else based conditional branching control.....	138
Figure 5.32: Fixed-point number line with bit selection operations	139
Figure 5.33: Main parts of a filter editor	140
Figure 5.34: Filter overlaid for edge pixels of an image.....	141
Figure 5.35: Filter architectures.....	142
Figure 5.36: Where pixels are stuffed into	144
Figure 5.37: Mirroring of pixels	144
Figure 5.38: A complex expression	146
Figure 5.39: Converted to a sequential design	146
Figure 5.40: Conversion to a pipeline	147
Figure 5.41: Pipelined Operation	147
Figure 5.42: Two phase pipeline	148
Figure 5.43: If else.....	148
Figure 5.44: Pipelined if else	149
Figure 5.45: Control of pipelined if else	149
Figure 5.46: Two phase pipelined if else	150
Figure 6.1: Linear epoch-based graphical representation for a histogram algorithm	157
Figure 6.2: Circular Epoch-based representation.....	158
Figure 6.3: The three process representations: (i) Sequential, (ii) Parallel, (iii) Pipelined.....	160

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.....	161
Figure 6.5: Three pipeline controls	164
Figure 6.6: <i>Whens</i> controlling architecture nodes in connected components analysis	165
Figure 6.7: State machine, showing <i>when</i> for Connection.....	165
Figure 7.1: Root architectural view for Barrel distortion	169
Figure 7.2: a generic 10-bit type used for correction factor and other internal variables.....	170
Figure 7.3: Control data type.....	171
Figure 7.4:Coordinate type.....	171
Figure 7.5: Input junction box for distortion correction block	172
Figure 7.6: Output junction box for distortion correction block.....	173
Figure 7.7: Distortion correction computational node view	173
Figure 7.8: Row update expression view.....	175
Figure 7.9: Pipeline controlled by if else	176
Figure 8.1: Junction box view with RGB components	189
Figure 8.2:Trade-offs adapted from (Green, 1996).....	195
Figure 8.3: Architectural and Computational nodes showing terminals.....	200
Figure 8.4: Type editor	200
Figure 8.5; A Handel-C pipeline with an error involving stages 3, 4, and 5.....	204
Figure 8.6: Graphical representation of the pipeline.....	204
Figure 8.7: VERTIPH representation of the pipeline.....	205
Figure 8.8: Nested <i>if-else</i>	212
Figure 11.1: Architectural and Computational views.....	246
Figure 11.2: Sequential, parallel, and pipelined arrangements of modules in a design	247
Figure 11.3 New Project.....	247

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



