

# Contents

<b>Foreword</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Contributions . . . . .	8
1.2 Publications . . . . .	10
<b>2 Foundations &amp; Terminology</b>	<b>11</b>
2.1 Basic Block . . . . .	11
2.2 Control Flow Graph (CFG) . . . . .	11
2.3 Dominance and Postdominance . . . . .	12
2.4 Loops . . . . .	13
2.5 Static Single Assignment (SSA) Form . . . . .	14
2.5.1 LCSSA Form (LCSSA) . . . . .	15
2.6 Control Dependence . . . . .	15
2.7 Live Values . . . . .	15
2.8 Register Pressure . . . . .	16
2.9 LLVM . . . . .	16
2.9.1 Intermediate Representation (IR) . . . . .	17
2.9.2 Data Types . . . . .	18
2.9.3 Important Instructions . . . . .	18
2.10 Single Instruction, Multiple Data (SIMD) . . . . .	21
<b>3 Overview</b>	<b>23</b>
3.1 Whole-Function Vectorization (WfV) . . . . .	23
3.2 Algorithmic Challenges . . . . .	24
3.3 Performance Issues of Vectorization . . . . .	25
<b>4 Related Work</b>	<b>31</b>
4.1 Classic Loop Vectorization . . . . .	31
4.2 Superword Level Parallelism (SLP) . . . . .	32
4.3 Outer Loop Vectorization (OLV) . . . . .	32
4.4 Auto-Vectorizing Languages . . . . .	33
4.4.1 OpenCL and CUDA . . . . .	34

4.5	SIMD Property Analyses . . . . .	35
4.6	Dynamic Variants . . . . .	36
4.7	Summary . . . . .	38
<b>5</b>	<b>SIMD Property Analyses</b> . . . . .	<b>39</b>
5.1	Program Representation . . . . .	40
5.2	SIMD Properties . . . . .	41
5.2.1	Uniform & Varying Values . . . . .	41
5.2.2	Consecutivity & Alignment . . . . .	42
5.2.3	Sequential & Non-Vectorizable Operations . . . . .	43
5.2.4	All-Instances-Active Operations . . . . .	43
5.2.5	Divergent Loops . . . . .	44
5.2.6	Divergence-Causing Blocks & Rewire Targets . . . . .	44
5.3	Analysis Framework . . . . .	45
5.4	Operational Semantics . . . . .	46
5.4.1	Lifting to Vector Semantics . . . . .	47
5.5	Collecting Semantics . . . . .	48
5.6	Vectorization Analysis . . . . .	49
5.6.1	Tracked Information . . . . .	49
5.6.2	Initial State . . . . .	51
5.6.3	Instance Identifier . . . . .	53
5.6.4	Constants . . . . .	54
5.6.5	Phi Functions . . . . .	54
5.6.6	Memory Operations . . . . .	57
5.6.7	Calls . . . . .	60
5.6.8	Cast Operations . . . . .	61
5.6.9	Arithmetic and Other Instructions . . . . .	62
5.6.10	Branch Operation . . . . .	65
5.6.11	Update Function for All-Active Program Points . . . . .	65
5.6.12	Update Function for Divergent Loops . . . . .	66
5.7	Soundness . . . . .	67
5.7.1	Local Consistency . . . . .	67
5.8	Improving Precision with an SMT Solver . . . . .	72
5.8.1	Expression Trees of Address Computations . . . . .	72
5.8.2	Translation to Presburger Arithmetic . . . . .	73
5.8.3	From SMT Solving Results to Code . . . . .	76
5.9	Rewire Target Analysis . . . . .	78
5.9.1	Running Example . . . . .	79
5.9.2	Loop Criteria . . . . .	80
5.9.3	Formal Definition . . . . .	81

---

5.9.4	Application in Partial CFG Linearization . . . . .	82
<b>6</b>	<b>Whole-Function Vectorization</b>	<b>85</b>
6.1	Mask Generation . . . . .	85
6.1.1	Loop Masks . . . . .	90
6.1.2	Running Example . . . . .	92
6.1.3	Alternative for Exits Leaving Multiple Loops . . . . .	93
6.2	Select Generation . . . . .	95
6.2.1	Loop Blending . . . . .	96
6.2.2	Blending of Optional Loop Exit Results . . . . .	98
6.2.3	Running Example . . . . .	100
6.3	Partial CFG Linearization . . . . .	102
6.3.1	Running Example . . . . .	105
6.3.2	Clusters of Divergence-Causing Blocks . . . . .	106
6.3.3	Rewire Target Block Scheduling . . . . .	107
6.3.4	Computation of New Outgoing Edges . . . . .	111
6.3.5	Linearization . . . . .	112
6.3.6	Repairing SSA Form . . . . .	120
6.3.7	Branch Fusion . . . . .	120
6.4	Instruction Vectorization . . . . .	121
6.4.1	Broadcasting of Uniform Values . . . . .	121
6.4.2	Consecutive Value Optimization . . . . .	122
6.4.3	Merging of Sequential Results . . . . .	122
6.4.4	Duplication of Non-Vectorizable Operations . . . . .	123
6.4.5	Pumped Vectorization . . . . .	124
6.5	Extension for Irreducible Control Flow . . . . .	124
<b>7</b>	<b>Dynamic Code Variants</b>	<b>127</b>
7.1	Uniform Values and Control Flow . . . . .	128
7.2	Consecutive Memory Access Operations . . . . .	130
7.3	Switching to Scalar Code . . . . .	131
7.4	WFV-SLP . . . . .	131
7.5	Instance Reorganization . . . . .	134
7.6	Skipping All-Inactive Paths . . . . .	138
<b>8</b>	<b>Evaluation</b>	<b>141</b>
8.1	Benchmark Setup and Reproducibility . . . . .	141
8.2	WFVOpenCL . . . . .	141
8.2.1	WFV Integration . . . . .	143
8.2.2	Runtime Callbacks . . . . .	143

---

8.2.3	Continuation-Based Barrier Synchronization . . . . .	145
8.2.4	Experimental Evaluation . . . . .	147
8.2.5	SMT-Based Memory Access Optimization . . . . .	154
8.3	AnySL: Efficient and Portable Shading for Ray Tracing . . . . .	156
8.3.1	WFV Integration . . . . .	157
8.3.2	Experimental Evaluation . . . . .	158
8.4	Noise: On-Demand Loop Vectorization . . . . .	160
8.4.1	WFV Integration . . . . .	162
8.4.2	Loops with Loop-Carried Dependencies . . . . .	163
8.4.3	Experimental Evaluation . . . . .	166
<b>9</b>	<b>Conclusion</b>	<b>171</b>
<b>10</b>	<b>Outlook</b>	<b>173</b>