PART 1 FOUNDATIONS OF OPENCL PROGRAMMING ..........1

Introducing OpenCL 3

1.1 The dawn of OpenCL 4
1.2 Why OpenCL? 5

Portability 6 • Standardized vector processing 6 • Parallel programming 7

1.3 Analogy: OpenCL processing and a game of cards 8
1.4 A first look at an OpenCL application 10
1.5 The OpenCL standard and extensions 13
1.6 Frameworks and software development kits (SDKs) 14
1.7 Summary 14
2 Host programming: fundamental data structures 16

2.1 Primitive data types 17

2.2 Accessing platforms 18

Creating platform structures 18 • Obtaining platform information 19 • Code example: testing platform extensions 20

2.3 Accessing installed devices 22

Creating device structures 22 • Obtaining device information 23 • Code example: testing device extensions 24

2.4 Managing devices with contexts 25

Creating contexts 26 • Obtaining context information 28 • Contexts and the reference count 28 • Code example: checking a context’s reference count 29

2.5 Storing device code in programs 30

Creating programs 30 • Building programs 31 • Obtaining program information 33 • Code example: building a program from multiple source files 35

2.6 Packaging functions in kernels 36

Creating kernels 36 • Obtaining kernel information 37 • Code example: obtaining kernel information 38

2.7 Collecting kernels in a command queue 39

Creating command queues 40 • Enqueuing kernel execution commands 40

2.8 Summary 41

3 Host programming: data transfer and partitioning 43

3.1 Setting kernel arguments 44

3.2 Buffer objects 45

Allocating buffer objects 45 • Creating subbuffer objects 47

3.3 Image objects 48

Creating image objects 48 • Obtaining information about image objects 51

3.4 Obtaining information about buffer objects 52

3.5 Memory object transfer commands 54

Read/write data transfer 54 • Mapping memory objects 58 • Copying data between memory objects 59
3.6 Data partitioning 62
   Loops and work-items 63 • Work sizes and offsets 64 • A simple one-dimensional example 65 • Work-groups and compute units 65

3.7 Summary 67

4 Kernel programming: data types and device memory 68
   4.1 Introducing kernel coding 69
   4.2 Scalar data types 70
      Accessing the double data type 71 • Byte order 72
   4.3 Floating-point computing 73
      The float data type 73 • The double data type 74 • The half data type 75 • Checking IEEE-754 compliance 76
   4.4 Vector data types 77
      Preferred vector widths 79 • Initializing vectors 80 • Reading and modifying vector components 80 • Endianness and memory access 84
   4.5 The OpenCL device model 85
      Device model analogy part 1: math students in school 85 • Device model analogy part 2: work-items in a device 87 • Address spaces in code 88 • Memory alignment 90
   4.6 Local and private kernel arguments 90
      Local arguments 91 • Private arguments 91
   4.7 Summary 93

5 Kernel programming: operators and functions 94
   5.1 Operators 95
   5.2 Work-item and work-group functions 97
      Dimensions and work-items 98 • Work-groups 99 • An example application 100
   5.3 Data transfer operations 101
      Loading and storing data of the same type 101 • Loading vectors from a scalar array 101 • Storing vectors to a scalar array 102
   5.4 Floating-point functions 103
      Arithmetic and rounding functions 103 • Comparison functions 105 • Exponential and logarithmic functions 106 • Trigonometric functions 106 • Miscellaneous floating-point functions 108
8 Development with C++ 167
8.1 Preliminary concerns 168
   Vectors and strings 168 • Exceptions 169
8.2 Creating kernels 170
   Platforms, devices, and contexts 170 • Programs and kernels 173
8.3 Kernel arguments and memory objects 176
   Memory objects 177 • General data arguments 181 • Local space arguments 182
8.4 Command queues 183
   Creating CommandQueue objects 183 • Enqueuing kernel-execution commands 183 • Read/write commands 185
   Memory mapping and copy commands 187
8.5 Event processing 189
   Host notification 189 • Command synchronization 191
   Profiling events 192 • Additional event functions 193
8.6 Summary 194

9 Development with Java and Python 196
9.1 Aparapi 197
   Aparapi installation 198 • The Kernel class 198 • Work-items and work-groups 200
9.2 JavaCL 201
   JavaCL installation 202 • Overview of JavaCL development 202
   Creating kernels with JavaCL 203 • Setting arguments and enqueuing commands 206
9.3 PyOpenCL 210
   PyOpenCL installation and licensing 210 • Overview of PyOpenCL development 211 • Creating kernels with PyOpenCL 212 • Setting arguments and executing kernels 215
9.4 Summary 219

10 General coding principles 221
10.1 Global size and local size 222
   Finding the maximum work-group size 223 • Testing kernels and devices 224
10.2 Numerical reduction 225
   OpenGL reduction 226 • Improving reduction speed with vectors 228
10.3 Synchronizing work-groups 230
10.4 Ten tips for high-performance kernels 231
10.5 Summary 233

PART 2 CODING PRACTICAL ALGORITHMS IN OPENCL 235

11 Reduction and sorting 237
11.1 MapReduce 238
  Introduction to MapReduce 238 • MapReduce and  
  OpenCL 240 • MapReduce example: searching for text 242
11.2 The bitonic sort 244
  Understanding the bitonic sort 244 • Implementing the bitonic sort  
  in OpenCL 247
11.3 The radix sort 254
  Understanding the radix sort 254 • Implementing the radix sort  
  with vectors 254
11.4 Summary 256

12 Matrices and QR decomposition 258
12.1 Matrix transposition 259
  Introduction to matrices 259 • Theory and implementation of  
  matrix transposition 260
12.2 Matrix multiplication 262
  The theory of matrix multiplication 262 • Implementing matrix  
  multiplication in OpenCL 263
12.3 The Householder transformation 265
  Vector projection 265 • Vector reflection 266 • Outer products  
  and Householder matrices 267 • Vector reflection in  
  OpenCL 269
12.4 The QR decomposition 269
  Finding the Householder vectors and R 270 • Finding the  
  Householder matrices and Q 272 • Implementing QR  
  decomposition in OpenCL 273
12.5 Summary 276

13 Sparse matrices 278
13.1 Differential equations and sparse matrices 279
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.2 Sparse matrix storage and the Harwell-Boeing collection</td>
<td>280</td>
</tr>
<tr>
<td>Introducing the Harwell-Boeing collection</td>
<td>281</td>
</tr>
<tr>
<td>Accessing data in Matrix Market files</td>
<td>281</td>
</tr>
<tr>
<td>13.3 The method of steepest descent</td>
<td>285</td>
</tr>
<tr>
<td>Positive-definite matrices</td>
<td>285</td>
</tr>
<tr>
<td>Theory of the method of steepest descent</td>
<td>286</td>
</tr>
<tr>
<td>Implementing SD in OpenCL</td>
<td>288</td>
</tr>
<tr>
<td>13.4 The conjugate gradient method</td>
<td>289</td>
</tr>
<tr>
<td>Orthogonalization and conjugacy</td>
<td>289</td>
</tr>
<tr>
<td>The conjugate gradient method</td>
<td>291</td>
</tr>
<tr>
<td>13.5 Summary</td>
<td>293</td>
</tr>
<tr>
<td>14 Signal processing and the fast Fourier transform</td>
<td>295</td>
</tr>
<tr>
<td>14.1 Introducing frequency analysis</td>
<td>296</td>
</tr>
<tr>
<td>14.2 The discrete Fourier transform</td>
<td>298</td>
</tr>
<tr>
<td>Theory behind the DFT</td>
<td>298</td>
</tr>
<tr>
<td>OpenCL and the DFT</td>
<td>305</td>
</tr>
<tr>
<td>14.3 The fast Fourier transform</td>
<td>306</td>
</tr>
<tr>
<td>Three properties of the DFT</td>
<td>306</td>
</tr>
<tr>
<td>Constructing the fast Fourier transform</td>
<td>309</td>
</tr>
<tr>
<td>Implementing the FFT with OpenCL</td>
<td>312</td>
</tr>
<tr>
<td>14.4 Summary</td>
<td>317</td>
</tr>
<tr>
<td>Part 3 Accelerating OpenGL with OpenCL</td>
<td>319</td>
</tr>
<tr>
<td>15 Combining OpenCL and OpenGL</td>
<td>321</td>
</tr>
<tr>
<td>15.1 Sharing data between OpenGL and OpenCL</td>
<td>322</td>
</tr>
<tr>
<td>Creating the OpenCL context</td>
<td>323</td>
</tr>
<tr>
<td>Sharing data between OpenGL and OpenCL</td>
<td>325</td>
</tr>
<tr>
<td>Synchronizing access to shared data</td>
<td>328</td>
</tr>
<tr>
<td>15.2 Obtaining information</td>
<td>329</td>
</tr>
<tr>
<td>Obtaining OpenGL object and texture information</td>
<td>329</td>
</tr>
<tr>
<td>Obtaining information about the OpenGL context</td>
<td>330</td>
</tr>
<tr>
<td>15.3 Basic interoperability example</td>
<td>331</td>
</tr>
<tr>
<td>Initializing OpenGL operation</td>
<td>331</td>
</tr>
<tr>
<td>Initializing OpenCL operation</td>
<td>331</td>
</tr>
<tr>
<td>Creating data objects</td>
<td>332</td>
</tr>
<tr>
<td>Executing the kernel</td>
<td>333</td>
</tr>
<tr>
<td>Rendering graphics</td>
<td>334</td>
</tr>
<tr>
<td>15.4 Interoperability and animation</td>
<td>334</td>
</tr>
<tr>
<td>Specifying vertex data</td>
<td>335</td>
</tr>
<tr>
<td>Animation and display</td>
<td>336</td>
</tr>
<tr>
<td>Executing the kernel</td>
<td>337</td>
</tr>
<tr>
<td>15.5 Summary</td>
<td>338</td>
</tr>
</tbody>
</table>
16 Textures and renderbuffers  340

16.1 Image filtering  341
   The Gaussian blur  343 • Image sharpening  344 • Image embossing  344

16.2 Filtering textures with OpenCL  345
   The init_gl function  345 • The init_cl function  345 • The configure_shared_data function  346 • The execute_kernel function  347 • The display function  348

16.3 Summary  349

appendix A Installing and using a software development kit  351
appendix B Real-time rendering with OpenGL  364
appendix C The minimalist GNU for Windows and OpenCL  398
appendix D OpenCL on mobile devices  412
index  415