1. Record Nr. UNINA9910790726103321

Autore	Banger Ravishekhar
Titolo	OpenCL programming by example / / Ravishekhar Banger, Koushik Bhattacharyya
Pubbl/distr/stampa	Birmingham : , : Packt Publishing, , 2013
ISBN	1-84969-235-1
Edizione	[1st edition]
Descrizione fisica	1 online resource (304 p.)
Collana	Community experience distilled
Altri autori (Persone)	BhattacharyyaKoushik
Disciplina	005.2/75
Soggetti	OpenCL (Computer program language) Paraelle programming (Computer science)
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes index.
Nota di contenuto	Cover; Copyright; Credits; About the Authors; About the Reviewers; www.PacktPub.com; Table of Contents; Preface; Chapter 1: Hello OpenCL; Advances in computer architecture; Different parallel programming techniques; OpenMP; MPI; OpenACC; CUDA; CUDA or OpenCL?; Renderscripts; Hybrid parallel computing model; Introduction to OpenCL; Hardware and software vendors; Advanced Micro Devices, Inc. (AMD); NVIDIA®; Intel®; ARM MaliTM GPUs; OpenCL components; An example OpenCL program; Basic software requirements; Windows; Linux; Installing and setting up an OpenCL compliant computer; Installation steps Installing OpenCL on a Linux system with an AMD graphics card Installing OpenCL on a Linux system with an NVIDIA graphics card; Installing OpenCL on a Windows system with an AMD graphics card; Installing OpenCL on a Windows system with an NVIDIA graphics card; Apple OSX; Multiple installations; Implement the SAXPY routine in OpenCL; Summary; References; Chapter 2: OpenCL Architecture; Platform model; AMD A10 5800K APUs; AMD RadeonTM HD 7870 Graphics Processor; NVIDIA® GeForce® GTC 680 GPU; Intel® IVY bridge; Platform versions; Query Platforms; Query devices; Execution model; NDRange OpenCL context OpenCL command queue; Memory model; Global memory; Constant memory; Local memory; Private memory; OpenCL ICD; What is an OpenCL ICD?; Application scaling; Summary; Chapter 3:

OpenCL Buffer Objects; Memory Objects; Creating Subbuffer objects; Histogram calculation; Algorithm; OpenCL Kernel Code; The Host Code; Reading and writing buffers; Blocking_read and Blocking_write; Rectangular or cuboidal reads; Copying buffers; Mapping buffer objects; Querying buffer objects; Undefined behavior of the cl_mem objects; Summary; Chapter 4: OpenCL Images; Creating images Image format descriptor cl_image_format Image details descriptor cl_image_desc; Passing image buffers to kernels; Samplers; Reading and writing buffers; Copying and filling images; Mapping image objects; Querying image objects; Image histogram computation; Summary; Chapter 5: OpenCL Program and Kernel Objects; Creating program objects; Creating and building program objects; OpenCL program building options; Querying program objects; Creating binary files; Offline and online compilation; SAXPY using the binary file; SPIR - Standard Portable Intermediate Representation; Creating kernel objects Setting kernel arguments Executing the kernels; Querying kernel objects; Querying kernel argument; Releasing program and kernel objects; Built-in kernels; Summary; Chapter 6: Events and Synchronization; OpenCL events and monitoring of these events; OpenCL event synchronization models; No synchronization needed; Single device in-order usage; Synchronization needed; Single device and out-of-order queue; Multiple devices and different OpenCL contexts; Multiple devices and single OpenCL context; Coarse grained synchronization; Event based or fine grained synchronization Getting information about cl_event

Sommario/riassunto

This book follows an example-driven, simplified, and practical approach to using OpenCL for general purpose GPU programming.If you are a beginner in parallel programming and would like to quickly accelerate your algorithms using OpenCL, this book is perfect for you! You will find the diverse topics and case studies in this book interesting and informative. You will only require a good knowledge of C programming for this book, and an understanding of parallel implementations will be useful, but not necessary.