Accelerating Computer Vision and More, with Hardware-Accelerated OpenCV on Heterogeneous Platforms

Aaron Kunze

Legal

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice.

All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Optimized Intel® HD Graphics P3000 only available on select models of the Intel® Xeon® processor E3 family. To learn more about Intel Xeon processors for workstation visit www.intel.com/go/workstation.

HD Graphics P4000 introduces four additional execution units, going from 8 in the HD P3000 to 12 in the HD P4000. Optimized Intel® HD Graphics P4000 only available on select models of the Intel® Xeon® processor E3-1200 v2 product family. For more information, visithttp://www.intel.com/content/www/us/en/architecture-and-technology/hdgraphics/hdgraphics-developer.html

Iris™ graphics is available on select systems. Consult your system manufacturer.

Any code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Intel product plans in this presentation do not constitute Intel plan of record product roadmaps. Please contact your Intel representative to obtain Intel's current plan of record product roadmaps.

Performance claims: Software and workloads used in performance tests may have been optimized for performance only on Intel® microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to : http://www.Intel.com/performance

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Haswell and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Intel, Iris, Iris Pro, Core, VTune and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright ©2014 Intel Corporation.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

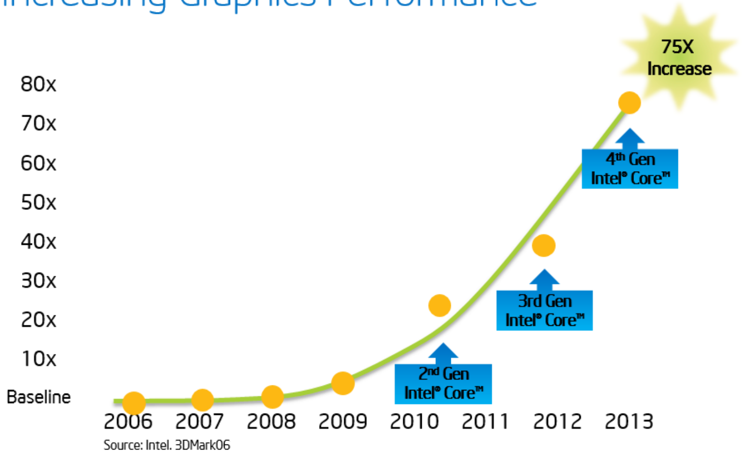
Notice revision #20110804

Agenda

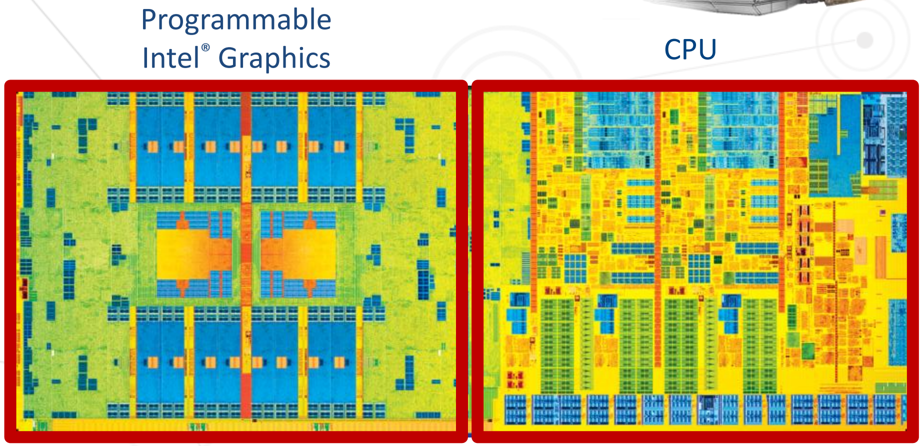
- OpenCL* on Intel® Graphics
- OpenCV 3.0 on Intel® Graphics
- Intel® Graphics Architecture Overview
- Optimization Techniques
  - Maximizing Occupancy
  - Optimizing Memory Access
  - Using Registers
  - Maximizing Computation
- Intel® VTune Support for OpenCL
- Summary / Questions

Intel® Graphics

Increasing Graphics Performance



Source: Intel. 3DMark06



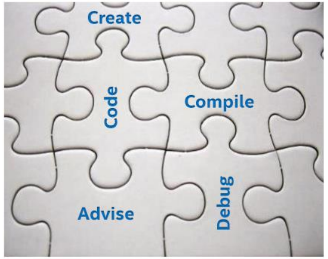
Intel® Core™ Processor code name Haswell

Programmable graphics has become a valuable resource to maximize the visual experience.

Intel® SDK for OpenCL™ Applications 2014

Development Environment

Intel® SDK for OpenCL™ Applications



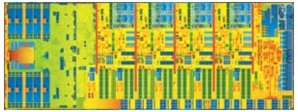
Integrated with Microsoft Visual Studio* and Eclipse*

The OpenCL™ development environment for Intel®-based platforms

Available for free

Platform

The OpenCL Advantage



4th Generation Intel® Core ™ Processor

- An open, standard, and portable API for heterogeneous computing
- Supported on Intel® CPUs, Intel® Graphics, and Intel® Xeon Phi™ coprocessor
- Designed for visual computing applications

The standard way to program Intel® HD Graphics and Intel® Iris™ Graphics family

Maximize the power of the platform. Optimize tasks with the best available compute engines.

OpenCL™ on Intel® Architecture: Success Stories

“OpenCL lets us write one line of code that will run on lots of different types of hardware”

*Eric Berdahl, Senior Engineering Manager, Adobe**
Adobe Optimizes with OpenCL* and Intel® Graphics: http://www.youtube.com/watch?v=IXdhhud5iH4

Dave Helmly | Sr. Manager
Solutions Consulting Pro Video/Audio Americas, Adobe Systems Inc.
0:12 / 2:59

“The Intel® Iris™ Pro graphics and the Intel® Core™ i7 processor are … allowing me to do all of this while the graphics and video are never stopping”

Dave Helmly, Solution Consulting Pro Video/Audio, Adobe
Adobe Premiere Pro demonstration: http://www.youtube.com/watch?v=u0J57J6Hppg

“We are very pleased that Intel is fully supporting OpenCL. We think there is a bright future for this technology.”

Michael Bryant, Director of Marketing, Sony Creative Software
Vegas* Software Family by Sony* Optimized with OpenCL* and Intel® Processor Graphics
http://www.youtube.com/watch?v=_KHVOCwTdno

Our customers report on benefits like productivity, performance, and use of open standard

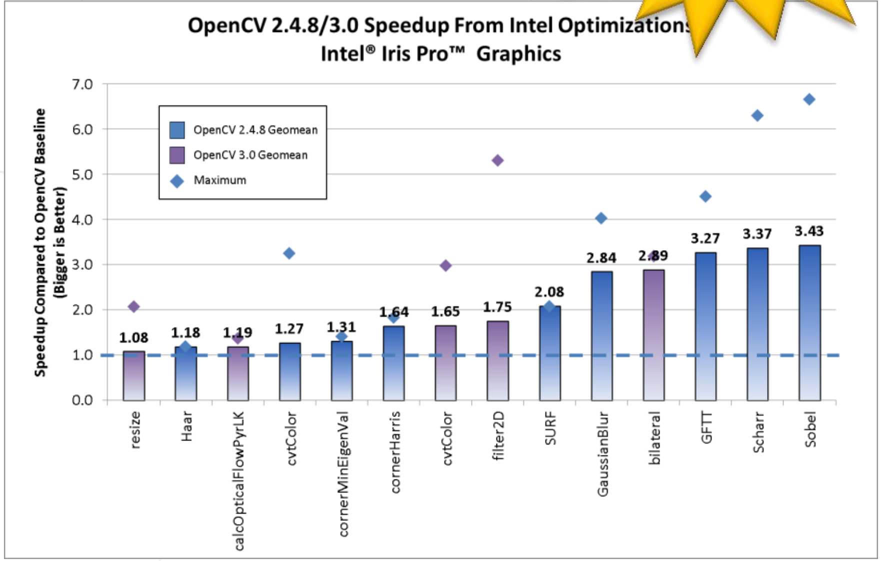
Agenda

- OpenCL* on Intel® Graphics
- **OpenCV 3.0 on Intel® Graphics**
- Intel® Graphics Architecture Overview
- Optimization Techniques
  - Maximizing Occupancy
  - Optimizing Memory Access
  - Using Registers
  - Maximizing Computation
- Intel® VTune Support for OpenCL
- Summary / Questions

OpenCV* Optimization for Intel® Graphics

- ✦ OpenCV*: the leading open-source computer vision library
- ✦ Intel is contributing optimizations for the OpenCL* code in OpenCV
- ✦ Intel optimizations delivering substantial performance improvements!
  - – Examples described throughout this talk

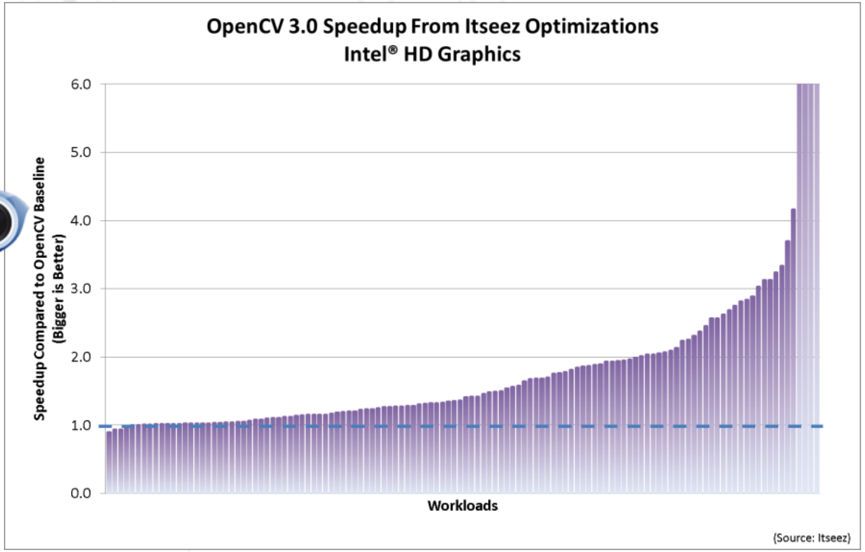
Up to 3.4X!



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance

OpenCV* Optimization for Intel® Graphics

- ✦ Intel working with Itseez* to further optimize OpenCL code in OpenCV 3.0
  - Itseez* is the official maintainer of OpenCV
  - Itseez* using optimization BKMs described in this talk



(Source: Itseez)

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance

OpenCV* Optimization for Intel® Graphics

- Upcoming OpenCV* 3.0 architecture further improves support for Intel® Graphics
  - "Transparent API" enables same code to use CPU or OpenCL*
    - Little or no code changes from existing OpenCV code
    - Code works on platforms without OpenCL using efficient CPU fallback

```
cv::UMat inMat, outMat;
vidInput >> inMat;
cv::cvtColor(inMat, outMat, cv::COLOR_RGB2GRAY);
vidOutput << outMat;
```

  - APIs operate asynchronously
  - Improved use of shared physical memory for integrated GPU performance

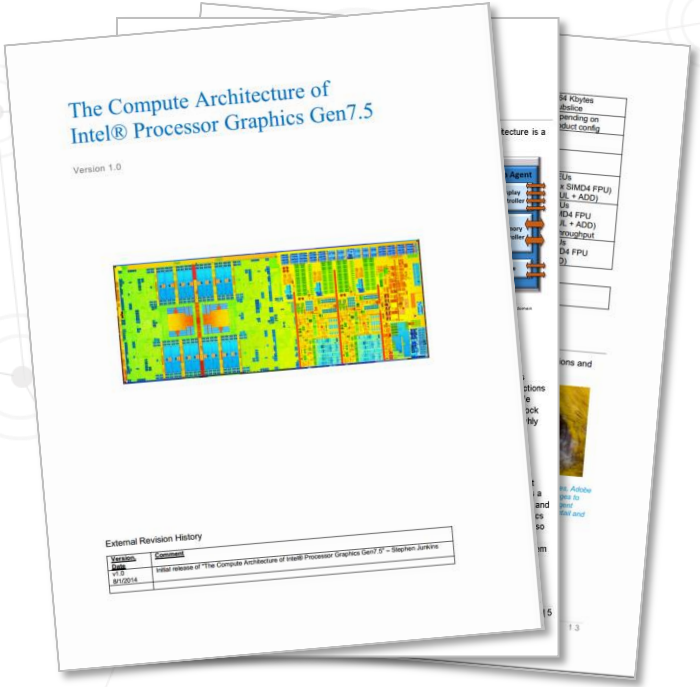
OpenCV 3.0 makes excellent use of OpenCL on Intel® Graphics!

Agenda

- OpenCL* on Intel® Graphics
- OpenCV 3.0 on Intel® Graphics
- Intel® Graphics Architecture Overview
- Optimization Techniques
  - Maximizing Occupancy
  - Optimizing Memory Access
  - Using Registers
  - Maximizing Computation
- Intel® VTune Support for OpenCL
- Summary / Questions

Intel® Graphics Architecture

- Today, our focus is on Intel® Iris™ Graphics and Intel® HD Graphics in 4th Generation Intel® Core™ Processors
  - Or, Intel® Processor Graphics Gen7.5
- For more details, see our whitepaper, The Compute Architecture of Intel® Processor Graphics Gen7.5
  - https://software.intel.com/en-us/articles/intel-graphics-developers-guides
  - Hardcopies in the back



The Compute Architecture of Intel® Processor Graphics Gen7.5

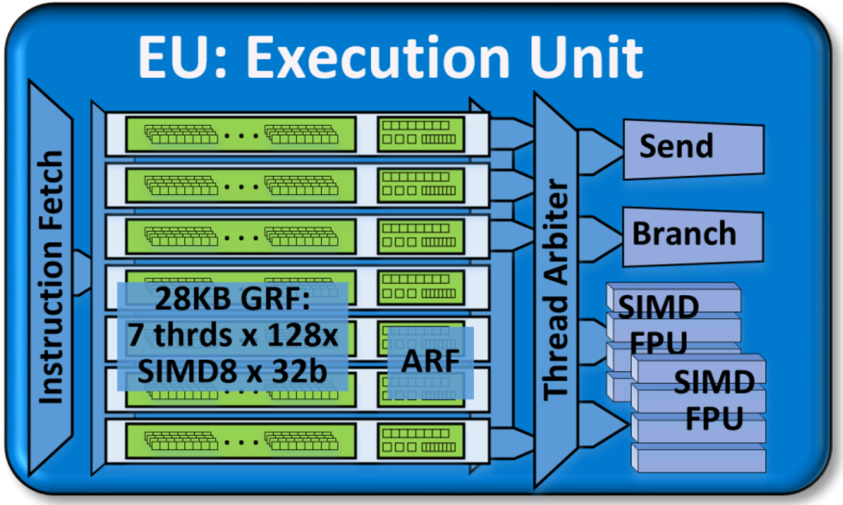
Version 1.0

External Revision History

Version, Date	Comment
v1.0 8/1/2014	Initial release of "The Compute Architecture of Intel® Processor Graphics Gen7.5" – Stephen Junkins

Intel® Graphics Architecture Building Blocks

- OpenCL* Kernels run on an Execution Unit (EU)
- Each EU is a Multi-Threaded SIMD Processor
- Up to 7 threads per EU
  - 128 x 8 x 32-bit registers per thread
  - Zero cycle thread switching
- Up to 8, 16, or 32 OpenCL work items per thread (compiler-controlled)
  - "SIMD8", "SIMD16", "SIMD32"
  - SIMD8 → More registers per work item
  - SIMD16 and SIMD32 → Higher IPC

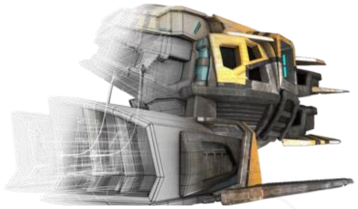


Intel® Graphics Architecture Building Blocks

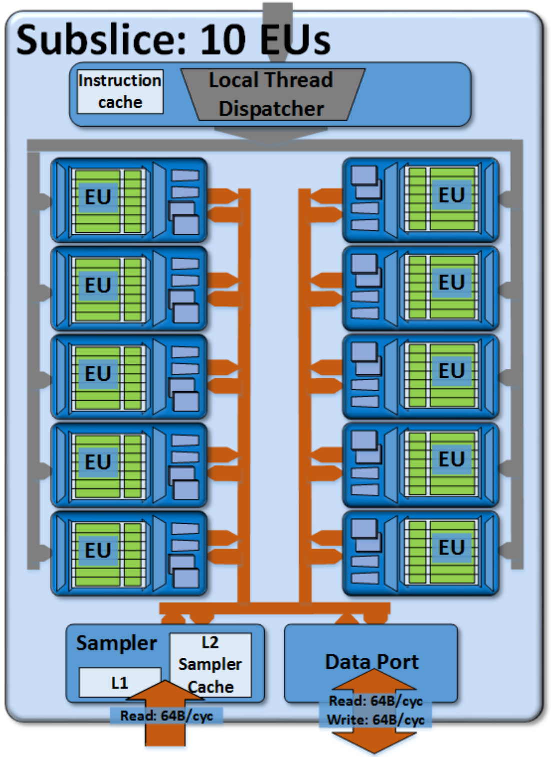
- EUs have 2 x 4-wide vector ALUs
- Second ALU has limitations:
  - Subset of instructions: **`add`**, **`mov`**, **`mad`**, **`mul`**, **`cmp`**
  - Instruction must come from another EU thread
  - Only float operands!
- Peak GFlops: #EUs x (2 x 4-wide ALUs) x (MUL + ADD) x Clock Rate

For Intel® Iris™ Pro 5200: 40 x 8 x 2 x 1.3 = 832 GFlops!
(Not counting CPU!)

Intel® Graphics Architecture Building Blocks



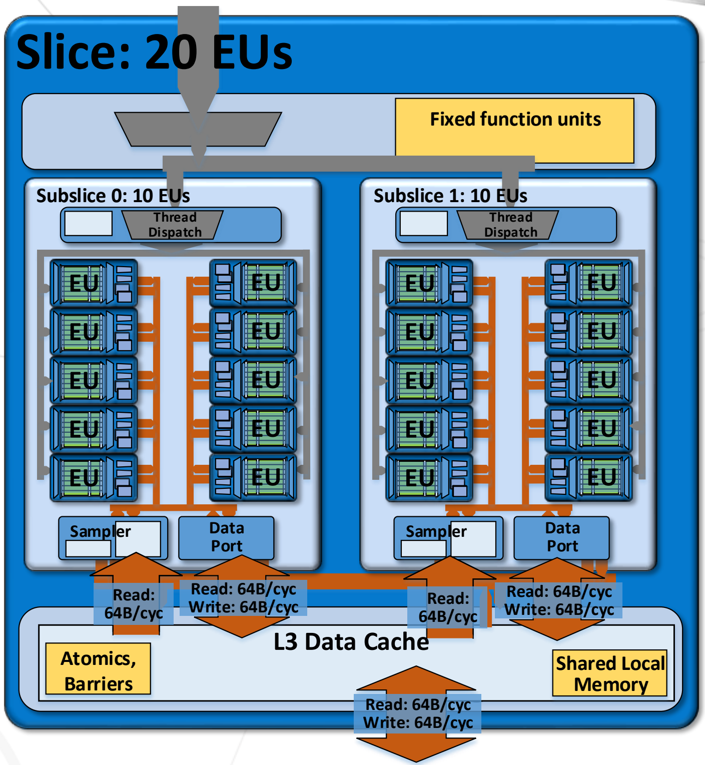
- OpenCL* Work Groups run on a Sub Slice
  - 10 EUs per Sub Slice
  - Texture Sampler (Images)
  - Data Port (Buffers)
  - Instruction and Texture Caches



OpenCL Work Groups may run on multiple EU threads, or even on multiple EUs!

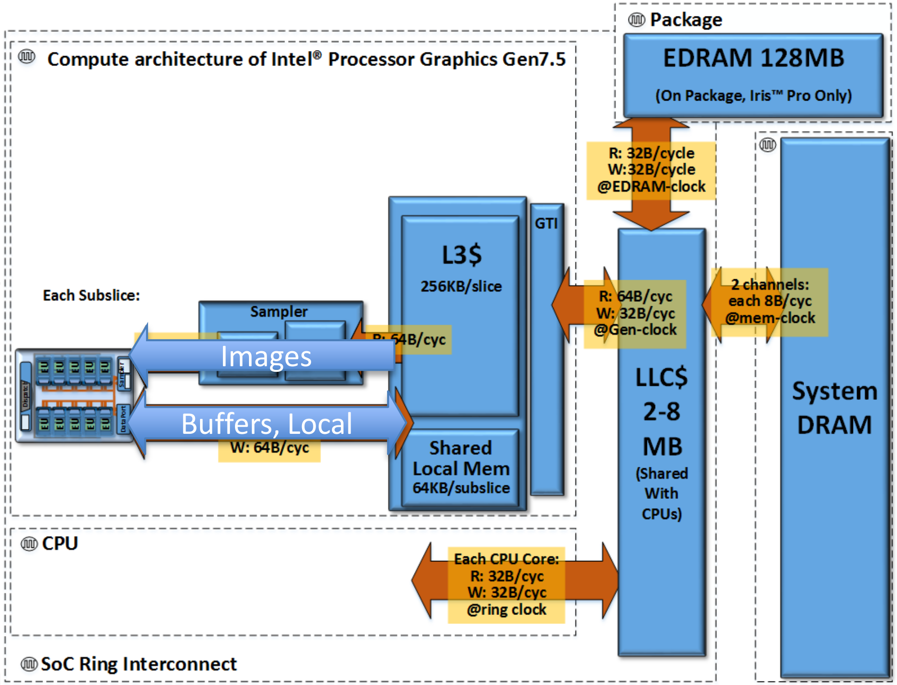
Intel® Graphics Architecture Building Blocks

- Two Sub Slices make a Slice
- Shared Resources: “Slice Common”
  - L3 Cache + Shared Local Memory
  - Barriers
- Intel® Iris™ Graphics has Two Slices
  - 2 x 2 = 4 Sub Slices
  - 4 x 10 = 40 EUs
  - Up to 40 x 7 = 280 EU threads
  - Up to 8960 OpenCL* work items in flight!



Intel® Graphics Cache Hierarchy

- Separate hierarchy for OpenCL* images and buffers
- Shared cache with CPU
- 128MB EDRAM available on Intel® Iris™ Pro Graphics devices
  - Acts as a large cache



Agenda

- OpenCL* on Intel® Graphics
- OpenCV 3.0 on Intel® Graphics
- Intel® Graphics Architecture Overview
- Optimization Techniques
  - Maximizing Occupancy
  - Optimizing Memory Access
  - Using Registers
  - Maximizing Computation
- Intel® VTune Support for OpenCL
- Summary / Questions

Maximizing Occupancy

- Occupancy is a measure of EU thread utilization
- Two primary things to consider:
  - Launch enough work items to keep EU threads busy
  - In short kernels: use short vector data types and compute multiple pixels to better amortize thread launch cost
    - For example, color conversion:

```
__global uchar* src, dst;
p = src[src_idx]     * B2Y +
    src[src_idx + 1] * G2Y +
    src[src_idx + 2] * R2Y;
dst[dst_idx] = p;
```

Before:

One pixel per work item

```
__global uchar* src_ptr, dst_ptr;
uchar16 src = vload16(0, src_ptr);
uchar4 c0 = src.s048c;
uchar4 c1 = src.s159d;
uchar4 c2 = src.s26ae;
uchar4 Y = c0 * B2Y +
           c1 * G2Y +
           c2 * R2Y;
vstore4(Y, 0, dst_ptr);
```

After:

Four pixels per work item

Occupancy Constraints

More subtle occupancy issues:

- Barriers – 16 barrier registers per sub slice
  - Limits number of in-flight work groups
  - Each work group running on a sub slice uses a barrier register (if the kernel uses barriers)
  - Can be a limiting factor for very small local work groups
  - Using a barrier? Use power-of-two workgroup sizes between 64 and 256!
- Shared Local Memory (SLM) – 64KB SLM per sub slice
  - Can be a limiting factor for kernels that use a lot of local memory
  - Using local memory? Use less than 64 bytes per work item!

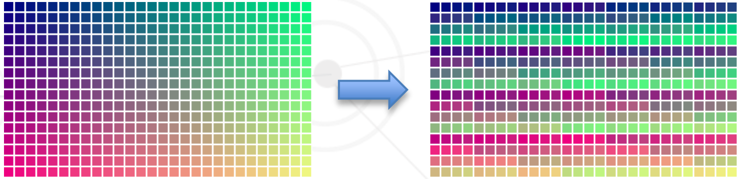
Maximize occupancy to keep EUs busy!

Optimizing Host to Device Transfers

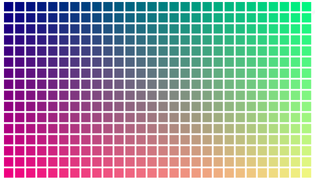
- Host (CPU) and Device (GPU) share the same physical memory
- For OpenCL* buffers:
  - Create buffer with system memory pointer and **`CL_MEM_USE_HOST_PTR`**
  - No transfer needed (zero copy)!
  - Allocate system memory aligned to a page (4096 bytes)
    - E.g., use _aligned_malloc or memalign to allocate
  - Allocate a multiple of cache line size (64 bytes)
  - Use **`clEnqueueMapBuffer()`** to access data
  - OpenCV 3.0 changes make excellent use of this feature!

Optimizing Host to Device Transfers

- For OpenCL images:
  - By default, tiled in device memory (transfer required)
  - Or, use **`cl_khr_image2d_from_buffer`** extension for zero copy!
- Texture sampler great for cases that need linear interpolation
- Used in some flavors of OpenCV resize
  - Up to 2X performance!



Default Image Creation Path

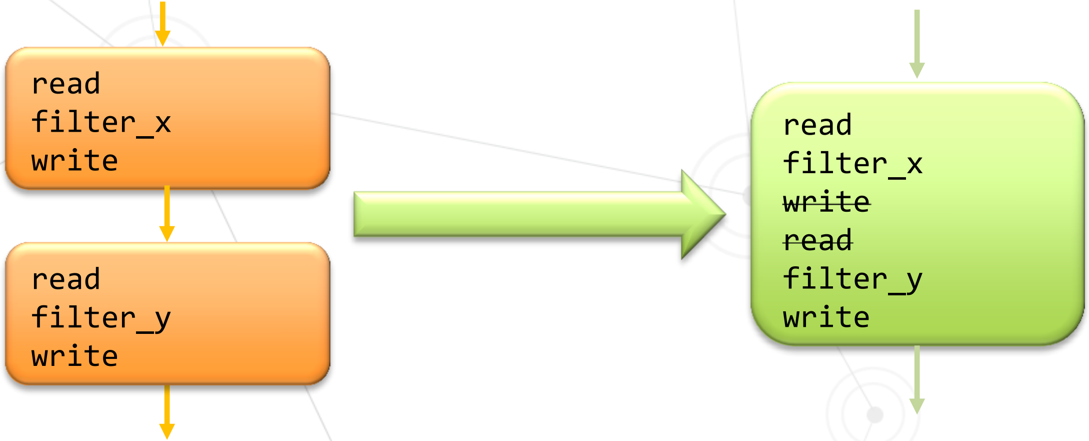


Using cl_khr_image2d_from_buffer (No copy!)

Take advantage of shared physical memory for buffers and images!

Optimizing Memory Accesses

- Merging kernels reduces memory traffic
  - Computer vision algorithms often form pipelines
  - Merging multiple kernels in a pipeline can reduce trips to memory
    - Also reduces runtime overhead!



read
filter_x
write

read
filter_y
write

read
filter_x
~~write~~
~~read~~
filter_y
write

- But mind instruction cache size (2K – 4K instructions)!

Used to speedup OpenCV* separable filters!

Memory Access Patterns

- SLM and L3$ are organized differently
  - The access pattern between work-items on a hardware thread is important!

L3$ (global and constant)

line	line 0	line 1	• • •
addr	0x00	0x40	

64-byte cache lines

Touch as few cache lines as possible

SLM (local)

bank	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	0	1	2	3	• • •
addr	0x00	0x04	0x08	0x0c	0x10	0x14	0x18	0x1c	0x20	0x24	0x28	0x2c	0x30	0x34	0x38	0x3c	0x40	0x44	0x48	0x4c	

16 banks on 4-byte boundaries

Touch as many banks as possible

Memory Access Patterns

✦ Examples

	L3$	SLM
`float x = data[get_global_id(0)];`	1 cache line Full bandwidth!	16 banks Full bandwidth!
`float x = data[get_global_id(0) + 1];`	2 cache lines Half bandwidth!	16 banks Full bandwidth!
`float x = data[get_global_id(0) * 2];`	2 cache line Half bandwidth!	8 banks Half bandwidth!
`float x = data[get_global_id(0) * 16];`	16 cache lines Worst case!	1 bank Worst case!
`float x = data[get_global_id(0) * 17];`	16 cache line Worst case!	16 banks Full bandwidth!

When picking a memory type, consider access patterns!

Registers Vs. Memory

- An OpenCL work item has access to up to 512 bytes of register space
- Bandwidth to registers faster than any memory
- Loading and processing blocks of pixels in registers is very efficient!
  - Example: non-separable convolution (filter2D) in OpenCV

```
float sum[PX_PER_WI_X] = { 0.0f };
float k[KERNEL_SIZE_X];
float d[PX_PER_WI_X + KERNEL_SIZE_X];
// Load filter kernel in k, input data in d
...
// Compute convolution
for (px = 0; px < PX_PER_WI_X; ++px)
    for (sx = 0; sx < KERNEL_SIZE_X; ++sx)
        sum[px]= mad(k[sx], d[px + sx], sum[px]);
```

Use registers instead of memory, where possible!

Maximizing Compute Performance

- Use `floats` instead of `ints` wherever possible to maximize co-issue
- Avoid `long` and `size_t` data types
  - Prefer `float` over `int`, if possible
  - Using `short` data types may improve performance
- Trade accuracy for speed, where appropriate
  - Use “native” built-ins, `-cl-fast-relaxed-math`
  - Use `mad()` / `fma()` (or use `-cl-mad-enable`)

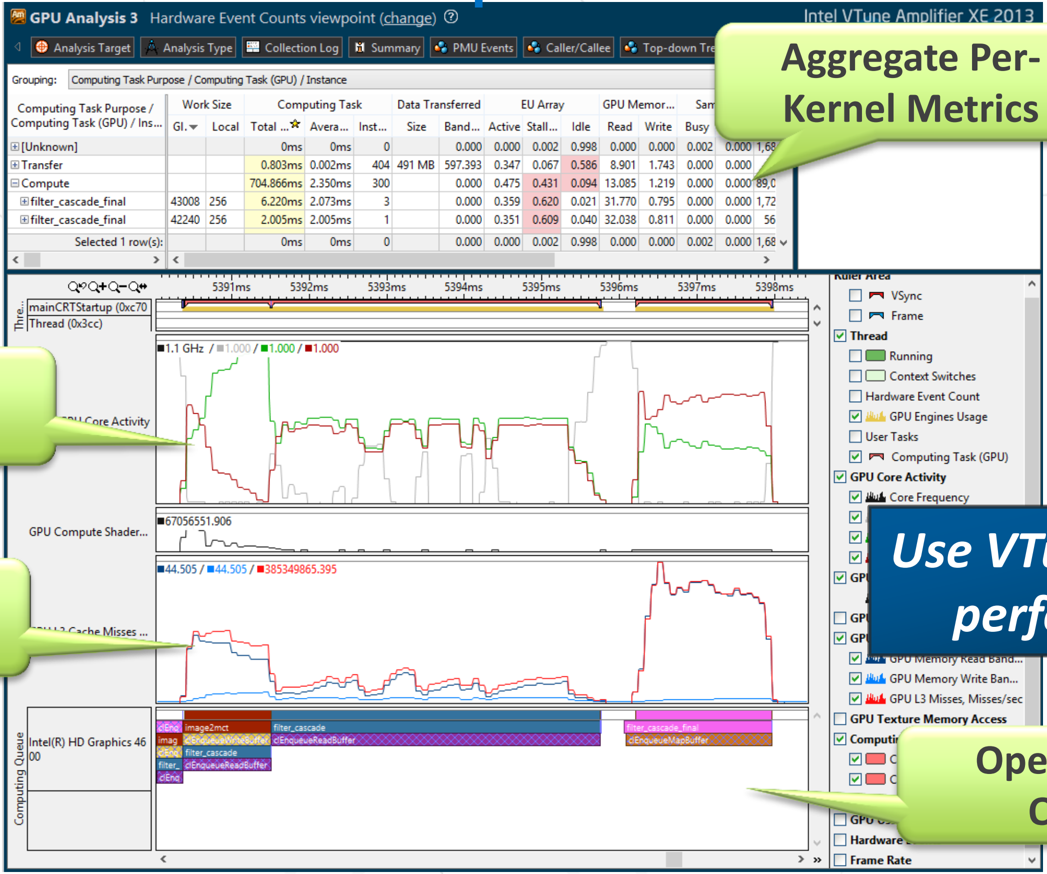
`x = cos(i);` → `x = native_cos(i);`

Used to speedup OpenCV* SURF and HOG!

Agenda

- OpenCL* on Intel® Graphics
- OpenCV 3.0 on Intel® Graphics
- Intel® Graphics Architecture Overview
- Optimization Techniques
  - Maximizing Occupancy
  - Optimizing Memory Access
  - Using Registers
  - Maximizing Computation
- Intel® VTune Support for OpenCL
- Summary / Questions

Intel® VTune™ Amplifier XE 2013



Grouping: Computing Task Purpose / Computing Task (GPU) / Instance

Computing Task Purpose / Computing Task (GPU) / Ins...	Work Size		Computing Task			Data Transferred		EU Array			GPU Memor...		Sam		
	Gl.	Local	Total ...	Avera...	Inst...	Size	Band...	Active	Stall...	Idle	Read	Write	Busy		
⊞ [Unknown]			0ms	0ms	0		0.000	0.000	0.002	0.998	0.000	0.000	0.002	0.000	1,68
⊞ Transfer			0.803ms	0.002ms	404	491 MB	597.393	0.347	0.067	0.586	8.901	1.743	0.000	0.000	
⊟ Compute			704.866ms	2.350ms	300		0.000	0.475	0.431	0.094	13.085	1.219	0.000	0.000	89,0
⊞ filter_cascade_final	43008	256	6.220ms	2.073ms	3		0.000	0.359	0.620	0.021	31.770	0.795	0.000	0.000	1,72
⊞ filter_cascade_final	42240	256	2.005ms	2.005ms	1		0.000	0.351	0.609	0.040	32.038	0.811	0.000	0.000	56
Selected 1 row(s):			0ms	0ms	0		0.000	0.000	0.002	0.998	0.000	0.000	0.002	0.000	1,68

Aggregate Per-Kernel Metrics

EU Activity

Memory Traffic

Use VTune for CPU and GPU performance analysis!

OpenCL* Command Queue Profile

Agenda

- OpenCL* on Intel® Graphics
- OpenCV 3.0 on Intel® Graphics
- Intel® Graphics Architecture Overview
- Optimization Techniques
  - Maximizing Occupancy
  - Optimizing Memory Access
  - Using Registers
  - Maximizing Computation
- Intel® VTune Support for OpenCL
- **Summary / Questions**

Summary

- OpenCL* makes excellent use of Intel® Graphics architecture in a standard programming model
- OpenCV* uses OpenCL* to take advantage of Intel® Graphics
- Following optimization advice for Intel® Graphics can provide dramatic performance improvements
  - Maximize occupancy
  - Optimize memory accesses
  - Use registers
  - Optimize compute
- Use Intel® VTune™ Amplifier to analyze your code and guide optimizations

Acknowledgements

- This presentation would not have been possible without material and review comments from many people – Thank you!
- Ben Ashbaugh, Murali Sundaresan, Stephen Junkins, Deepti Joshi, Tom Craver, Brijender Bharti, Michal Mrozek, Pavan Lanka, Adam Lake, Arnon Peleg, Raun Krisch, Berna Adalier, Allen Hux, Robert Ioffe, Mike MacPherson, Dan Petre, Konstantin Rodyushkin, Mikhail Letavin, Maxim Shevtsov, Alexander Batushin, Tim Bauer

Download, Learn, Code, Optimize

- Free download of Intel SDK for OpenCL Applications at: intel.com/software/opencl
- Follow us: @IntelOpenCL
- Contact as through our forum:

http://software.intel.com/en-us/forums/intel-opencl-sdk

Try related products

- Native client development with Intel® Integrated Native Developer Experience (Intel® INDE)
- Performance tuning with the Intel® VTune™ Amplifier XE
- Media performance with the Intel® Media SDK

Intel
SDK for
OpenCL™
Applications

OpenCL

What is available online?

- ✓ Free Downloads
- ✓ Code Samples
- ✓ Documentation
- ✓ Tech Articles
- ✓ Reviews
- ✓ Forums and Support
- ✓ Webinars

Questions?