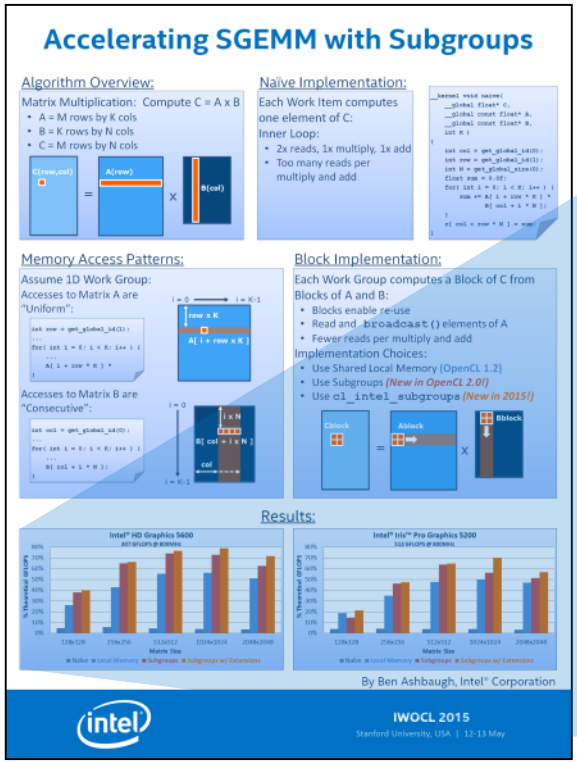
Modeling Explicit SIMD Programming With Subgroup Functions

Ben Ashbaugh, Biju George

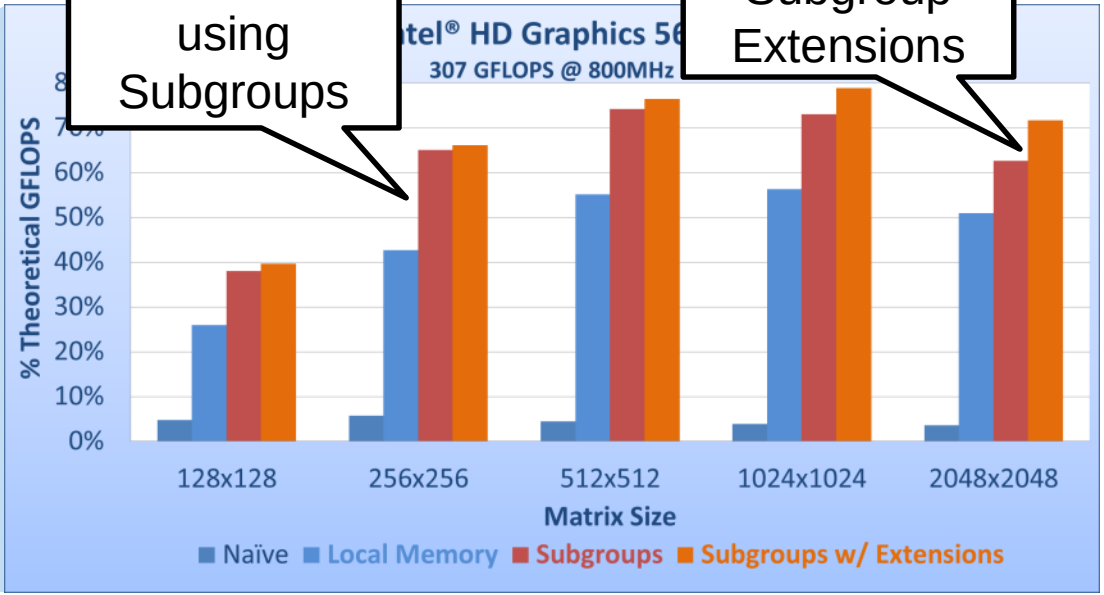
IWOCL 2017

From IWOCL 2015:



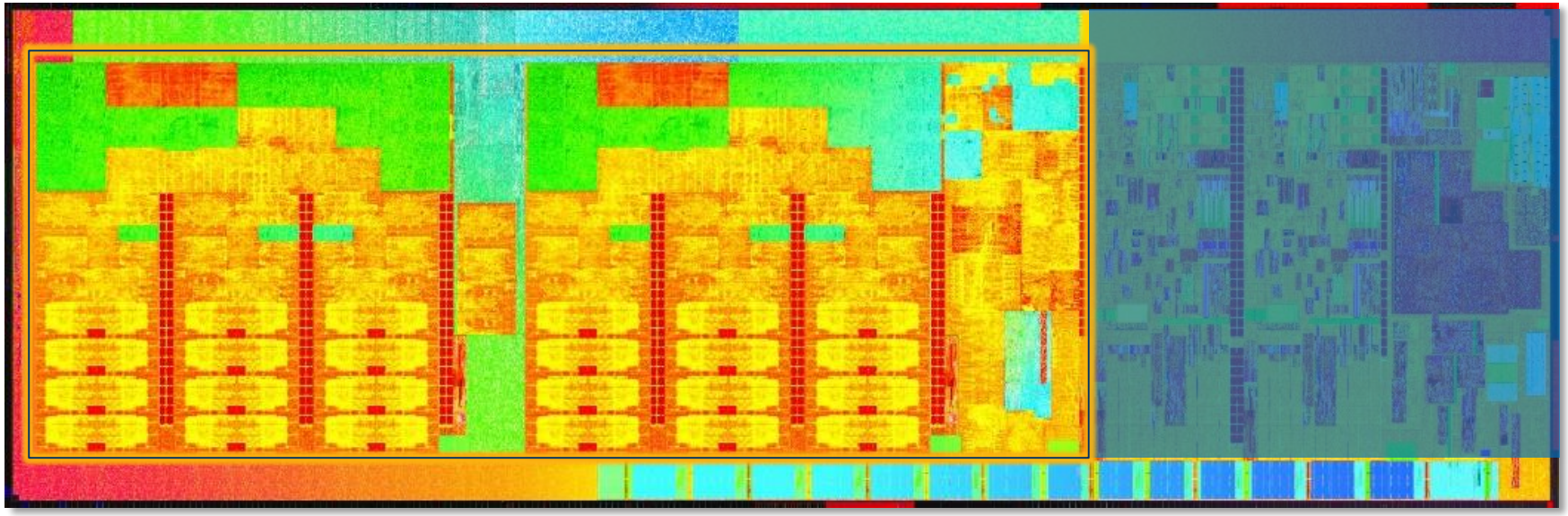
Large performance increase using Subgroups

Additional performance increase using Subgroup Extensions

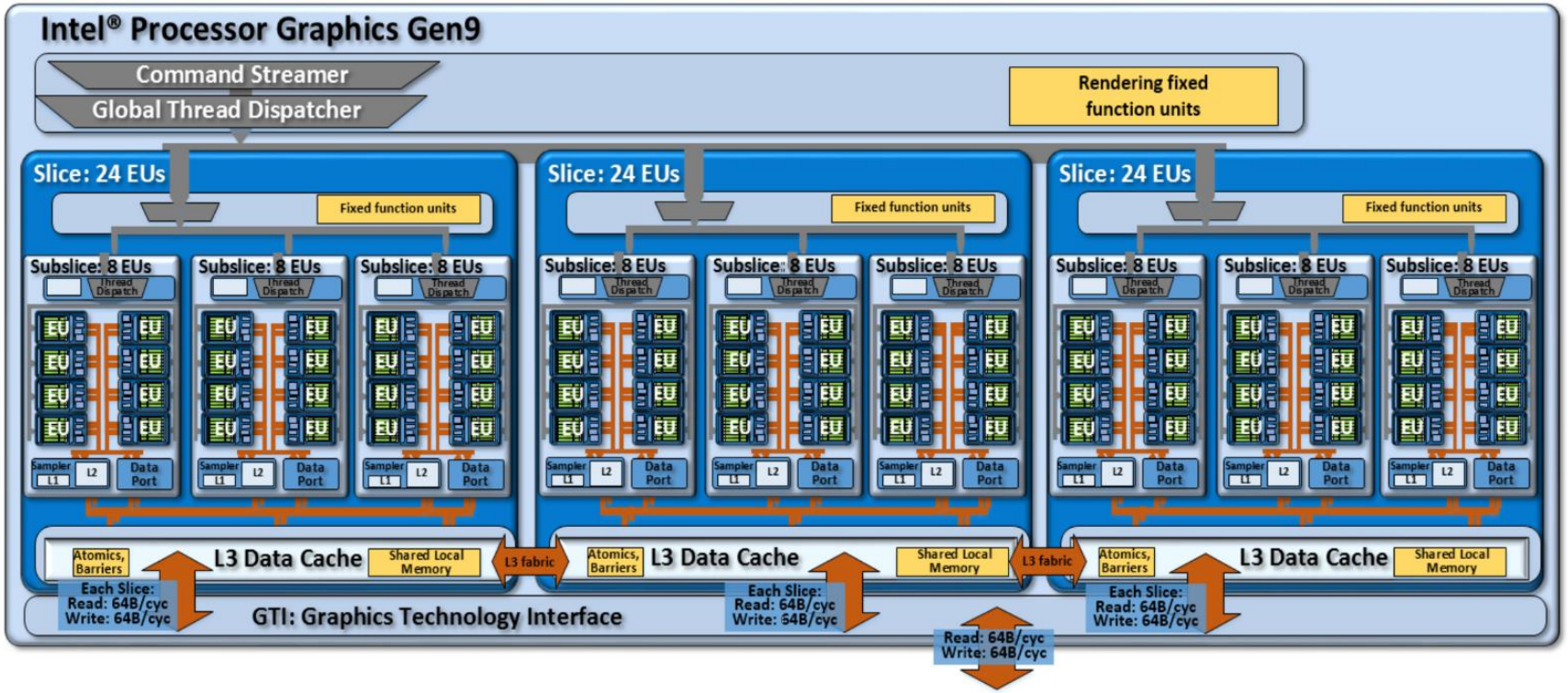


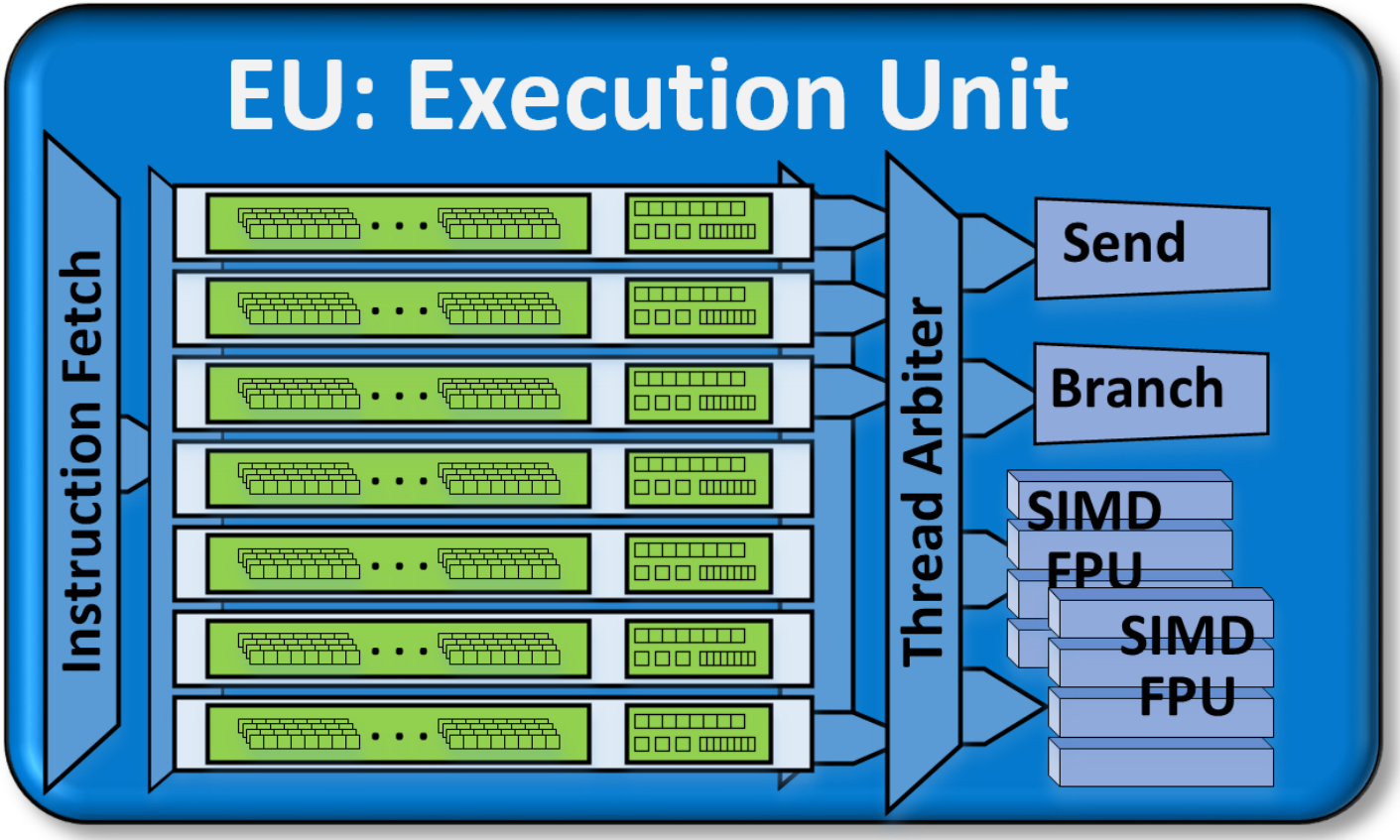
Executing OpenCL™ Kernels on Intel® Processor Graphics

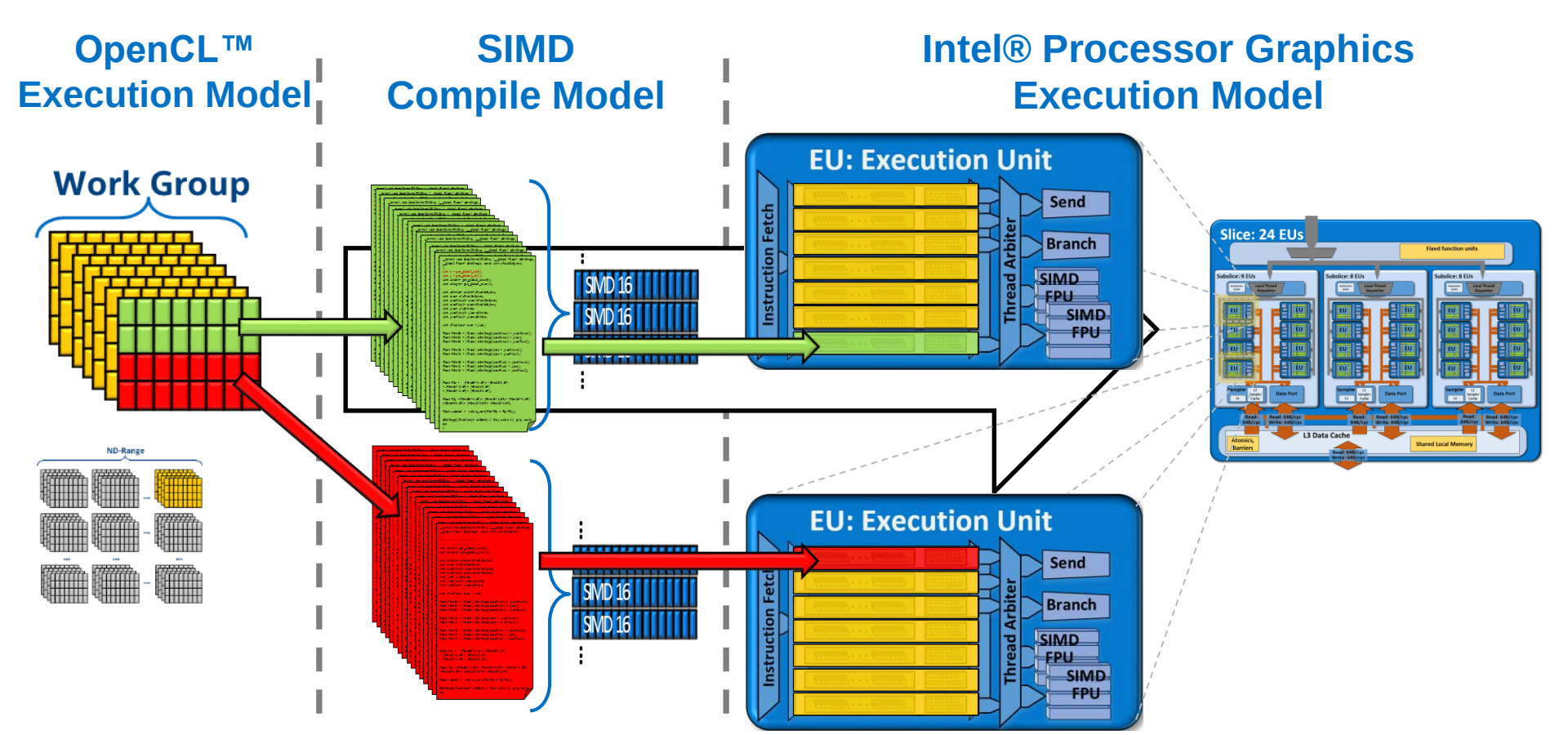
Intel® Core™ i5 with Iris™ Graphics 6100:



Intel® Iris™ Pro Graphics 580




EU: Execution Unit
Instruction Fetch
Thread Arbiter
Send
Branch
SIMD FPU
SIMD FPU



OpenCL™ Work Groups Assigned to One or More EU Threads, Across Multiple EUs

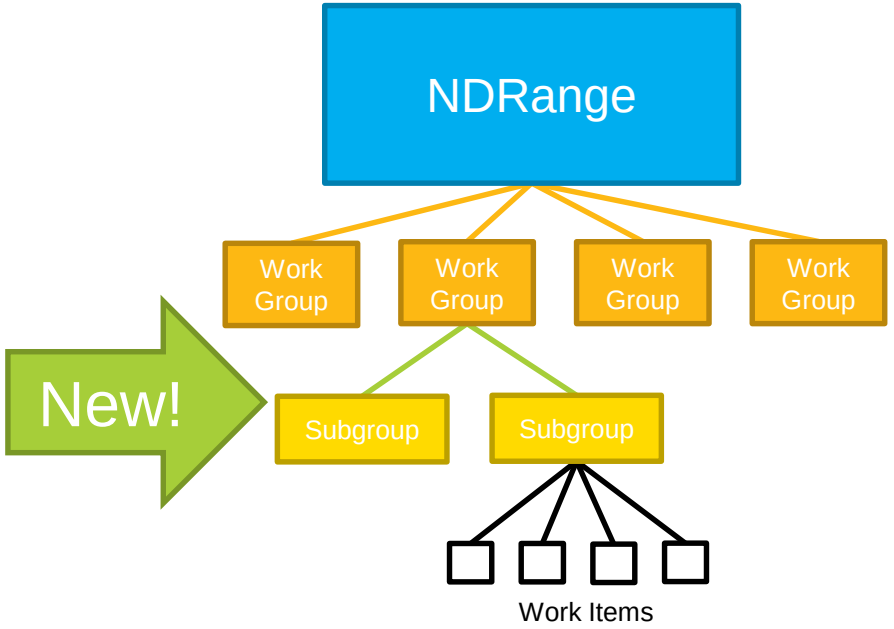
What is a Subgroup?

A Subgroup is a Collection of Work Items

- Another Level in the Execution Hierarchy
- Between Work Groups and Work Items

Key Takeaways:

- On Intel® Processor Graphics, work items in a subgroup execute on the same EU Thread!
- Subgroups can used specialized SIMD instructions for “block operations”



Subgroup Functions bring “Explicit SIMD” to OpenCL kernels!

Block Reads and Writes

Block Copies in Standard OpenCL™:

Function	Description
event_t **async_work_group_copy** (__local gentype *dst, const __global gentype *src, size_t *num_gentypes*, event_t *event*) event_t **async_work_group_copy** (__global gentype *dst, const __local gentype *src, size_t *num_gentypes*, event_t *event*)	Perform an async copy of *num_gentypes* `gentype` elements from *src* to *dst*. The async copy is performed by all work-items in a work-group and this built-in function must therefore be encountered by all work-items in a work-group executing the kernel with the same argument values; otherwise the results are undefined. Returns an event object that can be used by **wait_group_events** to wait for the async copy to finish. The *event* argument can also be used to associate the **async_work_group_copy** with a previous async copy allowing an event to be shared by multiple async copies; otherwise *event* should be zero. If *event* argument is non-zero, the event object supplied in *event* argument will be returned. This function does not perform any implicit synchronization of source data such as using a **barrier** before performing the copy.

(Potentially) Asynchronously Copy Data from Global Memory to Local Memory!

Problems:

- Requires Local Memory to Share Data
- Requires Work Group Barriers to Synchronize Access
- Specialized SIMD Instructions for Block Copies Operate on Registers (AKA Private Memory)

→ Infrequently Used, In Practice

Block Reads and Writes: `cl_intel_subgroups`

Intel `cl_intel_subgroups` Extension Adds Subgroup Block Reads and Writes:

For Buffers:

```
// block read
gentype intel_sub_group_block_read(
    const __global gentype* p );

// block write
void intel_sub_group_block_write(
    __global gentype* p,
    gentype data );
```

And Images:

```
// block read
gentype intel_sub_group_block_read(
    image2d_t image,
    int2 coord );

// block write
void intel_sub_group_block_write(
    image2d_t image,
    int2 coord,
    gentype data );
```

These functions were used to accelerate SGEMM.

Notes:

- Data is read into and written from registers.
- Block sizes are *implicit* – determined by subgroup size.

Block Reads and Writes: `cl_intel_media_block_io`

For Images, Intel GPUs also support *flexible* block reads and writes

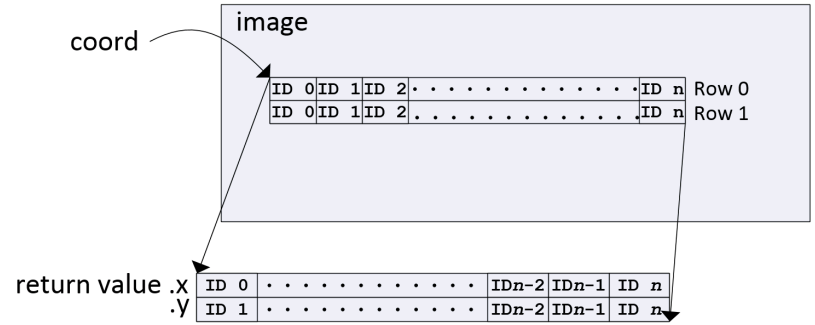
Intel `cl_intel_media_block_io` Extension Adds Additional Functions

- Explicit block sizes, full application control, still operates on registers

Implicit Block Size:

```
uint2 return_value = intel_sub_group_block_read(
    image,
    coord );
```

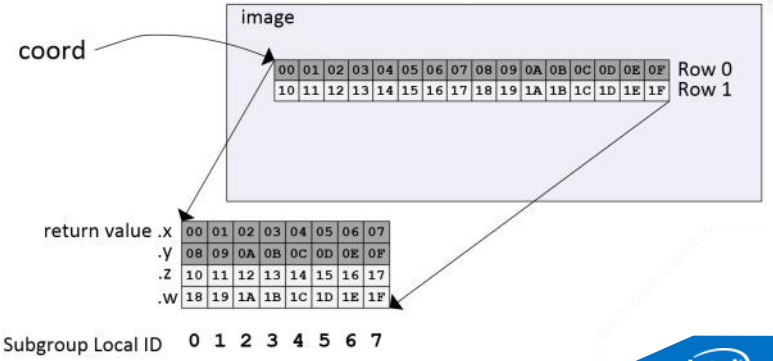
Two Component Block Read:



Explicit Block Size:

```
char4 return_value = intel_sub_group_media_block_read_uc4(
    coord,
    16,
    2,
    image );
```

16x2 Media Block Read for Subgroup Size 8:



Block Read and Write Benefits

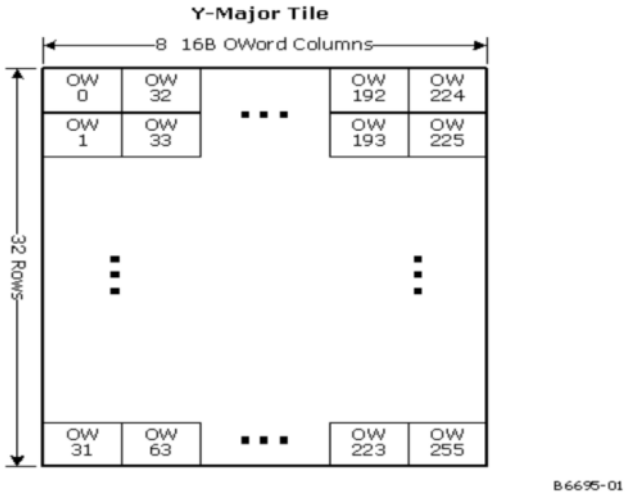
Performance!

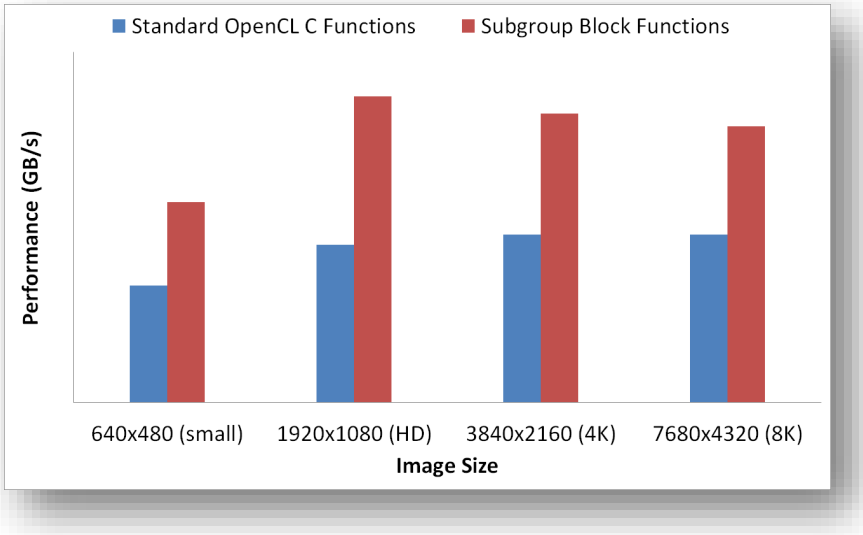
- Address Arithmetic: Compute one address per subgroup vs. per work item
- Block Sizes: Read or write lots of data per instruction

Block Read and Write Benefits

Particularly Beneficial for Images:

- "Raw" Reads and Writes: Process pixels from multiple rows and/or columns
- Cache-friendly Reads and Writes: Avoid partial cache lines with Tiled Images





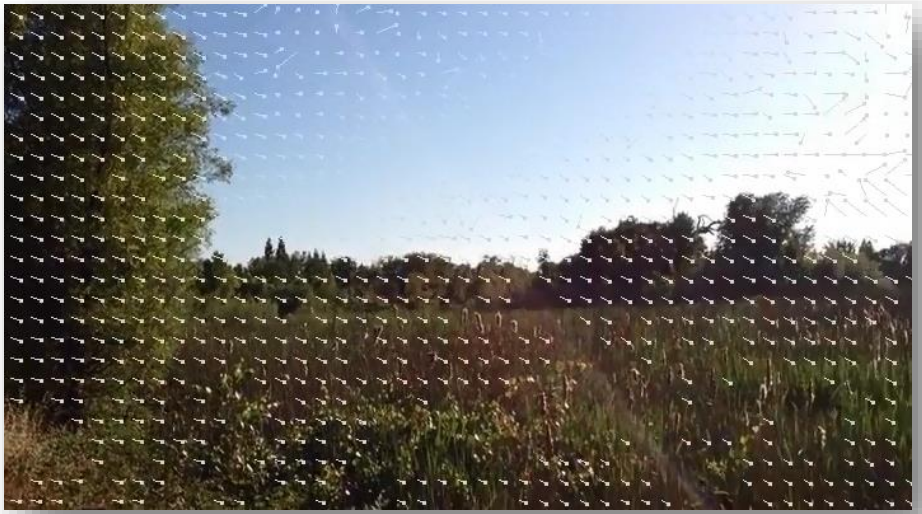
Video Motion Estimation (VME)

What is Video Motion Estimation?

A key algorithm component for Video Encoding, Frame Rate Conversion, Asynchronous Space Warping for Virtual Reality, more…

A Block Operation:

- Simplified: (In) Source and Reference Blocks → (Out) Motion Vectors
- In reality: much more!



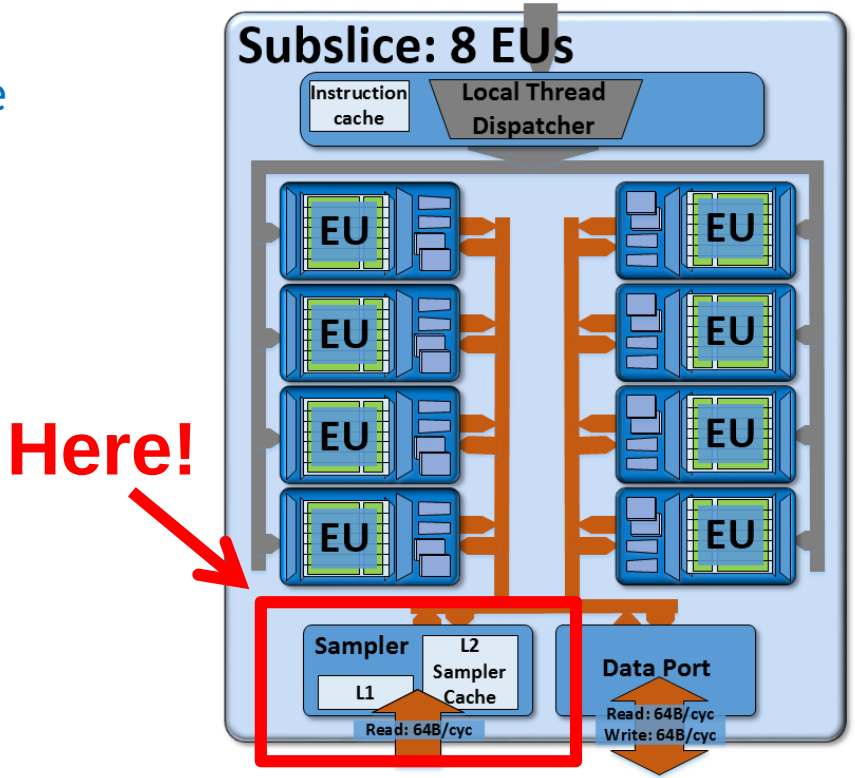
Video Motion Estimation Hardware

Intel® Processor Graphics has a dedicated Motion Estimation Engine

- Part of the Media Sampler

How to expose this capability to OpenCL™ kernels…

- Programmed at the EU Thread Level



Motion Estimation in OpenCL™ Kernels: `cl_intel_device_side_avc_motion_estimation`

Solution: Subgroup functions! (of course!)

- Described by the **`cl_intel_device_side_avc_motion_estimation`** extension

Unique Characteristic: Every Step is a Subgroup Operation!

- Initialization
- Configuration
- Execution
- Assigning Results

```
// Initialize the VME payload:
intel_sub_group_avc_ime_payload_t payload =
    intel_sub_group_avc_ime_initialize(...);

// Configure the VME payload:
payload =
    intel_sub_group_avc_ime_set_single_reference(
        ...,
        payload);
payload =
    intel_sub_group_avc_ime_set_motion_vector_cost_function(
        ...,
        payload);

// Perform the VME operation:
intel_sub_group_avc_ime_result_t result =
    intel_sub_group_avc_ime_evaluate_with_single_reference(
        ...,
        payload);

// Assign VME results:
long mvs =
    intel_sub_group_avc_ime_get_motion_vectors(result);
ushort sads =
    intel_sub_group_avc_ime_get_inter_distortions(result);
```

Summary and Future Work

Summary

OpenCL™ Subgroups are Great!

Subgroup Functions Bring “Explicit SIMD” Programming Concepts to “Implicit SIMD” OpenCL Kernels

- Utilize Additional Hardware Features
- Improve Performance
- Add New Functionality

Future Work:

- Application to other domains: AVX intrinsics?
- Programming Models: Hierarchical Parallelism?

Thank You!

Acknowledgements: This presentation would not have been possible without material and review comments from many people – thank you!

Stephen Junkins, Jeffrey McAllister, Robert Ioffe, ...

Useful Links:

The Compute Architecture of Intel® Processor Graphics Gen9

- https://software.intel.com/sites/default/files/managed/c5/9a/The-Compute-Architecture-of-Intel-Processor-Graphics-Gen9-v1d0.pdf

SGEMM for Intel® Processor Graphics Sample Code

- https://software.intel.com/en-us/articles/sgemm-for-intel-processor-graphics

Intel Subgroup Extensions

- https://www.khronos.org/registry/OpenCL/extensions/intel/cl_intel_subgroups.txt
- https://www.khronos.org/registry/OpenCL/extensions/intel/cl_intel_subgroups_short.txt
- https://www.khronos.org/registry/OpenCL/extensions/intel/cl_intel_required_subgroup_size.txt
- https://www.khronos.org/registry/OpenCL/extensions/intel/cl_intel_media_block_io.txt
- https://www.khronos.org/registry/OpenCL/extensions/intel/cl_intel_device_side_avc_motion_estimation.txt

Legal Notice and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit **http://www.intel.com/performance**.

Intel, the Intel logo and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

© 2017 Intel Corporation.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

Legal Disclaimer and Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

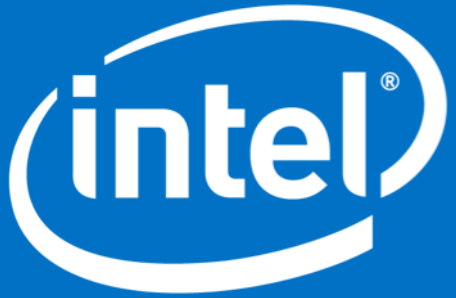
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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.

Copyright © 2017, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

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 SSSE3 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


intel