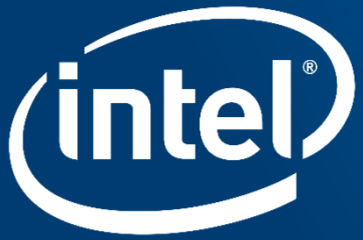


# EMPLOYING OUT-OF-ORDER QUEUES FOR BETTER GPU UTILIZATION IN OPENCL

Krzysztof Laskowski, Intel
Pavan K Lanka, Intel

# Legal Notices and Disclaimers

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

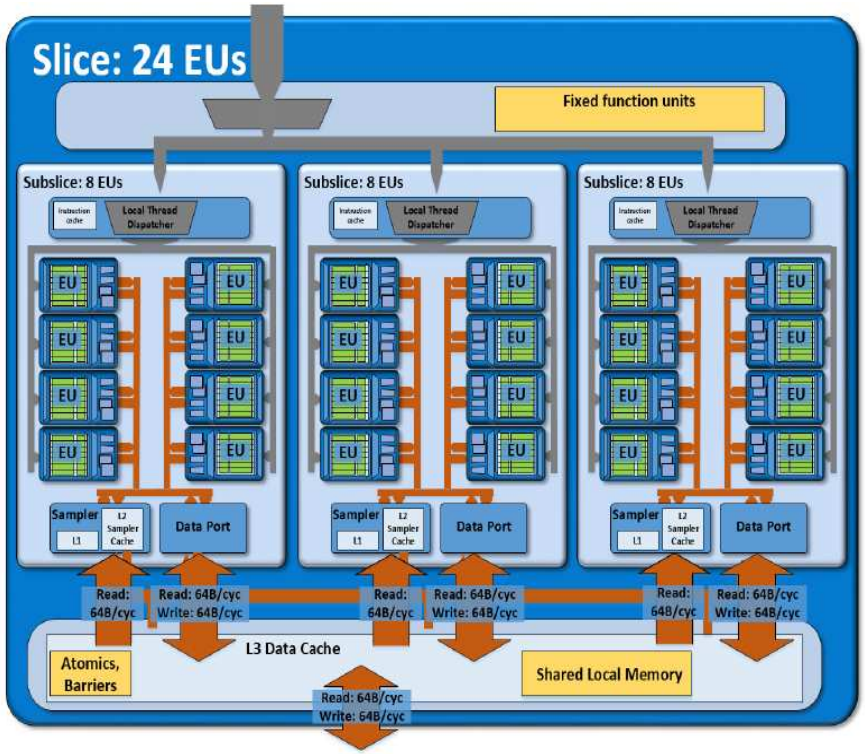
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors 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.
- All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.
- All products, platforms, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice. All dates specified are target dates, are provided for planning purposes only and are subject to change.
- This document contains information on products in the design phase of development. Do not finalize a design with this information. Revised information will be published when the product is available. Verify with your local sales office that you have the latest datasheet before finalizing a design.
- 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, Intel Inside, Intel Atom and Intel Core are trademarks of Intel Corporation in the U.S. and other countries.
- Other names and brands may be claimed as the property of others.
- 

# Executive Summary

- Efficient scheduling of work to GPU is important for overall performance of applications.
- The discussed optimization aims to fully utilize the General Purpose Graphics Processing Unit (GPGPU) Pipeline taking into consideration:
  - Workload characteristics
  - How the hardware actually works.

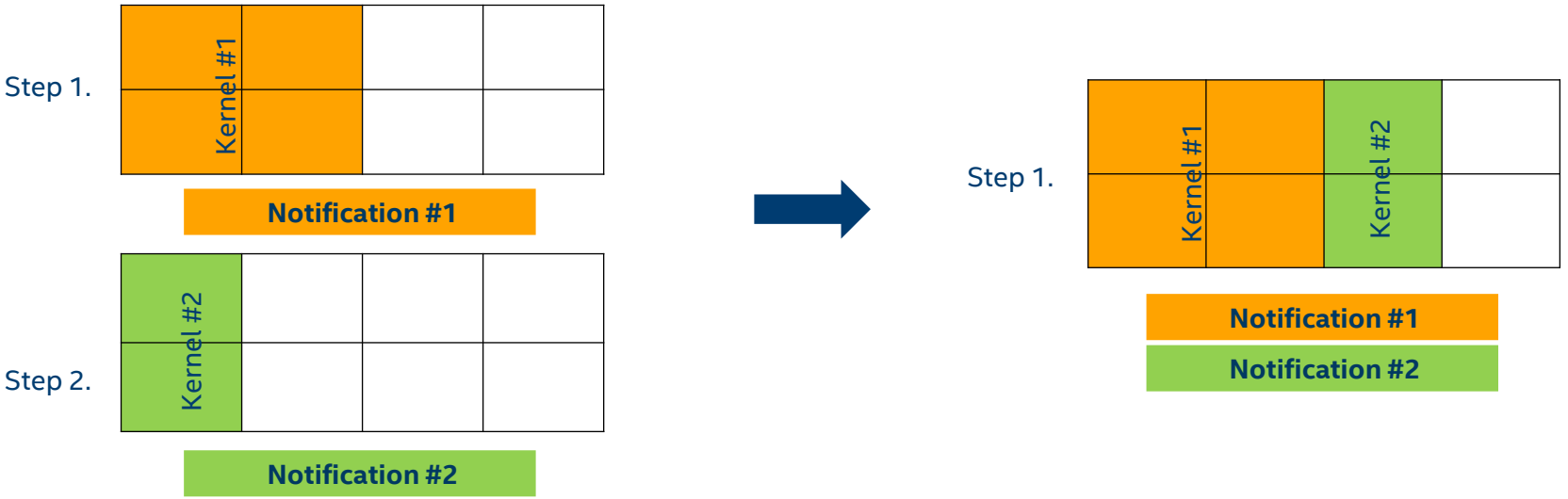
# CHALLENGE

# Challenge




- Many general purpose Execution Units (EUs) and dedicated Fixed Function HW blocks.
- The compute power of Intel® Processor Graphics is continuously growing over generations.
- **How to efficiently use all the compute power of the GPU for various workloads?**

# Goal




Enable independent kernels to execute simultaneously whenever possible to keep all GPU assets busy

# SOLUTION

# Solution: Alternatives Considered: In-Order

- Do the optimization implicitly as part of the default In-Order Execution Model.

  **In-order Execution**: A model of execution in OpenCL where the commands in a command queue are executed in order of submission with each command running to completion before the next one begins.

- OpenCL Runtime needs to detect independent kernels in the sequence of commands and remove the synchronization points between them.
- Not feasible due to Shared Virtual Memory (SVM) related corner cases for which the optimization would break the In-Order Execution model requirements.

# Solution: Alternatives Considered: Out-of-Order

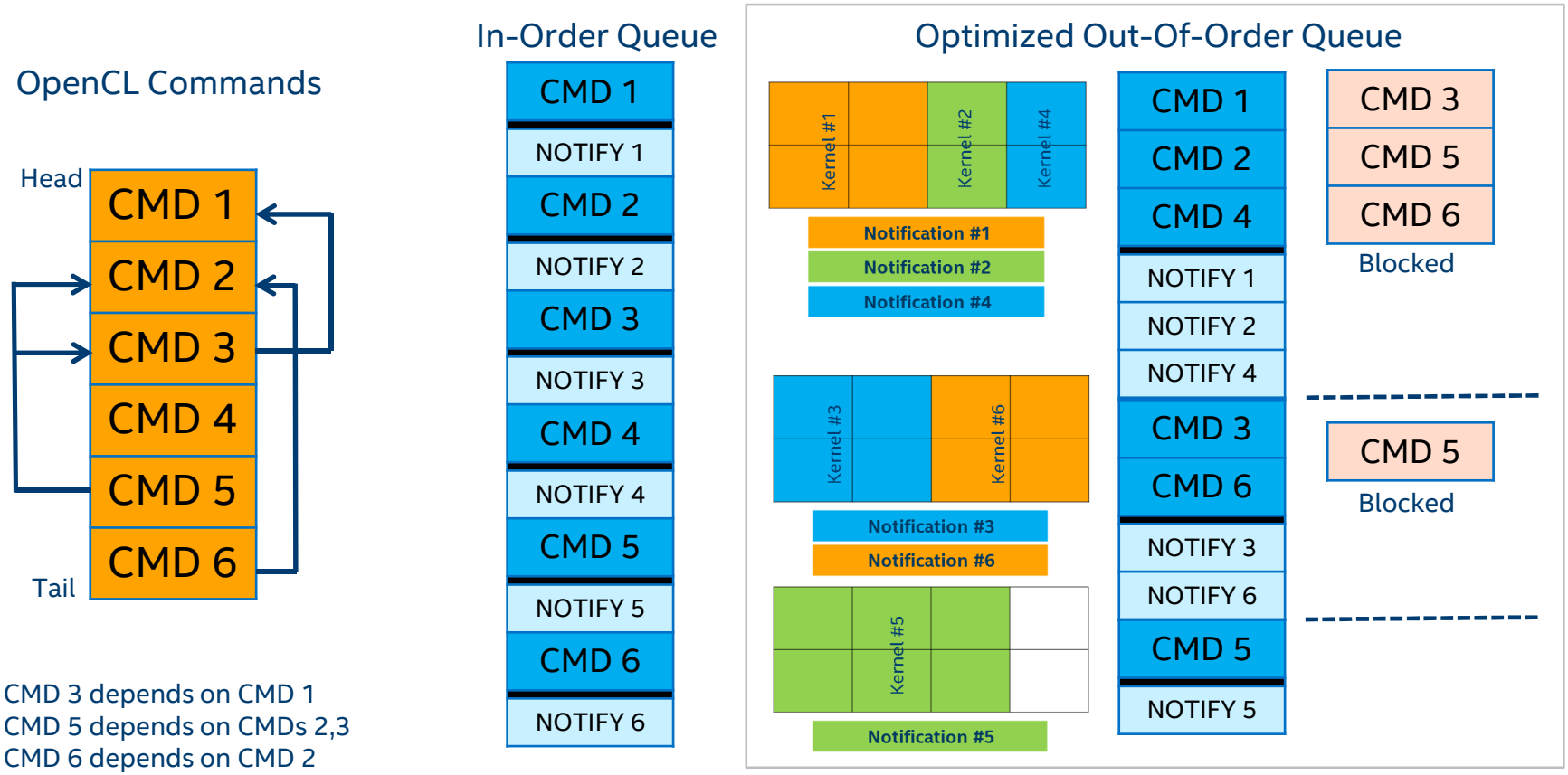
- **Add support for Out-of-Order Execution Model.**

  **Out-of-Order Execution**: A model of execution in which commands placed in the work queue may begin and complete execution in any order consistent with constraints imposed by event wait lists and command-queue barriers.

- Application is responsible for specifying the right dependencies between enqueues.

Leverage the existing Out-of-Order Execution model for the optimization

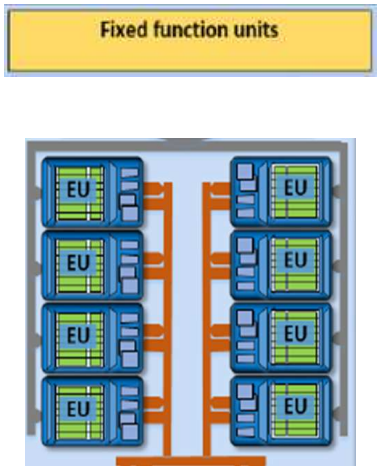
# Solution: Overview



# KEY RESULTS

# Key Results: VME+GPGPU

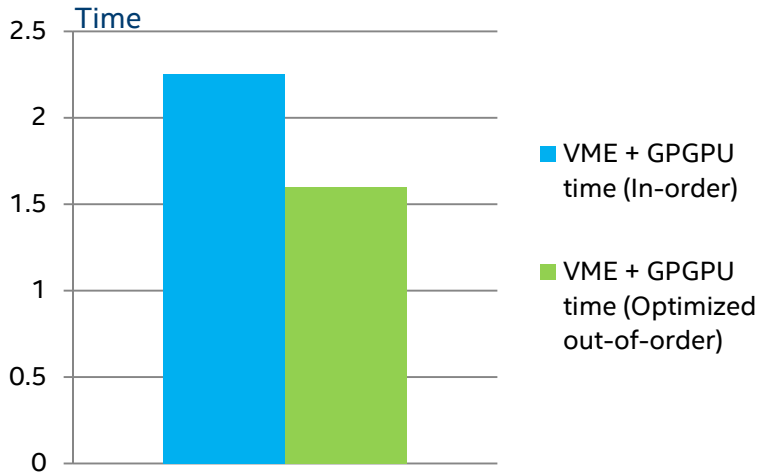
- VME (Video Motion Estimation) extension available in Intel's OpenCL uses a dedicated functional HW units to perform motion estimation algorithm and calculate the motion vectors.
- VME kernel still needs some EUs but number of EUs used for this purpose may be limited.
- In the optimized Out-of-Order Execution solution we can potentially execute the VME kernel in parallel with regular GPGPU OpenCL kernels.



# Key Results: VME+GPGPU

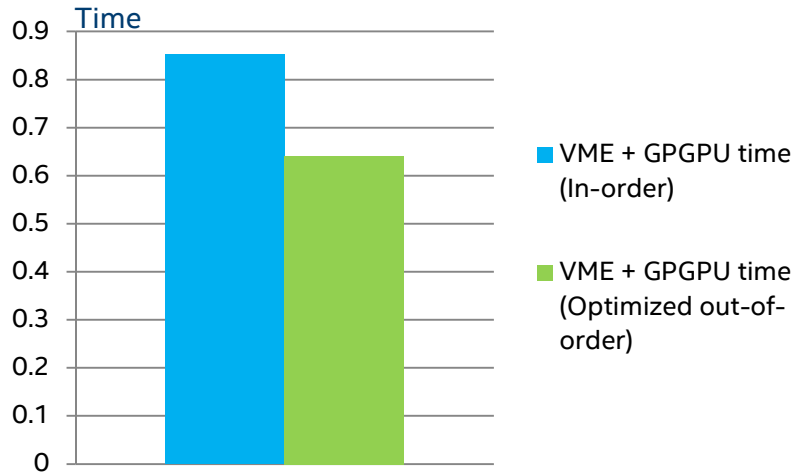
- Time(VME kernel) == Time(GPGPU kernel) == **T**
- Ideally: Time(VME kernel + GPGPU kernel) == **T** (2x gain)

Intel® HD Graphics 5500




Total execution time: **2.2s →1.6s (1.4x)**

Intel® Iris™ Graphics 6100

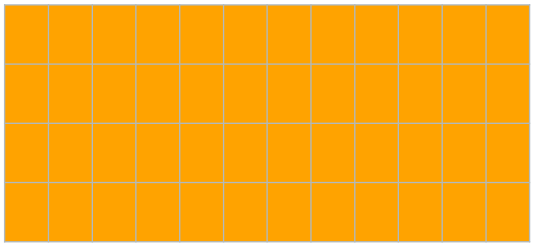



Total execution time: **0.9s →0.6s (1.3x)**

# Key Results: Multiple independent operations

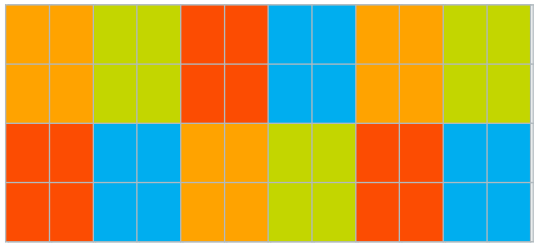
- Another use case is executing multiple streams of general purpose commands operating on independent sets of data.
- An example is a matrix multiplication application

Large data set



(GPU EUs)

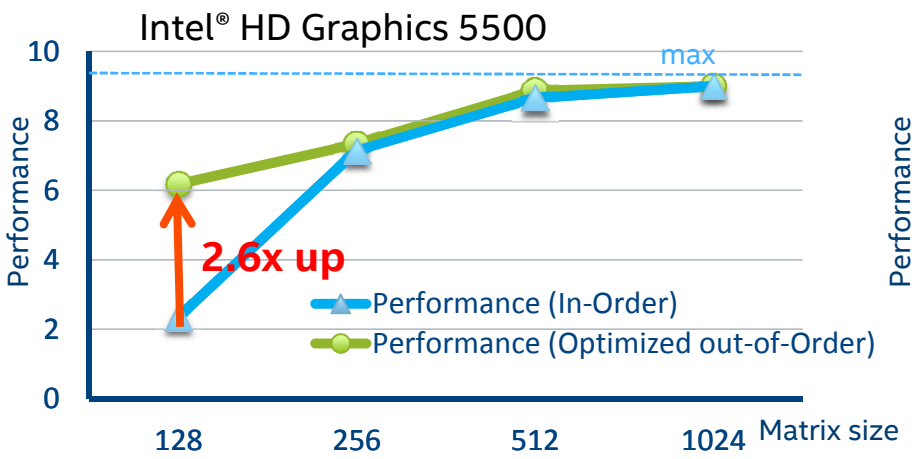
Small data sets – do multiple streams

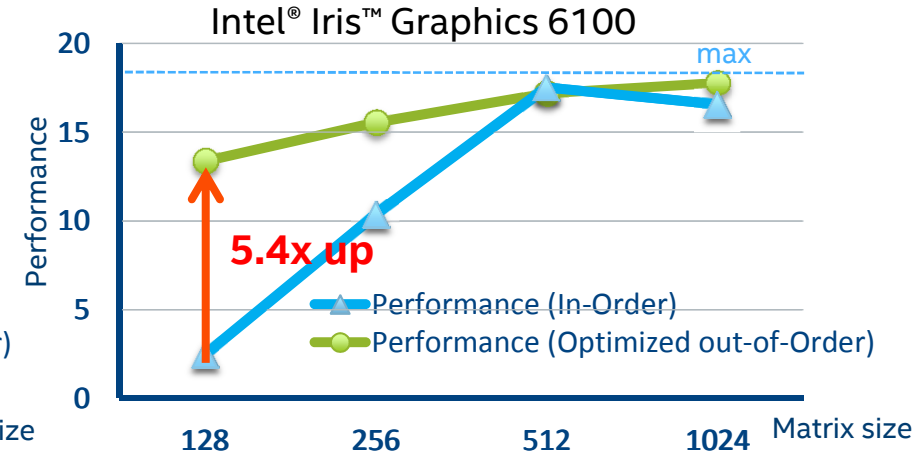


(GPU EUs)

# Key Results: Multiple independent operations

- Multiple matrix multiply operations (naïve algorithm) for different matrix sizes.




Application thread usage statistics:

| Matrix size (NxN) | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|
| Threads used | **8** | **32** | **128** | **512** |

Threads available in HW:

| HW configuration | Intel® HD Graphics 5500 | Intel® Iris™ Graphics 6100 |
|---|---|---|
| Threads available | **168** | **336** |

# Challenges & TODOs

- Out-of-Order Execution model is an opt-in feature.
- Developers need to adapt to the new model.
- Need to be aware of:
  - Limited GPU resources
  - OS restrictions
- With Event Profiling we can't guarantee parallel execution benefits.

# USER GUIDELINES

# Efficient use of Out-of-Order Queues

- Create an out-of-order command queue in the following manner:

```
cl_command_queue_properties qProperties = CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE;
cl_command_queue queue = clCreateCommandQueue(context, deviceIds[0], qProperties, &error);
```

- Avoid `CL_QUEUE_PROFILING_ENABLE` property with Out-of-Order Queues as it may severely limit the expected performance gains.

- Identify independent tasks that can run in parallel and prepare them to execute through one Out-of-Order command-queue:

```
for ( cl_uint i = 0 ; i < iterations ; i++)
{
    clEnqueueNDRangeKernel(queue, vme_kernel, 2, NULL, vme_gws, NULL, 0, NULL, NULL);
    clEnqueueNDRangeKernel(queue, gpgpu_kernel_1, 1, NULL, gpgpu_gws_1, NULL, 0, NULL, NULL);
    clEnqueueNDRangeKernel(queue, gpgpu_kernel_2, 1, NULL, gpgpu_gws_2, NULL, 0, NULL, NULL);
     //(etc.)
}
```

# Effiecient use of Out-of-Order Queues

- For each stream of commands avoid flushing or blocking operations such as clFlush, clFinish, clWaitForEvents or blocking enqueue commands and manage dependencies with event wait-lists, for example:

```
 cl_event events[streams];
for(int s = 0; s < streams; s++)
{
    clEnqueueNDRangeKernel(queue, gpgpu_kernel, 1, NULL, gpgpu_gws, NULL, 0, NULL, &events[s]);
    clEnqueueMapBuffer(queue,
                       buffer_arg1,
                       CL_FALSE, //non-blocking map
                       CL_MAP_READ,
                       0,
                       buffer_size,
                       1, &events[s], 0, &err);
}
clFinish(queue);
```

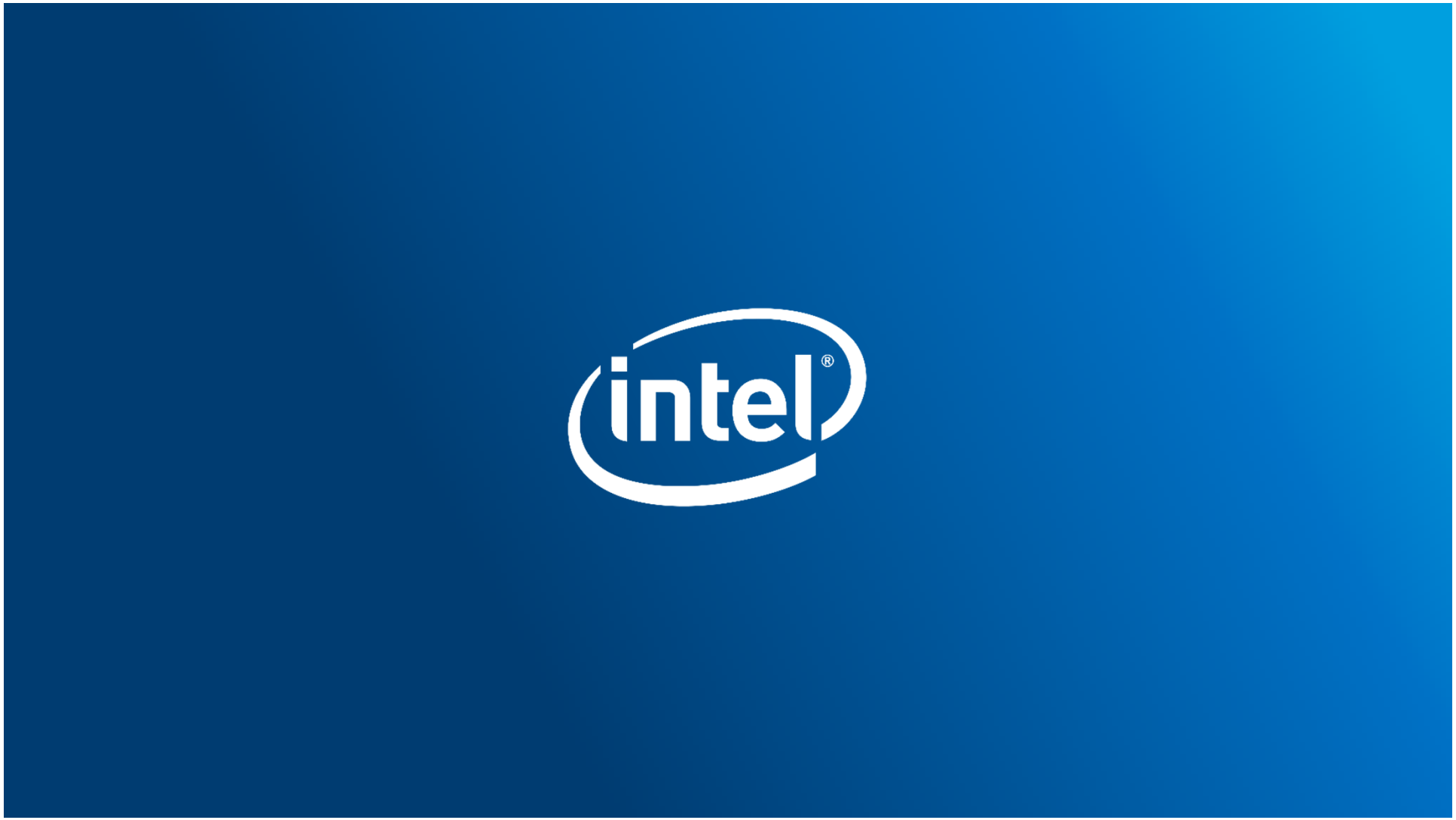
# Remarks

- Be aware of limited GPU resources and OS restrictions when expecting parallel execution benefits.
- When using Out-of-Order queues explicitly manage dependencies between enqueues through events and event_waitlists arguments as there is no in-order execution guarantee.
- The speed-up is observed in the total execution time of multiple commands when enqueued together into the same Out-of-Order command queue. Particular performance gains vary and depend on the workload and a given HW configuration characteristics.

# Conclusion

- Our optimized Out-of-Order implementation can speed up your application several times (up to **1.4x** or **5.4x** in our experiments) depending on workload characteristics and HW behavior/configuration.

Better HW utilization and better performance per Watt
in many applications


intel