Disclaimer and Legal Information

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.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

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.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number/.

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.

Intel, Intel logo, Intel Core, VTune, Xeon are trademarks of Intel Corporation in the U.S. and other countries.

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

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

Microsoft product screen shot(s) reprinted with permission from Microsoft Corporation.

Copyright © 2010-2011 Intel Corporation. All rights reserved.
Optimization Notice

Intel compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel compilers, associated libraries and associated development tools 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 Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (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.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20110307
About the Intel® Media SDK Interoperability Sample

The Intel® Media SDK Interoperability sample demonstrates how to use Intel® Media SDK and Intel® OpenCL SDK together for video decoding and post-processing.

Path

<table>
<thead>
<tr>
<th>Location</th>
<th>Executable</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;INSTALL_DIR&gt;\samples\MediaSDKInterop</td>
<td>Win32\Release\MediaSDKInterop.exe - 32-bit executable</td>
</tr>
<tr>
<td></td>
<td>x64\Release\MediaSDKInterop.exe - 64-bit executable</td>
</tr>
<tr>
<td></td>
<td>Win32\Debug\MediaSDKInterop.exe - 32-bit debug executable</td>
</tr>
<tr>
<td></td>
<td>x64\Debug\MediaSDKInterop.exe - 64-bit debug executable</td>
</tr>
</tbody>
</table>

Introduction

Intel® Media SDK [1] provides fast decoding of h264 and mpeg2 video streams, while Intel® OpenCL SDK is good for pixel processing of video frames (for example OpenCL allows implement pixel parallel algorithm). The Intel® Media SDK Interoperability sample shows how to combine Intel® Media SDK and Intel® OpenCL SDK to get fast video player with additional post processing effects.
Motivation

The Intel® Media SDK Interoperability sample demonstrates a CPU-optimized OpenCL implementation of several filters combined in the Intel® Media SDK pipeline, showing how to:

- integrate processing with Intel® OpenCL SDK into the Intel® Media SDK pipeline and get benefit from hardware-accelerated (if available) video decoding with the Intel® Media SDK pipeline
- implement simple video effects using OpenCL
  - implement kernels the way they benefit from SIMD units and multiple CPU cores
  - organize data exchange between the host and the OpenCL device
- output the final image on the screen using DirectX*.

Algorithm

The algorithm consists of the following stages:

1. Read the frame data from file.
2. Feed the data to the Intel® Media SDK Decoder and get decoded video frame back in NV12 (which is a specific layout for YUV) format.
3. Execute the OpenCL kernel which processes the decoded video frame.
4. Convert the video frame from NV12 format to conventional RGB format
5. Draw RGB image on the screen.

Implementation Details

The Intel® Media SDK Interoperability sample demonstrates implementation of the basic media processing pipeline. The pipeline consists of two main modules:

1. The Decoder module which leverages Intel® Media SDK for HW-assisted decoding mpeg2 and h264 formats.
2. The **Post-Processing** module which uses Intel® OpenCL SDK for efficient implementation of the video effects. This module actually serves as a plug-in for the Intel® Media SDK pipeline.

Both modules work asynchronously: while the Post-Processing module processes frame \( N \), the Decoder module processes \( N+1 \) frame without waiting for another module to complete the current frame. For more information about asynchronous pipelines, please refer to the Intel® Media SDK documentation [1].

**OpenCL™ Implementation**

The Post-processing module introduced above is a plug-in for Intel® Media SDK. The plug-in initializes Intel® OpenCL SDK and generates a list of filters scanning current folder and detecting all files with *.cl extension. Each file is a separate filter implementation.

The Intel® Media SDK Interoperability sample package includes several filters residing in separate *.cl files. Each file has at least one kernel which processes pixels. For example, ProcessY kernel processes the intensity component and ProcessUV processes the color component (which are separated in NV12). The sample contains the following cl filters:

- **_color_control.cl** - enables you to change intensity and color characteristics inside circle controlled by mouse.
- **_copy.cl** - just copies input intensity and color to the output buffer. You can use it as a template for creating your own filters.
- **_edge.cl** - calculates edge intensity and saves correspondence value into the output intensity buffer.
- **_neg.cl** - makes image negative inside selected area.
- **_wave.cl** - simulates wave propagation and distorts image according to simulated waves.

Files contain following kernels:

- **ProcessY** - processes Y component of each pixel of input frame
• **ProcessUV** – processes U and V component of four pixels of input frame

• **Mouse** – executes once for each frame to indicate where the mouse pointer is right now

• **Preprocess** – executes once for each frame to make some common task for whole frame.

When Intel® Media SDK calls the Intel® OpenCL SDK plug-in to execute filter for given frame, `clEnqueueTask` and `clEnqueueNDRangeKernel` functions execute kernels from *.cl files and return a sync event. The host program keeps the event to check whether Post-Processing module is done with a frame via `clGetEventInfo`.

For each frame the Post-Processing module creates OpenCL memory object with `CL_MEM_USE_HOST_PTR` flag to avoid extra memory copying from the Intel® Media SDK frame into Intel® OpenCL SDK buffer.

## Work-Group Size Considerations

Work-group size should be aligned with vertical and horizontal sizes of the input image in the range from 1 to image width, and 1 to image height. Or you can pass the NULL pointer as the local size parameter of `clEnqueueNDRange` to let Intel® OpenCL SDK select the right value automatically.

## Future Work and Enhancements

The Intel® Media SDK Interoperability sample kernels perform calculations for one pixel. That gives some overhead for pixel calls. One potential optimization is to process one line or tile of input image with each work-item.

Currently memory manager is simple. It uses conventional memory allocation routine to allocate frame. In future memory manager will use the OpenCL allocation routine. This will reduce memory manipulation during OpenCL filter execution.
**Project Structure**

The Intel® Media SDK Interoperability sample project has the following structure:

- **src** - folder contain main code, with Intel® OpenCL SDK and Intel® Media SDK initialization and processing functions
  - CSample.cpp – file initializes window and GUI controlled and draws the resulting frame to window
  - CSampleBase.cpp – file contains base class for CSample and has routine code for application initialization
  - main.cpp – file runs CSample class
  - pipeline_decode.cpp – file contains code which constructs processing pipeline for video decoding and post processing
  - sample_opencl_plugin.cpp – file contains code of Intel® Media SDK plug-in for post processing video frames using Intel® OpenCL SDK
- **src_MediaSDK** – folder contains memory management files of Intel® Media SDK samples
  - base_allocator.cpp – base memory allocators
  - sysmem_allocator.cpp – specific memory allocators which allocate buffers and frames in system memory
  - sample_defs.h – some macros and defines used from Intel® Media SDK APIs.

**Additional Requirements**

To build and run Media SDK Interoperability sample you need to:

1. Install Microsoft® DirectX® SDK (June 2010 or later) on your system.
2. Install Intel® Media SDK 3.0 Beta [2] or higher on you system. Create environment variable INTELMEDIASDKROOT if it is not exist. This variable have point to folder where Intel® Media SDK was installed to (for example INTELMEDIASDKROOT=C:\Program Files\Intel\Media SDK\3.0.442.32245 Beta3).
The Intel® Media SDK Interoperability sample uses the following APIs:

- clGetDeviceIDs
- clCreateContext
- clReleaseContext
- clCreateCommandQueue
- clReleaseCommandQueue
- clGetDeviceInfo
- clCreateBuffer
- clCreateProgramWithSource
- clBuildProgram
- clReleaseProgram
- clGetProgramBuildInfo
- clCreateKernel
- clReleaseKernel
- clEnqueueMapBuffer
- clEnqueueUnmapMemObject
- clCreateBuffer
- clReleaseMemObject
- clSetKernelArg
- clEnqueueTask
- clEnqueueNDRangeKernel
- clEnqueueCopyBuffer
- clReleaseEvent
- clFlush.
Controlling the Sample

- **Decode** check box switches Intel® Media SDK Decoder on/off
- **OpenCL** check box switches OpenCL post processing effect on/off
- **OPENCL filer** combo box enables you to choose current OpenCL program
- **Param1** and **Param2** sliders enable you to change parameters passed to OpenCL kernels
- Press **F11** key to switch on/off full screen mode
- Press **F2** key to show/hide currently executed OpenCL source code.

References