Date: 05/10/2022

IWOCL 2022

An Overview of the OpenCL Vendor Extensions Supported in Qualcomm Adreno GPUs

Hongqiang Wang and Balaji Calidas
GPU Team
Qualcomm Technologies, Inc.

Presented by Hongqiang Wang

Outline

- Introduction to OpenCL extensions
- Extensions in Qualcomm Adreno GPUs
- Summary

References

- Adreno GPU SDK - Tools - Qualcomm Developer Network: https://developer.qualcomm.com/software/adreno-gpu-sdk/tools
- Adreno Open CL Machine Learning SDK v2.2: https://developer.qualcomm.com/downloads/adreno-open-cl-machine-learning-sdk-v22
- OpenCL Machine Learning Acceleration on Adreno GPU - Qualcomm Developer Network: OpenCL Machine Learning Acceleration on Adreno GPU - Qualcomm Developer Network
- ML training at the edge: Training on mobile devices - Qualcomm Developer Network: ML training at the edge: Training on mobile devices - Qualcomm Developer Network
- Qualcomm Snapdragon™ Mobile Platform OpenCL General Programming and Optimizations: https://developer.qualcomm.com/download/adrenosdk/adreno-opencl-programming-guide.pdf
- IWOCL 2019, Accelerating Typical Image Processing Operations using Qualcomm Adreno OpenCL Extensions | Proceedings of the International Workshop on OpenCL (acm.org)
- Khronos OpenCL 3.0 extension spec: The OpenCL™ Extension Specification (khronos.org)

OpenCL Extensions

Allow OpenCL vendors to add new functionality and innovate

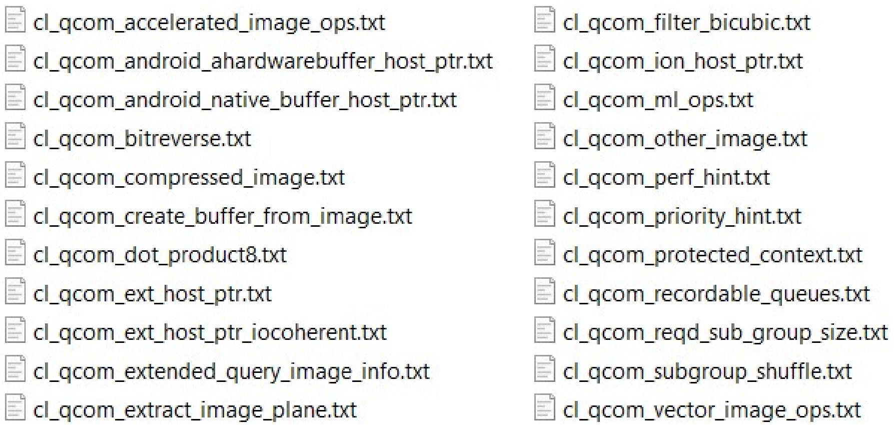
- KHR extensions:
  - Require multiple vendors to adopt
  - Require conformance tests to pass.
- EXT extensions:
  - No conformance tests are required
  - Experimental or "work in process" extensions
  - Could become KHR if proved to be useful
- Vendor extensions:
  - Only available on the vendor's OpenCL platforms
  - Less restrictive
  - Could be evolved into EXT or KHR extensions.

Extensions in Qualcomm OpenCL SDK

Scope/impact of different extensions varies

- Large extensions:
  - A comprehensive set of API and kernel functions to enable new use cases
  - Machine learning extension, image filter functions, recordable queue, etc.
- Kernel extensions:
  - Provide kernel functions to expose specific hardware capabilities
  - Bit-reverse, required_subgroup_size, etc.
- OS/SW centric extensions:
  - Provide new software and OS dependent features
  - Zero copy, Android, security, etc.

- [List of Extensions at Adreno GPU SDK - Tools - Qualcomm Developer Network]()



cl_qcom_accelerated_image_ops.txt
cl_qcom_android_ahardwarebuffer_host_ptr.txt
cl_qcom_android_native_buffer_host_ptr.txt
cl_qcom_bitreverse.txt
cl_qcom_compressed_image.txt
cl_qcom_create_buffer_from_image.txt
cl_qcom_dot_product8.txt
cl_qcom_ext_host_ptr.txt
cl_qcom_ext_host_ptr_iocoherent.txt
cl_qcom_extended_query_image_info.txt
cl_qcom_extract_image_plane.txt
cl_qcom_filter_bicubic.txt
cl_qcom_ion_host_ptr.txt
cl_qcom_ml_ops.txt
cl_qcom_other_image.txt
cl_qcom_perf_hint.txt
cl_qcom_priority_hint.txt
cl_qcom_protected_context.txt
cl_qcom_recordable_queues.txt
cl_qcom_reqd_sub_group_size.txt
cl_qcom_subgroup_shuffle.txt
cl_qcom_vector_image_ops.txt

Caveats for Using Extensions

- Query its availability before using an extension: to see if the extension string is returned for the following function calls:
  - Platform extension
    - *cl_platform_info CL_PLATFORM_EXTENSIONS* using *clGetPlatformInfo*
  - Or device extension
    - *cl_device_info CL_DEVICE_EXTENSIONS* using *clGetDeviceInfo*
- In case an extension is unavailable:
  - It may be adopted into core spec (KHR=>core, Vendor => KHR, etc.), or
  - It may be deprecated/defeatured
- For better compatibility or portability, use KHR instead of vendor extension if they provide similar functionalities

Machine Learning Extension

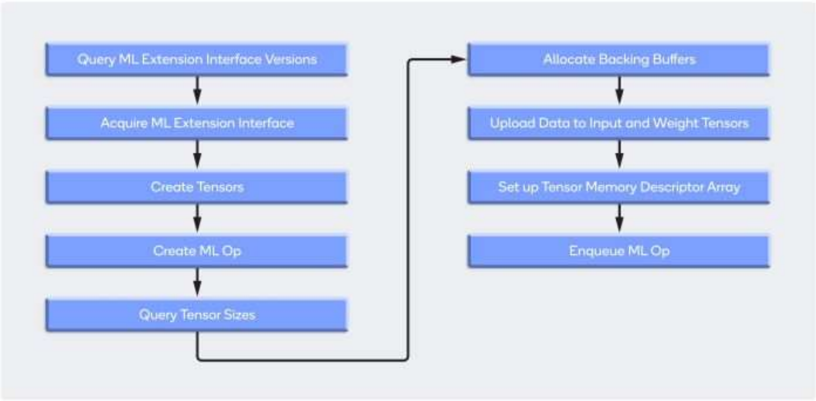
cl_qcom_ml_ops

- Objective:
  - Provide a set of API functions and mechanism to accelerate common machine learning operations on Adreno GPUs, for both inference and training
- Key features:
  - Support most common ML ops
  - Separate memory allocation of tensors from its creation to minimize memory requirement
  - Works with standard OpenCL features such as command queues and events.
  - Seamlessly share resources and synchronize with standard OpenCL kernels.
  - Tools provided to convert from TensorFlow model to the one supported by the extension
- Advantage:
  - Built-in optimized kernels in the API functions: no need to write your own kernels.
  - Support both inference operators (v1), and training operators (v2)
- A machine learning SDK with sample models and documentation is available in SDK.

Machine Learning Extension

cl_qcom_ml_ops

- Common ops:
    - Convolution, *depth-wise separable convolution and fused convolution + activation*
    - Activation*: Relu, Sigmoid, Tanh, Relu6*
    - Pooling*: Max, Average*
    - *GEMM, Transpose, Fully Connected, Softmax,*
    - *Binary Operations*
    - Concatenation, *Depth to Space, Permute, Reshape, BatchNorm, Fill*
    - *ResizeBilinear, Pad, CopyTensor*
- More Ops will be added in future releases


Query ML Extension Interface Versions
Acquire ML Extension Interface
Create Tensors
Create ML Op
Query Tensor Sizes
Allocate Backing Buffers
Upload Data to Input and Weight Tensors
Set up Tensor Memory Descriptor Array
Enqueue ML Op

For more details, please see the following blogs and the SDK:

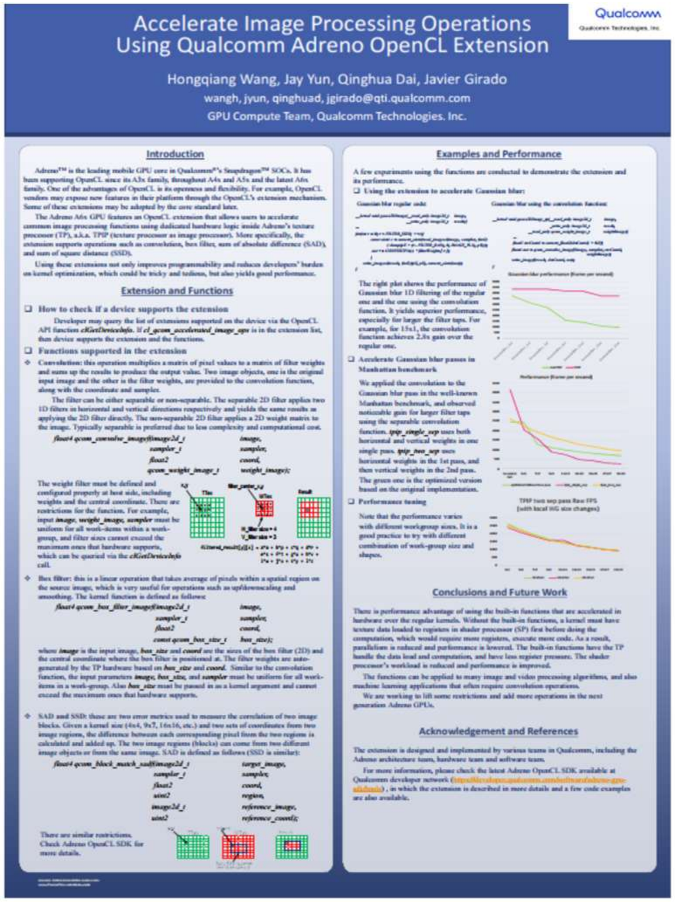
- OpenCL Machine Learning Acceleration on Adreno GPU - Qualcomm Developer Network: OpenCL Machine Learning Acceleration on Adreno GPU - Qualcomm Developer Network
- ML training at the edge: Training on mobile devices - Qualcomm Developer Network: ML training at the edge: Training on mobile devices - Qualcomm Developer Network

Advanced Image Processing Functions

cl_qcom_accelerated_image_ops

- Introduced a set of OpenCL-C built-in functions for imaging operations:
  - Convolution: separable/non-separable
  - Box filtering
  - SAD/SSD for block matching.
- Defined procedures and data structures for creating image and data objects required by the new built-ins.
- Demonstrated good performance and power benefits:
  - With the extension, 15x1 Gaussian Blur achieves up to 2.8x gain over the regular one.

- Refer to the 2019 IWOCL poster:



Accelerate Image Processing Operations
Using Qualcomm Adreno OpenCL Extension

Hongqiang Wang, Jay Yun, Qinghua Dai, Javier Girado
wangh, jyun, qinghuad, jgirado@qti.qualcomm.com
GPU Compute Team, Qualcomm Technologies. Inc.

Introduction

Extension and Functions

Examples and Performance

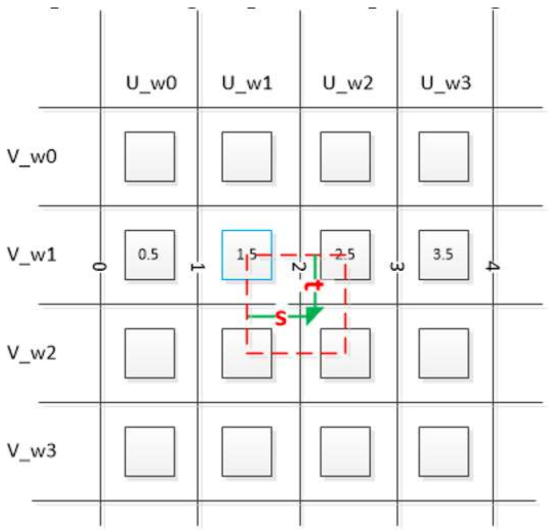
Conclusions and Future Work

Acknowledgement and References

Bicubic filter support for 2D images

cl_qcom_filter_bicubic

- When the filter mode is *QCOM_CLK_FILTER_BICUBIC*, a *4x4* square of image elements for the 2D image and the weights of each pixel are determined based on the coordinates and relative locations in the image.
- The sampler, which requires *CLK_NORMALIZED_COORDS_TRUE,* can be either passed as kernel argument or specified inside the kernel, like the regular one.
- For details on how weights/coordinates are determined, please refer to the full documentation.



Extended Image Functions Support

YUV, compression, and read/write of single components of multiple pixels

- *cl_qcom_other_image:*
  - Read from and/or write to non-conventional OpenCL image objects.
  - YUV images such as NV12
  - MIPI packed images, Bayer pattern images and tiled images.
- *cl_qcom_compressed_image:*
  - Enable an app to read from and write to OpenCL image objects holding compressed image data.
- *cl_qcom_vector_image_ops:*
  - Introduces a new set of OpenCL-C built-in functions for reading and writing a group of OpenCL image elements in a single operation.
  - These built-ins can offer performance gains as well as ease of development.

Recording sequences of kernel enqueues

cl_qcom_recordable_queues

- Introduces a new set of procedures for recording sequences of kernel enqueues,
  - A sequence only needs to be generated once but can be dispatched multiple times.
- Any argument to any kernel in a recording may be modified without having to re-record the entire command sequence.
- For apps that repetitively enqueue a fixed sequence kernels with only minor changes to arguments, recordable queues can provide savings in CPU power and improve dispatch latency.
- This is used in ML extensions to reduce the enqueueing overhead

Extensions for Performance and Usability

cl_qcom_reqd_sub_group_size

- Adreno GPUs typically support two different subgroup sizes, half-wave size and full-wave size.
  - A kernel can run at either mode, while likely result in different performance.
  - Typically, compiler automatically chooses the optimal mode.
  - Half-wave typically works better for kernels with 16-bit ALU ops.
- This extension allows kernels to override the default one.
- To use this extension, the wavesize attribute can be added to the kernel:

  __attribute__ ((qcom_reqd_sub_group_size(“MODE")))

  __kernel void full_sub_group_kernel(...) { ......}

  where MODE can be *half* or *full*.

Extensions for Performance and Usability

cl_qcom_perf_hint and cl_qcom_priority_hint

- cl_qcom_perf_hint:
  - Allows an app to request the perf level desired for device(s) on an OpenCL context.
  - Higher performance implies higher frequencies on the device.
  - Three levels of performance hint: high (by default), normal, and low.
  - Example:

```
cl_context_properties properties[] = {CL_CONTEXT_PERF_HINT_QCOM, CL_PERF_HINT_LOW_QCOM, 0};
clCreateContext(properties, 1, &device_id, NULL, NULL, NULL);
clSetPerfHintQCOM(context, CL_PERF_HINT_NORMAL_QCOM);
```

- cl_qcom_priority_hint:
  - Allow an app to specify the desired priority for enqueued kernels to be submitted to the device(s) on an OpenCL context.
  - Three levels of priorities: high, normal (by default), and low.
  - Example:

```
cl_context_properties properties[] = {CL_CONTEXT_PRIORITY_HINT_QCOM, CL_PRIORITY_HINT_HIGH_QCOM, 0};
clCreateContext(properties, 1, &device_id, NULL, NULL, NULL);
```

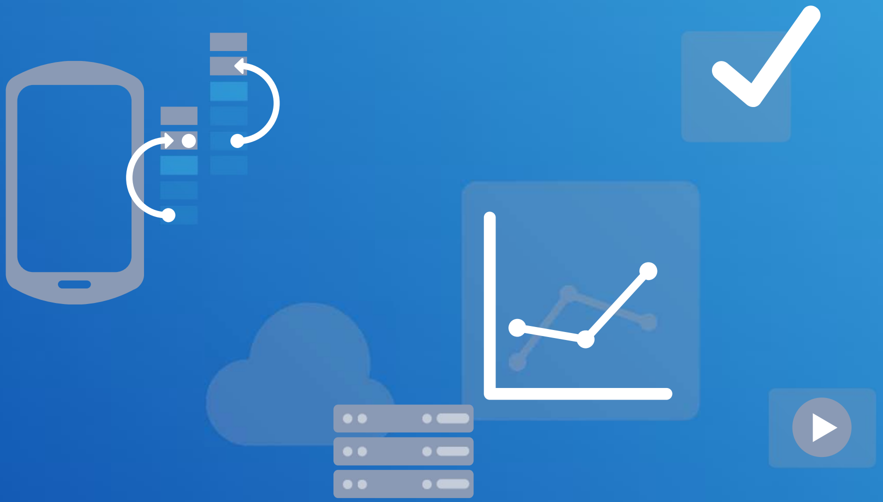
API and OS Dependent Extensions

- Several extensions as zero copy, memory object sharing across APIs or different layers of OSs, and the ability to securely access protected content, etc.
- For more details, please refer to the SDK document for these extensions:
  - *cl_qcom_ion_host_ptr,*
  - *cl_qcom_ext_host_ptr,*
  - *cl_qcom_android_native_buffer_host_ptr,*
  - *cl_qcom_dmabuf_host_ptr,*
  - *cl_qcom_android_ahardwarebuffer_host_ptr,*
  - *cl_qcom_ext_host_ptr_iocoherent,* and *cl_qcom_protected_context.*

Summary

- A rich set of vendor extensions are available in Qualcomm Adreno OpenCL SDK
- Detailed documentation and code examples are available in the SDK
- Some of the extensions may have KHR versions soon (recordable command queue, etc.)
- Any comments/feedbacks are welcome

Q&A



Thank you

Follow us on: f 🐦 in t

For more information, visit us at:
www.qualcomm.com & www.qualcomm.com/blog