Qualcomm

Qualcomm Technologies, Inc.

Offloading and Optimization of Adaptive Loop Filter of H.266 (VVC) on Qualcomm® Adreno™ GPUs

80-NB295-21 Rev. AA

February 2, 2024

Revision history

Revision	Date	Description
AA	February 2024	Initial release

Contents

Figures

Tables

1 Introduction to Adreno GPUs and OpenCL

With the full support of the latest general purpose computing standard, OpenCL 3.0, Adreno GPUs in Snapdragon® SoCs allow developers to fully leverage the GPU computing power without prior knowledge of graphics APIs. Meticulously designed OpenCL applications can take advantage of the full computing power of Adreno GPUs and run concurrently with graphics applications without affecting graphics rendering tasks such as UI. OpenCL on Snapdragon has been highly successful since its debut more than a decade ago: many smartphones powered by Snapdragon SOCs have OpenCL applications running behind the scenes for image, video, or compute vision, or machine learning workload.

To know more about how to develop, optimize, and profile OpenCL on Adreno GPUs, please refer to the latest OpenCL programming guide at https://developer.qualcomm.com/qfile/33472/80-nb295-11_a.pdf, and the Adreno OpenCL SDK at https://developer.qualcomm.com/software/adreno-gpu-sdk. In addition, blogs on how to optimize OpenCL on Snapdragon SOCs using specific examples are available at https://developer.qualcomm.com/blog/opencl-optimization-stop-leaving-compute-cycles-table, https://developer.qualcomm.com/blog/opencl-optimization-accelerating-sobel-filter-adreno-gpu, and https://developer.qualcomm.com/blog/accelerate-your-models-our-opencl-ml-sdk. There are also video tutorials available, e.g., https://www.youtube.com/watch?v=P0ljnmRIHJw.

2 Offloading ALF of VVC to GPU

2.1 What is VVC?

Versatile Video Coding (VCC/H.266) is a video standard finalized by the JVET in 2020. As a successor to the successful video codec, HEVC/H.265, VCC targets both compression efficiency and a broad range of applications. As of today, a software-based VCC decoder is the only viable solution on smartphone as the hardware-based decoder is not available due to a range of factors. The hardware accelerated VCC decoder on Snapdragon is in progress.

2.2 Why Offload to GPU?

It is natural to offload parts of the VVC to Adeno using OpenCL so that the VVC decoder takes full advantage of the computing power of a SoC. As VVC is very computationally demanding. GPU is a particularly excellent choice to offload the workload thanks to the following considerations:

1. Adreno GPU supports OpenCL.
    a. OpenCL is easy to program, debug, and profile thanks to all the resources available for Adreno, including Adreno OpenCL programming guide, Snapdragon profiler/debugger, and Adreno SDK examples.
    b. OpenCL has exceptionally good portability as all mainstream GPU vendors have adopted it.
    c. Many vendor extensions are available to allow developers to fully leverage Adreno's advanced features.
2. Adreno has superior computing power vs CPU.
    a. Adreno in premium tier SoCs features massive parallel SIMD computing engines.
    b. Adreno shares the same memory bandwidth as CPU and support the full 64-bit memory addressing.
    c. Adreno supports a wide variety of data types natively, including FP32, FP16, INT32, INT16, and INT8 (dot product), etc.
    d. Its FP16's peak performance is twice of FP32's.
3. Adreno offers superior power/energy performance advantage vs CPU.
    a. Adreno is usually running at much lower clock rates than CPU.
    b. Adreno has advanced mechanism to lower clock rates to conserve power if necessary.
    c. Adreno has on-chip memory and large cache that reduces memory traffic to the system memory.

2.3 How to Design Pipeline?

The partition of the workloads to CPU and GPU in a software decoder should follow the following guidelines:

1. The pipeline should be designed to minimize the synchronization overhead between CPU and GPU.
    a. Pipeline should avoid excessive wait between CPU and GPU.
    b. Ideally, they should reach the same point without redundant wait time.
2. The utilization of the cores should be as high as possible.
    a. The workload of a portion cannot be too small, as GPU is good at handling massive parallel data processing tasks while not good at handling too many small workloads.
    b. GPU has a software layer that roughly takes the same account of time regardless of the size of a workload. As a result, too many small workloads incur excessive software overhead as well as lower GPU's utilization.
    c. The decoding of a frame can be divided into multiple portions/lines and within each portion/line, CPU and GPU should work in a pipelining fashion.

2.4 Test Environment

Tencent266Dec, an in-house developed real-time VVC decoder by Tencent, is designed to support a variety of platforms, including PCs and mobile devices[1,2], and was customized and optimized for GPU based heterogeneous optimization, which served as the code base of this guideline and related industry demonstration. For instance, the line-based filter process enables simultaneous operation of the GPU and CPU, effectively masking runtime latency.

In this guideline, we will briefly outline the collaborative optimization efforts for H.266/VVC between Qualcomm and Tencent, providing guidance on GPU heterogeneous optimization. Owing to Tencent266Dec's real-time decoding performance plus the huge speedup from heterogeneous optimization, the world's first 4K10bit60fps VVC playback on Qualcomm Soc was demonstrated in ChinaJoy 2023. The test results in the following sections are also based on Tencent266Dec. For other H.266/VVC decoders, developers can utilize the methods described to optimize their H.266/VVC decoders for heterogeneous acceleration.

2.5 What Parts to Offload?

After reviewing the whole pipeline of VVC, the adaptive loop filter (ALF) in VVC is a natural choice for offloading into GPU, as this part is relatively more parallel friendly than the other stages in VVC and the workload itself is reasonably heavy for GPU processing. Therefore, ALF is used in this document to illustrate the OpenCL optimizations. This does not exclude the possibility of offloading other parts of VCC into GPU, e.g., interpolation filtering.

We started with an initial version of ALF that is far from optimized, as shown in the following table, that the GPU version is even slower than the CPU version.

Table 2-1 Initial latency (pure CPU v.s. CPU+GPU)

	CPU	CPU + GPU
single-frame	68ms	280ms

As shown in the following figure, there are three steps in the ALF using three GPU kernels (K1, K2, and K3). We will give an overview of key optimizations without too many low-level details.

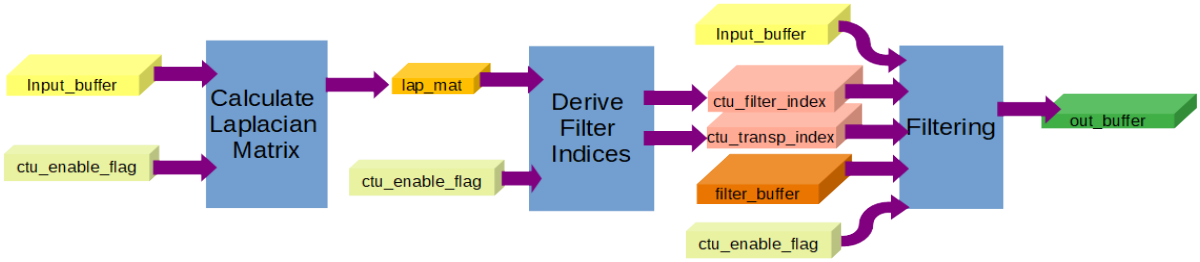


Figure 2-1 Adaptive loop filter (ALF) of H.266 (VVC)

3 Optimizations

This section presents the major OpenCL optimizations that we have done to achieve the performance target. We assume by now the developers should have read thoroughly the *Adreno OpenCL Programming Guide*, and have firsthand experience and feel comfortable on development, debugging, and profiling of OpenCL applications.

Developers need to be aware that the optimization of an OpenCL application has multiple levels, including algorithm level, host/API level, and kernel level optimization, and typically requires multiple rounds of trials and errors to achieve the performance target.

In this document, we focus on kernel optimizations. Before diving into details, one API level feature we like to highlight is an extension called *recordable command queue*, an OpenCL vendor extension recently created for Adreno. The extension is to record a serial of OpenCL kernel calls and then replay for the following workload that has the same set of OpenCL kernel calls with minor changes, such as kernels parameters. It is an extremely useful feature as the extension significantly reduces the software overhead for repetitive data processing like streamed video data processing. Please refer to the Adreno OpenCL programming guide and SDK for more details.

3.1 Workgroup Size Tuning

Tuning of workgroupsize is extremely important: each optimization described in this document should come with a new round of workgroup size tuning. Please refer to the programming guide on why tuning is important.

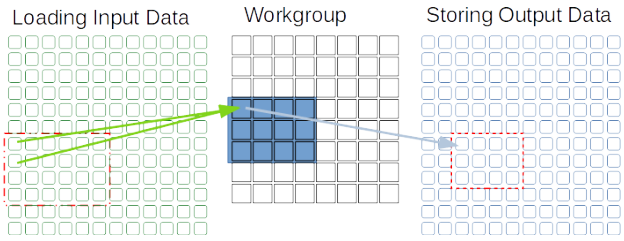


Figure 3-1 Workgroup, workgroup size, and load/store access patterns

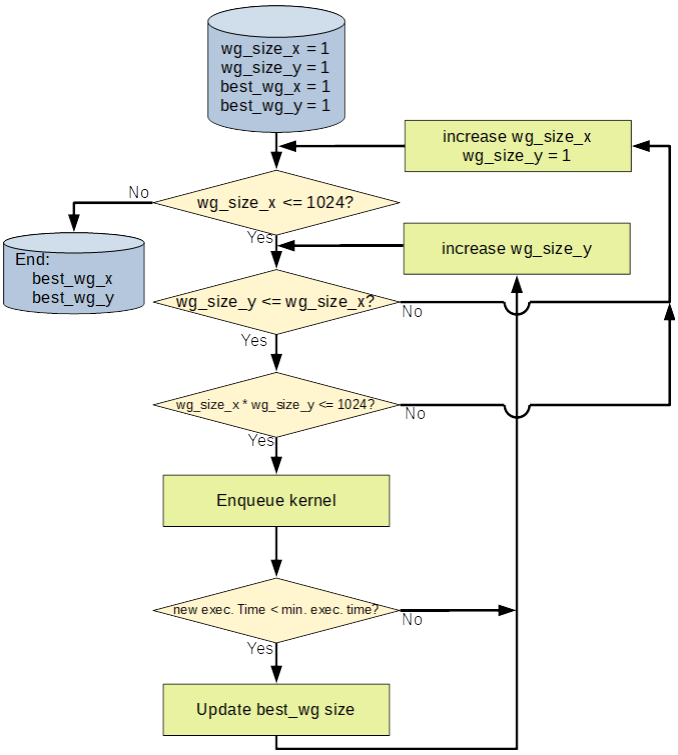


Figure 3-2 Workgroup size turning process

Figure 3-1 shows an example of load/store patterns(red boxes) and workitems in the middle. A grid scan on all WG sizes is shown in Figure 3-2.

3.2 Data Type/ALU Degradation

Data types are extremely important as, (1) it affects the memory traffic between memory and GPU, and (2) the computing capabilities of Adreno GPUs may vary based on different data types.

Figure 3-3 Downgrading data types

By using shorter data type for the arguments, input buffer, and many other variables, we were able to improve the parallel efficiency (reduced register footprint) and increase the ALU throughputs.

Figure 3-4 Downgrading ALU instructions (assuming c is calculated as 32 and invariant during the kernel execution)

Converting arithmetic to bit-wise operations and changing control flows to conditional operator ternary also benefit the performance significantly.

3.3 Constant Memory

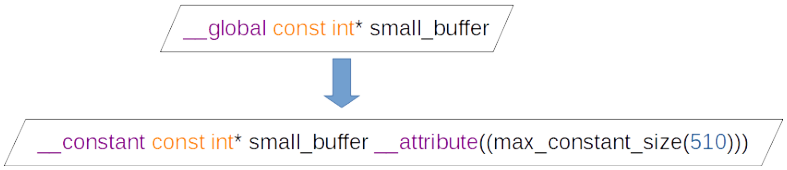


Figure 3-5 Promote global arrays to constant memory for preloading

Taking advantage of the broadcasting and preloading of constant member with an attribute, `max_const_size` dramatically improve performance. For example, `ctu_enable_flag_buffer` was a char array of length 510 bytes, which can be preloaded into constant memory which reduces data traffic and improve ALU utilization.

3.4 Access Pattern Simplification

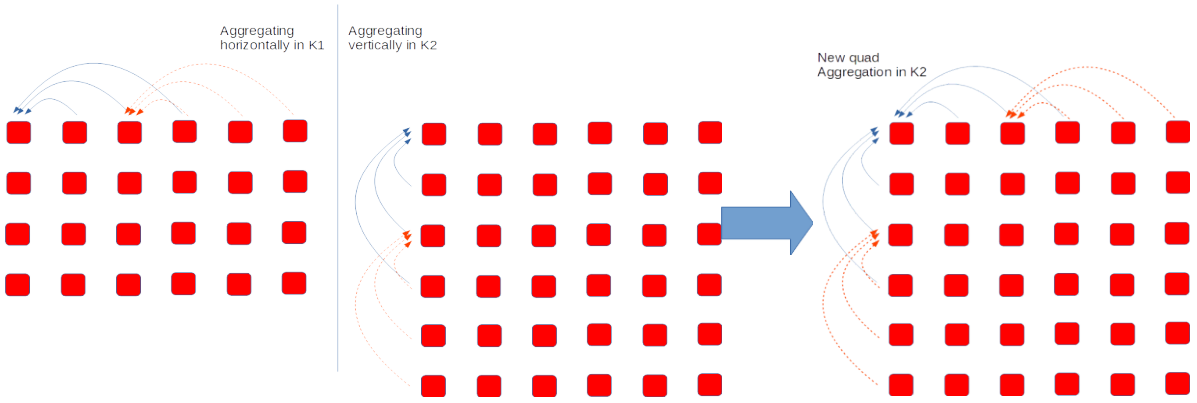


Figure 3-6 Moving similar access patterns into the same kernel.

As shown on the left of Figure 3-6, we found the access pattern of the Laplacian matrix accumulation in K1 was similar to the one in K2 while using the Laplacian matrix, so we move the accumulation to K2 to simplify K1's data access pattern for better cache hit rate. This allows K1 to use larger WG size to fully utilize shaders without increasing cache thrashing.

3.5 Data Packing

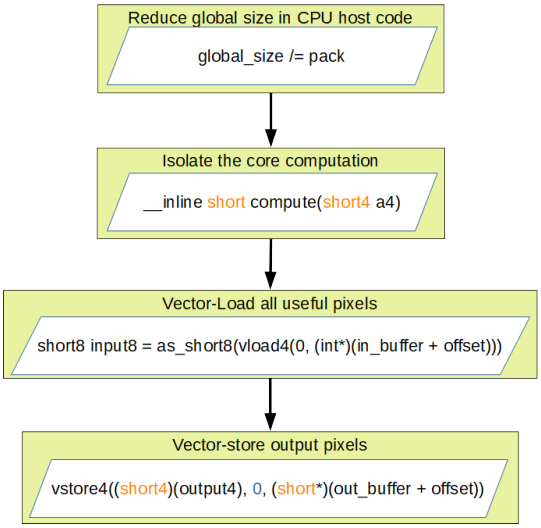
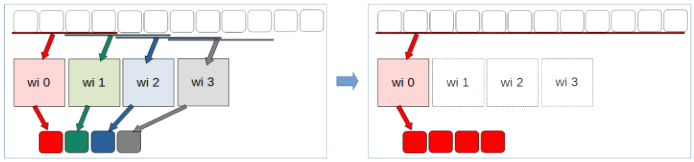


Figure 3-7 Packing input and output data.

Figure 3-7 shows the process of packing more workload into a single workitem. Better data packing improves the ALU utilization based on the Snapdragon profiler.

The following figure illustrates a toy example to pack four workitems' workload into one by loading twelve pixels and storing four output pixels at once.




```
short8 input8 = as_short8(vload4(0, (int*)(in_buffer + offset)))
short4 input4 = vload4(0, (int*)(in_buffer + offset + 8))

short4 a4 = input8.s0123
short out_a = compute(a4)

short4 b4 = input8.s2345
short out_b = compute(b4)

short4 c4 = input8.s4567
short out_c = compute(c4)

short4 d4 = (short4)(input8.s67, input4.s01)
short out_d = compute(d4)

vstore4((short4)(out_a, out_b, out_c, out_d), 0, (short*)(out_buffer + offset))
```

Figure 3-8 Example of data packing

3.6 Computation Offloading

Figure 3-9 Offloading arithmetic for invariants.

As shown in Figure 3-9, Identifying the computation of invariants and offloading them to CPU saved unnecessary computing cycles for GPUs.

3.7 Avoid Redundant Loads/Stores

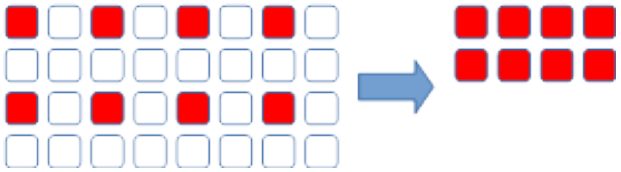
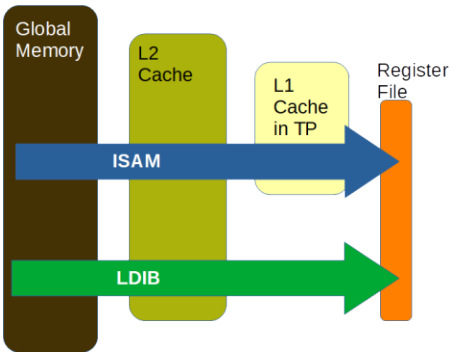


Figure 3-10 Avoid loading/storing zeros (shown in white boxes).

By studying the algorithm, we found that the pixels with real values in the Laplacian matrix have considerable number of zero pixels around. Removing these zeros greatly reduces memory traffic between global memory and GPU.

3.8 Replace Buffer with Texture Object



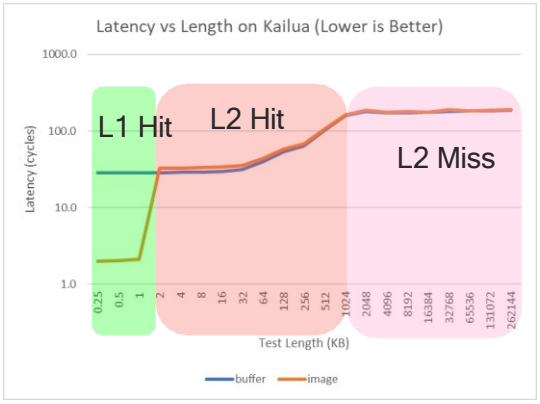


Figure 3-11 image vs buffer object

The initial kernel loaded all data using buffer objects, which provides flexibility for arbitrary indexing. However, the access patterns of some input arrays of K3 are aligned every four pixels and using texture instead of buffer leads to better performance as texture can leverage the L1 cache.

The following figure shows the in-kernel modification.

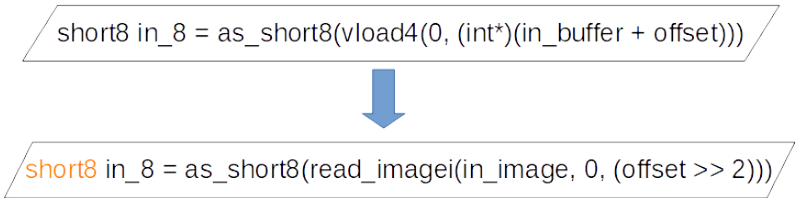


Figure 3-12 Load data with read_image

4 Summary

Based on the test environment Tencent266Dec, Offloading ALF into GPU in VVC shows that averaged latency on multi-frame streams achieved 17.4% to 23.87% perf uplift as compared against a pure CPU implementation (for the whole frame decoding).

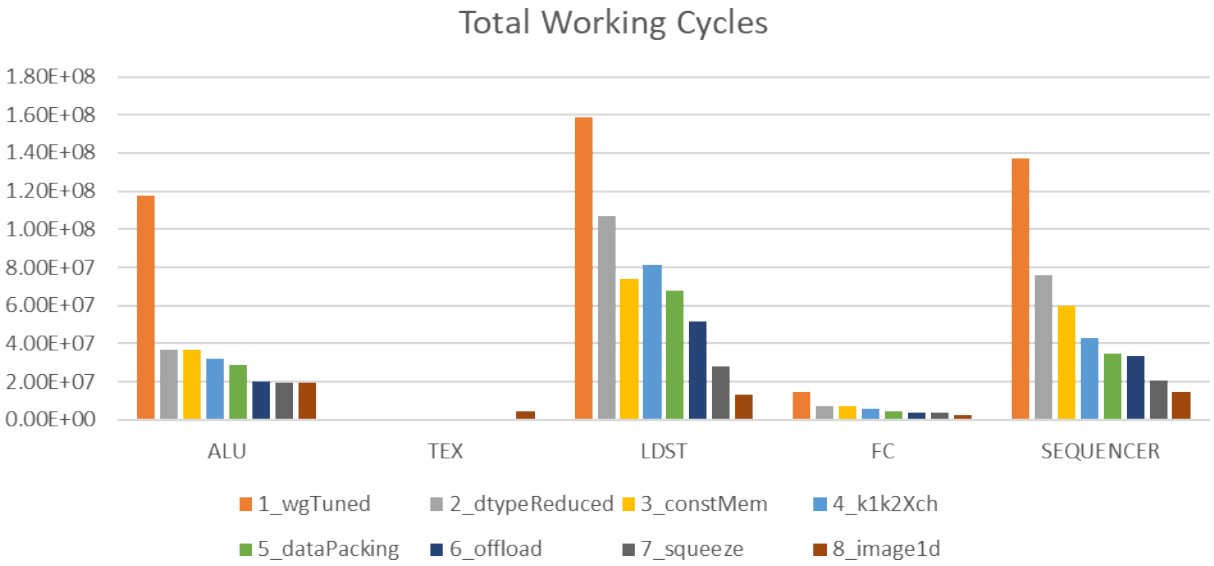


Figure 4-1 The Working Cycles after the eight optimizations

Throughout the optimizations, we profiled the kernels to understand the impact of each modification and verified whether the benefit reflected our knowledge and intention. Figure 4-1 shows a profiling results of the steps.

The following table lists all optimizations and their impacts we explored with Tencent266Dec as code base.

Table 4-1 Summary of all optimizations

	Description	Impact
1 wgTune	Workgroup size tuning	81.6%
2 dtypeRedu	Downgrading datatypes	43.6%
3 constMem	Promoting small buffers to constant memory	21.2%
4 k1k2Xch	Simplifying K1 access pattern by moving the last block of K1 to K2	27.6%
5.1 vload	Working toward data packing. Applying vload and vstore for adjacent pixels' load/store.	(accumulated to 5.3)
5.2 vload4	Replacing vload3 with vload4, vstore3 with vstore4.	(accumulated to 5.3)
5.3 dataPacking	Packing multiple workitems' workloads into one.	19.4%

	Description	Impact
6 offload	Combining dependent invariants and offloading some computations to CPU.	5.2%
7 squeezing	Avoid loading/storing zeros in K1 and K2	37.2%
8 image1d	Texturizing K3's input buffers.	30.8%

5 Industry Demonstrations

Upon the implementation of GPU heterogeneous optimization, Qualcomm & Tencent have also developed a real-time 4K H.266/VVC player based on Tencent266Dec, utilizing the Adreno GPU on a Snapdragon 8 Gen 2 mobile processor. This H.266/VVC player can support stable, real-time playback of ultra-high-definition VVC content at 4K 10-bit 60 frames per second (FPS). It was showcased at ChinaJoy 2023 and International Broadcasting Convention (IBC) 2023. More details can be found in [3].

6 Contact Us

For further information or if you have any questions, please contact us by:

https://support.qualcomm.com/

Tencent Media Lab

medialab@tencent.com

A References

[1]. Y. Li, S. Liu, Y. Chen, Y. Zheng, S. Chen, B. Zhu and J. Lou, "An optimized H. 266/VVC software decoder on mobile platform." 2021 Picture Coding Symposium (PCS). IEEE, 2021.

[2]. B. Zhu, S. Liu, X. Xu, X. Zhang, C. Gu, L. Wang, W. Feng, “Performance of a VVC software decoder,” ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 document JVET-T0095, teleconference, October, 2020.

[3]. https://www.qualcomm.com/news/onq/2023/07/the-end-of-video-buffering-are-we-there-yet, 2023

LEGAL INFORMATION

Your access to and use of this material, along with any documents, software, specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this "Material"), is subject to your (including the corporation or other legal entity you represent, collectively "You" or "Your") acceptance of the terms and conditions ("Terms of Use") set forth below. If You do not agree to these Terms of Use, you may not use this Material and shall immediately destroy any copy thereof.

1) Legal Notice.

This Material is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. ("**Qualcomm Technologies**"), its affiliates and/or licensors described in this Material, and shall not be used for any other purposes. If this Material is marked as "**Qualcomm Internal Use Only**", no license is granted to You herein, and You must immediately (a) destroy or return this Material to Qualcomm Technologies, and (b) report Your receipt of this Material to qualcomm.support@qti.qualcomm.com. This Material may not be altered, edited, or modified in any way without Qualcomm Technologies' prior written approval. Unauthorized use or disclosure of this Material or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and/or licensors for any such unauthorized uses or disclosures of this Material, in whole or part.

Qualcomm Technologies, its affiliates and/or licensors retain all rights and ownership in and to this Material. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Material or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Material.

THIS MATERIAL IS BEING PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND/OR LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES AND/OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS MATERIAL.

Certain product kits, tools and other items referenced in this Material may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Material may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Material is an offer to sell any of the components or devices referenced herein.

This Material is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com or the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through, including, without limitation any non-disclosure agreement) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate and/or licensor with respect to Your access to and use of this Material, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Material may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Material are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.