AMD APP SDK 3.0

FAQ

1 General Questions

1. **Do I need to use additional software with the SDK?**

   For information about the additional software to be used with the AMD APP SDK, see the AMD APP SDK Getting Started Guide.

   Also, we recommend using the debugging profiling and analysis tools contained in the AMD CodeXL heterogeneous compute tools suite.

2. **Which versions of the OpenCL™ standard does this SDK support?**

   AMD APP SDK 3.0 supports the development of applications using the OpenCL™ Specification version 2.0.

3. **Will applications developed to execute on OpenCL™ 1.2 still operate in an OpenCL™ 2.0 environment?**

   OpenCL™ is designed to be backwards compatible. The OpenCL™ 2.0 run-time delivered with the AMD Catalyst drivers run any OpenCL™ 1.2-compliant application. However, an OpenCL™ 2.0-compliant application will not execute on an OpenCL™ 1.2 run-time if APIs only supported by OpenCL™ 2.0 are used.

4. **How often can I expect to get AMD APP SDK updates?**

   Developers can expect that the AMD APP SDK may be updated two to three times a year. Actual release intervals may vary depending on available new features and product updates. AMD is committed to providing developers with regular updates to allow them to take advantage of the latest developments in AMD heterogeneous compute technology.

5. **What is the difference between the CPU and GPU components of OpenCL™ that are bundled with the AMD APP SDK?**

   The CPU component uses the compatible CPU cores in your system to accelerate your OpenCL™ compute kernels; the GPU component uses the compatible GPU cores in your system to accelerate your OpenCL™ compute kernels.

6. **What CPUs does the AMD APP SDK 3.0 support?**

   The CPU component of OpenCL™ bundled with the AMD APP SDK works with any x86 CPU with SSE3 or later, as well as SSE2.x or later. AMD CPUs have supported SSE3 (and later) since 2005.

   At the time of this writing, AMD CPUs support OpenCL 1.2. OpenCL 2.0 support may be added in the future.

7. **What APUs and GPUs does the AMD APP SDK 3.0 with OpenCL™ 2.0 support work on?**

   For the list of supported APUs and GPUs, see the AMD APP SDK System Requirements list at: http://developer.amd.com/AMDAPPSDK.

8. **Can my OpenCL™ code run on GPUs from other vendors?**

   At this time, AMD does not plan to have the AMD APP SDK support GPU products from other vendors; however, since OpenCL™ is an industry standard programming interface, programs written in OpenCL™ can be recompiled and run with any OpenCL-compliant compiler and runtime.

9. **What version of MS Visual Studio is supported?**

   The AMD APP SDK v3.0 with OpenCL™ 1.2 supports Microsoft® Visual Studio® 2010 Professional Edition, Microsoft® Visual Studio 2012, and Microsoft® Visual Studio 2013.

10. **Is it possible to run multiple AMD heterogeneous compute applications (compute and graphics) concurrently?**

    Multiple AMD heterogeneous compute applications can be run concurrently, as long as they do not access the same GPU at the same time. AMD heterogeneous compute applications that attempt to access the same GPU at the same time are automatically serialized by the runtime system.

11. **How does OpenCL™ compare to other APIs and programming platforms for parallel computing, such as OpenMP and MPI? Which one should I use?**

    OpenCL™ is designed to target parallelism within a single system and provide portability to multiple different types of devices (GPUs, multi-core CPUs, etc.). OpenMP targets multi-core CPUs and SMP systems. MPI is a message passing protocol most often used for communication between nodes; it is a popular parallel programming model for clusters of machines. Each programming model has its advantages. It is anticipated that developers mix APIs, for example programming a cluster of machines with GPUs with MPI and OpenCL™.

12. **If I write my code on the CPU version, does it work on the GPU version, or do I need to make changes?**

    Assuming the size limitations for CPUs are considered, the code works on both the CPU and GPU components. Performance tuning, however, is different for each.

13. **What is the precision of mathematical operations?**

    See Chapter 7, “OpenCL Numerical Compliance,” of the OpenCL™ 2.0 Specification for exact mathematical operations precision requirements.

14. **Are byte-addressable stores supported?**

    Byte-addressable stores are supported.

15. **Are long integers supported?**

    Yes, 64-bit integers are supported.

16. **Are operations on vectors supported?**

    Yes, operations on vectors are supported.

17. **How does one verify if OpenCL™ has been installed correctly on the system?**

    Run `clinfo` from the command-line interface. The clinfo tool shows the available OpenCL™ devices on a system.

18. **How do I know if I have installed the latest version of the AMD APP SDK?**

    On installation of the SDK, if an internet connection is available, the installer states whether or not a newer SDK is available in the file `VersionInfo.txt` in the directory `C:\Users\<username>\Documents\AMD APP SDK\<AMD APP SDK version>\`. Alternatively, you can check for the latest available version of the AMD APP SDK at http://developer.amd.com/appsdk.

19. **Does it require administrator permissions to install the AMD APP SDK?**

    On Windows, one needs administrator permissions to install the AMD APP SDK. On Linux, however, users with root as well as non-root permissions can install the AMD APP SDK.

20. **Can multiple versions of the AMD APP SDK co-exist on a system?**

    On Windows, the AMD APP SDK installer supports co-existence with earlier versions of the SDK. However on Linux, currently co-existence is not supported. The User must uninstall any existing version(s) of AMD APP SDK before installing a new version.

21. **What setups/environments (OS, driver, CPU etc.) has the AMD APP SDK 3.0 been tested on? Is my setup listed on the AMD APP SDK tested environment list?**

    See the System Requirements details on this page: http://developer.amd.com/appsdk.

22. **My application is running fine with the existing AMD APP SDK. Do I need to switch to the new AMD APP SDK?**

    The AMD APP SDK provides improved experience, updated samples, latest OpenCL development kit and optimized samples with every new release. These samples are validated to run on all the latest AMD hardware and the AMD Catalyst drivers. Hence it is recommended to update to the newer AMD APP SDK.

23. **Where can I find information about the limitations and known issues of the new AMD APP SDK?**

    For information about the limitations and known issues in the AMD APP SDK 3.0, see the *AMD APP SDK Sample Release Notes* document.

24. **I have a question that is not covered in FAQ.**

    For any questions related to AMD platforms, the AMD APP SDK, and OpenCL, you can post your queries on the AMD OpenCL Developer forum.

2 OpenCL™ Questions

25. What is OpenCL™?

OpenCL™ (Open Computing Language) is the first truly open and royalty-free programming standard for general-purpose computations on heterogeneous systems. OpenCL™ lets programmers preserve their expensive source code investment and easily target both multi-core CPUs and the latest GPUs, such as those from AMD.

Developed in an open standards committee with representatives from major industry vendors, OpenCL™ gives users a cross-vendor, non-proprietary solution for accelerating their applications on their CPU and GPU cores.

26. How much does the AMD OpenCL™ development platform cost?

AMD bundles support for OpenCL™ as part of its AMD APP SDK product offering. The AMD APP SDK is offered to developers and users free of charge.

27. What operating systems does the AMD APP SDK 3.0 support?

AMD APP SDK 3.0 runs on 32-bit and 64-bit versions of Windows and Linux. For the exact list of supported operating systems, see the AMD APP SDK 3.0 System Requirements list at: http://developer.amd.com/APPSDK.

28. Can I write an OpenCL™ application that works on both CPU and GPU?

Applications that program to the core OpenCL™ API and kernel language should be able to target both CPUs and GPUs. At runtime, the appropriate device (CPU or GPU) must be selected by the application.

29. Does the AMD OpenCL™ compiler automatically vectorize for SSE on the CPU?

The CPU component of OpenCL™ that is bundled with the AMD APP SDK takes advantage of SSE3 instructions on the CPU. It also takes advantage of the AVX instructions where supported. In addition to AVX, OpenCL™ math library functions also leverage XOP and FMA4 capabilities on CPUs that support them.

30. Does the AMD APP SDK 3.0 work on multiple GPUs (ATI CrossFire)?

OpenCL™ applications can explicitly invoke separate compute kernels on multiple compatible GPUs in a single system. The partitioning of the algorithm to multiple parallel compute kernels must be done by the developer. It is recommended that ATI CrossFire be turned off in most system configurations so that AMD heterogeneous compute applications can access all available GPUs in the system.

ATI CrossFire technology allows multiple AMD GPUs to work together on a single graphics-rendering task. This method does not apply to AMD heterogeneous compute computational tasks because it is not compatible with the compute model used for AMD heterogeneous compute applications.

31. Can I ship pre-compiled OpenCL™ application binaries that work on either CPU or GPU?

By using OpenCL™ runtime APIs, developers can write OpenCL™ applications that can detect the available compatible CPUs and GPUs in the system. This lets developers pre-

compile applications into binaries that dynamically work on either CPUs or GPUs that execute on targeted devices. Including LLVM IR in the binary provides a means for the binary to support devices for which the application was not explicitly pre-compiled.

32. Is the OpenCL™ double precision optional extension supported?

The Khronos and AMD double precision extensions are supported on certain devices. Your application can use the OpenCL™ API to query if this functionality is supported on the device in use.

33. Is it possible to write OpenCL™ code that scales transparently over multiple devices?

For OpenCL™ programs that target only multi-core CPUs, scaling can be done transparently; however, scaling across multiple GPUs requires the developer to explicitly partition the algorithm into multiple compute kernels, as well as explicitly launch the compute kernels onto each compatible GPU.

34. What should I do if I get wrong results on the Apple platform with AMD devices?

Apple handles support for the Apple platform; please contact them.

35. Is it possible to dynamically index into a vector?

No, this is not possible because a vector is not an array, but a representation of a hardware register.

36. What is the difference between `local int a[4]` and `int a[4]`?

`local int a[4]` uses hardware local memory, which is a small, low-latency, high-bandwidth memory; `int a[4]` uses per-thread hardware scratch memory, which is located in uncached global memory.

37. How come my program runs slower in OpenCL™ than in CUDA/Brook+/IL?

When comparing performance, it is better to compare code optimized for our OpenCL™ platform against code optimized against another vendor's OpenCL™ platforms. By comparing the same toolchain on different vendors, you can find out which vendors hardware works the best for your problem set.

38. Does prefetching work on the GPU?

Prefetch is not needed on the GPU because the hardware has a built-in mechanism to hide latency when many work-groups are running. The LDS can be used as a software-controlled cache.

39. How do you determine the max number of concurrent work-groups?

The maximum number of concurrent work-groups is determined by resource usage. This includes number of registers, amount of LDS space, and number of threads per work group. There is no way to directly specify the number of registers used by a kernel. Each SIMD has a 64-wide register file, with each column consisting of 256 x 32 x 4 registers.

40. Is it possible to tell OpenCL™ not to use all CPU Cores?

Yes, use the `device fission` extension. Also, OpenCL 1.2 introduces the sub-device concept. The CPU can be partitioned into sub-devices/cores and the program can choose which sub-device/core to use. The Device Fission feature of OpenCL is demonstrated in the **DeviceFission** AMD APP SDK sample.

41. Can two applications, one using OpenCL CPU and another using OpenCL GPU run in parallel?

Yes, they can run in parallel.

42. Can data transfer between the host and the device, and kernel execution run in parallel?

Yes. Data transfer and kernel execution can be overlapped. For an example, see the AMD APP SDK sample, AsyncDataTransfer.

2.1 OpenCL™ 2.0 Questions

43. Does an OpenCL 1.2 application run on the OpenCL 2.0 stack (driver) without any changes?

Yes, it does.

However, a few APIs are deprecated in OpenCL 2.0, such as atomic built-in functions for 32-bit integer and floating-point data types in the OpenCL 1.2 specification. For a complete set of deprecated APIs in OpenCL 2.0, see Section "F.3 Summary of changes from OpenCL 1.2" in the OpenCL 2.0 specification.

44. Do OpenCL 2.0 samples run on a CPU-only machine ?

Any device that supports OpenCL 2.0 can be used to execute the samples using OpenCL 2.0 APIs.
However, at the time of this writing, AMD CPUs do not support OpenCL 2.0.

45. Which operating systems support OpenCL 2.0 on AMD platforms?

For the list of operating systems with OpenCL 2.0 support on AMD platforms, see the "Compatible Operating Systems" section under the AMD Catalyst driver download page.

At the time of this writing, AMD supports OpenCL 2.0 on only 64-bit operating systems.

46. Which AMD devices (APU and discrete GPUs) support OpenCL 2.0 features ?

At present, the AMD Kaveri APU supports OpenCL 2.0.

All the AMD discrete GPUs with Sea Islands and above architectures support OpenCL 2.0. The AMD discrete GPUs with Southern Islands, Northern Islands, Evergreen and below architectures do not support OpenCL 2.0.

47. Is SVM Atomics supported on AMD platforms?

SVM Atomics is an optional feature of OpenCL 2.0.

At present, on AMD platforms, SVM Atomics is supported on only the Kaveri APU on Linux 64-bit OS.

48. What is new in OpenCL 2.0?

The OpenCL 2.0 specification is a significant evolution of OpenCL. It introduces features that allow closer collaboration between the host and OpenCL devices, such as Shared Virtual Memory (SVM) and device-side enqueue. Other features, such as pipes, generic address space, program scope variables and new image-related additions provide effective ways of expressing heterogeneous programming constructs.

49. Does OpenCL 2.0 support multiple graphics cards in one system?

At the time of this writing, on AMD platforms with multiple GPUs, only one GPU gets reported as OpenCL 2.0 though more than one are OpenCL 2.0 compatible.

50. Is there any AMD OpenCL best practices guide for APUs?

For details on optimizing applications on various AMD devices including APUs, see the *AMD OpenCL Optimization Guide*.

51. In what way is the SVM buffer different from a device buffer?

A device buffer cannot contain structures with pointers, as the host and the device do not share the same virtual address space. Also, the developer needs to explicitly manage the data transfer between the host and the device.

In the case of Shared Virtual Memory (SVM), the host and OpenCL devices share the same virtual address space. This allows structures with pointers to be used in SVM buffers. The developer need not explicitly manage copies of the data (although coarse-grain SVM buffers require a map/unmap on the host before any read/write to ensure memory consistency).

52. What kind of algorithms can benefit the most from device-side enqueue?

Device-side enqueue is suitable for applications that are inherently recursive and those that require dynamic work generation in the kernel. An example of the latter would be a tree search in which new nodes are discovered as the tree is traversed from the root to the leaves. Device-side enqueue eliminates the need for the application to return control to the host for relaunching a kernel when new work is found.

For additional insights on the device-enqueue feature, review the blog post on device-side enqueue and workgroup built-in functions. Also, review the BinarySearchDeviceSideEnqueue, DeviceSideEnqueueBFS and RegionGrowingSegmentation samples in the AMD APP SDK, which demonstrate the use of this feature.

53. What is best use-case for the Pipe feature in OpenCL 2.0?

A pipe is useful in scenarios in which the OpenCL application requires a queue-like structure. The pipe works as a First-In-First-Out (FIFO). It can be used in producer-consumer design kernels in which the producer writes to the Pipe object at one end of queue and the consumer reads from other end.

54. How is the OpenCL 2.0 memory model different from the OpenCL 1.2 memory model?

OpenCL 2.0 adopts the memory model defined in C++11 with some extensions. The memory orders taken from C++11 are: "relaxed", "acquire", "release", "acquire-release", and

"sequential consistent". For additional details on the OpenCL 2.0 memory model, see the *AMD OpenCL User Guide*.

55. How is data transfer managed for Coarse Grain SVM buffers on AMD platforms?

The AMD OpenCL runtime manages the transfer of data between the host and the OpenCL devices at synchronization points.

For coarse-grained SVM, the synchronization points are: the mapping or unmapping of the SVM memory and kernel launch or completion. This means that any updates are visible only at the end of the kernel or at the point of un-mapping the region of memory. The process is transparent to the programmer, who sees a unified address space of the SVM pointer.

56. What is the best form to share data between host and device: device buffers or coarse-grain SVM or fine-grain SVM?

The form of data buffer to be used depends on the data access pattern in the application. On AMD OpenCL 2.0 platforms, coarse-grain SVM buffers could be used instead of device buffers as the SVM buffers have the same virtual address space across the host and the device, allowing one to share pointer-based data structures with the host program. Using coarse-grain SVM buffers also ensures that the data transfer between host and device at synchronization points is transparent to the programmer.

In the case of fine-grain SVM buffers, sharing occurs at the granularity of individual loads/stores into bytes within OpenCL buffer memory objects. If OpenCL atomics is supported, the synchronization points include those defined for coarse-grained SVM as well as atomic operations. This means that updates are visible at the level of atomic operations on the SVM buffer. Fine-grain SVM is thus suitable for applications in which host and device need to simultaneously access the same memory.

In general, coarse-grain buffers provide faster access compared to fine grain buffers as the memory is not required to be consistent across devices.

57. I have an SVM buffer that contains a pointer to other SVM buffers. But the kernel is not able to access those SVM buffer pointers. Why?

As per the OpenCL 2.0 specification, any coarse-grain or fine-grain buffer SVM pointers used by a kernel that are not passed as a kernel arguments must be specified by using `clSetKernelExecInfo` with `CL_KERNEL_EXEC_INFO_SVM_PTRS`. This API is only required when the SVM buffer contains a pointer to another SVM buffer (i.e. other than the containing SVM buffer). It is not required when the pointer refers to an address within the same SVM buffer.

58. Will I observe any performance difference if I compile and run a kernel against OpenCL 1.2 and 2.0 using the same Catalyst driver?

There may be some difference in performance using OpenCL 2.0 compiler and OpenCL 1.2 compiler in the same 2.0 driver, as the compilers used on the AMD platform for 1.2 and 2.0 are different.

59. OpenCL2.0 introduces new features like SVM, device-side enqueue to provide better collaboration between host and devices. Are there any performance benefits to using these features? If so, how different is it on APU vs discrete GPU?

SVM and device-side enqueue features provide ease of programming as well as improved performance in selected applications. The performance benefits vary depending on the type of application and the data access pattern. For example, in the case of fine-grain atomics SVM, applications in which the host and the device have to work co-operatively, will derive performance benefits as they can simultaneously update the same buffer without depending on the kernel to complete to access the buffer. Similarly, using device-side enqueue instead of an iterative approach to launch kernels will give better performance as the overhead of transferring control back to host is eliminated.

At the time of this writing, AMD platforms support the optional OpenCL 2.0 feature, fine-grain SVM buffer, only on the AMD Kaveri APU on Linux 64-bit OS. Support on discrete GPUs is not currently available.

Device-side enqueue is supported on all AMD 2.0 compliant devices.

60. Can I enqueue a kernel on any device (including CPU if supported) other than the currently running device?

No. Device enqueue enqueues only to the device on which the parent kernel is running.

61. Can I use/run kernels compiled against different OpenCL versions in the same application?

Yes. As long as the hardware supports the highest version of OpenCL version that the application was compiled against, you can use and run kernels compiled against different OpenCL versions in the same application.

3 OpenCL™ Optimizations

62. How do I use constants in a kernel for best performance?

For performance using constants, highest to lowest performance is achieved using:

- Literal values
- Constant pointer with compile time constants indexing.
- Constant pointer with runtime constant indexing that is the same for all threads.
- Constant pointer with linear access indexing that is the same for all threads.
- Constant pointer with linear access indexing that is different between threads.
- Constant pointer with random access indexing.

63. Why are literal values the fastest way to use constants?

Up to 96 bits of literal values are embedded in the instruction; thus, in theory, there is no limit on the number of usable literals. In practice, the limit is 16K unique literals in a compilation unit.

64. Why does `a * b + c` not generate a mad instruction?

Depending on the hardware and the floating point precision, the compiler may not generate a mad instruction for the computation of `a * b + c` due to the floating-point precision requirements in the OpenCL™ specification. Here, developers who want to exploit the mad instruction for performance can replace that computation with the `mad()` built-in function in OpenCL.

65. What is the preferred work-group size?

The preferred work-group size on the AMD platform is a multiple of 64.

66. What is more efficient, the ternary operator `?:` or the `select` function?

The select function compiles to the single cycle instruction, `cmov_logical`; in most cases, `?:` also compiles to the same instruction. In some cases, when memory is in one of the operands, the `?:` operator is compiled down to an IF/ELSE block. An IF/ELSE block takes more than a single instruction to execute.

67. What is the difference between 24-bit and 32-bit integer operations?

24-bit operations are faster because they use floating point hardware and can execute on all compute units. Many 32-bit integer operations also run on all stream processors, but if both a 24-bit and a 32-bit version exist for the same instruction, the 32-bit instruction executes only one per cycle.

4 Hardware Information

68. How are 8/16-bit operations handled in hardware?

The 8/16-bit operations are emulated with 32-bit registers.

69. Do 24-bit integers exist in hardware?

No, there are 24-bit instructions, such as `MUL24`/`MAD24`, but the smallest integer in hardware registers is 32-bits.

70. What are the benefits of using 8/16-bit types over 32-bit integers?

8/16-bit types take less memory than a 32-bit integer type, increasing the amount of data you are able to load with a single instruction. The OpenCL™ compiler up-converts to 32-bits on load and down-converts to the correct type on store.

71. How often are wavefronts created?

Wavefronts are created by the hardware to execute as long as resources are available. If they are created but cannot execute immediately, they are put in a wait queue where they stay until currently running wavefronts are finished.

72. What is the maximum number of wavefronts?

The maximum number of wavefronts is determined by which resource limits the number of wavefronts that can be spawned. This can be the number of registers, amount of local memory, required stack size, or other factors. Developers are recommended to profile the

OpenCL application using CodeXL to figure out the resources that limit the number of wavefronts and that thereby affect the performance.

73. Why do I get blue or black screens when executing longer running kernels?

The GPU is not a preemptable device. If you are running the GPU as your display device, ensure that a compute program does not use the GPU past a certain time limit set by Windows. Exceeding the time limit causes the watchdog timer to trigger; this can result in undefined program results.

74. What is the cost of a clause switch? How can I reduce clause switches?

Clause switches are relevant only for Evergreen and Northern Island devices. In general, the latency of a clause switch is around 40 cycles.

Clause switches are almost directly related to source program control flow. The latency of a clause switch can be hidden by executing multiple wavefronts. Also, reducing source program control flow can reduce the clause switch latency.

75. How does the hardware execute a loop on a wavefront?

The loop only ends execution for a wavefront once every thread in the wavefront breaks out of the loop. Once a thread breaks out of the loop, all of its execution results are masked, but the execution still occurs.

76. How does flow control work with wavefronts?

There are no flow control units for each individual thread, so the whole wavefront must execute the branch if any thread in the wavefront executes the branch. If the condition is false, the results are not written to memory, but the execution still occurs.

77. What happens with out-of-bound memory operations?

Writes are dropped, and reads return a pre-defined value.

5 Bolt

78. What is Bolt?

Bolt is a C++ template library optimized for heterogeneous computing. Bolt is designed to provide high-performance library implementations for common algorithms such as scan, reduce, transform, and sort. The Bolt interface was modeled on the C++ Standard Template Library (STL). Developers familiar with the STL will recognize many of the Bolt APIs and customization techniques.

79. Where can I learn more about Bolt ?

You can learn more about Bolt on this page: https://github.com/HSA-Libraries/Bolt.

80. How to force the AMD APP SDK Bolt samples to run on a specific code path GPU/Multicore TBB/Serial CPU?

AMD APP SDK Bolt sample provides the "`--device`" option to run on GPU or Multi-core CPU(TBB) or Serial CPU. The usage is as follows:

For Bolt OpenCL samples:

```
<BoltSample>.exe --device        [auto|OpenCL|SerialCpu|MultiCoreCpu]
```

For Bolt C++ AMP Samples:

```
<BoltSample>.exe --device        [auto|GPU|SerialCpu|MultiCoreCpu]
```

81. Is installing TBB necessary ?

No. The TBB libraries are needed if one wants to make use of the multi-core CPU path supported in BOLT.

82. Do all the AMD APP SDK Bolt samples support running all 3 Bolt code paths GPU/Multicore TBB/Serial CPU?

Except for the introductory samples, BoltIntro and BoltAMPIntro, all the other AMD APP SDK Bolt samples support running in all 3 code paths GPU, Multicore TBB, and Serial CPU.

These Bolt projects include additional build configuration (Debug_TBB and Release_TBB) for enabling TBB path and have precompiler flag, "ENABLE_TBB" set. Binaries generated in this build configuration have "_TBB " as suffix and can be used to run in Multicore TBB path in addition to GPU and Serial CPU path. An additional requirement for running the multicore path is to have the environment variable TBB_ROOT, that points to install location of Intel TBB.

Samples compiled with regular Debug and Release configuration can only be used to run in GPU and Serial CPU modes.

83. Which GPU compute technologies are supported by Bolt?

AMD APP SDK 3.0 is aligned with Bolt 1.3 libraries. The GPU compute technologies supported by Bolt 1.3 are OpenCL and C++ AMP.

6 C++ AMP

84. What is C++ AMP?

C++ Accelerated Massive Parallelism (C++ AMP) accelerates execution of C++ code by taking advantage of data-parallel hardware, such as the compute units on and APU. C++ AMP is an open specification that extends regular C++ through the use of the `restrict` keyword to denote portions of code to be executed on the accelerated device.

85. What AMD devices support C++ AMP?

Any AMD APU or GPU that supports Microsoft® DirectX 11 Feature Level 11.0 and higher.

86. Can you run a C++ AMP program on a CPU?

You can run a C++ AMP program on a CPU, but its usage is not recommended in a production environment. For more information, see: http://blogs.msdn.com/b/nativeconcurrency/archive/2012/03/10/cpu-accelerator-in-c-amp.aspx

87. Where can I get support for C++ AMP?

C++ AMP is supported by the C++ compiler in Microsoft® Visual Studio® version 2012 and higher.

88. Is C++ AMP supported on Linux?

At this time, C++ AMP is not supported on Linux; however, as C++ AMP is defined as an open standard, Microsoft has opened the door for implementations on operating systems other than Windows.

89. Where can I learn more about C++ AMP?

There is an MSDN page on C++ AMP: http://msdn.microsoft.com/en-us/library/vstudio/hh265137.aspx. Also, the book *C++ AMP: Accelerated Massive Parallelism with Microsoft Visual C++*, by Kate Gregory and Ade Miller, may be useful.

7 OpenCV

90. Where to download OpenCV libraries from?

OpenCV can either be installed by downloading pre-builts or by building from sources from opencv.org.

91. Which version of OpenCV does AMD APP SDK 3.0 support?

AMD APP SDK 3.0 OpenCV samples require OpenCV 2.4.9 to be installed as a prerequisite. The Windows OpenCV 2.4.9 installation includes prebuilt binaries with OpenCL bindings.

92. Does OpenCV-CL APIs support interoperability with raw OpenCL kernels?

Yes. The CartoonFilter sample in the AMD APP SDK demonstrates it.

93. I am getting the opencv2/core/core.hpp: No such file or directory error while building.

Verify that the `OPENCV_DIR` environment variable is set/exported to the installed OpenCV directory and that the `OCVCL_VER` environment variable is set to OpenCV version number(249). For details, see the OpenCV section in the *AMD APP SDK Getting Started Guide* document.

94. I am not able to run the built examples. I am getting the "error while loading shared libraries: libopencv_core.so.2.x: cannot open shared object file: No such file or directory" error.

Export LD_LIBRARY_PATH (in Linux) or set PATH (in Windows) to the installed OpenCV `lib' directory.

95. I am getting the error - Unspecified error (The function is not implemented. Rebuild the library with Windows, GTK+ 2.x or Carbon support. If you are on Ubuntu or Debian, install libgtk2.0-dev and pkg-config, then re-run cmake or configure script).

The reason for this error could be that the above tools might not have been installed before building OpenCV. The tools are required for displaying the images (cv::imshow calls). Install the above packages and rebuild the OpenCV library.

96. Do the OpenCV samples depend on any other library apart from the OpenCV binaries and OpenCL?

The Gesture Recognition sample depends on an additional library called `OpenNI`. The `OpenNI` library is used to extract frames from the input video.

97. Where must one download the OpenNI libraries from?

The `OpenNI` libraries can be downloaded from: http://structure.io/openni.

98. What version of the OpenNI library is preferred?

The `OpenNI` library version `2.2.0.33 Beta` is preferred.

99. I am getting the "Cannot find OpenNI.h" or "Cannot find OpenNI2.lib" error.

For 32-bit builds, set `OPENNI2_INCLUDE` and `OPENNI2_LIB`, and for 64-bit builds, set `OPENI2_INCLUDE64` and `OPENNI2_LIB64`, to the corresponding include and library locations.

100. Where can I know more about the OpenNI APIs?

You can know more about the `OpenNI` APIs here: http://structure.io/openni.

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000



The contents of this document are provided in connection with Advanced Micro Devices, Inc. (“AMD”) products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. The information contained herein may be of a preliminary or advance nature and is subject to change without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD’s Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD’s products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD’s product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Copyright and Trademarks

© 2015 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ATI, the ATI logo, Radeon, FireStream, and combinations thereof are trademarks of Advanced Micro Devices, Inc. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos. Other names are for informational purposes only and may be trademarks of their respective owners.