Why did AMD open source ROCm's OpenCL driver-stack?

AMD open sourced the OpenCL driver stack for ROCm in the beginning of May. With this they kept their promise to open source (almost) everything. The hcc compiler was open sourced earlier, just like the kernel-driver and several other parts.

Why this is a big thing?
There are indeed several open source OpenCL implementations, but with one big difference: they're secondary to the official compiler/driver. So implementations like PortableCL and Intel Beignet play catch-up. AMD's open source implementations are primary.

It contains:

- OpenCL 1.2 compatible language runtime and compiler
- OpenCL 2.0 compatible kernel language support with OpenCL 1.2 compatible runtime
- Support for offline compilation right now - in-process/in-memory JIT compilation is to be added.

For testing the implementation, see Khronos OpenCL CTS framework or Phoronix openbenchmarking.org.

Why is it open sourced?

There are several reasons. AMD wants to stand out in HPC and therefore listened carefully to their customers, while taking good note on where HPC was going. Where open source used to be something not for businesses, it is now simply required to be

commercially successful. Below are the most important answers to this question.

Give deeper understanding of how functions are implemented

It is very useful to understand how functions are implemented. For instance the difference between sin() and native_sin() can tell you a lot more on what's best to use. It does not tell how the functions are implemented on the GPU, but does tell which GPU-functions are called.

Learning a new platform has never been so easy. Deep understanding is needed if you want to go beyond "it works".

Debug software

When you are working on a large project and have to work with proprietary libraries, this is a typical delay factor. I think every software engineer has this experience that the library does not perform as was documented and work-arounds had to be created. Depending on the project and the library, it could take weeks of delay - only sarcasm can describe these situations, as the legal documents were often a lot better than the software documents. When the library was open source, the debugger could step in and give the "aha" that was needed to progress.

When working with drivers it's about the same. GPU drivers and compilers are extremely complex and ofcourse your project hits that bug which nobody encountered before. Now all is open source, you can now step into the driver with the debugger. Moreover, the driver can be compiled with a fix instead of work-around.

Get bugs solved quicker

A trace now now include the driver-stack and the line-numbers. Even a suggestion for a fix can be given. This not only improves reproducibility, but reduces the time to get the fix for all steps. When a fix is suggested AMD only needs to test for regression to accept it. This makes the work for tools like CLsmith a lot easier.

Have "unimportant" specific improvements done

Say your software is important and in the spotlight, like Blender or the LuxMark benchmark, then you can expect your software gets attention in optimisations. For the rest of us, we have to hope our special code-constructions are alike one that is targeted. This results in many forums-comments and bug-reports being written, for which the compiler team does not have enough time. This is frustrating for both sides.

Now everybody can have their improvements submitted, giving it does not slow down the focus software ofcourse.

Get the feature set extended

Adding SPIR-V is easy now. The SPIRV-frontend needs to be added to ROCm and the right functions need to be added to the OpenCL driver. Unfortunately there is no support for OpenCL 2.x host-code yet - I understood by lack of demand.

For such extensions the AMD team needs to be consulted first, because this has implications on the test-suite.

Get support for complete new things

It takes a single person to make something completely new - this becomes a whole easier now.

More often there is opportunity in what is not there yet, and research needs to be done to break the chicken-egg. Optimised 128 bit computing? Easy complex numbers in OpenCL? Native support for Halide as an alternative to OpenCL? All high performance code is there for you.

Initiate alternative implementations (?)

Not a goal, but forks are coming for sure. For most forks the goals would be like the ones above, to later be merged with the master branch. There are a few forks that go their own direction - for now hard to predict where those will go.

Improve and increase university collaborations

If the software was protected, it was only possible under strict contracts to work on AMD's compiler infrastructure. In the end it was easier to focus on the open source backends of LLVM than to go through the legal path.

Universities are very important to find unexpected opportunities, integrate the latest research in, bring potential new employees and do research collaborations. Added bonus for the students is that the GPUs might be allowed to used for games too.

Timour Paltashev (Senior manager, Radeon Technology Group, GPU architecture and global academic connections) can be reached via timour dot paltashev at amd dot com for more info.

Get better support in more Linux distributions

It's easier to include open source drivers in Linux distributions. These OpenCL drivers do need a binary firmware (which were disassembled and seem to do as advertised), but the discussion is if this is part of the hardware or software to mark it as "libre".

There are many obstacles to have ROCm complete stack included as the default, but with the current state it makes much more chance.

Performance

Phoronix has done some benchmarks on ROCm 1.4 OpenCL in January on several systems and now ROCm 1.5 OpenCL on a Radeon RX 470. Though the 1.5 benchmarks were more limited, the important conclusion is that the young compiler is now mostly on par with the closed source OpenCL implementation combined with the AMDGPU-drivers. Only Luxmark AMDGPU was (much) better. Same comparison for the old proprietary fgrlx drivers, which was fully optimised and the first goal to get even with. You'll see that there will be another big step forward with ROCm 1.6 OpenCL.

Get started

You can find the build instructions here. Let us know in the comments what you're going to do with it!