The Windsor Build and Testing Framework (WBTF)

Shane Peelar (peelar@uwindsor.ca) and Paul Preney (preney@uwindsor.ca)

School of Computer Science, University of Windsor

Overview

The purpose of the Windsor Build and Testing Framework (WBTF) is to automate:

- **downloading** and **building** of **software** components in a platform-independent manner,
- **building** header-only, host-installed, or device-installed **code** using specific versions of Khronos software components, and,
- **testing/running** of such.

Supported software includes the:

- OpenCL ICD Loader [4],
- OpenCL C header files [3],
- OpenCL C++ Standard Library [2],
- Khronos Reference OpenCL C and OpenCL C++ compiler [5], and
- Khronos Reference LLVM Framework with SPIR-V support [1].

Overall Design

The WBTF is comprised of **two subsystems:**

1. the download-and-build subsystem (DABS), and,
2. the build-and-test subsystem (BATS).

DABS

The download-and-build subsystem (DABS) is used to:

- **download/update** various software **tools and libraries**, and,
- **build those tools** using a user-installed C/C++ compiler.

The result of the DABS is:

- the **installation** of various **OpenCL-related headers and libraries,** and,
- an installed **toolchain** capable of **compiling OpenCL C/C++ code.**

BATS

The build-and-test subsystem (BATS) uses the installations from the DABS to **build programs** and/or to **run conformance-style tests.**

BATS Configuration

BATS configurations are specified using identifiers for:

1. a **build system** that identifies the system code/tests are being run on,
2. a **host system** that identifies the system that will execute code/tests, e.g., "native" and "android",
3. a **device** that identifies the targeted OpenCL device attached to the system.

Such identifies a **test configuration** that defines parameters for the tests to be performed.

DABS and BATS in a Nutshell

DABS is installed/updated by running `build.sh` (Linux/Unix systems) or `build.bat` (Windows).

BATS test configurations are invoked by running `runtests.sh` (Linux/Unix systems) or `runtests.bat` (Windows) followed by a designed configuration file, e.g.,

```
runtests configs/compiletime-host-opencl-2.1.cmake
runtests configs/runtime-native-opencl-2.2.cmake
```

Ease-of-use

With a typical compiler toolchain installed, the WBTF:

- will work without any explicit configuration under Linux/Unix,
- may require some configuration under Windows, and,
- requires appropriate OpenCL runtime environments to be set up in advance for all device tests.

BATS OpenCL Test Types

BATS supports these types of tests:

1. a **run-time test**, i.e., a host program linked against an OpenCL implementation
2. a **header-only** or **compile-time test**, i.e., a C or C++ source file that #includes OpenCL headers whose resulting code after compilation is not executed. Compile-time checks are intended to check host/device code and/or headers for validity.

There are two types of compile-time tests:

1. a **host test** is a test that is run using the host compiler; and
2. a **device test** is a test that is run using an OpenCL C or OpenCL C++ offline (device) compiler.

NOTE: The ability to run device tests requires appropriately installed OpenCL implementations.

Future Work

Future work aims to:

- add software hooks to execute when code is deployed and run
- add additional support for some of the more common SDKs
- add support to deploy-and-run device tests on foreign architectures

Poster Presentation @ IWOCL, May 16-18, 2017.

Summary

Our tool makes it easy to download, install, and use The Khronos Group's OpenCL C/C++ compiler toolchain and libraries. This allows interested hobbyists, researchers, and professionals to develop and explore OpenCL and SPIR-V in addition to the downloaded sources. Although still a work-in-progress, the BATS permits those who have at least one OpenCL implementation available to run available Khronos conformance tests (e.g., OpenCL C++) and to explore how OpenCL software can be built including the use of the ICD.

References

[1] The Khronos Group. *LLVM Framework with SPIR-V Support*. 2016. URL: https://github.com/KhronosGroup/SPIRV-LLVM.git.

[2] The Khronos Group. *OpenCL C++ Standard Library Repository*. 2016. URL: https://github.com/KhronosGroup/libclcxx.git.

[3] The Khronos Group. *OpenCL Headers Repository*. 2016. URL: https://github.com/KhronosGroup/OpenCL-Headers.git.

[4] The Khronos Group. *OpenCL ICD Loader Repository*. 2015. URL: https://github.com/KhronosGroup/OpenCL-ICD-Loader.

[5] The Khronos Group. *SPIR Generator/Clang Compiler with OpenCL C and OpenCL C++ Support*. 2016. URL: https://github.com/KhronosGroup/SPIR.git.

Acknowledgements

We acknowledge and are very thankful for the support of The Khronos Group OpenCL Working Group as well as the support of Dr. Robert Kent and Dr. Ziad Kobti of the School of Computer Science, University of Windsor.

University of Windsor