CMake Build System in AOCL BLAS

Eleni Vlachopoulou

AMD

Agenda

Introduction to the new CMake system

Enabling CMake for new configurations

Future features

Q&A

Introduction to the new CMake system (1/2)

Objective: Have a robust CMake system that works on both Linux and Windows which is easy to maintain alongside the Make system.

Design: Have a CMake system which is as close as possible to Make system and reuse its functionality.

- Use config_registry to deduce config_list, kernel_list and kconfig_map.
- Use files in blis/build/ directory.
- Do the same filtering and “globbing” to create lists of files that need be compiled to generate the library.
- Use the same target names as much as possible.

Result: A new CMake system, available from AOCL 4.2.

- BLIS Windows-only CMake system existed in previous versions of AOCL, but it has been completely rewritten to adhere to the design above.
- Overcoming the limitations of previous system that required adding files by hand, had hardcoded flags, and did not use information in config_registry.

Introduction to the new CMake system (2/2)

- Parts of blis/configure, blis/Makefile and blis/common.mk have been ported to Python so that the functionality is platform independent.
- Python scripts along with other files are stored in blis/build/cmake directory.

- Quick overview:
  - read_registry.py is used to return the config and kernel lists, and the k-config map.
  - check_blatest.py and check_blitest.py check the output of the tests for failures. The difference to the Make system is that they return 0 or 1, for easy CI integration.
  - CMake system is based on object libraries which are then used to generate the static and shared libraries (similarly to object files in Make), which enables us to generate both libraries in a single step, like in Make system.
  - New files are picked up from CMake (as long as the configuration step is run) in a similar way to Make, no need to add the files one-by-one.

- Find more information in docs/CMakeBuildSystem.md or run “`cmake .. -DPRINT_CONFIGURE_HELP=ON`”.

Enabling CMake for new configurations

1. CMake will throw an error for builds where the requested configuration is not enabled. Enable a new configuration with a quick change in blis/CMakeLists.txt.
2. Add a make_defs.cmake file that corresponds to a make_defs.mk like below. This file is used to derive the correct flags that are used to build the different parts of the library.

make_defs.mk

git > blis-test > config > zen4 > make_defs.mk

```
78
79  # gcc or clang version must be at least 4.0
80  ifeq ($(CC_VENDOR),gcc)
81    GCC_VERSION := $(strip $(shell $(CC) -dumpversion | cut -d. -f1))
82
83    ifeq ($(shell test $(GCC_VERSION) -ge 13; echo $$?),0)
84      # gcc 13.0 or later
85      CKVECFLAGS += -march=znver4
86      CRVECFLAGS += -march=znver4
87      # Update CKOPTFLAGS for gcc to use O3 optimization without
88      # -ftree-pre and -ftree-partial-pre flag. These flag results
89      # in suboptimal code generation for instrinsic based kernels.
90      # The -ftree-loop-vectorize results in inefficient code gen
91      # for amd optimized l1 kernels based on instrinsics.
92      CKLPOPTFLAGS += -fno-tree-partial-pre -fno-tree-pre -fno-tree-loop-vectorize
93    else ifeq ($(shell test $(GCC_VERSION) -ge 11; echo $$?),0)
94      # gcc 11.0 or later
95      CKVECFLAGS += -march=znver3 -mavx512f -mavx512dq -mavx512bw -mavx512vl -mavx512vnni -ma
```

make_defs.cmake

git > blis-test > config > zen4 > make_defs.cmake

```
64
65  if("${CMAKE_C_COMPILER_ID}" STREQUAL "GNU")
66      if(CMAKE_C_COMPILER_VERSION VERSION_GREATER_EQUAL 13.0.0)
67          # gcc 13.0 or later
68          list(APPEND CKVECFLAGS -march=znver4)
69          list(APPEND CRVECFLAGS -march=znver4)
70          # Update CKLPOPTFLAGS for gcc to use O3 optimization without
71          # -ftree-pre and -ftree-partial-pre flag. These flag results
72          # in suboptimal code generation for instrinsic based kernels.
73          # The -ftree-loop-vectorize results in inefficient code gen
74          # for amd optimized l1 kernels based on instrinsics.
75          list(APPEND CKLPOPTFLAGS -fno-tree-partial-pre -fno-tree-pre -fno-tree-loop-vecto
76      elseif(CMAKE_C_COMPILER_VERSION VERSION_GREATER_EQUAL 11.0.0)
77          # gcc 11.0 or later
78          list(APPEND CKVECFLAGS -march=znver3 -mavx512f -mavx512dq -mavx512bw -mavx512vl -
79          list(APPEND CRVECFLAGS -march=znver3)
80          list(APPEND CKLPOPTFLAGS -fno-tree-partial-pre -fno-tree-pre -fno-tree-loop-vecto
```

Future features

- Use scripts to generate CMake preset files for new option combinations.
  - No need to remember the names of CMake variables, use something like “linux-make-clang-st-lp64-amdzen-static” instead.

- Enable FetchContent and functionality so that AOCL-BLAS can be built and used in external projects which require their dependencies to be built on demand.

- Enable find_package() functionality so that AOCL-BLAS can be integrated in CMake projects in a platform-independent fashion.

Questions?

DISCLAIMER AND ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD


AMD