Get started with BSbox-tools and use CMake

Dushan Bikov *
dusan.bikov@ugd.edu.mk

VERSION 0.3, updated August 20, 2024

Abstract

This guide explain how can install and start BSbox-tools software. Here also is presented use of CMake software for installation and build of BSbox-tools on Windows and Linux from source code.

Contents

1 Installing and starting BSbox-tools

BSbox-tools is CUDA C application and is intended for PC hardware configuration that contain NVIDIA graphic card with minimal 3.0 CUDA compute capability. It is recommended to use newer than 9.x and above, because some of the used functions are deprecated in the older CUDA versions. Even if PC do not contain NVIDIA graphic card program will work but without GPU optional features. This program can be run under Windows 7, 10, 11 and on some Linux distributions.

Source code, *.exe* file and documentation concerning BSbox-tools is available on: https://github.com/BoolSPLG/BSbox-tools_v0.3. BSbox-tools main branch of GitHub repository have two directories. The directory named **"BSbox-tools_v0.3_Release"** directory contain Windows build *.exe* file. The other directory **"BSbox-tools_v0.3 CMake"** contain BSbox-tools source code, documentation, CMake configuration (*CMakeLists.txt*) files and BoolSPLG library [4] necessary files. To be able to build BSbox-tools from the source code it is necessary BoolSPLG library to be installed [5]. This request is because BSbox-tools is developed on the foundation of the BoolSPLG library.

If you want to automate process of building BSbox-tools project, whether under Windows or Ubuntu (or other Linux distribution) you could use CMake.

1.1 Windows user

For installing just save (unpack the archive) files from the GitHub directory **"BSbox-tools_v0.3_Release"** in the separate directory on your local machine. You can run the software (.exe file) in the usual way.

Note. To be able to run and use BSbox-tools_v0.3 there is necessary to install Visual C++ Redistributive Packages (Visual Studio 2019).

*Faculty of Computer Science, Goce Delcev University, Stip, Macedonia

1.2 Linux user

To be able to build BSbox-tools under Ubuntu you need to have installed CUDA Toolkit. Toolkit installation contain **NVCC** proprietary compiler by Nvidia intended for use with CUDA. The CUDA compiler NVCC requires a compatible host compiler to be installed on the system [3]. Apart of CUDA Toolkit it is necessary have installed BoolSPLG library [5].

If you have proper installed and setting up CUDA Toolkit and BoolSPLG library it is quite easy to build BSbox-tools_v0.3 under Linux from the source code. First need to save (unpack the archive) the GitHub directory **"BSbox-tools_v0.3 CMake"** in the separate directory on your local machine. Into the terminal navigate working directory *"BSbox-tools"* containing source code. The build command is:

$$nvcc\ kernel.cu\ -o\ build\_name$$

Only write in terminal *./build_name* to run build file *build_name*. Command for building execution file can contain compiler optimisation options as "-O3" at etc.

2 CMake

CMake is free open-source BSD [1] license software, powerful cross-platform system of tools designed for build automation, testing, packaging and installation of software by using compiler-independent manner [2]. It is designed to be used in conjunction with the native build environment such as Make, Qt Creator, Android Studio, Microsoft Visual Studio, and etc. CMake supports directory hierarchies and uses independent simple configuration files called *CMakeLists.txt* placed in each source directory. These configuration files are used to generate native *makefiles* standard build files and workspaces that can be used to compile the source code in the native compiler environment by choice. CMake supports static and dynamic library builds, generates wrappers and build executables. It also supports in-place and out-of-place builds, which means that multiple builds can be derived from a single source tree. Convenient CMake feature is generation of cache file where information about files' locations, libraries, executables, and optional build directives is gathered. The cache file is designed to be used with a graphical editor. The gathered cache information my be changed by the user prior generating build files.

The design of CMake supports applications dependent on several libraries combined by complex directory hierarchies support that allow project to consist multiple toolkits. Each toolkit can contain several directories, and the application depends on the toolkits plus additional code. The CMake can handle various situations, for instance there is situation where executables must be built in order to generate code which is then compiled and linked into a final application. CMake has simple, extensible design and because it is open source it allows expansion with new features.

The CMake is simple to use. The build process is controlled by one or more *CMakeLists.txt* configuration files placed in the project directory and subdirectories. Every configuration file *CMakeLists.txt* consists of one or more commands. The command placed in configuration file have the form "COMMAND(args...)" where COMMAND is the name of the command, and *args* is a list of arguments separate by white-space. CMake sustains many pre-defined commands, and is possible to add custom commands. Additionally, the more advanced users can add custom *makefile* generators for a particular compiler/OS combination.

3 BSbox-tools and CMake

CMake can help to automate the whole process of building BSbox-tools project. Beside the building BSbox-tools project, CMake can be use for BoolSPLG library installation. The configuration files *CMakeLists.txt* placed in the each source directory are used to generate standard build files on a different platform environment.

The use the latest version of CMake 3.26.0 is recommended. The basic CMake setting up require selecting the configuration file *CMakeLists.txt* and selecting the where to build the binaries. Into the main configuration file *CMakeLists.txt* the required CUDA version is set to 10.0. The necessary version can be changed by changing the argument of the configuration command (CUDA **version** REQUIRED). Keep in mind that CMake not always recognizes the correct GPU architecture, this may also be subject of adjustment. By pressing of button "Configure" from the graphic interface will show it is any configuration problems. If there isn't any configuration problems you may proceed with the generating native *makefiles* standard build files (projects) by pressing the button "Generate".

3.1 BSbox-tools Building project with CMake on Windows

The native IDE for CUDA applications on Windows is MSVC (Microsoft Visual Studio). CMake is designed to be used as connector with native build environment and the configuration file creates projects in Windows

MSVC.

CMake can help to automate the generating the BSbox-tools_vx.x MSVC project. After the basic CMake setting up and before to generate the MSVC project it is necessary to check GPU architecture into CMake graphic interface.

- It is possible CMake do not recognise the correct GPU architecture. This will require to set correct GPU architecture before generating MSVC project by change configuration field CMAKE_CUDA_ARCHITECTURE with correct argument value.

The MSVC project must be run with **administrator** rights so that it can copy library files into CUDA include directory in created "BSbox-tools" directory. Generated BSbox-tools_vx.x MSVC project contains target BSbox-tools and CMakePredefinedTargets include ALL_BUILD, INSTALL, ZERO_CHECK. Our targets of interest are INSTALL and BSbox-tools. Target INSTALL copies the header library files into CUDA include directory and the other target build, compile and execute BSbox-tools program.

3.2 BSbox-tools Building project with CMake on Linux

On Linux it is needed to generate Makefile for the each target directory. The *Makefiles* can be generated from CMake through the graphical or terminal interface.

General approach when use CMake with graphic interface is the same as in Windows. Here CMake is used to generate *Makefiles* in the predefined directories which are standard build files for the Linux environment. After *Makefiles* generation it is needed to execute the command *"sudo make install"* in terminal. Before executing the command, the current working terminal directory needs to be set where is the main *Makefile*. This command will install (copy) library header files into CUDA include directory in created "BoolSPLG" directory. Apart of the installation the command will create BSbox-tools program.

The use of the terminal interface require use of few basic commands as *mkdir*, *cd* and *cmake*. The command *mkdir* will be use to create "build" directory. The command *cd* is used to change the current working directory. Command *cmake* is a *Makefile* generator. Executing this command in some particular order with additional option that will point the directory of main configuration file *CMakeLists.txt* will generate the *Makefile* files.

***Example** how to use CMake through the terminal.

- Open the terminal and go to the working directory (project source directory);
- Create a build directory in the top source directory: *mkdir build*
- Change the terminal working directory, go inside it: *cd build*
- Run *cmake* command and point to the parent directory: *cmake ..*
- Finally run *cmake* or in our case: *sudo make install*

Note. *make* and *cmake* are different commands.

The compile and build of the BSbox-tools program practically is done with the command *make*. Executable file is generated by using the command *make* on the BSbox-tools target name. To run BSbox-tools executable file first is need to set BSbox-tools working directory into the terminal. After setting up BSbox-tools working directory BSbox-tools program can be run with the command *"./BSbox-tools"*. If is necessary to debug the project it is much more comfortable to use IDE.

***Note.** There is possible for an error to occur while install library through the terminal. In this case comment the targets in the main configuration file *CMakeLists.txt* and after library installation build the BSbox-tools separately.

3.2.1 BoolSPLG import into Nsight Eclipse

There is no native IDE for CUDA application on Linux. Eclipse with addition plugin Nsight can be use as integrated environment to edit, build, debug and profile CUDA C applications. One possible way to import existing CUDA project (who uses CMake) into Eclipse Nsight is by following these steps ([6] with suitable modification).

Before you generate *Makefiles*, there is need to set the path to build the binaries. Create directory "build" into main directory of BSbox-tools CMake project and set the path. After generating the *Makefiles* through CMake the next steps are:

1. In Eclipse, go to File->New->CUDA C/C++ Project;
2. Uncheck "Use default location" and select your root directory, BSbox-tools CMake project directory;
3. Set the "Project name:" as the name of your BSbox-tools CMake project;
4. In the Project Type tree select "Executable/Empty Project";
5. Select "CUDA Toolkit on the path" in the Toolchains list;
6. Finish with creating of the project;
7. **Build the project.**

   Note. Depending on the IDE version, setting up *"Build location"* [6] is required and this step can be adjusted in the last step of project creation or after creating the project. In some IDE version there is button "Advanced settings" that can be use before to "Finish" with project creation.
8. After building successfully, right click: Project name ->Run As ->Local C/C++ Application, then select which binary you want to execute.

- Also the program can be run through the terminal.

  Open the terminal and set the working directory project build directory (Debug/Release).
  BSbox-tools program can be run with the usual command *./'build name'*, example *./'BSbox-tools v0.3'*.

If some of the files didn't show into IDE Project Explorer but they exist into the file system, refresh the project by right click: Project name ->Refresh.

Here is presented general frame for using CMake with Nsight Eclipse development environment. Of course it can customise additional IDE properties option that will make the environment more user friendly.

Data Availability

The software BSbox-tools, as well as source code and documentation, is available at `https://github.com/BoolSPLG/BSbox-tools_v0.3`. A CMake configuration files is provided for easy compilation.

The library BoolSPLG, as well as the user manual and documentation, is available at `https://github.com/BoolSPLG/BoolSPLG-v0.3` and `https://doi.org/10.5281/zenodo.7825493`. There are also detailed descriptions of each function of the library and test examples. A CMake files is provided for easy compilation.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

The development of the library is supported by the Bulgarian National Science Fund under Contract No KP-06-N32/2-2019 and Contract No KP-06-N62/2/13.12.2022.

References

[1] BSD Licenses CMake, Availaible on:
`https://gitlab.kitware.com/cmake/cmake/-/tree/master/Licenses`

[2] CMake official web site, Availaible on:
`https://cmake.org/`

[3] CUDA Installation Guide for Linux, Availaible on:
`https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html`

[4] D. Bikov, I. Bouyukliev, M Dzhumalieva-Stoeva, BooLSPLG: A Library with Parallel Algorithms for Boolean Functions and S-Boxes for GPU, *Mathematics*, 11(8):1864, 2023.

[5] D. Bikov, I. Bouyukliev, User Guide for CMake and BoolSPLG CUDA, Availaible on:
`https://github.com/BoolSPLG/BoolSPLG-v0.3/blob/main/docs/CMakeBoolSPLG.pdf`

[6] Import existing CUDA project into Nsight web site, Availaible on:
`https://nanxiao.me/en/import-existing-cuda-project-into-nsight/`