Using the NEMA™| SHADER-Edit

A Comprehensive Overview

Version 1.1

February 8, 2018

History

Version	Date	Description
1.1	07-2-2018	NemaP mode included
1.0	10-5-2017	Initial Version

Disclaimer

Questions or comments may be directed to:
Think Silicon S.A
Suite B8
Patras Science Park
Rion Achaias 26504
Greece
web: http://www.think-silicon.com
email:info@think-silicon.com
Tel: +30 2610 911543
Fax: +30 2610 911544

Contents

List of Figures

1 Overview

NEMA™| SHADER-Edit is a graphical user interface that eases the executable generation for NEMA|p and NEMA|t GPUs. It is an easy-to-use, user-friendly vertex and fragment shading editor with an integrated compiler. The current version of this application supports two shading languages:

NEMA|SL : NEMA Shading Language, where the fragment shader is only covered since the vertex processing stage is not programmable in NEMA|p. NEMA|SL is a restricted subset of C++ coupled with advanced NEMA|p builtin functions responsible for specific GPU operations.

ESSL : OpenGL® ES Shading Language, version 1.00, where only the vertex and fragment shaders are covered.

Developers can easily build, debug and optimize NEMA|SL and ESSL shaders without knowing about the internal operation of NEMA GPU's toolchain. This is a very useful path for applications that wish to remain portable by shipping pure source shaders, yet would like to avoid the cost of compiling their shaders at runtime. The editor also reports various statistics as well as includes disassembly functionality that can be further used for debugging purposes.

The core interface of the NEMA™| SHADER-Edit consists of three parts (or one radio box and four panels) as shown in Figure 1:

Setting of the NEMA|GPU shader compilation mode via the radio box controller.

Editing of (left) Vertex and (right) Fragment shaders, located on the upper side of the GUI.

Logging of (left) Build and (right) Toolchain information, located on the bottom side of the GUI.

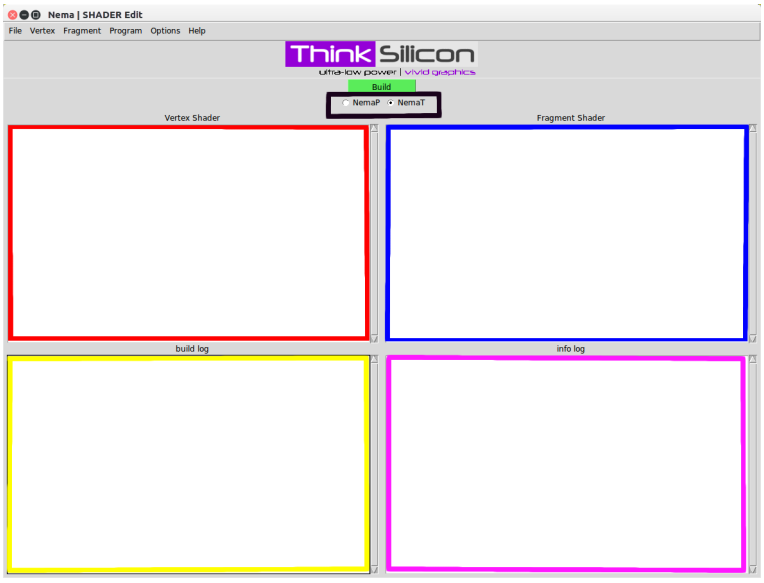

Figure 1: NEMA™| SHADER-Edit interface divided in one radio box and four panels

2 Execution

NEMA™| SHADER-Edit can be executed in the following path:

```
$ cd TSi_host_usr
$ source use_tsi_gl_drivers.sh --version release
$ cd shader_editor
$ ./shader_editor
```

Note that since the NEMA™| SHADER-Edit runs on top of an OpenGL® ES application, it is mandatory to activate the Think Silicon's graphics drivers before executing the tool. Drivers activation sets the operating system environmental variables therefore it is necessary to be issued for every new terminal.

To activate the OpenGL® software stack without (*release*) or with (*debug*) activating driver log tracing, the user has to run:

```
$ cd TSi_host_usr
$ source use_tsi_gl_drivers.sh --version release / debug
```

Note that if *debug* option is selected, it will highly degrade the performance due to the heavy logging.

When the application opens for the first time, the user is prompted to set the toolchain path, since the application is available only for x86 platforms. If for example, the path is in */TSi_host_usr* folder (see Figure 2). and the path is correct, a certification message will appear in the **Info Log** panel:

```
"Setting Toolchain path done!!"
```

Otherwise, an error message will appear:

```
"Toolchain path was not set properly,
 try to set it via Options-> Set Toolchain Path".
```

In that case, the path can be changed by the user by selecting the **Set Toolchain Path** option in the **Options** menu.

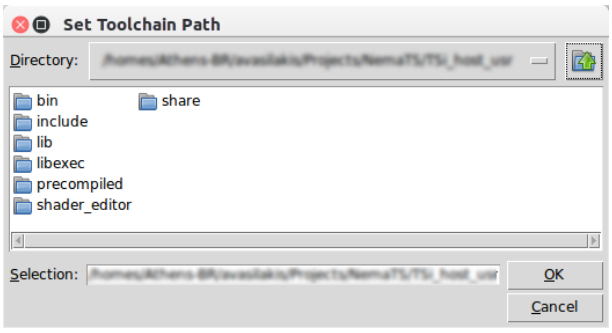

Figure 2: Set toolchain path in NEMA™| SHADER-Edit

3 Usage

3.1 Changing Shader Compilation Mode

First of all, we have to select the target hardware in which the shaders (in either NEMA|SL or ESSL format) will be compiled. This can be done through the radio box in the upper side of the GUI as highlighted with the black box in Figure 1. An alternative way is by selecting the **Enable NemaP mode** or the **Enable NemaT mode** option in the **Options** menu.

3.2 NEMA|t mode

The user can choose the input shader sources by selecting the **Open vertex** and **Open fragment** options in the **Vertex** and **Fragment** menus respectively. A simple example, that contains one vertex and one fragment shader, is provided inside the '*TSi_host_usr/shader_editor/Example/NemaT*' folder to ease the compilation testing usage of NEMA™| SHADER-Edit . When a shader is selected, the source ESSL code is shown and can be editted in either the **Vertex Shader** or the **Fragmet Shader** panel. Note that the name of each selected shader is also displayed above the corresponding panel. Moreover, the user has to define the name of the shader program binary by selecting the **Save Program As** option in the **Program** menu.

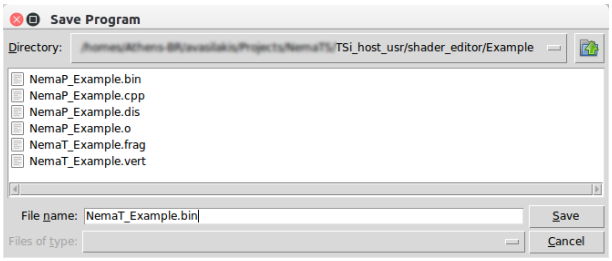

Figure 3: Set shader program binary filename in NEMA™| SHADER-Edit

When the configuration is setup properly, the user can compile the shaders for NEMA GPU by either pressing the **Build** button or selecting the **Build** option in the **File** menu. If no shader compilation errors arise, then the shader program binary is successfully generated which afterwards can be efficiently loaded into any OpenGL® ES graphics application. In that case, the **Build Log** panel will show this message:

```
"Generation of Binary Shader Program ‘example.bin’ Completed Successfully".
```

An appropriate message will also appear in the **Info Log** panel regarding where the graphics acceleration is going to be executed. Hardware graphics acceleration in NEMA GPU depends on the actual shader implementations due to its limited instruction set architecture. In the case of NEMA|t GPU hardware acceleration, **Info Log** panel further offers statistics about the number of required execution cycles and total instructions:

```
"Vertex Shader Info
 Vertex Instructions      : 94
 Vertex Execution Cycles  : 68
```

```
Fragment Shader Info
Fragment Instructions     : 3
Fragment Execution Cycles: 1"
```

Figure 4 depicts how the application looks like when the '*NemaT_example.**' shaders have been selected and compiled. Note that the these shaders can be found in the **Example** folder.

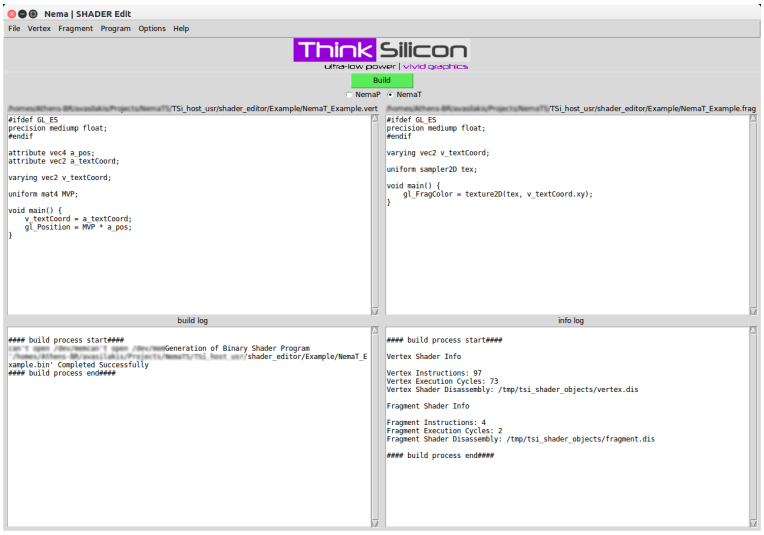

Figure 4: Compiling ESSL shaders

3.3 NEMA|p mode

Since the vertex processing stage is fixed, the user can choose only the input fragment shader source by selecting the **Open Fragment CPP** options in the **Fragment** menu. The source NEMA|SL code is shown and can be editted in the **Fragmet Shader** panel. The rest functionalities remains the same as above. A set of six different NEMA|SL test cases (fragments) can be found in '*TSi_host_usr/shader_editor/Example/NemaP*' folder in order to walk through the NEMA|SL developer learning curve. Developers can explore new shading results by altering the provided source files, guided by the header file '*TSi_host_usr/include/nemats/NemaP_builtins.h*' which contains the NEMA|p buildin function declarations. Figure 5 depicts how the application looks like when the '*example1/test.cpp*' shader has been selected and compiled.

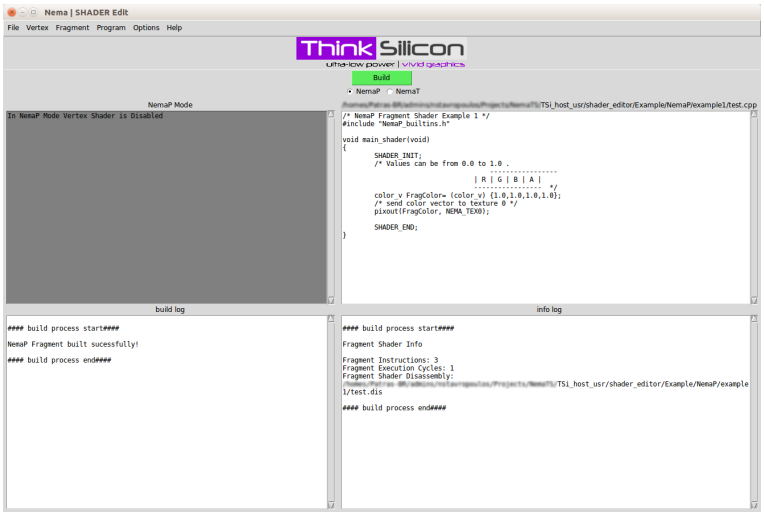

Figure 5: Compiling NEMA|SL shaders

3.4 Terminate Application

Finally, the NEMA™| SHADER-Edit application can be closed by pressing the **Quit** option in the **File** menu.

4 How to use offline ESSL shader compilation from terminal

The offline compiler tool can also be found in the *TSi_host_usr/bin* folder and works by compiling and linking the given vertex and fragment shaders returning back the final program binary through a command-line interface:

```
$ cd TSi_host_usr/bin
```

If *sh.vert* and *sh.frag* correspond to the vertex and fragment ESSL source shaders respectively, then the generated precompiled binary shader program, entitled *sh.bin*, can be generated by executing this command:

```
$ ./tsi_offline_shader_compiler -v sh.vert -f sh.frag -o sh.bin
```

Note that, the *BINARY_PROG* macro preprocessor in the *CMakeLists.txt* file has to be used in the *CMAKE_C_FLAGS* to inform graphics applications.

5 Troubleshooting

As mentioned above, NEMA™| SHADER-Edit will not work properly if the Think Silicon's graphics drivers have not been activated for the current terminal. In that case, reset the configuration setup by deleting the hidden file '*.path.config*' and then execute the application as described in Sec. 2.

```
$ cd TSi_host_usr/shader_editor
$ rm .path.config
$ cd ..
$ source use_tsi_gl_drivers.sh --version release
$ cd shader_editor
$ ./shader_editor
```