Mali Offline Compiler 6.2.0

User Guide

© Arm Limited 2017

Contents

Chapter 1

Introduction

Topics:

Overview

The Mali Offline Compiler is a command line tool that compiles vertex, fragment, compute, geometry, tessellation control, and tessellation evaluation shaders written in the OpenGL® ES Shading Language (ESSL), or kernels written in OpenCL™ C, or Vulkan® SPIR-V binary modules and prints information about the compiled code. Shaders can be used in the OpenGL ES 2.0, 3.0, 3.1, and 3.2 APIs, and kernels can be used in the OpenCL 1.1, and 1.2 APIs. SPIR-V modules can be used in the Vulkan 1.0 API.

Vertex, fragment, compute, geometry, tessellation control, and tessellation evaluation shaders can be compiled into binary shaders that you can link and run on a Mali GPU.

The resulting binary file must be moved to the target platform, read in by the application, and given as parameter in a call to the appropriate OpenGL ES function.

You can use the Mali Offline Compiler to:

- Pre-compile shaders into binary code that you can distribute with your application. The binary code is then passed to the OpenGL ES API as an alternative to shader source code. Binary shaders reduce the time it takes to load a shader load time.
- Assist software development, by checking that shaders compile properly without having to pass them through an OpenGL ES application.
- Optimize your shaders by collecting feedback about the number of cycles each execution of the shader takes when you run it on the GPU.
- Ship your application with pre-compiled binary shaders instead of the OpenGL ESSL source code that is required if you use the on-target compiler.

Note:

- Using the Mali Offline Compiler is optional.
- Because a particular binary shader can only be used on one type of GPU, using compiled shaders reduces portability.
- If you require portability, you can use the on-target Compiler which enables the OpenGL ES drivers to compile OpenGL ESSL shader code or Vulkan SPIR-V binary code dynamically when application runs on a target platform.
- Binary output is currently not available for OpenCL kernels. For OpenCL, the Mali Offline Compiler can only be used to test the validity of OpenCL kernels and to view information about the compiled code.

Installation Package

The installation package for the Mali Offline Compiler contains everything you need to start compiling OpenGL ES shader code and OpenCL kernels on a desktop computer.

You should have downloaded the installation package appropriate to your platform. The Mali Offline Compiler is available for Windows, Linux, and macOS. On Linux and Windows it comes in both 32-bit and 64-bit variants. On macOS only a 64-bit version is available.

Typically, you should download the package that most closely resembles your host environment. If you aren't sure which package to choose, the 32-bit version will work across the widest variety of platforms.

This Release

This release adds the Mali-Gxx r8p0 and Mali-T600 r20p0 driver.

Mali-G72 / r0p0, r0p1, r0p2, r0p3 support has been added.

Chapter 2

Installation

Topics:

Supported Platforms

The Mali Offline Compiler has been tested on the following platforms:

- Microsoft Windows 7 x86
- Microsoft Windows 7 x64
- Ubuntu Linux 14.04 x86
- Ubuntu Linux 14.04 x64
- OS X 10.9 Mavericks x64
- macOS 10.12 Sierra x64

Linux and macOS

On Linux and macOS, the Mali Offline Compiler is provided as a `tgz` package that can be extracted to any location on disk. This package can be extracted using any modern version (i.e. > 1.13) of GNU `tar`:

```
tar xvzf Mali_Offline_Compiler_v6.2.0.7d271f_<platform>_<arch>.tgz
```

Windows

On Windows, the Mali Offline Compiler is provided as both a `zip` package that can be extracted to any location on disk, and as an executable installer package.

To install the software, run the installer and follow the instructions on screen. A link to the installed Mali Offline Compiler directory will be added to the Start menu.

By default, the malisc executable file will be located in:

```
C:\Program Files\Arm\Mali Development Tools\Mali Offline Compiler v6.2.0
```

The installer will add the install location to your PATH environment variable so you can access malisc from any location.

Recent compiler plugins (i.e. versions later than Mali-T600_r20p0-00rel0.dll and Mali-Gxx_r8p0-00rel0.dll) require the *Microsoft Windows C Runtime*.

The Mali Offline Compiler will inform you if a compiler plugin cannot be loaded showing a warning message similar to the following:

```
WARNING: failed to load C:\Program Files\Arm\Mali Development Tools\Mali
Offline Compiler v6.2.0\openglessl\libMali-T600_r20p0-00rel0.dll: error code
126, please refer to the Mali Offline Compiler User Guide.
```

Chapter 3

Usage

Topics:

Using the Mali Offline Compiler

On Windows, if you used the installer, the Mali Offline Compiler installation folder will be automatically added to the PATH environment variable, allowing you to run the `malisc` executable from the command line in any working directory.

Otherwise, if you used the Windows `zip` or the Linux or macOS `tgz` archives, you will either need to run `malisc` from the folder you extracted the archives to, or to manually add this folder to the PATH environment variable.

To see the full list of available options and further help information, on Windows run:

```
malisc --help
```

or on Linux and macOS, run:

```
./malisc --help
```

Sample Shaders

Some sample shaders are provided in the 'samples' folder in the Mali Offline Compiler installation directory.

Testing a Shader

After compiling a shader, it can be tested as follows:

1. Load the compiled shader into an OpenGL ES application using a call to `glShaderBinary`.
2. Draw geometry using the shader.
3. Examine the visual output. If unsatisfactory, alter the uniforms used to control the shader, or use a different shader.

Note:

- If a shader pair, consisting of a vertex shader and a fragment shader, is successfully compiled using the Offline Compiler, the linking stage cannot fail because of resource constraints.
- Linking only fails if the uniform or varying variables of the shaders are incompatible.

Prerotate

Device manufacturers integrating certain Arm Mali devices can choose to enable or disable a 'prerotate' option. If prerotate is enabled on your device it will not be able to load non-prerotate binary shaders.

If you select an offline compiler driver which supports prerotate ('supports prerotate' will be shown when the --list option is passed in) two binaries will be produced by the offline compiler:

1. Prerotate binaries (output-filename.prerotate) can be loaded on all devices but performance is expected be lower.
2. Non-prerotate binaries (output-filename.non-prerotate) will only work on non-prerotate drivers but performance is expected to be better.

For best compatibility, we suggest that developers use the prerotate binary. For best performance, you can try to load the non-prerotate shader, and, if an error is returned, then load the prerotate shader instead. Code for this might look like:

```
#include <GLES2/gl2.h>
#include <GLES2/gl2ext.h>
...
```

```
vertex_shader = glCreateShader(GL_VERTEX_SHADER);
glShaderBinary(1, &vertex_shader, GL_MALI_SHADER_BINARY_ARM,
 non_prerotate_binary, length);
error = glGetError();
if(error == GL_INVALID_VALUE)
{
    int errorlog_length;
    glGetShaderiv(vs, GL_INFO_LOG_LENGTH, &errorlog_length);
    char* errorlog = malloc(errorlog_length * sizeof(char));
    glGetShaderInfoLog(vertex_shader, errorlog_length, NULL, errorlog);
    /*
     * If the non-prerotate binary is not supported then you will see the
     * following message in the errorlog:
     * ---- "The driver is too new for the shader. Try to upgrade your
     * compiler."
     * in this case you should use the prerotate binary instead, e.g.
     * glShaderBinary(1, &vertex_shader, GL_MALI_SHADER_BINARY_ARM,
     * prerotate_binary, length);
     */
}
...
```

OpenCL

OpenCL support is currently only available on Linux and macOS.

The OpenCL C language allows you to use header files in your source code, with the `#include` preprocessor directive. The directory that the Offline Shader Compiler looks in to resolve any relative path header inclusions, such as `#include “header.h”`, is the current working directory. Absolute path header inclusions, such as `#include “/work/header.h”`, may also be used.

Alternatively, multiple source files can be passed in to `malisc`. If multiple source files are specified, they will be concatenated together in the order that they were passed in.

Vulkan

The Mali Offline Compiler accepts Vulkan SPIR-V binary modules. You can create a SPIR-V binary module from a shader written in ESSL using *glslang*.

Note that shaders written using *Vulkan-specific features* cannot be compiled as OpenGL ES SL shaders by the Mali Offline Compiler.

Chapter 4

Known Issues

Topics:

Compilation of a vertex shader may fail with the error "Register allocation failed for vertex shader"

Status

Affected compiler: Mali-400 series.

Description

MaliGP2 has only limited internal bandwidth between its registers and execution units. In some rare cases, the register allocator for MaliGP2 in the shader compiler runs into a situation where there are more operations executed in one cycle than can be fed from the registers simultaneously. The compiler aborts the compilation in this case.

Implications

Some very big or very complex vertex shaders fail to compile.

Workaround

Changing the shader code slightly will often eliminate the problem. Try to rewrite some of the shader to do the calculations differently.

Offline compiler libraries crash when using unsupported extensions

Status

Affected compiler: Mali-T600 series r3p0-00rel0.

Description

If the shader being compiled uses unsupported extensions, the compiler will crash.

Implications

It is not obvious when unsupported extensions are used. It is not possible to get shader statistics for these shaders.

Workaround

Check for extensions in the shader if the compiler crashes. Identify the unsupported extensions by removing them systematically and recompiling.

Mali Offline Compiler does not support SPIR-V entry point names

Status

Affected compiler: Mali-T600 series r18p0-00rel0 and Mali-Gxx series r6p0-00rel0.

Description

Current implementation does not allow to specify the entry point name to be used.

Implications

It is not possible to compile SPIR-V modules containing one or more entry points whose name is not *main*.

Workaround

Compile different stage shaders into separate SPIR-V binary modules and make sure the entry point name is *main*.

Chapter 5

Maintenance

Topics:

Supported Driver Versions

Mali Offline Compiler Version	Mali-400 Series Driver	Mali-T600 Series Driver	Mali-Gxx Series Driver
2.3	r2p2		
3.0.0	r2p3-01rel10		
4.0.0	r3p1-01rel1	r1p0-04rel0	
4.1.0	r3p1-01rel1	r2p1-00rel0*	
4.2.0	r3p1-01rel1	r2p1-00rel0* r3p0-00rel0*	
4.3.0	r3p1-01rel1	r2p1-00rel0* r3p0-00rel0* r4p0-00rel0	
4.4.0	r4p0-00rel1	r2p1-00rel0* r3p0-00rel0* r4p0-00rel0 r4p1-00rel0	
4.5.0	r4p0-00rel1 r5p0-01rel0	r2p1-00rel0* r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0	
4.6.0	r4p0-00rel1 r5p0-01rel0	r2p1-00rel0* r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0	
5.0.0	r4p0-00rel1 r5p0-01rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0	
5.1.0	r4p0-00rel1 r5p0-01rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0	
5.2.0	r4p0-00rel1	r3p0-00rel0*	

Mali Offline Compiler Version	**Mali-400 Series Driver**	**Mali-T600 Series Driver**	**Mali-Gxx Series Driver**
	r5p0-01rel0	r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0	
5.3.0	r4p0-00rel1 r5p0-01rel0 r6p1-00rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0 r9p0-00rel0 r10p0-00rel0	
5.4.0	r4p0-00rel1 r5p0-01rel0 r6p1-00rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0 r9p0-00rel0 r10p0-00rel0 r11p0-00rel0 r12p0-00rel0	
5.5.0	r4p0-00rel1 r5p0-01rel0 r6p1-00rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0 r9p0-00rel0 r10p0-00rel0 r11p0-00rel0 r12p0-00rel0 r13p0-00rel0	
5.6.0	r4p0-00rel1 r5p0-01rel0 r6p1-00rel0 r7p0-00rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0 r9p0-00rel0 r10p0-00rel0	r3p0-00rel0*†

Mali Offline Compiler Version	Mali-400 Series Driver	Mali-T600 Series Driver	Mali-Gxx Series Driver
		r11p0-00rel0 r12p0-00rel0 r13p0-00rel0	
5.7.0	r4p0-00rel1 r5p0-01rel0 r6p1-00rel0 r7p0-00rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0 r9p0-00rel0 r10p0-00rel0 r11p0-00rel0 r12p0-00rel0 r13p0-00rel0 r17p0-00rel0	r3p0-00rel0*† r5p0-00rel0
6.0.0	r4p0-00rel1 r5p0-01rel0 r6p1-00rel0 r7p0-00rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0 r9p0-00rel0 r10p0-00rel0 r11p0-00rel0 r12p0-00rel0 r13p0-00rel0 r17p0-00rel0 r18p0-00rel0	r3p0-00rel0*† r5p0-00rel0 r6p0-00rel0
6.1.0	r4p0-00rel1 r5p0-01rel0 r6p1-00rel0 r7p0-00rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0 r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0 r9p0-00rel0 r10p0-00rel0 r11p0-00rel0 r12p0-00rel0 r13p0-00rel0 r17p0-00rel0 r18p0-00rel0 r19p0-00rel0	r3p0-00rel0*† r5p0-00rel0 r6p0-00rel0 r7p0-00rel0
6.2.0	r4p0-00rel1 r5p0-01rel0 r6p1-00rel0	r3p0-00rel0* r4p0-00rel0 r4p1-00rel0 r5p0-00rel0	r3p0-00rel0*† r5p0-00rel0 r6p0-00rel0 r7p0-00rel0

Mali Offline Compiler Version	Mali-400 Series Driver	Mali-T600 Series Driver	Mali-Gxx Series Driver
	r7p0-00rel0	r5p1-00rel0 r6p0-00rel0 r7p0-00rel0 r8p0-00rel0 r9p0-00rel0 r10p0-00rel0 r11p0-00rel0 r12p0-00rel0 r13p0-00rel0 r17p0-00rel0 r18p0-00rel0 r19p0-00rel0 r20p0-00rel0	r8p0-00rel0

* These driver versions are not available for macOS.

† These driver versions are not available for Windows.

Support

For further support on this product, please visit the *Arm Connected Community*.

It should be noted that continuing support of the product is at Arm's discretion unless explicitly included in a current contract between Arm and you.

Chapter 6

Legal

Topics:

Proprietary Notice

Words and logos marked with ® or TM are registered trademarks or trademarks of Arm Limited in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

OpenGL is a registered trademark and the OpenGL ES logo is a trademark of Silicon Graphics Inc. used by permission by Khronos.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

Vulkan and the Vulkan logo are registered trademarks of the Khronos Group Inc.

SPIR and the SPIR logo are trademarks of the Khronos Group Inc.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

The Mali OpenCL compiler library is based on and incorporates portions of LLVM Version 3.5. The LLVM source code is available from the LLVM website.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by Arm Limited in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Arm Limited shall not be liable for any loss for damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.