LLVM/Clang integration to Buildroot

Valentin Korenblit
Romain Naour

Smile

valentinkorenblit@gmail.com
romain.naour@smile.fr

August 28, 2018

Overview

Plan

Internship objectives

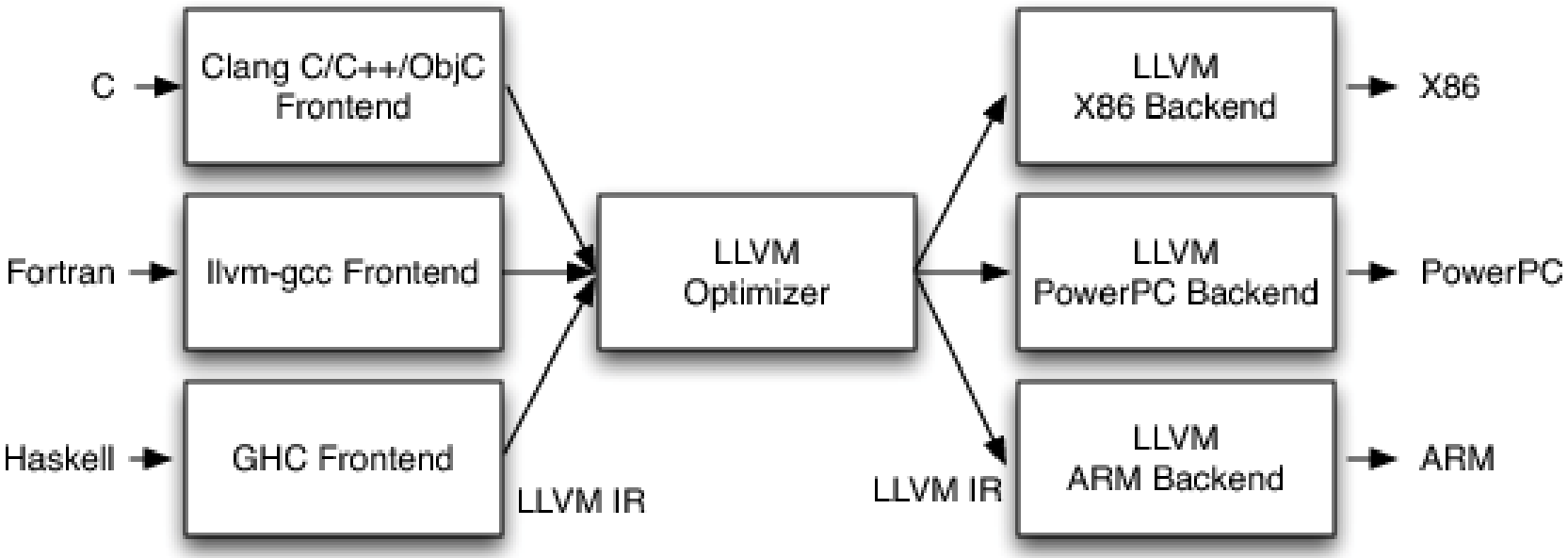
- Preliminary study of LLVM/Clang
- LLVM/Clang integration to Buildroot
  - llvmpipe for Mesa 3D
  - AMDGPU backend
  - OpenCL implementations
- OpenCL support for already existing packages in Buildroot
- Integration of new packages that can benefit from OpenCL: image processing (i.e. Darktable), simulation, cryptography, etc.

Plan

LLVM

- Open source project started in 2000. LLVM 1.0 released in October 2003
  - Several subprojects: LLVM Core, Clang, lldb, compiler-rt, libclc, lld
- Provides a compiler infrastructure written in C++
  - Designed as an API from the beginning
  - Focusing on compile time and performance of the generated code
- Well structured and documented
- Some existing backends:
  - **ARM**, ARM64, Hexagon, Mips, Mipsel, NVIDIA PTX 32/64, PowerPC 32/64, **AMDGPU**, Sparc, Thumb, x86, **x86-64**, XCore

LLVM - Three-phase approach



LLVM - Internal aspects

- Intermediate Representation (IR)
  - Mostly architecture-independent instruction set (RISC)
  - Strongly typed
  - Unlimited number of virtual registers in SSA
- IR Code example:

```
define i32 @main() #0 {
entry:
  %retval = alloca i32, align 4
  %c = alloca i32, align 4
  store i32 0, i32* %retval, align 4
  %0 = load i32, i32* @a, align 4
  %1 = load i32, i32* @b, align 4
  %add = add nsw i32 %0, %1
  store i32 %add, i32* %c, align 4
  ret i32 0
}
```

Plan

Clang

- Frontend C/C++, Objective C/C++ and OpenCL C for LLVM
- Clear and concise diagnostics (error and warning messages)
- Natively a cross-compiler: `-target <triple>`
- Sanitizers
- Goals
  - Designed to be highly compatible with GCC
  - C++14 supported since Clang 3.4
  - C++17 supported since Clang 5
- Performance vs GCC ?[1]

[1] `http://www.phoronix.com/vr.php?view=25742`

Plan

Who is using LLVM/Clang

- Android: Renderscript compiler based on LLVM
- Apple
  - All operating systems built with Clang
  - Xcode IDE uses Clang compiler and static analyzer by default
- FreeBSD can be entirely built with Clang/LLVM
- Google is using Clang for building:
  - Android user space
  - Chrome for all platforms (since March 2018)
- OpenCL: AMD, Apple, Intel, NVidia (runtime compiler)
- OpenJDK: Shark JIT compiler for Zero
- Sony Interactive Entretainment: CPU compiler for PS4

Plan

Compiling Linux with Clang

- Challenges
  - Linux kernel expects to use some GCC behavior that is not supported by Clang:
    - Variable Length Arrays inside structures
    - Nested functions
    - Explicit register variables
    - LLVM assembler cannot be used to build the kernel
- LLVMLinux Project: Kernel 4.4 and 4.9 built with Clang for x86_64 and ARM64 (patches applied)[2]
- Still depends on GNU `as` and `ld`
- glibc[3] ?

[2] https://lwn.net/Articles/734071/

[3] https://sourceware.org/glibc/wiki/GlibcMeetsClang

Plan

Toolchain components

Component	LLVM	GNU
C/C++ Compiler	clang	gcc
Assembler	cc1as	as
Linker	lld	ld
Runtime	compiler-rt	libgcc
Debugger	lldb	gdb
Unwinder	libunwind	libgcc_s
C++ library	libc++abi, libc++	libsupc++ libstdc++
Tools	llvm-ar, llvm-as etc.	ar, objdump, etc.
C library	-	libc

Plan

LLVM/Clang integration to Buildroot - What does it enable?

- Gallium llvmpipe driver: software rasterizer that uses LLVM to do runtime code generation
  - It is the fastest software rasterizer for Mesa3D
- OpenSWR for scientific visualization (AVX, AVX2)
- Most OpenCL implementations rely on LLVM

Project planning

1. Provide LLVM support for x86, ARM and AArch64
2. Provide LLVM support for AMDGPU (R600 to GCN)
3. Enable LLVM support for Mesa 3D:
   - Gallium Drivers: llvmpipe, R600, RadeonSI
4. Add Clang
5. Activate OpenCL
   - AMD GPUs (Clover)
   - Broadcom Videocore IV (VC4CL)

Available hardware for testing

- Platform 1 - x86_64 (HP ProBook)
  - Processor: AMD A4-3300M Dual Core @ 1.9 GHz
  - GPU: AMD Radeon Dual Graphics (HD6480G + HD7450M)
- Platform 2 - ARM (Raspberry Pi 2 Model B)
  - Processor: ARMv7 Cortex-A7 Quad Core @ 900 MHz
  - GPU: Broadcom Videocore IV
- Platform 3 - ARM/AArch64 (Raspberry Pi 3 Model B)
  - Processor: ARMv8 Cortex-A53 Quad Core @ 1.2 GHz
  - GPU: Broadcom Videocore IV

LLVM package - Directory Layout

- llvm/lib/ Most source files are here
  - IR
  - AsmParser
  - Bitcode
  - Transforms
  - Target
- llvm/tools/ Executables built out of the libraries [4]
  - llvm-as
  - **llvm-config**
  - lli
  - llc
  - opt
- llvm/utils/ Utilities for working with LLVM source code
  - codegen-diff
  - llvmgrep
  - **TableGen**

[4] https://llvm.org/docs/CommandGuide/index.html

LLVM package

- CMake-based project
  - Buildroot provides a CMake infrastructure ☺
- Plenty of options, some with misleading names
  - `LLVM_TARGETS_TO_BUILD`
  - `LLVM_TARGET_ARCH`
  - `LLVM_DEFAULT_TARGET_TRIPLE`
  - `LLVM_HOST_TRIPLE`
- Difficult to be cross-compiled
  - At least llvm-tblgen must be compiled for the host first
  - It requires a modern and fully-featured toolchain
- Takes a lot of time to compile

LLVM packaging for Buildroot

- LLVM 5.0.1 selected
- Only LLVM libraries are needed (libLLVM.so), no tools
- The main problem: **llvm-config** not giving the desired output
  - It's a compiled program. Normally "config" programs are scripts.
  - llvm-config compiled for the host must be installed to the target's sysroot.
  - Some output depends on its location and some other is contained in the binary 😕
- **Solution:** do a full installation of LLVM for the host using the same[5] configuration options as for the target and link LLVM tools with libLLVM.

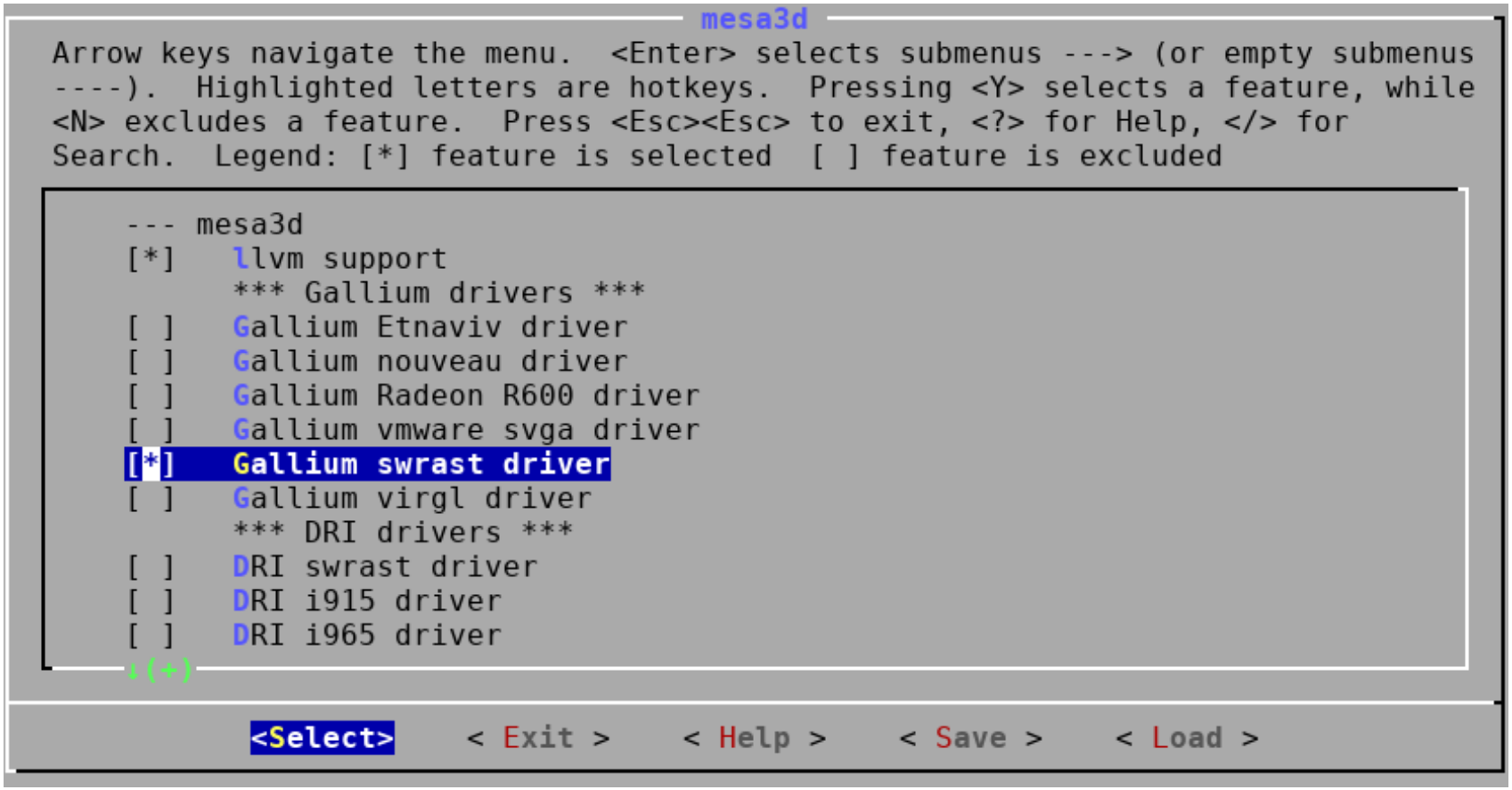
[5] Tools are not built for the target

LLVM support for Mesa 3D drivers

- Gallium llvmpipe: fastest software rasterizer for Mesa
  - Uses LLVM to do runtime code generation
- Gallium R600 and RadeonSI
  - Both use AMDGPU LLVM backend
- Some benchmarks:

Gallium Driver	GLMark2	GLMark2-es2
R600 on HD6480	156	156
softpipe on AMD A4-3300M	3	3
softpipe on Cortex-A53 (32-bit)	-	0
softpipe on Cortex-A53 (64-bit)	-	0
llvmpipe on AMD A4-3300M	47	52
llvmpipe on Cortex-A53 (32-bit)	-	11
llvmpipe on Cortex-A53 (64-bit)	-	13

Selecting llvmpipe



OpenCL

- API enabling general purpose computing on GPUs, CPUs, DSP,s FPGAs, etc. Well suited for certain kinds of parallel computations:
  - Hash cracking (SHA, MD5, etc.)
  - Image processing
  - Simulations
- OpenCL presents itself as a library with a simple interface:
  - Standarized API headers for C and C++
  - The OpenCL library (libOpenCL.so): collection of types and functions which all conforming implementations must provide.

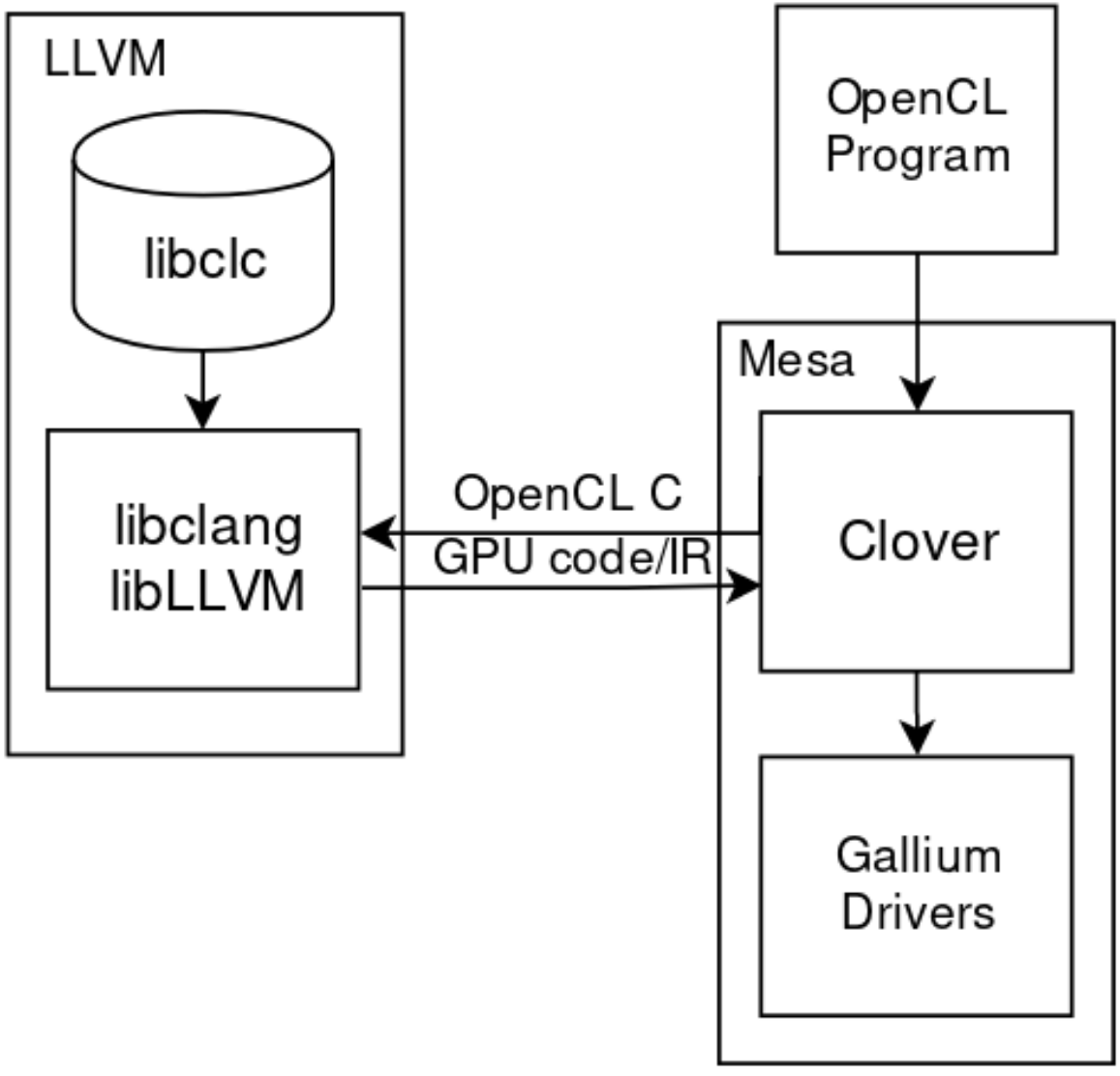
OpenCL - Open Source Implementations

Project	Version	Hardware
Clover	1.1	AMD
Pocl	1.2	CPU, NVIDIA[6], AMD[7], TCE/TTA
Beignet	2.0	Intel
ROCm OpenCL	1.2	AMD[8]

- OCL ICD Loader allows multiple OpenCL implementations to co-exist on the same system

[6] Needs propietary drivers
[7] HSA compatible hardware
[8] ROCm compatible hardware

Clover


LLVM
libclc
libclang
libLLVM
OpenCL C
GPU code/IR
OpenCL
Program
Mesa
Clover
Gallium
Drivers

Clang packaging for Buildroot

- host-clang is needed to compile libclc
  - Many functions writen in LLVM IR (.ll)
- It is normally built inside LLVM source tree (llvm/tools) but Buildroot uses per-package build directories.
  - Some more tweaks are needed
- Binaries, headers and some scripts must be removed from the target.
  - Only libclang.so is necessary.

libclc packaging for Buildroot

- Builtin functions defined in the OpenCL 1.1 specification
  - sin, cos, min, max, etc.
- LLVM IR bitcode
- Also provides headers needed to compile OpenCL kernels by calling `clCreateProgramWithSource`
- Main problem: Buildroot removes `/usr/include` from the target filesystem

Testing Clover: Piglit

Result summary

Currently showing: all

Show: all | changes | enabled | fixes | regressions | skips | problems | disabled

	cl (info)
all	**3206/3330**
api	**45/48**
clbuildprogram	pass
clcompileprogram	skip
clcreatebuffer	pass
clcreatecommandqueue	pass
clcreatecontext	pass

Total	Skip	Pass	Fail	Crash
704	94	541	60	9

Bonus: OpenCL for VideoCore IV (Raspberry Pi)

- VC4CL Project: OpenCL 1.2 EMBEDDED PROFILE
  - VC4C: compiles OpenCL kernels into machine code.
  - VC4CLStdLib: platform-specific implementation of the OpenCL C standard library.
- VC4C calls Clang driver (binary) instead of linking against libclang.so
  - Buildroot does not allow a compiler to be installed on the target
- Kernel compilation is extremely slow on the target.
  Solution: compile kernels to LLVM IR bitcode on the host with host-clang and call `clCreateProgramWithBinary` instead of `clCreateProgramWithSource`
  - Attention: code must be device-specific

LLVM/Clang for Buildroot - Current State

- Commited to Buildroot's master:
  - LLVM package ✓
  - LLVM support for Mesa 3D ✓
  - Clang package ✓
- Patchs sent to the mailing list:
  - libclc package
  - OpenCL support for AMD GPUs
- TODO
  - Add OpenCL support for already existing packages
  - Add new packages that depend on LLVM

For full details on this subject, take a look at this article and download my internship report:

`http://www.linuxembedded.fr/2018/07/llvmclang-integration-into-buildroot/`

Linux Embedded

Le blog des technologies libres et embarquées

Questions?