Current State of Open Source GPGPU

Tom Stellard
XDC 2017
September 20, 2017

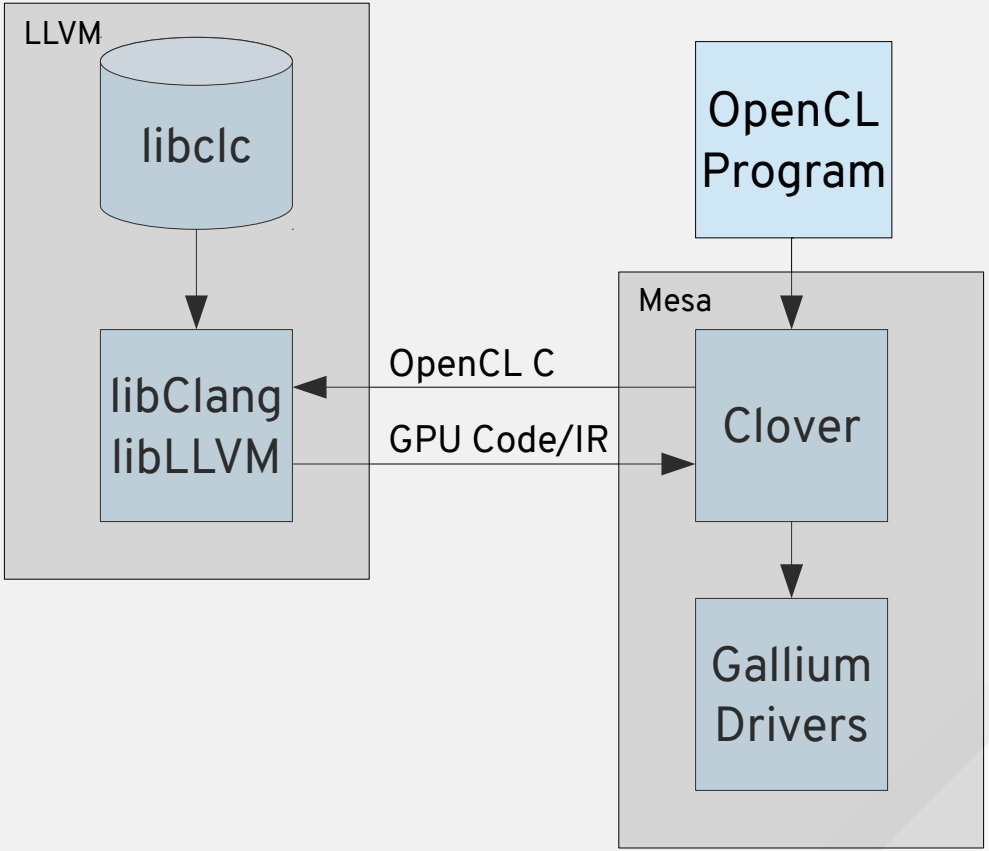
OpenCL

```
const char * program_src =
"__kernel\n"
"void pi(__global float * out) \n"
"{\n"
"  out[0] = 3.14159f;\n"
"}\n";

int main(int argc, char ** argv) {

  // Insert device setup code here.

  program = clCreateProgramWithSource(context,1,&program_src,NULL,&error);

  error = clBuildProgram(program,1,&device_id,NULL,NULL,NULL);

  // Insert buffer allocation and program execution code here.

}
```

Clover

Overview

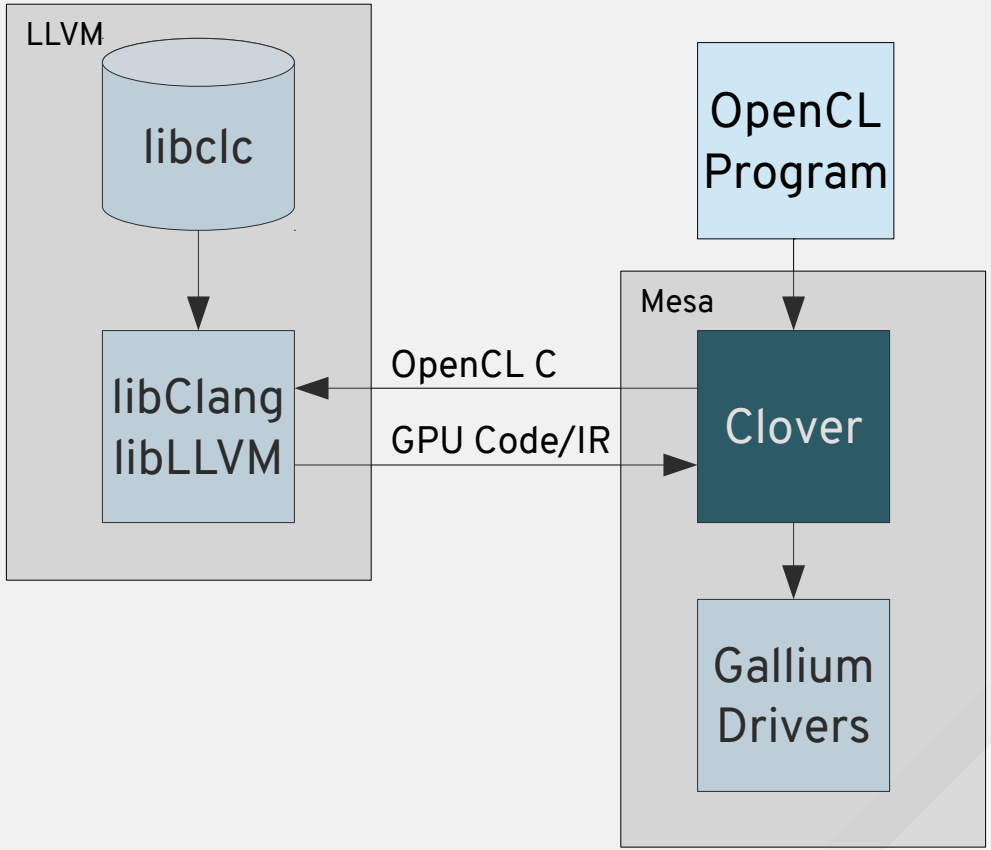
- Clover – hardware independent OpenCL API implementation.
- Gallium Drivers – Hardware dependent userspace GPU drivers.
- libClang – OpenCL C language frontend.
- libLLVM – Hardware dependent codegen.
- libclc – LLVM IR bitcode library with device builtin functions (e.g. sin, cos, min, max).



Clover

API Status

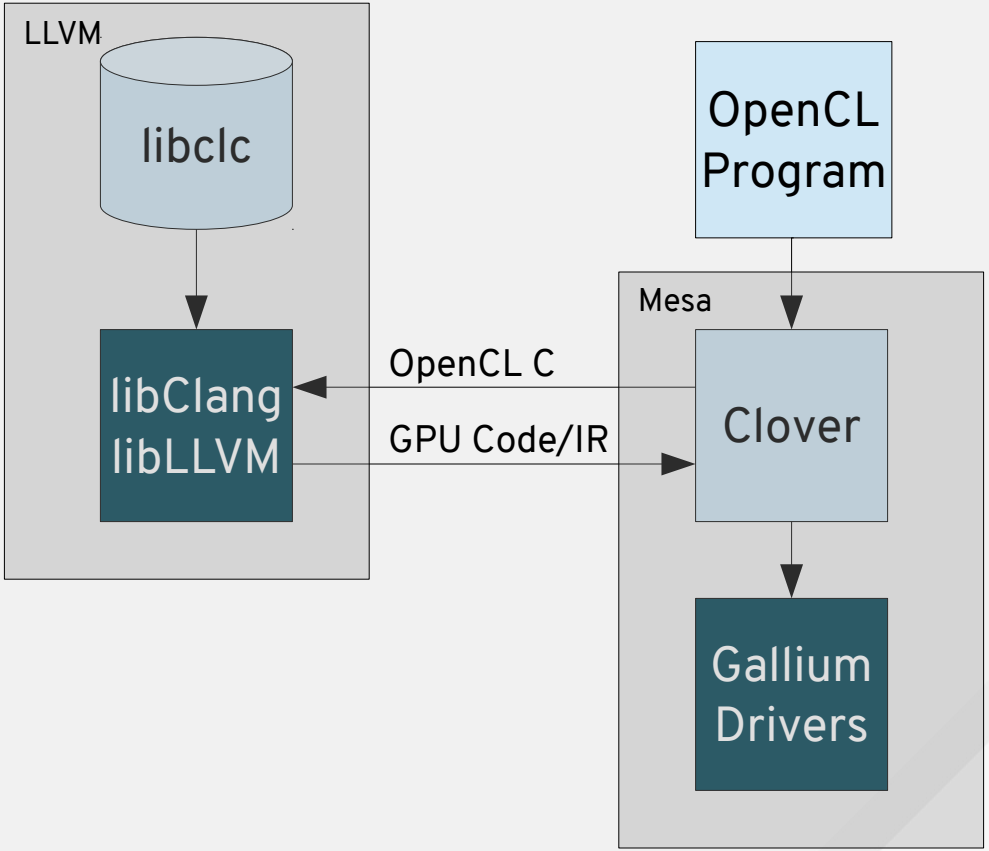
- OpenCL 1.1 Supported.
- Missing OpenCL 1.2 Functions:
  - clEnqueueMigrateMemObjects(), clGetKernelArgInfo(), clEnqueueFillBuffer(), clEnqueueFillImage()
- OpenCL C 1.2 supported via clang.



Clover

Gallium - AMD

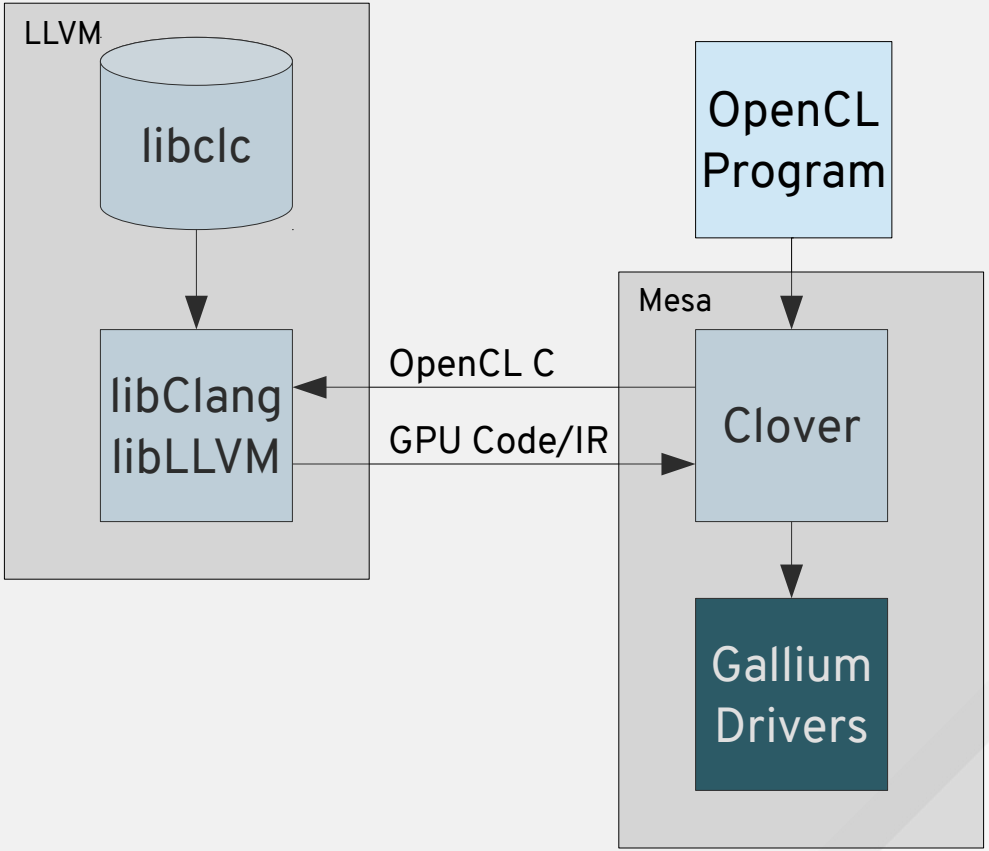
- Missing image support in LLVM/libclc.
- Open Source CTS making development easier.
- cl_khr_fp16 almost done.
- Benefiting from shared compiler with ROCm.



Clover

Gallium - nouveau

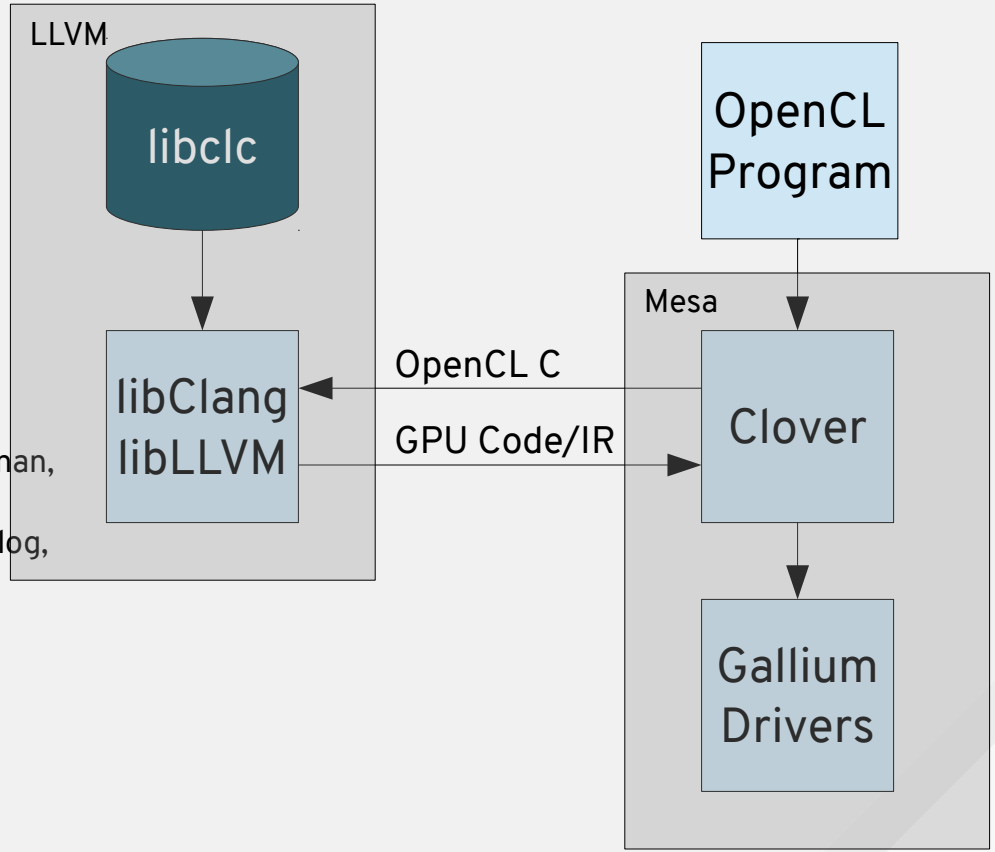
- Work in progress OpenCL C support.
  - OpenCL C → SPIR-V → NV machine code.
- Compute API implementation mostly in place.



libclc

Status

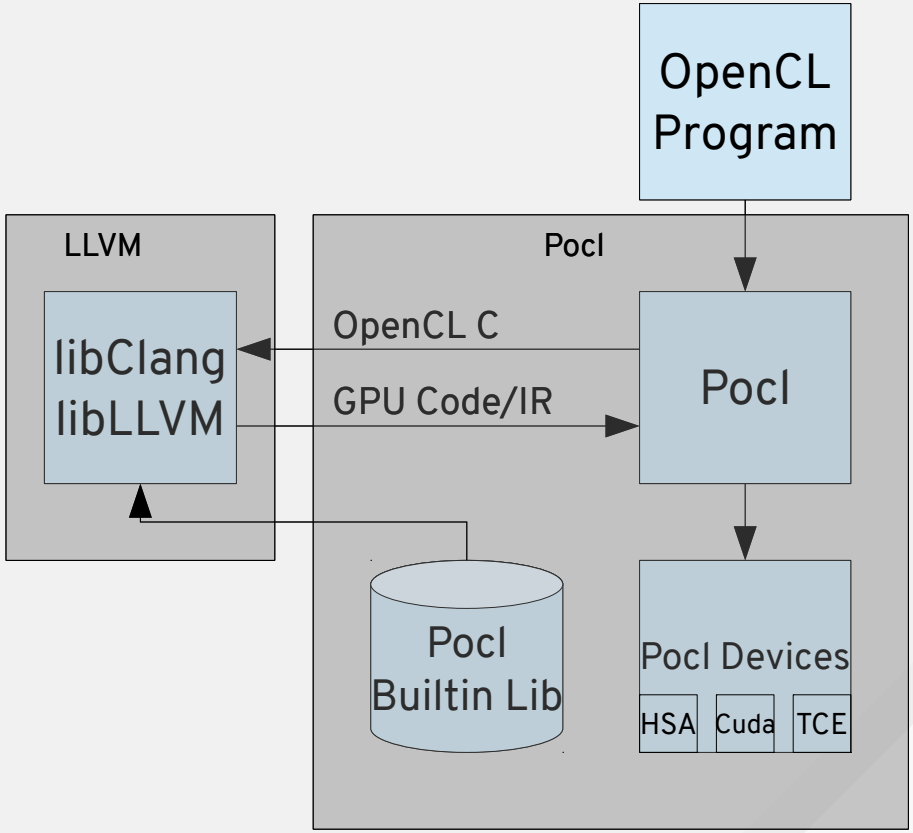
- nvptx – basic support.
- amdgpu - well supported.
- Missing OpenCL 1.2 functions: maxmag, minmag, nan, powr, remainder, remquo, rootn, tanpi, half_cos, half_divide, half_exp, half_exp2, half_exp10, half_log, half_log2, half_log10, half_powr, half_recip, half_rsqrt, half_sin, half_sqrt, native_log10
- Missing Image functions.



Pocl

Status

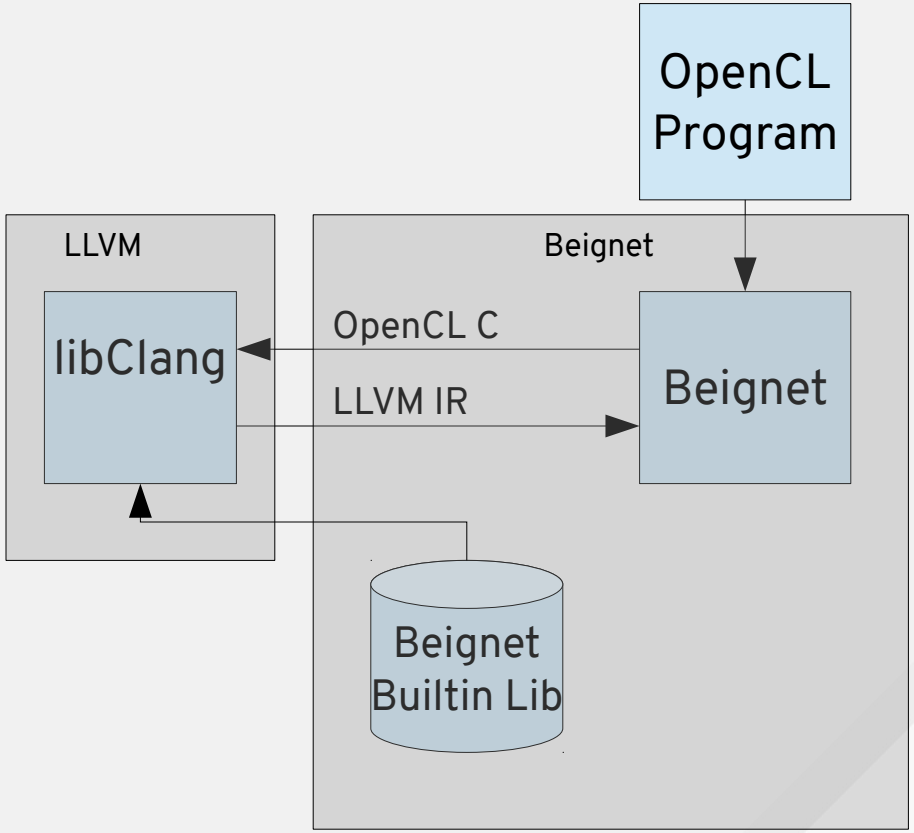
- Supports LLVM 5.0.
- Approaching OpenCL 1.2 completeness.
- New Cuda backend.



Beignet

Status

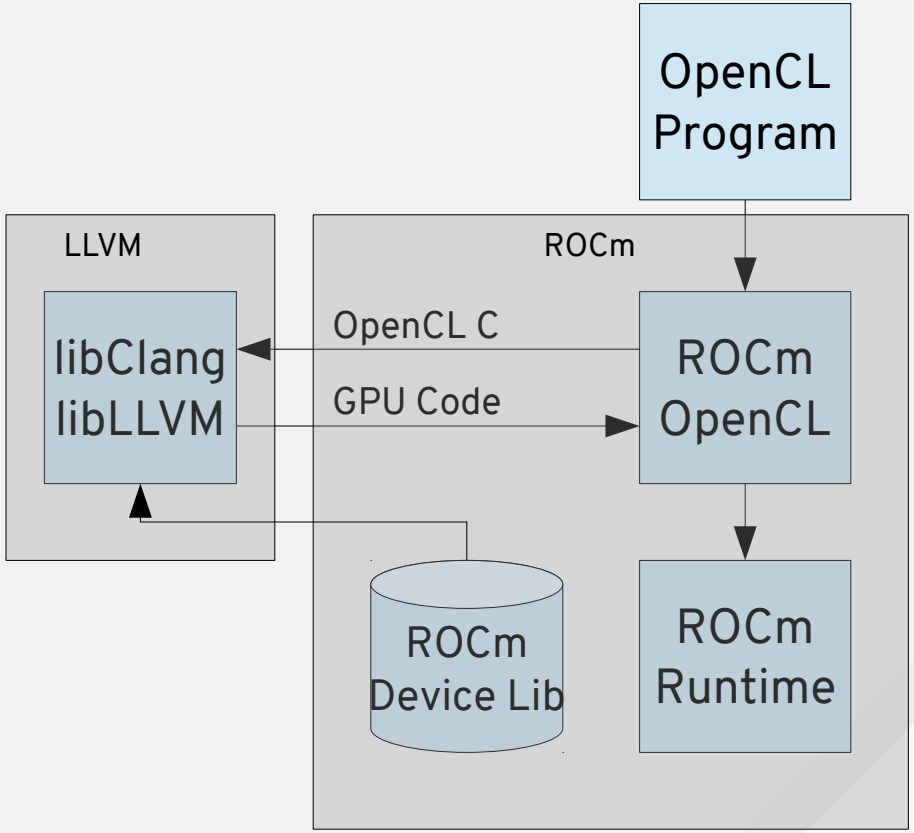
- Supports LLVM 5.0.
- OpenCL 2.0.
- Passed conformance for 1.2 in 2015.



ROCm OpenCL

Status

- Supports ROCm compatible AMD hardware.
- OpenCL 1.2 API with OpenCL C 2.0.
- Function call support in progress.
- Compiler optimizations for LLVM AMDGPU backend.

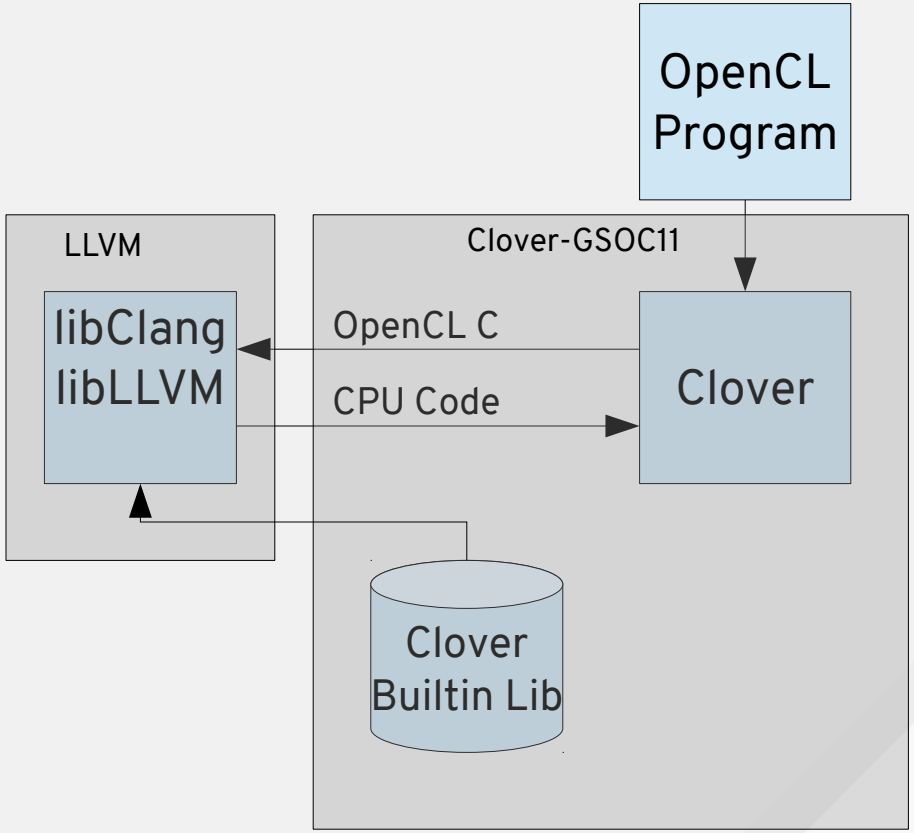


Clover-GSOC11

Status

Inspiration for Clover

- Development started in 2009.
- Goal was OpenCL over gallium API.
- Picked up as Google Summer of Code Project in 2011.
  - Gallium support dropped.
  - Focused on supporting CPU targets.
- Development stopped after 2011.
- Inspired clover in mesa, but the mesa code is a complete rewrite.

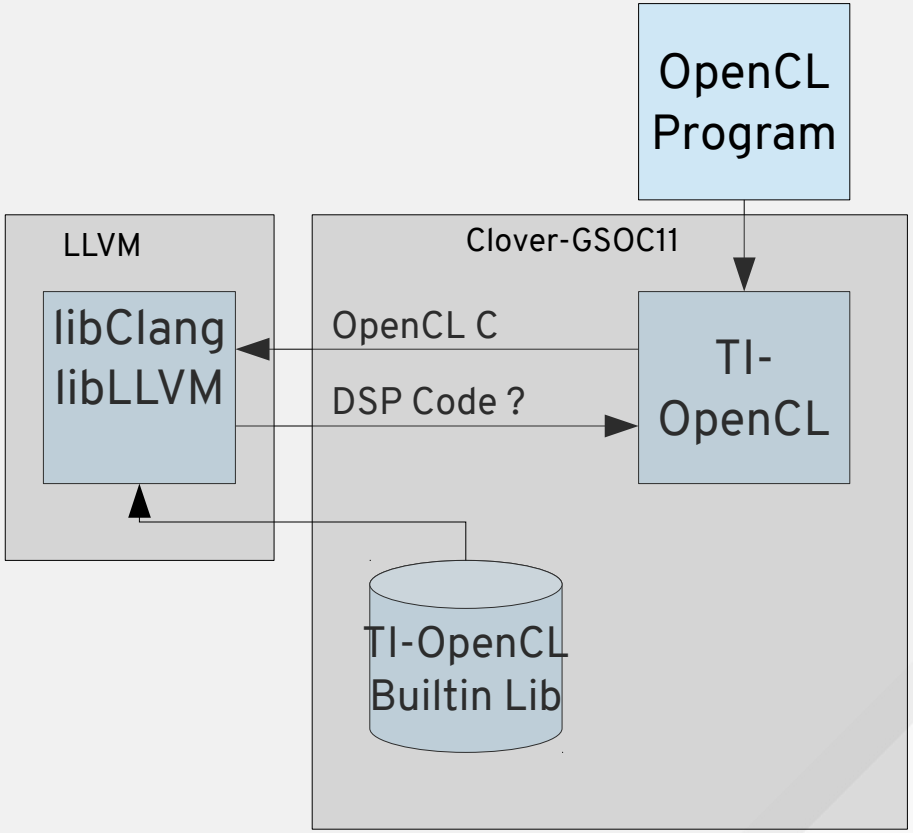


TI-OpenCL

Status

OpenCL for DSP

- Based on Clover-GSOC2011.
- Still under active development.
- Borrowed device library code from libclc/pocl.
- OpenCL 1.1
- Some CPU support.

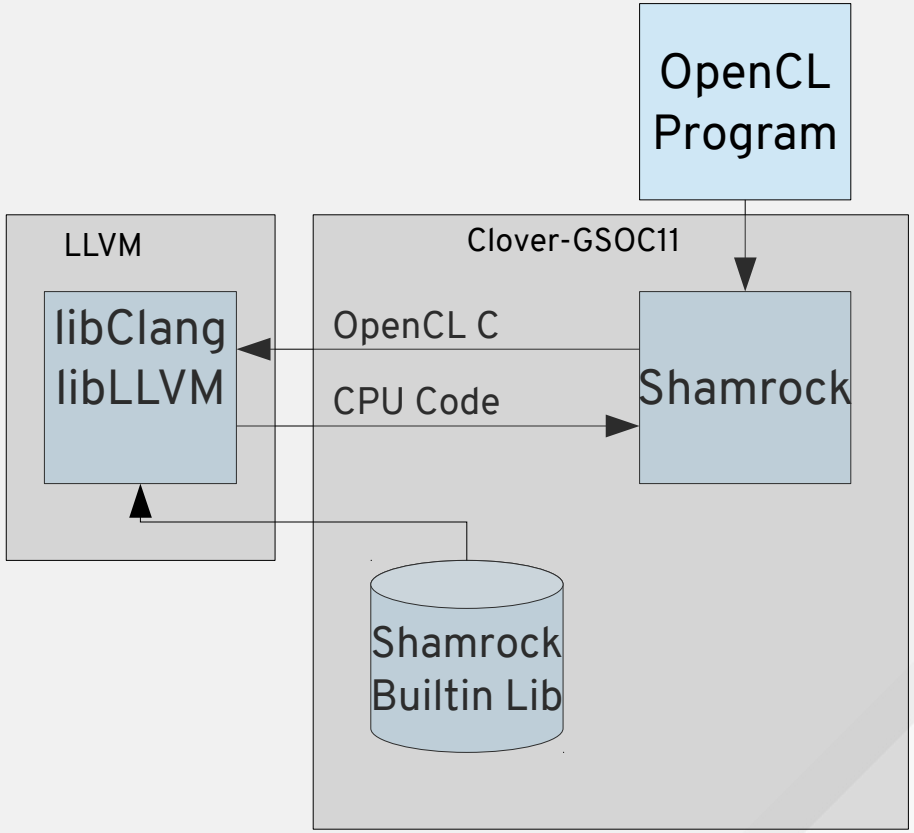


Shamrock

Status

Fork of TI-OpenCL

- Improved CPU support in TI-OpenCL.
- CPU support for OpenCL 1.2.
- No commits for one year.



OpenCL

Overview

Lots of fragmentation

- 4 Different device library implementations.
- 3 Implementations supporting various subsets of AMD hardware.
- Beignet only conformant implementation (1.2).

Project	Version	Hardware
Clover	1.1	AMD
Pocl	1.2	CPU, NVIDIA[1], AMD[2] TCE/TTA
Beignet	2.0	Intel
ROCm OpenCL	1.2	AMD[3]

1 Requires proprietary drivers
2 HSA Compatible Hardware
3 ROCm Compatible Hardware

OpenCL

Overview

Why so much fragmentation ?

- Missing features lead to new implementation and prevent consolidation:
  - Clover missing CPU support.
  - POCL missing interface to open GPU drivers.
  - Clover/Gallium not supporting Intel GPUs.
- Well-tested closed implementation open sourced
  - ROCm OpenCL

Project	Version	Hardware
Clover	1.1	AMD
Pocl	1.2	CPU, NVIDIA[1], AMD [2] TCE/TTA
Beignet	2.0	Intel
ROCm OpenCL	1.2	AMD[3]

1 Requires proprietary drivers
2 HSA Compatible Hardware
3 ROCm Compatible Hardware

OpenCL

Overview

Other incentives for fragmentation:

- OpenCL ICD makes it easy for multiple implementations to co-exist.
- Leveraging LLVM/Clang for OpenCL C support makes new implementations easier.

Project	Version	Hardware
Clover	1.1	AMD
Pocl	1.2	CPU, NVIDIA[1], AMD[2] TCE/TTA
Beignet	2.0	Intel
ROCm OpenCL	1.2	AMD[3]

1 Requires proprietary drivers
2 HSA Compatible Hardware
3 ROCm Compatible Hardware

OpenCL

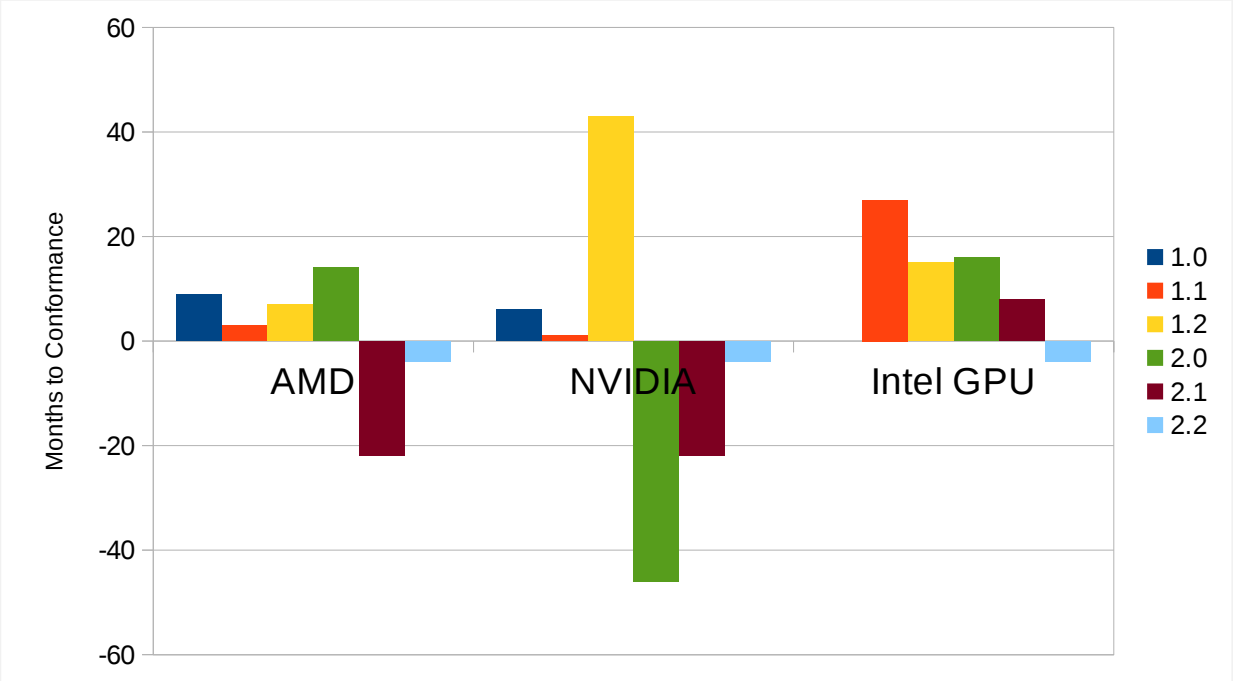
Vendor Status

Vendor adoption has slowed for recent versions.

- Current OpenCL version is 2.2, spec released May 2017.
- OpenCL 1.2 spec was released in November 2011.
- NVIDIA achieved 1.2 conformance in May 2015.
- Most SOC vendors support 1.2.
- Only ARM and Qualcomm only SOC vendors to support 2.0.
- 1 successful conformance submission in 2017:
  - ARM Mali OpenCL 2.0

Vendor	Highest Conformant Version
Intel	2.1
AMD	2.0
NVIDIA	1.2

OpenCL Implementation Rates



OpenCL

Open Source Outlook

OpenCL 1.0 → 2.2

- Slow vendor adoption has been a positive:
  - Open source implementations not that far behind.
- Fragmentation still an issue going forward.
- Incentive for implementing OpenCL 2.0+ may be low.

OpenCL

Future Direction

‘OpenCL Next’ discussed at IWOCL 2017

- Convergence of OpenCL and Vulkan API.
  - Not clear what this means.
- Fine grained feature capabilities.
  - API subsets for different devices / use cases.
- More language flexibility.

OpenCL

Open Source Outlook

‘OpenCL Next’ could be good for Open Source implementations if:

- Full API implementation not required for conformance.
- Shared effort with Vulkan.
- Compiler frontends not part of API.

SPIR-V

Status

Translators to/from SPIR-V

- Khronos SPIR-V LLVM
  - Bi-directional SPIR-V to LLVM IR converter.
  - Fork of LLVM 3.8.
- clspv
  - OpenCL C to SPIR-V.
  - Stand-alone project.
  - Uses libclang to compile OpenCL C to LLVM IR.
  - Converts LLVM-IR to SPIR-V.

```
     OpCapbility Shader
     OpMemoryModel Logical Simple
     OpEntryPoint GLCompute %3 "main"
     OpExecutionMode %3 LocalSize 64 64 1
%1 = OpTypeVoid
%2 = OpTypeFunction %1
%3 = OpFunction %1 None %2
%4 = OpLabel
     OpReturn
     OpFunctionEnd
```

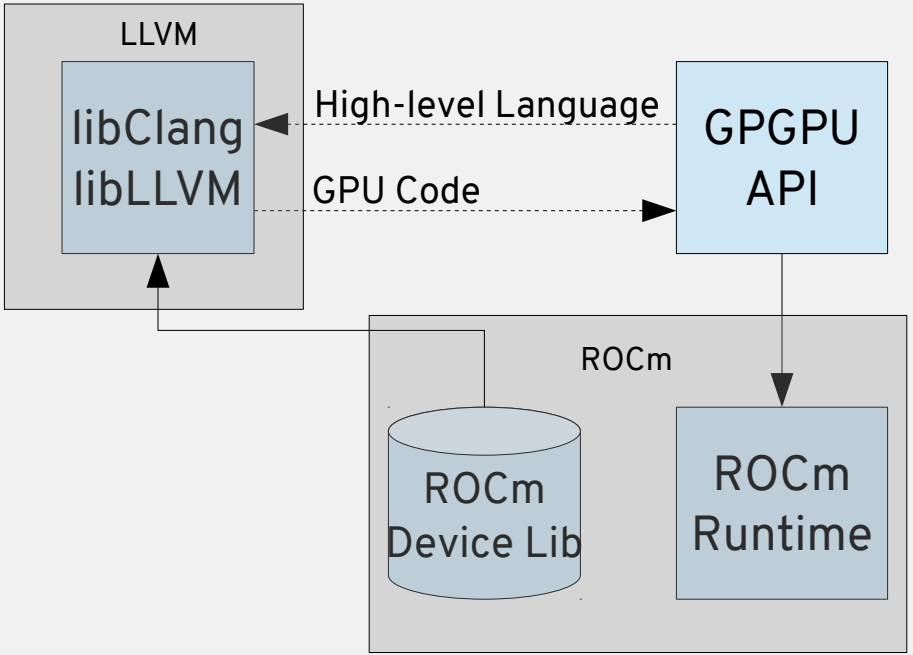
https://github.com/KhronosGroup/SPIRV-Tools/blob/master/syntax.md

ROCm

status

GPU Runtime / Toolchain for AMD hardware

- Full stack: Kernel Drivers, Userspace Drivers, toolchain.
- Completely Open Source.
- Low-level compute API for AMD.
- Clang based toolchain:
  - Direct-to-ISA compilation.
  - Assembler / Disassembler.
- HCC compiler for C++AMP/HCC language.



OpenMP

Status

Open Source Runtimes

- GCC Runtime (libgomp).
- Clang Runtime (libomp).
- Both runtimes have Cuda backends.

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
int nthreads, tid;

/* Fork a team of threads giving them their own copies of variables */
#pragma omp parallel private(nthreads, tid)
  {

  /* Obtain thread number */
  tid = omp_get_thread_num();
  printf("Hello World from thread = %d\n", tid);

  /* Only master thread does this */
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }
  }  /* All threads join master thread and disband */
}
```

Cuda

Overview

Single-source GPGPU

- Low-level: Driver API.
- Higher-level: Runtime API.
- Compiler inserts API calls into program for launching kernel.

```
#include <iostream>

__global__ void axpy(float a, float* x, float* y) {
  y[threadIdx.x] = a * x[threadIdx.x];
}

int main(int argc, char* argv[]) {
  const int kDataLen = 4;

  float a = 2.0f;
  float host_x[kDataLen] = {1.0f, 2.0f, 3.0f, 4.0f};
  float host_y[kDataLen];

  // Copy input data to device.
  float *device_x, *device_y;
  cudaMalloc(&device_x, kDataLen * sizeof(float));
  cudaMalloc(&device_y, kDataLen * sizeof(float));
  cudaMemcpy(device_x, host_x, kDataLen * sizeof(float),
             cudaMemcpyHostToDevice);

  // Launch the kernel.
  axpy<<<1, kDataLen>>>(a, device_x, device_y);

  // Copy output data to host.
  cudaDeviceSynchronize();
  cudaMemcpy(host_y, device_y, kDataLen * sizeof(float),
             cudaMemcpyDeviceToHost);

  // Print the results.
  for (int i = 0; i < kDataLen; ++i) {
    std::cout << "y[" << i << "] = " << host_y[i] << "\n";
  }

  cudaDeviceReset();
  return 0;
}
```

Cuda

Status

Open Source Work

- gdev
  - Kernel module, userspace API.
  - Last public commit 3 years ago.
- Clang CUDA frontend
- AMD HIP

```
#include <iostream>

__global__ void axpy(float a, float* x, float* y) {
  y[threadIdx.x] = a * x[threadIdx.x];
}

int main(int argc, char* argv[]) {
  const int kDataLen = 4;

  float a = 2.0f;
  float host_x[kDataLen] = {1.0f, 2.0f, 3.0f, 4.0f};
  float host_y[kDataLen];

  // Copy input data to device.
  float *device_x, *device_y;
  cudaMalloc(&device_x, kDataLen * sizeof(float));
  cudaMalloc(&device_y, kDataLen * sizeof(float));
  cudaMemcpy(device_x, host_x, kDataLen * sizeof(float),
             cudaMemcpyHostToDevice);

  // Launch the kernel.
  axpy<<<1, kDataLen>>>(a, device_x, device_y);

  // Copy output data to host.
  cudaDeviceSynchronize();
  cudaMemcpy(host_y, device_y, kDataLen * sizeof(float),
             cudaMemcpyDeviceToHost);

  // Print the results.
  for (int i = 0; i < kDataLen; ++i) {
    std::cout << "y[" << i << "] = " << host_y[i] << "\n";
  }

  cudaDeviceReset();
  return 0;
}
```

CUDA

Future Directions in Open Source

- Revive gdev.
- Gallium state tracker.
- Conversion tools to/from PTX.
- Clang as a Cuda frontend.
- Open source replacement for Cuda toolchain

Future Directions

Summary

Common trends for GPGPU API

- Smaller, low level APIs:
  - OpenCL Next
  - ROCm
  - CUDA Device API
- IR as driver input.
- Device language specs not part of API definition.

Thank You

Slide Contributors

Pekka Jääskeläinen

Jan Vesely

Aaron Watry

Pierre Moreau

References

- https://www.khronos.org/conformance/adopters/conformant-products
- https://www.khronos.org/assets/uploads/developers/library/2017-iwocl/IWOCL-Neil-Trevett-Keynote_May17.pdf
- https://github.com/shinpei0208/gdev
- https://cgit.freedesktop.org/mesa/mesa/
- https://github.com/pocl/pocl
- https://cgit.freedesktop.org/beignet/

References

- http://downloads.ti.com/mctools/esd/docs/opencl/index.html
- https://git.ti.com/opencl/ti-opencl
- https://git.linaro.org/gpgpu/shamrock.git
- https://github.com/RadeonOpenCompute/
- http://docs.nvidia.com/cuda/cuda-runtime-api/index.html
- https://github.com/google/clspv
- https://github.com/KhronosGroup/SPIRV-LLVM