Vulkanised - AN(G|C)LE as an OpenCL Compute Driver

Pavan Lanka, Samsung Austin Research Center (SARC)
Co-presenter Austin Annestrand, SARC
Co-presenter Gowtham Tammana, SARC
AGENDA

• Acknowledgements
• Recap
• ANGLE Introduction
• Why ANGLE for OpenCL?
• Current Progress
• Next Steps
Acknowledgements

CLSPV team - Alan, Romaric

Samsung AN(G|C)LE team

Google ANGLE team
Background

• Previously at Vulkanised 2023...

• Our Story:
  • Commitment to reduce the Total Cost of Ownership for various APIs including OpenCL

• Vulkan as HAL
What is OpenCL?

OpenCL is Widely Deployed and Used

The industry’s most pervasive, cross-vendor, open standard for low-level heterogeneous parallel programming

Molecular Modelling Libraries
CHARMM
GROMACS
Fram acoustic

Math and Physics Libraries
Mathematica
GNU Octave
Wolfram

Vision, Imaging and Video Libraries
VisionCpp
OpenCV

Machine Learning Compilers
Apocrypha

Visioning
Halide

OpenCL Accelerated Implementations
NVIDIA
AMD
arm
Qualcomm
Google
Imagination
SAMSUNG
ZSTech
Texas Instruments
VeriSilicon
XILINX

OpenCL is making a comeback thanks to ML
**ANGLE Introduction**

- Allow multiple flavors of OS to run OpenGL ES content seamlessly by translating to a native hardware-supported API.

- **NEW!**
  - Extend existing functionality to support Compute/OpenCL by translating OpenCL to Vulkan.
ANGLE Introduction Cont’d
Why add OpenCL to ANGLE?

OpenCL is a favored high-level (front-end) language!
  • Easier to write than VK (SPIR-V)

No need for a native implementation stack
  • Compute pipeline (less state, etc.), kernels compiled offline/optimized, low added overhead

Improved Ecosystem
  • Non-fragmented mobile driver story
  • GL/CL Inter-op
  • Flexible and Open Source
How are we doing this?

- Use open-source CLSPV compiler
  - (CL → SPIRV) translation

- Integrate CLSPV into ANGLE code base

- Implement CL Runtime in ANGLE
  - Leverage existing ANGLE infrastructure and SW Architecture
CLSPV / SPIR-V Language Ecosystem

The SPIR-V ecosystem includes a rich variety of language front-ends, tools and run-times.
Mapping OpenCL to Vulkan

Almost a 1-1 mapping for most pieces
  • e.g. Devices, Buffers, Images, Samplers, etc.

Pieces that are different still have similar API paradigms
  • clCreateCommandQueue → VKCommandBuffer
  • clCreateKernel → VKCreateComputePipeLines

Command processing
  • clEnqueue* → vkCmd*
Current Progress

• Target: OpenCL 3.0 Full Profile
  • Not Immediately Supported: SVM, Device Enqueue

• CLSPV has been validated for OpenCL 3.0: ~95% CTS passing.

• Started ANCLE development efforts ~6 months ago, and we are already ~50% CTS passing OpenCL 3.0.
Expected Performance*

### OpenCL Information
- **Platform Vendor**: Samsung Electronics Co., Ltd.
- **Platform Name**: Samsung Mobile GPU Platform
- **Device Vendor**: Samsung Electronics Co., Ltd.
- **Device Name**: Samsung Galaxy 920
- **Board Name**: Samsung Galaxy 920
- **Compute Units**: 3
- **Maximum Frequency**: 1366 MHz
- **Device Memory**: 4.00 GB

### Vulkan Information
- **Device Name**: Samsung Galaxy 920

### Vulkan Performance
- **Vulkan Score**: 12216
- **Background Blur**: 4567
  - 19.2 images/second
- **Face Detection**: 3120
  - 9.66 images/second
- **Horizon Detection**: 13709
  - 393.3 Mipiddles
- **Edge Detection**: 12315
  - 468.9 Mipiddles
- **Gaussian Blur**: 13253
  - 466.9 Mipiddles
- **Feature Matching**: 3529
  - 108.9 Mipiddles
- **Stereo Matching**: 41201
  - 50.9 Operations
- **Particle Physics**: 32022
  - 1620.2 TPF
Next steps

• 100% Conformance
• Performance/Profiling!
• Productize for upcoming mobile phones!
• Benefit from the Open Source Community
  • Portable OpenCL via a Vulkan HAL reality

Make OpenCL a first-class citizen in Android by relying on Vulkan as its Native Driver
Backup

Q&A

Thank you for your time
Example (SGEMM -CPU)

```c
for (int m=0; m<M; m++) {
    for (int n=0; n<N; n++) {
        float acc = 0.0f;
        for (int k=0; k<K; k++) {
            acc += A[k*M + m] * B[n*K + k];
        }
        C[n*M + m] = acc;
    }
}
```
Example (SGEMM - OpenCL)

```c
kernel = clCreateKernel(program, "myGEMM1", &err)
err = clSetKernelArg(kernel, 0, sizeof(int), (void*)&M);
err = clSetKernelArg(kernel, 1, sizeof(int), (void*)&N);
err = clSetKernelArg(kernel, 2, sizeof(int), (void*)&K);
err = clSetKernelArg(kernel, 3, sizeof(cl_mem), (void*)&A);
err = clSetKernelArg(kernel, 4, sizeof(cl_mem), (void*)&B);
err = clSetKernelArg(kernel, 5, sizeof(cl_mem), (void*)&C);
const int TS = 32;
const size_t local[2] = { TS, TS };
const size_t global[2] = { M, N };
err = clEnqueueNDRangeKernel(queue, kernel, 2, NULL,
global, local, 0, NULL, &event);
err = clWaitForEvents(1, &event);

// First naive implementation
__kernel void myGEMM(const int M, const int N, const int K,
    const __global float* A,
    const __global float* B,
    __global float* C) {

    // Thread identifiers
    const int globalRow = get_global_id(0); // Row ID of C[0..M)
    const int globalCol = get_global_id(1); // Col ID of C[0..N)

    // Compute a single element (loop over K)
    float acc = 0.0f;
    for (int k=0; k<K; k++) {
        acc += A[k*M + globalRow] * B[globalCol*K + k];
    }

    // Store the result
    C[globalCol*M + globalRow] = acc;
}
```

Classification: Samsung General Use
# API Mappings

<table>
<thead>
<tr>
<th>OpenCL API call</th>
<th>VK API Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>clCreateCommandQueue</td>
<td>vkCreateCommandPool</td>
</tr>
<tr>
<td>clCreateProgram/Kernel</td>
<td>vkCreateComputePipelines</td>
</tr>
<tr>
<td>clCreateBuffer</td>
<td>vkCreateBuffer + vkCreateDeviceMemory</td>
</tr>
<tr>
<td>clCreateImage</td>
<td>vkCreateImage + vkCreateDeviceMemory</td>
</tr>
<tr>
<td>clCreateSampler</td>
<td>vkCreateSampler</td>
</tr>
<tr>
<td>clEnqueue*</td>
<td>vkCmd*</td>
</tr>
<tr>
<td>clFinish</td>
<td>vkQueueWaitIdle</td>
</tr>
<tr>
<td>clGetProfilingInfo(START,END)</td>
<td>vkQueryPool*/vkCmdWriteTimeStamp</td>
</tr>
</tbody>
</table>
API translation Example

```
CL CreateBuffer: context = 0xb400007a441ba860, flags = 1, size = 4096, host_ptr = 0x0000000000000000, errcode_ret = 0x00000007fd33e5fd4
vkCreateBuffer
vkGetBufferMemoryRequirements
vkGetBufferMemoryRequirements
vkDestroyBuffer
vkCreateBuffer
vkGetBufferMemoryRequirements
vkGetBufferMemoryRequirements
vkAllocateMemory
vkBindBufferMemory
vkMapMemory
```

```
CL CreateKernel: program = 0xb400007a541ab330, kernel_name = 0x0000005906cb81cc, errcode_ret = 0x00000007fd33e5fd4
vkCreateDescriptorSetsLayout
vkCreatePipelineLayout
vkCreateDescriptorPool
vkAllocateDescriptorSets
```

```
CL EnqueueNDRangeKernel: command_queue = 0xb400007a541aaf10, kernel = 0xb400007a541ad380, work_dim = 1, global_work_offset = 0x0000000000000000, global_work_size = 0x00000007fd33e5fd0, local_work...
vkUpdateDescriptorSets
vkCreatePipelineCache
vkDestroyPipelineCache
vkCreateShaderModule
vkCreateComputePipelines
```
API translation Example Contd..

307   CL Finish: command_queue = 0xb400007a541aaf10
308   vkBeginCommandBuffer
309   vkCmdBindDescriptorSets
310   vkCmdPushConstants
311   vkCmdBindPipeline
312   vkCmdDispatch
313   vkCmdPipelineBarrier
314   vkCmdCopyBuffer
315   vkEndCommandBuffer
316   vkCreateFence
317   vkQueueSubmit
318   vkGetFenceStatus
319   vkGetFenceStatus
320   vkWaitForFences
321   vkGetFenceStatus
322   vkUnmapMemory
323   vkDestroyBuffer
324   vkResetCommandBuffer
325   vkFreeMemory
__kernel void math_kernel16( __global float16* out, __global float16* in )
{
    size_t i = get_global_id(0);
    out[i] = acosh( in[i] );
}

http://htmlpreview.github.io/?https://github.com/Khronos
Group/ SPIRV-
Registry/blob/master/nonsemantic/Nonsemantic_ClspvRefl
ection.html