EVALUATION OF MODERN GPGPU TECHNOLOGIES FOR IMAGE PROCESSING

JOACHIM MEYER
IMAGE PROCESSING
GPGPU ENGINEER

VISION. RIGHT. NOW.
(TOO?) MANY DIFFERENT GPGPU PROGRAMMING MODELS / APIS
(TOO?) MANY DIFFERENT GPGPU PROGRAMMING MODELS / APIS

Which one fits this awesome new project?
AGENDA

- SELECTION OF COMPARED APIS
- EVALUATION SETUP
- PERFORMANCE
- USABILITY
- PLATFORM INDEPENDENCE
- CONCLUSION
- FUTURE PROSPECTS
BASICS

Selection of APIs

- Vulkan
- CUDA
- OpenCL
- SYCL
BASICS

Test Project setup

- Targeting all 4 APIs + CPU reference implementation
- Targeted devices: CPUs & GPUs
- OSs: Windows & Linux 64bit
- Implementations:
  - CUDA 10.1
  - Vulkan 1.1
  - OpenCL 1.2
  - ComputeCpp (Win) & hipSYCL (Linux)
- Algorithms for polarization camera image processing
PERFORMANCE

It's comparable.
How hard is it?

**USABILITY**
# WHAT'S THE IMPLEMENTATION COST?

<table>
<thead>
<tr>
<th></th>
<th>CUDA</th>
<th>SYCL</th>
<th>OpenCL</th>
<th>Vulkan</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoC basic setup</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>65</td>
</tr>
<tr>
<td>LoC realistic setup</td>
<td>25</td>
<td>27</td>
<td>34</td>
<td>128 (+ 25 GLSL ➔ SPIRV)</td>
</tr>
<tr>
<td>LoC / new kernel</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>C++ kernels</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Implicit asynchronity</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Taskgraph</td>
<td>✅</td>
<td>✅</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TOOLS MAKE DEVELOPMENT EASIER

#### ANY TOOLS TO HELP?

<table>
<thead>
<tr>
<th>CUDA</th>
<th>SYCL</th>
<th>OpenCL</th>
<th>Vulkan</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Solid dev tooling:</td>
<td>• Hardly any specific tools, but native OpenCL / HIP tools usable</td>
<td>• Mostly vendor specific dev tools</td>
<td>• Mainly graphics focused tooling</td>
</tr>
<tr>
<td>• kernel debugging</td>
<td>• Host-device enables native IDE debugging</td>
<td>• LPGPU² CodeXL: generalization of AMD project</td>
<td>• Validation layers</td>
</tr>
<tr>
<td>• profiling</td>
<td></td>
<td></td>
<td>• Emulator (Talvos)</td>
</tr>
<tr>
<td>• IDE integration</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### LIBRARIES REDUCE DEVELOPMENT COST

#### LIBRARIES?

<table>
<thead>
<tr>
<th>Library</th>
<th>Details</th>
</tr>
</thead>
</table>
| CUDA    | Many optimized libraries  
      | FFT, BLAS, image processing, ... |
| SYCL    | Some libraries  
      | BLAS, DNN, RNG, Parallel STL, image processing  
      | Native (OpenCL / HIP) libraries usable |
| OpenCL  | Number of libraries with some device-specific optimization  
      | FFT, BLAS, DGEMM, image processing, ... |
| Vulkan  | Hardly any Compute specific libraries |

**LIBRARIES & SYCLcon**

10/04/2020 | Slide 11
APPLICATION ADOPTION AS INDICATOR FOR COMMUNITY SUPPORT

HELP ANYONE?

CUDA

- Widely used by scientists and application devs
- De-facto standard in ML libraries

SYCL

- Few applications known
  - Tensorflow
  - Eigen

OpenCL

- Wide adoption in consumer applications
  - Adobe Creative Cloud
  - Final Cut Pro

Vulkan

- Increasing adoption for mobile device support / combined with graphics
  - Adobe Premier Rush
  - OcataneRender

- SO Questions: 12.380
- SO Questions: 28
- SO Questions: 5.040
- SO Questions: 1.020
EACH API HAS ITS OWN (KERNEL) COMPILATION WORKFLOW

HOW DOES THE CODE COME TO LIFE?

CUDA source
- Clang
- nvcc
- PTX
- CUDA runtime

SYCL source
- ComputeCpp
- hipSYCL
- DPC++
- SPIR
- Executable

OpenCL C kernel
- to SPIR-V
- OpenCL driver

Vulkan GLSL kernel
- shaderc
- glslang
- Vulkan driver

Library
- libshaderc
VARIETY OF IMAGE FORMATS WITH VARIOUS DATA TYPES

HOW TO HANDLE DYNAMIC DATA TYPES?

CUDA
- Generic programming

SYCL
- Generic programming

OpenCL
- (Dynamic) online compilation with required data type as macro
- Preprocessor programming to dispatch temporary data types

Vulkan
- Online / offline compilation with required data type as macro / in shader names
- Preprocessor programming to dispatch temporary data types
Can it target XYZ?

PLATFORM INDEPENDENCE
### CAN IT TARGET XYZ?

<table>
<thead>
<tr>
<th></th>
<th>CUDA</th>
<th>SYCL</th>
<th>OpenCL</th>
<th>Vulkan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most recent version</td>
<td>10.2</td>
<td>1.2.1</td>
<td>2.2</td>
<td>1.2</td>
</tr>
<tr>
<td>Nvidia</td>
<td>10.2</td>
<td>1.2.1</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>AMD</td>
<td>HIP</td>
<td>1.2.1</td>
<td>2.0</td>
<td>1.2</td>
</tr>
<tr>
<td>Intel</td>
<td>1.2.1</td>
<td>2.1</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>ARM</td>
<td>1.2.1</td>
<td>2.1</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Linux</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>macOS</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Android</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>CPU</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>FPGAs</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PROJECTS INCREASING PORTABILITY OF THE APIS

PORTABILITY INITIATIVES

- SwiftShader
- clspv
- clvk
- hipCL
- CUDA-on-CL
- CLonD12

- CUDA-on-CL
- HIP
- clvk
- clspv
- SwiftShader
- CLonD12
CONCLUSION

So which API should be used?

<table>
<thead>
<tr>
<th>CUDA</th>
<th>SYCL</th>
<th>OpenCL</th>
<th>Vulkan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-source programming</td>
<td>Single-source programming</td>
<td>Cross-platform (incl. FPGAs,..)</td>
<td>Fully OS and GPU-vendor independent</td>
</tr>
<tr>
<td>Highly optimized and powerful libraries and tools</td>
<td>Multi-platform (incl. FPGAs,..)</td>
<td>Mature libraries</td>
<td>High setup cost but possibility to optimize</td>
</tr>
<tr>
<td>Vendor lock-in acceptable? (Maybe use HIP instead?)</td>
<td>Tools for underlying implementation usable</td>
<td>Big community</td>
<td>Lack of compute specific tooling &amp; libraries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not-up-to-date implementations</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Emerging SYCL-specific tool and library support</td>
<td></td>
</tr>
</tbody>
</table>

Full decision matrix: doi.org/10.1145/3388333.3388645
FUTURE PROSPECTS
SOME POSSIBLE DEVELOPMENTS

WHAT’S UP NEXT?

**CUDA**
- Extended ARM & data-center support
- Continuous optimization and feature updates
- Fast support for new GPU features
- HIP porting to Windows?

**SYCL**
- Maturing and optimization of implementations
- Extended hardware and OS support
- Removal of OpenCL as conformance required backend
- Specific tooling and libraries
- News @ IWOCL
- SYCL-on-Vulkan?

**OpenCL**
- (Hopefully) improved vendor support with OpenCL Next
- Updated to new hardware features
- Higher-level kernel language support
- News @ IWOCL

**Vulkan**
- Continued wide support
- Extended compute capabilities to serve as portability backend for other APIs
- Compute specific libraries & tools
- Fast support for new GPU features
THANK YOU VERY MUCH FOR YOUR ATTENTION

JOACHIM MEYER

STEMMER IMAGING AG
J.MEYER@STEMMER-IMAGING.COM
STEMMER-IMAGING.COM
JOAMEYER.DE

© Copyright STEMMER IMAGING AG. All rights reserved. All texts, images, graphics, sound-, video- and animation files, as well as their arrangements are copyright protected. Reprint, processing and duplication for commercial purposes or use on websites are forbidden. Some STEMMER IMAGING pages contain images that are subject to copyright of the respective owner.