Leveraging OpenCL to create differentiation

Salvatore De Dominicis
IWOCL 2015
Today’s opportunity

Differentiated multimedia applications

- Android’s expanded camera subsystem now modelled after professional camera
- In an attempt to maintain a consistent platform running on multiple SoCs, Google limits rate of adoption of new features

<table>
<thead>
<tr>
<th>Android version</th>
<th>Camera HAL</th>
<th>Addition to android.camera.hal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lollipop</td>
<td>Noise reduction</td>
<td></td>
</tr>
<tr>
<td>KitKat</td>
<td>3.1</td>
<td>None</td>
</tr>
<tr>
<td>JB MR2</td>
<td>3.0</td>
<td>None</td>
</tr>
<tr>
<td>JB MR1</td>
<td>2.0</td>
<td>HDR</td>
</tr>
<tr>
<td>JB</td>
<td></td>
<td>Auto focus</td>
</tr>
<tr>
<td>ICS</td>
<td></td>
<td>Video stabilization</td>
</tr>
<tr>
<td>ICS</td>
<td>1.0</td>
<td>Face detection</td>
</tr>
</tbody>
</table>
Today’s opportunity

**Differentiated multimedia applications**

- OEMs will choose SoCs that allow them to differentiate their Android products
- Features most requested are *computational photography* and *computer vision*:
  - Sensor processing – stereo, array and ToF
  - Panoramas – real-time and high-res
  - **Depth of field (focus stacking)**
  - Gesture recognition
  - Augmented reality – real lighting
- Bringing new features to market fast requires:
  - large amounts of processing power
  - programmability

*GPU compute* delivers high performance for many image processing algorithms

---

Notes:

- **CPU** Compute
- **GPU** Graphics and compute

---

© Imagination Technologies

IWOCL 2015 - 3
GPU increasingly dominates SoC area

Particularly in premium SoCs

- SMP configurations unlikely to scale efficiently beyond four CPUs
- GPU multi-processor and multi-pipe configurability enables far more extensive processor scaling
- OpenCL unlocks the full potential of GPUs

© Imagination Technologies
GPU increasingly dominates SoC area

Particularly in premium SoCs

- SMP configurations unlikely to scale efficiently beyond four CPUs
- GPU multi-processor and multi-pipe configurability enables far more extensive processor scaling
- *OpenCL unlocks the full potential of GPUs*
GPU increasingly dominates SoC area

Particularly in premium SoCs

- SMP configurations unlikely to scale efficiently beyond four CPUs
- GPU multi-processor and multi-pipe configurability enables far more extensive processor scaling
- OpenCL unlocks the full potential of GPUs
GPU increasingly dominates SoC area

- Rogue Series 8
- Rogue Series 7
- Rogue Series 6
- SGX 544

2012 to 2015

100s of Giga Flops

CPU
Compute
- SMP
- CPU0
- CPU1
- CPU2
- CPU3

GPU
Graphics and compute
- USC (16 pipes)
- USC (16 pipes)
- USC (16 pipes)
- USC (16 pipes)
- USC (16 pipes)
- USC (16 pipes)
- USC (16 pipes)
- USC (16 pipes)
- USC (16 pipes)

Scheduler

© Imagination Technologies
IWOCL 2015 - 7
GPU compute – more performance, less power

The correct application partitioning is critical to success

CPU
- Compute
- CPU0
- CPU1

Large Cache

GPU
- Graphics and compute
- Scheduler
  - GPU0
    - SIMD processor
    - Small Cache
  - GPU1
    - SIMD processor
    - Small Cache
  - GPU3
    - SIMD processor
    - Small Cache

Unified System Memory
GPU compute – more performance, less power

The correct application partitioning is critical to success

**CPU**
- Typical 1-2GHz
- Sequential processing
- Small data sets
- Heavily branched
- Low latency tasks
- Few ALUs
- Large Cache

**GPU**
- Typical 300-600MHz
- Parallel processing, large data sets, unbranched, highly repetitive tasks
- Many ALUs

Unified System Memory
The problem with SoC bandwidth

*SoC bandwidth is usually much more constrained than on desktop machines*

- In mobile SoCs the Unified system memory is shared between all the I.P. blocks
Android’s problem with buffer copies

- Android dictates the formats of camera and video data presented to apps developers
- The OS APIs may copy frames from one format to another
  - Unnecessarily increases bandwidth
  - Unnecessarily reduces achievable GPU compute performance
- Performance losses can be quickly compounded, especially when processing HD video content

Unnecessary copying of buffers can cripple performance and power
Zero-copy software: no redundant buffer copies

Direct processing of YUV semi-planar images

- Create an Android gralloc buffer, and create a native window from this buffer
- Use Imagination’s PowerVR Imaging Framework for Android to bind the gralloc buffer to the camera HAL
- Call eglCreateImageKHR with a special flag to create two or three EGLImageKHR images that point to the YUV planes
- Call clCreateFromEGLImageKHR to create OpenCL Image objects
- In the kernel, call read_imagef to sample Y, U and V values

```
__kernel void foo(image2d_t Y, image2d_t UV, sampler_t s) {
    float y = read_imagef(Y, s, (i,j)).x;
    float2 uv = read_imagef(UV, s, (i,j)).xy;
    ...
}
```
Zero-copy software: no redundant buffer copies

Direct processing of YUV planar images

- Create an Android gralloc buffer, and create a native window from this buffer
- Use Imagination’s PowerVR Imaging Framework for Android to bind the gralloc buffer to the camera HAL
- Call eglCreateImageKHR with a special flag to create two or three EGLImageKHR images that point to the YUV planes
- Call clCreateFromEGLImageKHR to create OpenCL Image objects
- In the kernel, call read_imagef to sample Y, U and V values

ANativeWindow
Gralloc buffer
NV21
image2d_t
EGLImageKHR
EGLImageKHR
EGLImageKHR

GPU
Graphics and compute

ISP
Camera

VDE
Encode, Decode

Memory controller, caches and interconnect

__kernel void foo(
    image2d_t Y,
    image2d_t U, image2d_t V,
    sampler_t s)
{
    float y, u, v;
    y = read_imagef(Y, s, (i,j)).x;
    u = read_imagef(U, s,f(i,j)).x;
    v = read_imagef(V, s,g(i,j)).x;
}
Zero-copy software: no redundant buffer copies

*Dynamically converting pixels from YUV to RGB color space*

- Create an Android gralloc buffer, and create a native window from this buffer
- Use Imagination’s PowerVR Imaging Framework for Android to bind the gralloc buffer to the camera HAL
- Enable the extension CL_IMG_YUV_image and call eglCreateImageKHR to create one EGLImageKHR image that points to the YUV image
- Call clCreateFromEGLImageKHR to create an OpenCL Image object
- In the kernel, call read_imagef to sample RGB values

```c
__kernel void foo(image2d_t RGBA, sampler_t s)
{
    float4 x;

    x = read_imagef(RGBA, s, (i,j));
    ...
}
```
PowerVR Imaging Framework for Android

Zero-copy extensions that OEMs need to enable differentiation

- A suite of software extensions that enables efficient interoperability of software running on PowerVR GPUs with many other SoC hardware blocks
  - Interoperable with
    - CPU
    - ISP
    - VDE
- Images produced by the ISP can be directly consumed by the GPU
PowerVR Imaging Framework for Android

Zero-copy extensions that OEMs need to enable differentiation

- A suite of software extensions that enables efficient interoperability of software running on PowerVR GPUs with many other SoC hardware blocks
- Interoperable with
  - CPU
  - ISP
  - VDE
- Images produced by the ISP can be directly consumed by the GPU
- Images produced by the GPU can be directly consumed by the CPU
- Many complex vision and computational software pipelines can be created, incorporating the VDE and other compatible hardware on the SoC

A suite of software extensions that enables efficient interoperability of software running on PowerVR GPUs with many other SoC hardware blocks.

- Interoperable with CPU, ISP, and VDE.
- Images produced by the ISP can be directly consumed by the GPU.
- Images produced by the GPU can be directly consumed by the CPU.
- Many complex vision and computational software pipelines can be created, incorporating the VDE and other compatible hardware on the SoC.
PowerVR imaging framework examples

**Image Stabilization**

- Reduces frame-to-frame jitter when user is walking/in motion
- Provides smooth recording and playback of user-generated content
- Improves low-light performance

Without GPU compute

With GPU compute

- H264 encode
- YUV-to-RGB conversion
- Apply transformation matrix
- Sensor acquisition
- Defective pixel correction
- Auto white balance
- Lens shading
- Read gyro data
- Construct transformation matrix
- Defective pixel correction
- Auto white balance
- Sensor acquisition
- YUV-to-RGB conversion
- Apply transformation matrix
- Read gyro data
- Construct transformation matrix
PowerVR imaging framework examples

Face Detection

- Accurate face detection enables camera auto-focus and auto-exposure
- Enables selective high-fidelity encoding of key regions of interest (and bit-rate savings on background)
Case study: Image processing on MT8135

Simple 3x3 edge detection on Y component

```c
__kernel __attribute__((reqd_work_group_size(32, 1, 1)))
void edgeDetect(__read_only image2d_t srcImageY, __write_only image2d_t dstImageY)
{
    sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE |
                        CLK_ADDRESS_CLAMP_TO_EDGE | 
                        CLK_FILTER_NEAREST;

    int2 coords = (int2)( get_global_id( 0 ), get_global_id( 1 ) );

    float luma;
    luma = read_imagef( srcImageY, sampler, coords + (int2)( 1, 0 ) ).x;
    luma += read_imagef( srcImageY, sampler, coords + (int2)( 0, 1 ) ).x;
    luma += read_imagef( srcImageY, sampler, coords + (int2)( 1, 1 ) ).x;
    luma -= read_imagef( srcImageY, sampler, coords + (int2)( -1, -1 ) ).x;
    luma -= read_imagef( srcImageY, sampler, coords + (int2)( 0, -1 ) ).x;
    luma -= read_imagef( srcImageY, sampler, coords + (int2)( -1, 0 ) ).x;

    write_imagef( dstImageY, coords, luma );
}
```
Case study: Image processing on MT8135

Free-running 1080p camera processing using CPU versus GPU

**CPU is throttled to 70% of peak performance**

**GPU sustains same performance at 2x less power**

(Current measurement for board excluding backlight)
Developer boards

- Many developer boards and OEM products are now available in the market with a PowerVR Rogue GPU and OpenCL driver
- Most platforms now support PowerVR imaging framework extensions

**Merrii OptimusBoard**  
AllWinner A80  
Rogue G6230

**Meizu MX4**  
MediaTek MT6595  
Rogue G6200

**ASUS ZenFone 2**  
Intel Atom Z3580  
Rogue G6430

**Dell Venue 7/8**  
Intel Atom Z3580  
Rogue G6430
Conclusion

- OpenCL has been successfully deployed in 2015 mobile and tablet products to enable new camera and multimedia use cases

- Sensibly partitioning an application across all available components including an ISP, CPU and GPU can help improve performance and reduce power consumption

- Efficient ‘zero-copy’ buffer management is crucial to avoid saturating the limited available SoC bandwidth

---

**Game changing technology available from Imagination**

- Imagination’s PowerVR imaging framework for Android provides everything needed to add new OpenCL-based software into a camera application

- Increasing availability in OEM products in 2015

[www.imgtec.com/gpucompute](http://www.imgtec.com/gpucompute)
Thank You