Renderscript Accelerated Advanced Image and Video Processing on ARM Mali T-600 GPUs

Lihua Zhang, Ph.D.
MulticoreWare Inc.
lihua@multicorewareinc.com
Overview

- More & more mobile apps are beginning to require higher compute performance, in order to deliver new cool features to end-users.

- RenderScript allows the programmer to develop high performance kernels that can leverage the power of GPU, which allows compute intensive parts to be accelerated, while freeing the CPU to manage user interactions, etc.

- This talk will describe the technical work involved in implementing high performance image and video processing engines based on RenderScript, leveraging both the ARM CPU and Mali GPU.
Agenda

- Introduction to Mali-T604 GPU
- Renderson Brief
- Renderson on Mali GPU
- Use Renderson to Accelerate Image & Video Processing
  - Image Processing
  - Video Processing
  - Highlights of Optimizations
- Summary
- About MulticoreWare Inc.
First GPU based on the Midgard architecture and offers scalability from one to four cores.

Support atomic & true IEEE double-precision floating-point math in hardware for Full Profile OpenCL & Renderscript compute support, world’s 1st for mobile devices

Production-quality software support, via a single driver stack for all multicore configurations

Multicore scheduling and performance scaling is fully handled within the graphics system, with no special considerations required from the application developer.
Renderscript Brief

- High performance computation API at the native level that developer writes in C (C99 standard).
- Give apps the ability to run operations with automatic parallelization across all available processor cores.
- Supports different types of processors such as the CPU, GPU or DSP.
  - Single code base to support different arch or different core#
  - Do not need to recompile
- Useful for apps that do image processing, mathematical modeling, or any operations that require lots of mathematical computation.
Two (same or different) Renderscript kernels can run on CPU & GPU in parallel without influencing each other.

- Key for heterogeneous compute

Currently, Renderscript kernel can run on Mali GPU automatically if your kernel,

- doesn’t use global variables
- has no recursive function calling
- does not call rsDebug function
Why used Renderscript in our image/video APKs?

- Video/image’s sizes are becoming bigger and bigger
- Advanced image/video processing algorithms need more compute power (such as deshake, deblur)
- Renderscript supports vector operations, suitable for RGBA compute

Two APKs developed & demoed in CES2013 & MWC2013,

- Advanced Photo Editing APK:
  - process images with filters such as Motionblur, Sobel, pencil, gray, blur, etc
- Video Transcode APK for video de-shake and upscaling
- By using Renderscript we are seeing 20X maximum and 4X average speedup on Mali-T604 GPU than on CPU.
Image Processing

- **Edit Mode**
  - allows user to take pictures and edit pictures
Image Processing

Image editing

- Up to 21 filers, and support both CPU and GPU
- Can save or share the result picture onto Facebook and Twitter.
21 filters implemented for Advanced Photo Editing APKs

- bw: black & white
- motionblur: motionblur effect
- negative: color negative
- oldphoto: oldphoto effect
- pixelate: pixelate effect
- sobel: edge detect
- relievo: relievo effect
- sepia: sepia effect
- emboss: emboss effect
- whirlpinch: Distortion effect
histogram: three colors’ histogram

gaussian: Gaussian blur effect

pencil: Sketch effect

histequal: histogram equal

cloud: cloudy effect

labyrinth: labyrinth effect

tile: tile effect

wave: wave effect

light: light effect

Bicubic: bicubic Interpolation scale

gray: gray effect
Live Camera support

- Select filters to apply to live video from device camera & check effects right away
- Choose between CPU and GPU
Batch mode

- To compare the performance between CPU and GPU for image processing

- Allows user to
  - Select single or multiple images
  - Select single or multiple filters
  - Select devices: CPU, GPU or both
Result: (for 2560x1920 image)

<table>
<thead>
<tr>
<th>Filter Name</th>
<th>CPU Time (ms)</th>
<th>GPU Time (ms)</th>
<th>X-Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>MotionBlur</td>
<td>3317.250</td>
<td>939.250</td>
<td>3.532</td>
</tr>
<tr>
<td>Cloud</td>
<td>3301.250</td>
<td>783.750</td>
<td>4.212</td>
</tr>
<tr>
<td>Labyrinth</td>
<td>2898.750</td>
<td>763.250</td>
<td>3.798</td>
</tr>
<tr>
<td>TileReflection</td>
<td>10588.250</td>
<td>1456.250</td>
<td>7.271</td>
</tr>
<tr>
<td>WhirlPinch</td>
<td>1244.500</td>
<td>343.750</td>
<td>3.620</td>
</tr>
<tr>
<td>Wave</td>
<td>1358.750</td>
<td>193.000</td>
<td>7.040</td>
</tr>
<tr>
<td>Bicubic</td>
<td>3282.250</td>
<td>213.250</td>
<td>15.392</td>
</tr>
</tbody>
</table>

Time remaining until next run: 27 seconds.
Video Processing

- MCW Video Transcode/Processing Pipeline
  - A well designed parallel pipeline is key to design performance critical video processing apps, e.g. DirectShow.
    - Android doesn’t supply APIs to help construct video pipeline
  - We used Khronos’ OpenMAX framework to implement video pipeline in native code with C++.
    - OpenMAX provides cross-platform open source API for multimedia applications
    - Its basic element is called *component*, and any *component* runs on its own thread in parallel.
    - By implementing some filters on CPU and others on GPU, we can use CPU & GPU together to achieve maximum throughput.
Some details of our pipeline

- HW decoder & HW encoder
- Color space conversion: YCbCr to RGBA
- Complicated filters can be split to multiple phases, and every phase as an independent component running in parallel.
Video Transcode APK

◆ Deshake

Before and after comparisons are shown, with a video player interface displaying the video's CPU and GPU performance metrics.
Video Transcode APK

◆ Upscaling

Rainbow Video Demo

CPU: 3 FPS  GPU: 20 FPS X-Factor: 6.667 Until next round reminding time 27 seconds.
Video Transcode APK

Filters

- A few filters implemented with Renderscript, and can execute on either CPU or GPU.
- As deshake needs lot of computation, we uses both CPU & GPU.

<table>
<thead>
<tr>
<th>Filter</th>
<th>FPS (GPU/CPU)</th>
<th>X-factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deshake (720P)</td>
<td>28 / 8</td>
<td>3.5</td>
</tr>
<tr>
<td>Upscaling (720P to 1080P)</td>
<td>20 / 3</td>
<td>6.7</td>
</tr>
</tbody>
</table>
Video Transcode APK

- Using GPU can gain more than just performance, but also help reduce energy consumption!

<table>
<thead>
<tr>
<th>Deshake transcode for 30 times</th>
<th>Total time (s)</th>
<th>Battery % before transcode</th>
<th>Battery % after transcode</th>
<th>Battery % consumed</th>
<th>Performance ratio (GPU vs CPU)</th>
<th>Energy efficiency (GPU vs CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1484.624</td>
<td>89%</td>
<td>79%</td>
<td>10%</td>
<td>2.9</td>
<td>5.0</td>
</tr>
<tr>
<td>GPU</td>
<td>504.691</td>
<td>91%</td>
<td>89%</td>
<td>2%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Highlights of Optimizations

- For Image processing

  - Optimized most of the algorithms in order to let them more suitable to run on GPU
    - Use vector to process one pixel’s RGBA channels together or several pixels together.
    - Use shift and logical operations to replace divide and modulo calculation.

  - Optimized Bitmap loading time
    - Load bitmaps in a separate background thread.

  - Live camera mode
    - Implemented YUV to RGBA with CPU using NEON as a new pipeline component to execute in a separate thread.
Highlights of Optimizations

➢ For video processing
  ➢ Optimized the deshake algorithm itself
  ➢ Adopt a pipeline & split deshake algorithm to two phases
    ➢ Compute frame’s transform parameters: on CPU with NEON
    ➢ Transform frame according to the transformation: on GPU with RS

➢ Some general opt hints
  ➢ Use intrinsic RS functions if possible (clamp, mad, dot …)
  ➢ Minimize branch codes (if, while, for …)
  ➢ Leave same calculation for all the RS threads outside the kernels, and feed to kernel as UserData parameter. This has helped reduce GPU computation a lot.
  ➢ Use integer instead of float to do the computation if precision is ok.
Mobile CPU & GPU are becoming more powerful, which has enabled the possibility of implementing real-time compute intensive “practical” applications, like advanced image & video processing, on mobile devices.

The best performance always comes out of efficient task allocations & scheduling on heterogeneous platform, i.e. tasks must be allocated to CPU & GPU, based on their types.

We can expect to see more & more mobile apps to follow the trend and these practice would also help shape proper tools chain to ease the software development effort while get better performance.
About MulticoreWare

- Founded in 2008
- Leading tools, libraries & services provider for heterogeneous computing
  - CTO: Prof. Wen-Mei Hwu from UIUC
- Our Mission
  - To empower people to leverage heterogeneous computing so as to deliver enhanced value everywhere
- One of largest heterogeneous computing teams worldwide
- More than 200 strong engineers globally with offices in the USA, China, India, and Singapore

St. Louis
Champaign IL

Changchun
Beijing

Chennai
Industry Leadership

• Tools leadership role on HSA Foundation
• Khronos Contributor Member

- Strategic relationship with University of Illinois at Urbana Champaign
- Partnerships with CPU/GPU/FPGA vendors
  - Mobile, Desktop, and Cloud
Expertise in LLVM and Clang based compiler technologies

- MxPA – High Performance OpenCL stack for CPUs
- GMAC – Automated Memory Data Movement and Coherence
- DL – Optimized Data Structure Layout
- TM – Heterogeneous Task Scheduler
- SM – Automated Packing of Work Items
- PPA – Profiler and Analyzer for Heterogeneous System
Heterogeneous Libraries

- MulticoreWare has built up accelerated libraries implemented in OpenCL, that we can integrate for use in your application

- Video
  - Broadcasting, Video Processing, Encoding, Decoding, Transcoding

- Imaging
  - Machine Vision, Image Filters, Feature Detection

- Security
  - Data Encryption

- Storage
  - Data Compression
Expertise across Platforms

- Video and Imaging implementations done across many platforms

- Experience across heterogeneous compute platforms
  - Mobile device platforms to workstations and cloud based platforms
  - AMD Fusion, Nvidia Kepler GPUs, ARM Mali and NEON
  - Altera OpenCL FPGA
  - FPGA development in HDL

- Experience across heterogeneous programming models
  - CUDA
  - OpenCL
  - Renderscript
  - C++AMP
  - etc
MulticoreWare Services

- MulticoreWare will work with you through your application development cycle
- Allows you to keep focus on your core product and feature development
- Leave all optimization work across different processor platforms to MulticoreWare
- Maximize your application’s performance with every new generation processor released
- Services involvement span from full software or silicon architecture design to porting, benchmarking, and support

Develop the Application

MulticoreWare optimizes

Multiple processor architectures
Products & Services for Mobile Platforms

- Mobile Video & Image Library (MVIL)
  - Across platforms
  - Easy to use APIs
  - Ready to use real-scenario samples for quick product prototype
  - Full parallel pipeline design to enable optimal performance & energy efficiency on heterogeneous mobile system

- Tools for Heterogeneous Mobile Compute
  - PPA for Android:
    - Powerful perf analysis tools for heterogeneous platform, which supports C/C++, Java, Renderscript & OpenCL
  - MxPA for Android:
    - Enable OpenCL on any multi-core mobile devices
    - Single code base across platforms

- Mobile GPU compute application services

- MCW China is going to hold technical seminars focusing on heterogeneous compute on mobile platforms
  - We are happy to share our knowledge and expertise and help promote the adoption of mobile GPU compute
Thank You!

Lihua Zhang, Ph.D.
Email: lihua@multicorewareinc.com

MulticoreWare website:
http://www.multicorewareinc.com