Accelerating Mobile Augmented Reality

Neil Trevett
Khronos President
Vice President Mobile Content, NVIDIA
Accelerating AR to Meet User Expectations

• Mobile is an enabling platform for Augmented Reality
  - Mobile SOC and sensor capabilities are expanding quickly

• But we need mobile AR to be 60Hz buttery smooth AND low power
  - Power is now the main challenge to increasing quality of the AR user experience

• Overview of silicon acceleration APIs on today’s mobile OS
  - And how they can be used to optimize AR performance AND power

• Highlight silicon-level opportunities and challenges still be to solved
  - While exploring the state of the art in mobile programming

SOC =
‘System On Chip’
Complete compute system minus memory and some peripherals
Mobile – a New Era in Computing

- IDC - 1.8 billion mobile phones will ship in 2012
  - By the end of 2016, 2.3 billion mobile phones will ship per year

Smartphones account for approximately half of the total phone market.

Mobile Industry is 20 years faster to 100M/year shipments than PC.
Mobile SOC Performance Increases

<table>
<thead>
<tr>
<th>Year</th>
<th>Mobile</th>
<th>Performance Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>Tegra 2</td>
<td>5x (KAL-EL)</td>
</tr>
<tr>
<td>2012</td>
<td>Tegra 3</td>
<td>10x</td>
</tr>
<tr>
<td>2013</td>
<td>LOGAN</td>
<td>50x</td>
</tr>
<tr>
<td>2014</td>
<td>STARK</td>
<td>75x</td>
</tr>
</tbody>
</table>

25x perf increase in next three years

- HTC One X
- Google Nexus 7
- Core i5
- Core 2 Duo
Mobile Thermal Design Point

Leakage current in latest silicon geometries means that Moore’s Law no longer delivers more performance without increasing power.

- Typical max system power levels before thermal failure:
  - 4-5” Screen takes 250-500mW
  - 7” Screen takes 1W
  - 10” Screen takes 1-2W

Resolution makes a difference!
- The iPad3 screen takes up to 8W

Even as battery technology improves - these thermal limits remain.

- 2-4W
- 4-7W
- 6-10W
- 30-90W
How to Save Power?

- Much more expensive to MOVE data than COMPUTE data
- Energy efficiency must now be key metric during silicon AND software design
  - Awareness of where data lives, where computation happens, how is it scheduled
- \textbf{Need} to use hardware acceleration
  - Lots of processing in parallel
  - Efficient caching and memory usage
  - Reduces data movement

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy Cost (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit Integer Add</td>
<td>1pJ</td>
</tr>
<tr>
<td>32-bit Float Operation</td>
<td>7pJ</td>
</tr>
<tr>
<td>32-bit Register Write</td>
<td>0.5pJ</td>
</tr>
<tr>
<td>Send 32-bits Off-chip</td>
<td>50pJ</td>
</tr>
<tr>
<td>Send 32-bits 2mm</td>
<td>24pJ</td>
</tr>
<tr>
<td>Write 32-bits to Memory</td>
<td>600pJ</td>
</tr>
</tbody>
</table>

For 40nm, 1V process, Write 32-bits to Memory = 600pJ
Example - Typical Camera ISP

- Camera ISP (Image Signal Processor) has little or no programmability
  - Scan-line-based, Data flows through compact hardware pipe
  - No global memory used to minimize power

- BUT... computational photography apps now want to mix non-programmable ISP processing with more flexible GPU or CPU processing
  - ISP pipelines will provide tap/insertion points to/from CPU/GPU at critical points

- ~760 math Ops
- ~42K vals = 670Kb
- ~250Gops @ 300MHz
Engineering Software Apps for Power

• Use Dark Silicon - specialized hardware only turned on when needed
  - Lots of space for transistors – just can’t turn them all on at same time
  - Increase locality and parallelism of computation to save power compared to programmable processors

• Be power smart when using programmable processors
  - Instrumentation for energy-aware compilers and profilers
  - Dynamic and feedback-driven software power optimization
  - Power optimizing compiler back-end compilers / installers

• Smart, holistic use of sensors and peripherals
  - Wireless modems and networks
  - Motion sensors, cameras, networking, GPS

Apple A5

Significant portion of die is already not CPU/GPU or IO = dedicated hardware
Programmers View of Typical SOC c. 2012

Unified memory is critical departure from PC architecture

CPU Complex
1-5 Cortex A9/A15 Cores with NEON
L1 Cache
L2 Cache

Unified Memory Controller for 0.5-2GB DDR3L/LP-DDR3

2D/3D GPU Complex
Graphics Cache

2D/3D GPU Complex

Significant programmable acceleration. Fully C-programmable soon. 10-50+ cores

HD Audio Engine / IO
Peripheral Busses
Sensor Array

1-4 Display Controllers

Image Signal Processor
Video Encoder Decoder
Vision Processor (Future)

1-3 Cameras

Typically NO or very limited programmable functionality

Automatic scaling of frequency/voltage and # cores to meet current software load

Software configurability

Software configurability

Programmable? Or dedicated hardware for power efficiency?
Khronos Connects Software to Silicon

ROYALTY-FREE, OPEN STANDARD APIs for advanced hardware acceleration

Graphics, video, audio, compute, visual and sensor processing

Low level silicon to software interface needed on every platform

Defines the forward looking roadmap for the silicon community

Shipping on billions of devices across multiple operating systems

Rigorous conformance tests for cross-vendor consistency

Khronos is OPEN for any company to join and participate

Acceleration APIs BY the Industry FOR the Industry
Example use of Khronos APIs in AR

Positional Sensors

Positional and GPS Sensor Data

Computer Vision and Tracking

Synchronization and sensor fusion

OpenVX / OpenCL

Video TAP to Vision Subsystem

Camera Processing

Control Camera, Preprocess and generate video streams

Video stream to GPU

3D Rendering and Video Composition On GPU

Audio Rendering

Application on CPUs

Position and Tracking Semantics

© Copyright Khronos Group 2012 | Page 11
3D API Family Tree

- **OpenGL ES 1.0**
- **OpenGL ES 1.1**
- **OpenGL ES 2.0**
- **OpenGL ES 3.0**

*Mobile 3D*

- **OpenGL ES 1.0**
- **OpenGL ES 1.1**
- **OpenGL ES 2.0**

*Desktop 3D*

- **OpenGL 1.3**
- **OpenGL 1.5**
- **OpenGL 2.0**
- **OpenGL 2.1**
- **OpenGL 3.0**
- **OpenGL 3.1**
- **OpenGL 3.2**
- **OpenGL 4.0**
- **OpenGL 4.1**
- **OpenGL 4.2**
- **OpenGL 4.3**

- **WebGL 1.0**
- **WebGL 2.0**
- **WebGL-Next**
- **ES-Next**
- **GL-Next**

**Fixed function 3D Pipeline**

**Programmable vertex and fragment shaders**

**ES3 is backward compatible so new features can be added incrementally**

**OpenGL 4.3 is a superset of DX11**
OpenCL – Heterogeneous Computing

A low-level, cross-platform, cross-vendor standard for harnessing all system compute resources

- **C Platform Layer API**
  - Query, select and initialize compute devices

- **Kernel Language Specification**
  - Subset of ISO C99 with language extensions
  - Well-defined numerical accuracy - IEEE 754 rounding with specified max error
  - Rich built-in functions: cross, dot, sin, pow, log ...

- **C Runtime API**
  - Runtime or build-time compilation of kernels
  - Execute compute kernels across multiple devices

One code tree can be executed on CPUs or GPUs
OpenMAX AL Streaming Media Framework

- Enables key video, image stream and camera use cases
  - Enables optimal hardware acceleration with app portability

- Create Media Objects to play and process images and video with AV sync
  - Connect to variety of input and output objects to PLAY and RECORD media

- Full range of video effects and controls
  - Including playback rate, post processing, and image manipulation

OpenMAX AL includes sophisticated camera controls

OpenMAX AL Streaming Media Framework Diagram:

- Sources: Analog Radio, Camera, Audio Input, URI, Memory
- DSrc
- Efficient data routing for CPU processing
- Data Tap to CPU
- OpenMAX AL Media Object Processing
- DSk
- Sinks: EGLStream to GPU, Audio Mix, Display Window, URI, Memory

EGLStreams extension enables efficient transfer of image stream to GPU texture memory
OpenVX

- **Vision Hardware Acceleration Layer**
  - Enable hardware vendors to implement accelerated imaging and vision algorithms
  - For use by high-level libraries or apps directly

- **Primary focus on enabling real-time vision**
  - On mobile and embedded systems

- **Diversity of efficient implementations**
  - From programmable processors to dedicated hardware pipelines

Dedicated hardware can help make vision processing performant and low-power enough for pervasive ‘always-on’ use
Portable Access to Sensor Fusion

Advanced Sensors Everywhere
RGB and depth cameras, multi-axis motion/position, touch and gestures, microphones, wireless controllers, haptics keyboards, mice, track pads

Apps request semantic sensor information
StreamInput defines possible requests, e.g. “Provide Skeleton Position” “Am I in an elevator?”

Processing graph provides sensor data stream
Utilizes optimized, smart, sensor middleware
Apps can gain ‘magical’ situational awareness

Apps Need Sophisticated Access to Sensor Data
Without coding to specific sensor hardware

E.g. align samples from camera and other sensors
## OS API Adoption

<table>
<thead>
<tr>
<th>Mobile Operating Systems</th>
<th>Installed Bases (millions of US users)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>60m</td>
</tr>
<tr>
<td>iPhone</td>
<td>48m</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>36m</td>
</tr>
<tr>
<td>Microsoft Other</td>
<td>24m</td>
</tr>
<tr>
<td>Microsoft Other</td>
<td>12m</td>
</tr>
</tbody>
</table>

### OpenGL ES 2.0
- Shipping - Android 2.2

### OpenSL ES 1.0 (subset)
- Shipping - Android 2.3

### OpenMAX AL 1.0 (subset)
- Shipping - Android 4.0

### EGL 1.4
- Shipping under SDK -> NDK

### Opera and Firefox WebGL now
- Chrome soon

### OpenGL 3.2
- on MacOS

### OpenCL 1.1
- on MacOS

### OpenGL ES 2.0
- on iOS

### Can enable on MacOS Safari
- iOS 5 enables WebGL for iAds

### Microsoft Windows RT:
- Only Microsoft native APIs
- HTML5 but not yet WebGL
Using Android Java for AR

- **SensorManager**
  - Positional and GPS Sensor Data
  - Computer Vision and Tracking
  - Synchronization and sensor fusion
  - Custom C code
  - No advanced camera controls: ROI, burst parameter control, focus modes, format choices

- **MediaSDK**
  - Camera Processing
  - Control Camera, Preprocess and generate video streams

- **JetPlayer**
  - Audio Rendering
  - Application on CPUs
  - 3D Rendering and Video Composition On GPU

- **Video stream to GPU**
  - SurfaceTexture
  - GLSurfaceView
  - 30FPS but 2-3 frames latency

- **Synchronization and sensor fusion**
  - Tracking needs to be in native code and accelerated on CPUs AND GPUs

- **Limited fusion and cross sensor synch. No virtual sensors**

Native APIs can supplement Android capabilities now and be absorbed into platform C and Java APIs by Google over time.
Portability with Middleware Engines

- **Middleware SDKs and tools for application developers**
  - Hides OS details – provides OS portability

- **Porting and optimization through close cooperation with silicon vendors**
  - NVIDIA has close relationships with Metaio and Unity
  - Optimizing for Tegra power/performance

Applications

Apps Engines

Native APIs and Java APIs
OpenGL ES, OpenMAX AL, EGL, OpenSL ES, StreamInput

Partners

© Copyright Khronos Group 2012 | Page 19
Summary

• Advances in SOC silicon processing and associated APIs are enabling significant new mobile use cases – including Augmented Reality

• Holistic cooperation between hardware and software needed to deliver increasing computational loads in a fixed power budget

• Cooperative API standards working hard to enable mobile OS and apps to tap into the potential of mobile SOC acceleration

• Middleware engines and SDKs can provide high-level functionality and optimized performance across platforms for apps developers

• www.khronos.org
• ntrevett@nvidia.com