

Mali GPU Developer Resources

Akshay Agarwal

Ecosystem Director, North America

Media Processing Division

May 2013



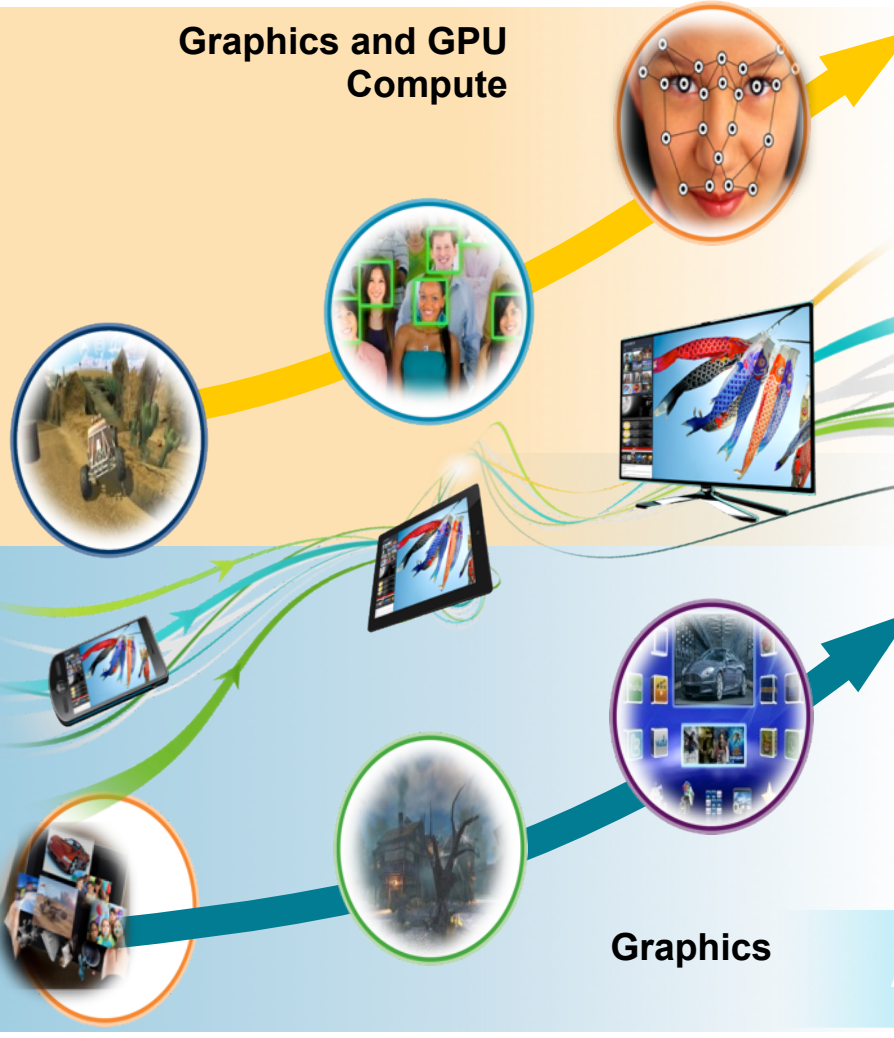
Agenda

- Mali GPU Overview
- Mali Developer Tools
 - Overview
 - DS-5 toolchain & streamline
 - Mali Graphics Debugger

Bridging Graphics Canyon – Mali GPUs

Performance

Graphics and GPU Compute

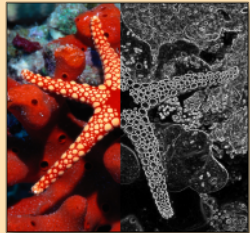


- Higher resolution displays (1080p → 4k)
- Enhanced performance for gaming with next generation Graphics API OpenGL ES 3.0 from Khronos
- New efficient texture compression standard – Adaptive Scalable Texture Compression (ASTC)
- First to market with Full Profile OpenCL™ & Renderscript for mobile

ALL WITHIN THE SAME SOC POWER ENVELOPE

Best for Graphics AND GPU Compute

Performance



Mali-T678

High end solution
Max compute capability
Optimized for tablets



Mali-T624

Mali-T628

50% performance uplift
OpenGL ES 3.0 support
Scalable to 8 cores

Mali-T604

First Midgard architecture product
OpenGL ES 3.0 support
Scalable to 4 cores

2012

2013

2014

● Date of production chips

Designed for GPU Compute

- Uncompromised support for OS / API choice
- Full Profile, 64-bit Compute

Closer CPU-GPU links

- Efficient use of all device resources
- Maximize performance and battery life
- Coherent memory links
- Right task in the right place

Protecting partner investments

- Common software platform reduces costs and TTM
- Multicore delivers performance scalability over multiple form factors

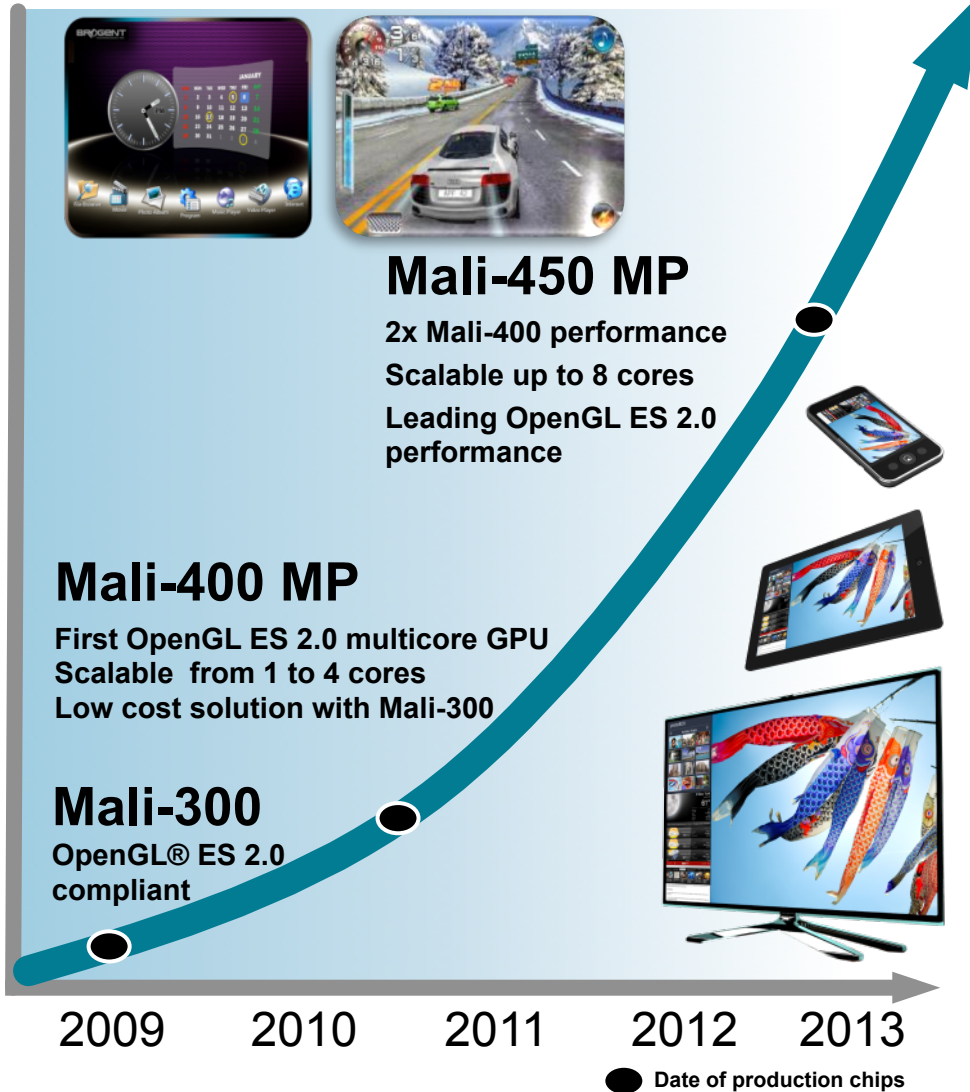
Advanced products in market early

- Mali-T604 silicon shipping now in consumer products

Product is based on a published Khronos Specification, and is expected to pass the Khronos Conformance Testing Process. Current conformance status can be found at www.khronos.org/conformance

Best for Graphics

Performance



Market leading performance density

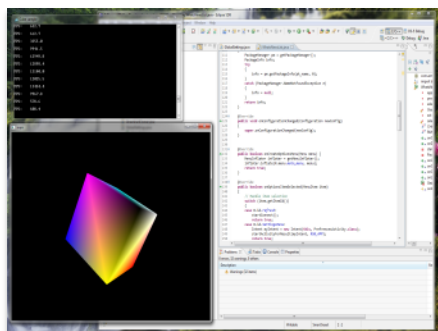
Multicore delivers performance scalability over many form factors

Common software platform reduces costs and TTM

Large, dynamic and vibrant ecosystem built on the success of over 150M Mali GPUs

Complete solution from ARM Hardware, Software & Support enabling fastest TTM for graphics IP

Mali Developer Tools



■ Software Development

- SDKs for OpenGL ES & Open CL
- OpenGL ES Emulators
- Fast Models
- Shader Development Studio
- Shader Library



■ Asset Creation

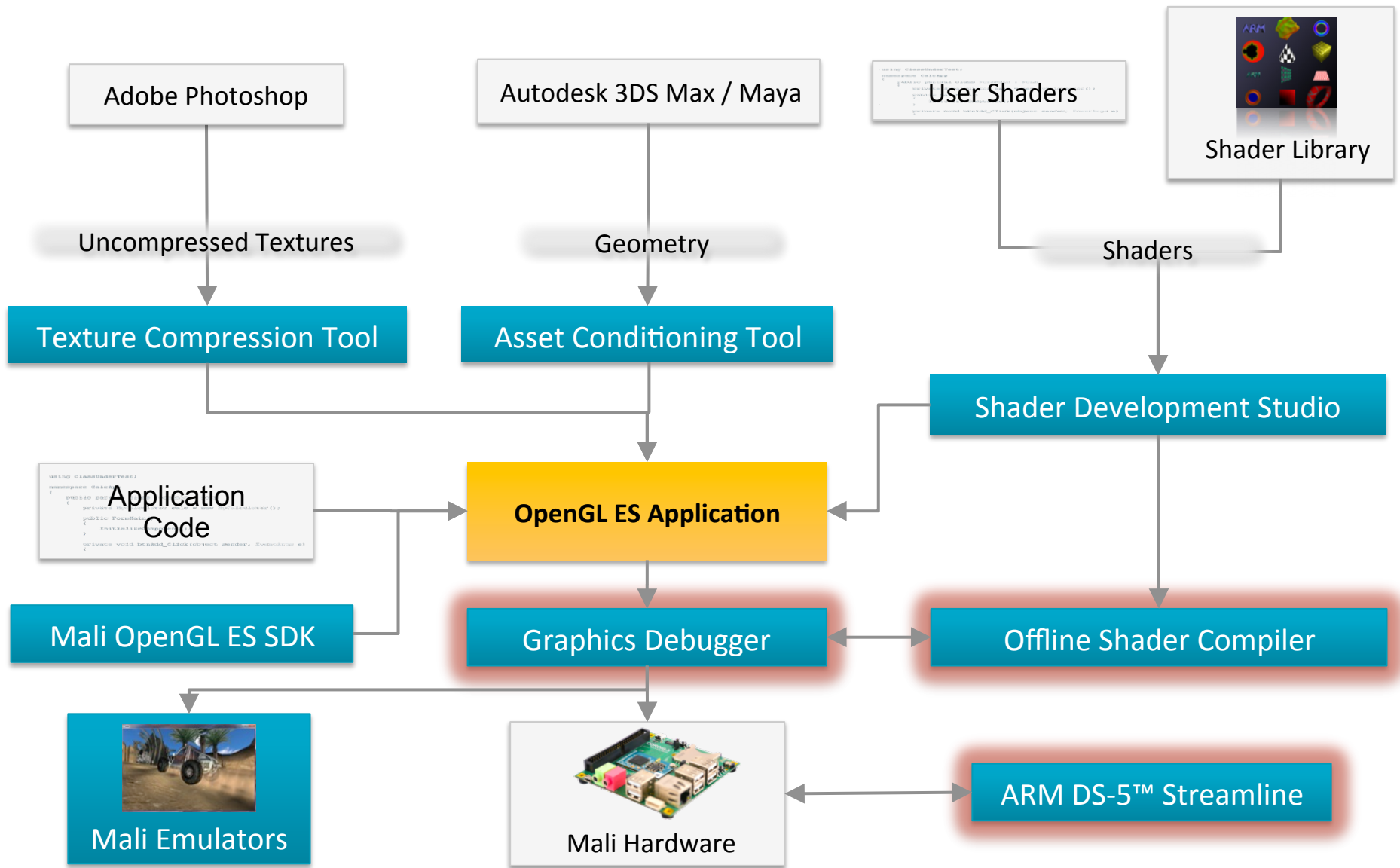
- Texture Compression Tool (includes ASTC)
- Asset Conditioning Tool
- Binary Asset Exporter



■ Performance Analysis & Debug

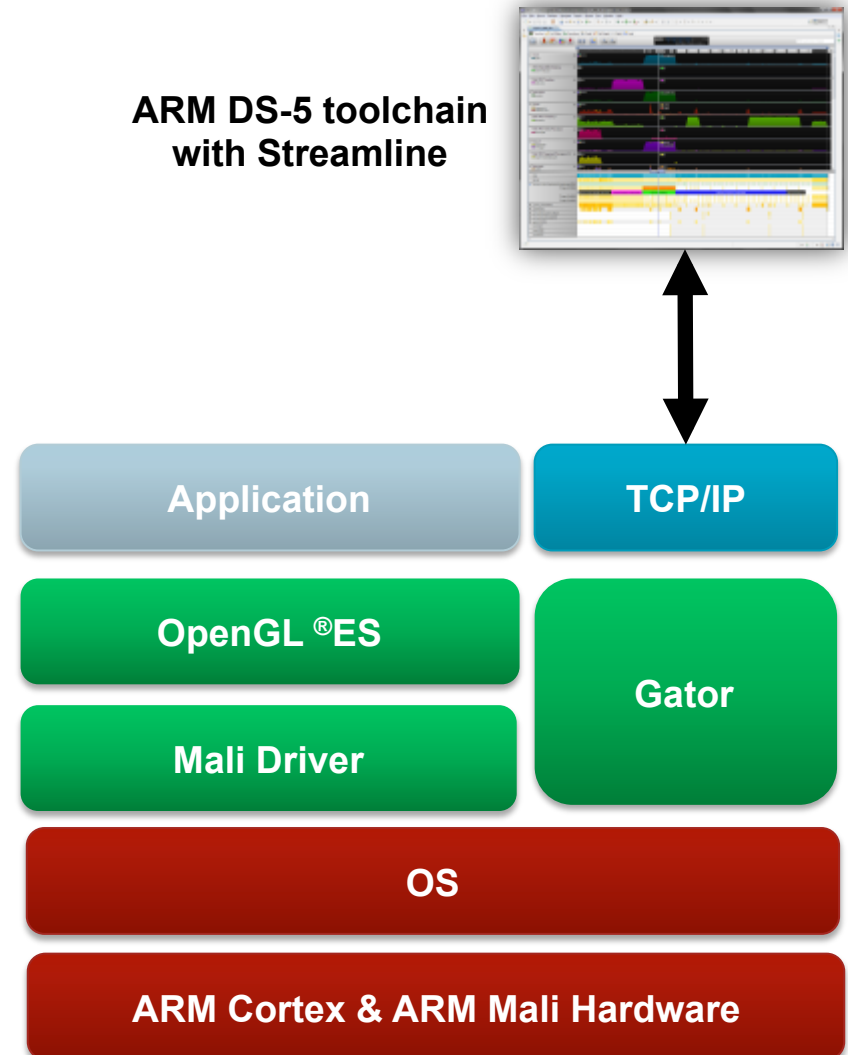
- DS-5 Toolchain & Streamline Performance Analyzer
- Mali Graphics Debugger
- Offline Shader Compiler

Mali Developer Tools Flow

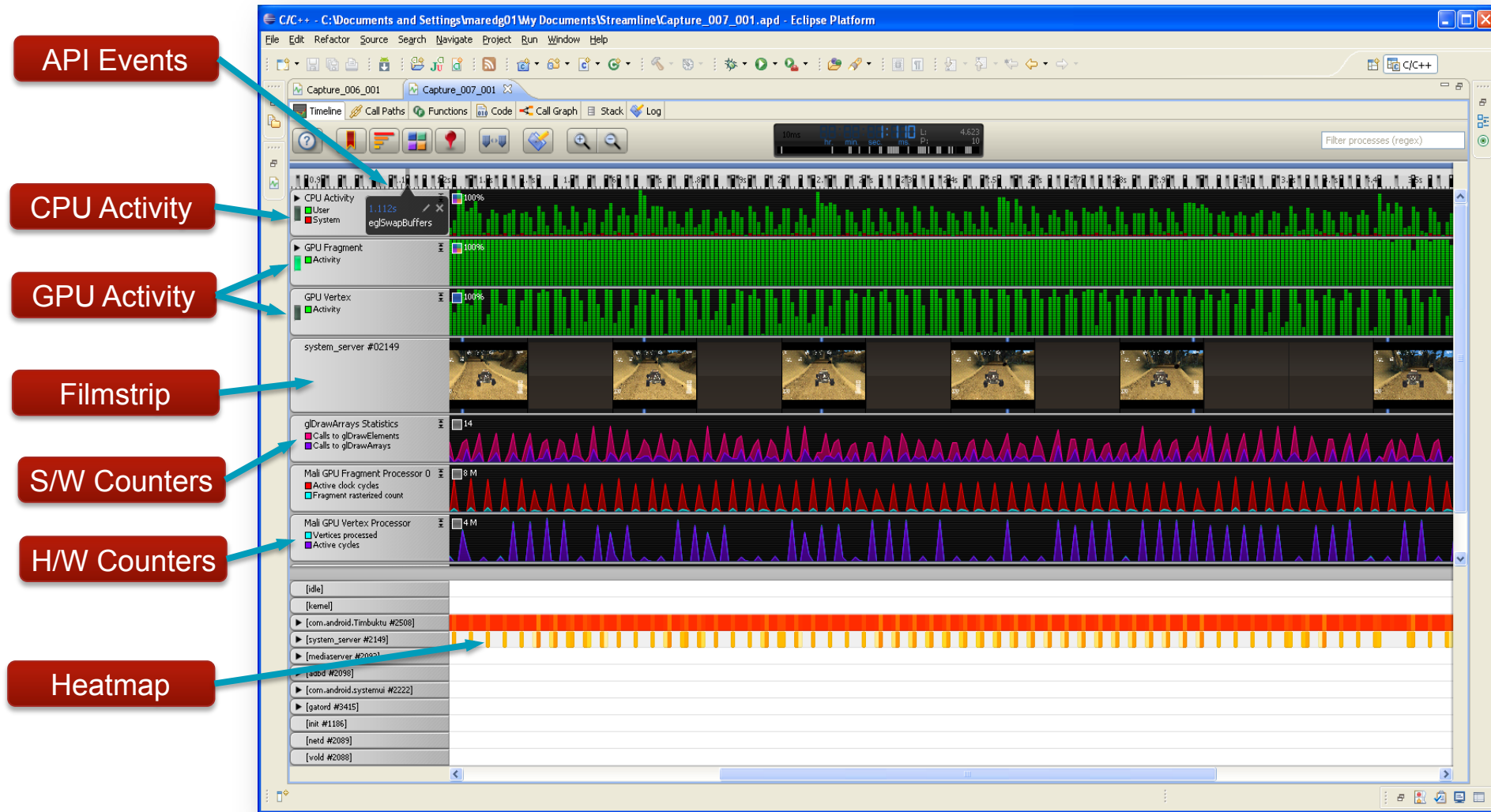


DS-5 Streamline Architecture

- Gator interfaces with Kernel & Mali Drivers
 - Extracts H/W & S/W counters
 - Extract frame buffer
 - Passes through events & annotations
 - Transmit data over TCP/IP to DS-5 tools
- Transparent to user application
 - Option to add annotations to user application for more advanced debugging
- Negligible performance impact
 - Zero impact when profiling disabled
 - Minimal impact in performance when enabled



Streamline Performance Analyzer



Offline Shader Compiler

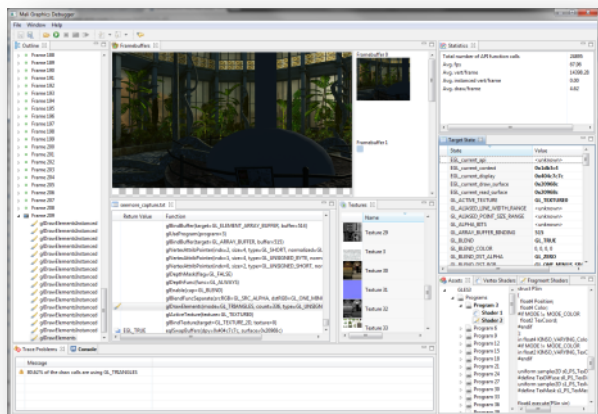
■ Offline Shader Compiler

- Compiles shader code written OpenGL ES Shading Language (ESSL) in offline mode
- Provides verbose shader performance & error messages for optimization and debug
- Support for:
 - Mali-200, Mali-400 and Mali-450,
 - Mali-T6xx
- Integration with Shader Development studio and Online Shader Editor
- **Release : v4.0 Available Now**

Mali Graphics Debugger

■ Mali Graphics Debugger

- Graphics debugging and performance debugging for content developers
- Understand issues and causes at frame level
- Support for OpenGL ES 2.0, 3.0, EGL & OpenCL 1.1
- Complimentary to DS-5 Streamline
- **Available NOW**



Mali Graphics Debugger

Frame Outline

Framebuffer / Render Targets

State View

Frame Stats

The screenshot displays the Mali Graphics Debugger interface with several panels:

- Frame Outline:** A tree view on the left showing a list of frames from 184 to 209, with Frame 209 selected.
- Framebuffer / Render Targets:** A central view showing a 3D scene of a modern building interior. To its right, a 'Framebuffer 0' and 'Framebuffer 1' are shown as small images on a checkerboard background.
- State View:** A table on the right showing the current state of various OpenGL ES parameters. For example, `GL_ACTIVE_TEXTURE` is set to `GL_TEXTURE0`.
- Frame Stats:** A statistics panel on the far right showing performance metrics: Total number of API function calls (24895), Avg. fps (67.96), Avg. vert/frame (14398.28), Avg. instanced vert/frame (0.00), and Avg. draw/frame (4.62).
- Dynamic Help:** A console at the bottom left showing a message: "80.62% of the draw calls are using GL_TRIANGLES".
- API Trace:** A panel below the frame outline showing a list of OpenGL ES function calls, such as `glUnmapBuffer`, `glBindBuffer`, and `glUseProgram`.
- Textures:** A table below the API trace listing textures: Texture 29 (1024 x 1024, GL_COMPRESSED_RGBA8_ETC2), Texture 3 (256 x 256, GL_RGBA4), Texture 30 (1024 x 1024, GL_COMPRESSED_RGBA8_ETC2), Texture 31 (1024 x 1024, GL_COMPRESSED_RGBA8_ETC2), and Texture 32 (1024 x 1024, GL_COMPRESSED_RGBA8_ETC2).
- Asset/Shader View:** A panel on the bottom right showing a tree view of assets and shaders, including a GLSL shader program with code like `float4 Position;` and `float4 Color;`.

Dynamic Help

API Trace

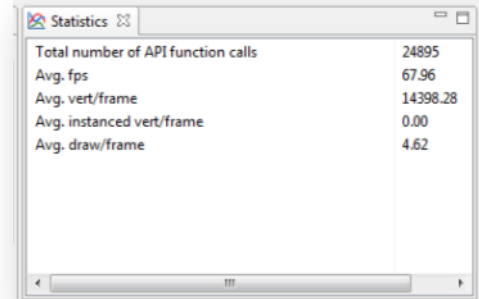
Textures

Asset/Shader View

Identify Issues

■ Statistics View to identify :

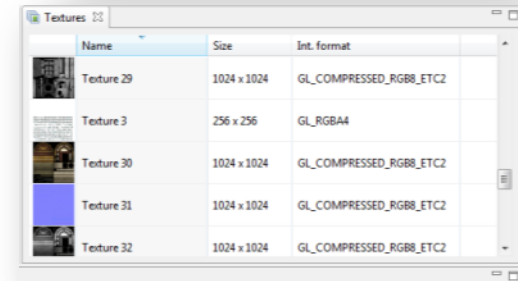
- High vertex count,
- Large number of draw calls



Metric	Value
Total number of API function calls	24895
Avg. fps	67.96
Avg. vert/frame	14398.28
Avg. instanced vert/frame	0.00
Avg. draw/frame	4.62

■ Texture View to identify :

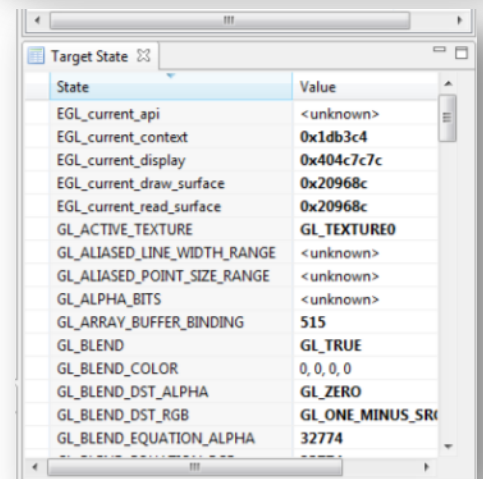
- Frames where compression is not being used
- Unnecessarily large texture dimensions
- Unnecessary pixel format being used



Name	Size	Int. format
Texture 29	1024 x 1024	GL_COMPRESSED_RGBA_ETC2
Texture 3	256 x 256	GL_RGBA4
Texture 30	1024 x 1024	GL_COMPRESSED_RGBA_ETC2
Texture 31	1024 x 1024	GL_COMPRESSED_RGBA_ETC2
Texture 32	1024 x 1024	GL_COMPRESSED_RGBA_ETC2

■ Target State View to identify :

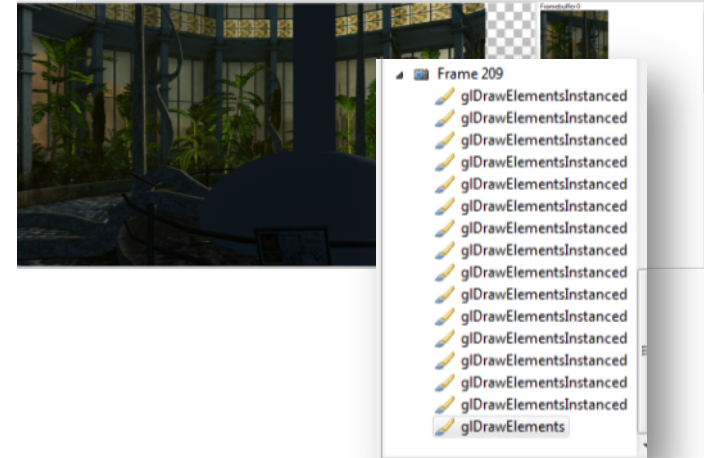
- Surprise changes to state
- Unnecessary changes



State	Value
EGL_current_api	<unknown>
EGL_current_context	0x1db3c4
EGL_current_display	0x404c7c7c
EGL_current_draw_surface	0x20968c
EGL_current_read_surface	0x20968c
GL_ACTIVE_TEXTURE	GL_TEXTURE0
GL_ALIASED_LINE_WIDTH_RANGE	<unknown>
GL_ALIASED_POINT_SIZE_RANGE	<unknown>
GL_ALPHA_BITS	<unknown>
GL_ARRAY_BUFFER_BINDING	515
GL_BLEND	GL_TRUE
GL_BLEND_COLOR	0, 0, 0, 0
GL_BLEND_DST_ALPHA	GL_ZERO
GL_BLEND_DST_RGB	GL_ONE_MINUS_SRC
GL_BLEND_EQUATION_ALPHA	32774

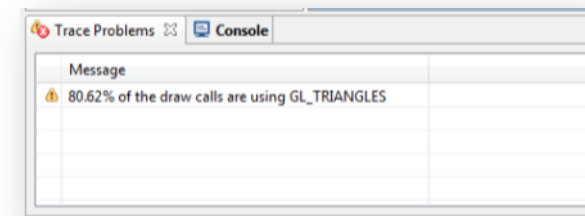
Identify Issues

- **Draw-Call stepping to identify :**
 - Redundant draw-calls
 - Potential opportunity for batching
- **Shader Statistic View to identify :**
 - Expensive shaders by cycle count
- **Dynamic Help to highlight optimization opportunity**



A screenshot of a Shader Statistic View table. The table has four columns: 'Name', 'Instruc...', 'Shortes...', and 'Longes...'. The rows list various shaders and their corresponding instruction counts.

Name	Instruc...	Shortes...	Longes...
Shader 2	47	47	47
Shader 11	34	34	34
Shader 14	33	33	33
Shader 5	23	23	23
Shader 8	23	23	23
Shader 16	23	23	23



For More Details, visit us at:
<http://malideveloper.arm.com>

Mali DEVELOPER CENTER

search... 

LEARN about Mali ▾

DEVELOP for Mali ▾

ENGAGE with Mali ▾

ARM ▾



OpenGL ES 3.0 Developer Resources
OpenGL ES 3.0 SDK • OpenGL ES 3.0 Emulator • Texture Compression Tool

ARM LINKS

- ARM Homepage
- About ARM
- ARM Products
- ARM Markets
- ARM Community
- ARM Support

Welcome to Mali Developer Center

GETTING STARTED

FORUM