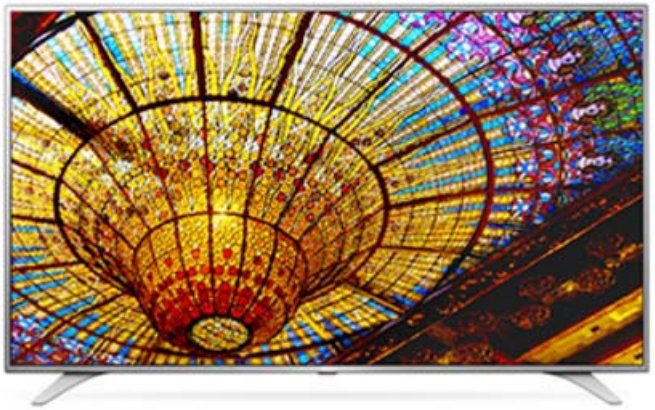
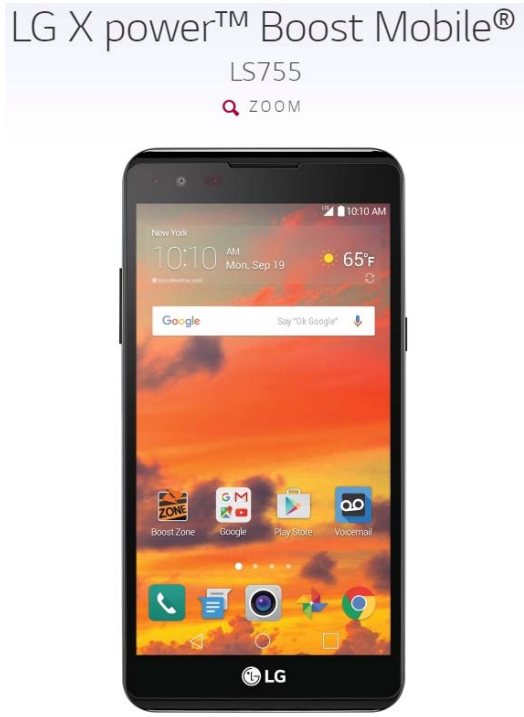
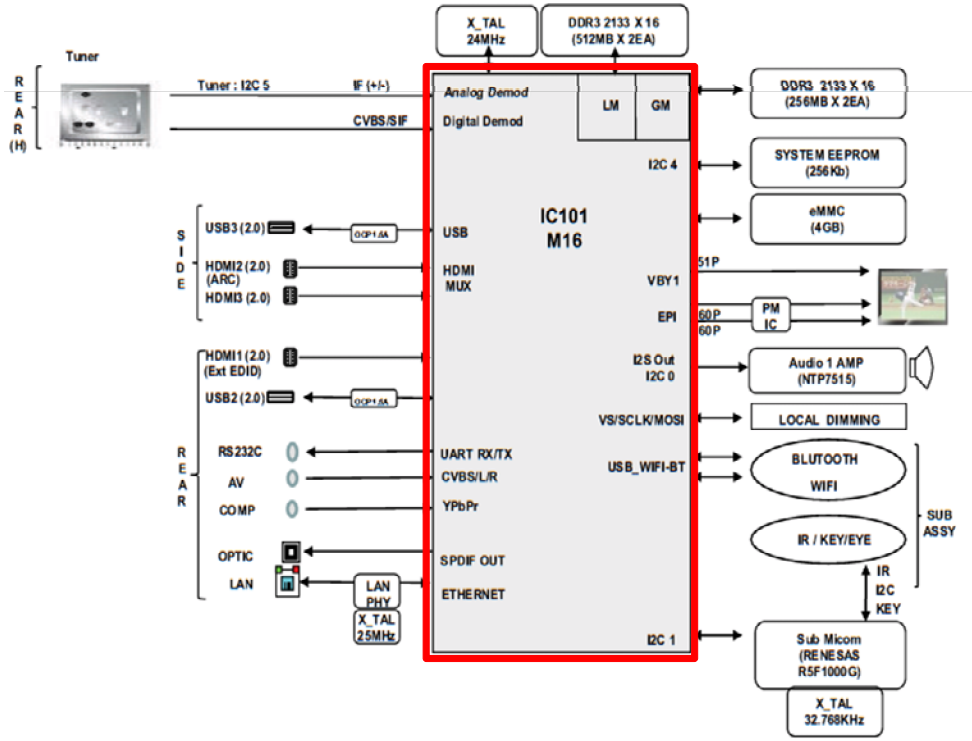
PUBLIC
EXHIBIT 20

U.S. Patent No. 7,796,133

LG / MediaTek Products

1. A unified shader comprising:	The LG 49UH6500 television and X Power LS755 phone (collectively, the "LG Products") include a unified shader.  *See* http://www.lg.com/us/support-product/lg-49UH6500.  *See* http://www.lg.com/us/cell-phones/lg-LS755-x-power-boost-mobile.

The LG Products include one of the following System-on-Chips (SoCs): M16 and MediaTek MT6755M.



See LG LED TV Service Manual, Chassis: UA63J, Model: 43UH6500, p.28, *available at* https://lg.encompass.com/shop/model_research_docs/?file=/ZEN/sm/43UH6500UB.pdf.[1/]

[1/] The LG 49UH6500 television and the LG 43UH6500 television are part of the LG UH6500 Series televisions. *See* http://www.lg.com/us/support/products/documents/UH6500_Series_Spec_Sheet_Updated_10112016.pdf.

Technical Specifications

Carrier	Boost Mobile®
Display	5.3" (1280 x 720) HD TFT Display
Battery	4,100 mAh non-removable
Platform	Android 6.0.1 Marshmallow
Processor	MediaTek 1.8 GHz Octa-Core MT6755M

See http://www.lg.com/us/cell-phones/lg-LS755-x-power-boost-mobile.

The SoCs include one of the following ARM Mali graphics processing units (the "Mali GPUs"): T760 MP2 and T860 MP2.

		M16
Smart Function	CPU	CA53 x4 1.1GHz / 1MB
	GPU	Mali T760 MP2 (650MHz)
	OSD	Separated 2K@60p
	HEVC	4K @60,10bit
	DDR	DDR3-2133/ DDR4-2400
	Audio DSP	HiFi3 Dual @370MHz

See LG LED TV Service Manual, Chassis: UA63J, Model: 43UH6500, p.123, *available at* https://lg.encompass.com/shop/model_research_docs/?file=/ZEN/sm/43UH6500UB.pdf.

	MediaTek MT6755 Helio P10 Specs **Release** Q4 2015 **Process** 28nm **Apps CPU** 8x Cortex-A53, up to 2.0GHz **GPU** ARM Mali-T860 MP2 at 700 MHz *See* http://cnoemphone.com/blog/mediatek-mt6755-helio-p10-specs-benchmark-and-smartphone-list. The Mali GPUs share substantially similar structure, function, and operation.

High performance

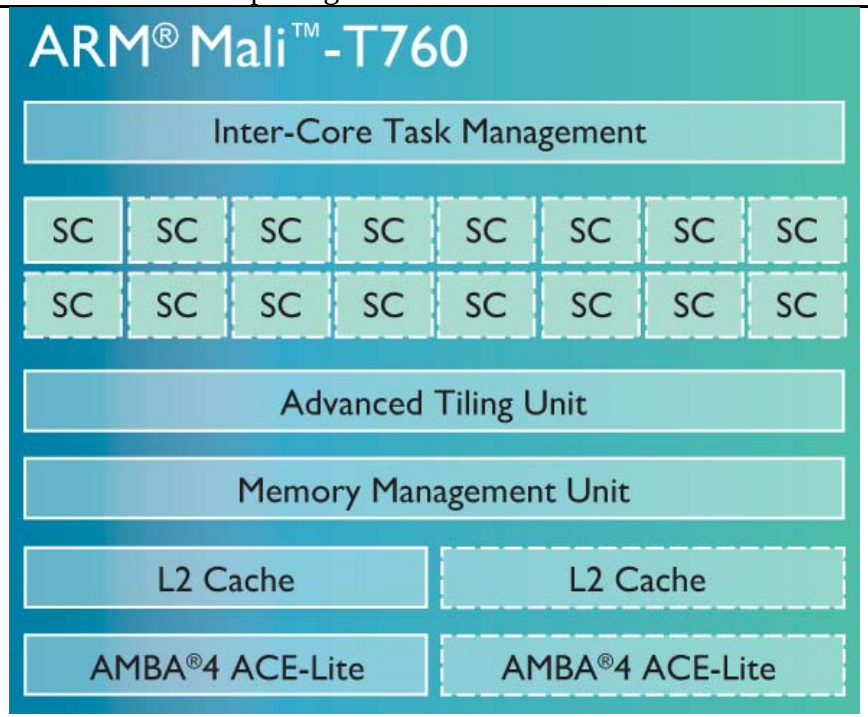
With stunning graphics capabilities, ARM Mali High Performance GPUs combine GPU Compute functionality with micro-architecture enhancements and system-wide, bandwidth-saving technology to bring energy efficiency to advanced mobile and consumer devices. GPU Compute solutions enable each task to be executed on the most suitable processor within the system. The resultant efficiencies guarantee superior graphics performance and extended battery life.

High performance GPUs:

- Mali-G71
- Mali-T860 & T880
- Mali-T760
- Mali-T628
- Mali-T624

See http://www.arm.com/products/graphics-and-multimedia/mali-gpu/

The Mali GPUs include a unified shader.

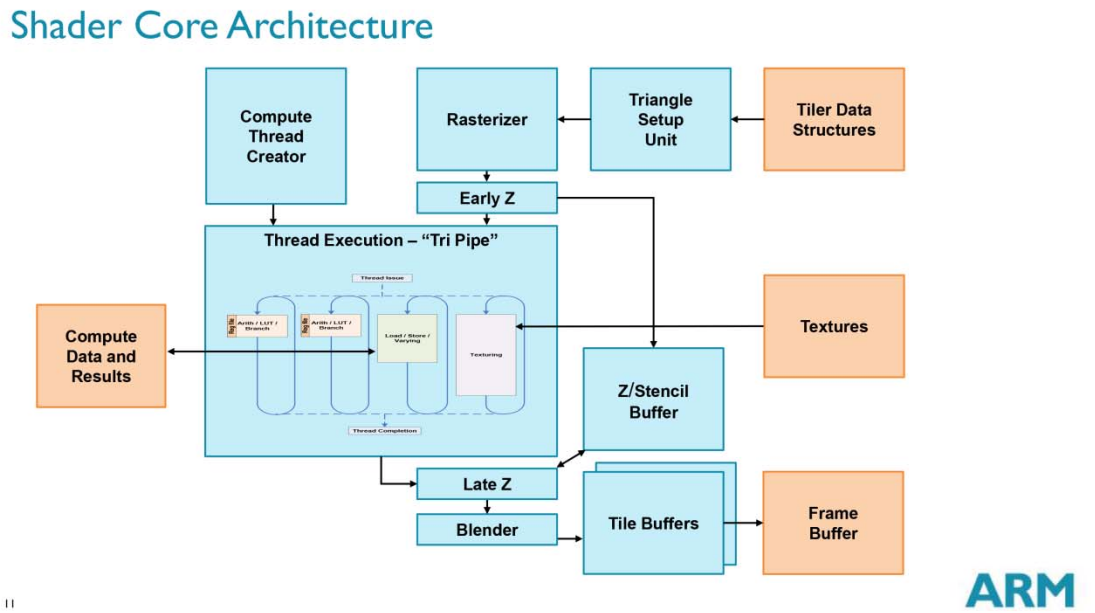


See http://www.arm.com/products/multimedia/mali-gpu/high-performance/mali-t860-t880.php.

GPU Architecture

The "Midgard" family of Mali GPUs (the Mali-T600 and Mali-T700 series) use a unified shader core architecture, meaning that only a single type of shader core exists in the design. This single core can execute all types of programmable shader code, including vertex shaders, fragment shaders, and compute kernels.

See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

an input interface for receiving a packet from a rasterizer;	The LG Products include an input interface for receiving a packet from a rasterizer. For example, "[t]he Mali shader core is structured as a number of fixed-function hardware blocks wrapped around a programmable tri-pipe execution core. The fixed function units perform the setup for a shader operation - such as rasterizing triangles or performing depth testing - or handling the post-shader activities - such as blending, or writing back a whole tile's worth of data at the end of rendering. The tripipe itself is the programmable part responsible for the execution of shader programs." *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core  *See* Ryan Smith, ARM's Mali Midgard Architecture Explored, http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.

Primitive assembly

In primitive assembly the vertices are assembled into geometric primitives. The resulting primitives are clipped to a clipping volume and sent to the rasterizer.

Rasterization

Output values from the vertex shader are calculated for every generated fragment. This process is known as interpolation. During rasterization, the primitives are converted into a set of two-dimensional fragments that are then sent to the fragment shader.

Transform feedback

Transform feedback, enables writing selective writing to an output buffer that the vertex shader outputs and is later sent back to the vertex shader. This feature is not exposed by Unity but it is used internally, for example, to optimize the skinning of characters.

Fragment shader

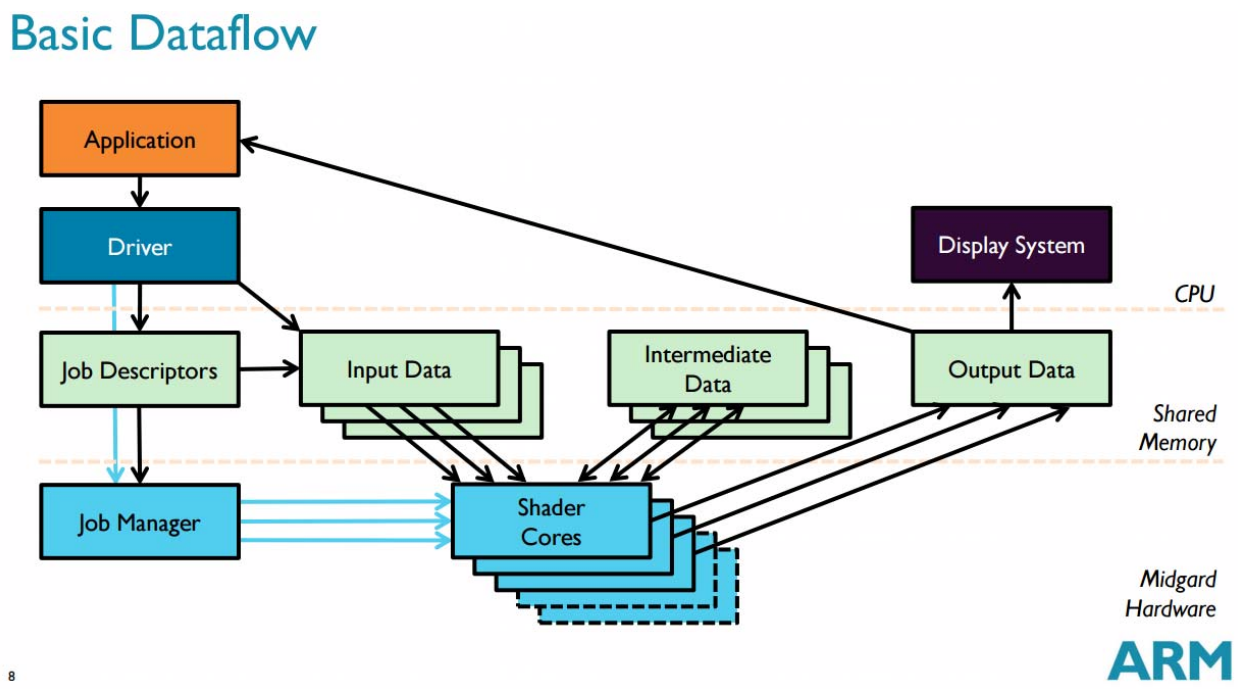
The fragment shader implements a general-purpose programmable method for operating on fragments before they are sent to the next stage.

Per-fragment operations

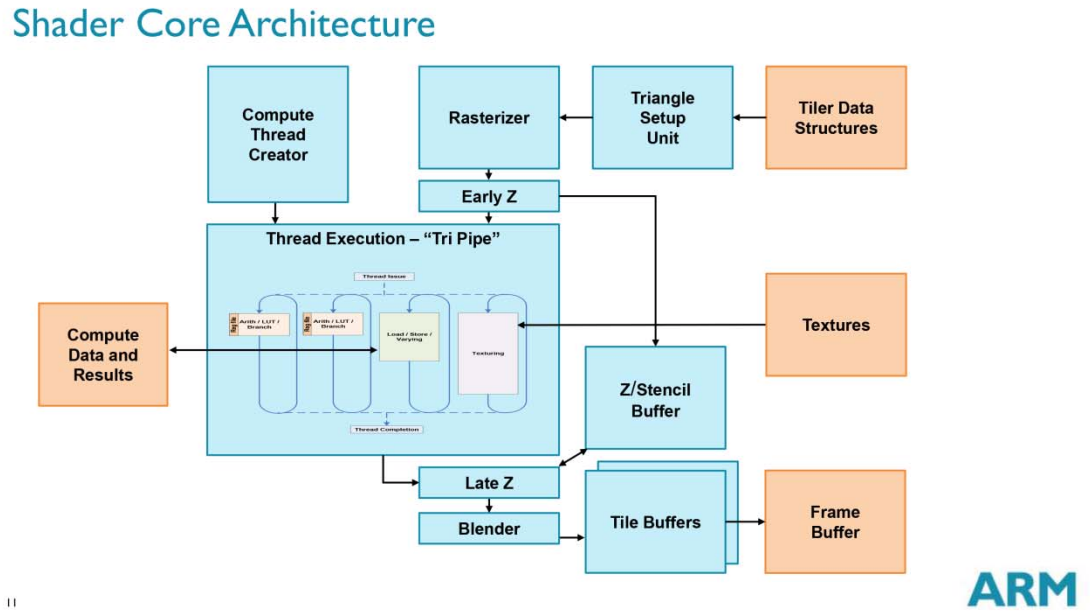
In Per-fragment operations several functions and tests are applied on each fragment: pixel ownership test, scissor test, stencil and depth tests, blending and dithering. As a result of this per-fragment stage either the fragment is discarded or the fragment color, depth or stencil value is written to the frame buffer in screen coordinates.

See "ARM Guide to Unity" Version 2.1 page 6-77 available at http://infocenter.arm.com/help/topic/com.arm.doc.100140_0201_00_en/arm_guide_to_unity_enhancing_your_mobile_games_100140_0201_00_en.pdf (accessed 10/27/2016)

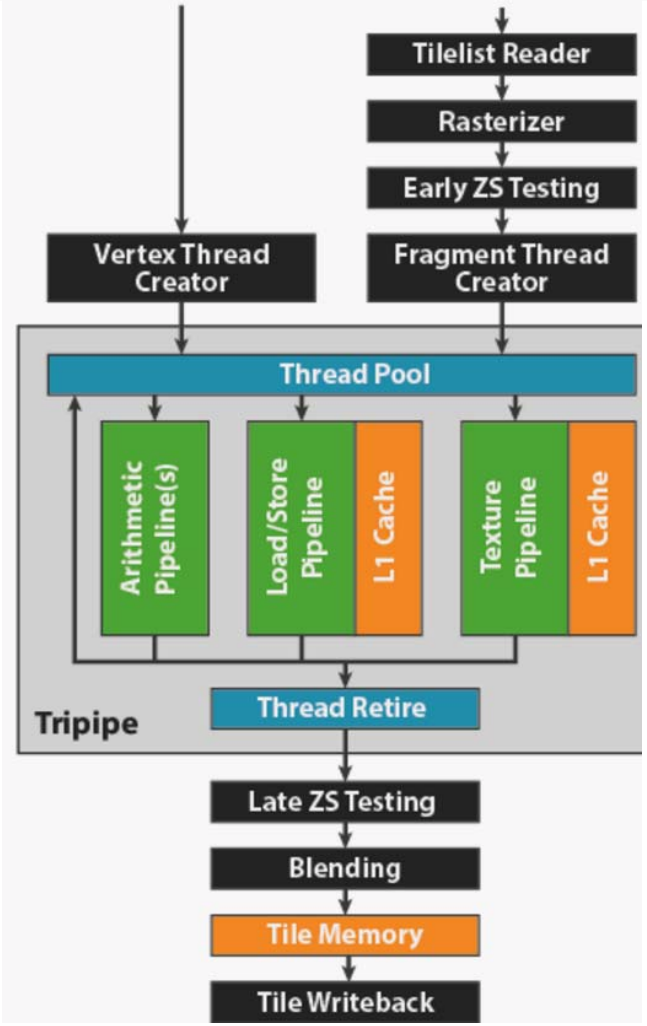
"a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations,"

a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations,	The LG Products include a shading processing mechanism configured to produce a resultant value from a rasterizer packet by performing one or more shading operations. For example, the Mali GPU has three classes of pipelines "in the tripipe design: one handling arithmetic operations, one handling memory load/store and varying access, and one handling texture access. There is one load/store and one texture pipe per shader core, but the number of arithmetic pipelines can vary." *See* http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.  *See* http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/2-Mali-GPU-architecture-overview-and-tile-local-storage.pdf.

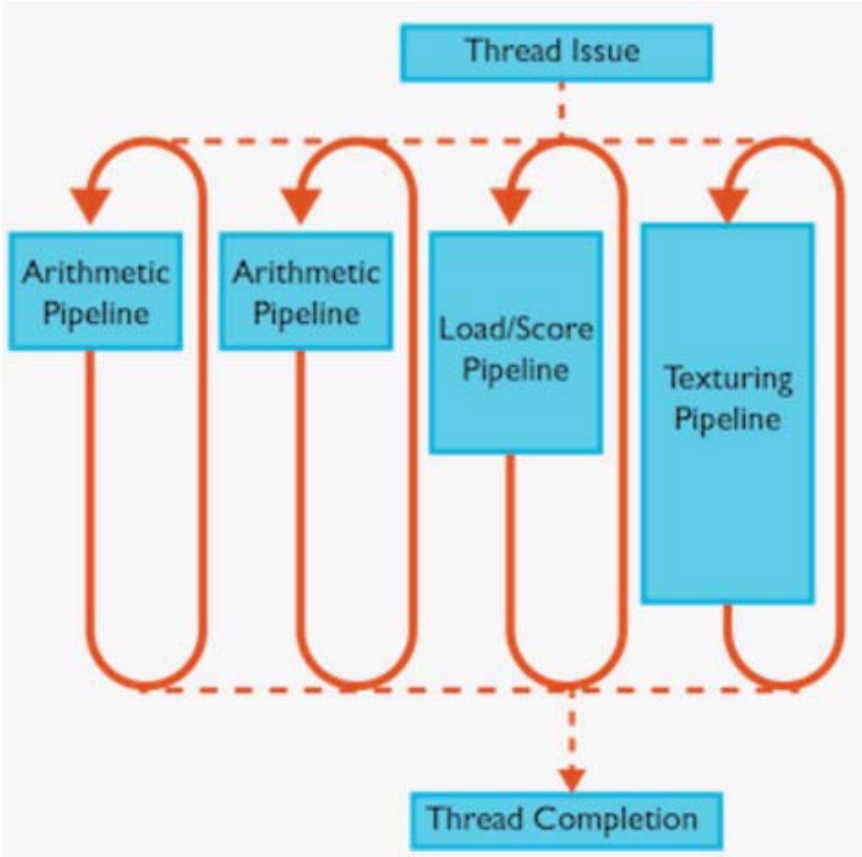
"a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations,"



See Ryan Smith, ARM's Mali Midgard Architecture Explored, http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.



See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

wherein said shading operations comprise both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations	The LG Products include shading operations comprising of both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations. In particular, the shading operations comprise texture operations. For example, the SC includes a texture pipeline, load/store pipeline and a plurality of arithmetic pipelines. The "texture pipeline (T-pipe) is responsible for all memory access to do with textures. The texture pipeline can return one bilinear filtered texel per clock; trilinear filtering requires us to load samples from two different mipmaps in memory." *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.  *See* http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.

7.2.2 About Mali GPU architectures

Mali GPUs use a SIMD architecture. Instructions operate on multiple data elements simultaneously.

The peak throughput depends on the hardware implementation of the Mali GPU type and configuration.

The Mali GPUs contain 1 to 16 identical shader cores. Each shader core supports up to 384 concurrently executing threads.

Each shader core contains:

- One to four arithmetic pipelines.
- One load-store pipeline.
- One texture pipeline.

See "ARM Guide to Unity" Version 2.1 page 7-64 available at http://infocenter.arm.com/help/topic/com.arm.doc.100140_0201_00_en/arm_guide_to_unity_enhancing_your_mobile_games_100140_0201_00_en.pdf (accessed 10/27/2016).

The arithmetic pipeline ("ALU") performs texture operations. For example, the "ALU pipeline can read/write to 32 128-bit registers" including "texture pipeline results" from the texture pipe.

Registers

The ALU pipeline can read/write to 32 128-bit registers, which can be divided into 4 32-bit (highp in GLSL) components (one vec4) or 8 16-bit (mediump) components (two vec4's). Some of the registers, however, are dedicated to special purposes (see below) and are read-only or write-only.

Special Registers

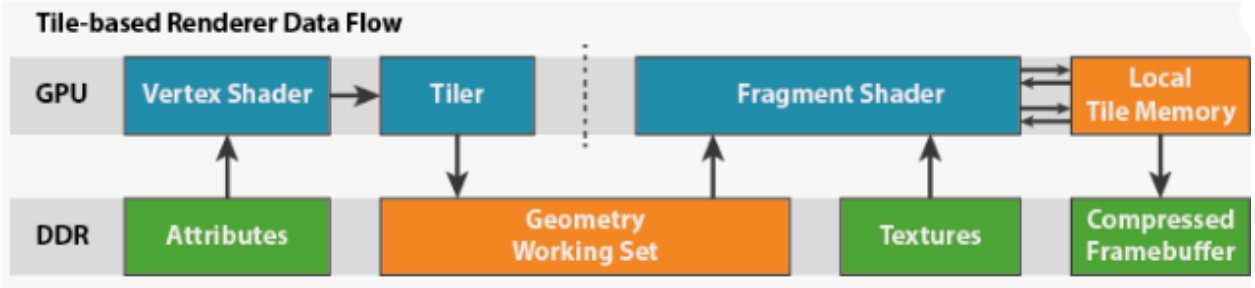
```
r24 - can mean "unused" for 1-src instructions, or a pipeline register
r26 - inline constant
r27 - load/store offset when used as output register
r28-r29 - texture pipeline results
r31.w - conditional select input when written to in scalar add ALU
```

r0 - r23 is divided into two spaces: work registers and uniform registers. A configurable number of registers can be devoted to each; if there are N uniform registers, then r0 - r(23-N) are work registers and r(24-N)-r23 are uniform registers.

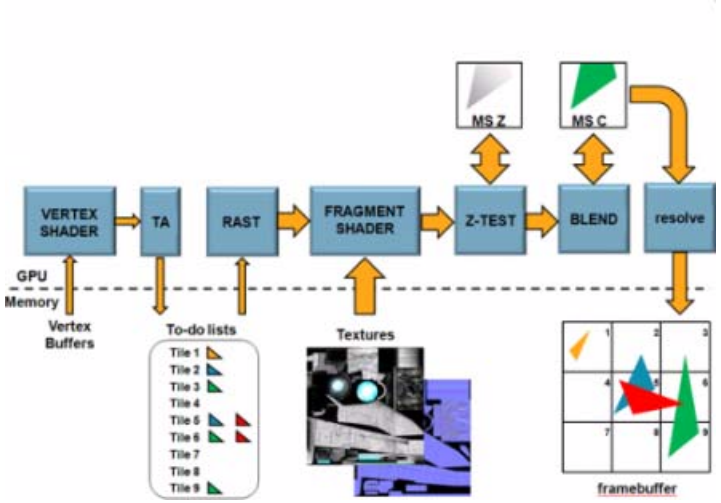
See http://limadriver.org/T6xx+ISA/.

The ALU also performs color operations. For example, the "Mali [GPU] only has to write the color data for a

single tile back to memory at the end of the tile." *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/02/20/the-mali-gpu-an-abstract-machine-part-2.

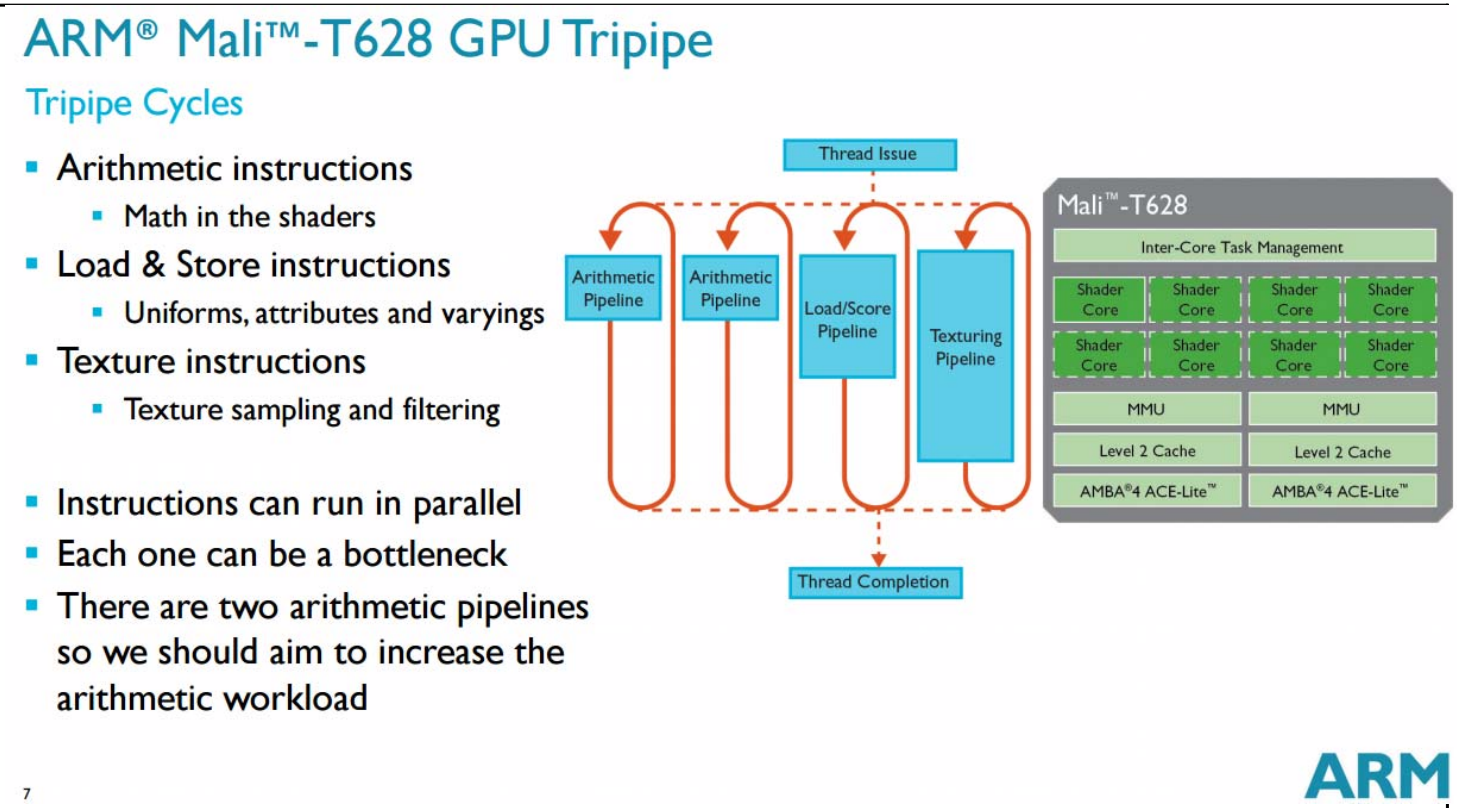


See https://community.arm.com/groups/arm-mali-graphics/blog/2014/02/20/the-mali-gpu-an-abstract-machine-part-2.



See https://community.arm.com/groups/arm-mali-graphics/blog/2012/08/17/how-low-can-you-go-building-low-power-low-bandwidth-arm-mali-gpus.

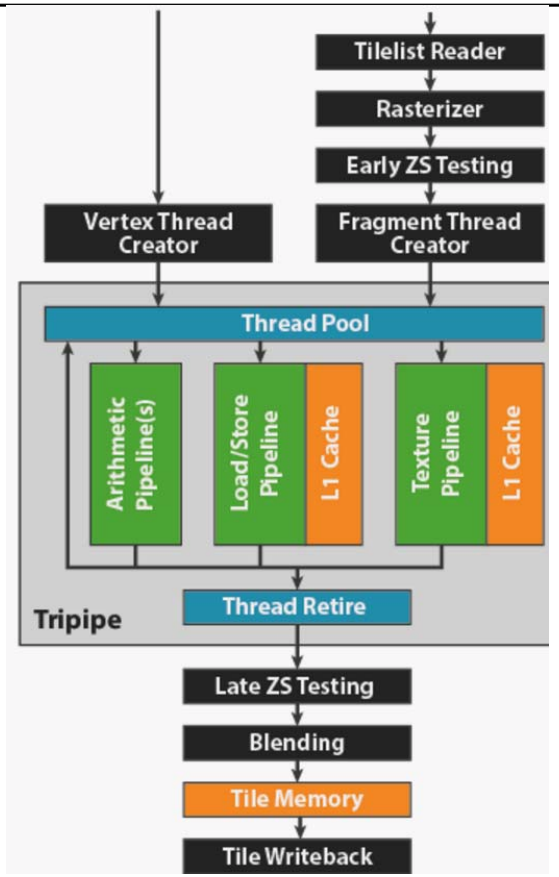
Moreover, the ALU is responsible for performing the "[m]ath in the shaders[.]"



See http://malideveloper.arm.com/downloads/GDC14/Weds/11.15amStreamlineMaliHWCounters.pdf.

Additionally, "there are three classes of execution pipeline in the tripipe design: one handling arithmetic operations, one handling memory load/store and varying access, and one handling texture access. There is one load/store and one texture pipe per shader core, but the number of arithmetic pipelines can vary depending on which GPU you are using; most silicon shipping today will have two arithmetic pipelines." *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

See http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/2-Mali-GPU-architecture-overview-and-tile-local-storage.pdf.



See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

1.1 About Mali GPUs

ARM produces the following families of Mali GPUs:

Mali Midgard GPUs

Mali Midgard GPUs include the following:

- Mali-T600 series.
- Mali-T720.
- Mali-T760.
- Mali-T820.
- Mali-T830.
- Mali-T860.
- Mali-T880.

Mali Utgard GPUs

The Mali Utgard GPUs include the following:

- Mali-300.
- Mali-400 MP.
- Mali-450 MP.
- Mali-470 MP.

Note

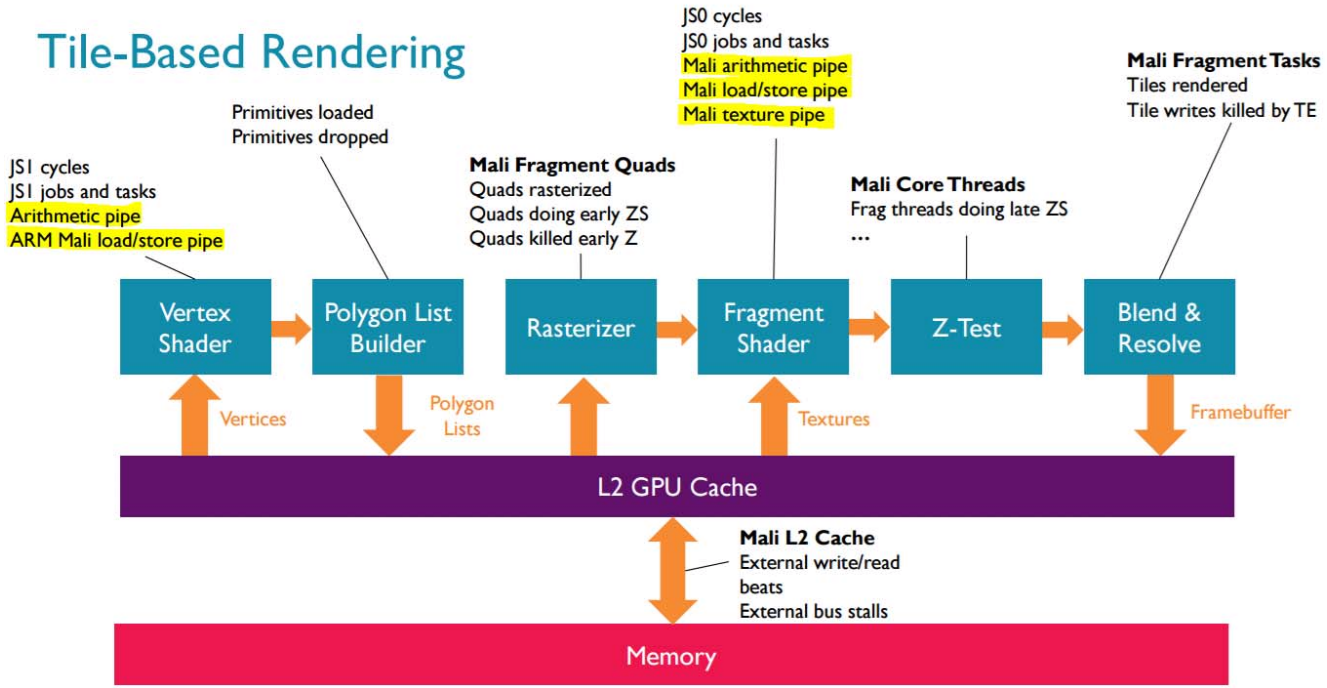
The Mali Utgard GPUs do not support OpenCL.

Mali GPUs can have one or more shader cores. Each shader core contains one or more *Arithmetic Logic Units* (ALUs).

The ALUs are based on a *Single Instruction Multiple Data* (SIMD) architecture. Instructions operate on multiple data elements simultaneously.

Mali GPUs run data processing tasks in parallel that contain relatively little control code. Mali GPUs typically contain many more processing units than application processors. This enables Mali GPUs to compute at a higher rate than application processors, without using more power.

See "ARM Guide to Unity" Version 2.1 page 1.1 available at http://infocenter.arm.com/help/topic/com.arm.doc.100140_0201_00_en/arm_guide_to_unity_enhancing_your_mobile_games_100140_0201_00_en.pdf (accessed 10/27/2016).

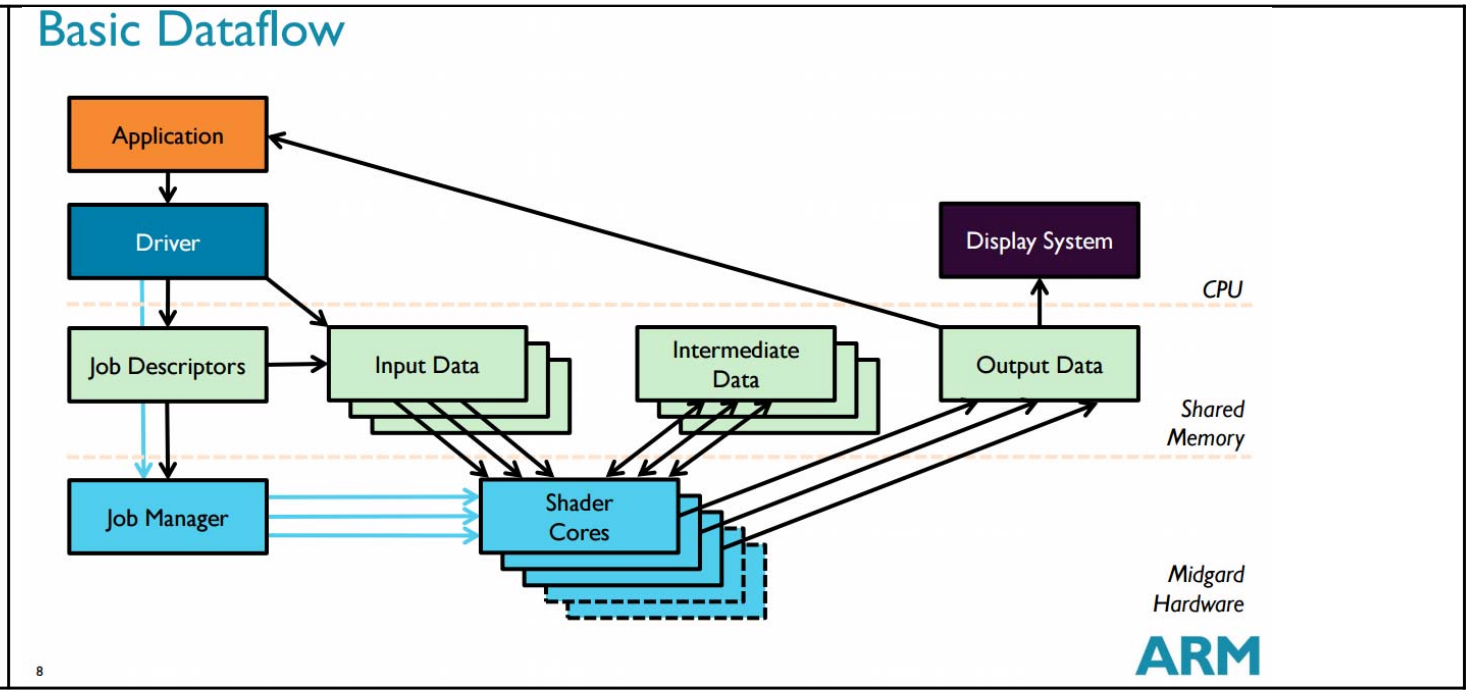


See ARM, How to Optimize Your Mobile Game with ARM Tools and Practical Examples, p.33, http://malideveloper.arm.com/downloads/GDC15/How%20to%20Optimize%20Your%20Mobile%20Game%20with%20ARM%20Tools%20and%20Practical%20Examples.pdf.

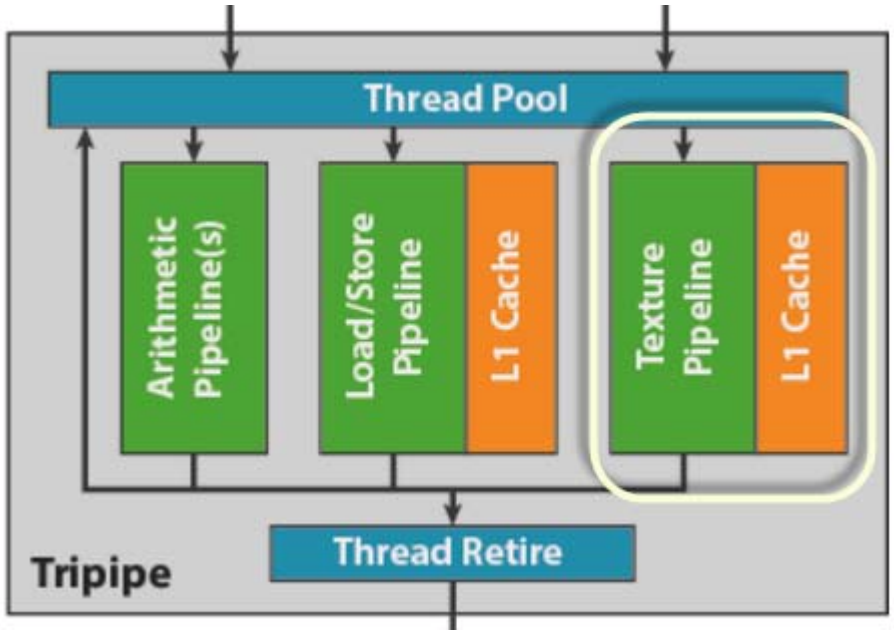
Furthermore, the unified shader comprises at least one ALU/memory pair. For example, as depicted below, each shader core is paired up with shared memory.

U.S. Patent No. 7,796,133: Claim 1

“wherein said shading operations comprise both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations”



"wherein texture operations comprise at least one of: issuing a texture request to a texture unit and writing received texture values to the memory and. "

wherein texture operations comprise at least one of: issuing a texture request to a texture unit and writing received texture values to the memory and.	The texture operations comprise at least one of: issuing a texture request to a texture unit and writing received texture values to the memory. For example, as depicted below, the "thread pool" issues texture packets to the "texture pipeline." Moreover, the "texture pipeline (T-pipe) is responsible for all memory access to do with textures. The texture pipeline can return one bilinear filtered texel per clock; trilinear filtering requires us to load samples from two different mipmaps in memory, so requires a second clock cycle to complete." *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.  *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

"wherein texture operations comprise at least one of: issuing a texture request to a texture unit and writing received texture values to the memory and. "

Texture Pipeline

The texture pipeline (T-pipe) is responsible for all memory access to do with textures. The texture pipeline can return one bilinear filtered texel per clock; trilinear filtering requires us to load samples from two different mipmaps in memory, so requires a second clock cycle to complete.

See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core

The Texture pipeline

Texture accesses use cycles in the Texture pipeline and use memory bandwidth. Using large textures can be detrimental because cache misses are more likely and this can cause multiple threads to stall while waiting for data.

To improve the performance of the Texture pipeline try the following:

Use mipmaps

Mipmaps increase the cache hit rate because it selects the best resolution of the texture to use based on the variation of texture coordinates.

Use texture compression

This is also good for reducing the memory bandwidth and increasing the cache hit rate. Each compressed block contains more than one texel, so accessing it makes it more cacheable.

Avoid trilinear or anisotropic filtering

Trilinear and anisotropic filtering increase the number of operations required to fetch texels. Avoid using these techniques unless you absolutely require them.

See "ARM Guide to Unity" Version 2.1 page 1-6 available at http://infocenter.arm.com/help/topic/com.arm.doc.100140_0201_00_en/arm_guide_to_unity_enhancing_your_mobile_games_100140_0201_00_en.pdf (accessed 10/27/2016)

“wherein texture operations comprise at least one of: issuing a texture request to a texture unit and writing received texture values to the memory and. “

Images and Compute

Another way to update textures

- Compute shaders mandate image load/store operations
  - These have been optional in other shader stages
- Allow random read/write access to a texture bound as an *image sampler*
  - Use `image*D` as shader sampler type
- Layer parameters control whether a single image, or an entire level is made accessible
  - Think texture array or 3D textures

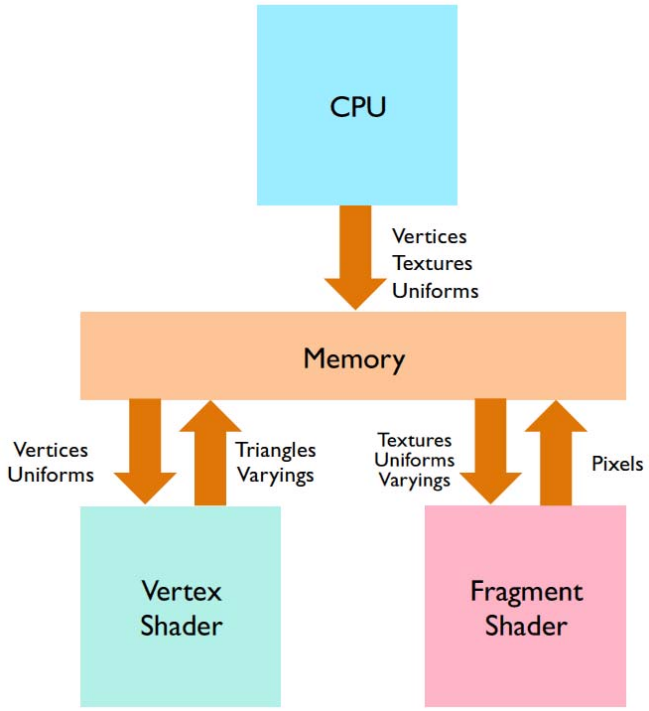
```
// Setup
glGenTextures( … );
glBindTexture( GL_TEXTURE_2D, texId );
glTextureStorage2D( GL_TEXTURE_2D, levels,
   format, width, height );
glBindImageTexture( unit, texId, layered,
   layer, GL_READ_WRITE, GL_RGBA32F );

// Update
glUseProgram( compute );
glDispatchCompute( … );
glMemoryBarrier( GL_SHADER_STROAGE_BARRIER_BIT );

// Use
glUseProgram( render );
glDrawArrays( … );
```

17

ARM

See http://www.gdcvault.com/play/1020140/Getting-the-Most-Out-of.

“wherein texture operations comprise at least one of: issuing a texture request to a texture unit and writing received texture values to the memory and. “



See ARM, ARM Tools Part 2, Best Optimization Practices for Mobile Platforms, p.13, *available at* http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/6%20-%20ARM%20Tools%20Part%202-%20Best%20Optimization%20Practices%20for%20Mobile%20Platforms.pdf.

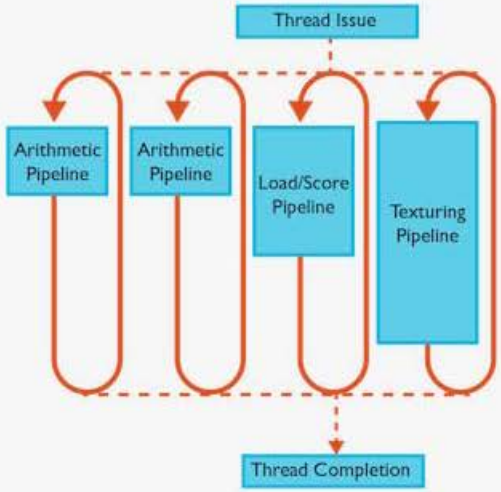
Furthermore, the received texture values are written into memory.

“wherein texture operations comprise at least one of: issuing a texture request to a texture unit and writing received texture values to the memory and. “

	Registers
	
	The ALU pipeline can read/write to 32 128-bit registers, which can be divided into 4 32-bit (highp in GLSL) components (one vec4) or 8 16-bit (mediump) components (two vec4's). Some of the registers, however, are dedicated to special purposes (see below) and are read-only or write-only.
	
	Special Registers
	
	`r24 - can mean "unused" for 1-src instructions, or a pipeline register`
	`r26 - inline constant`
	`r27 - load/store offset when used as output register`
	`r28-r29 - texture pipeline results`
	`r31.w - conditional select input when written to in scalar add ALU`
	
	r0 - r23 is divided into two spaces: work registers and uniform registers. A configurable number of registers can be devoted to each; if there are N uniform registers, then r0 - r(23-N) are work registers and r(24-N)-r23 are uniform registers.
	See http://limadriver.org/T6xx+ISA/.

“wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and”

wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and	The LG Products include at least one ALU that is operative to read from and write to the memory to perform both texture and color operations. For example, the ALU is designed to “strike a closer balance between shading and texturing.”



As we've stated before, for our purposes we're primarily looking at the Mali-T760. On the T760 ARM uses 2 ALU blocks per tri pipe, which is the most common configuration that you will see for Midgard. However ARM also has Midgard designs that have 1 ALU block or 4 ALU blocks per tri pipe, which is one of the reasons why seemingly similarly GPUs such as T760, T720, and T678 can look so similar and yet behave so differently.

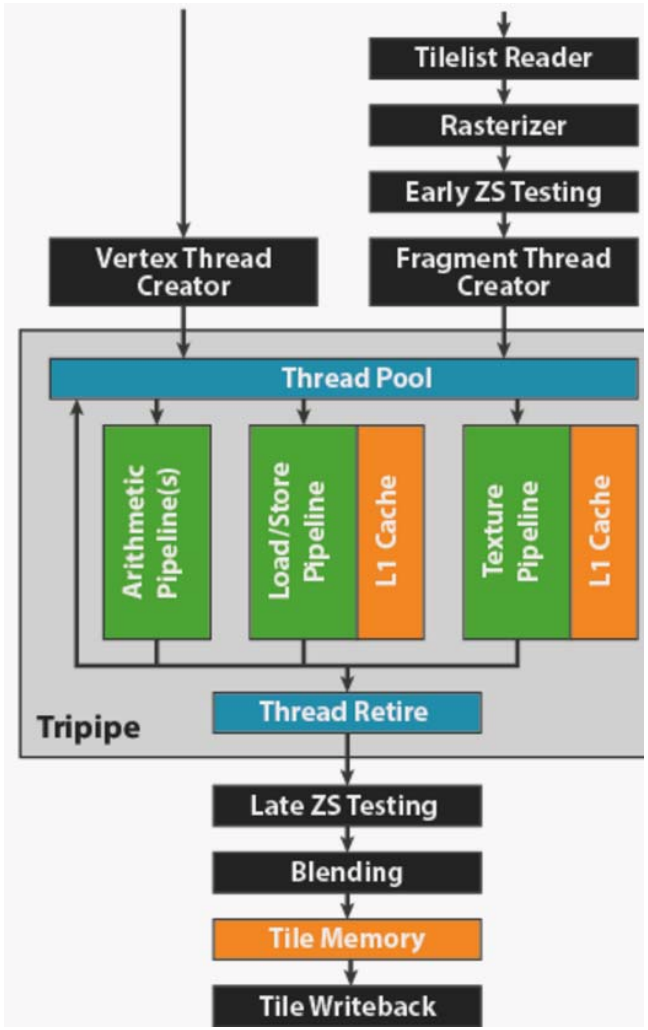
ARM Mali Midgard Arithmetic Pipeline Count (Per Core)	
T628	2
T678	4
T720	1
T760	2

Without being fully exhaustive, among various Midgard designs T628 and T760 are 2 ALU designs, while T720 is a 1 ALU design, and T678 is a 4 ALU design.

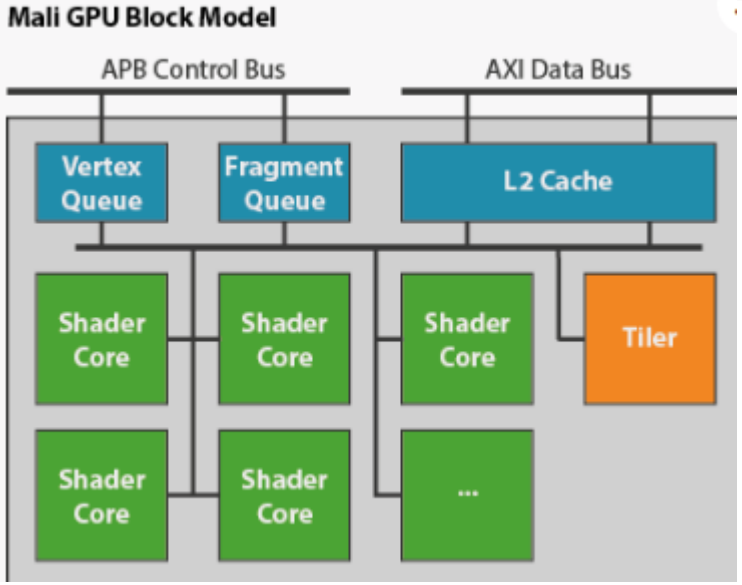
As one would expect, the different number of arithmetic pipelines per tri pipe has a knock-on effect on performance in all aspects, due to the changing ratio between the number of arithmetic pipelines and the number of load/store units and texture units. T678, for example, would be fairly shader-heavy, whereas the 2 ALU designs strike a closer balance between shading and texturing. Among the various Midgard designs ARM has experimented with several configurations, and with the T700 series they have settled on 2 ALU designs for the high-end T760 and 1 ALU for the mid-range T720 (although ARM likes to point out that T720 has some further optimizations just for this 1 ALU configuration).

See http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.

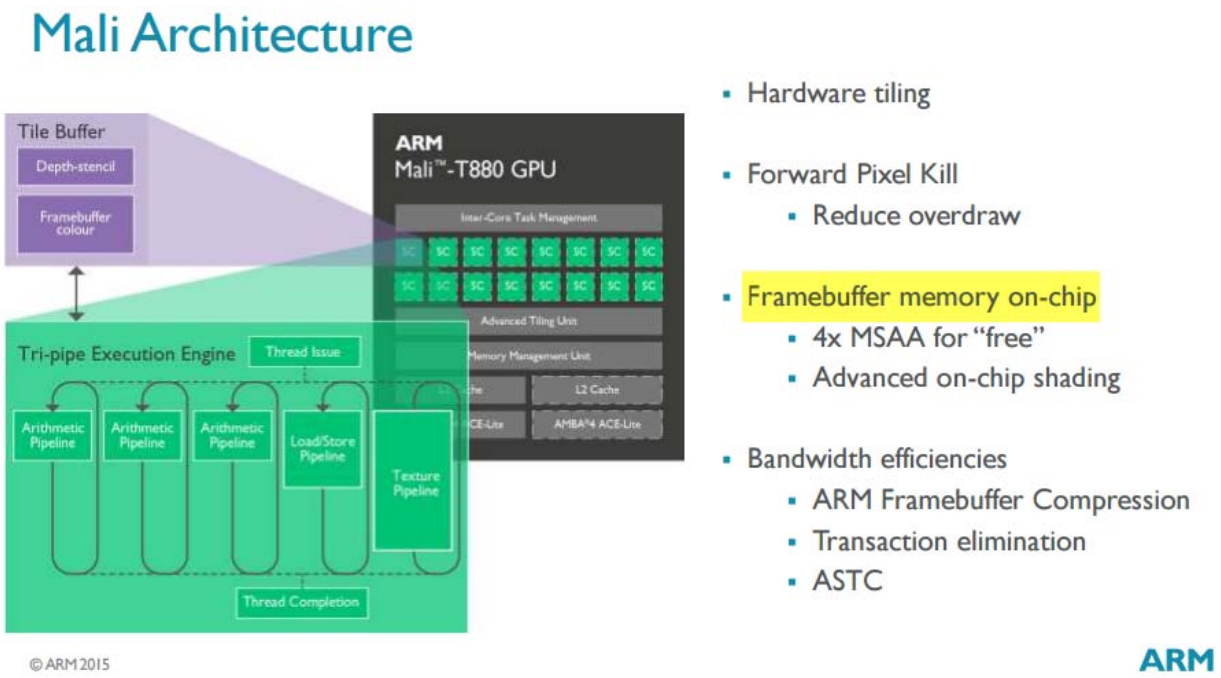
"wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and"

	Further, the ALUs are operative to read from and write to memory to perform texture and color operations as shown below.  *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

"wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and"



See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.



See "Arm Mali GPU Architecture," a presentation by Sam Martin, ARM Graphics Architect at

“wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and”

http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/LondonDec15/presentations/Mali_GPU_Architecture.pdf (accessed on 10/27/16).

Registers

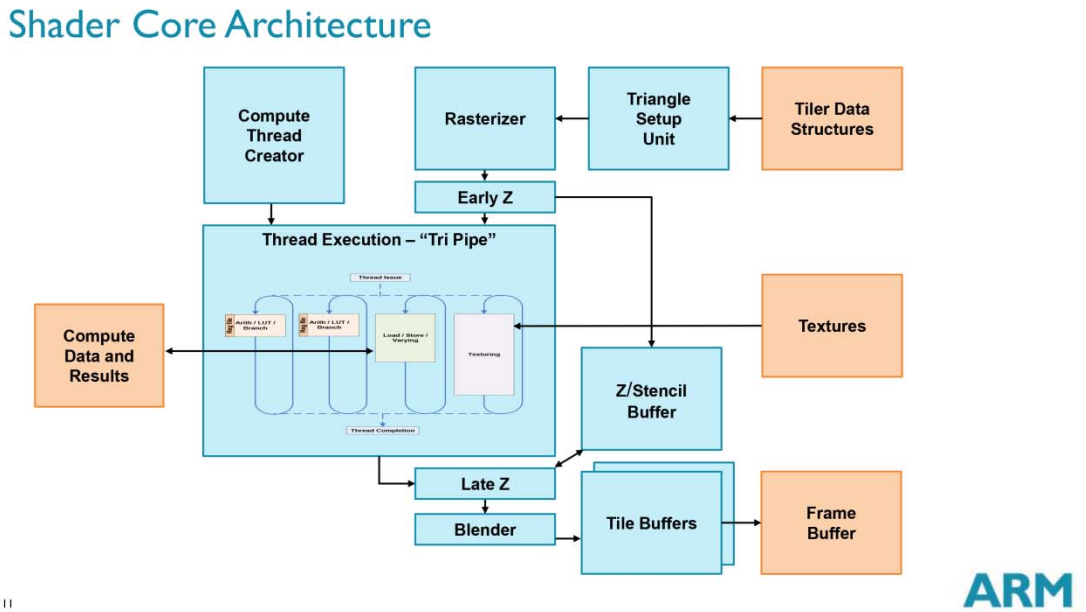
The ALU pipeline can read/write to 32 128-bit registers, which can be divided into 4 32-bit (highp in GLSL) components (one vec4) or 8 16-bit (mediump) components (two vec4's). Some of the registers, however, are dedicated to special purposes (see below) and are read-only or write-only.

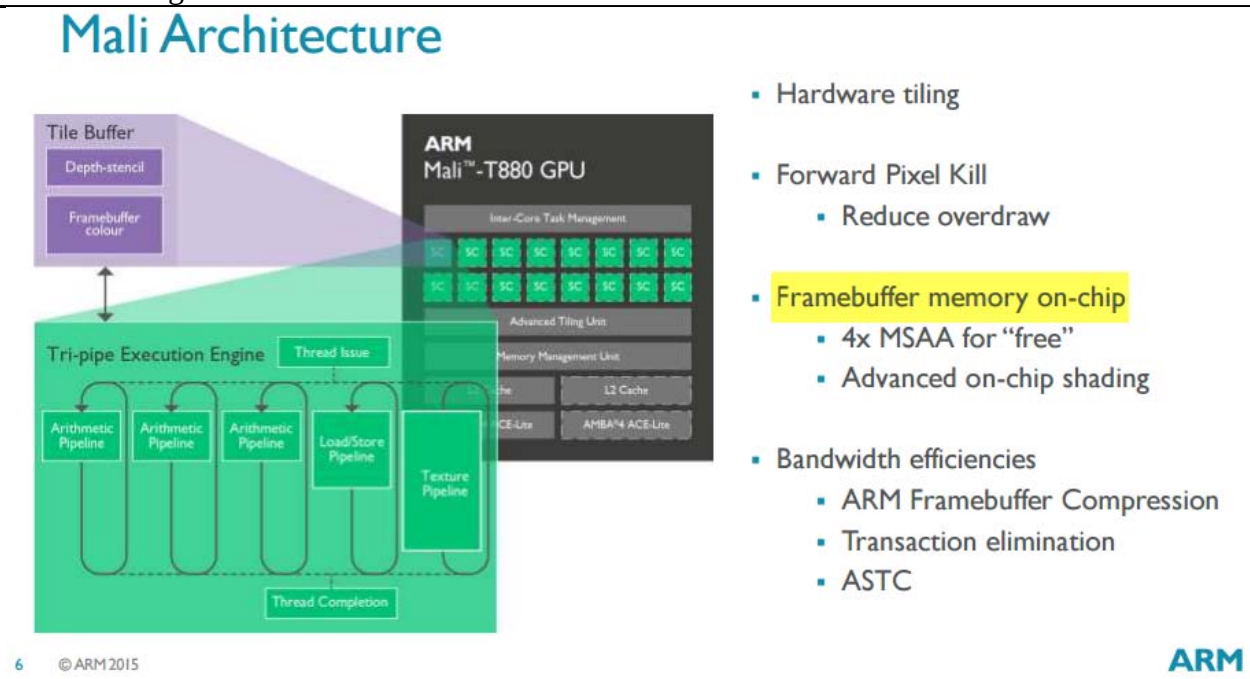
Special Registers

```
r24 - can mean "unused" for 1-src instructions, or a pipeline register
r26 - inline constant
r27 - load/store offset when used as output register
r28-r29 - texture pipeline results
r31.w - conditional select input when written to in scalar add ALU
```

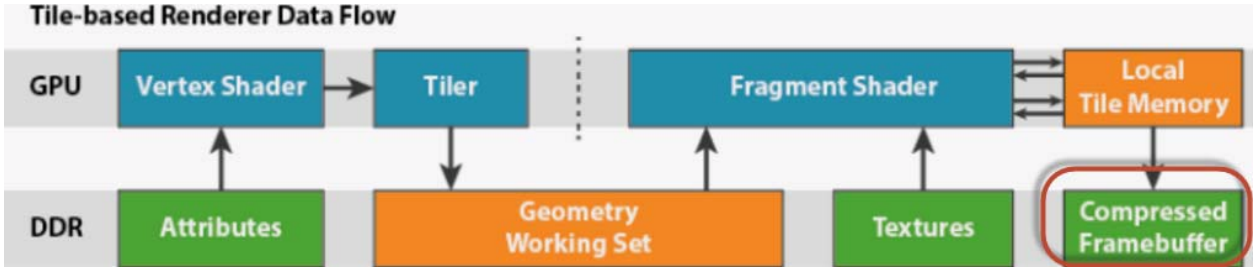
r0 - r23 is divided into two spaces: work registers and uniform registers. A configurable number of registers can be devoted to each; if there are N uniform registers, then r0 - r(23-N) are work registers and r(24-N)-r23 are uniform registers.

See http://limadriver.org/T6xx+ISA/.

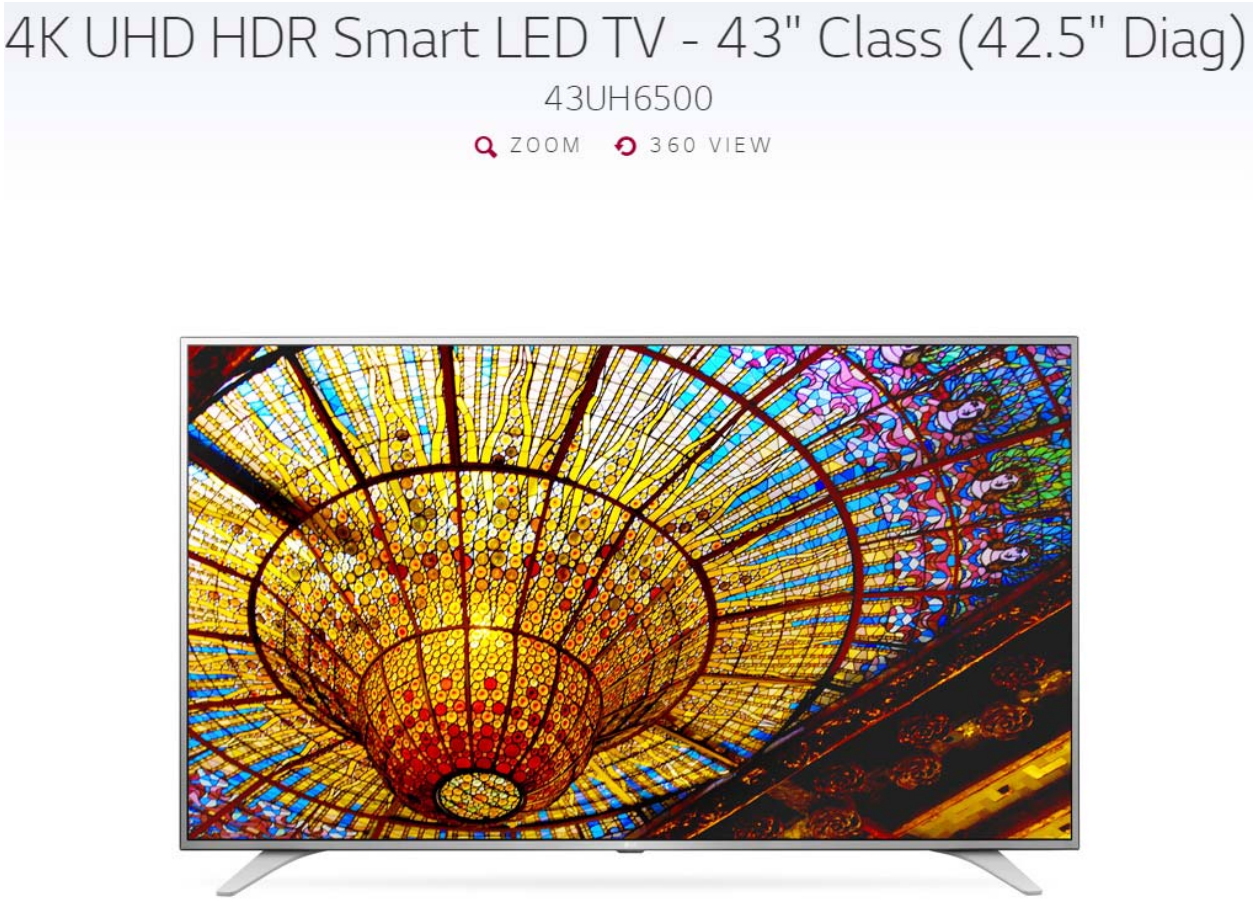
an output interface configured to send said resultant value to a frame buffer.	The LG Products include an output interface configured to send said resultant values to a frame buffer. For example the LG Product includes an internal and external frame buffer. Moreover, the Mali GPU includes an output interface for writing to the frame buffers.  *See* Ryan Smith, ARM's Mali Midgard Architecture Explored, http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.



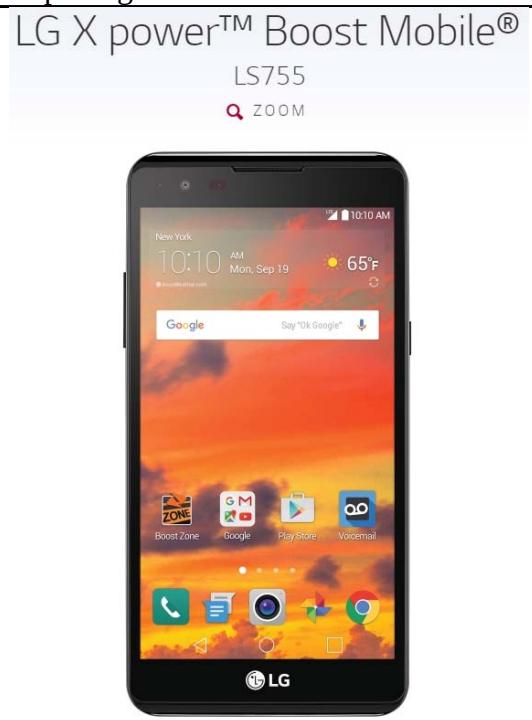
See "Arm Mali GPU Architecture," a presentation by Sam Martin, ARM Graphics Architect at http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/LondonDec15/presentations/Mali_GPU_Architecture.pdf (accessed on 10/27/16).



See http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.

40. A device comprising:	The LG 43UH6500 television and X Power LS755 phone (collectively, the "LG Products") include a device.  *See* http://www.lg.com/us/tvs/lg-43UH6500-4k-uhd-tv.

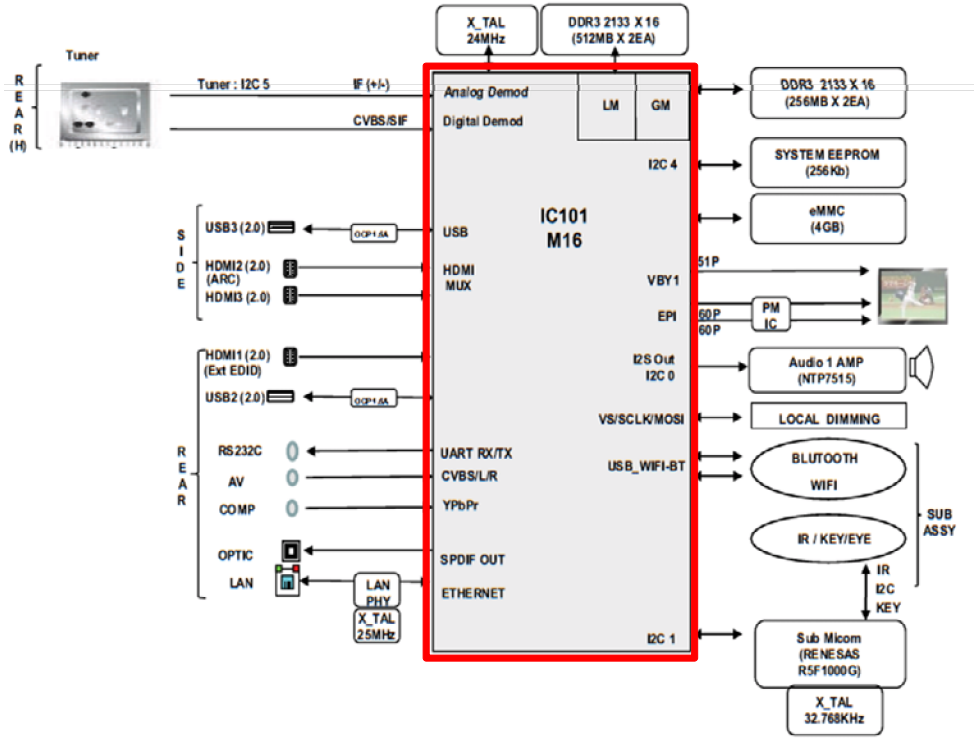
“40. A device comprising:”



See http://www.lg.com/us/cell-phones/lg-LS755-x-power-boost-mobile.

The LG Products include one of the following System-on-Chips (SoCs): M16 and MediaTek MT6755M.

"40. A device comprising:"



See LG LED TV Service Manual, Chassis: UA63J, Model: 43UH6500, p.28, *available at* https://lg.encompass.com/shop/model_research_docs/?file=/ZEN/sm/43UH6500UB.pdf.

Technical Specifications

Carrier	Boost Mobile®
Display	5.3" (1280 x 720) HD TFT Display
Battery	4,100 mAh non-removable
Platform	Android 6.0.1 Marshmallow
Processor	MediaTek 1.8 GHz Octa-Core MT6755M

See http://www.lg.com/us/cell-phones/lg-LS755-x-power-boost-mobile.

The SoCs include one of the following ARM Mali graphics processing units (the "Mali GPUs"): T760 MP2 and T860 MP2.

		M16
Smart Function	CPU	CA53 x4 1.1GHz / 1MB
	GPU	Mali T760 MP2 (650MHz)
	OSD	Separated 2K@60p
	HEVC	4K @60,10bit
	DDR	DDR3-2133/ DDR4-2400
	Audio DSP	HiFi3 Dual @370MHz

See LG LED TV Service Manual, Chassis: UA63J, Model: 43UH6500, p.123, *available at* https://lg.encompass.com/shop/model_research_docs/?file=/ZEN/sm/43UH6500UB.pdf.

MediaTek MT6755 Helio P10 Specs

Release	Q4 2015
Process	28nm
Apps CPU	8x Cortex-A53, up to 2.0GHz
GPU	ARM Mali-T860 MP2 at 700 MHz

See http://cnoemphone.com/blog/mediatek-mt6755-helio-p10-specs-benchmark-and-smartphone-list.

The Mali GPUs share substantially similar structure, function, and operation.

"40. A device comprising:"

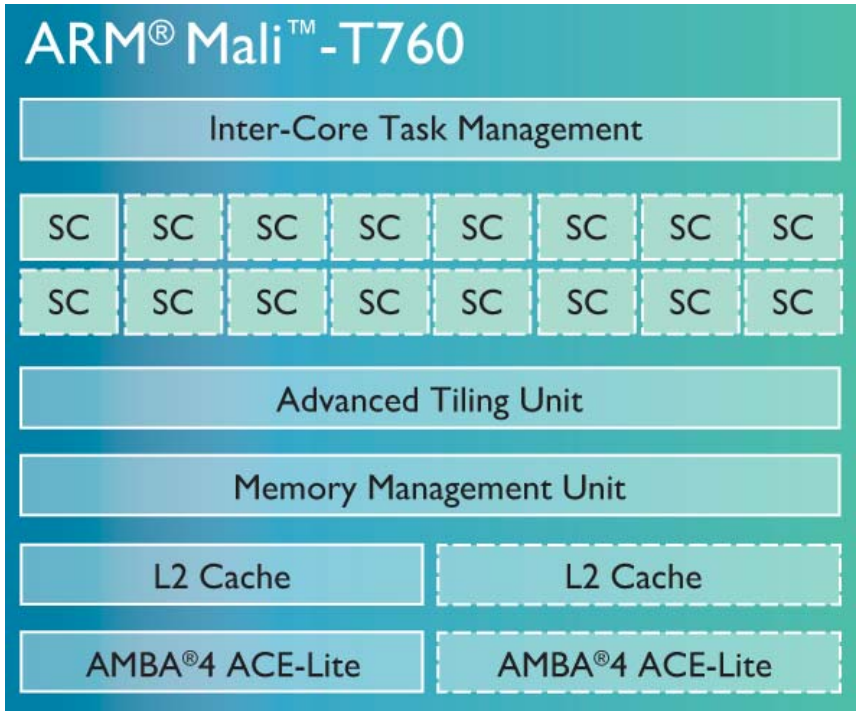
High performance

With stunning graphics capabilities, ARM Mali High Performance GPUs combine GPU Compute functionality with micro-architecture enhancements and system-wide, bandwidth-saving technology to bring energy efficiency to advanced mobile and consumer devices. GPU Compute solutions enable each task to be executed on the most suitable processor within the system. The resultant efficiencies guarantee superior graphics performance and extended battery life.

High performance GPUs:

- Mali-G71
- Mali-T860 & T880
- Mali-T760
- Mali-T628
- Mali-T624

See http://www.arm.com/products/graphics-and-multimedia/mali-gpu/

a plurality of unified shaders synchronized by a clock mechanism to process shading operations together,	The LG Products include a plurality of unified shaders synchronized by a clock mechanism to process shading operations together. For example, the Mali GPUs include a plurality of unified shaders.  *See* http://www.arm.com/products/multimedia/mali-gpu/high-performance/mali-t860-t880.php. **GPU Architecture** The "Midgard" family of Mali GPUs (the Mali-T600 and Mali-T700 series) use a unified shader core architecture, meaning that only a single type of shader core exists in the design. This single core can execute all types of programmable shader code, including vertex shaders, fragment shaders, and compute kernels. *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core. Further, the plurality of unified shaders are synchronized by a clock mechanism to process shading operations together.

For example, the GPU can issue multiple instructions in parallel for each shader core per clock cycle.

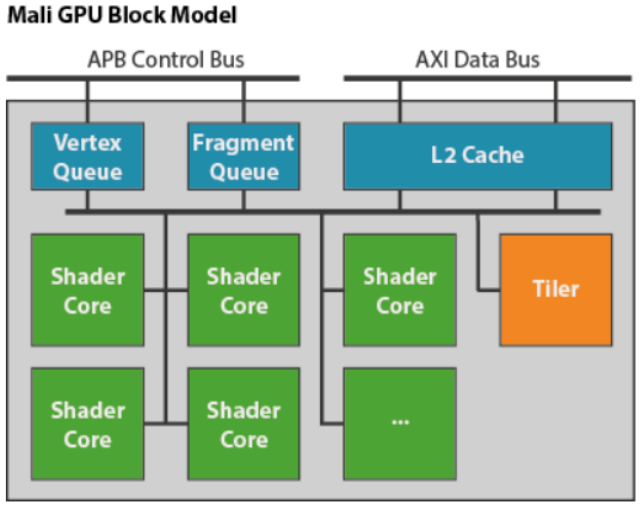
GPU Limits

Based on this simple model it is possible to outline some of the fundamental properties underpinning the GPU performance.

- The GPU can issue one vertex per shader core per clock
- The GPU can issue one fragment per shader core per clock
- The GPU can retire one pixel per shader core per clock
- We can issue one instruction per pipe per clock, so for a typical shader core we can issue four instructions in parallel if we have them available to run
  - We can achieve 17 FP32 operations per A-pipe
  - One vector load, one vector store, or one vector varying per LS-pipe
  - One bilinear filtered texel per T-pipe
- The GPU will typically have 32-bits of DDR access (read and write) per core per clock [configurable]

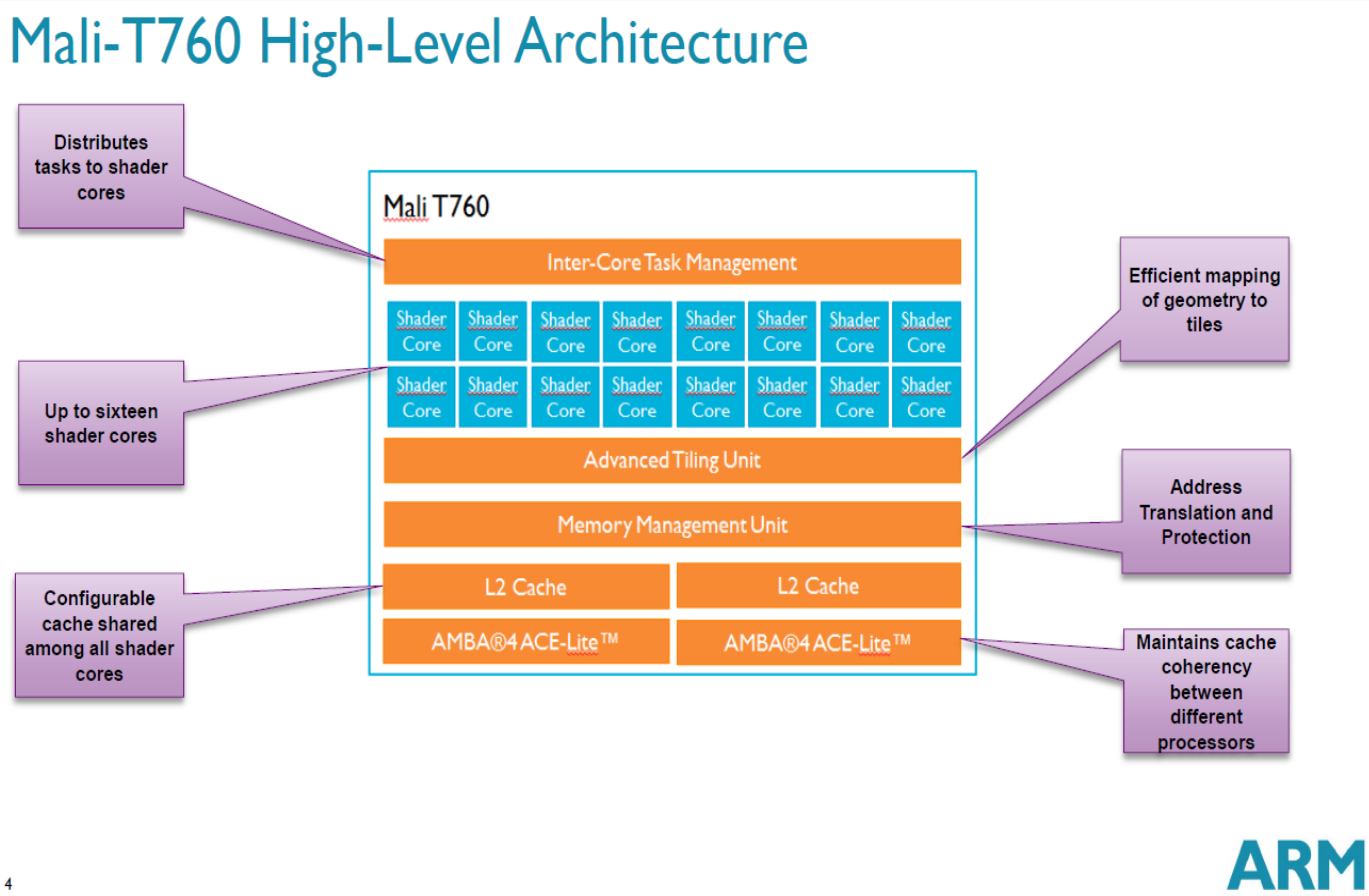
See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

Further, for example, the workloads for the Mali GPU are queued and "processed by the GPU at the same time, so vertex processing and fragment processing for different render targets can be running in parallel." Further, the "workload for a single render target is broken into smaller pieces and distributed across all of the shader cores in the GPU."



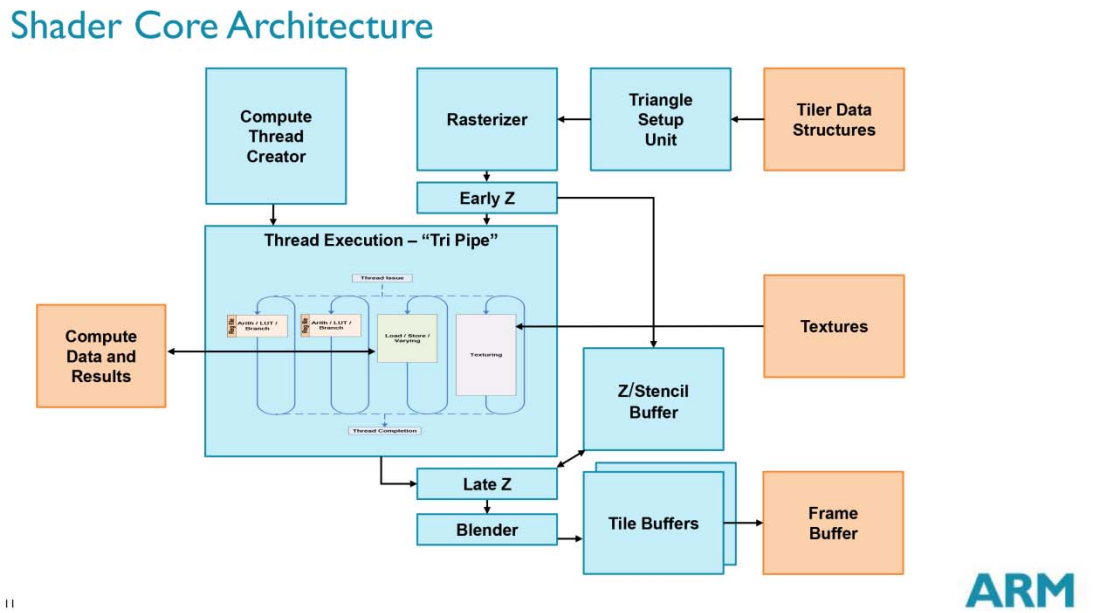
The graphics work for the GPU is queued in a pair of queues, one for vertex/tiling workloads and one for fragment workloads, with all work for one render target being submitted as a single submission into each queue. Workloads from both queues can be processed by the GPU at the same time, so vertex processing and fragment processing for different render targets can be running in parallel (see the first blog for more details on this pipelining methodology). The workload for a single render target is broken into smaller pieces and distributed across all of the shader cores in the GPU, or in the case of tiling workloads (see the second blog in this series for an overview of tiling) a fixed function tiling unit.

See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.



See http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/2-Mali-GPU-architecture-overview-and-tile-local-storage.pdf.

"wherein each of the unified shaders comprises: an input interface for receiving a packet from a rasterizer;"

wherein each of the unified shaders comprises: an input interface for receiving a packet from a rasterizer;	Each of the unified shaders in the LG Products includes an input interface for receiving a packet from a rasterizer. For example, "[t]he Mali shader core is structured as a number of fixed-function hardware blocks wrapped around a programmable tri-pipe execution core. The fixed function units perform the setup for a shader operation - such as rasterizing triangles or performing depth testing - or handling the post-shader activities - such as blending, or writing back a whole tile's worth of data at the end of rendering. The tripipe itself is the programmable part responsible for the execution of shader programs." *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.  *See* Ryan Smith, ARM's Mali Midgard Architecture Explored, http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.

"wherein each of the unified shaders comprises: an input interface for receiving a packet from a rasterizer;"

Primitive assembly

In primitive assembly the vertices are assembled into geometric primitives. The resulting primitives are clipped to a clipping volume and sent to the rasterizer.

Rasterization

Output values from the vertex shader are calculated for every generated fragment. This process is known as interpolation. During rasterization, the primitives are converted into a set of two-dimensional fragments that are then sent to the fragment shader.

Transform feedback

Transform feedback, enables writing selective writing to an output buffer that the vertex shader outputs and is later sent back to the vertex shader. This feature is not exposed by Unity but it is used internally, for example, to optimize the skinning of characters.

Fragment shader

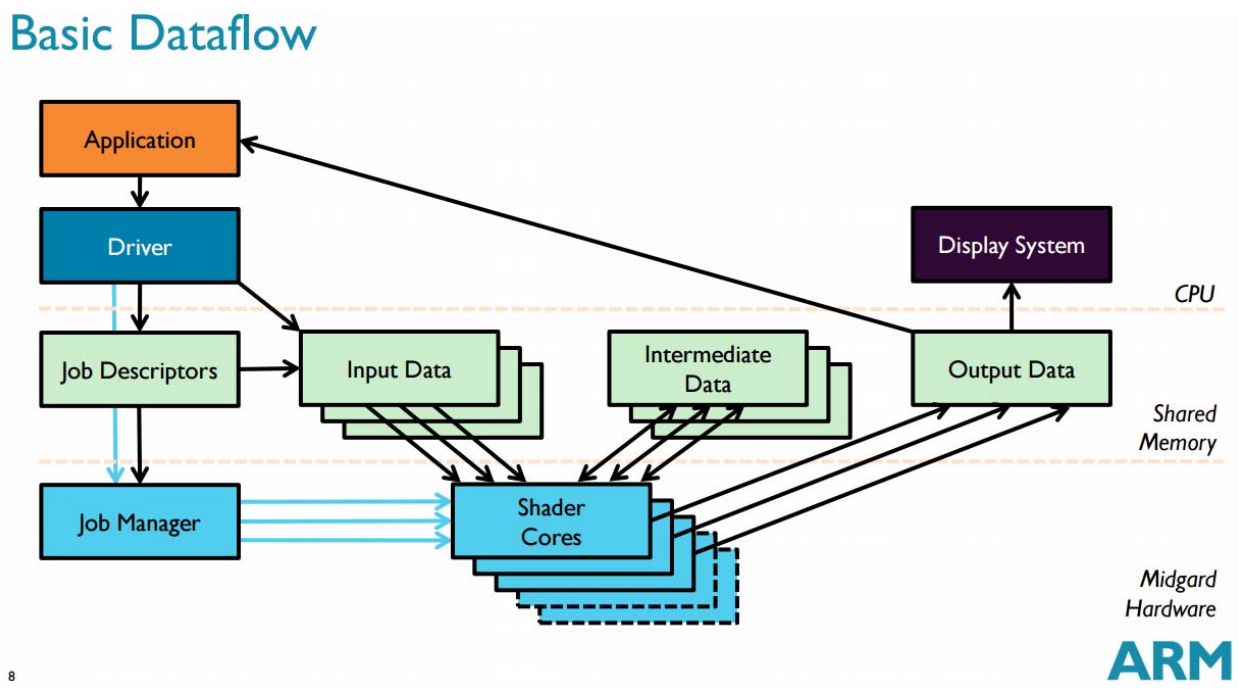
The fragment shader implements a general-purpose programmable method for operating on fragments before they are sent to the next stage.

Per-fragment operations

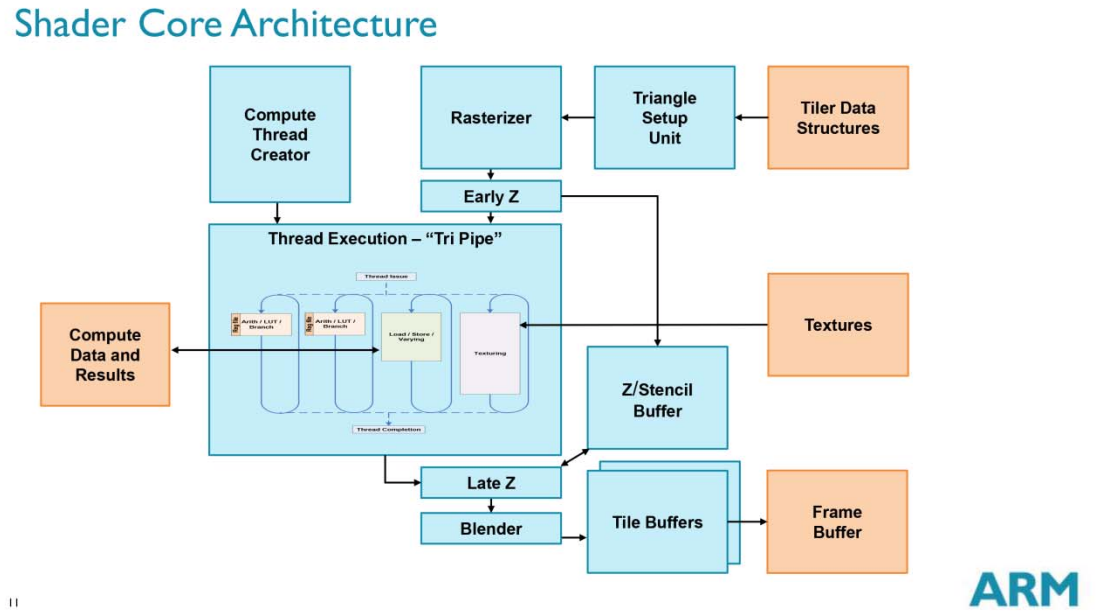
In Per-fragment operations several functions and tests are applied on each fragment: pixel ownership test, scissor test, stencil and depth tests, blending and dithering. As a result of this per-fragment stage either the fragment is discarded or the fragment color, depth or stencil value is written to the frame buffer in screen coordinates.

See "ARM Guide to Unity" Version 2.1 page 6-77 available at http://infocenter.arm.com/help/topic/com.arm.doc.100140_0201_00_en/arm_guide_to_unity_enhancing_your_mobile_games_100140_0201_00_en.pdf (accessed 10/27/2016)

"a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations,"

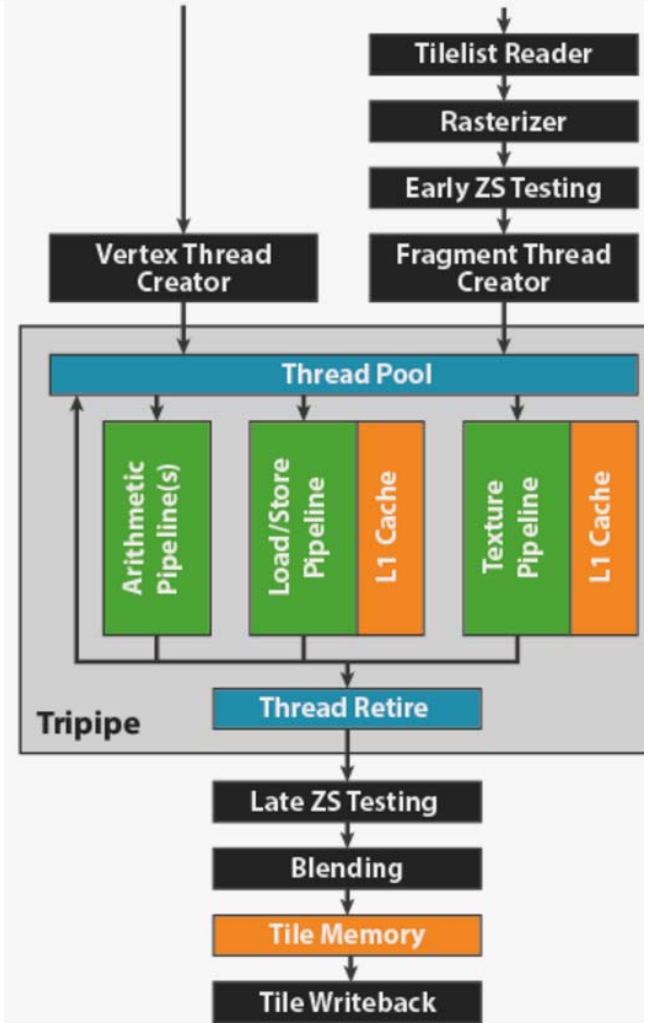
a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations,	Each of the unified shaders in the LG Products includes a shading processing mechanism configured to produce a resultant value from a rasterizer packet by performing one or more shading operations. For example, the Mali GPU has three classes of pipelines "in the tripipe design: one handling arithmetic operations, one handling memory load/store and varying access, and one handling texture access. There is one load/store and one texture pipe per shader core, but the number of arithmetic pipelines can vary." *See* http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.  *See* http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/2-Mali-GPU-architecture-overview-and-tile-local-storage.pdf.

"a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations,"

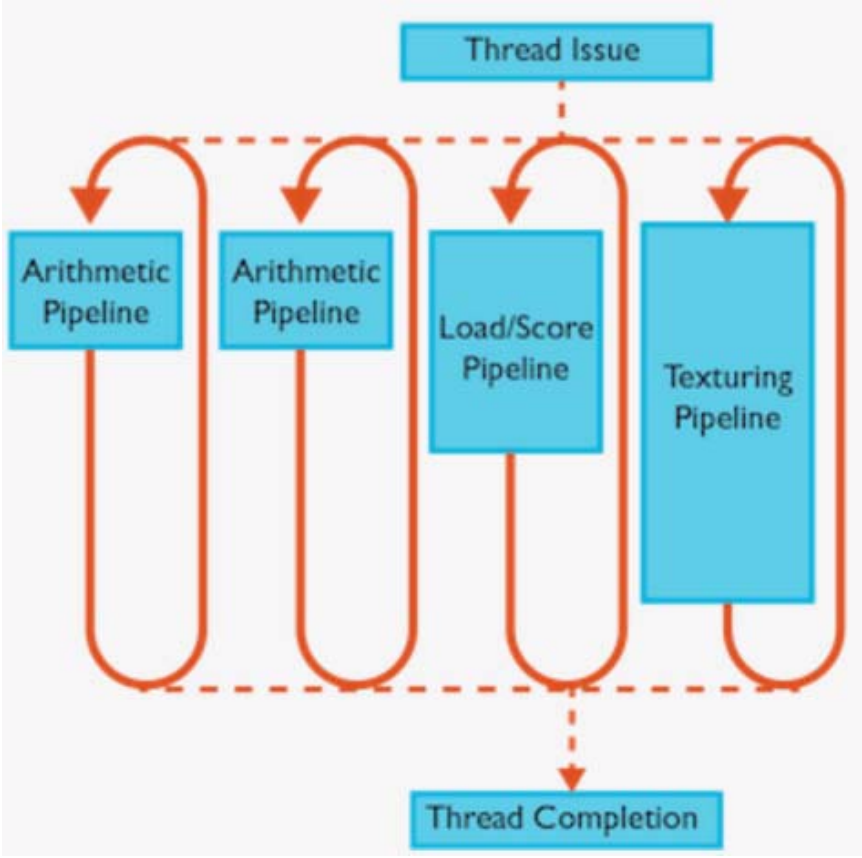


See Ryan Smith, ARM's Mali Midgard Architecture Explored, http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.

"a shading processing mechanism configured to produce a resultant value from said packet by performing one or more shading operations,"



See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

wherein said shading operations comprise both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations	The LG Products include shading operations comprising of both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations. In particular, the shading operations comprise texture operations. For example, the SC includes a texture pipeline, load/store pipeline and a plurality of arithmetic pipelines. The "texture pipeline (T-pipe) is responsible for all memory access to do with textures. The texture pipeline can return one bilinear filtered texel per clock; trilinear filtering requires us to load samples from two different mipmaps in memory." *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.  *See* http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.

7.2.2 About Mali GPU architectures

Mali GPUs use a SIMD architecture. Instructions operate on multiple data elements simultaneously.

The peak throughput depends on the hardware implementation of the Mali GPU type and configuration.

The Mali GPUs contain 1 to 16 identical shader cores. Each shader core supports up to 384 concurrently executing threads.

Each shader core contains:

- One to four arithmetic pipelines.
- One load-store pipeline.
- One texture pipeline.

See "ARM Guide to Unity" Version 2.1 page 7-64 available at http://infocenter.arm.com/help/topic/com.arm.doc.100140_0201_00_en/arm_guide_to_unity_enhancing_your_mobile_games_100140_0201_00_en.pdf (accessed 10/27/2016).

The arithmetic pipeline ("ALU") performs texture operations. For example, the "ALU pipeline can read/write to 32 128-bit registers" including "texture pipeline results" from the texture pipe.

Registers

The ALU pipeline can read/write to 32 128-bit registers, which can be divided into 4 32-bit (highp in GLSL) components (one vec4) or 8 16-bit (mediump) components (two vec4's). Some of the registers, however, are dedicated to special purposes (see below) and are read-only or write-only.

Special Registers

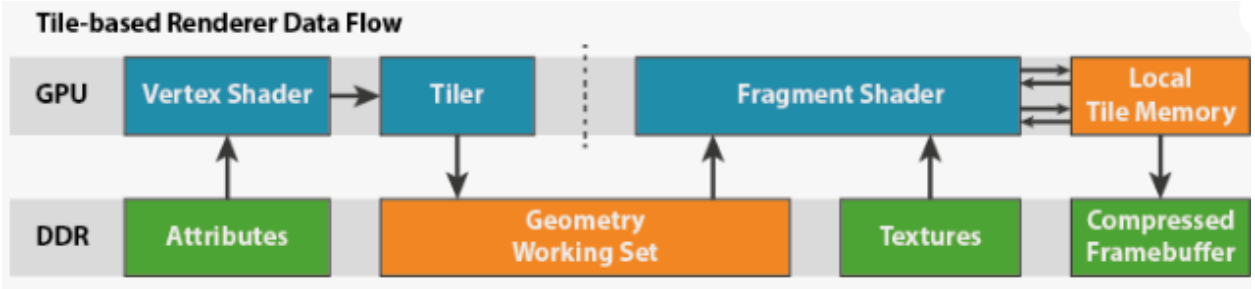
```
r24 - can mean "unused" for 1-src instructions, or a pipeline register
r26 - inline constant
r27 - load/store offset when used as output register
r28-r29 - texture pipeline results
r31.w - conditional select input when written to in scalar add ALU
```

r0 - r23 is divided into two spaces: work registers and uniform registers. A configurable number of registers can be devoted to each; if there are N uniform registers, then r0 - r(23-N) are work registers and r(24-N)-r23 are uniform registers.

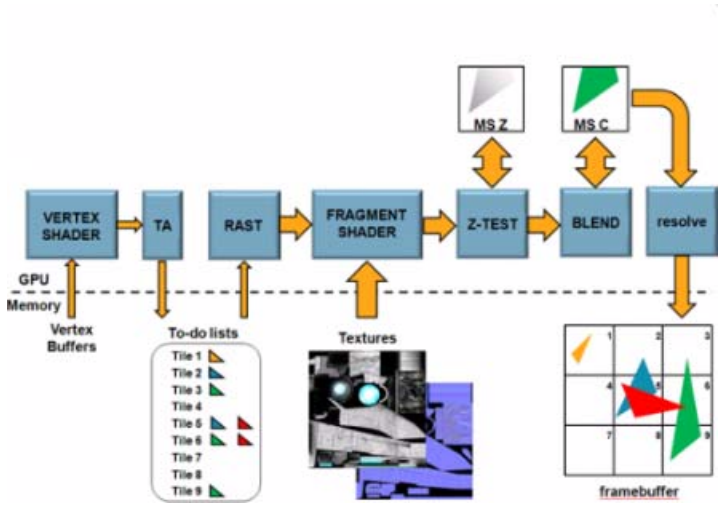
See http://limadriver.org/T6xx+ISA/.

The ALU also performs color operations. For example, the "Mali [GPU] only has to write the color data for a single tile back to memory at the end of the tile."

See https://community.arm.com/groups/arm-mali-graphics/blog/2014/02/20/the-mali-gpu-an-abstract-machine-part-2.

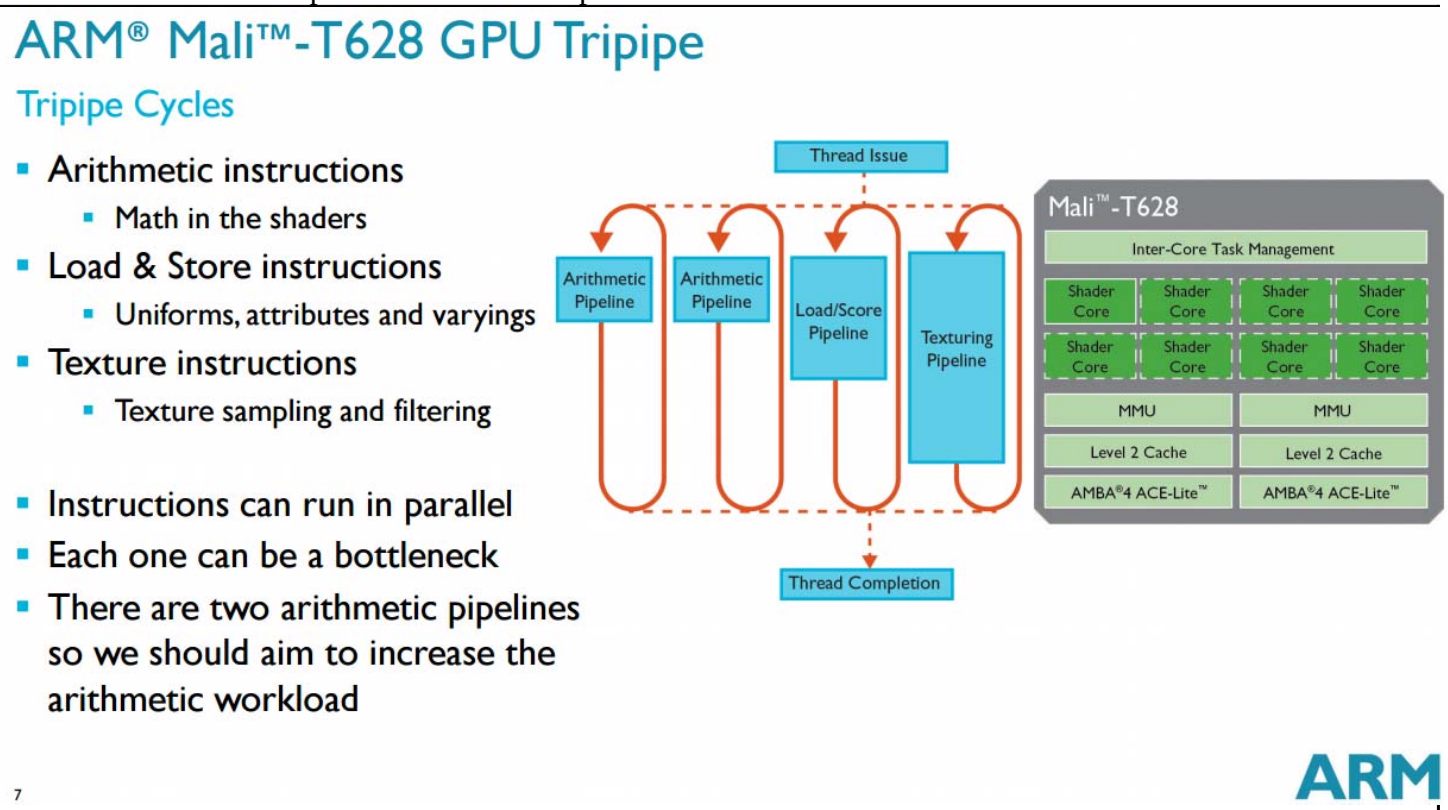


See https://community.arm.com/groups/arm-mali-graphics/blog/2014/02/20/the-mali-gpu-an-abstract-machine-part-2.



See https://community.arm.com/groups/arm-mali-graphics/blog/2012/08/17/how-low-can-you-go-building-low-power-low-bandwidth-arm-mali-gpus.

Moreover, the ALU is responsible for performing the "[m]ath in the shaders[.]"

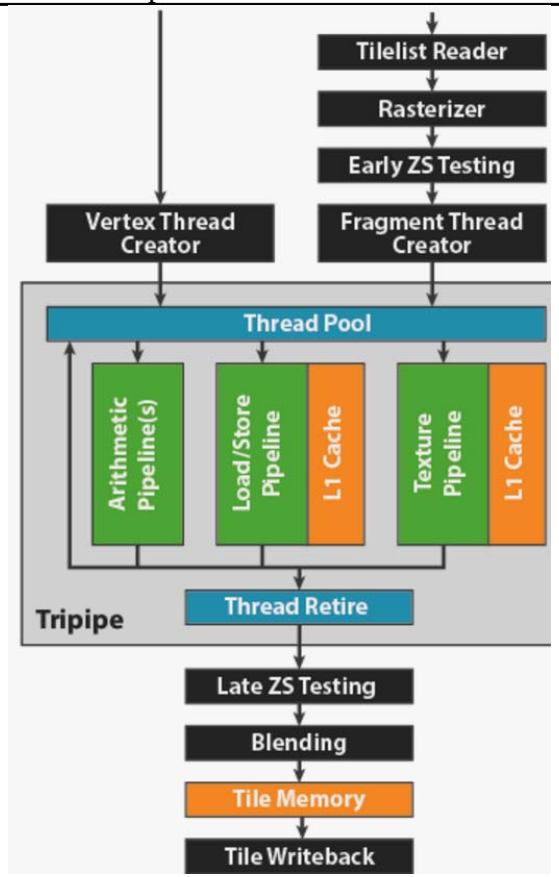


See http://malideveloper.arm.com/downloads/GDC14/Weds/11.15amStreamlineMaliHWCounters.pdf.

Additionally, "there are three classes of execution pipeline in the tripipe design: one handling arithmetic operations, one handling memory load/store and varying access, and one handling texture access. There is one load/store and one texture pipe per shader core, but the number of arithmetic pipelines can vary depending on which GPU you are using; most silicon shipping today will have two arithmetic pipelines."
See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

See http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/2-Mali-GPU-architecture-overview-and-tile-local-storage.pdf.

"wherein said shading operations comprise both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations"



See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

1.1 About Mali GPUs

ARM produces the following families of Mali GPUs:

Mali Midgard GPUs

Mali Midgard GPUs include the following:

- Mali-T600 series.
- Mali-T720.
- Mali-T760.
- Mali-T820.
- Mali-T830.
- Mali-T860.
- Mali-T880.

Mali Utgard GPUs

The Mali Utgard GPUs include the following:

- Mali-300.
- Mali-400 MP.
- Mali-450 MP.
- Mali-470 MP.

Note

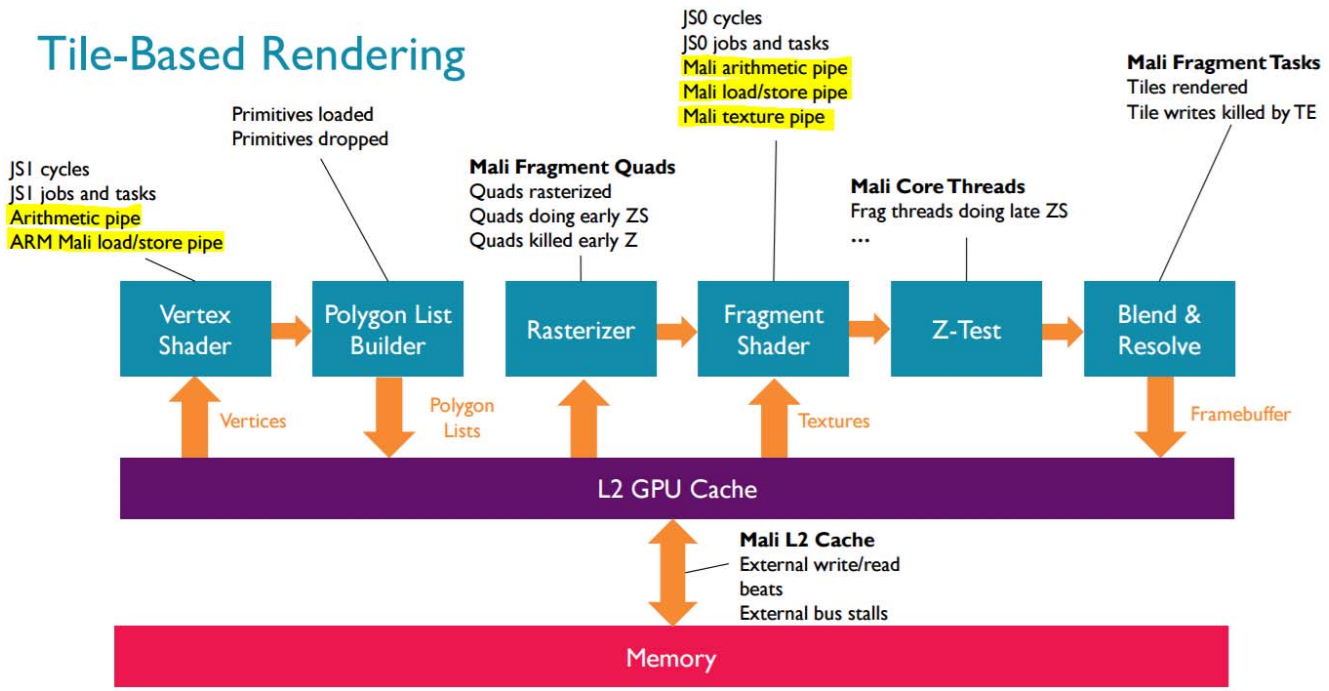
The Mali Utgard GPUs do not support OpenCL.

Mali GPUs can have one or more shader cores. Each shader core contains one or more *Arithmetic Logic Units* (ALUs).

The ALUs are based on a *Single Instruction Multiple Data* (SIMD) architecture. Instructions operate on multiple data elements simultaneously.

Mali GPUs run data processing tasks in parallel that contain relatively little control code. Mali GPUs typically contain many more processing units than application processors. This enables Mali GPUs to compute at a higher rate than application processors, without using more power.

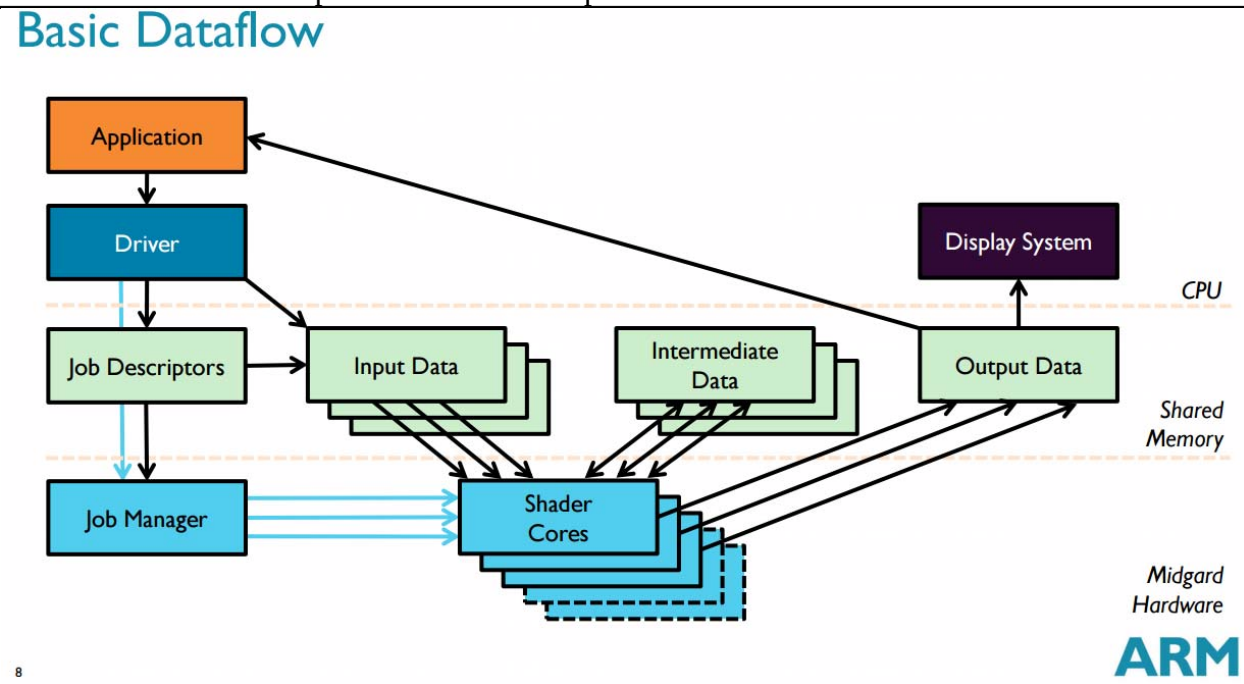
See "ARM Guide to Unity" Version 2.1 page 1.1 available at http://infocenter.arm.com/help/topic/com.arm.doc.100140_0201_00_en/arm_guide_to_unity_enhancing_your_mobile_games_100140_0201_00_en.pdf (accessed 10/27/2016).



See ARM, How to Optimize Your Mobile Game with ARM Tools and Practical Examples, p.33, http://malideveloper.arm.com/downloads/GDC15/How%20to%20Optimize%20Your%20Mobile%20Game%20with%20ARM%20Tools%20and%20Practical%20Examples.pdf.

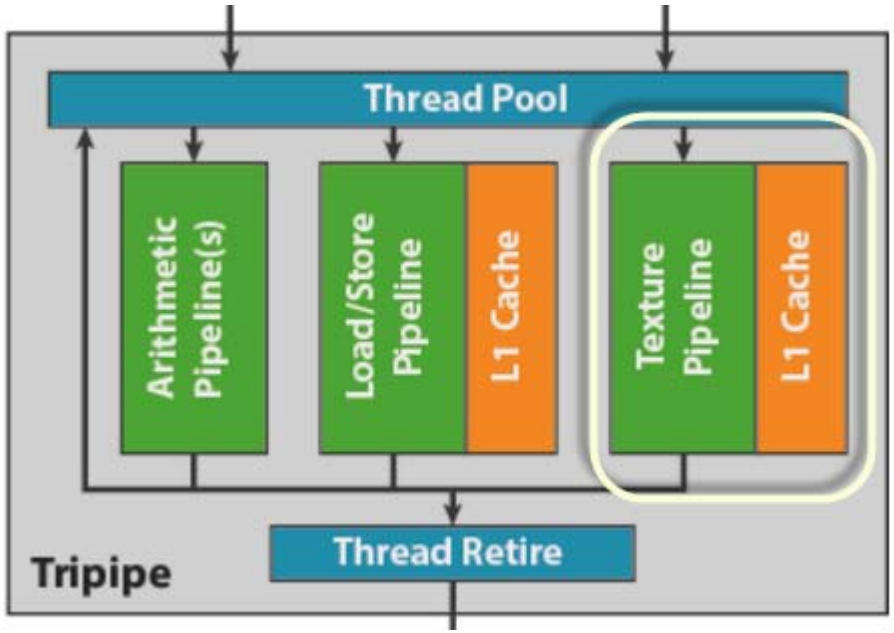
Furthermore, the unified shader comprises at least one ALU/memory pair. For example, as depicted below, each shader core is paired up with shared memory.

"wherein said shading operations comprise both texture operations and color operations and comprising at least one ALU/memory pair operative to perform both texture operations and color operations"



See http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/2-Mali-GPU-architecture-overview-and-tile-local-storage.pdf.

"wherein texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory and"

wherein texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory and	The texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory. For example, as depicted below, the "thread pool" issues texture packets to the "texture pipeline." Moreover, the "texture pipeline (T-pipe) is responsible for all memory access to do with textures. The texture pipeline can return one bilinear filtered texel per clock; trilinear filtering requires us to load samples from two different mipmaps in memory, so requires a second clock cycle to complete." *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.  *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

Texture Pipeline

The texture pipeline (T-pipe) is responsible for all memory access to do with textures. The texture pipeline can return one bilinear filtered texel per clock; trilinear filtering requires us to load samples from two different mipmaps in memory, so requires a second clock cycle to complete.

See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core

The Texture pipeline

Texture accesses use cycles in the Texture pipeline and use memory bandwidth. Using large textures can be detrimental because cache misses are more likely and this can cause multiple threads to stall while waiting for data.

To improve the performance of the Texture pipeline try the following:

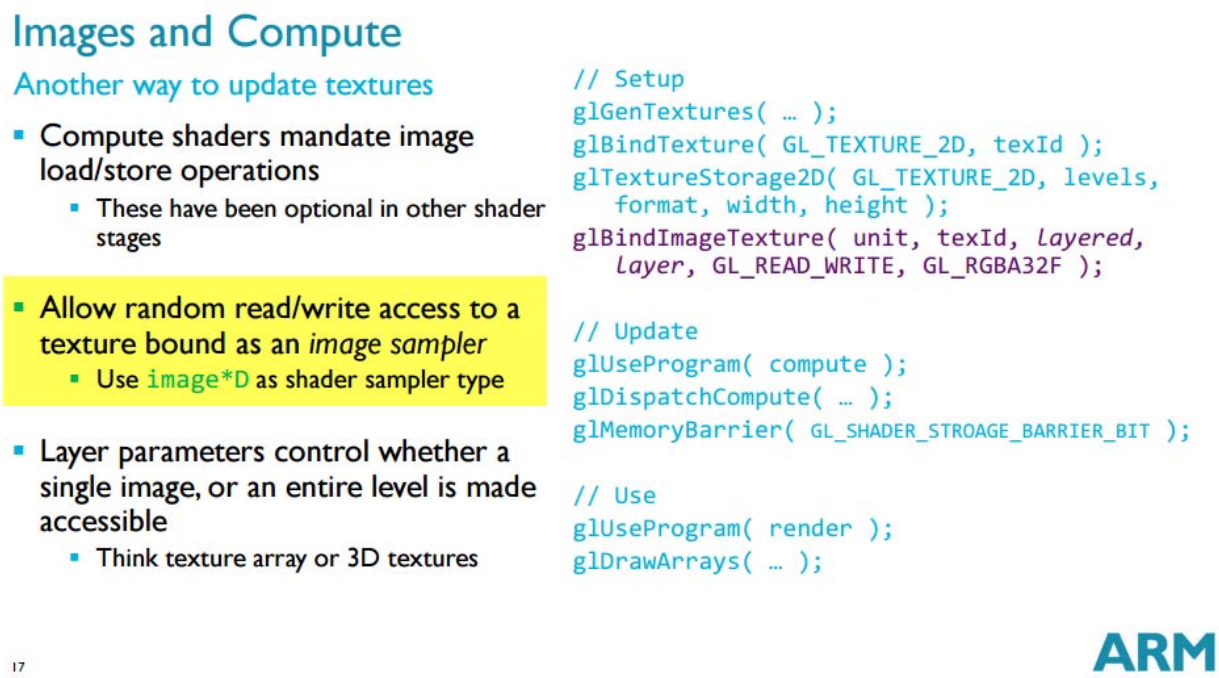
Use mipmaps
Mipmaps increase the cache hit rate because it selects the best resolution of the texture to use based on the variation of texture coordinates.

Use texture compression
This is also good for reducing the memory bandwidth and increasing the cache hit rate. Each compressed block contains more than one texel, so accessing it makes it more cacheable.

Avoid trilinear or anisotropic filtering
Trilinear and anisotropic filtering increase the number of operations required to fetch texels. Avoid using these techniques unless you absolutely require them.

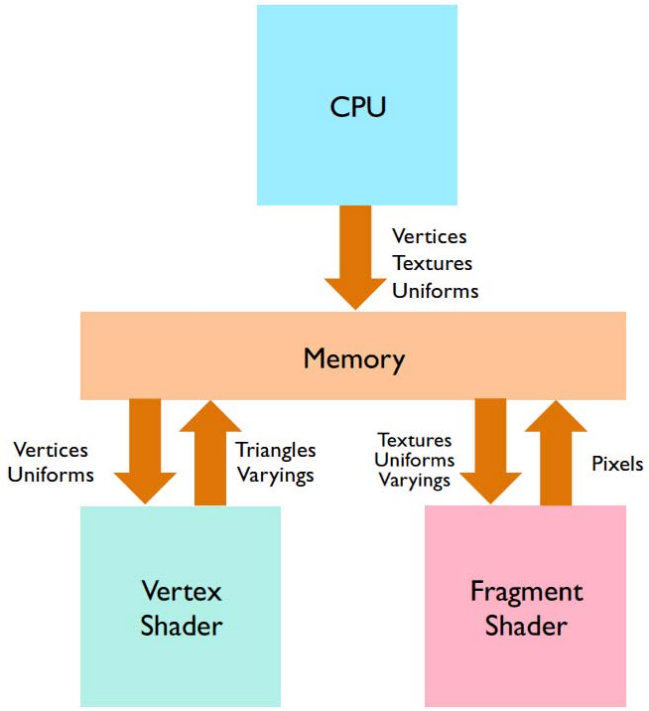
See "ARM Guide to Unity" Version 2.1 page 1-6 available at http://infocenter.arm.com/help/topic/com.arm.doc.100140_0201_00_en/arm_guide_to_unity_enhancing_your_mobile_games_100140_0201_00_en.pdf (accessed 10/27/2016)

"wherein texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory and"



See http://www.gdcvault.com/play/1020140/Getting-the-Most-Out-of.

"wherein texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory and"



See ARM, ARM Tools Part 2, Best Optimization Practices for Mobile Platforms, p.13, *available at* http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/PDFs/6%20-%20ARM%20Tools%20Part%202-%20Best%20Optimization%20Practices%20for%20Mobile%20Platforms.pdf.

Furthermore, the received texture values are written into memory.

“wherein texture operations comprise issuing a texture request to a texture unit and writing received texture values to the memory and”

Registers

The ALU pipeline can read/write to 32 128-bit registers, which can be divided into 4 32-bit (highp in GLSL) components (one vec4) or 8 16-bit (mediump) components (two vec4's). Some of the registers, however, are dedicated to special purposes (see below) and are read-only or write-only.

Special Registers

```
r24 - can mean "unused" for 1-src instructions, or a pipeline register
r26 - inline constant
r27 - load/store offset when used as output register
r28-r29 - texture pipeline results
r31.w - conditional select input when written to in scalar add ALU
```

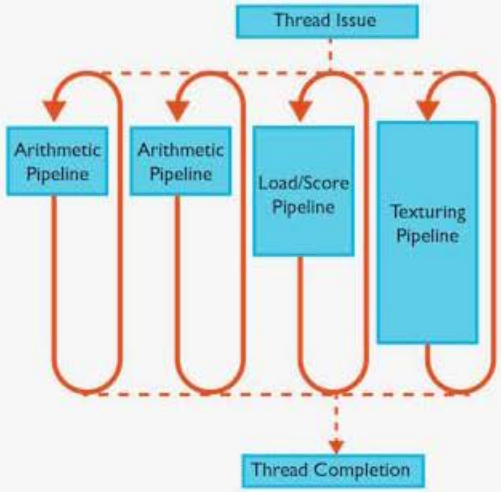
r0 - r23 is divided into two spaces: work registers and uniform registers. A configurable number of registers can be devoted to each; if there are N uniform registers, then r0 - r(23-N) are work registers and r(24-N)-r23 are uniform registers.

See http://limadriver.org/T6xx+ISA/.

“wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and”

wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and	The LG Products include at least one ALU that is operative to read from and write to the memory to perform both texture and color operations. For example, the ALU is designed to “strike a closer balance between shading and texturing.”

"wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and"



As we've stated before, for our purposes we're primarily looking at the Mali-T760. On the T760 ARM uses 2 ALU blocks per tri pipe, which is the most common configuration that you will see for Midgard. However ARM also has Midgard designs that have 1 ALU block or 4 ALU blocks per tri pipe, which is one of the reasons why seemingly similarly GPUs such as T760, T720, and T678 can look so similar and yet behave so differently.

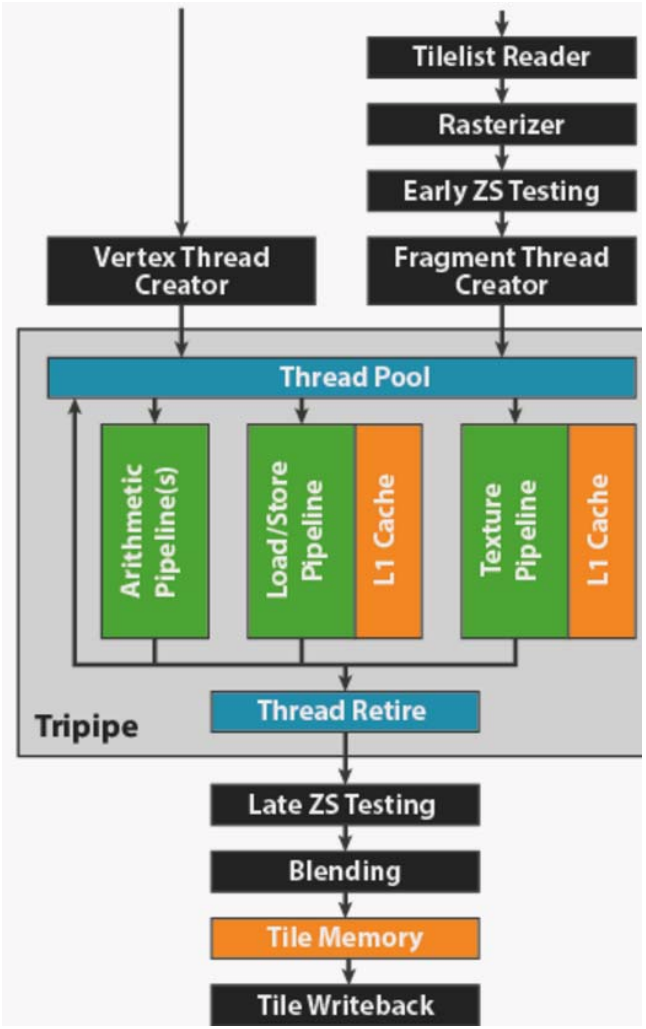
ARM Mali Midgard Arithmetic Pipeline Count (Per Core)	
T628	2
T678	4
T720	1
T760	2

Without being fully exhaustive, among various Midgard designs T628 and T760 are 2 ALU designs, while T720 is a 1 ALU design, and T678 is a 4 ALU design.

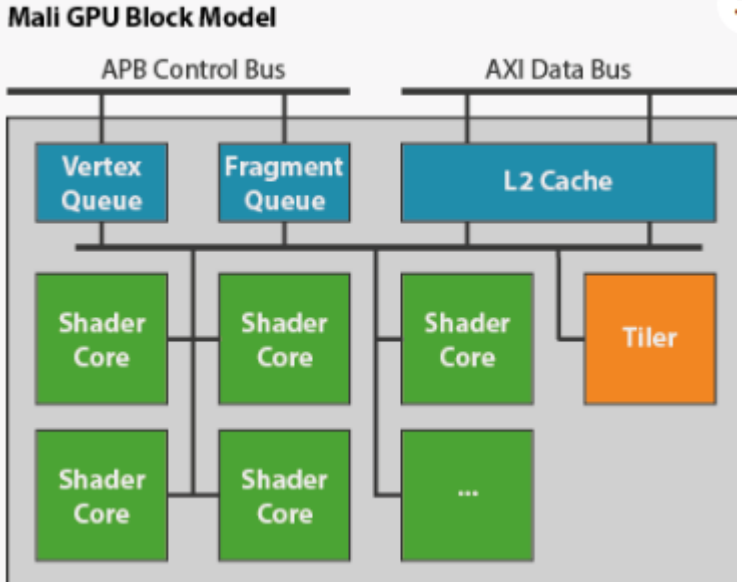
As one would expect, the different number of arithmetic pipelines per tri pipe has a knock-on effect on performance in all aspects, due to the changing ratio between the number of arithmetic pipelines and the number of load/store units and texture units. T678, for example, would be fairly shader-heavy, whereas the 2 ALU designs strike a closer balance between shading and texturing. Among the various Midgard designs ARM has experimented with several configurations, and with the T700 series they have settled on 2 ALU designs for the high-end T760 and 1 ALU for the mid-range T720 (although ARM likes to point out that T720 has some further optimizations just for this 1 ALU configuration).

See http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.

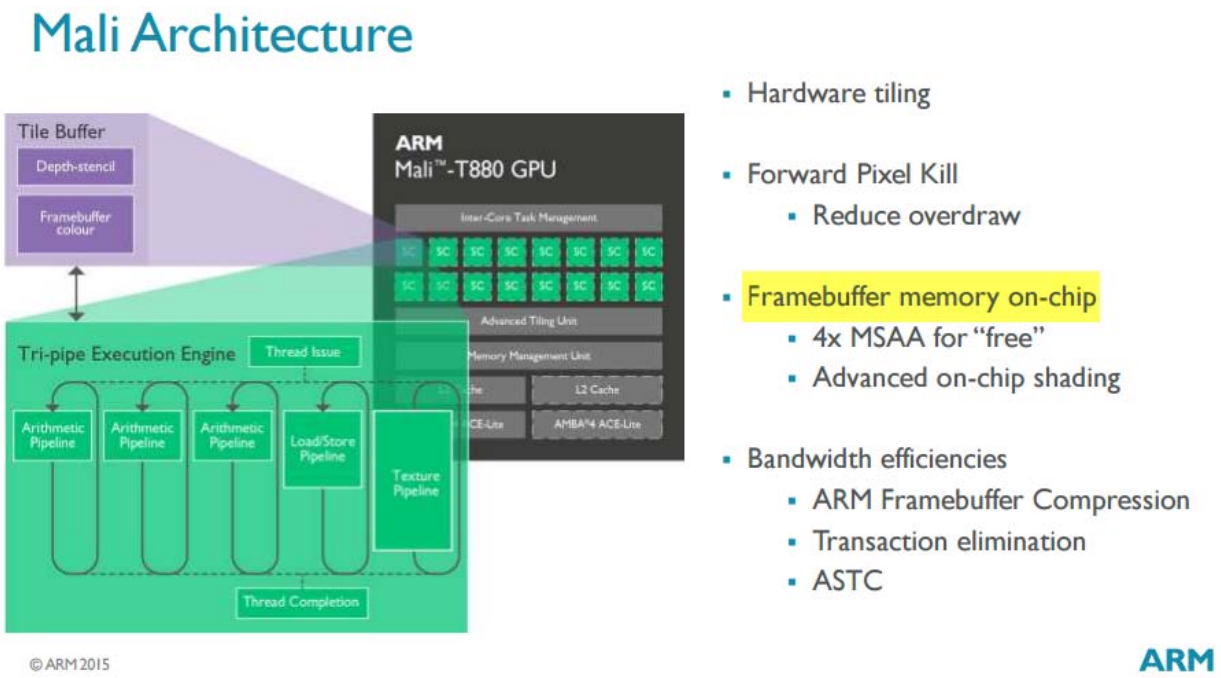
“wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and”

	Further, the ALUs are operative to read from and write to memory to perform texture and color operations as shown below.  *See* https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.

"wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and"



See https://community.arm.com/groups/arm-mali-graphics/blog/2014/03/12/the-mali-gpu-an-abstract-machine-part-3--the-shader-core.



See "Arm Mali GPU Architecture," a presentation by Sam Martin, ARM Graphics Architect at

“wherein the at least one ALU is operative to read from and write to the memory to perform both texture and color operations; and”

http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/LondonDec15/presentations/Mali_GPU_Architecture.pdf (accessed on 10/27/16).

Registers

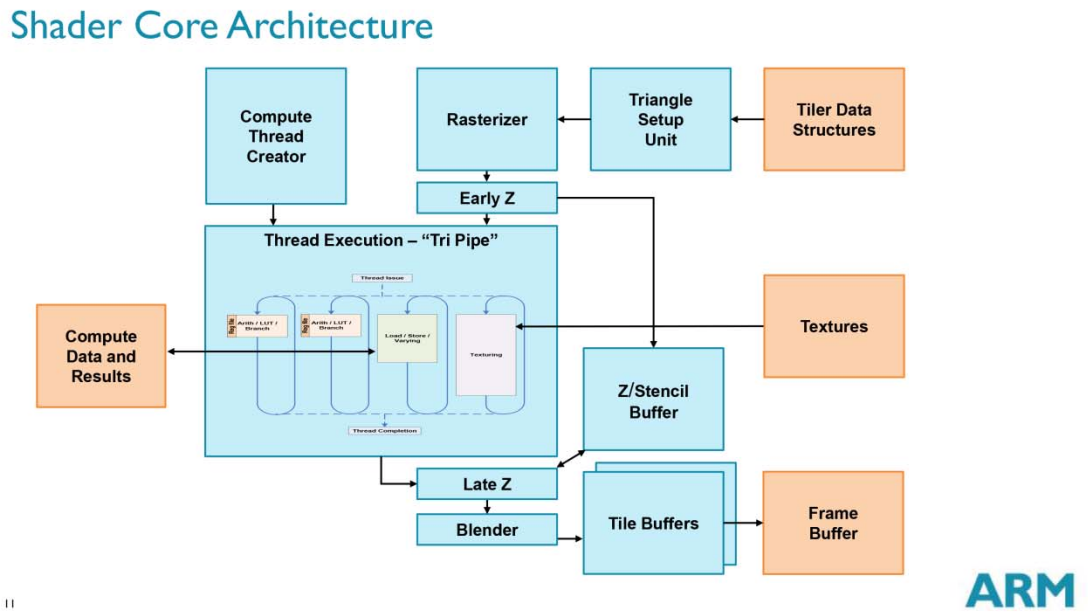
The ALU pipeline can read/write to 32 128-bit registers, which can be divided into 4 32-bit (highp in GLSL) components (one vec4) or 8 16-bit (mediump) components (two vec4's). Some of the registers, however, are dedicated to special purposes (see below) and are read-only or write-only.

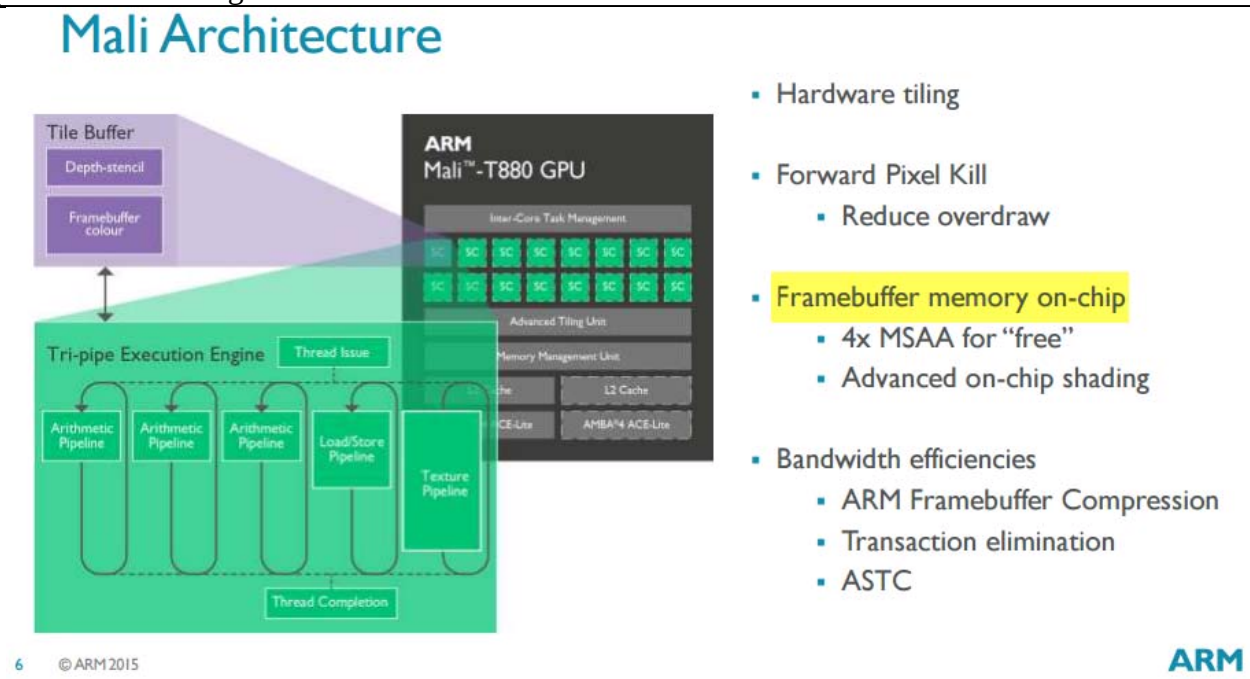
Special Registers

```
r24 - can mean "unused" for 1-src instructions, or a pipeline register
r26 - inline constant
r27 - load/store offset when used as output register
r28-r29 - texture pipeline results
r31.w - conditional select input when written to in scalar add ALU
```

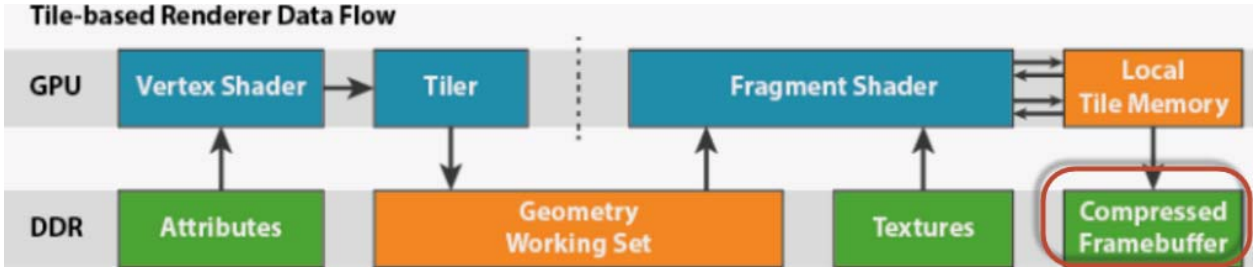
r0 - r23 is divided into two spaces: work registers and uniform registers. A configurable number of registers can be devoted to each; if there are N uniform registers, then r0 - r(23-N) are work registers and r(24-N)-r23 are uniform registers.

See http://limadriver.org/T6xx+ISA/.

an output interface configured to send said resultant value to a frame buffer.	Each of the unified shaders in the LG Products includes an output interface configured to send said resultant values to a frame buffer. For example the LG Product includes an internal and external frame buffer. Moreover, the Mali GPU includes an output interface for writing to the frame buffers.  *See* Ryan Smith, ARM's Mali Midgard Architecture Explored, http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.



See "Arm Mali GPU Architecture," a presentation by Sam Martin, ARM Graphics Architect at http://malideveloper.arm.com/downloads/ARM_Game_Developer_Days/LondonDec15/presentations/Mali_GPU_Architecture.pdf (accessed on 10/27/16).



See http://www.anandtech.com/show/8234/arms-mali-midgard-architecture-explored/4.