Bifrost - The GPU architecture for next five billion

Hessed Choi
Senior FAE / ARM

ARM Tech Forum
June 28th, 2016
Vulkan
What is Vulkan?

- A 3D graphics API for the next twenty years
  - Logical successor to OpenGL and OpenGL ES
  - Modern, efficient design
  - Open, industry-controlled standard

- Here, now
  - Released in February, with unprecedented support
  - Available today for desktop Windows and Linux
  - Officially supported in Android™ ‘N’
  - Shipping today in Samsung Galaxy S7

- Engaged, active developer community
Why ARM loves Vulkan

- A great fit for mobile graphics architectures!
  - No wasted effort trying to look like a desktop GPU
  - Designed to enable mobile-specific optimizations

- Radical commitment to efficiency
  - CPU load is greatly reduced, even on a single core

- Makes your multi-core CPU more useful!
  - Driver work can be distributed across many threads
  - This helps performance and power

- Makes your multi-core GPU more useful too
  - Easier for applications to keep a powerful GPU busy
Bifrost
Bifrost: The new GPU architecture

The increasing pixel impact of modern mobile gaming continues to drive innovation

2010: Utgard

2013: Midgard

2016: Bifrost
ARM Mali processor generations

**BIFROST**
- Mali-G71 GPU
- Unified shader cores, scalar ISA, clause execution, full coherency, Vulkan, OpenCL

**MIDGARD**
- Mali-T600 GPU series
- Mali-T700 GPU series
- Mali-T800 GPU series
- Unified shader cores, SIMD ISA, OpenGL ES 3.x, OpenCL, Vulkan

**UTGARD**
- Mali-200 GPU
- Mali-300 GPU
- Mali-400 GPU
- Mali-450 GPU
- Mali-470 GPU
- Separate shader cores, SIMD ISA, OpenGL ES 2.x
Mali-G71 efficiency drives performance

- 20% Higher energy efficiency*
- 32 Shader cores
- 40% Better performance density*
- 20% Bandwidth Improvement*

Optimized for next generation, advanced, real-world content

*Compared to Mali-T880, on same process node under the same conditions.
Bifrost features

- A more efficient architecture:
  - More performance overall, per mm\(^2\) and per line of *real world* shader code

- Major shader core redesign
  - New scalar, clause-based ISA
  - New quad-based arithmetic units
  - New core fabric

- New geometry data flow
  - Reduces memory bandwidth and footprint
Architectural innovations
Bifrost architectural innovations

Energy efficiency
- Claused shaders
- Index Driven Vertex Shading
- Wire light pipelines

Developer friendly
- Designed for Vulkan and VR/AR

Heterogeneous computing
- Full system coherency

[Diagram showing architectural components of Bifrost and Midgard]
Bifrost GPU design

- Driver Software
- Job Manager
- Shader Core 0
- Shader Core 1
- Shader Core 2
- Shader Core 31
- Control Fabric
- Tiler
- MMU
- L2 Cache Segment
- AXI Memory Bus
- L2 Cache Segment
- AXI Memory Bus
- L2 Cache Segment
- AXI Memory Bus
Scalable system design

Up to 32 shader cores supported
Execution core improvements
Bifrost core design
Bifrost core design

- Compute Frontend
- Fragment Frontend
- Quad Creator
- Quad Creator
- Quad Manager
- Control Fabric
- Execution Engine 0
  - Quad State
- Execution Engine 1
  - Quad State
- Execution Engine 2
  - Quad State
- Execution Engine 3
  - Quad State
- Quad Control
- ZS Memory
- Load/store Unit
- Attribute Unit
- Varying Unit
- Texture Unit
- To L2 Mem Sys
- Blender & Tile Access
- Depth & Stencil
- Tile Memory
- Tile Writeback
- To L2 Mem Sys
Quad vectorization

- Bifrost uses quad-parallel execution
  - Four scalar threads executed in lockstep in a “quad”
  - One quad at a time executes in each pipeline stage
  - Each thread fills one 32-bit lane of the hardware
  - 4 threads doing a vec3 FP32 add takes 3 cycles
  - Improves utilization

- Quad vectorization is compiler friendly
  - Each thread only sees a stream of scalar operations
  - Vector operations can always be split into scalars
Clause execution

- Back-to-back execution guaranteed within a clause
- Allows aggressive optimisation
• Back-to-back register access is common
  • The result from one instruction is often only used as input to the next
Back-to-back register access is common

- Register file bypass saves power.
- Allows use of simpler, smaller register files.
Clause scheduling

- Delay next clause if asynchronous data not ready
Clause scheduling

- Another quad can use this execution unit
- High utilization, high efficiency
Arithmetic functional units
Bifrost arithmetic functional units

- Executes quad-parallel scalar operations
  - 4x32-bit multiplier FMA
  - 4x32-bit adder ADD
  - Adder includes special function unit

- Smaller and more area efficient

- Simplified layout eases compilation
  - Better scheduling in today’s code
  - Better utilization

- One instruction word contains two instructions
Bifrost arithmetic functional units

- Retains support for smaller width data types
  - Integers useful for deep learning
  - 2x performance for FP16 useful for pixel shaders

8-bit integers
16-bit integers
32-bit integers

16-bit floating point
32-bit floating point
Special arithmetic operations

- Special function hardware is smaller than Midgard equivalent
  - Many transcendental functions supported
  - Special functions provide building blocks for compiled shader code
  - Part of the built-in function libraries
Load/store units
New core design

- Execution Engine 0
  - Quad State
- Execution Engine 1
  - Quad State
- Execution Engine 2
  - Quad State

Control Fabric

- Load/store Unit
- Attribute Unit
- Varying Unit

To L2 Mem Sys

- Texture Unit
- Blender & Tile Access
- Depth & Stencil

To L2 Mem Sys

Quad Manager

- Compute Frontend
- Fragment Frontend
- Quad Creator
- Quad Creator

Quad State

ZS Memory

Tile Memory

Tile Writeback

To L2 Mem Sys
### Bifrost load/store units

- Separate units, scheduled separately, for better utilization

<table>
<thead>
<tr>
<th>Unit</th>
<th>Attribute</th>
<th>Varying</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Load/store Unit</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Attribute Unit</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Varying Unit</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Handles most general memory accesses
- Includes memory address translation and coherent caching
- Handles attribute indexing and addressing
- Defers to load/store for actual memory access
- Handles varying interpolation
- Lower power, but more range and precision than Midgard
Geometry flow improvement
Geometry flow: Midgard

Processing

Memory

- Positions
- Attribs
- Trans. Positions
- Varyings
- Indices
- Polygon List

Vertex Shading -> Tiling -> Fragment Shading

Read+Write Bandwidth
[x times of storage size]

- Positions: 3.5x
- Attribs: 2.5x

Leading data stream at the numbered stage

Bandwidth used relative to memory storage size

1x
Geometry flow: Bifrost - index-driven vertex shading

Processing

Memory

Indices Positions Trans. Positions Polygon List Attribs Varyings

Bandwidth used relative to memory storage size

Leading data stream at the numbered stage

Read+Write Bandwidth

[\times \text{times of storage size}]

- Positions: 3.5x
- Attribs: 2.0x
- Positions: 2.5x
- Attribs: 1.5x
Memory system
Memory system

- Driver Software
- Job Manager
- Shader Core 0
- Shader Core 1
- Shader Core 2
- Shader Core 31
- Tiler
- MMU
- L2 Cache Segment
- L2 Cache Segment
- L2 Cache Segment
- AXI Memory Bus
- AXI Memory Bus
- AXI Memory Bus

Full coherency using ACE protocol
Memory system

- Full system coherency support
  - Supports tightly coupled CPU+GPU use cases

- L2 cache improvements
  - Single logical L2 cache makes software easier
  - Fewer partial lines written to AXI which improves LPDDR4 performance