Unified Shader Design

Lan-Da Van (范倫達), Ph. D.
Department of Computer Science
National Chiao Tung University
Taiwan, R.O.C.

Fall, 2016
Source from:

A 195 mW, 9.1 Mvertices/s Fully Programmable 3-D Graphics Processor for Low-Power Mobile Devices

Outline

• Introduction
• 3-D Graphics Processor
  – Bandwidth Reduction
  – Pixel-Vertex Multi-Threading
• Mobile Unified Shader
  – Internal Architecture
  – Low-Power Lighting Engine
  – Unified Shader Datapath
  – Micro-Level Power Management
• Results
• Conclusion
Introduction

• Low-power & small-area for mobile devices.
• Fully programmable 3-D graphics.
• Single unified shader architecture.
3-D Graphics Processor
3-D Graphics Processor

• ISSUE:

- The mobile unified shader uses floating-point matrices, which take thousands of cycles to generate on an embedded RISC processor. The cycle overhead limits the 3-D graphics performance.
3-D Graphics Processor

**SOLUTION:**

- Employ the matrix generator.
- It performs floating-point matrix operation
  - addition, multiplication, inversion, transposition, rotation, deterministic, and so on.
- All those matrix computation is controlled by the host RISC processor using the dedicated matrix generator instructions.
- The workload of 3-D graphics pipeline is well distributed to the RISC processor and 3-D graphics processor.
• ISSUE:
  – The 3-D graphics processor is located in a mobile multimedia SoC as an IP.
  – The 3-D graphics processor shares the bus bandwidth with the other components.
  – The whole system performance is dependent on the communication cycles for transferring the graphics data between external memory and the graphics hardware.
• ISSUE:
  – The 3-D graphics processor fetches required data through AMBA bus and it consumes 696 MB/s bus bandwidth.
    • 216 MB/s for vertex attributes
    • 480 MB/s for rendering data.
  – The AMBA bus which has 400 MB/s capacity can allow less than 120 MB/s for 3-D graphics processor.
  – This bandwidth mismatch causes graphics performance degradation.
Bandwidth Reduction

System Bus

External Memory

MEM CTRL

Input Vertex Data

Graphics Data

Rendering Data

ARM-9

D$

I$

Unified Shader

Transformed Vertices

Vertex Buffer

Rasterizer

Graphics Cache

ROP

3D Graphics Processor

Pixel Buffer

Shaded Pixels
Bandwidth Reduction

- **SOLUTION:**
  - Pre/post vertex cache
    - The graphics data has high data-locality.
    - Due to the triangle strip and fan arrangement
  - 2-D direct mapped cache
    - Up to 98% hit ratio in real 3-D applications
  - Texture cache
  - 4:1 texture compression
SOLUTION:

- With those techniques the rendering data transaction is reduced to 91.2 MB/s.
- As a result, the 3-D graphics processor consumes less than 120 MB/s and it provides its peak-performance in the mobile multimedia SoC.
Bandwidth Reduction

- Vertex Align
  - Post TnL$
  - Tex. Comp.
- Vertex Align
  - Post TnL$
  - Tex. Comp.
  - 2D Mapped $
- Vertex Align
  - Post TnL $
  - Tex. Comp.
  - 2D Mapped $
  - Texture-$
ISSUE:

- Since there is only one processing unit, the graphics data traverse the unified shader twice in a single graphics pipeline.
- 3-D graphics performance is highly related to utilization of the unified shader.
- How to utilize the unified shader effectively?
SOLUTION:

- Pixel-vertex multi-threading (PVMT)
  
  - When the texture cache miss occurs during per-pixel operations and vertex buffer contains vertices to be computed, PVMT issues next vertices and SIMD datapath and SFU perform per-vertex operations until the texture cache filling is finished.

- The contents characteristics such as texture cache miss rate or vertex/pixel ratio highly affect the effect of the PVMT.

- By employing PVMT with 5 entries vertex buffer, 94% of the vertex operations are interleaved into the pixel operations with only 6% performance degradation.
Pixel-Vertex Multi-Threading (3/4)
Pixel-Vertex Multi-Threading (4/4)
Outline

• Introduction
• 3-D Graphics Processor
  – Bandwidth Reduction
  – Pixel-Vertex Multi-Threading
• Mobile Unified Shader
  – Internal Architecture
  – Low-Power Lighting Engine
  – Unified Shader Datapath
  – Micro-Level Power Management
• Results
• Conclusion
Internal Architecture
Internal Architecture

<table>
<thead>
<tr>
<th>F</th>
<th>General SIMD INSTR Fetch</th>
<th>Graphics INSTR Fetch</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Initial INSTR Decoding</td>
<td>Const. Reg. ADDR Generation</td>
</tr>
<tr>
<td>D</td>
<td>Final INSTR Decoding</td>
<td>Const. Reg. (SRAM) Read</td>
</tr>
</tbody>
</table>

**SIMD ALU Pipeline**
- E1: 4-way 32b Floating ALU
- E2: 4-way 32b Floating ALU
- E3: Pipeline Register (E3-E4)
- E4: Pipeline Register (E4-W)

**SIMD MUL Pipeline**
- 4-way 32x16 F-P MUL
- 4-way 32x16 Integer MUL

**SFU Pipeline**
- Log Convert
- Calculate Exp Conv.
- Pipeline Reg. (E3-E4)
- Pipeline Reg. (E4-W)

**LIT Pipeline**
- Log Convert
- Calculate Exp Conv.
- Pipeline Reg. (E3-E4)
- Pipeline Reg. (E4-W)

W All Register Files Writeback
Low-power Lighting Engine

\[
\text{Color}_{\text{ORD}} = C_{\text{amb}} + \left\{ (N' \bullet L_{\text{dir}}) \times C_{\text{diff}} \right\} \\
+ \left\{ (N' \bullet H)^{C_{\text{pow}}} \times C_{\text{spec}} \right\}
\]

\[
\text{Color}_{\text{LNS}} = C_{\text{amb}} + \left\{ (N' \bullet L_{\text{dir}}) \times C_{\text{diff}} \right\} \\
+ 2\left\{ \log_2 (N' \bullet H) \times C_{\text{pow}} + \log_2 C_{\text{spec}} \right\}
\]
Low-power Lighting Engine

![Diagram of Low-power Lighting Engine](image)
Low-power Lighting Engine

Conventional

- DP3
- Result1.x, Ldir, N’;
- Result1.y, H, N’;
- Result1.w, C_{POW};

LIT

- Result2, Result1;

MAD

- Result3, C_{DIFF}, Result2.y, C_{AMB};
- Output_Color.xyz, C_{SPEC}, Result2.z, Result3;

Proposed

- DP3
- Result1.x, Ldir, N’;
- Result1.y, H, N’;
- Result1.w, C_{POW};

MOV

- Result2.x, C_{AMB};
- Result2.y, C_{DIFF};

TLT

- O[COL].xyz, Result1, C_{SPEC}, Result2;

R1.x = Ldir \cdot N’
R1.y = (H \cdot N’)
R1.w = C_{POW}

Compute Lighting Values

Diffuse Light + Ambient Light

+ Specular Light

R2.x = C_{AMB}
R2.y = C_{DIFF}
Unified Adder

[Diagram of Unified Adder with various components labeled]

- Floating Point Input Align Logic
  - EXP COMP.
  - Input Swap
  - Sign Chk
  - Negate
  - Shift

- 8-bit Adder
  - OPS

- 24-bit Adder
  - OPS

- Pipeline Register

- Floating Point Output Align Logic
  - Negate
  - Sign Cal.
  - Normalize / Round

- Fixed-Point Add
- Floating Point Add
Unified Multiplier
Micro-level Power Management
Outline

- Introduction
- 3-D Graphics Processor
  - Bandwidth Reduction
  - Pixel-Vertex Multi-Threading
- Mobile Unified Shader
  - Internal Architecture
  - Low-Power Lighting Engine
  - Unified Shader Datapath
  - Micro-Level Power Management
- Experimental Results
- Conclusion
Chip and Demo Results
Comparison Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation Frequency (MHz)</td>
<td>132/33</td>
<td>132/33</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>Vertex Fill Rate (Mvertices/s)</td>
<td>0.38</td>
<td>1.1</td>
<td>3.6</td>
<td>9.1</td>
</tr>
<tr>
<td>Power Consumption (mW)</td>
<td>38</td>
<td>210</td>
<td>155</td>
<td>195</td>
</tr>
<tr>
<td>Process Technology</td>
<td>0.18μm Logic</td>
<td>0.16μm DRAM</td>
<td>0.18μm Logic</td>
<td>0.13μm Logic</td>
</tr>
<tr>
<td>Programmability</td>
<td>N/A</td>
<td>N/A</td>
<td>Vertex Shading</td>
<td>Vertex Shading Pixel Shading</td>
</tr>
</tbody>
</table>
## Characteristic Results

### Feature Summary

<table>
<thead>
<tr>
<th>Implementation Results</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>0.13μm 7M CMOS</td>
</tr>
<tr>
<td>Power Supply</td>
<td>1.2V</td>
</tr>
<tr>
<td>Transistor Counts</td>
<td>1.3M logic gates</td>
</tr>
<tr>
<td>Die Size</td>
<td>3.3mm x 3.0mm</td>
</tr>
<tr>
<td>Frequency</td>
<td>100MHz</td>
</tr>
</tbody>
</table>
| Power Consumption      | 195mW @ Full 3D Graphics Processing  
ARM-9 : 48mW  
System Peripherals : 22mW  
Mobile Unified Shader : 68mW  
Vertex / Pixel / Fragment Generator : 57mW |

<table>
<thead>
<tr>
<th>Graphics Performance</th>
<th>Description</th>
</tr>
</thead>
</table>
| Performance            | Vertex Fill Rate : 9.1Mvertices/s (w/ TnL)  
Raster Fill Rate : 100Mpixels/s  
Texel Fill Rate : 400Mtexels/s  
Pixel Fill Rate : 7.4Mpixels/s  
(with Vertex Shading, Pixel Shading) |
| Programmability        | Programmable Vertex Shader  
Programmable Pixel Shader |
| Standard API           | OpenGL|ES ver. 1.1  
OpenGL|ES ver. 2.0 |
| Screen Resolution      | Up to 1024 x 1024 pixels |
Conclusion

• Propose low-power fully programmable 3D graphics unified shader engine for mobile devices
  – 35% area reduction
  – 28% power reduction
  – Vertex fill rate: 9.1 Mvertices/s (w/ TnL)

• Propose PVMT scheme
  – 94% of the vertex operations are interleaved into the pixel operations.