A fixed-point 3D graphics library with energy-efficient cache architecture for mobile multimedia system

Min-wuk Lee

2004.12.14
Semiconductor System Laboratory
Department Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology [KAIST]
Outline

- Introduction
- Motivation
- MobileGL: Mobile 3D graphics library
- Energy-efficient CPU cache
- Energy-efficient texture cache
- Conclusion
Performance-energy co-optimization for mobile 3D graphics

- Software system: High speed graphics library (MobileGL)
- Hardware system: Energy-efficient Cache architecture
Target system

- **Low-cost target**
  - High speed graphics library
  - Energy-efficient CPU cache system

- **High quality target**
  - High speed and good quality graphics library
  - Energy-efficient CPU cache, texture cache system
Previous work

For low-cost target

- PC, workstation platform graphics library
  - Too huge GL supported by FPU, special graphics engine
- Embedded platform graphics library: Fixed point arithmetic
  - Yoshida’s work[1]
    - Limited operation (Without texturing)
    - No research on memory bandwidth bottleneck
  - Previous work in our group
    - Analysis with memory-only, without cache system

For high quality target

- Texture cache in PC platform
  - Hakura’s work[2]
  - Analysis based on miss rate
    - Did not consider energy, execution time, system limitation

Motivation

- **Graphics library of this work**
  - Extended operation
    - Lighting, Texturing, Alpha blending, Face culling, etc.
  - Optimization of memory transaction
    - 3D graphics characteristic
    - Analysis with cache system

- **Texture cache of this work**
  - Energy-efficient texture cache in embedded system
    - With negligible performance degradation
Outline

- Introduction
- Motivation
- MobileGL: Mobile 3D graphics library
- Energy-efficient CPU cache
- Energy-efficient texture cache
- Conclusion
Mobile 3D graphics library

- A fixed-point arithmetic
  - 32bit integer
- Optimized memory transaction
  - To reduce instruction and data traffic

- Selective pipeline
  - Applications
  - To reduce branch
  - 45KB total code size

MobileGL block diagram

View transformation → Lighting → Perspective projection → Clipping → Perspective division → Screen mapping → Cull face → Rendering stage

Model

View transformation → Lighting enable → Perspective projection → Clipping → Perspective division → Screen mapping → Cull face → Rendering stage

View transformation → Lighting disable → Perspective projection → Clipping → Perspective division → Screen mapping → Cull face → Rendering stage

Lighting and Texturing
  - Lighting only
  - Texture-only

Pixel

Include r,g,b,a calculation
Exclude r,g,b,a calculation
Disabling option for perspective correction of texture address

- Due to small screen size
- Trade-off between correctness and speedup

<table>
<thead>
<tr>
<th>View transformation</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perspective projection</td>
<td></td>
</tr>
</tbody>
</table>

- Clipping
- Perspective division
- Screen mapping
- Cull face
- Triangle setup
- Horizontal setup
- Pixel interpolation

![Diagram showing execution time reduction](image)

- 30% reduction
- StrongARM at 200 MHz
- Conventional
- This work
- Triangle setup
- Horizontal setup
- Texturing
Division reduction in interpolation
- Use shift instead of reciprocal
- High probability of 1, 2 or 4 in denominator value

1: @ ▲ is 1, 2 or 4, using shift

Execution time (ms) per 1000 Polygons
- Triangle setup
- Horizontal setup
- Texturing

StrongARM @ 200 MHz

27% reduction
Min-wuk Lee

MobileGL(4/6)

- **Z comparison in advance**
  - To avoid unnecessary shading and texturing [3]

- **Selective precision of matrix multiplication**
  - 67% improvement @geometry stage

![Diagram showing standard OpenGL pipeline and performance comparison]

Should extend to 64bit for result

Z fail / Z access

Speed improvement

Library performance

- 67K polygons/sec @ texture only application due to several optimization steps

![Performance Improvement Chart]

- Original C code
- Optimized code

ARM7 @ 80MHz
ARM9 @ 200MHz
StrongARM @ 200MHz
Previous work[1]

67K Polygons/sec
6.7 times Performance improvement

Min-wuk Lee 12
MobileGL (6/6)

- Implementation result
Outline

- Introduction
- Motivation
- MobileGL: Mobile 3D graphics library
- Energy-efficient CPU cache
- Energy-efficient texture cache
- Conclusion
Simulation environment

- From ARM SDK: 1, memory transaction
- From cache model using memory transaction: 2, 3

\[
T_{\text{exe\_total}} = T_{\text{exe\_CPU}} + \sum_{\text{memory\_access\_time}}
\]

\[
T_{\text{exe\_CPU}} = \sum_{K=1}^{\text{instruction\ counts}} T_{\text{exe\_instruction}} + \sum_{K=1}^{\text{cache\ hit\ counts}} CPU\_cycle
\]
Energy-efficient CPU cache (2/4)

Cache model: about execution time

\[ \text{hit\_time} = \text{clock\_period}_{\text{core}} \]

\[ \text{memory\_access\_time} \begin{cases} t_{\text{RCD}} + \text{CAS\_latency} + \text{clock\_period}_{\text{mem}} & \text{(non\_sequential\_access)} \\ \text{clock\_period}_{\text{mem}} & \text{(burst\_access)} \end{cases} \]

\[ \text{miss\_time} \begin{cases} \text{memory\_access\_time} + \text{hit\_time} & \text{(read)} \\ \text{memory\_access\_time} & \text{(write)} \end{cases} \]
Energy-efficient CPU cache (3/4)

Energy modeling

- Tool and research documentation based model
  - Cache hit energy: from *CACTI 3.0* [4]
  - Cache miss energy: from Power & Energy Characterization of the *Itsy Pocket Computer* by *Compaq Western Research Laboratory*
    - 4.70nJ/bus_clock [5], [6]

[4]: CACTI 3.0: An integrated cache timing, power, and area model, Compaq Western Research Laboratory
[5]: Power and energy characterization of the Itsy pocket computer
[6]: A simulation framework for energy-consumption analysis of OS-driven embedded applications, TCAS 2003
Simulation results

Direct mapped data cache, 8E/line (32B line size)

16KB data cache, 32B line size

Using 2-way cache, **13% energy saving**, **1% performance degradation** compared with conventional 4-way cache
Introduction
Motivation
MobileGL: Mobile 3D graphics library
Energy-efficient CPU cache
Energy-efficient texture cache
Conclusion
Energy-efficient texture cache (1/12)

- Texture mapping (Introduction)
  - Map from 3D surface to 2D texel domain (image)
  - Texture coordinate
  - Lookup color in image
  - Lookup method
    - Nearest texel
    - Interpolation of surrounding texels
    - MIPMAP
      - Image pyramid

2D texture diagram

Image pyramid

Level 0

\[ F(x,y,z) = (s,t) \]
Texture filtering methods (Introduction)

- Point sampling, Bilinear filtering, Bilinear MIPMAP, Trilinear MIPMAP

1. Point sampling
2. Bilinear filtering
3. Bilinear MIPMAP
4. Trilinear MIPMAP

Texture space ↔ Screen space ↔ Texture space
Energy-efficient texture cache (3/12)

- Obstacle of texture mapping
  - Requirement of extremely high bandwidth

- Texture cache
  - To reduce the off-chip memory access bottlenecks
  - Image conversion (texture map representation): Reduce conflict miss
  - Address conversion unit (A few logical operations and two additions)
Energy-efficient texture cache (4/12)

- Simulation models

Tiny
6833 polygons

Stealth
542 polygons

Alien
854 polygons

Tiny : LOD[0:1] → 80%, LOD[1:2] → 10%
Stealth : LOD[0:1] → 67%, LOD[1:2] → 15%
Alien : LOD[0:1] → 48%, LOD[1:2] → 44% @ trilinear MIPMAP
Energy-efficient texture cache (5/12)

- Proposed texture map representation
  - Reduce conflict miss at bank change
  - Miss rate reduction, energy saving (17.4%), execution time reduction (15.2%)

Blocked representation → Recursive Sub Block
Energy-efficient texture cache (6/12)

- Address conversion unit for RSB2X2
  - Use one-to-one correspondence and find rule
    - Hardware implementation: only thirteen 2:1mux in trilinear MIPMAP

Address conversion unit of this work: RSB 2X2

256 X 256, RSB 2X2

64 X 64, RSB 2X2

128 X 128, RSB 2X2

32 X 32, RSB 2X2
Energy-efficient texture cache (7/12)

- Texture cache model using bank interleaved

Morton order representation ← previous work
Proposed RSB2X2 also free from bank conflict
Energy-efficient texture cache (8/12)

- Performance and Energy comparison between filtering method
  - Energy consumption, Execution time
    - Point sampling < Bilinear filtering < Bilinear MIPMAP < Trilinear MIPMAP
  - Trade off point: Image quality (aliasing criterion)

Normalized energy:

- D.M.
- 2WAY
- 4WAY

Normalized @

2KB, 16entries/line, Tiny_model
Energy-efficient texture cache (9/12)

- Image quality analysis
  - Textile model
  - LOD[0:1]: 44%, LOD[1:2]: 40% in MIPMAP

- DCT analysis
  - Low frequency term in top-left
Energy-efficient texture cache (10/12)

- Image quality metric in terms of aliasing criterion:

\[
\text{image\_quality\_0.5} = \frac{\sum_{fx=0}^{\pi/2} \sum_{fy=0}^{\pi/2} \text{amplitude}}{\sum_{fx=0}^{\pi} \sum_{fy=0}^{\pi} \text{amplitude}}
\]

\[
\text{image\_quality\_0.75} = \frac{\sum_{fx=0}^{3\pi/4} \sum_{fy=0}^{3\pi/4} \text{amplitude}}{\sum_{fx=0}^{\pi} \sum_{fy=0}^{\pi} \text{amplitude}}
\]

- \(\text{Index}_Q, \text{Index}_E\):
  - To find relative value
  - Normalize from 0 to 1

\[
\text{Index}_Q = \frac{\text{cur}_Q - \text{min}_Q}{\text{max}_Q - \text{min}_Q}
\]

\[
\text{Index}_E = \frac{\text{max}_E - \text{cur}_E}{\text{max}_E - \text{min}_E}
\]
Energy-efficient texture cache (11/12)

- **Index = Index_Q + Index_E**
  - Almost same quality between B.M. and T.M. in QVGA
  - Large different energy between B.M. and T.M.
  - Poor image quality in P.S.
  - Bilinear MIPMAP get the largest score.

2-way set associative, 2KB texture cache
Simulation results

Energy comparison while changing cache size

@ 2-way, using bilinear MIPMAP

4KB texture cache, 16B line size (2B per 1 texel) → energy-efficient, low cost, high-quality
Outline

- Introduction
- Motivation
- MobileGL: Mobile 3D graphics library
- Energy-efficient CPU cache
- Energy-efficient texture cache
- Conclusion
Conclusion

- For performance-energy co-optimization in Mobile3D graphics
  - MobileGL / Cache architecture
- MobileGL : Mobile 3D graphics library
  - 67K polygons/sec
  - 66.1% performance improvement in average
- Energy-efficient CPU cache
  - 2-way set associative cache to save energy
- Energy-efficient texture cache
  - Proposed texture map representation
  - Bilinear MIPMAP shows good quality to energy ratio
  - 16B line size , 4KB size cache is the optimal point
Supplemental Materials
Min-wuk Lee

Graphics pipeline

- **Geometry stage**
  - Camera position
  - View frustum
  - Unit-cube

- **Rendering stage**
  - Camera direction
  - View transform
  - Projection Clipping 1/w
  - Screen mapping

**Triangle setup**:
For line1, line2, line3 using 1st, 2nd, 3rd

**Horizontal setup**:
For line_x using start, end

**Pixel interpolation**:
Each pixel shading, texturing

**Rendering stage**

```
1st
Top

Mid

3rd
Line1

Line2

Line3

2nd
Bot

Direction_x

start

end

Direction_y
```
ARM920T and M*CORE: Caches consume 50% of total processor system power (Segars 01, Lee et al. 99)
Min-wuk Lee

**Blocked representation**

- Conventional texture map representation (16X16 blocked)
  - Conflict miss @ block change
    - A path : 2
    - B path : 16
    - C path : 16

**Assumptions**

1. Bilinear filtering
2. Cache size = block size
3. 16 entries / 1 line
Proposed texture map representation

- Recursive Sub Block texture map representation (RSB4X4)
  - Conflict miss @block change
    - A path: 4
    - B path: 4
    - C path: 4

Assumptions
1. Bilinear filtering
2. Cache size = block size
3. 16 entries/1line

@ recursive sub-block 4X4 method
Simulation between representation methods

- Simulation results between texture representations
  - Bilinear filtering, 2-way, 1KB texture cache
  - 27% performance improvement in average
  - Low miss rate doesn’t mean high performance
  - 8entries/line or 16entries/line shows good performance
Simulation between representation methods

- @ bilinear filtering
  - Low miss rate doesn’t mean low energy consumption
  - RSB accomplish 17.4% energy saving compared to the best of conventional methods

- @ point sampling
  - 25% performance improvement in average
Morton order

- Multi-ported cache
  - To access more than 1 texel in the same cycle
- Interleaving the cache lines across multi-banks
- Morton order
- RSB4X4 : Not free from bank conflict
  - → RSB2x2
- Trilinear filtering : Not free from bank conflict
  - → Cache for even, odd LOD
Proposed texture map representation

- RSB2X2 map representation
  - Bank conflict free