A High Performance 3D Graphics Rasterizer with Effective Memory Structure

Woo-Chan Park, Kil-Whan Lee, Seung-Gi Lee, Moon-Hee Choi, Won-Jong Lee, Cheol-Ho Jeong, Byung-Uck Kim, Woo-Nam Jung, Il-San Kim, Won-Ho Chun, Won-Suk Kim, Tack-Don Han, Moon-Key Lee, Sung-Bong Yang, and Shin-Dug Kim

Media System Lab.
Yonsei University
Seoul, Korea
E-mail: kiwh@kurene.yonsei.ac.kr
Outline

- Introduction
- David Simulator
- High Performance 3D Graphics Rasterizer with Effective Memory Structure (David Rasterizer)
- Performance Analysis
- Conclusions
Introduction
**NRL Project**

**Title**
The Design of High Performance 3D Graphics Accelerator for Realistic Image

**Properties**
- Institution for bringing up an excellent lab. with a core technology
- A Government-initiated Project

**Necessities**
- Leadership of an advanced research area
- Synergy effect through research interchanges

---

**Yearly Research Plan**

**1st Year**
- Basic Environment & Research
  - Study Simulation Environment
  - 3D GA Simulator development
  - Survey of Related Works

**2nd Year**
- Technology Prevalent 3D Graphic Accelerator
  - Propose an Effective Architecture
  - Performance Evaluation
  - IP Co-Development

**3rd Year**
- High Performance 3D Graphic Accelerator
  - High Performance Architecture
  - Building international core IP
  - Implementation of Prototype System
  - Parallel Rendering Architecture

**5th Year**
- High Performance Architecture
  - Implementation of Prototype System

---

COOL Chips IV
The Design of A High Performance 3D Graphics Accelerator for Realistic Image & Building Core IPs

<table>
<thead>
<tr>
<th>Architecture Research</th>
<th>Technology Prevalent 3D Graphics Accelerator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Geometry Processing Unit, Rendering Unit</td>
</tr>
<tr>
<td></td>
<td>• Realization Mapping Unit</td>
</tr>
<tr>
<td></td>
<td>• P–M Architecture, Memory Architecture for 3D GA</td>
</tr>
<tr>
<td></td>
<td>• Cache &amp; Memory Hierarchy</td>
</tr>
<tr>
<td></td>
<td>• Parallel 3D Rendering System</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design Research</th>
<th>High Performance 3D Graphics Accelerator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Execution Model (VLIW, SIMD, RISC etc.), Control &amp; Interface</td>
</tr>
<tr>
<td></td>
<td>• Appliance to other system library by implementing VHDL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SW Research</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• API &amp; Rendering Algorithm</td>
</tr>
<tr>
<td></td>
<td>• Geometry Compression / Modeling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verification/Integration</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Construction of Simulation Environment &amp; Verification by Simulation</td>
</tr>
<tr>
<td></td>
<td>• Prototype System</td>
</tr>
</tbody>
</table>

COOL Chips IV
Current Research Work

Major Research

- High performance floating point adder/subtractor
- High performance floating point multiplier
- High performance floating point divider
- Overlapped light geometry processing (OLGP)
- New method for topological compression
- Object-oriented rendering using the analytic model
- Effective reuse buffer for triangle mesh
- Order-independent transparency
- Perspective texture mapping
- Efficient bump mapping
- Modified anti-aliasing execution model
- Texture cache simulation for various cache architectures
- Texture Cache Sharing
- Memory bandwidth saving scheme for texture data

Related Works & Basic Research

David Simulator

- Vertex data (x, y, z, w)
- Model-view Transform
- (Lighting)
- Clipping
- Projection
- Divide by w
- Viewport Transform
- Triangle Setup
- Scan-conversion
- Bump map
- Perspective Texture Mapping
- Fog/Alpha blending
- Z-buffering
- Anti-aliasing
- Pixel data
David Simulator
setup

Edge Walk

span processing

z-test pipeline

mapping pipeline

lambda calculation

gradient value of \( U \) and \( V \), \( W \), \( D \)

lambda calculation

\( X \), \( Y \), \( Z \)

\( P, G, B, \alpha \)

x, y

x, y

z

distortion

address calculation

texel addresses, bump xel addresses, exel address

tags

cache addresses

z-test pipeline

request fifo

reorder buffer

texture cache

texels, bump xel, exel

interpolation

bump engine

eenv. engine

color blend

fragment fifo

z-buffer

compare

z-buffer

frame buffer

address

address z-buffer

frame buffer

pixel cache

memory controller

address 24 bits

data 128 bits

frame buffer/texture memory/bump map/environment map

cool chips iv
High Performance 3D Graphics Rasterizer with Effective Memory Structure (David Rasterizer)
Architectural features of David rasterizer

- Performing z-test pipeline before TBE (Texture, Bump, and Environment) mapping completion
  - Saving memory bandwidth
  - Solve the inconsistency problem with tagging scheme for pixel cache

- Texture cache sharing with BE (Bump and Environment) mapping
  - Efficient structure
Rasterizer model: Neon, S3

Neon

1. Pixel information
2. Texture read / filter
3. Texture blend
4. Z read
5. Z test
6. Alpha test
7. Z write
8. Destination read
9. Alpha blend
10. Destination write

S3

1. Pixel information
2. Z read
3. Z test
4. Z write
5. Texture read / filter
6. Texture blend
7. Alpha test
8. Destination read
9. Alpha blend
10. Destination write

COOL Cmps IV
David Rasterizer

Pixel information
- Tag test and Z read
- Z test
- Texture read / filter
- Texture blend
- Alpha test
- Tag update and Z write
- Destination read
- Alpha blend
- Destination write

Pixel cache
Texture cache

memory

wide separate
## Architecture Comparison

<table>
<thead>
<tr>
<th></th>
<th>Neon</th>
<th>S3</th>
<th>David</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>When is texture mapping performed?</strong></td>
<td>before Z test</td>
<td>after Z test</td>
<td>after Z test</td>
</tr>
<tr>
<td><strong>OpenGL semantics for perfectly transparent texture</strong></td>
<td>Support</td>
<td>Not support</td>
<td>Support</td>
</tr>
</tbody>
</table>
| **Advantages**                       | • Support OpenGL semantics | • No wasting bandwidth  
                               |               |
|                                      |               | → No fetching texture data that are obscured | • Simple scheme  
                               |               |
|                                      | • Wasting bandwidth | • Unable to support OpenGL semantics | • Wide separation  
                               |               |
|                                      |               |               | → Inconsistency problem  
                               |               |
|                                      |               |               | → Solve it using additional flag bits in a pixel cache |
Texture Cache Sharing

Current 3D Architecture

Texture Mapping #1 → Cache → DRAM
Bump Mapping → DRAM
Environment Mapping → DRAM

David 3D Architecture

Texture Mapping #1
Shared H/W
Bump Mapping
Texture Mapping #2
Shared H/W
Environment Mapping

Current Architecture

<table>
<thead>
<tr>
<th>Mapping Hardware</th>
<th>Independent H/W</th>
<th>David Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce H/W cost (about 30%)</td>
<td>Remove Pipeline Stalls due to DRAM Access</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cache Size</th>
<th>Same</th>
<th>Same</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Read Port in Cache</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Throughput</td>
<td>1 Cycle</td>
<td>Texture Mapping : 1 Cycle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BE Mapping : 2 Cycles (infrequent operations)</td>
</tr>
</tbody>
</table>

COOL Chips IV
Performance Analysis
Environments for Performance Evaluation

Model Data

- OpenGL format
- Mesa Library Call
- Rasterizer Simulator Call

Setup pipeline
Edge work pipeline
Span processing
Z Compare
Color Blend
MC for frame buffer access

Rasterizer

- Texture cache
- Mapping unit (texture, bump, environment, displacement)
- MC for image map access

Trace Generation
- Texture Cache Simulator
- Pixel Cache Simulator

Miss Ratio, Performance

Z Depth Complexity, # of Z Test Fails
Model Data

SPECviewperf

ProCDRS

Lightscape

Crystal Space
(Game engine)

Blocks

Flarge

COOL Chips IV
Bandwidth Saving in Texture Data (ProCDRS)

- Depth Complexity vs. Bandwidth Saving

Average Depth Complexity: 2.22
Average Bandwidth Saving: 21.55%
Bandwidth Saving in Texture Data (Light)

Depth Complexity vs. Bandwidth Saving

Frame Number

Depth Complexity | Bandwidth Saving
---|---
Average | 2.31 | 29.16%
**Bandwidth Saving in Texture Data (Blocks)**

- Depth Complexity
- Bandwidth Saving

Frame Number

<table>
<thead>
<tr>
<th>Depth Complexity</th>
<th>Bandwidth Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1.43</td>
</tr>
<tr>
<td></td>
<td>4.76%</td>
</tr>
</tbody>
</table>

COOL Chips IV
Bandwidth Saving in Texture Data (Large)

Depth Complexity vs. Bandwidth Saving

Frame Number

Depth Complexity, % Bandwidth Saving

Average Depth Complexity: 1.31
Average Bandwidth Saving: 4.93%

COOL Chips IV
Conclusions
Conclusions

- Simulation Environment (David Simulator)
  - Evaluation for 3D graphics accelerator architecture
  - Performance comparison

- David Architecture
  - Performing z-test before TBE mapping completion
    - 5%~29% bandwidth savings for texture data in Scenes with 1~3 depth complexity
    - As the depth complexity grows, the amount of bandwidth savings become large
      - Recently, a 3D graphic application shows high depth complexity
  - Texture cache sharing with BE mapping
    - Hardware saving from sharing and hardware reduction for BE mappings