An Architecture Extension for Efficient Geometry Processing

Radhika Thekkath,
Mike Uhler,
Chandlee Harrell,
Ying-wai Ho

MIPS Technologies, Inc.
1225 Charleston Road
Mountain View, CA 94043
Talk Outline

- Motivation---why enhance the MIPS architecture
- Background on 3D graphics geometry operations and current MIPS architecture
- What are the enhancements?
- Performance and cost
- Summary
Current 3D Rendering Limited by Geometry Processing

- Front-end Geometry and Lighting operations
- Back-end: Rendering
- A solution: dedicated hardware --- high-performance, but expensive.
  Eg. Sony Emotion Engine
Our Solution

- Enhance the MIPS architecture to improve 3D geometry performance: MIPS-3D™ ASE (Application Specific Extension) includes 13 new instructions
- Lower cost than dedicated geometry hardware
- Main processor improvements are leveraged
  - technology/speed
  - parallelism/pipelining
Talk Outline

- Motivation---why enhance the MIPS architecture
- **Background on 3D graphics geometry operations and current MIPS architecture**
- What are the enhancements?
- Performance and cost
- Summary
Geometry and Lighting Operations

- Vertex transformation (matrix multiplication)
- Clip-check (compare and branch)
- Transform to screen coordinates (perspective division using reciprocal)
- Lighting: infinite and local (normalization using reciprocal square root)
Already in the MIPS Architecture

S - Single FP format (32 bits)
D - Double FP format (64 bits)
PS - Paired-Single, two singles

Floating point operations
- MUL (S, D, PS)
- ADD (S, D, PS)
- MADD (S, D, PS) (multiply-add)
- RECIP (S, D)
- RSQRT (S, D)

<table>
<thead>
<tr>
<th>S</th>
<th>S</th>
</tr>
</thead>
</table>

64 bits
Talk Outline

● Motivation---why enhance the MIPS architecture
● Background on 3D graphics geometry operations and current MIPS architecture
● **What are the enhancements?**
● Performance and cost
● Summary
**ADDR: for Vertex Transformation**

\[
\begin{bmatrix}
x & y & z & w
\end{bmatrix} \times \begin{bmatrix}
m0 & m4 & m8 & m12 \\
m1 & m5 & m9 & m13 \\
m2 & m6 & m10 & m14 \\
m3 & m7 & m11 & m15
\end{bmatrix} = \begin{bmatrix}
x^t & y^t & z^t & w^t
\end{bmatrix}
\]

Eg. \( x^t = m0x + m1y + m2z + m3w \)

---

**MUL.PS FP10, FP0, FP8**

- FP0 = \([m1 \mid m0]\)
- FP1 = \([m3 \mid m2]\)
- FP8 = \([y \mid x]\)
- FP9 = \([w \mid z]\)
- FP10 = \([m1y \mid m0x]\)
- \([m3w \mid m2z]\)

**MADD.PS FP11, FP10, FP1, FP9**

- FP11 = \([m1y+m3w \mid m0x+m2z]\)

---

Reorganize register to enable add
ADD.PS ...

**ADDR.PS FP11, FP?, FP11**

- FP11 = \([-\mid m1y+m3w+m0x+m2z]\)

---

An Architecture Extension for Efficient Geometry Processing
Clip Check (Compare)

Is the vertex within the viewing pyramid?

\[
\begin{align*}
x & \geq -w, \ x \leq w \\
y & \geq -w, \ y \leq w \\
z & \geq -w, \ z \leq w
\end{align*}
\]

Set 6 Condition Code (CC) bits

Observation: Can use magnitude compares.

\[
\begin{align*}
|x| & \leq |w| \\
|y| & \leq |w| \\
|z| & \leq |w|
\end{align*}
\]

Set only 3 CC bits
CABS: for Clip Check Compare

Transformed \([w \mid z] \ [y \mid x]\) in FP registers

- \(\text{PUU.PS}\) to get \([w \mid w]\)
- \(\text{NEG.PS}\) to get \([-w \mid -w]\)
- \(\text{C.NGE.PS}\) \(! (y >= -w)? ! (x >= -w)?\)
- \(\text{C.NGE.S}\) \(! (z >= -w)?\)
- \(\text{C.LE.PS}\) \(y <= w? \ x <= w?\)
- \(\text{C.LE.S}\) \(z <= w?\)

Replace with absolute compares

\(\text{CABS.LE.PS}\) \(|y| <= |w|?, \ |x| <= |w|?\)
\(\text{CABS.LE.PS}\) \(|w| <= |w|?, \ |z| <= |w|?\)
BC1ANY4F: for Clip Check Branch

- Without absolute compare, need 6 branch instructions to check the 6 CC bits.
- With absolute compare, need 3 branch instructions to check the 3 CC bits.
- New MIPS-3D™ ASE instruction --- BC1ANY4F, a single branch instruction that checks 4 CC bits.
Geometry and Lighting Operations

- Vertex transformation (matrix multiplication)
- Clip-check (compare and branch)
- Transform to screen coordinates (perspective division using reciprocal)
- Lighting: infinite and local (normalization using reciprocal square root)
Perspective Division and Normalization

- In MIPS V architecture
  - RECIP
  - RSQRT
- Full precision
- Long latency
- Not fully pipeline-able
- Only S and D formats
- New MIPS-3D™ ASE instructions:
  - RECIP1
  - RECIP2
  - RSQRT1
  - RSQRT2
- Reduced & full precision
- Pipeline-able
- PS format
Talk Outline

- Motivation---why enhance the MIPS architecture
- Background on 3D graphics geometry operations and current MIPS architecture
- What are the enhancements?
- Performance and cost
- Summary
Implementation Cost

- **Die Area (of the Ruby processor)**
  - Implementation of PS adds 6-8% to FP die area.
  - MIPS-3D™ ASE adds 3% to the floating point die area. (FP is less than 15% of the total die area).

- **Logic/pipeline complexity**
  - ADDR, CABS, BC1ANY4F, etc. - minimal impact on both die area and FP pipeline logic.
  - RECIP1, RSQRT1 - 128 word lookup tables contribute to most of the 3% die area increase.
Other Instruction Sets

- 3DNow! -- enhance 3D graphics and multimedia
  - 2-packed FP SIMD (PS)
  - PFACC - accumulate
  - PFRCPI, PFRCPI1, PFRCPI2 - reciprocal
  - PFRRSQR, PFRRSQIT1 - reciprocal square root
  - PF2ID, PI2FD - convert

- AltiVec
  - 4 SIMD (32-bits)
  - vrefp, vnmsubfp, vmaddfp - reciprocal
  - vrsqrtefp, etc - reciprocal square root
  - vcmpbfp - bounds compare
  - vcsx, vctsxs - convert
Performance: Number of Instructions

Note: Inner-loop instructions=cycles

<table>
<thead>
<tr>
<th></th>
<th>No PS + No MIPS-3DTM ASE</th>
<th>PS + No MIPS-3DTM ASE</th>
<th>PS + MIPS-3DTM ASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform (matrix transform + clip + perspective divide)</td>
<td>29</td>
<td>28</td>
<td>20</td>
</tr>
<tr>
<td>Transform + complex lighting</td>
<td>90</td>
<td>67</td>
<td>49</td>
</tr>
</tbody>
</table>
Experiment/Coding Assumptions

- FP pipeline has 4-cycle data dependency
- Loop interleaves computations of 2 vertices
- Transform constants locked in cache
- Vertex co-ordinates are pre-fetched from memory to cache, every loop iteration
- Code uses full precision reciprocal and reduced precision reciprocal square-root
Performance: M polygons/s
Using today’s high-end desktop processor frequency---500MHz

An Architecture Extension for Efficient Geometry Processing
Summary

- MIPS-3D™ ASE adds thirteen instructions to the current MIPS 64-bit architecture
- Low cost (3% die area)
- Increases polygons/sec count by 45% for the transform code
- Increases polygons/sec count by 36% -- 83% for the transform+complex light code
Appendix: Vertex Transformation Code

FP0--FP7 hold m0--m15 in pair-single
FP8, FP9 hold x,y,z,w in pair-single

MUL.PS FP10,FP8,FP0
MUL.PS FP11,FP8,FP2
MUL.PS FP12,FP8,FP4
MUL.PS FP13,FP8,FP6
MADD.PS FP11,FP11,FP9,FP3
MADD.PS FP12,FP12,FP9,FP5
MADD.PS FP13,FP13,FP9,FP7
PLL.PS FP14,FP11,FP10
PUU.PS FP15,FP11,FP10
PLL.PS FP16,FP13,FP12
PUU.PS FP17,FP13,FP12
ADD.PS FP8, FP15,FP14
ADD.PS FP9,FP17,FP16

ADDR.PS FP8,FP11,FP10
ADDR.PS FP9,FP13,FP12

FP8 <-- m4x+m5y+m6z+m7w | m0x+m1y+m2z+m3w
FP9 <-- m12x+m13y+m14z+m15w |
     m8x+m9y+m10z+m11w

Replace with
# Appendix: The 13 MIPS-3D™ ASE Instructions

<table>
<thead>
<tr>
<th>Type</th>
<th>Mnemonic</th>
<th>Valid Formats</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>ADDR</td>
<td>PS</td>
<td>Floating point reduction add</td>
</tr>
<tr>
<td></td>
<td>MULR</td>
<td>PS</td>
<td>Floating point reduction multiply</td>
</tr>
<tr>
<td></td>
<td>RECIP1</td>
<td>S, D, PS</td>
<td>Reciprocal first step – reduced precision</td>
</tr>
<tr>
<td></td>
<td>RECIP2</td>
<td>S, D, PS</td>
<td>Reciprocal second step – enroute to full precision</td>
</tr>
<tr>
<td></td>
<td>RSQRT1</td>
<td>S, D, PS</td>
<td>Reciprocal square root first step – reduced precision</td>
</tr>
<tr>
<td></td>
<td>RSQRT2</td>
<td>S, D, PS</td>
<td>Reciprocal square root second step</td>
</tr>
<tr>
<td>Format Conversion</td>
<td>CVT.PS.PW</td>
<td>PW</td>
<td>Convert a pair of 32-bit fixed point integers to a pair-single floating point value</td>
</tr>
<tr>
<td></td>
<td>CVT.PW.PS</td>
<td>PS</td>
<td>Convert a paired-single floating point value to a pair of 32-bit fixed point integer values</td>
</tr>
<tr>
<td>Compare</td>
<td>CABS</td>
<td>S, D, PS</td>
<td>Magnitude compare of floating point values</td>
</tr>
<tr>
<td>Branch</td>
<td>BC1ANY2F</td>
<td></td>
<td>Branch if either one of two (consecutive) CC bits is F</td>
</tr>
<tr>
<td></td>
<td>BC1ANY2T</td>
<td></td>
<td>Branch if either one of two (consecutive) CC bits is T</td>
</tr>
<tr>
<td></td>
<td>BC1ANY4F</td>
<td></td>
<td>Branch if any one of four (consecutive) CC bits is F</td>
</tr>
<tr>
<td></td>
<td>BC1ANY4T</td>
<td></td>
<td>Branch if any one of four (consecutive) CC bits is T</td>
</tr>
</tbody>
</table>