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

MIPS TECHNOLOGIES

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

MIPS TECHNOLOGIES

Current 3D Rendering Limited by Geometry Processing

- Front-end: Geometry and Lighting operations
  - General-purpose processors: 0.5 - 2 M polygons/s. Eg. R5000® (1996,200MHz), PIII (1999,500MHz).
- Back-end: Rendering
  - Graphics processors: 6 - 8 M polygons/s. Eg. ATI Rage 128(1999), 3Dfx Voodoo3(1999).
- Dedicated hardware, eg., Sony Emotion Engine---silicon-intensive, but feeds higher performance rendering engines.

MIPS TECHNOLOGIES

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

MIPS TECHNOLOGIES

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

MIPS TECHNOLOGIES

Geometry and Lighting Operations

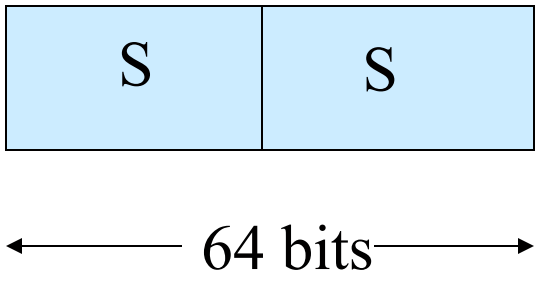
- Vertex transformation (matrix multiplication)
- Clip-check (compare and branch)
- Transform to screen coordinates (perspective division using reciprocal)
- Lighting: infinite and local (normalization using reciprocal square root)

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)

MIPS TECHNOLOGIES

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

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

Eg. $x^t = m0x + m1y + m2z + m3w$

FP0 = [m1 | m0]

FP1 = [m3 | m2]

*

*

FP8 = [y | x]

FP9 = [w | z]

MUL.PS FP10, FP0, FP8

FP10 = [m1y | m0x]

\+ [m3w | m2z]

MADD.PS FP11, FP10, FP1, FP9

FP11 = [m1y+m3w | m0x+m2z]

Reorganize register to enable add ADD.PS ...

ADDR

ADDR.PS FP11, FP?, FP11

FP11 = [y^t | x^t=m1y+m3w+m0x+m2z]

MIPS TECHNOLOGIES

Clip Check (Compare)

Is the vertex within the viewing pyramid?

$$\left.\begin{array}{l} x >= -w,\ x <= w \\ y >= -w,\ y <= w \\ z >= -w,\ z <= w \end{array}\right\}$$ Set 6 Condition Code (CC) bits

Observation : Can use magnitude compares.

$$\left.\begin{array}{l} |x| <= |w| \\ |y| <= |w| \\ |z| <= |w| \end{array}\right\}$$ Set only 3 CC bits

CABS: for Clip Check Compare

Transformed [w | z] [y | x] in FP registers

PUU.PS to get [w | w]

NEG.PS to get [-w | -w]
C.NGE.PS !(y >= -w)? !(x >= -w)?
C.NGE.S !(z >= -w)?
C.LE.PS y<=w? x<=w?
C.LE.S z<=w?

Replace with absolute compares

CABS.LE.PS |y|<=|w|?, |x|<=|w|?
CABS.LE.PS |w|<=|w|?, |z|<=|w|?

MIPS TECHNOLOGIES

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.

MIPS TECHNOLOGIES

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)

MIPS TECHNOLOGIES

Perspective Division and Normalization

- In MIPS® IV 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
- S, D, and PS format

MIPS TECHNOLOGIES

Other Instruction Sets

- 3DNow!™ Technology -- enhance 3D graphics and multimedia
  - 2-packed FP SIMD (PS)
  - PFACC - accumulate
  - PFRCP, PFRCPIT1, PFRCPIT2 - reciprocal
  - PFRSQRT, PFRSQIT1 - reciprocal square root
  - PF2ID, PI2FD - convert
- AltiVec™ Technology
  - 4 SIMD (32-bits)
  - vrefp, vnmsubfp, vmaddfp - reciprocal
  - vrsqrtefp, etc - reciprocal square root
  - vcmpbfp - bounds compare
  - vcfsx, vctsxs - convert

MIPS TECHNOLOGIES

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

MIPS TECHNOLOGIES

Implementation Cost

- Die Area (of the Ruby processor)
  - Implementation of PS adds 6-7% 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 - 2x64 word lookup tables contribute to most of the 3% die area increase.

MIPS TECHNOLOGIES

Performance: Number of Instructions

Note: Inner-loop instructions/vertex = cycles/vertex

	No PS + No MIPS-3D	PS + No MIPS-3D	PS + MIPS-3D
Transform (matrix transform + clip + perspective divide)	29	28	20
Transform + complex lighting	90	67	49

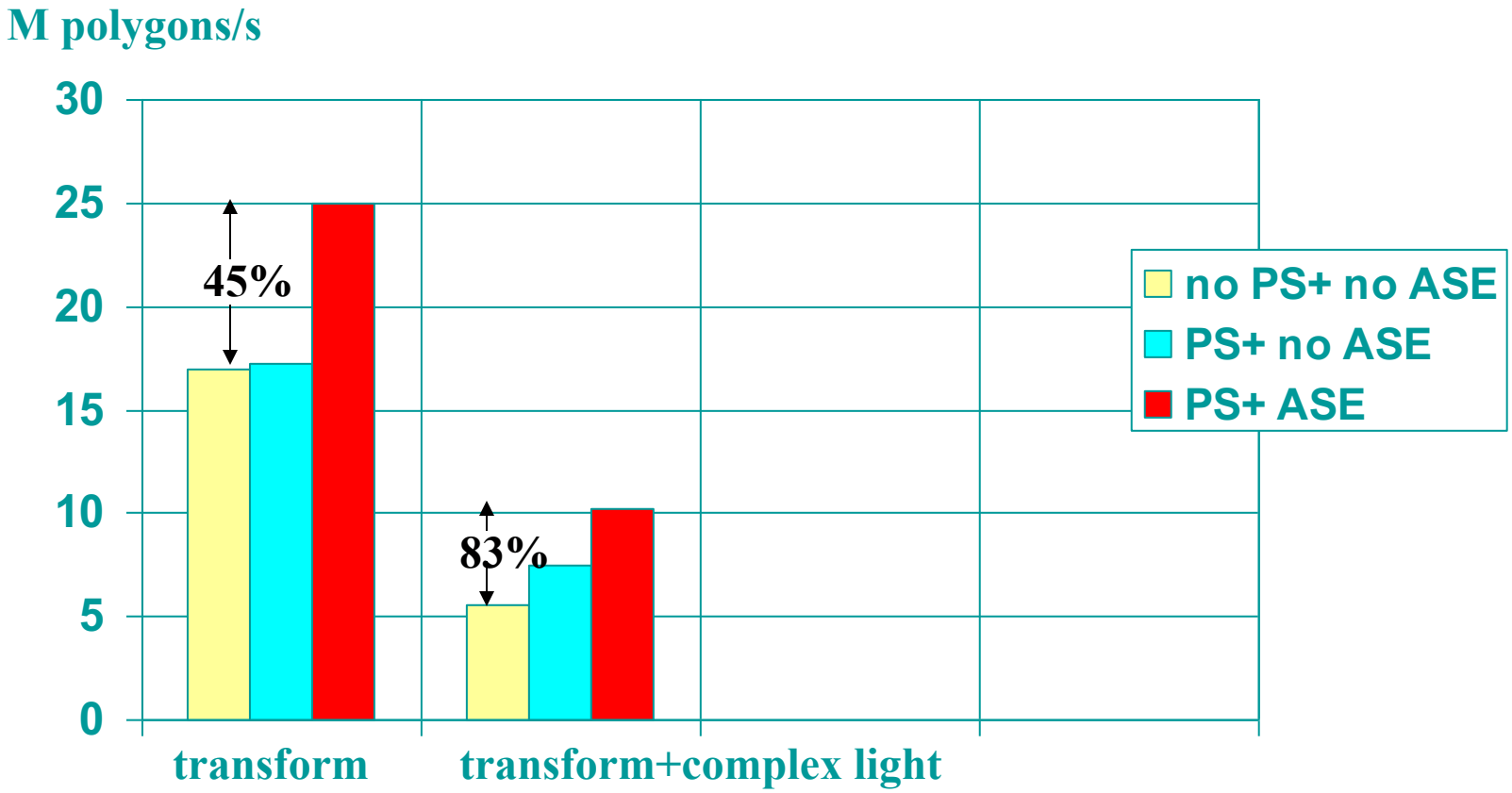
MIPS TECHNOLOGIES

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



MIPS TECHNOLOGIES

Summary

- MIPS-3D™ ASE adds thirteen instructions to the current MIPS64™ architecture
- Low cost (3% of FP die area)
- Increases polygons/sec count by 45% for the transform code to obtain 25 M polygons/s
- Increases polygons/sec count by 83% for transform together with complex lighting to obtain 10 M polygons/s

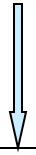
Appendix:Vertex Transformation Code

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

MUL.PS FP10,FP8,FP0	FP10 <-- m1*y | m0*x
MUL.PS FP11,FP8,FP2	FP11 <-- m5*y | m4*x
MUL.PS FP12,FP8,FP4	FP12 <-- m9*y | m8*x
MUL.PS FP13,FP8,FP6	FP13 <-- m13*y | m12*x
MADD.PS FP10,FP10,FP9,FP1	FP10 <-- m3*w+m1*y | m2*z+m0*x
MADD.PS FP11,FP11,FP9,FP3	FP11 <-- m7*w+m5*y | m6*z+m4*x
MADD.PS FP12,FP12,FP9,FP5	FP12 <-- m11*w+m9*y | m10*z+m8*x
MADD.PS FP13,FP13,FP9,FP7	FP13 <-- m15*w+m13*y | m14*z+m12*x

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

Replace with



ADDR.PS FP8,FP11,FP10	FP8 <-- m4x+m5y+m6z+m7w | m0x+m1y+m2z+m3w
ADDR.PS FP9,FP13,FP12	FP9 <-- m12x+m13y+m14z+m15w | m8x+m9y+m10z+m11w

MIPS TECHNOLOGIES

Appendix:The 13 MIPS-3D™ ASE Instructions

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