MS Thesis

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

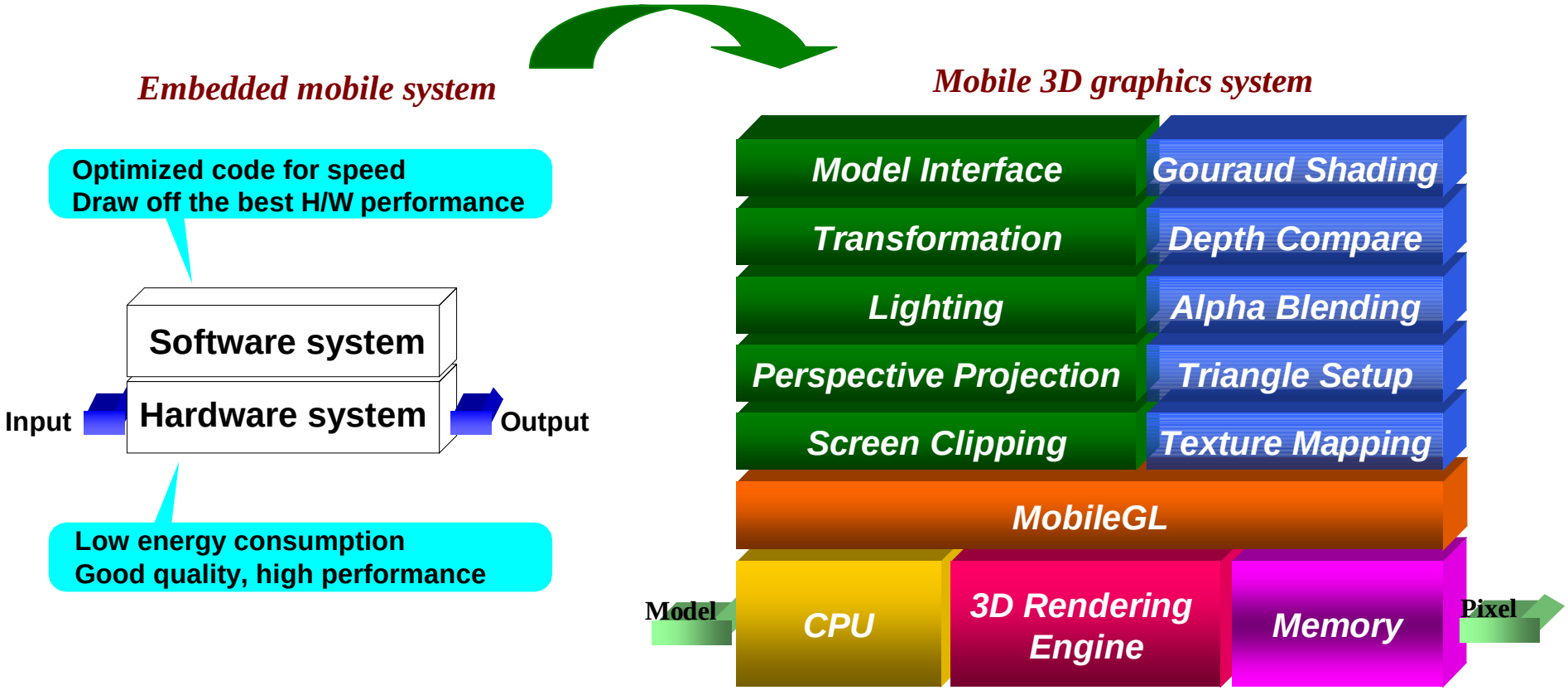
Min-wuk Lee

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

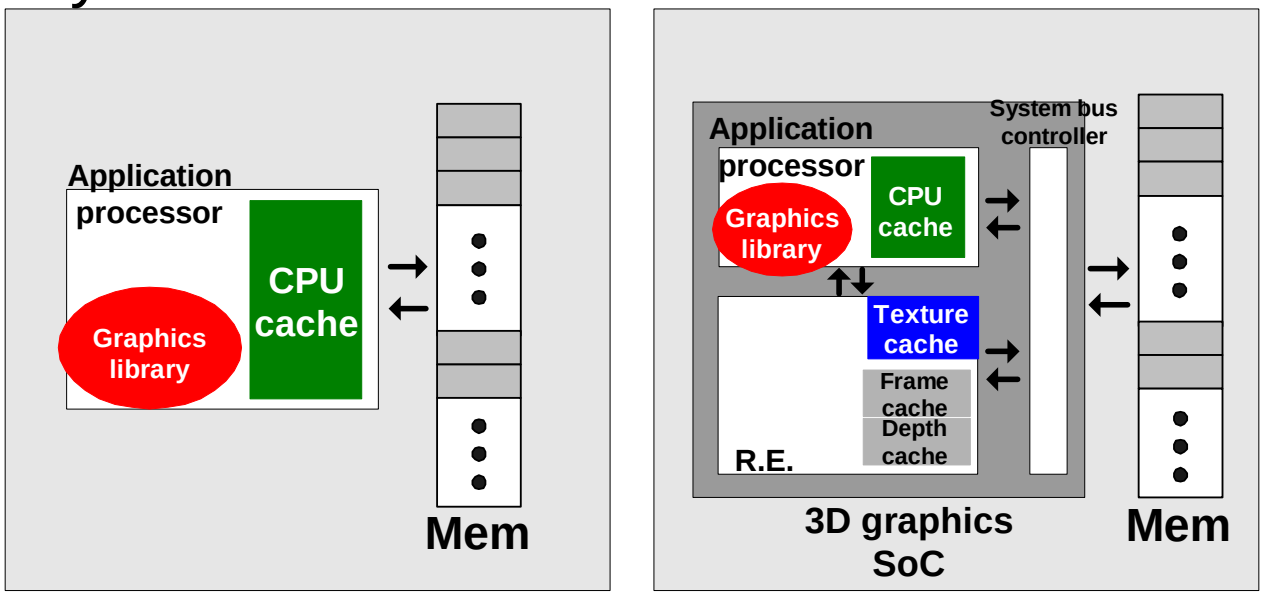
Introduction(1/2)



- Performance-energy co-optimization for mobile 3D graphics
  - Software system : High speed graphics library (MobileGL)
  - Hardware system : Energy-efficient Cache architecture

Introduction(2/2)

- 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

[1] : K.Yoshida, Consumer Electronics, IEEE Transactions on,1998.

[2] : Ziyad s. Hakura, ISCA ,1997.

Motivation

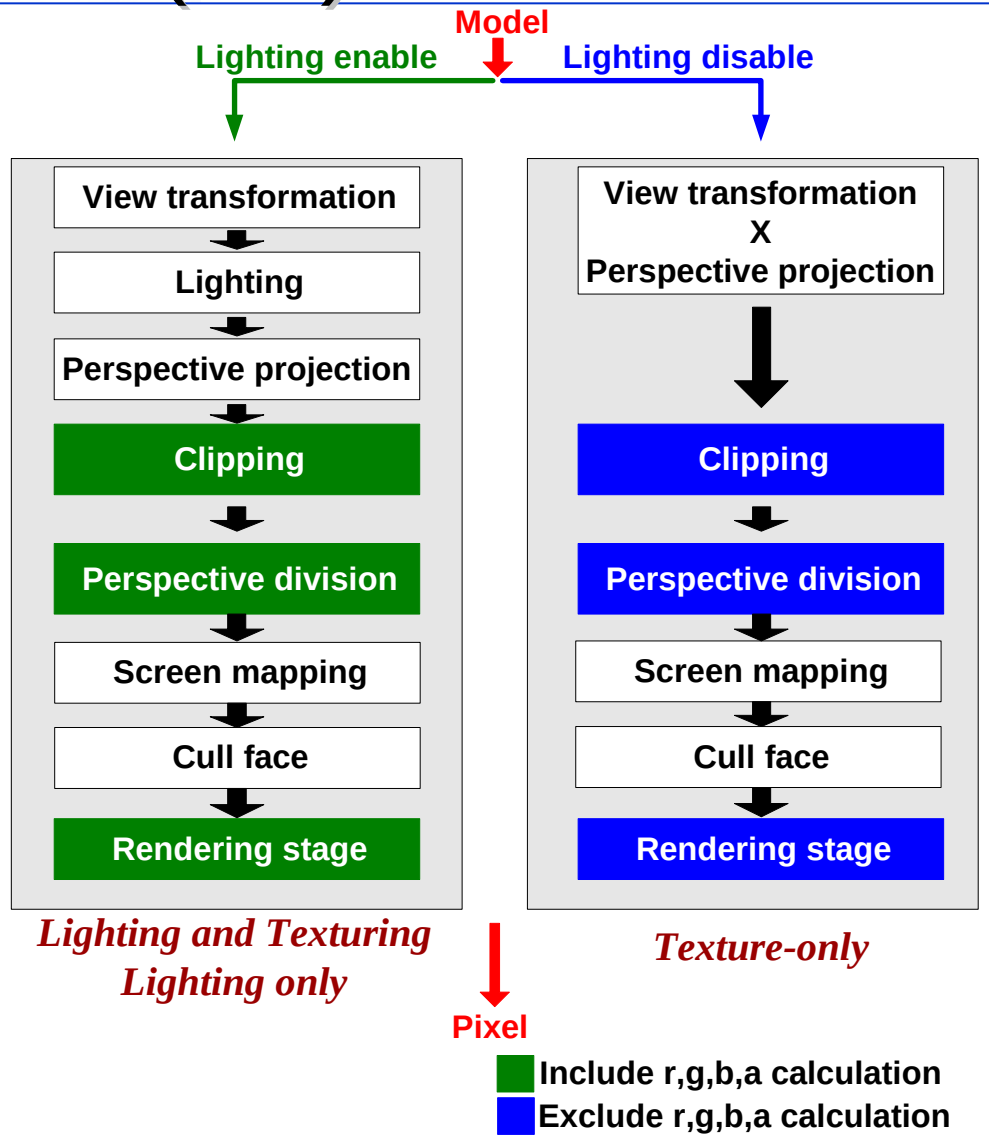
- Graphics library of this work
  - Extended operation
    - Lighting, Texturing, Alpha blending, Face culling, etc.
  - Optimization of memory transaction
    - 3D graphics characteristic
    - Analysis with cache system
- Texture cache of this work
  - Energy-efficient texture cache in embedded system
    - With negligible performance degradation

Outline

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

MobileGL(1/6)

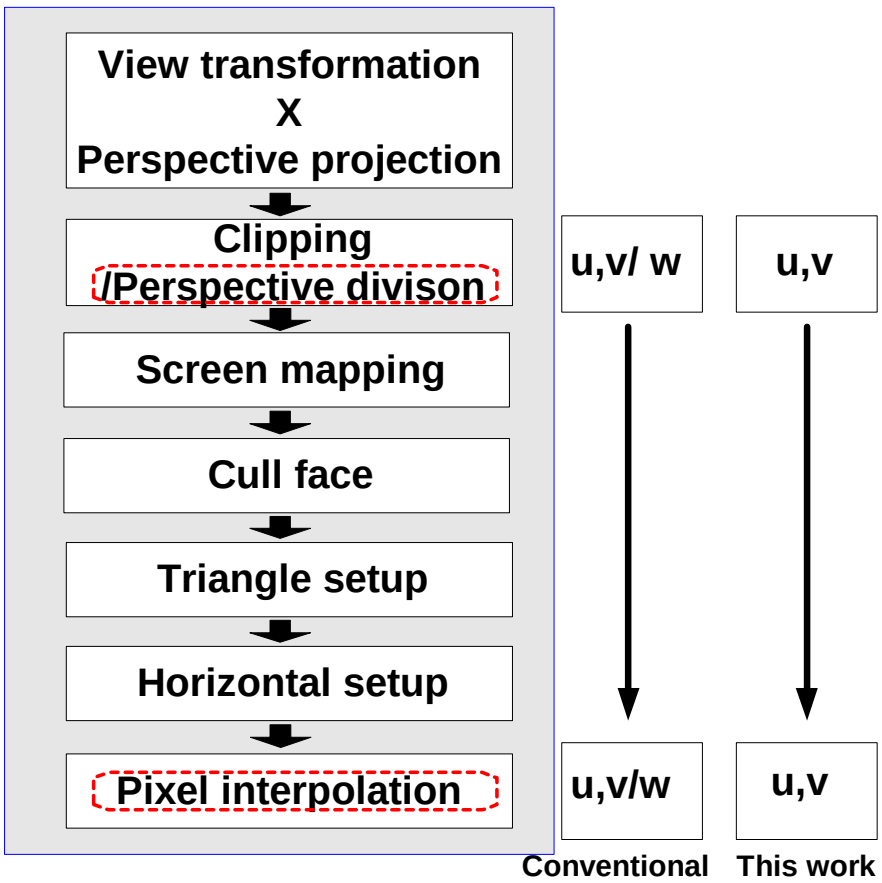
- Mobile 3D graphics library
- A fixed-point arithmetic
  - 32bit integer
- Optimized memory transaction
  - To reduce instruction and data traffic
- Selective pipeline
  - Applications
  - To reduce branch
  - 45KB total code size

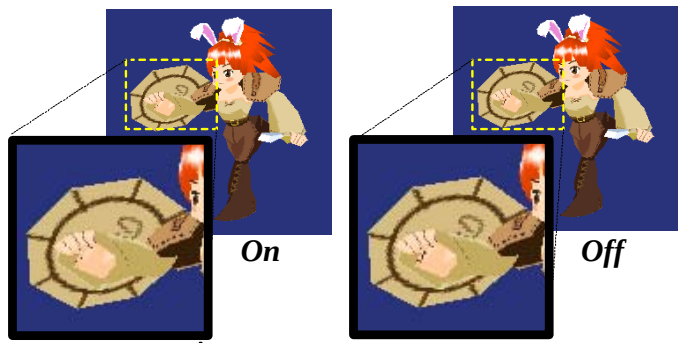


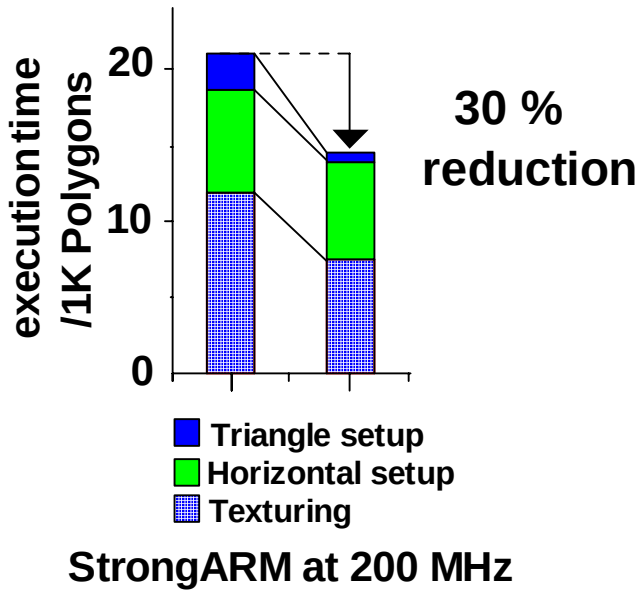
MobileGL block diagram

MobileGL(2/6)

- Disabling option for perspective correction of texture address
  - Due to small screen size
  - Trade-off between correctness and speedup

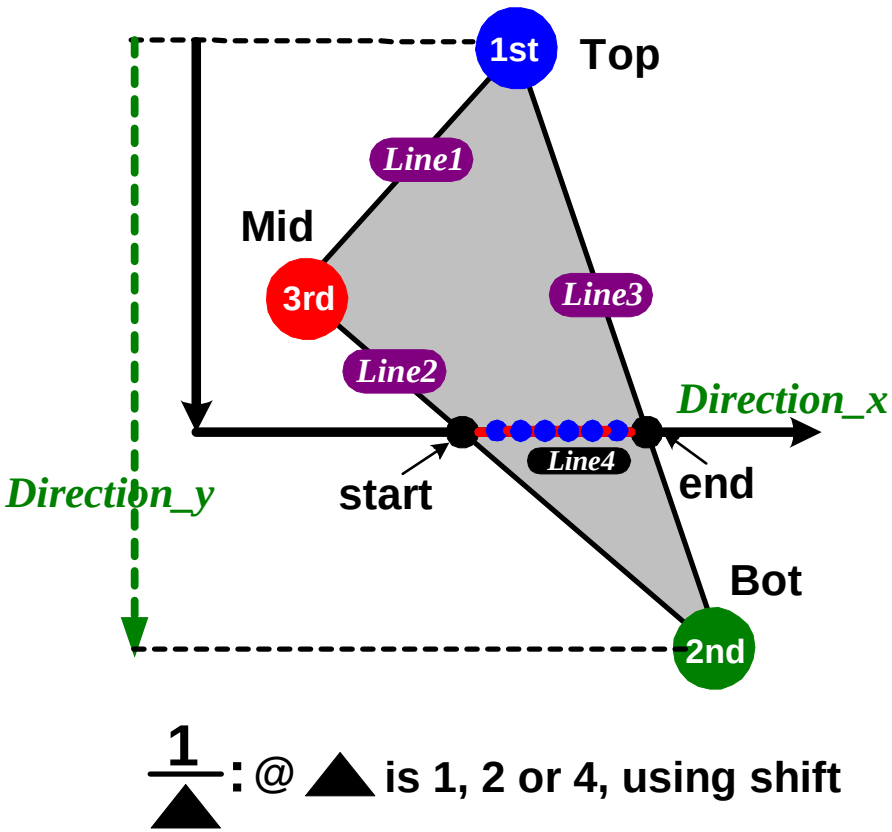




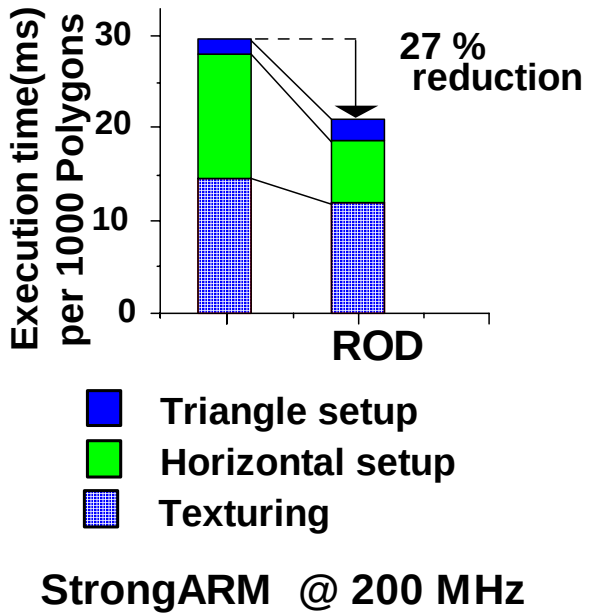


MobileGL(3/6)

- Division reduction in interpolation
  - Use shift instead of reciprocal
  - High probability of 1,2 or 4 in denominator value



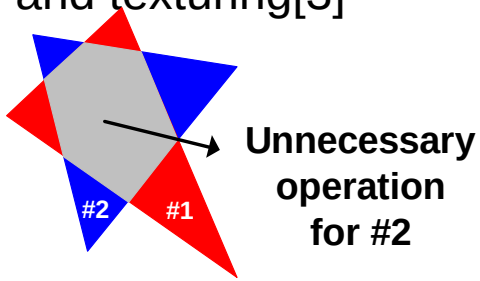
$\frac{1}{\blacktriangle}$: @ ▲ is 1, 2 or 4, using shift

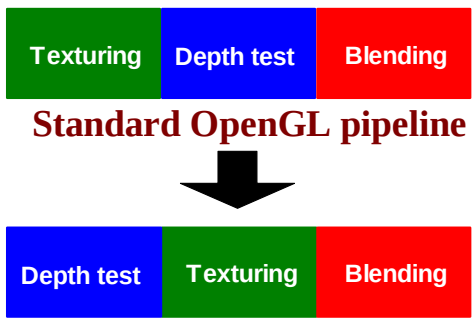


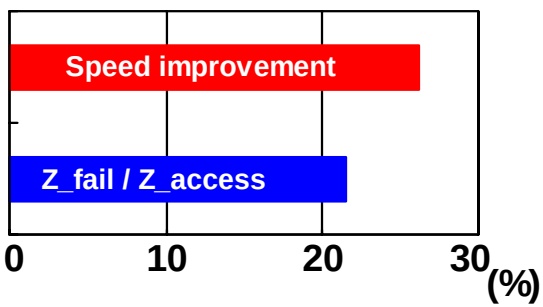
StrongARM @ 200 MHz

MobileGL(4/6)

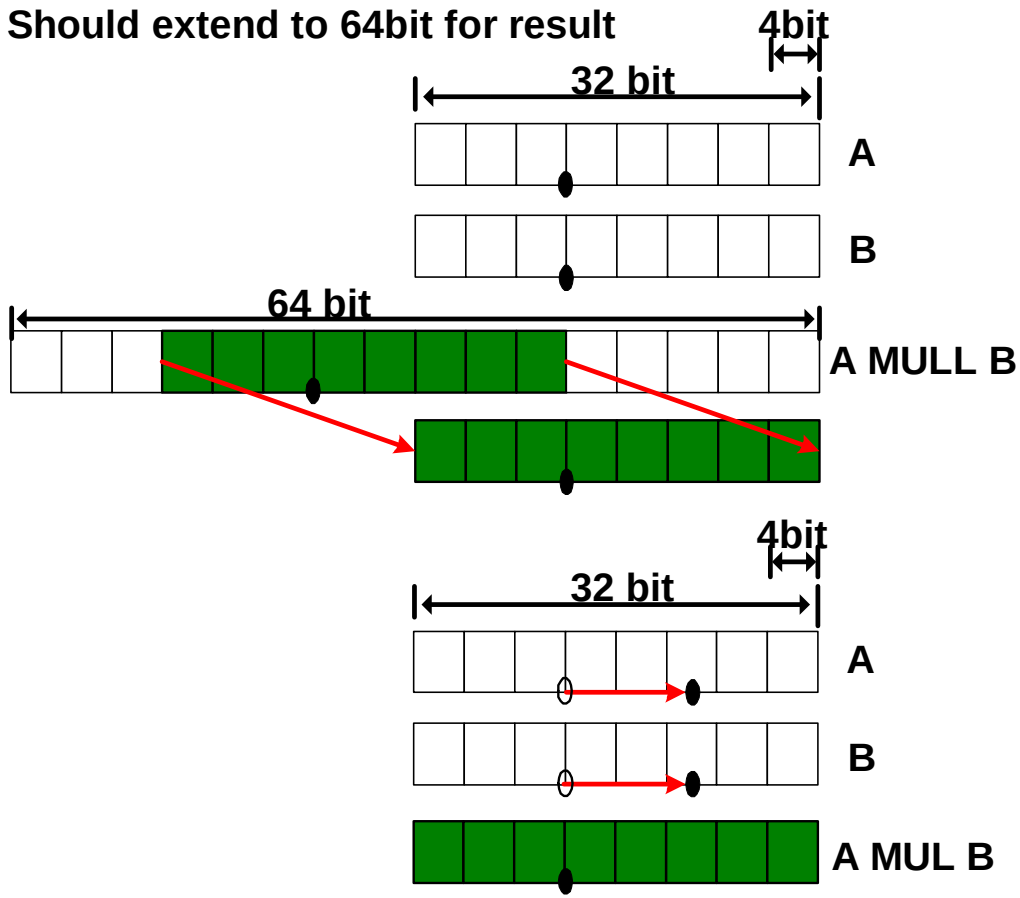
- Z comparison in advance
  - To avoid unnecessary shading and texturing[3]







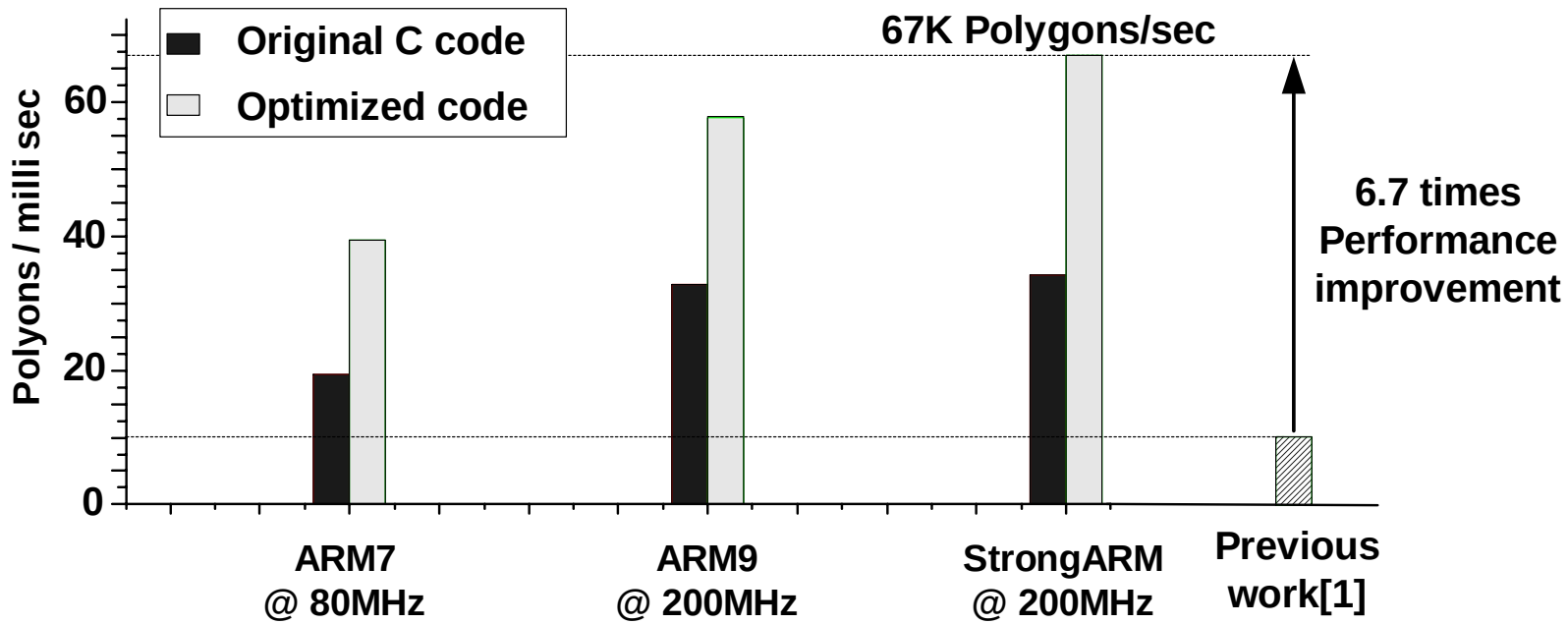
- Selective precision of matrix multiplication
  - 67% improvement @geometry stage



[3] : Ramchan Woo, ISSCC 2003

MobileGL(5/6)

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



MobileGL(6/6)

- Implementation result

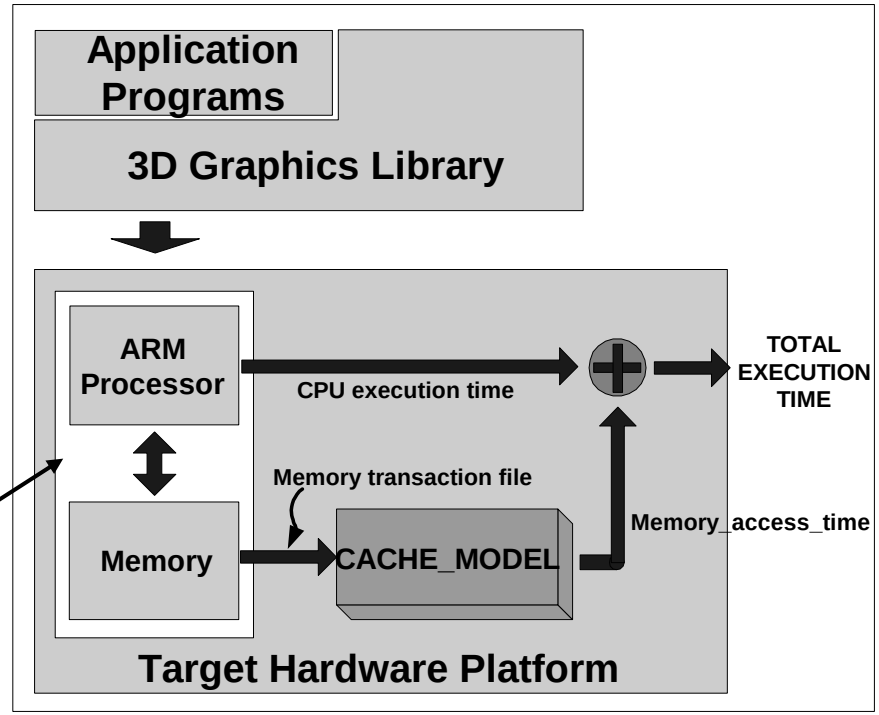
Outline

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

Energy-efficient CPU cache (1/4)

- Simulation environment

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

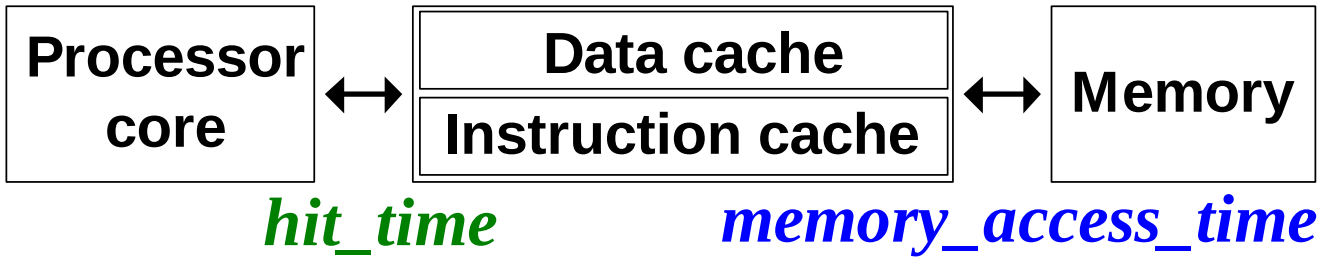


$$T_{exe\_total} = T_{exe\_CPU} + memory\_access\_time \quad (3)$$

$$T_{exe\_CPU} = \sum_{K=1}^{instruction\_counts} T_{exe\_instruction} \quad (1) + \sum_{K=1}^{cache\_hit\_counts} CPU\_cycle \quad (2)$$

Energy-efficient CPU cache (2/4)

- Cache model : about execution time



$$hit\_time = clock\_period_{core}$$

$$memory\_access\_time \begin{cases} t_{RCD} + CAS\_latency + clock\_period_{mem} \cdots (non\_squential\_access) \\ clock\_period_{mem} \cdots (burst\_access) \end{cases}$$

$$miss\_time \begin{cases} memory\_access\_time + hit\_time \ldots (read) \\ memory\_access\_time \cdots (write) \end{cases}$$

Energy-efficient CPU cache (3/4)

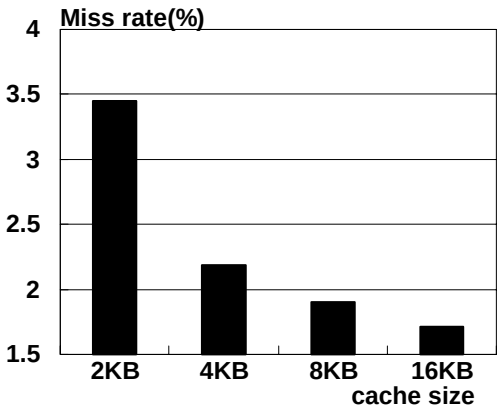
- Energy modeling
  - Tool and research documentation based model
    - Cache hit energy : from ***CACTI 3.0*** [4]
    - Cache miss energy : from Power & Energy Characterization of the ***Itsy Pocket Computer*** by ***Compaq Western Research Laboratory***
      - ***4.70nJ / bus_clock*** [5], [6]

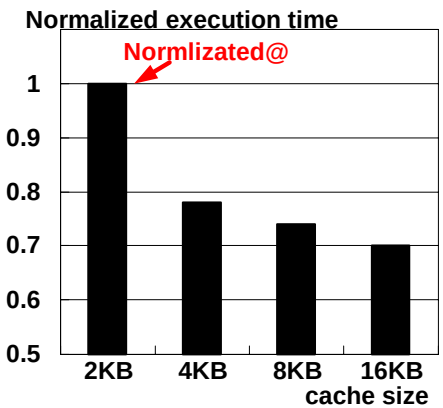
[4] : CACTI 3.0 : An integrated cache timing, power, and area model, Compaq Western Research Laboratory
[5] : Power and energy characterization of the Itsy pocket computer
[6] : A simulation framework for energy-consumption analysis of OS-driven embedded applications, TCAS 2003

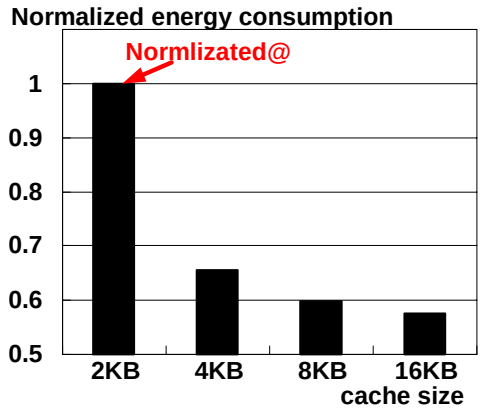
Energy-efficient CPU cache (4/4)

- Simulation results

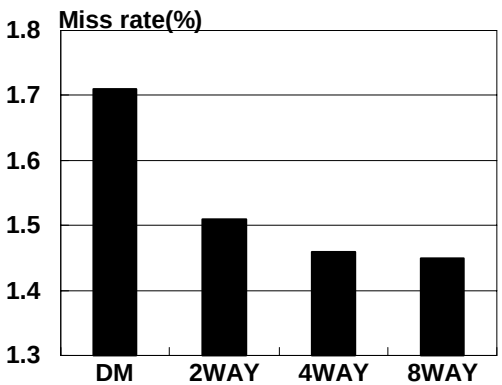
Direct mapped data cache, 8E/line (32B line size)

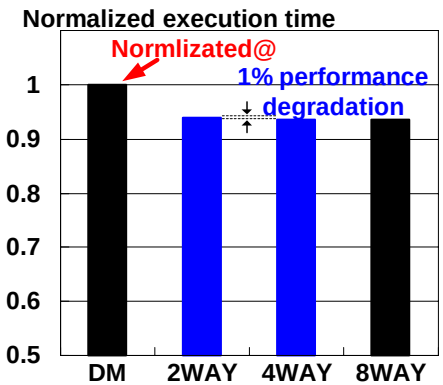


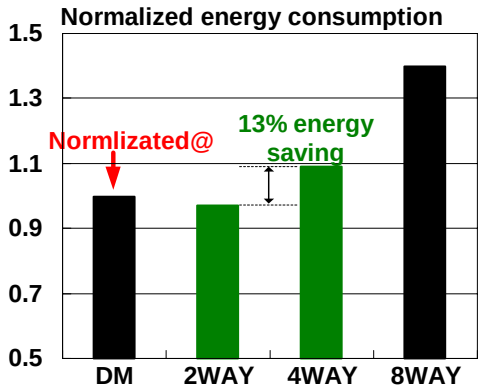




16KB data cache, 32B line size







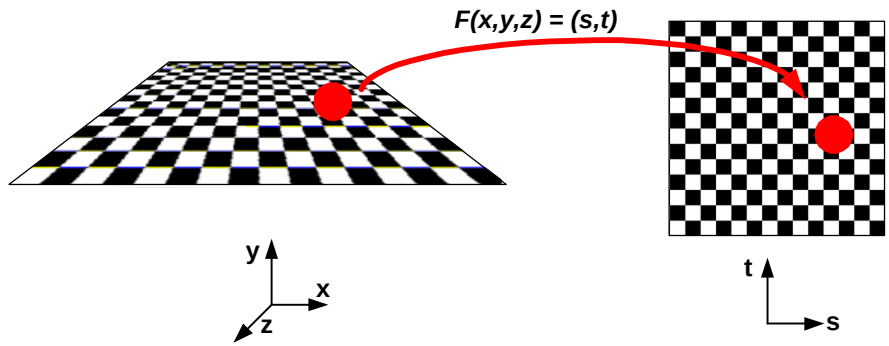
Using 2-way cache, 13% energy saving, 1% performance degradation compared with conventional 4-way cache

Outline

- 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 texles
    - MIPMAP
      - Image pyramid



2D texture diagram

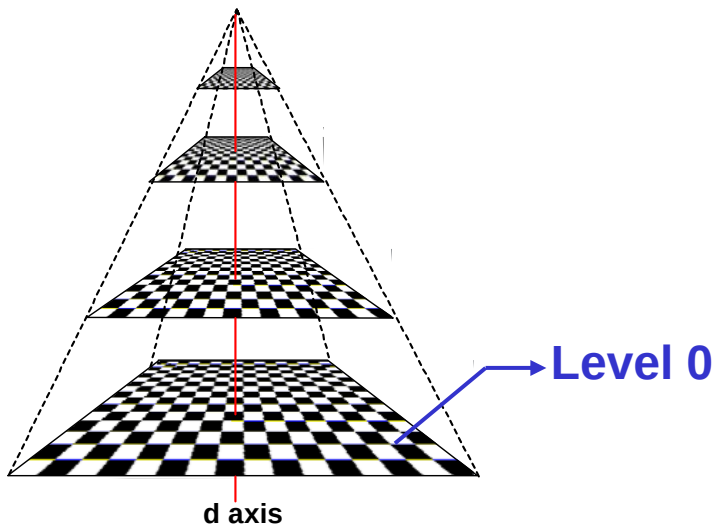
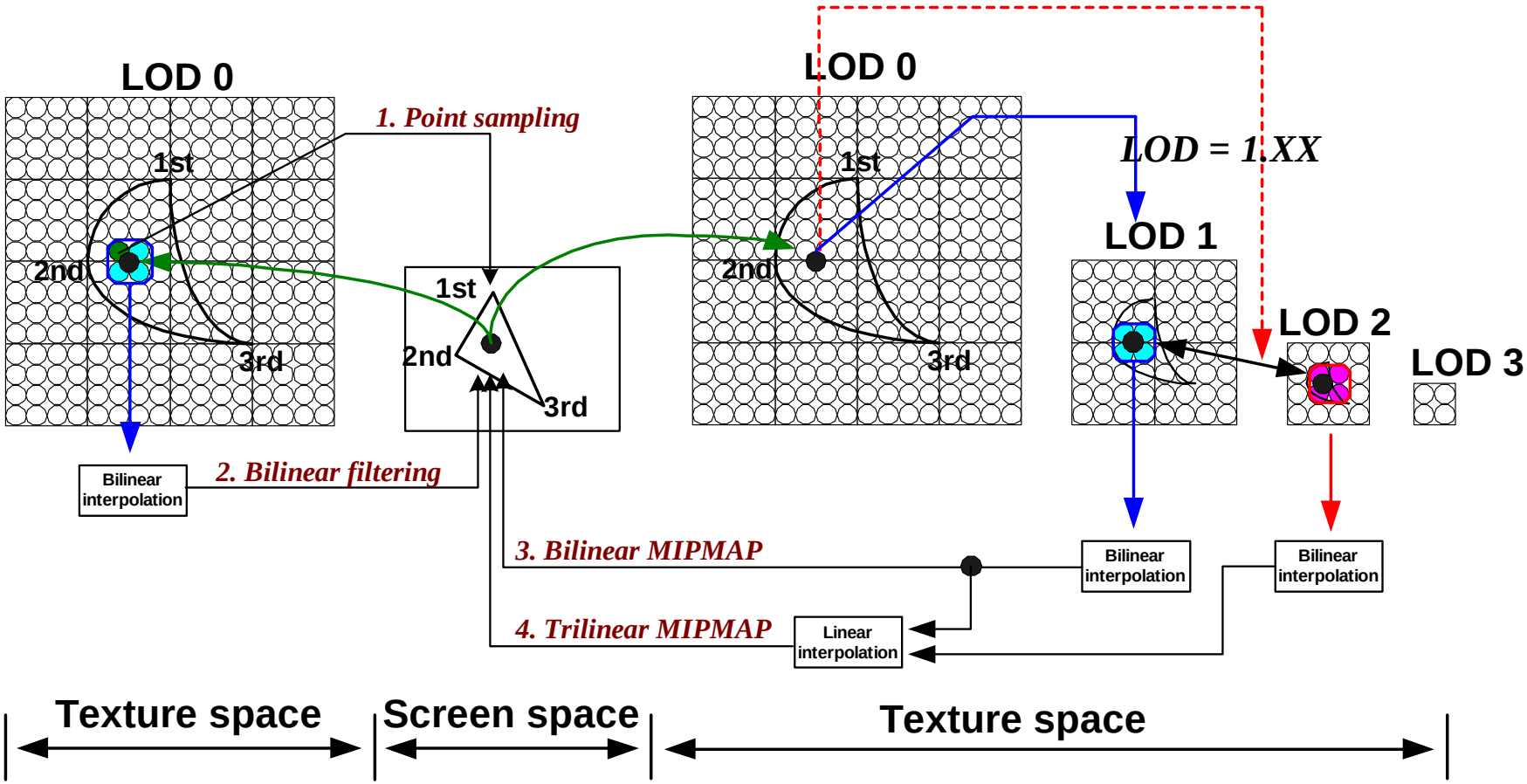


Image pyramid

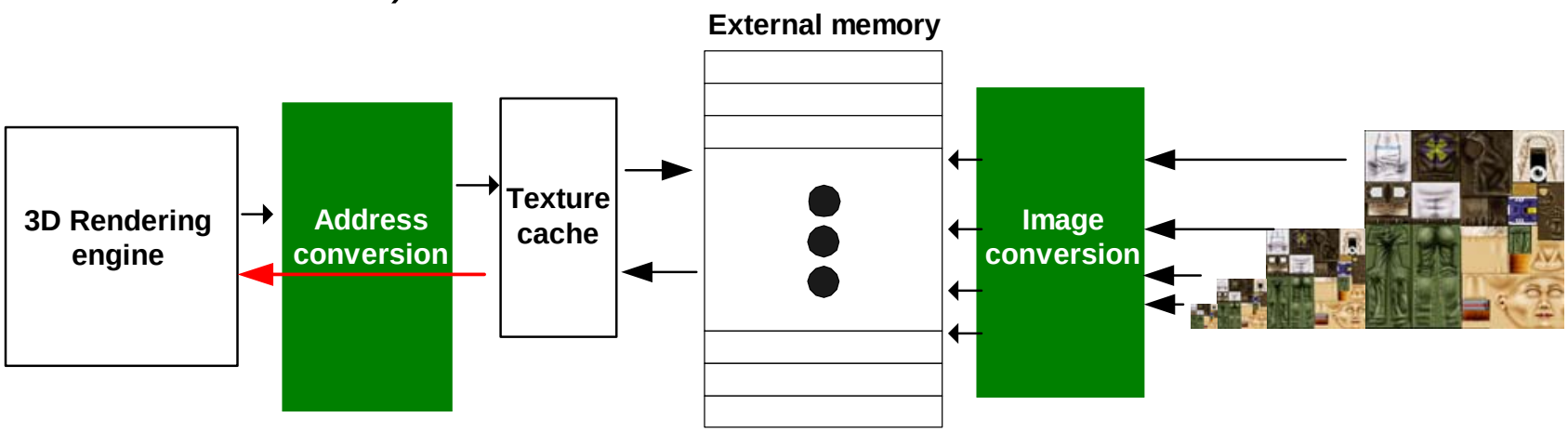
Energy-efficient texture cache (2/12)

- Texture filtering methods (Introduction)
  - Point sampling, Bilinear filtering, Bilinear MIPMAP, Trilinear MIPMAP



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)



Texture cache system

Energy-efficient texture cache (4/12)

- Simulation models

Tiny
6833 polygons

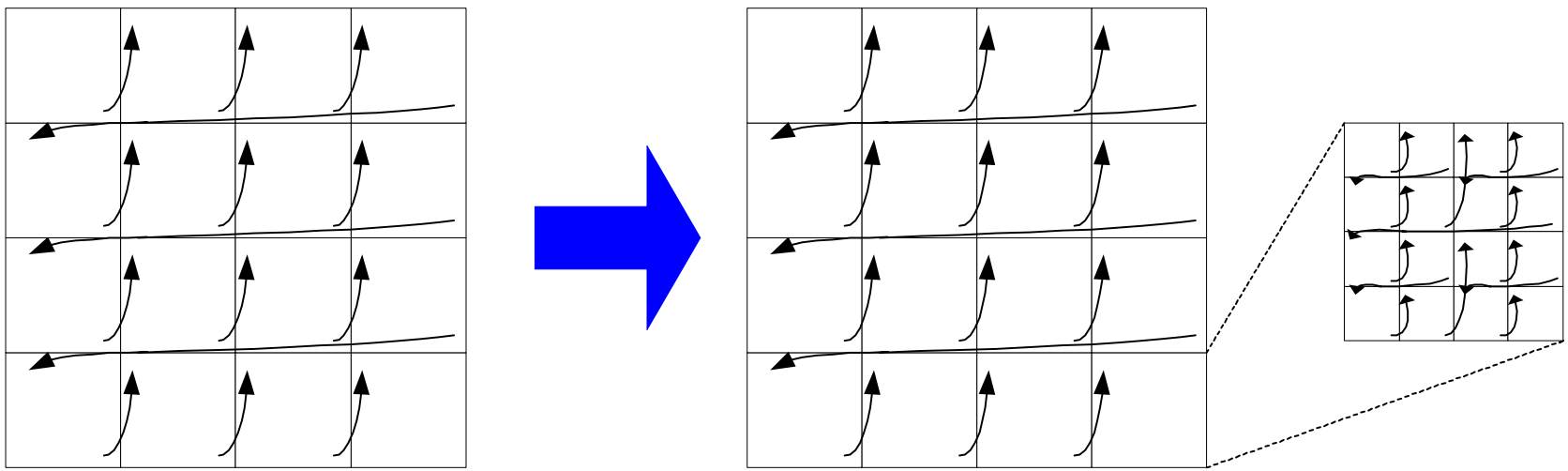
Stealth
542 polygons

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%)



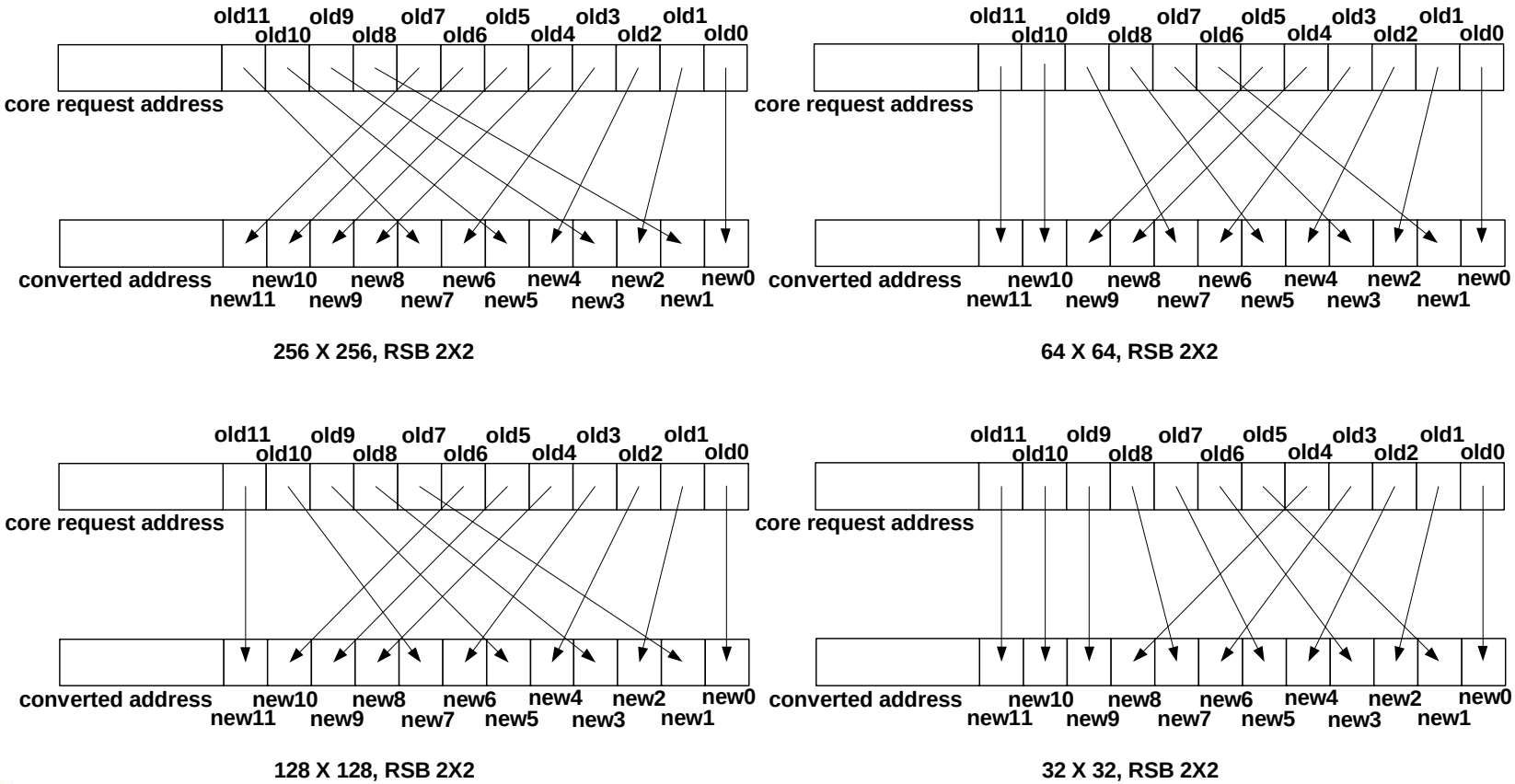
Blocked representation

Recursive Sub Block

Energy-efficient texture cache (6/12)

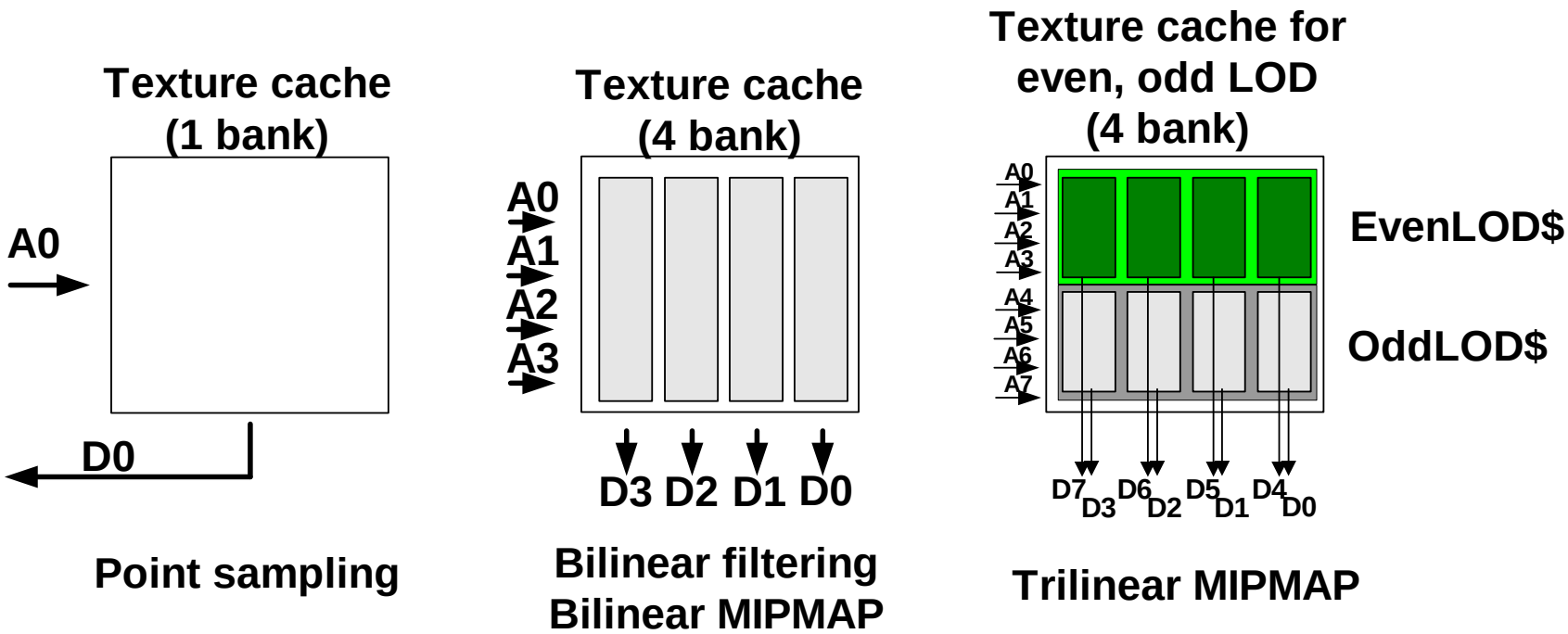
- Address conversion unit for RSB2X2
  - Use one-to-one correspondence and find rule
    - Hardware implementation : only thirteen 2:1mux in trilinear MIPMAP

Address conversion unit of this work : RSB 2X2



Energy-efficient texture cache (7/12)

- Texture cache model using bank interleaved

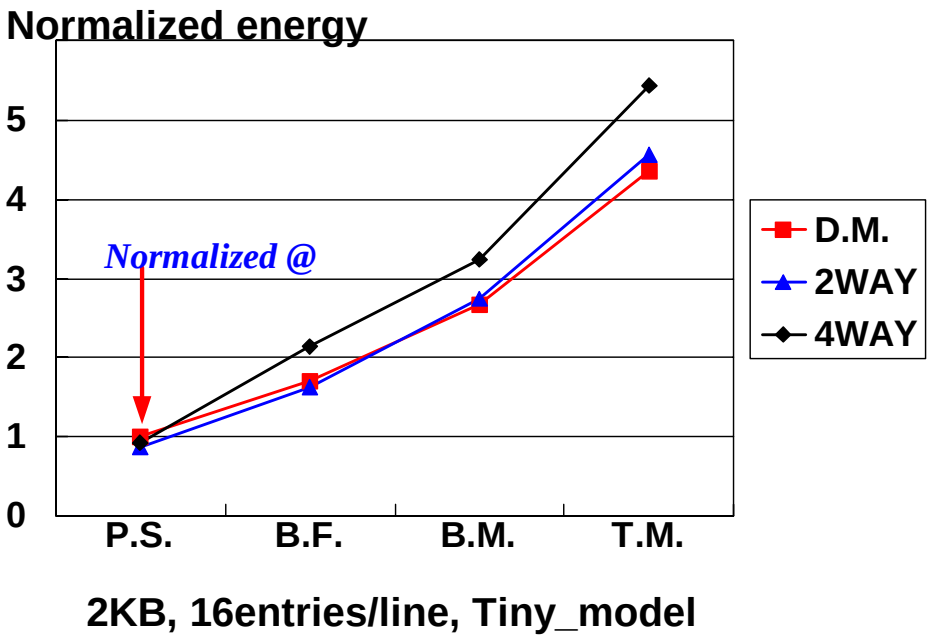


Morton order representation ← previous work

Proposed RSB2X2 also free from bank conflict

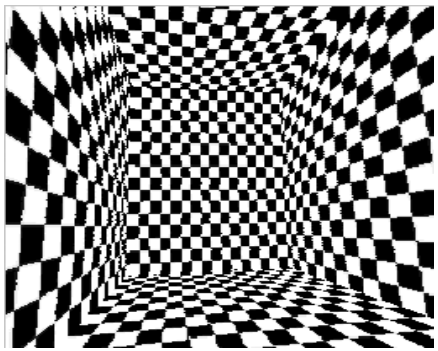
Energy-efficient texture cache (8/12)

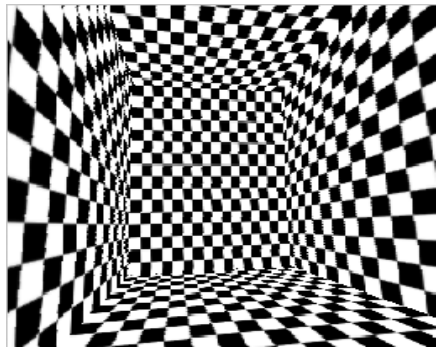
- Performance and Energy comparison between filtering method
  - Energy consumption, Execution time
    - Point sampling < Bilinear filtering < Bilinear MIPMAP < Trilinear MIPMAP
  - Trade off point : Image quality (aliasing criterion)

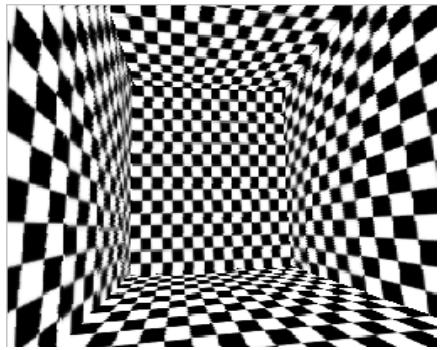


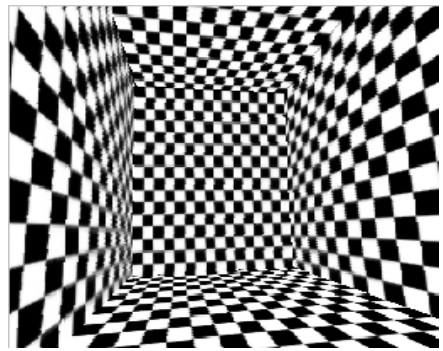
Energy-efficient texture cache (9/12)

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

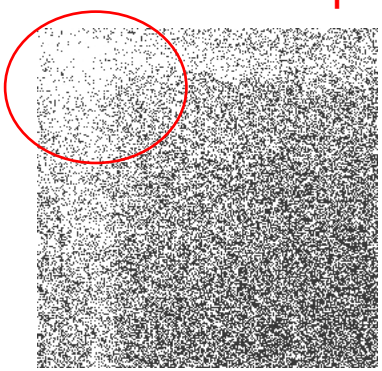

Point smapling

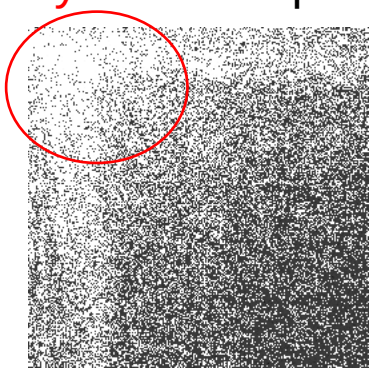

Bilinear filtering

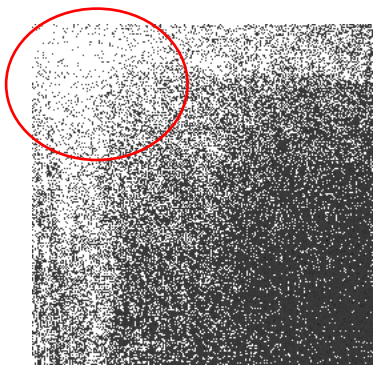

Bilinear mipmap

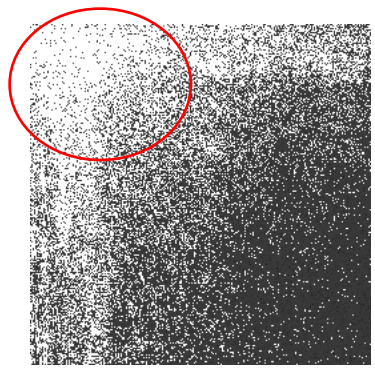

Trilinear mipmap

  - DCT analysis
    - Low frequency term in top-left


Point smapling


Bilinear filtering


Bilinear mipmap

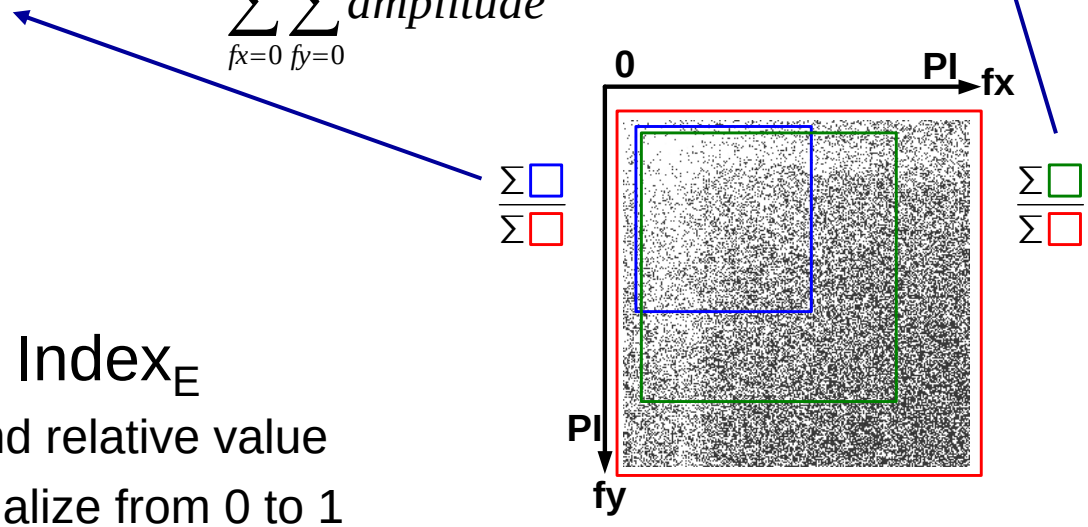

Trilinear mipmap

Energy-efficient texture cache (10/12)

- Image quality metric in terms of aliasing criterion

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

$$image\_quality\_0.75 = \frac{\sum_{fx=0}^{\frac{3\pi}{4}}\sum_{fy=0}^{\frac{3\pi}{4}} amplitude}{\sum_{fx=0}^{\pi}\sum_{fy=0}^{\pi} amplitude}$$



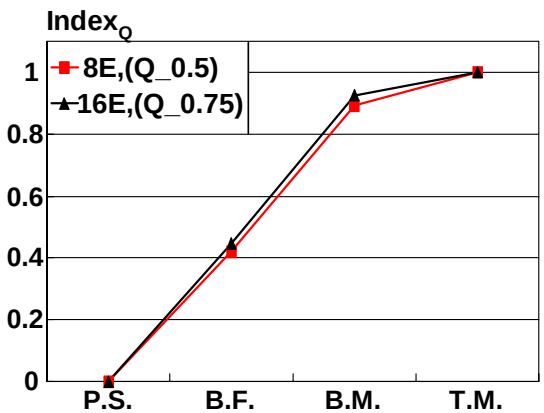
- $Index_Q$, $Index_E$
  - To find relative value
  - Normalize from 0 to 1

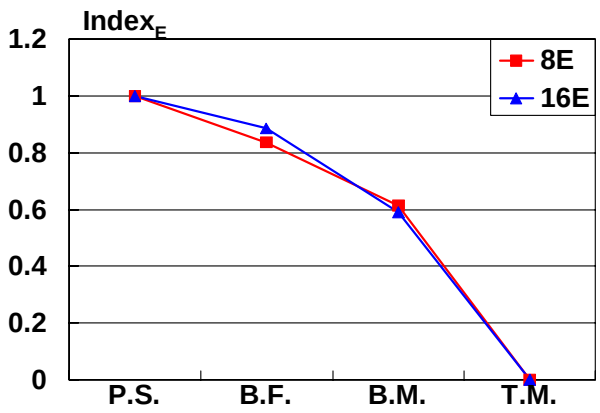
$$Index_Q = \frac{cur_Q - \min_Q}{\max_Q - \min_Q}$$

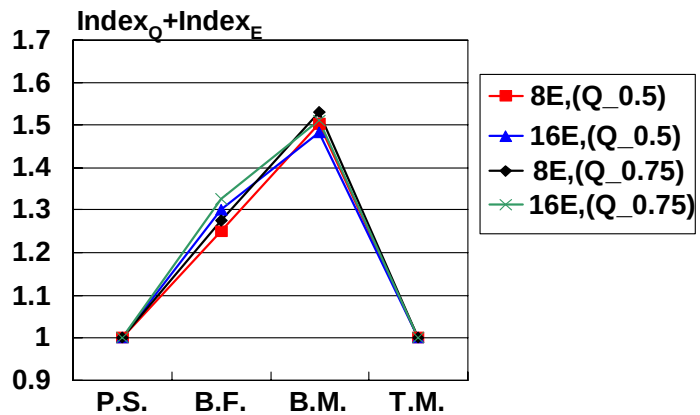
$$Index_E = \frac{\max_E - cur_E}{\max_E - \min_E}$$

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.



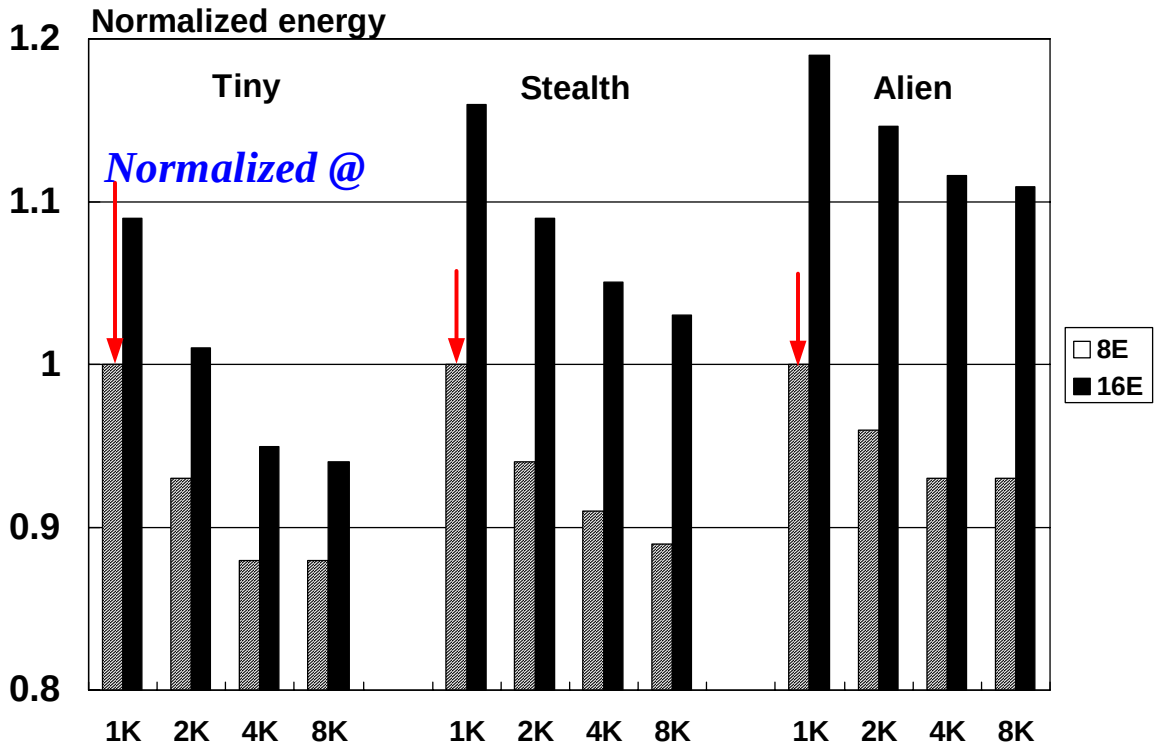




2-way set associative, 2KB texture cache

Energy-efficient texture cache (12/12)

- Simulation results



Energy comparison while changing cache size

@ 2-way, using bilinear MIPMAP

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

Outline

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

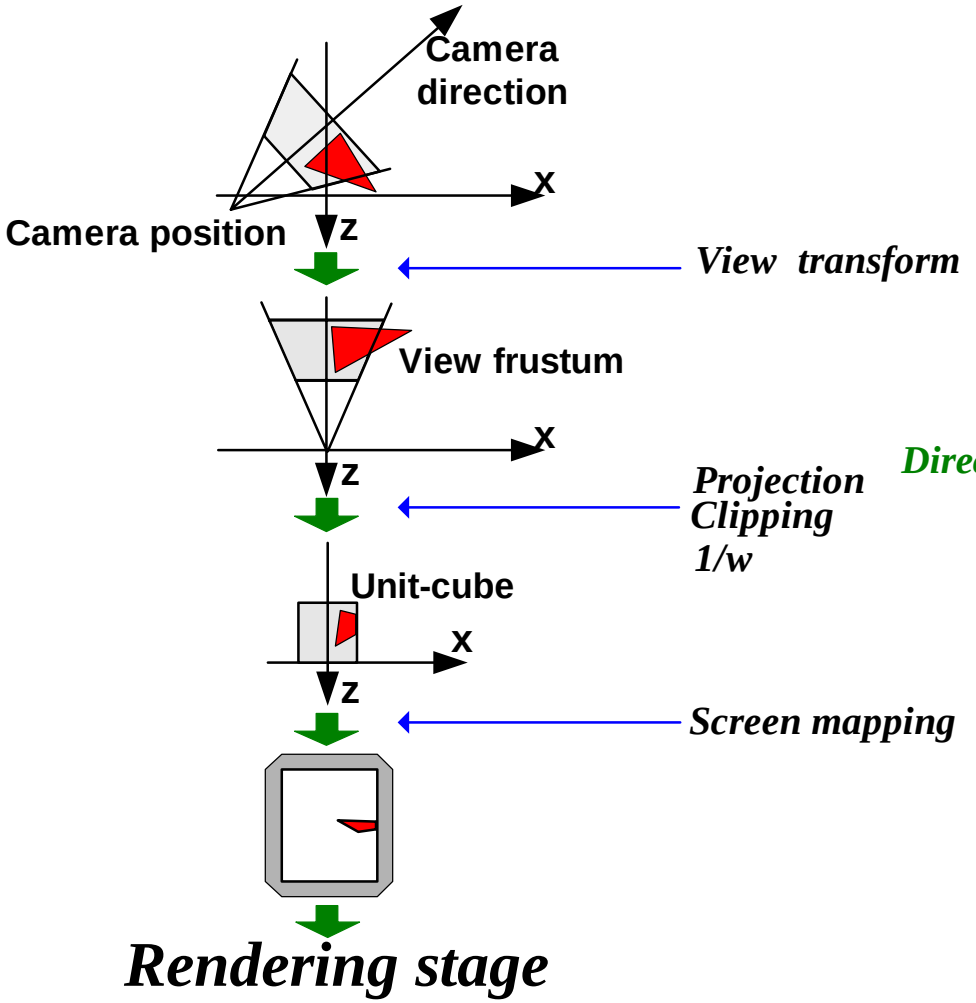
Conclusion

- 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

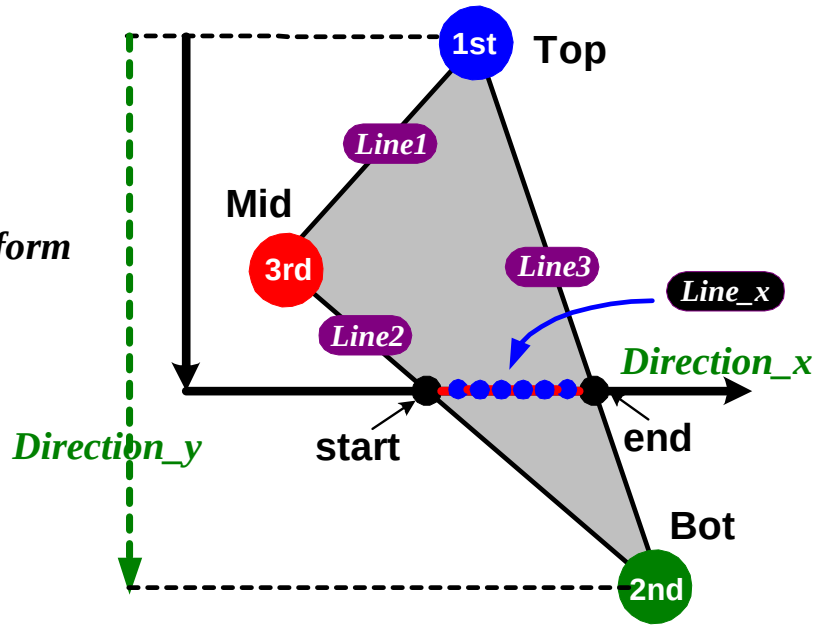
Supplemental Materials

Graphics pipeline

- Geometry stage



- Rendering stage



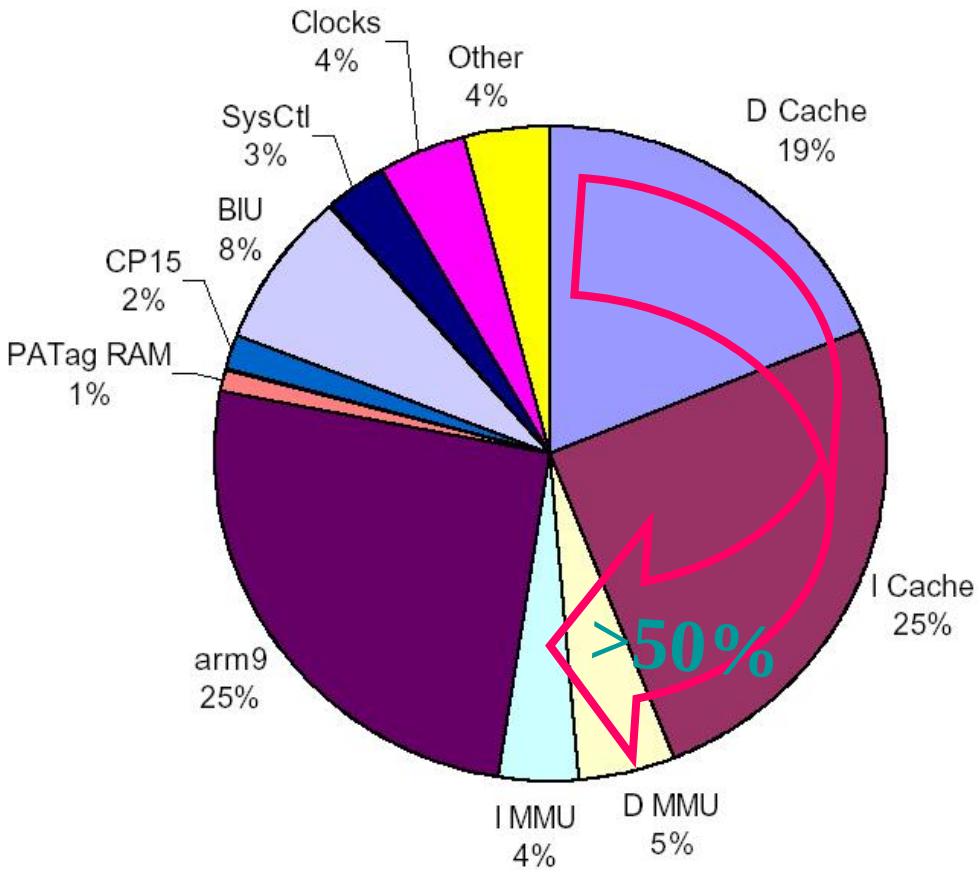
Triangle setup :
For line1, line2, line3 using1st, 2nd, 3rd

Horizontal setup :
For line_x using start, end

Pixel interpolation :
Each pixel shading, texturing

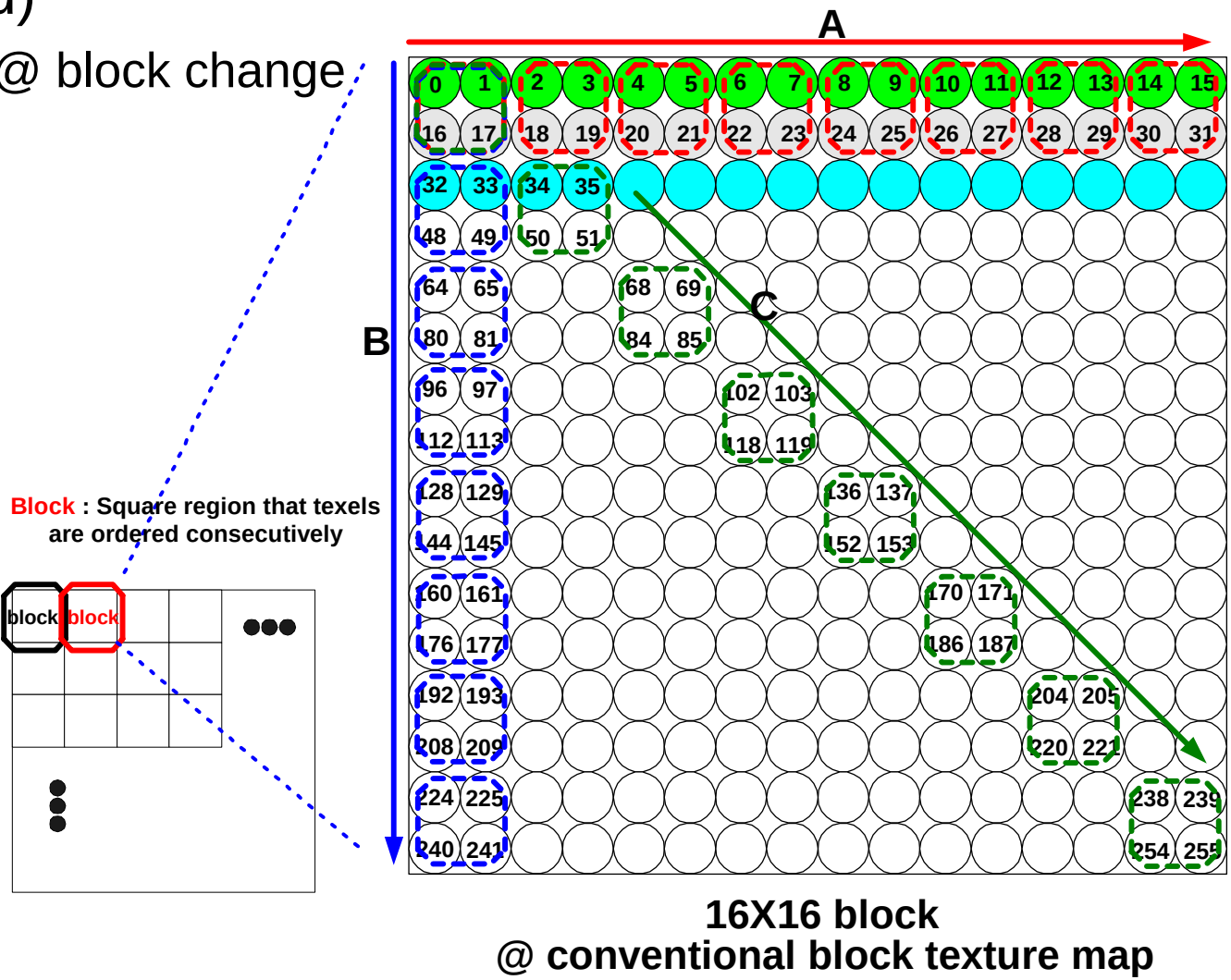
Energy portion of cache

- ARM920T and M*CORE : Caches consume 50% of total pr ocessor system power *(Segars 01,Lee et.al. 99)*



Blocked representation

- Conventional texture map representation (16X16blocked)
  - Conflict miss @ block change
    - A path : 2
    - B path : 16
    - C path : 16



Assumptions

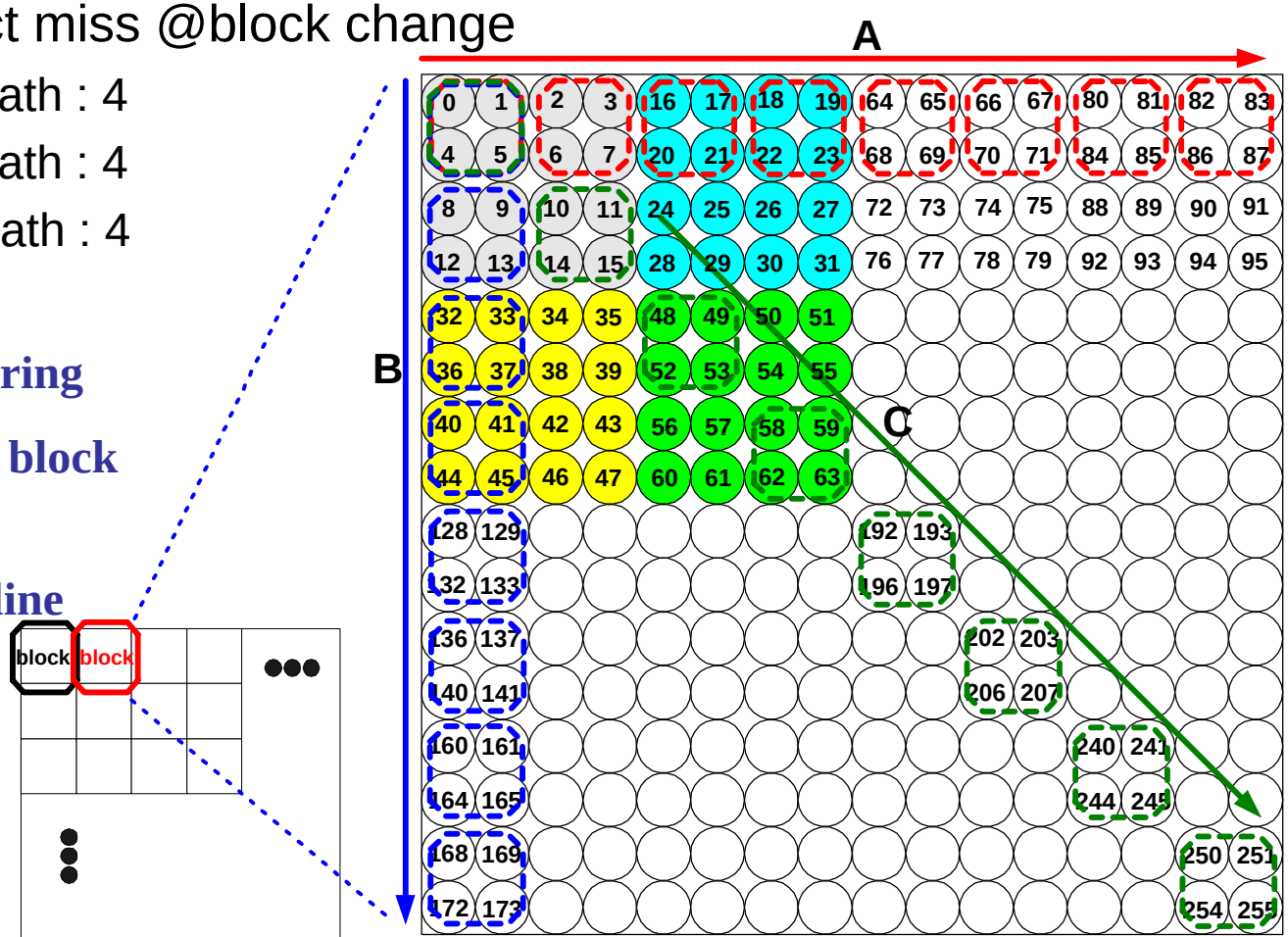
1. **Bilinear filtering**
2. **Cache size = block size**
3. **16entries / 1 line**

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

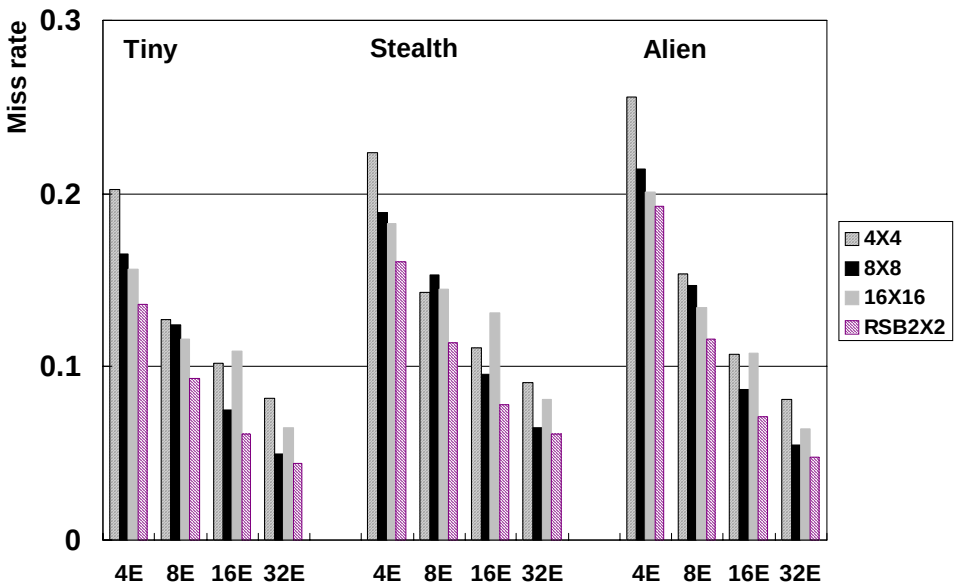
1. **Bilinear filtering**
2. **Cache size = block size**
3. **16 entries/ 1line**

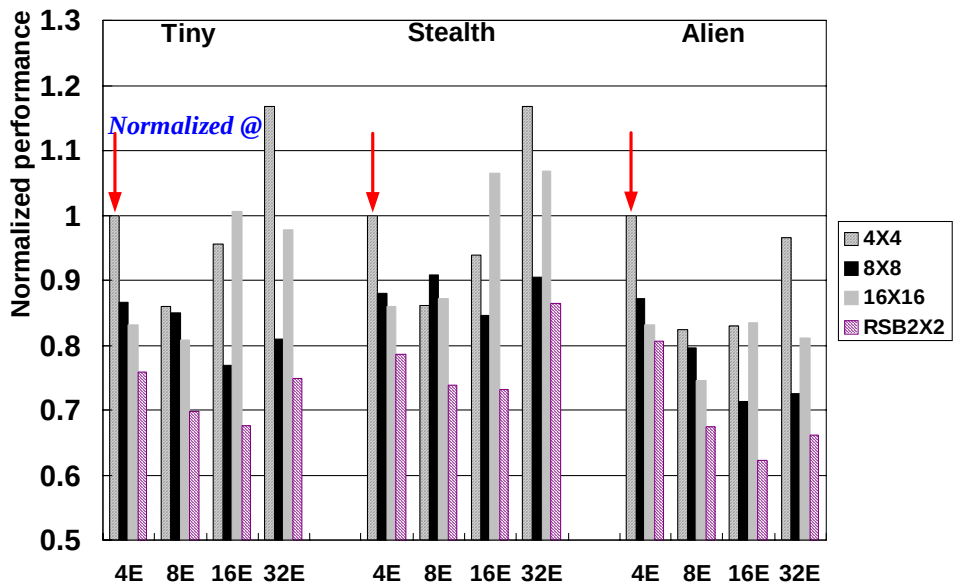


@ recursive sub-block 4X4 method

Simulation between representation methods

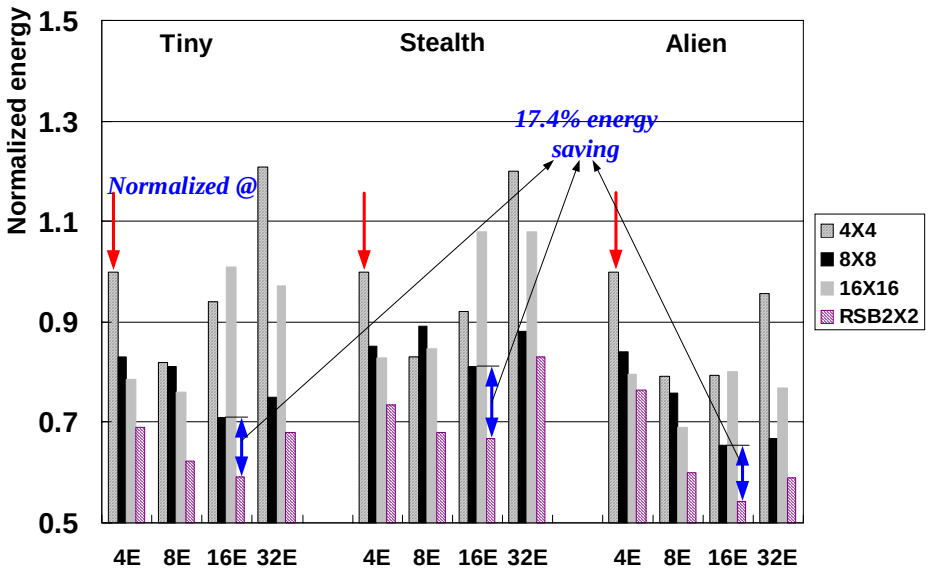
- Simulation results between texture representations
  - Bilinear filtering, 2-way, 1KB texture cache
  - 27% performance improvement in average
  - Low miss rate doesn't mean high performance
  - 8entries/line or 16entries/line shows good performance



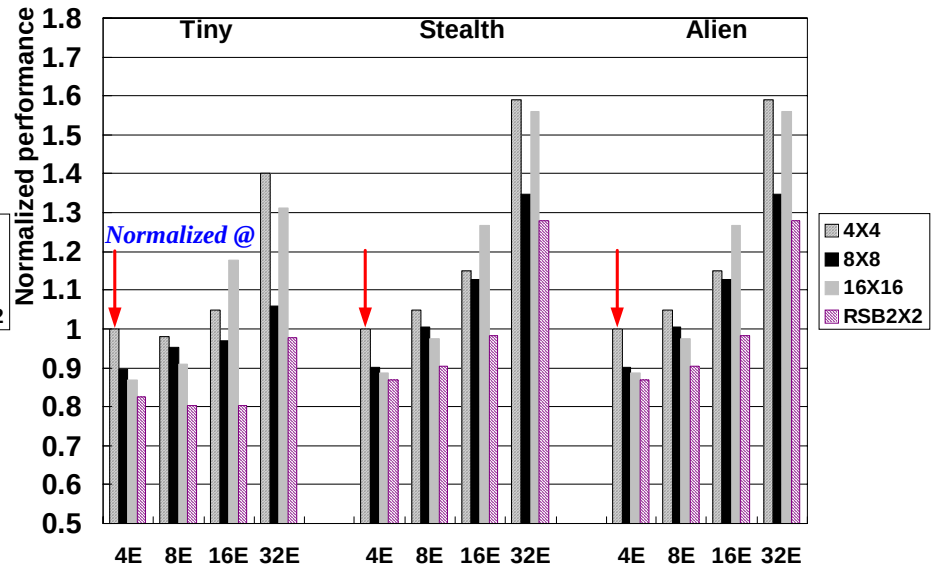


Simulation between representation methods

- @ 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



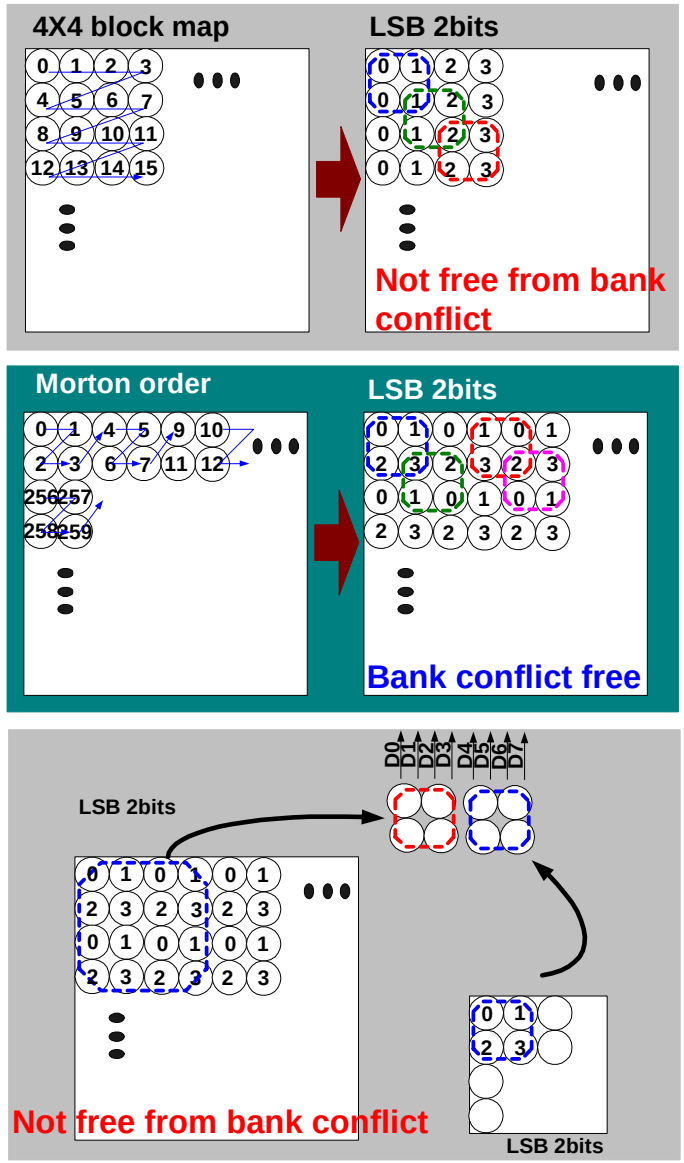
Bilinear filtering, 1KB cache, 2way



Point sampling, 1KB cache, 2way

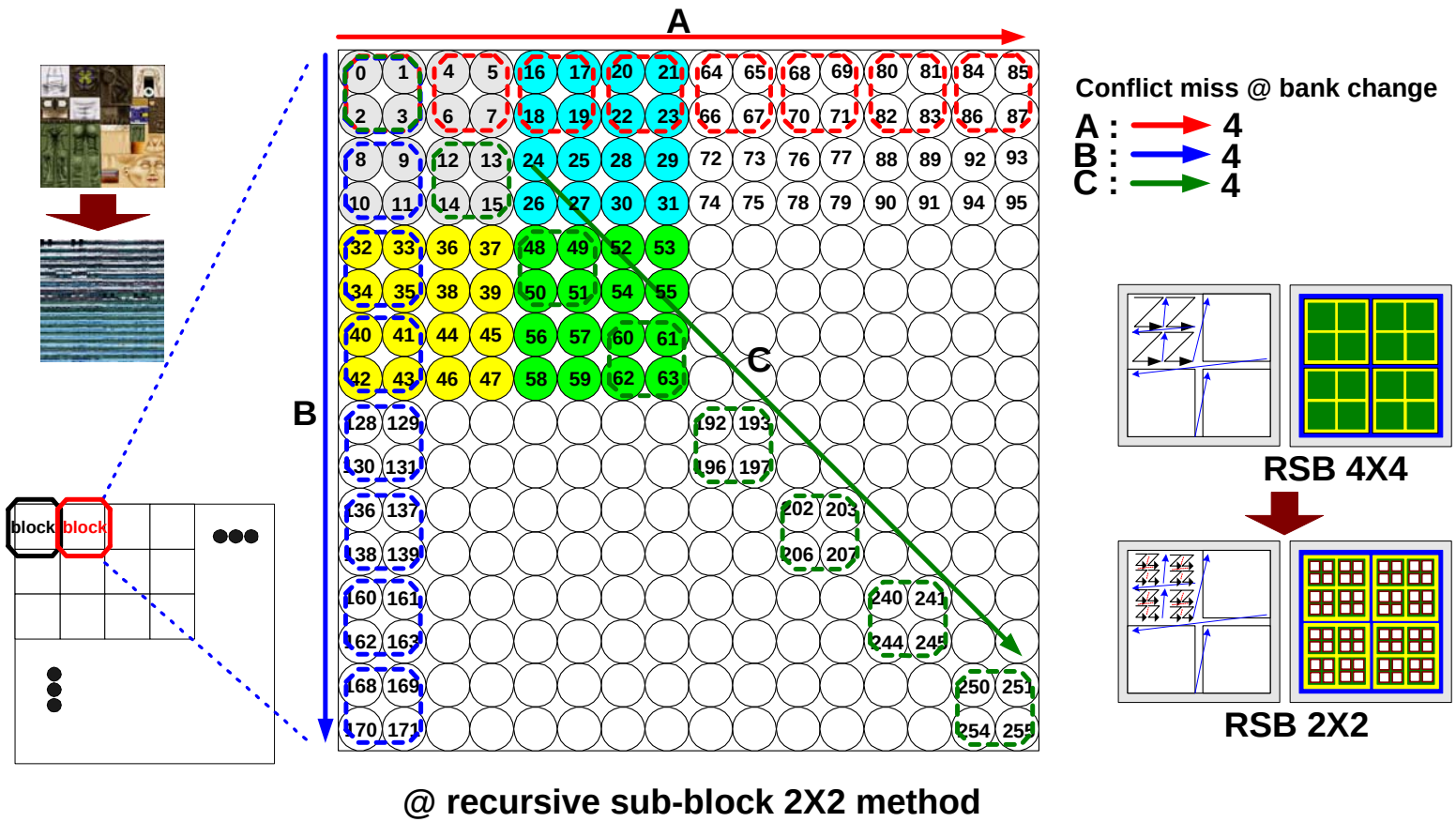
Morton order

- 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



@ recursive sub-block 2X2 method