Tapping into the power of PlayStation2

Rethinking software design to exploit the cutting-edge technology of PS2.

Speaker:Mark Breugelmans

Technology Manager - SCEE

Topics

- Looking at the way PS2 works
- How that affects your software design
- Making effective use of GS drawing ability
- Utilising a micro-programmable graphics pipeline effectively
- Data handling for parallel processing
- Pushing PS2 software to 2nd generation

Designing PS2

- Requirements
  - Hardware is fixed for lifetime of machine.
  - High-quality graphics requires massive bandwidth to maintain pixel fill-rate
  - Huge calculation ability is required to transform and light large numbers of polygons
  - Large bandwidth is required for data such as geometry and textures

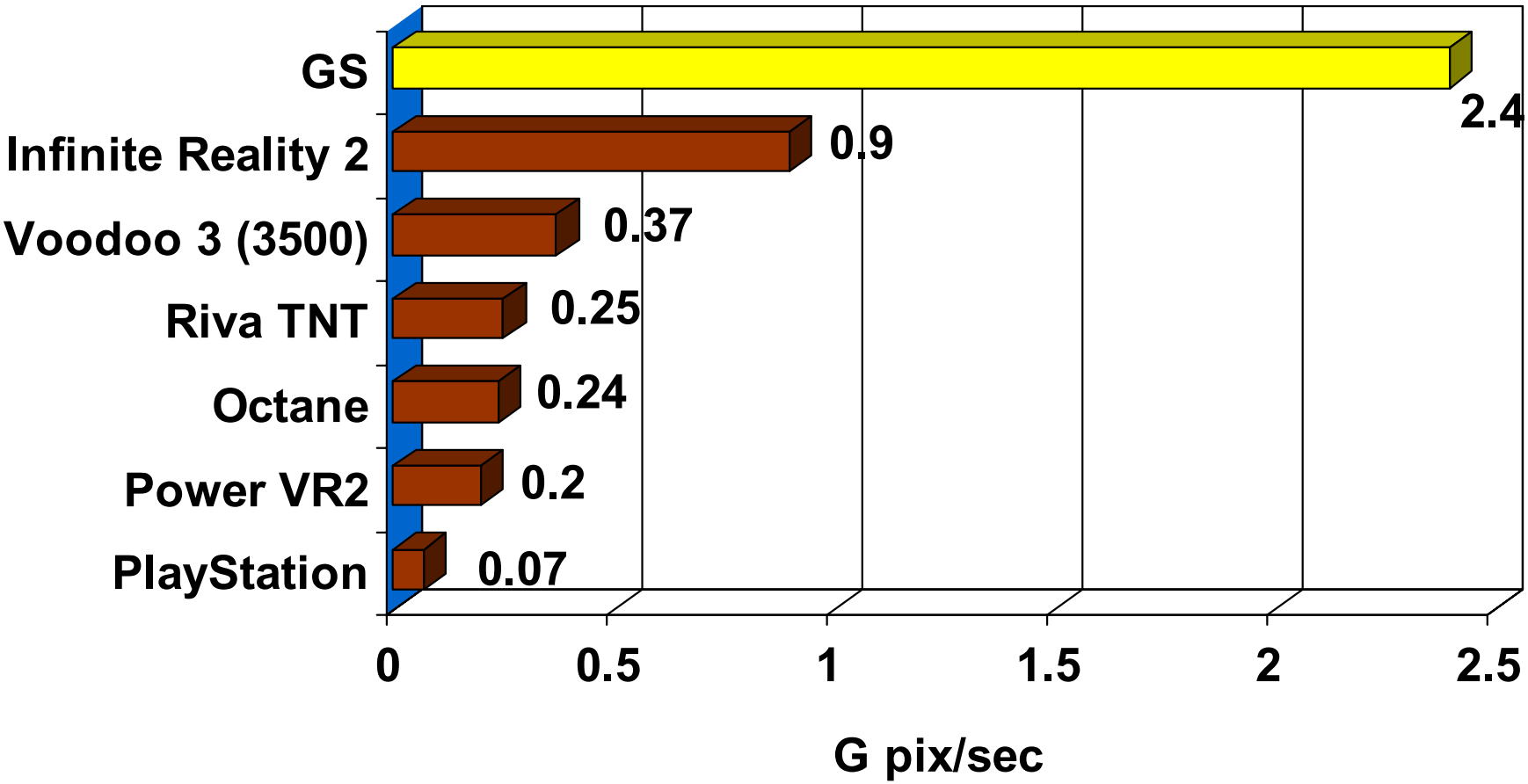
.

PS2 Design
Graphics Synthesizer

- Achieving Graphics performance
  - Embedded DRAM gives 48 GByte/sec memory bandwidth to eliminate pixel fill bottleneck.
  - Fill rate is 2.4Gpixel/sec for 32bit + Alpha + Z
  - Free alpha enables multi-pass effects
  - 1.2GByte/sec bandwidth main memory to GS for texture and geometry.

.

Pixel Fill Rate

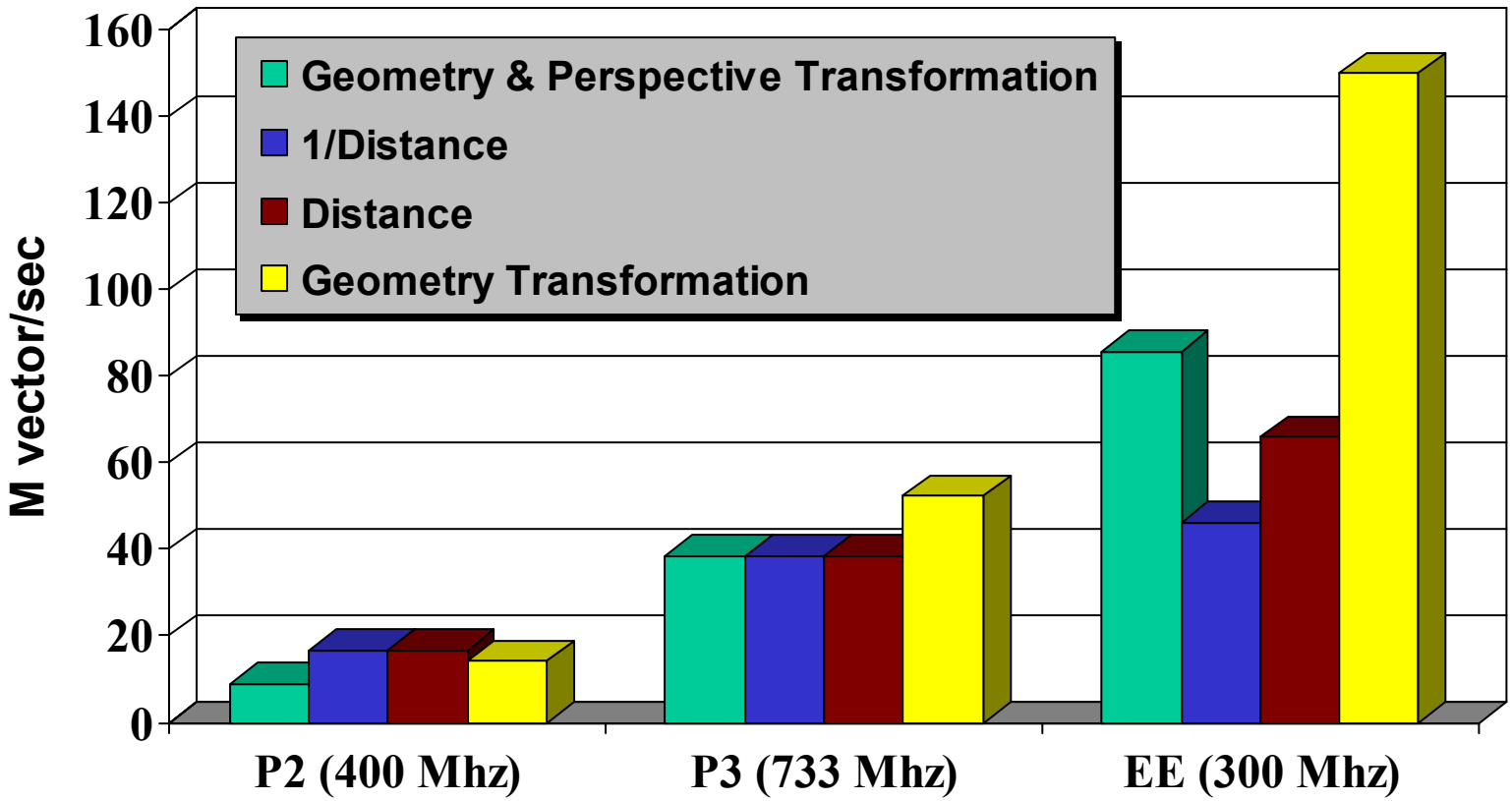


PS2 Design - Vector Units

- Total 6.2GFlops including CPU FPU
- Twin geometry engines running @300mhz
- Optimised for 3D calculations
- Easy to develop pipelined programs
- VLIW run 2 instructions in parallel
- SIMD instructions run 4 FMACS in parallel with one cycle throughput

.

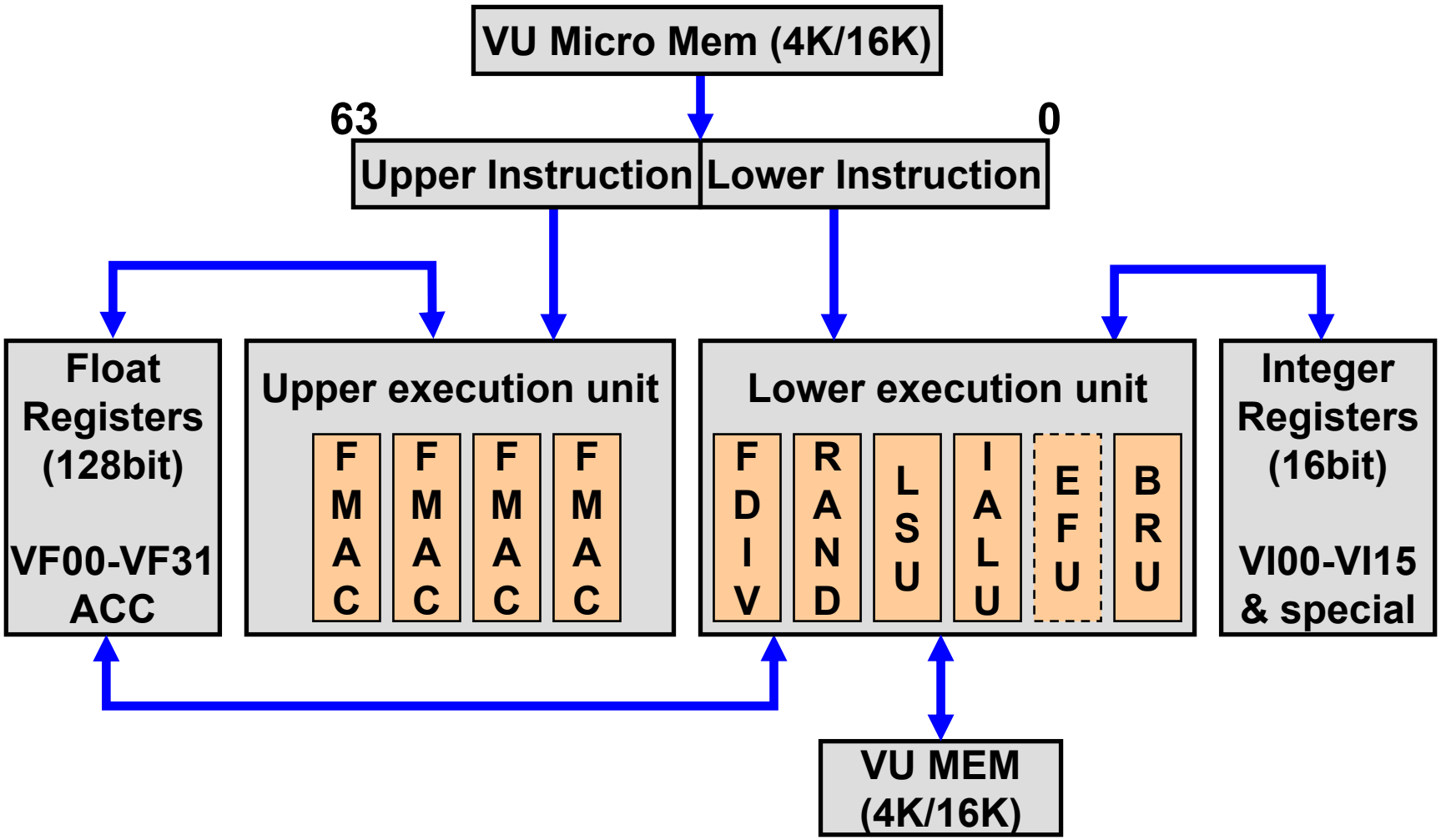
PS2 Design
Geometry Performance



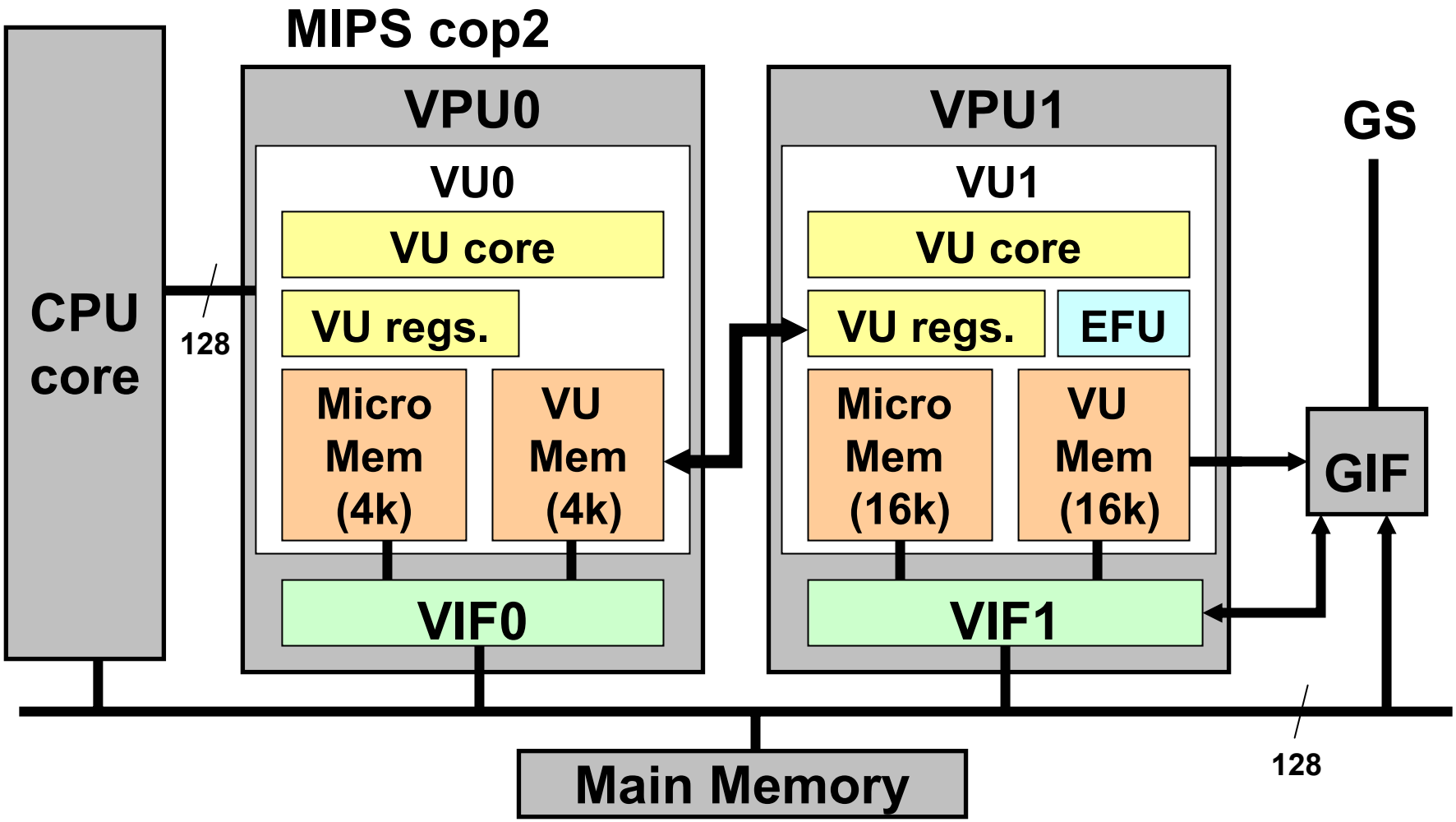
PS2 Design - Vector Units

- Vector Unit SIMD instructions (2 forms)
- Broadcast
  - X*bc, Y*bc, Z*bc, W*bc
  - (where bc is X,Y,Z,W)
- Normal
  - X*X, Y*Y, Z*Z, W*W
- There are also single cycle multiply and accumulate instructions

PS2 Design - Vector Units



Vector System outline



PS2 Design - Vector Units

- Vector Unit 0
  - Available as a co-processor for CPU
  - More flexible calculation for dynamic data such as physics, animation
- Vector Unit 1
  - Direct path to Graphics Synthesiser
  - Very fast transformation

.

Vector Code

- SIMD single cycle vector multiply
- Apply matrix in 4 cycles (uses accumulator)

```
MUL ACC, vector1, vector5x
MADD ACC, vector2, vector5y
MADD ACC, vector3, vector5z
MADD vector6, vector4, vector5w
```

- Alternatively use the 4 FMACs to perform parallel calculations eg. 4 lights.

Vector Code Example
Parallel dot product

- Store normals across vectors

  Vector 1 = (nx1, nx2, nx2, nx3)

  Vector 2 = (ny1, ny2, ny3, ny4)

  Vector 3 = (nz1, nz2, nz3, nz4)

  MULL ACC, Vector1, Vector1

  MADD ACC, Vector2,Vector2

  MADD result, Vector3,Vector3

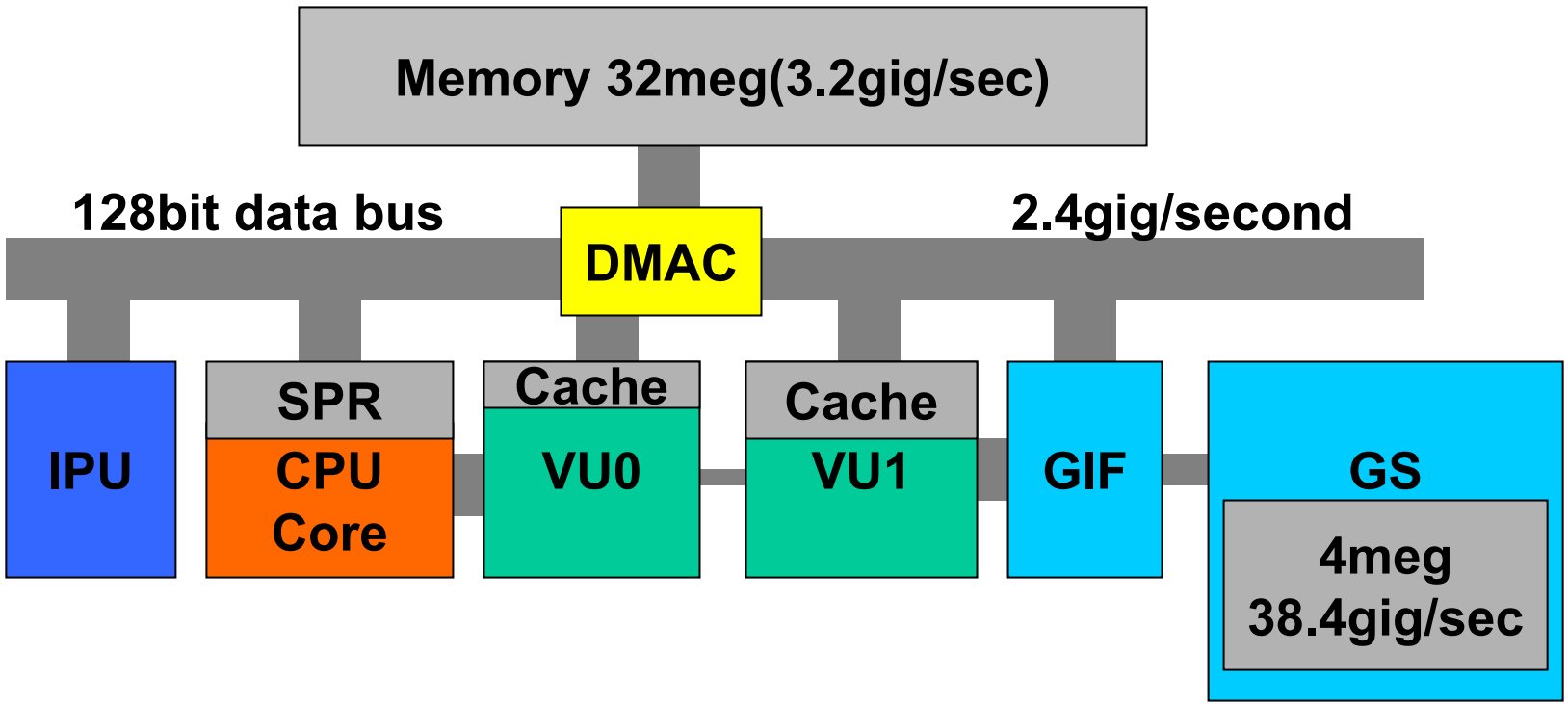
- Result is 4 dot products stored in x, y, z, w

PS2 Design - DMA System

- Data handling
  - 128bit bus provides 2.4GByte/sec bandwidth to devices
  - Additional 128bit connection VU1 to GS
  - DMA intelligent data transport frees CPU
  - Image decompression unit
  - Geometry decompression: delta + unpack

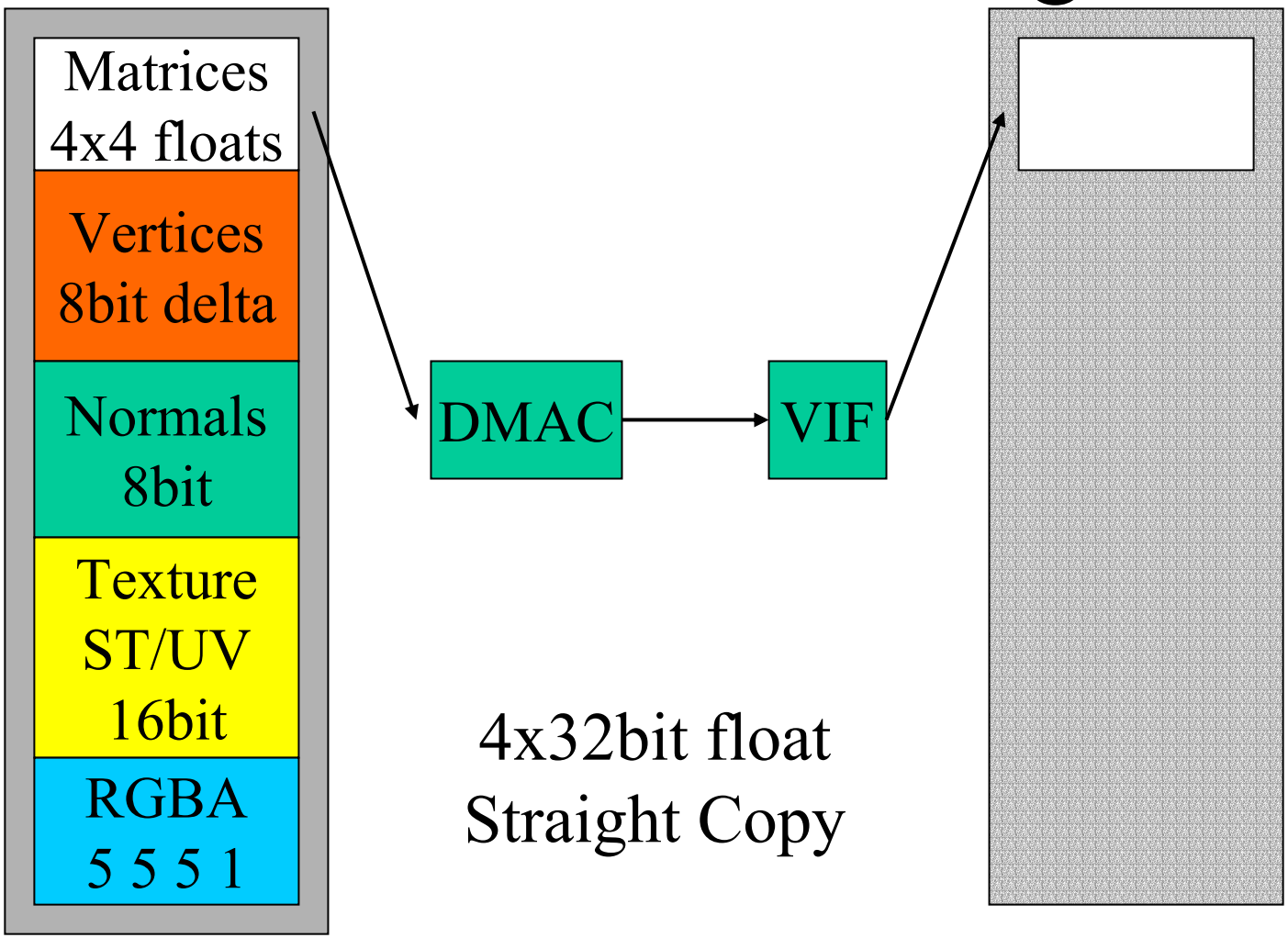
.

PS2 data handling

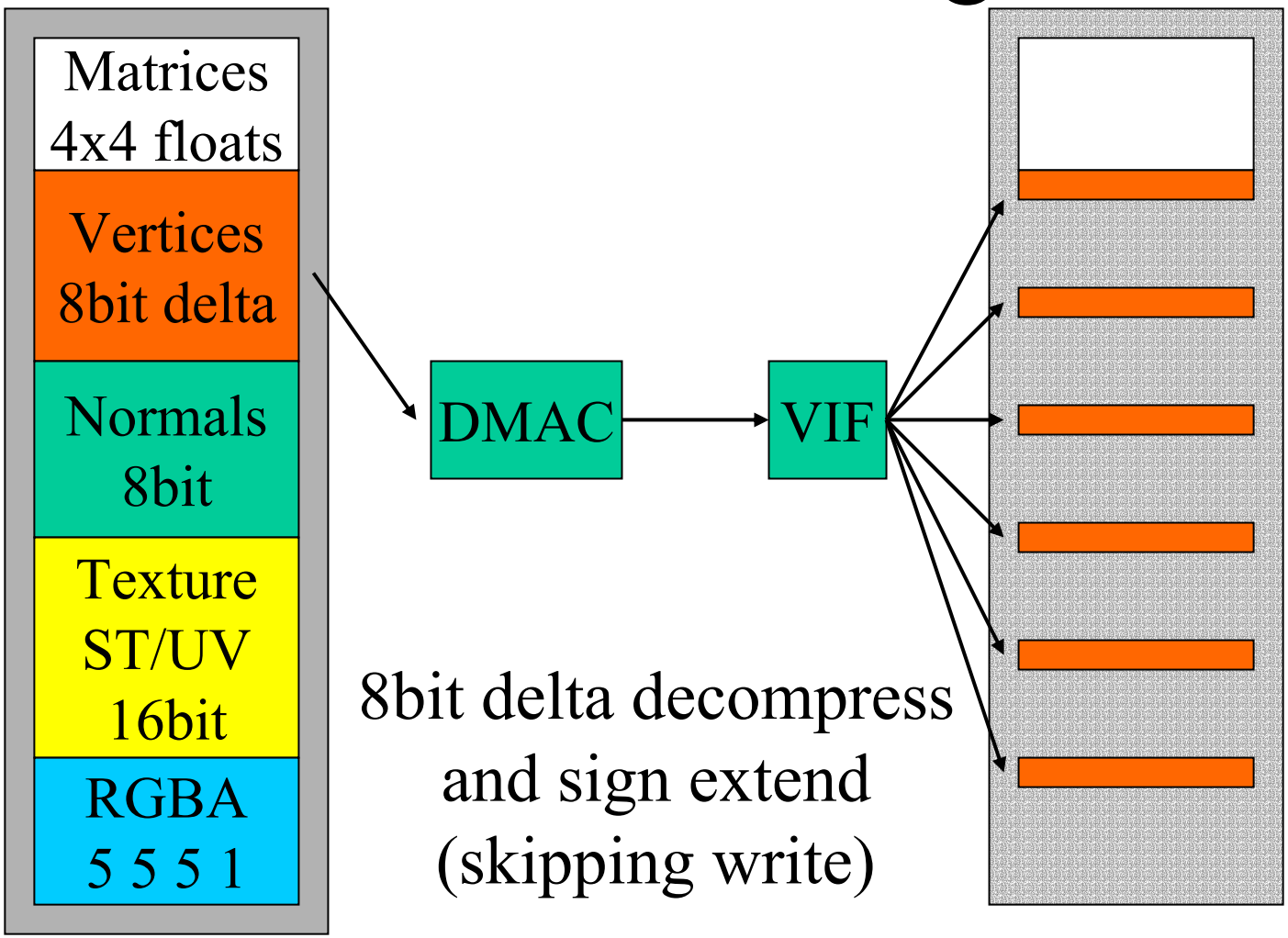


Hybrid UMA = 40 meg/frame can be transferred between devices
Intelligent DMA handler completely frees CPU

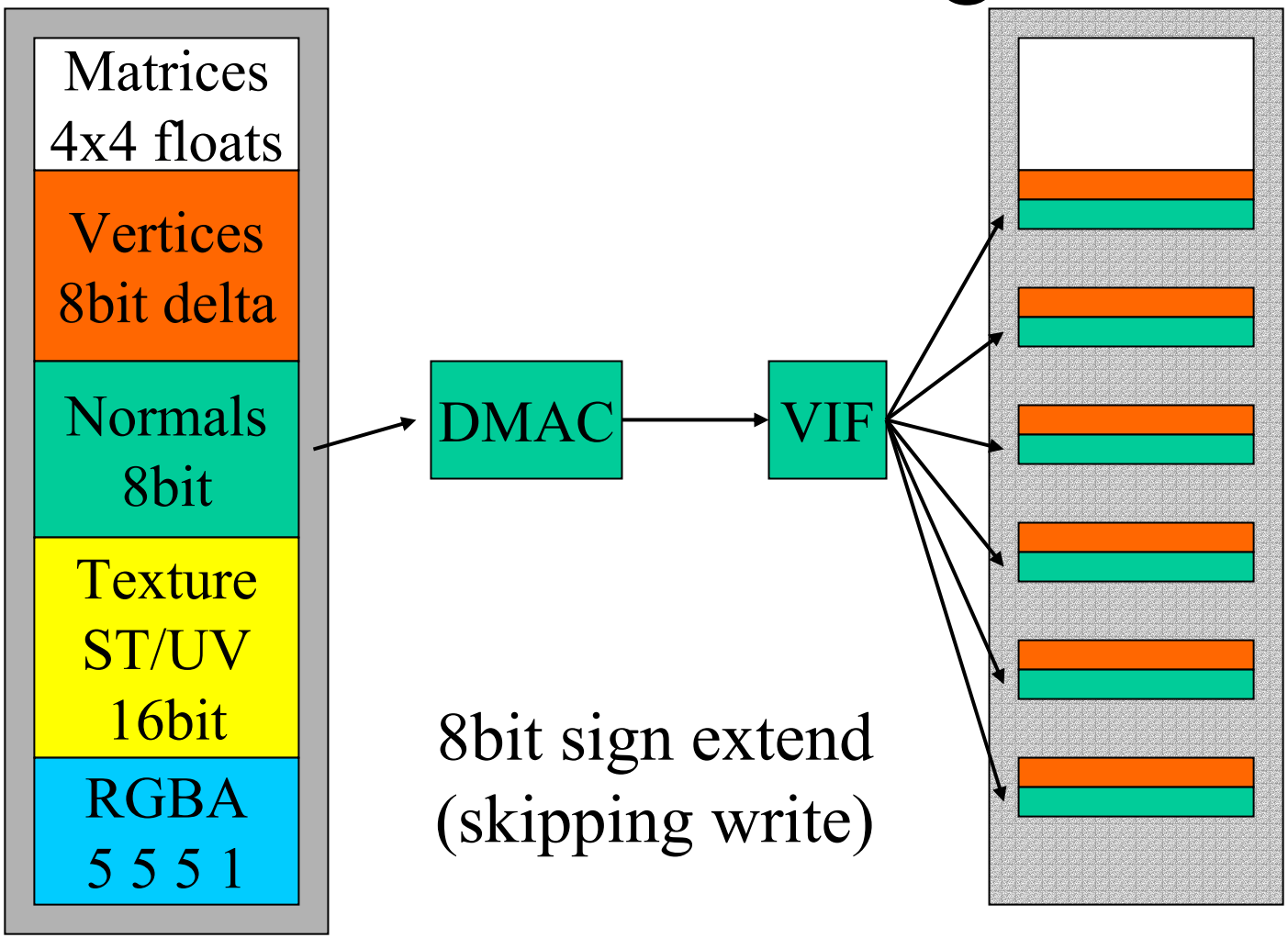
PS2 Data handling



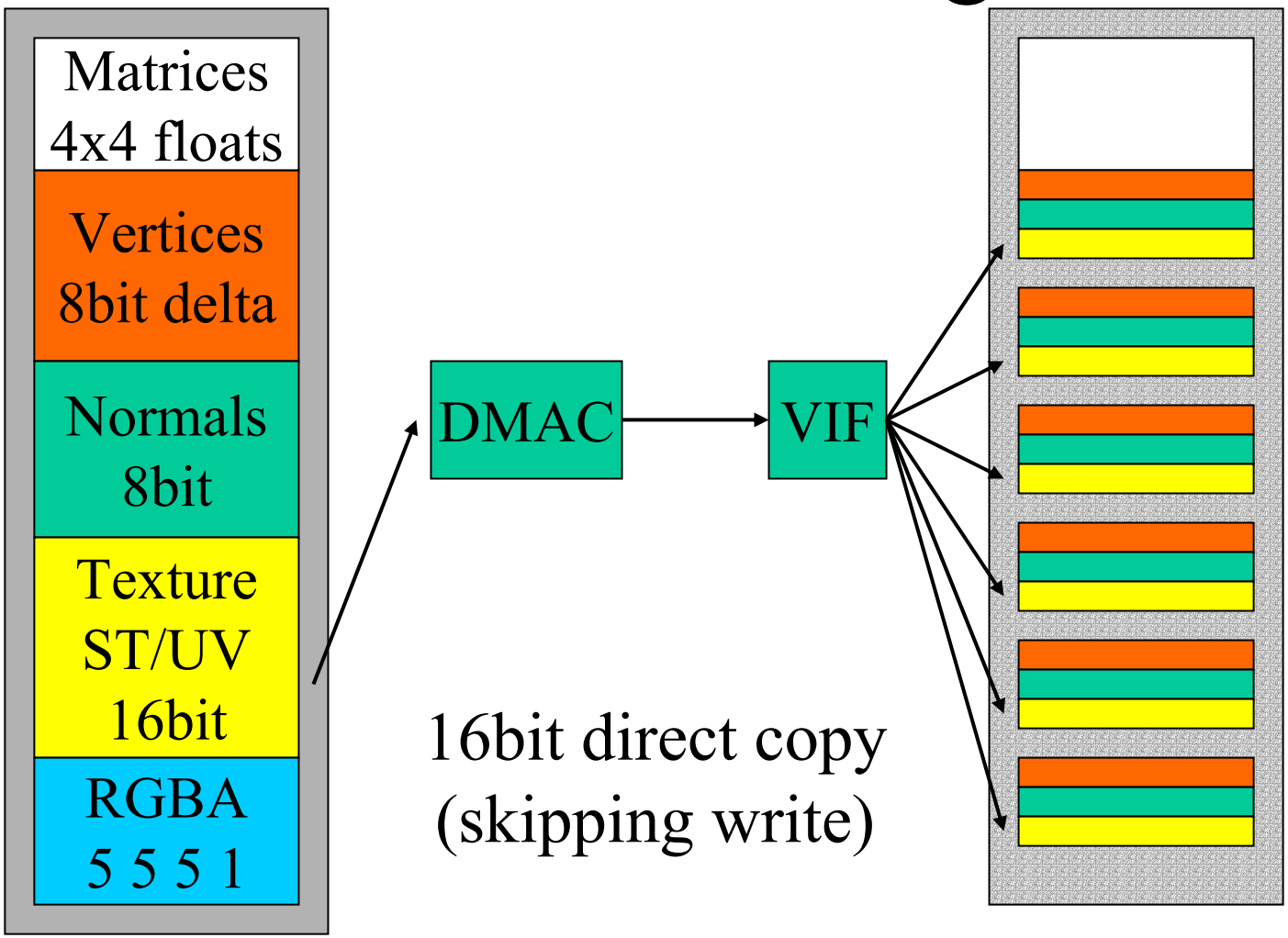
Data Handling



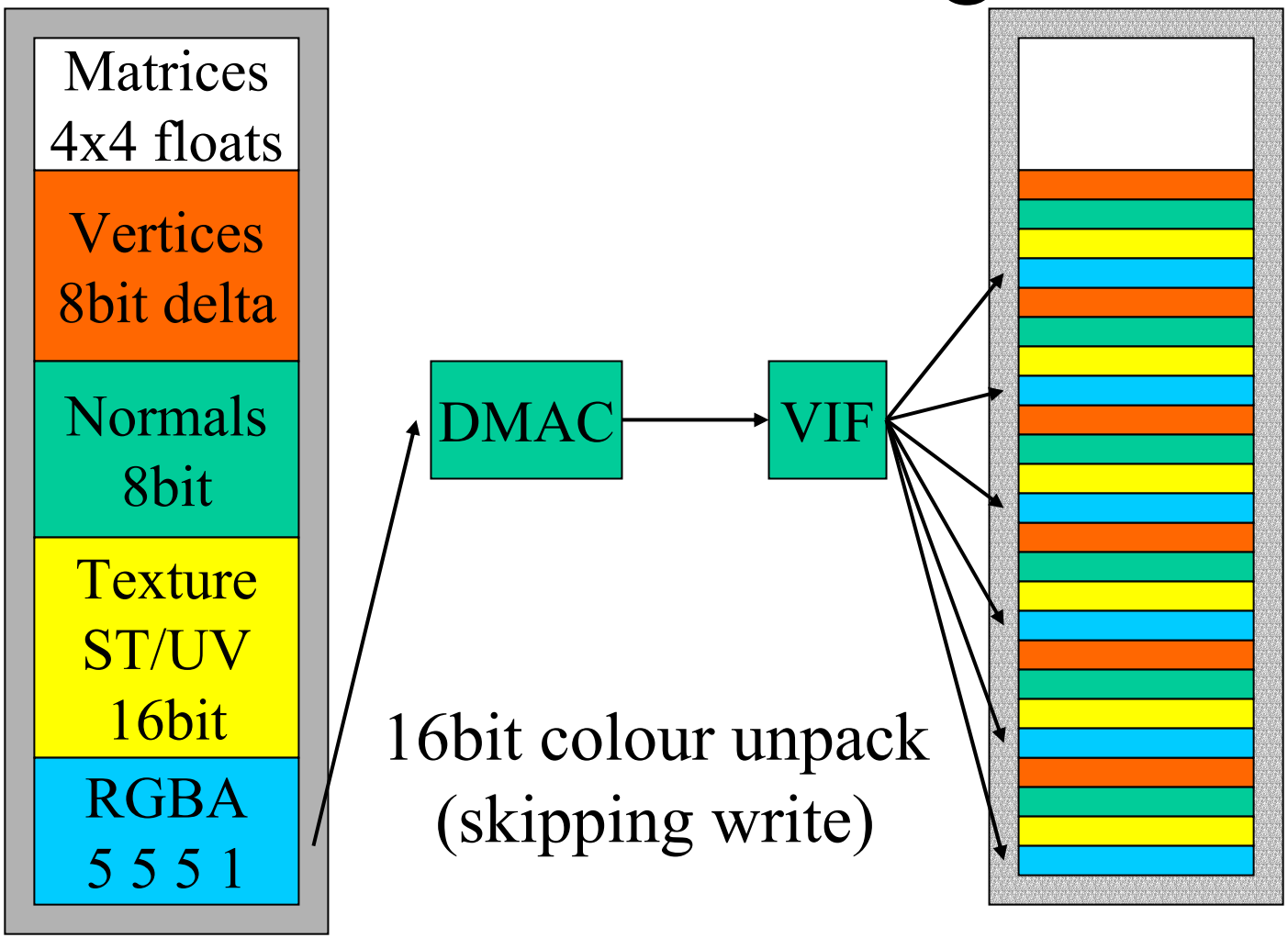
Data Handling



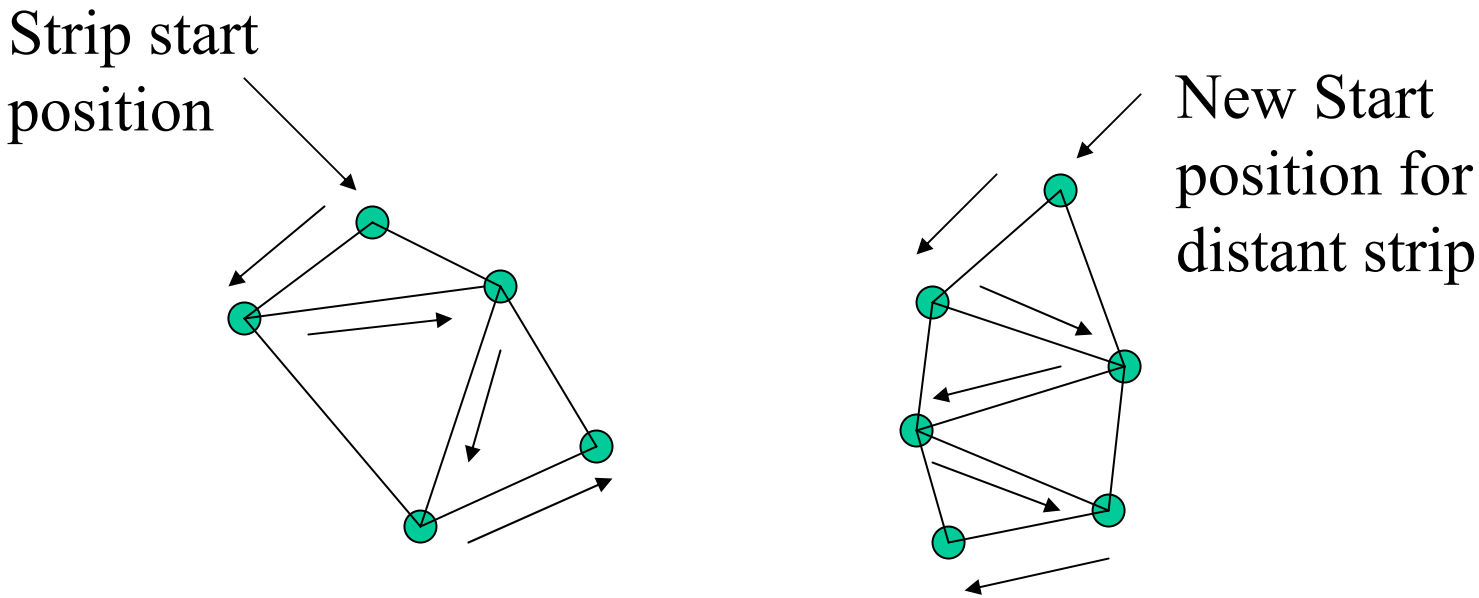
Data Handling



Data Handling



Delta Decompression

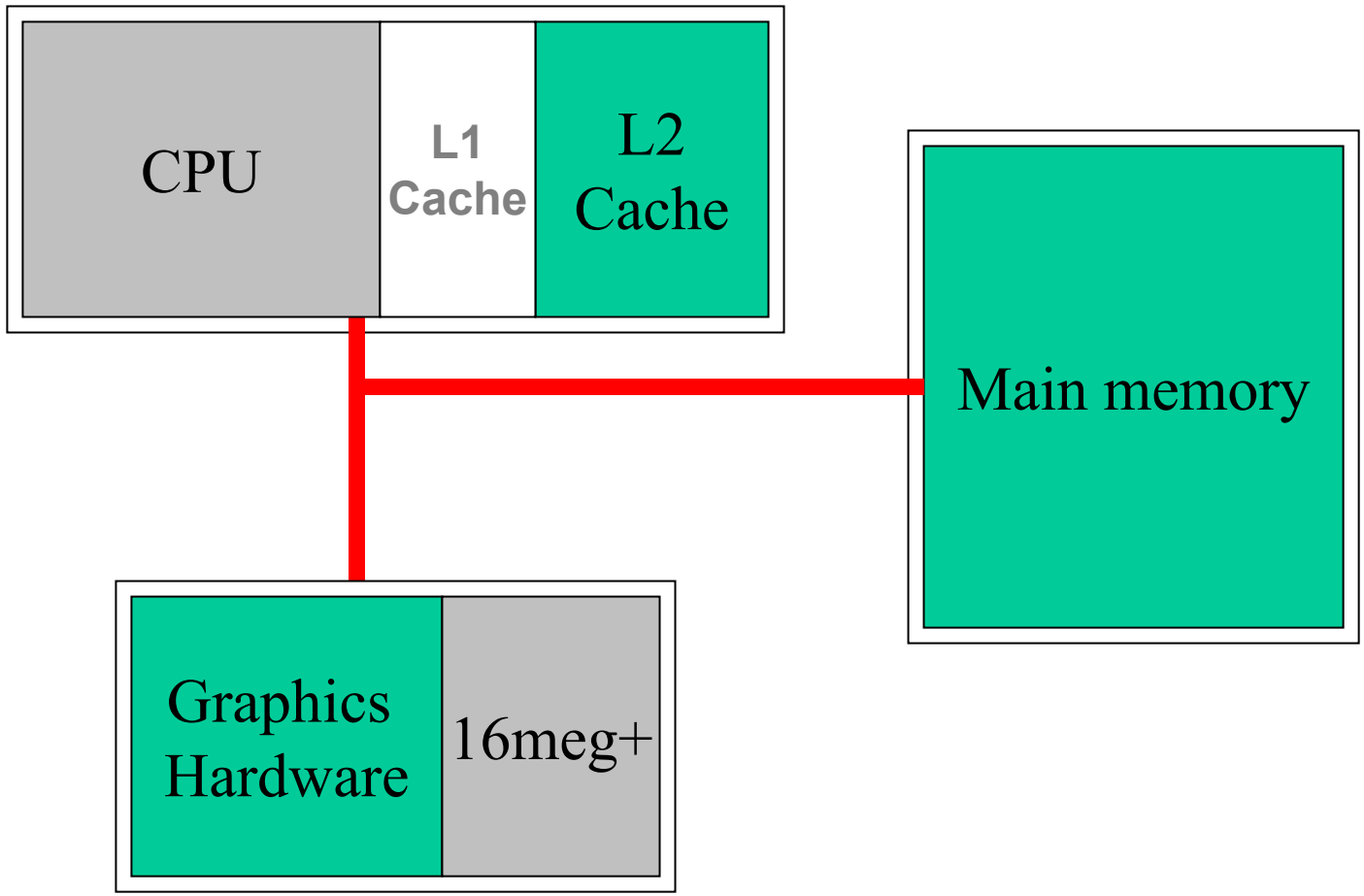


- Points in a strip tend to be close together so delta is useful.
- After the start position each point is offset from the previous position.
- This packs vertex from 12 bytes for 3 floats down to 3 bytes

Cache usage

- Features of cache
  - High speed memory
  - Expensive
  - Maintains CPU speed for data already in cache
- Dynamic media such as geometry does not stay in the cache for any length of time so large cache is unhelpful.
- Data throughput is more important

Simplified PC architecture



PC games design

- PC has large cache but poor bandwidth
  - Big indexed meshes
  - Extensive culling and clipping to reduce bandwidth to graphics card
  - Textures cached in Graphics card ram for speed
  - Higher resolution and good monitors require high resolution textures
- This doesn't work well on PS2!

PS2 Game

- PS2 has good bandwidth and small caches to buffer DMA
  - Enables on the fly texture downloading
  - Extra DMA geometry compression is available
- Vector units enable alternative efficient model representations
  - Procedural models / patches
  - Micro-models

.

1st generation PS2

- Fighting small CPU cache,lack of VRAM
- Existing game with bigger textures
- Poly counts upped on models.
- Basic VU1 Directional lighting / Prelit
- Recompiled PC or PS1 code
- Minimal post processing
- Inefficient geometry

.

2nd generation PS2

- Working with the memory system
  - Maximising DMA use
  - Using efficient data representations
- Exploiting the architecture of micro programmable graphics pipeline
  - Many different types of renderers
- Using GS fill rate
  - Enhanced graphics effects

Examining GS chip

- Huge bandwidth and fill rate
- Very fast alpha blending
- Setup time is the limitation to poly counts
- Always use strips
- Designed for TV
  - Note (640x256*50=8mil pixel/sec)
- Use the high poly count with multi-pass

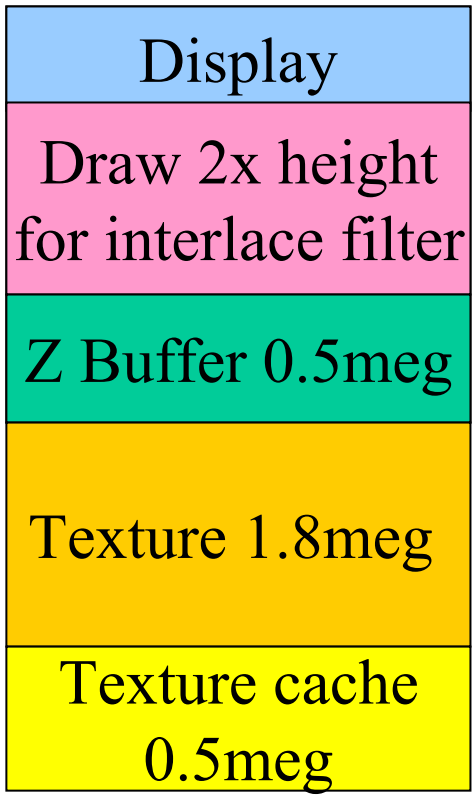
.

Using GS effectively

- Use multi-pass (alpha is free) and saves on vertex re-calculation
- Strips offer a huge saving over triangles
- Keep polygons small - many small polys are much quicker than one large polygon.
- Penalty on miss is not huge.
- Texture tiling and wrap modes offer efficient resolution improvement

Suggested VRAM arrangement

- For 60hz frame-rates



640x224x24bit = 430k

640x448x24bit = 860k

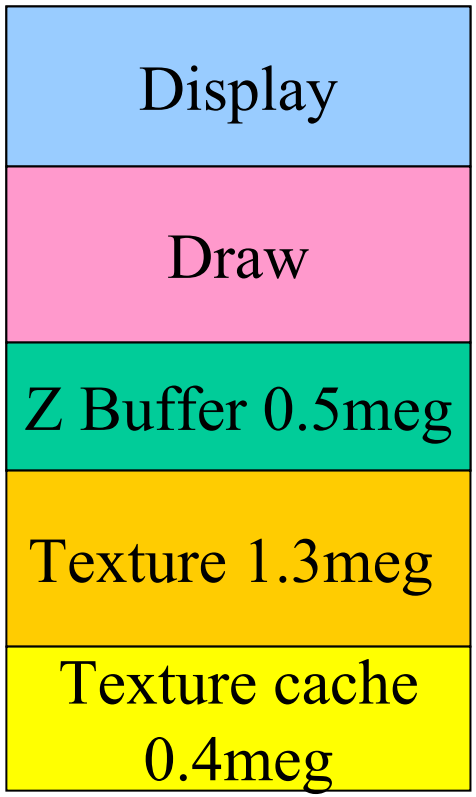
640x448x16bit = 573k

Static textures = 1.8meg

Texture cache=8*256x256 (8bit)

Suggested VRAM arrangement

- For <60hz frame-rates



640x448x24bit = 860k

640x448x24bit = 860k

640x448x16bit = 573k

Static textures = 1.3meg

Texture cache=6*256x256 (8bit)

Using GS effectively

- 640x448 is maximum visible pixels per frame with 2 sample / pixel full screen filter
- Assuming fairly inefficient MIPMAP and multi-pass using 20 texels per pixel at 8bit = 6meg
- With 2meg static texture and 4meg/frame DMA this leaves 16meg geometry bandwidth
- Savings can be made with tiling and multiple passes with 4bit textures

.

Paletted textures

- Efficiency of paletted texture improves with size
- Quality of paletted texture degrades with size
- For 256 colour pallet minimum 16x16 useful size (all colours used). 256x256 still good
- For 16 colour pallet minimum 4x4 useful size. Works well for monochromatic textures

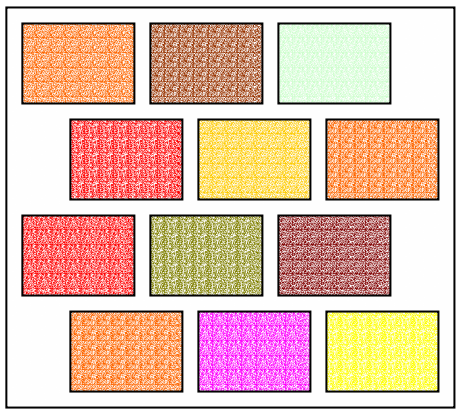
.

Paletted textures example
Only 4bit pallet!

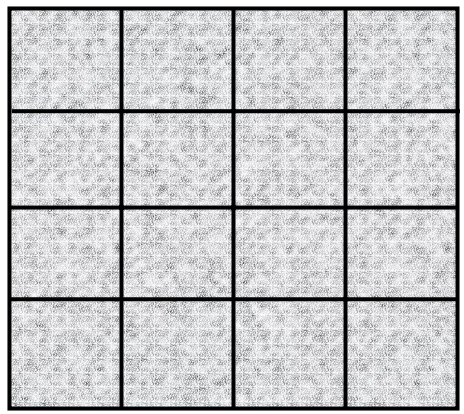
.

Using GS effectively

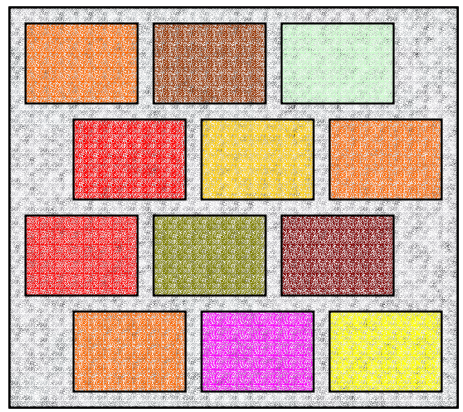
- Improving texture detail (2 layers)



Texture 1(4bit)
128x128x4bit=8k



Texture 2(Tiled)
64x64x4bit=2k



Texture 1*Texture 2
256x256x24bit = 196k

The result is 10k versus 196k = nearly 20:1 saving

GS Effects

- Smoke, Fire
- Motion Blur
- Focus effects
- Heat haze
- Anti aliasing
- Interlace blending

GS Effects : Smoke, Fire

- Use particle emitter to render off screen
- Alpha blend large percentage of previous sprite
- Use scale and translate to create drift
- Add ‘noise’ to emitter position to create lifelike effect

GS Effects : Motion Blur

- Draw new screen
- Copy on top with transparency the previous frame
- Effect is adjusted by alpha value
- Fill rate easily allows this during the Vertical Blank

GS Effects : Depth of field

- Draw new screen into off screen buffer
- Copy to display buffer
- Sprite copy screen with scale to display buffer using Z test
- Repeat for different depths

.

GS Effects : Heat Haze

- Draw new screen into off screen buffer
- Create mesh of squares texture mapped to draw buffer
- Copy squares to display with texture coordinate offsets based on sinusoidal functions & time
- Display distorts like heat haze

GS Effects : Interlace blending

- Odd and Even lines are drawn alternately.
- Any image not drawn on both lines flickers.
- Scan-line blending solves the problem.
- Techniques to fix this
  - GS blend circuits with interlaced buffers(free)
  - Motion blur % of previous field(low VRAM)
  - Shrink full height to display height with bilinear

.

GS Effects : Anti-aliasing

- Hardware edge anti-aliasing.
  - Requires sorted polygons but looks excellent.
- Full screen anti-aliasing
  - Render to large draw buffer then use bi-linear filter to blend pixels smoothly.
  - Render 4x with 25% alpha and 1/2pixel offset in 4 directions. Same effect without the VRAM hit

.

PS2 parallel geometry pipelines

- You have to use Vector Units for good performance
- Many games don’t use them both effectively
- These are very flexible : sub-div surfaces, patches, terrain etc
- Micro-models
- Procedural models / Bezier patches

.

Using VU effectively

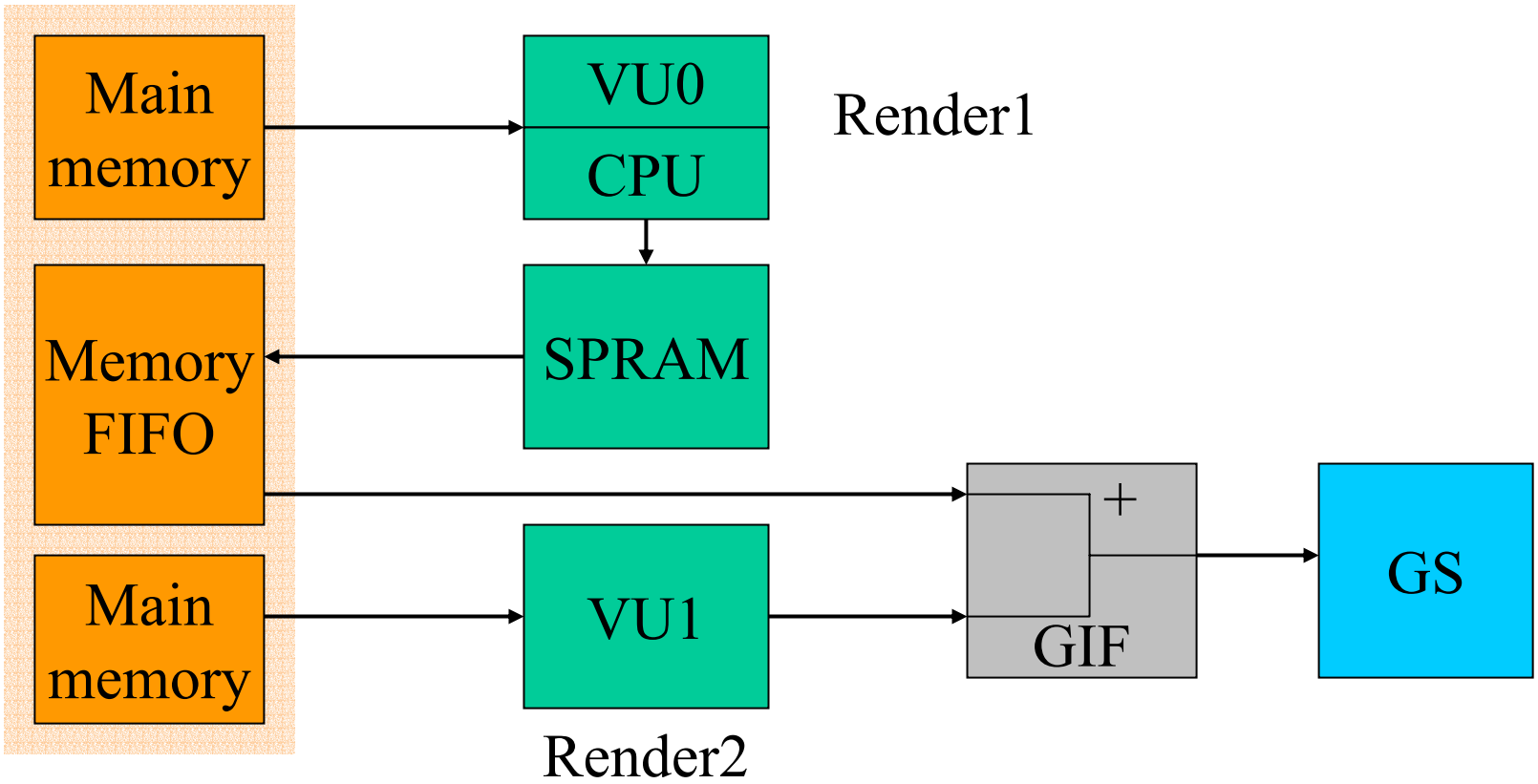
- Use many renderers (clipping/non clipping)
- Reuse vertices (micro-mesh / patch)
- Remember polys = 300MHZ/(cycles per vertex) so keep the VU busy every cycle
- Use them both!
- Pack your data

.

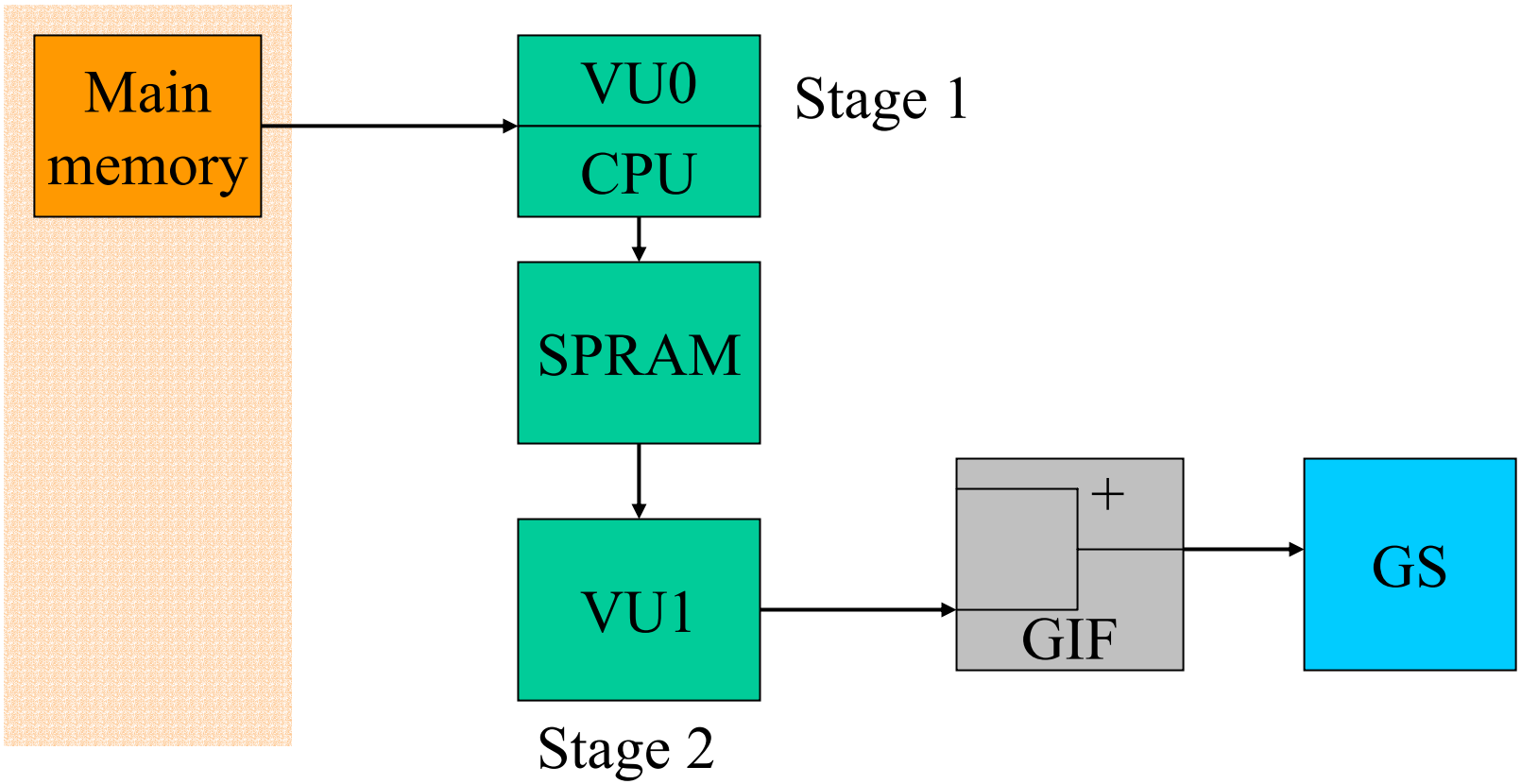
Vector Unit Optimisations

- Remember
  - Vertices = Clock speed/number of cycles
- Removing latencies
- Prologue, Epilog, Rolling up loops
- Vertex reuse
- Short vs Long loops

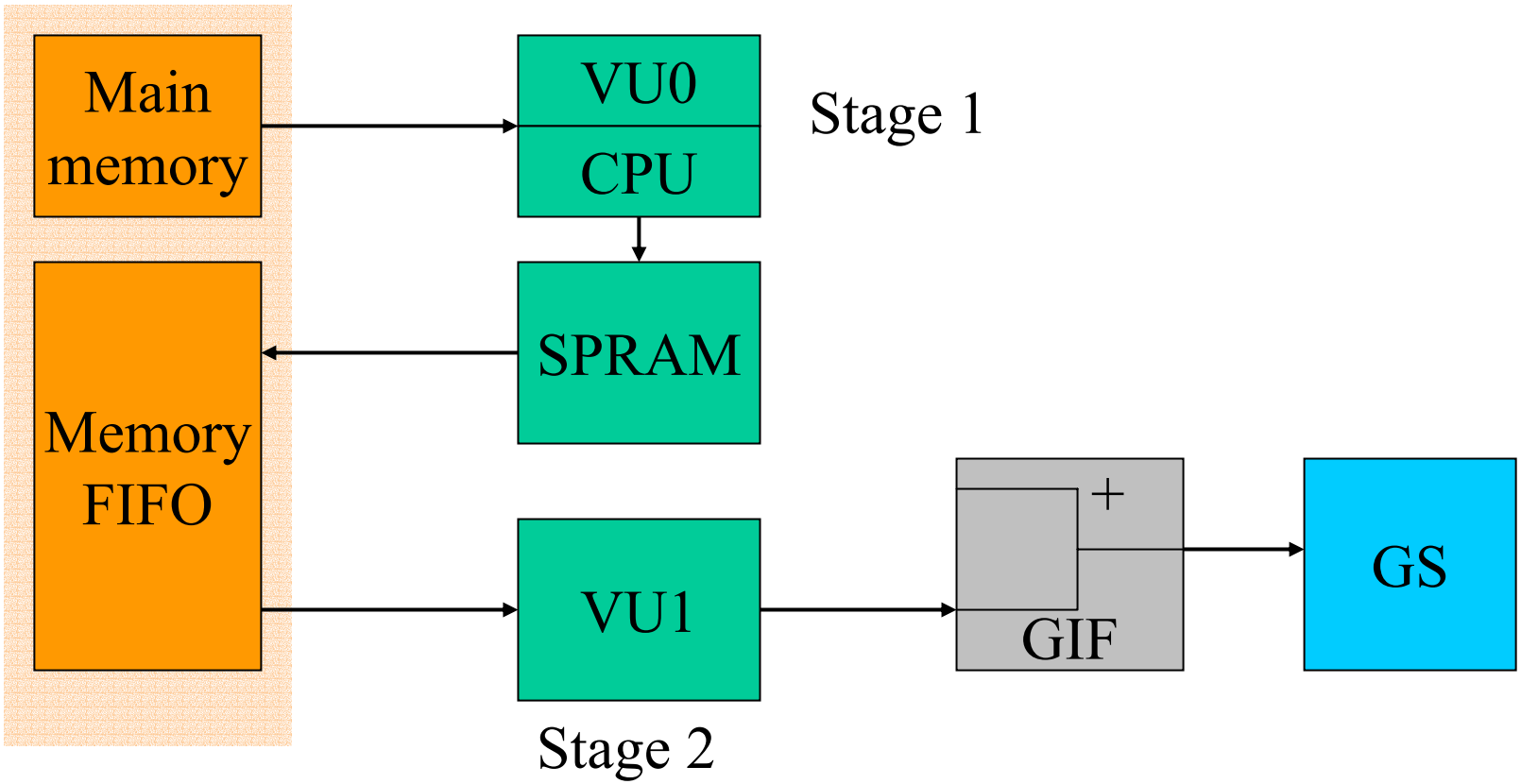
Using Vector Units Together Parallel connection + MFIFO



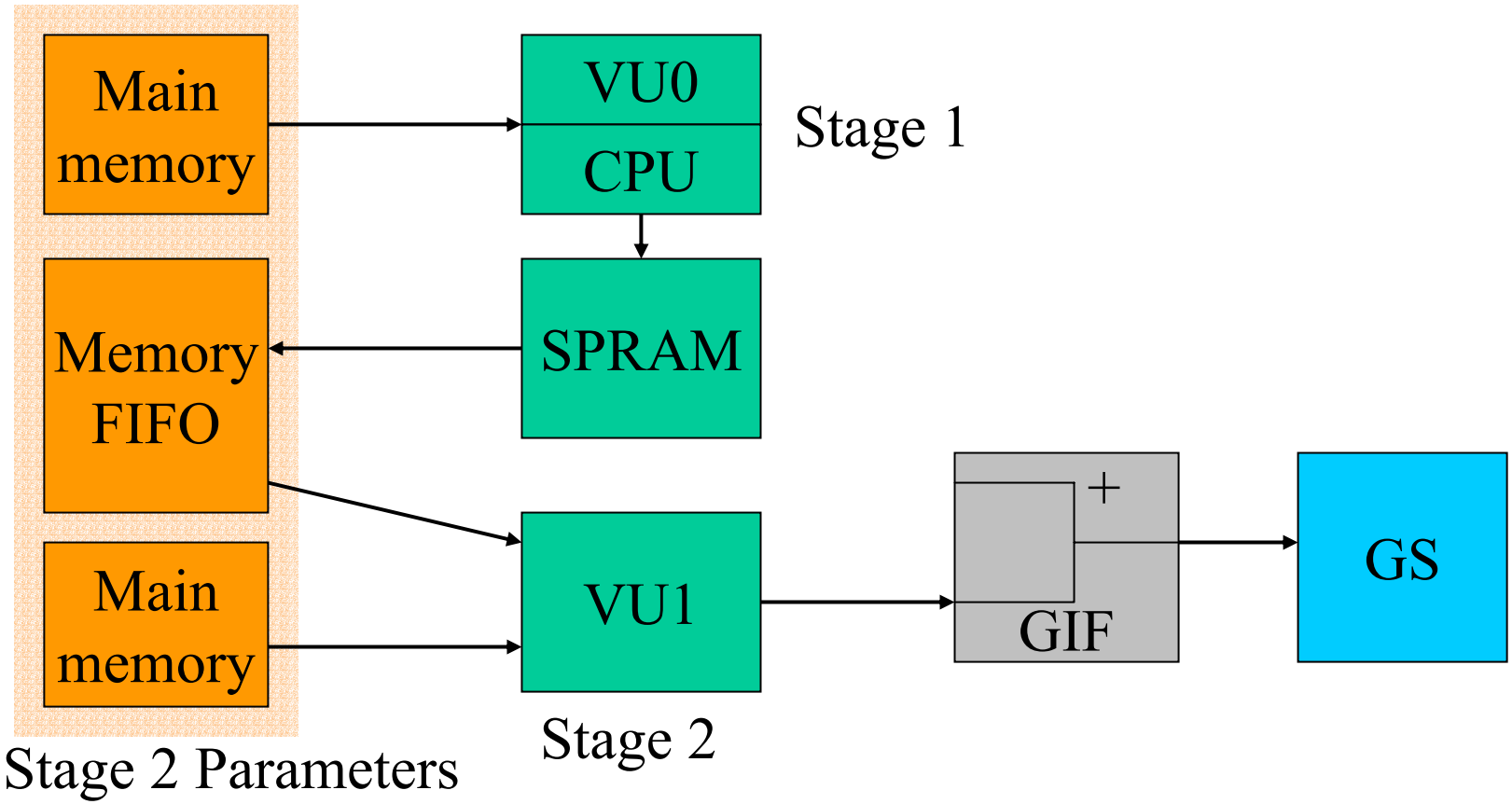
Using Vector Units Together
Serial connection 1



Using Vector Units Together
Serial connection 2 + MFIFO

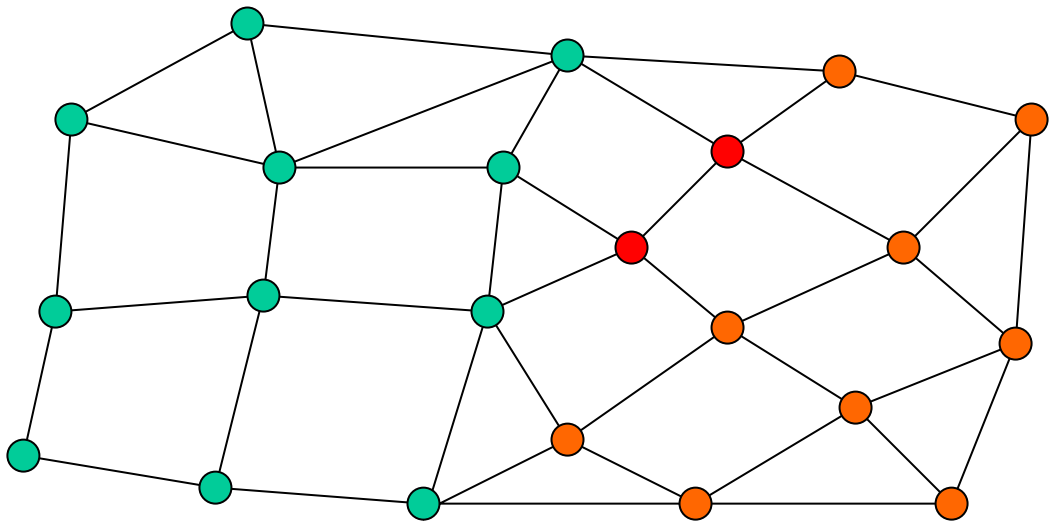


Using Vector Units Together
Serial connection 3



Micro Models

- Most models can be separated into patches of points
- These share vertices, normals and a common texture
- Vertex reuse improves rendering efficiency
- This allows more complex lighting models



.

Micro models

- Vertex cache
  - Divide model into patches using shared vertices
  - DMA set of vertices
  - Transform vertices
  - Indices used to generate triangle strips
  - Aim to achieve more than 1 triangle per vertex
- Circular FIFO style vertex cache

VU procedural renderers

- High efficiency vertex reuse
- Free bandwidth
- Wheels, spheres,
- Trees
- Terrain
- Bezier patches
- etc etc

VU deformers

- Displacement maps
- Random displacement
- Time based deformation
- Noise functions
- Texture warping

Conclusion

- Code with the system not against it
  - DMA is the key
  - Keep the Vector Units busy
  - Use that fill rate!

- The results can be excellent if the system is used well