# PS2 Programming Optimisations

George Bain

SCEE Technology Group

SONY

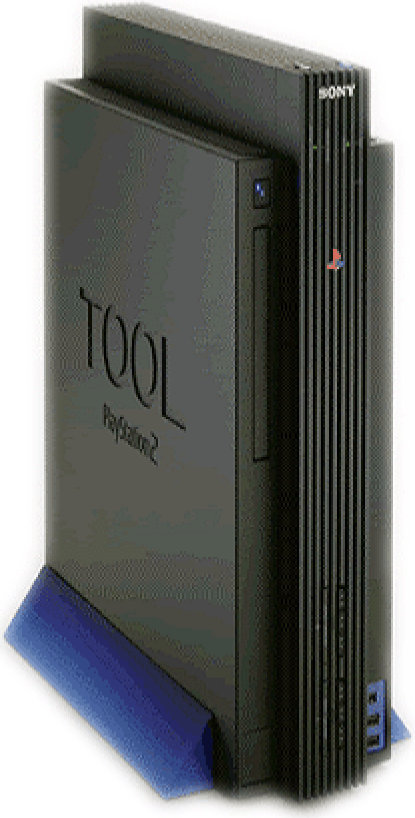
COMPUTER ENTERTAINMENT

# Topics Covered

- Performance Analyser
- DMA Transfers
- Vector Units
- Graphics Synthesizer
- EE Core: CPU
- File loading

# Performance Analyser

- Capture snapshot of
  - EE (Core, Bus, Vu0, and Vu1)
  - GIF and GS
- 7 frames of bus activity
- Identify bottlenecks!
- Also used as a Dev Kit

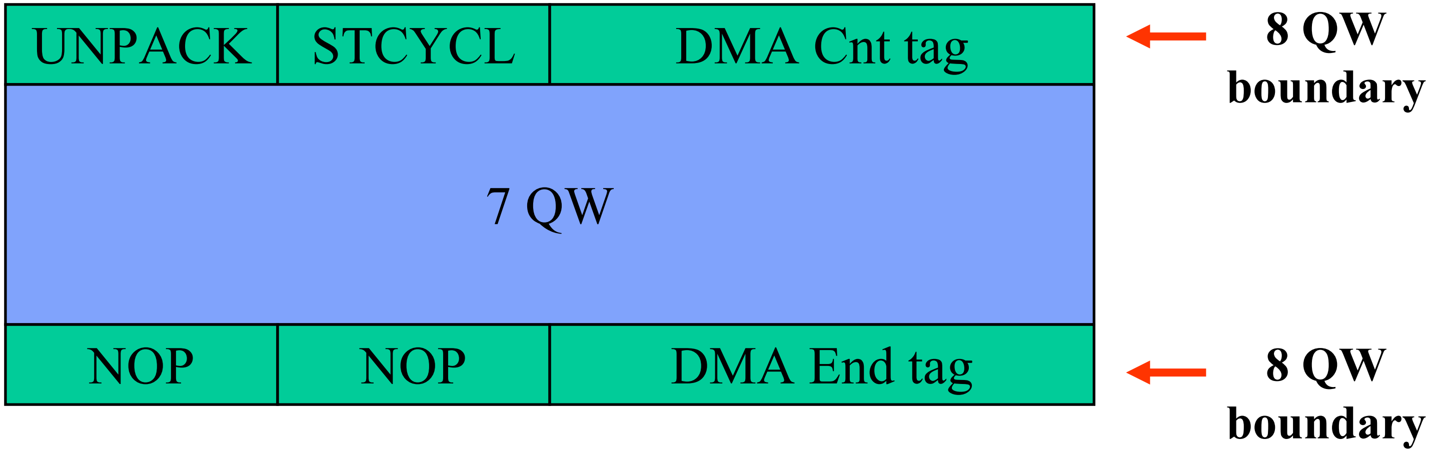


# PS2 Memory

| | | |
|---|---|---|
| CPU | 8K Data | 32MB RDRAM |
| | 16K Instruction | |
| | 16K Scratchpad | |
| Graphics Synthesizer | 8K Frame | 4MB Embedded |
| | 8K Texture | |
| Vector Unit 0 | 4K Data | N/A |
| | 4K Instruction | |
| Vector Unit 1 | 16K Data | |
| | 16K Instruction | |

# DMA Bus Bandwidth

- EE RDRAM to Device = 2.4 GB/Sec
- DMAC Transfers in 8QW slices
- Align DMA Reference data on 8QW Boundary
  - Increase DMA transfer speed 30-40%
- Limit DMA tags
- Tag alignment

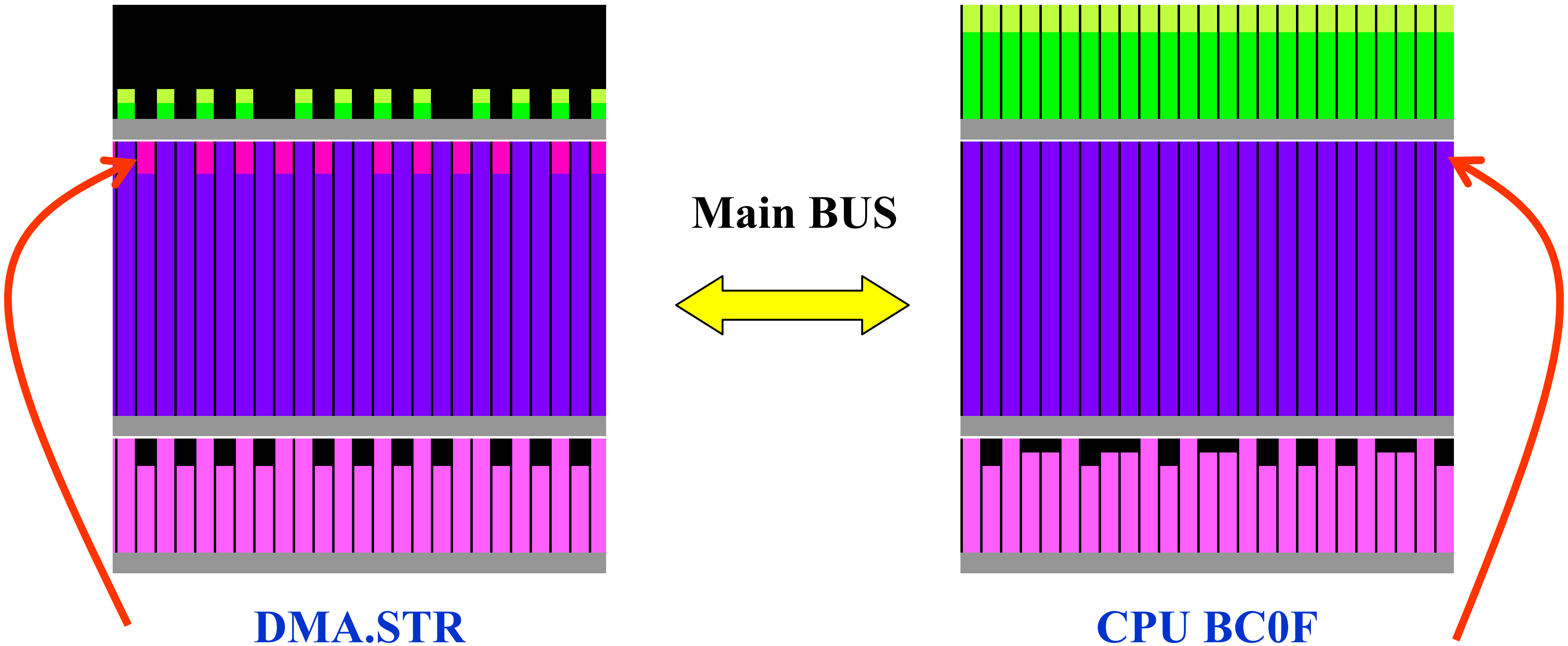


| UNPACK | STCYCL | DMA Cnt tag |
|---|---|---|
| 7 QW | | |
| NOP | NOP | DMA End tag |

**8 QW boundary**

**8 QW boundary**

# Checking DMA completion



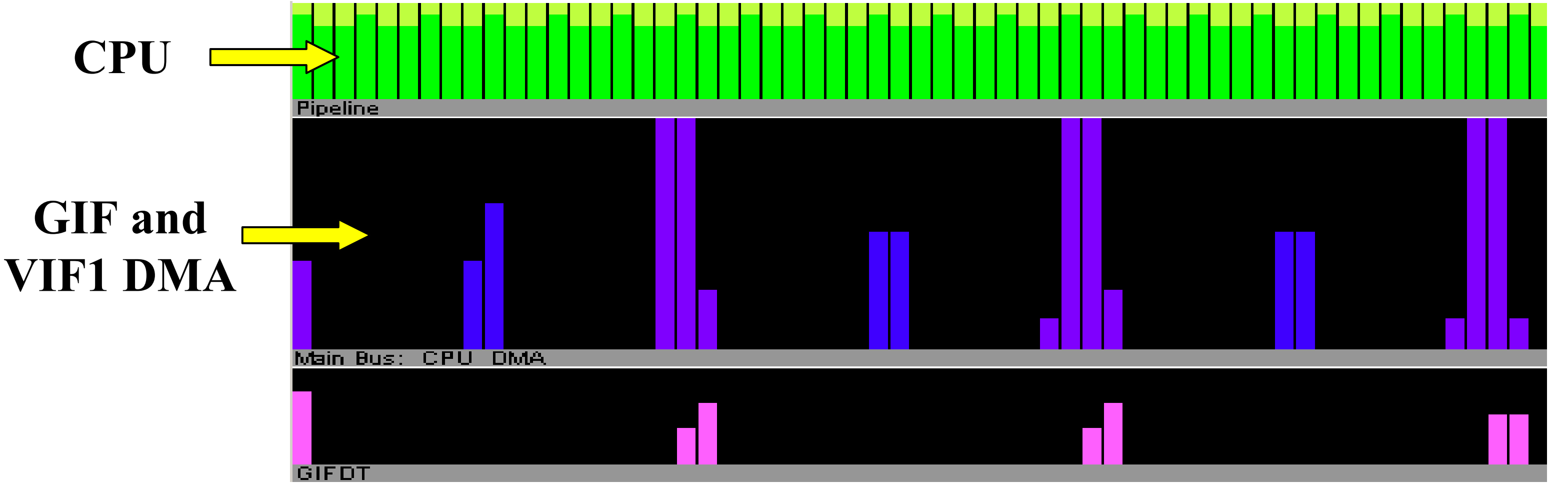
Main BUS

**DMA.STR**
**Register polling**

**CPU BC0F**
**Polling**

# Cycle Stealing

- Cycle Stealing ON or OFF?
  - release is time between two DMA slices
  - allow more time for CPU to access the main bus
  - slows down overall DMA transfer



CPU

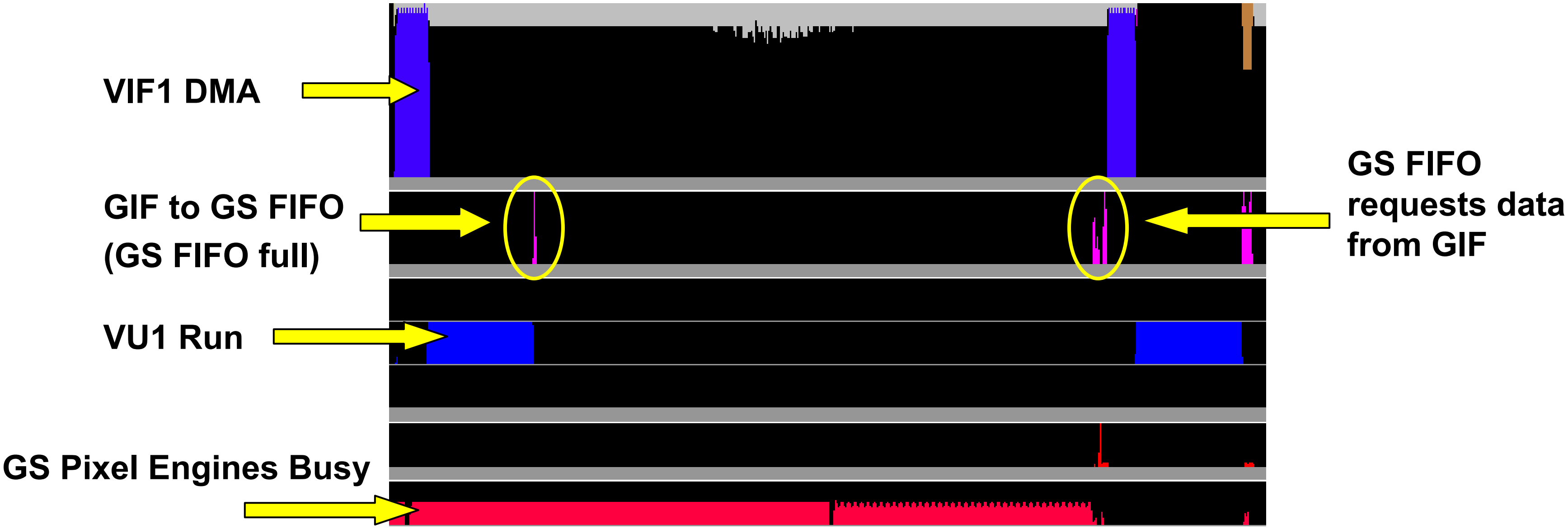
GIF and VIF1 DMA

# Memory FIFO

- What are the advantages?
  - MFIFO can buffer DMA packets if stall occurs on Drain DMA channel
    - When VU1 or GS becomes the bottleneck
  - Avoid Data Cache and perform memory writes to 16K scratchpad memory
  - Scratchpad DMA provides maximum DMA transfer speed to Memory FIFO

# GS FIFO

- What can cause the GS FIFO to become full?
  - Large primitives such as a full screen sprite
  - Multiple texture passes



VIF1 DMA

GIF to GS FIFO
(GS FIFO full)

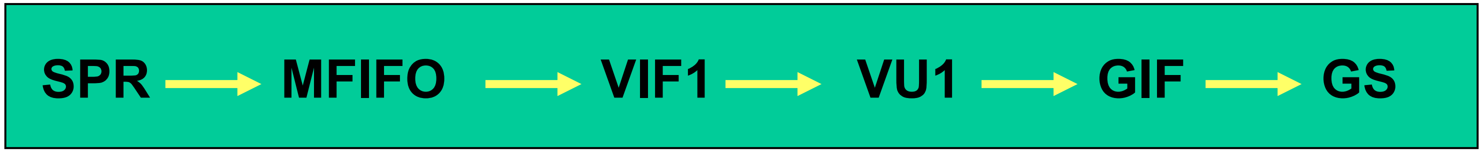
VU1 Run

GS Pixel Engines Busy

GS FIFO
requests data
from GIF

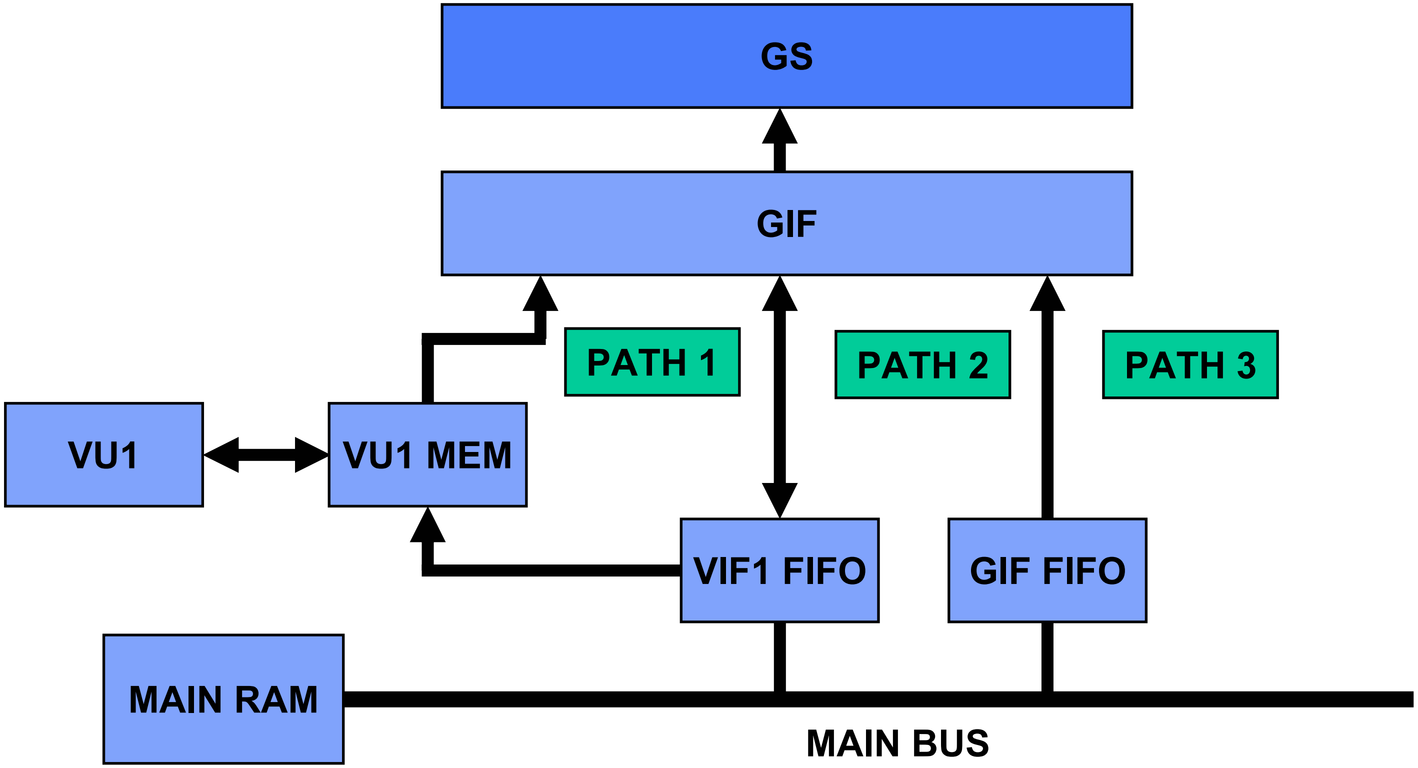
# Draining MFIFO with VIF1

- What can cause the MFIFO to become full?
  1. If GS FIFO is full, GIF doesn’t request any data
  2. XGKICK instruction will stall VU1
  3. VIF1 stalls on sync related instructions such as MSCNT and FLUSHA


SPR → MFIFO → VIF1 → VU1 → GIF → GS

# Geometry and Texture Syncing

- 1.2 GB/Sec Bandwidth to GS
- PATH1 for Geometry and PATH3 for Textures



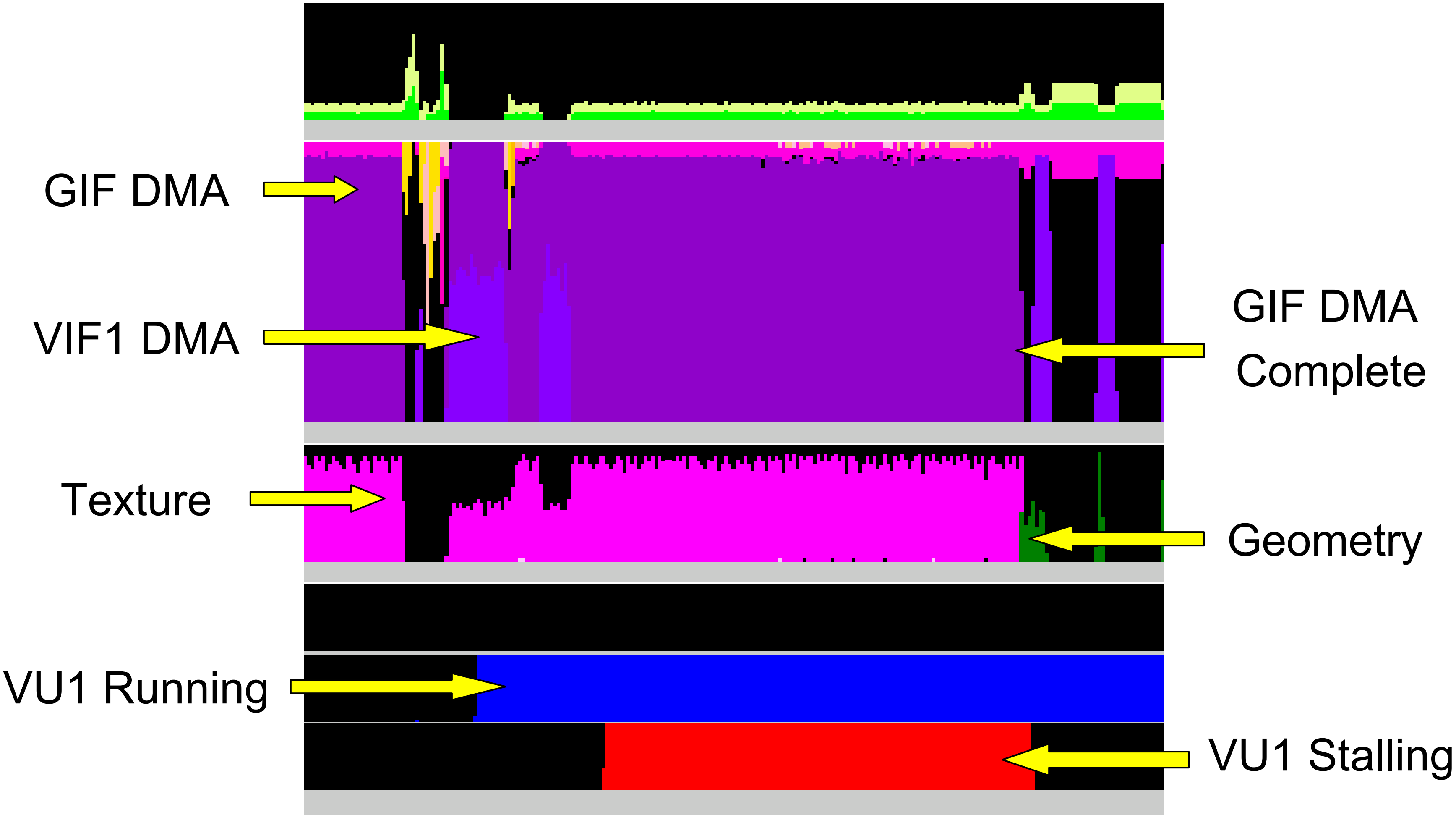
# Texture Transfer Paths

- PATH2
  - Advantages
    - Easy to transfer textures and set other GS registers
    - No geometry and texture data sync problems
  - Disadvantages
    - PATH1 will stall if PATH2 is still in progress
- PATH3
  - Advantages
    - Parallel DMA transfers through VIF1 and GIF channels
    - GIF can operate in 2 different modes when using IMAGE mode
    - Avoids PATH1 stalls when operating GIF in IMT mode
  - Disadvantages
    - Sometimes difficult to synchronize geometry and texture data

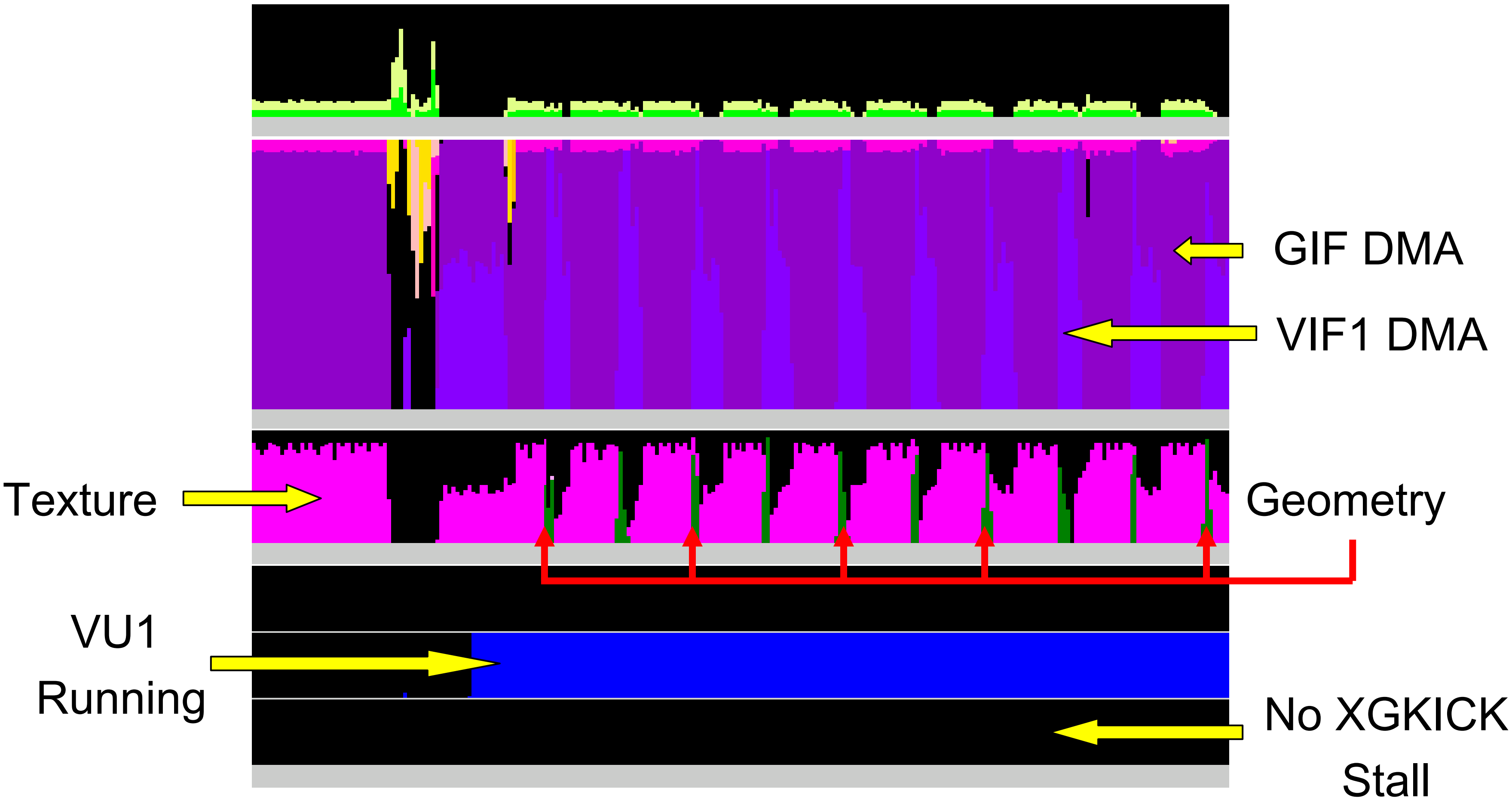
# GIF in Intermittent Mode

- What are the benefits?
  - Allows texture transfers via the GIF while VIF1 and VU1 continue to process data
- What are some things I should consider?
  - IMT Mode is good when loading large texture blocks
  - If GIF is constantly being occupied by PATH1 then texture transfer via PATH3 is reduced
  - Can’t draw and transfer textures at same time!
  - Batch textures together to limit overhead!

# GIF IMT Mode OFF



# GIF IMT Mode ON



# Packing Texture Data

- Pack 4-Bit and 8-Bit texture data
  - – 32-Bit textures provide maximum transfer speed
  - – 4/8-Bit textures must be converted by the GS
- Consider the transfer speed and block layouts
  - – 16 and 32-Bit pixel modes have very similar speeds

| Format | Size W | Size H | PATH2 MB/S | PATH3 MB/S |
|---|---|---|---|---|
| 32-Bit | 256 | 256 | 1090 | 1070 |
| 16-Bit | 256 | 256 | 1075 | 1050 |
| 8-Bit | 256 | 256 | 800 | 785 |
| 4-Bit | 256 | 256 | 385 | 380 |

# VCL Tool

- Application that simplifies Vu1 Programming
- Available for Linux and Windows
- Generates VSM source code
- Handles many tasks
  - Dual Pipeline processing
  - Loop unrolling
  - Register allocation
  - Instruction scheduling

# Vu0 Usage

- Transferring Data to Vu0
  - Cop2 connection you can transfer 1QW in 2Cycles
  - DMA transfer you can transfer 1QW in 4Cycles
- Processing Data with Vu0
  - Vu0 running Micro code
  - Triple Buffer Scratchpad memory
    - Transfer data to Block A
    - Process Block A and Transfer Block B
    - Drain Block A, Process B, Transfer C

# Geometry Data Transfer

- Reduce memory consumption and bandwidth
  - Remember Vector Unit register VF00.w = 1.0

### 4QW Per Vertex

| | | | |
|---|---|---|---|
| 1.0f | Z | Y | X |
| 1.0f | 1.0f | T | S |
| A | B | G | R |
| 1.0f | Nz | Ny | Nx |

### 3QW Per Vertex

| | | | |
|---|---|---|---|
| A | B | G | R |
| Ny | Nx | T | S |
| Nz | Z | Y | X |
| 🟦 | 🟦 | 🟦 | 🟦 |

# Compress Geometry Data

- use the VIF to convert integer to float
- use the VU to convert integer to float

Compress 4 QW to 1.25 QW

| Vector | Unpack Mode | VU Instruction |
|---|---|---|
| X,Y,Z | 16 Bit | ITOF0 |
| S,T | 16 Bit | ITOF12 |
| RGBA | 8 Bit | ITOF0 |
| Nx,Ny,Nz | 16 Bit | ITOF15 |

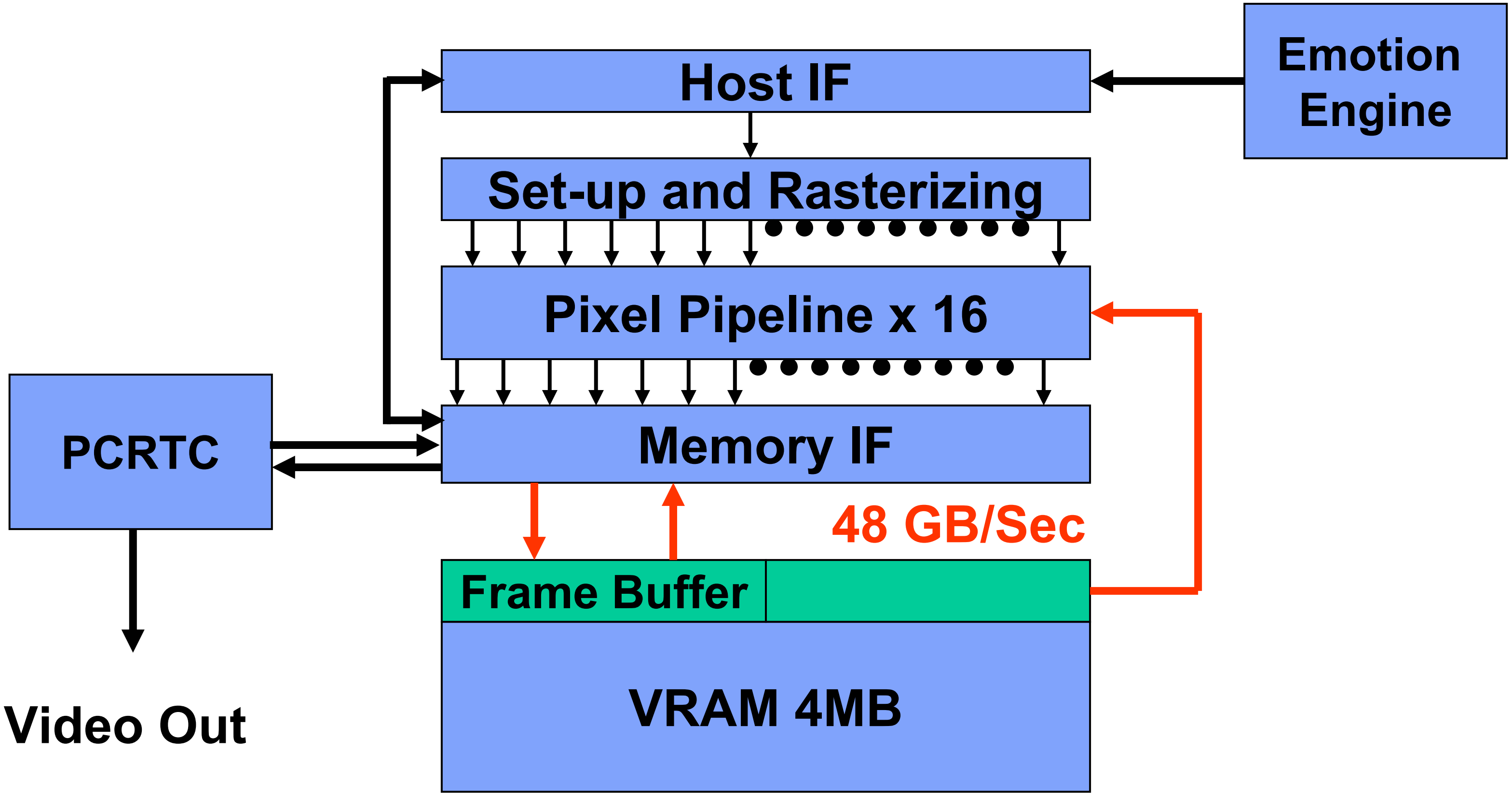
# GS Frame Buffers

- Total of 4 MB of Embedded DRAM
- Draw, Display, Z and Texture Buffers
- What are some recommended buffer sizes?
  - PAL (512 x 512), NTSC (512 x 448)
  - Progressive scan support with full height buffers
- 2-Circuits of the GS to reduce interlace flicker
  - alpha blend odd/even fields at no cost

# GS Capabilities

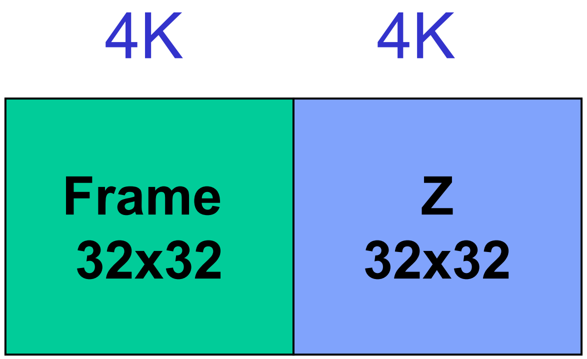
- Bandwidth
  - Massive total of 48 GB/Sec
  - Frame Buffer 38.4 GB/Sec
  - Texture Buffer 9.6 GB/Sec
- Drawing Speed
  - 16 Pixel for non-textured (2.4 Gpixels/Sec)
    - 75M Flat shaded Triangles/Sec
  - 8 Pixel for textured (1.2 Gpixels/Sec)
    - 37.5M Textured and Gouraud shaded Triangles/Sec

# GS Pipeline


Emotion Engine
Host IF
Set-up and Rasterizing
Pixel Pipeline x 16
PCRTC
Memory IF
48 GB/Sec
Frame Buffer
VRAM 4MB
Video Out

# GS Frame/Z Cache

- Quick Page refills!
  - 8192bits per cycle
  - 8K page buffer refilled in 8 GS cycles



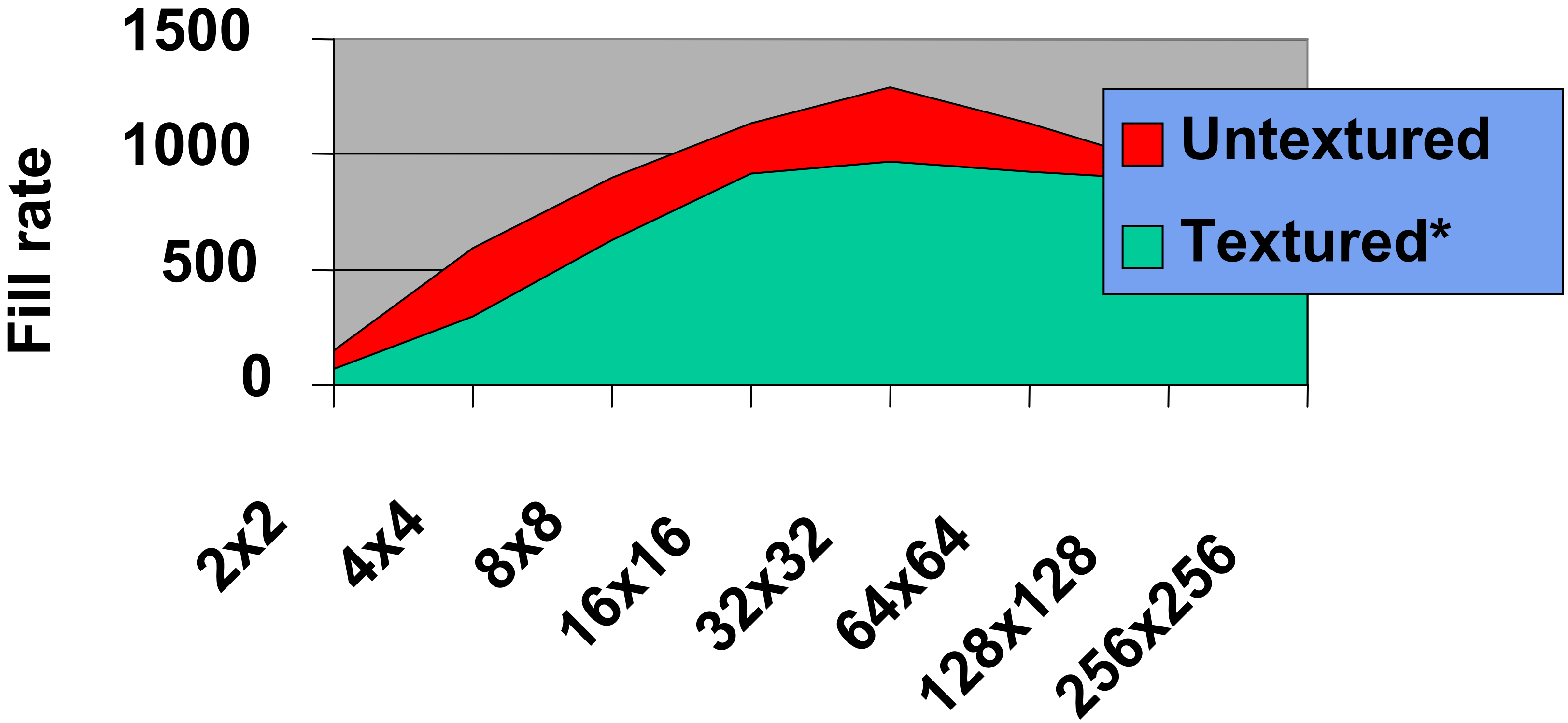
# Reducing Frame Page Misses

- Fill rate is roughly constant if varying height
- Wide Primitives will cause page misses
  - Use 32 Pixel wide strips to reduce page misses
- Rarely drop below 1Gpixel/Sec if miss occurs
- Primitives using textures greater than a page size are usually more of a problem
- 8Bit texture page is 128x64

# Texture Fill Rates

- Texture Page misses have biggest effect
  - Subdivide large texture co-ordinate ranges
  - Keep mip-maps in the same page
- Texture reduction reduces the fill rate
  - 32 pixel wide strips won’t increase performance
  - Texel read becomes bottleneck
- Texture expansion doesn’t affect fill rate

# Fill Rate VS Triangle Size



*Texture is on cache without reducing size

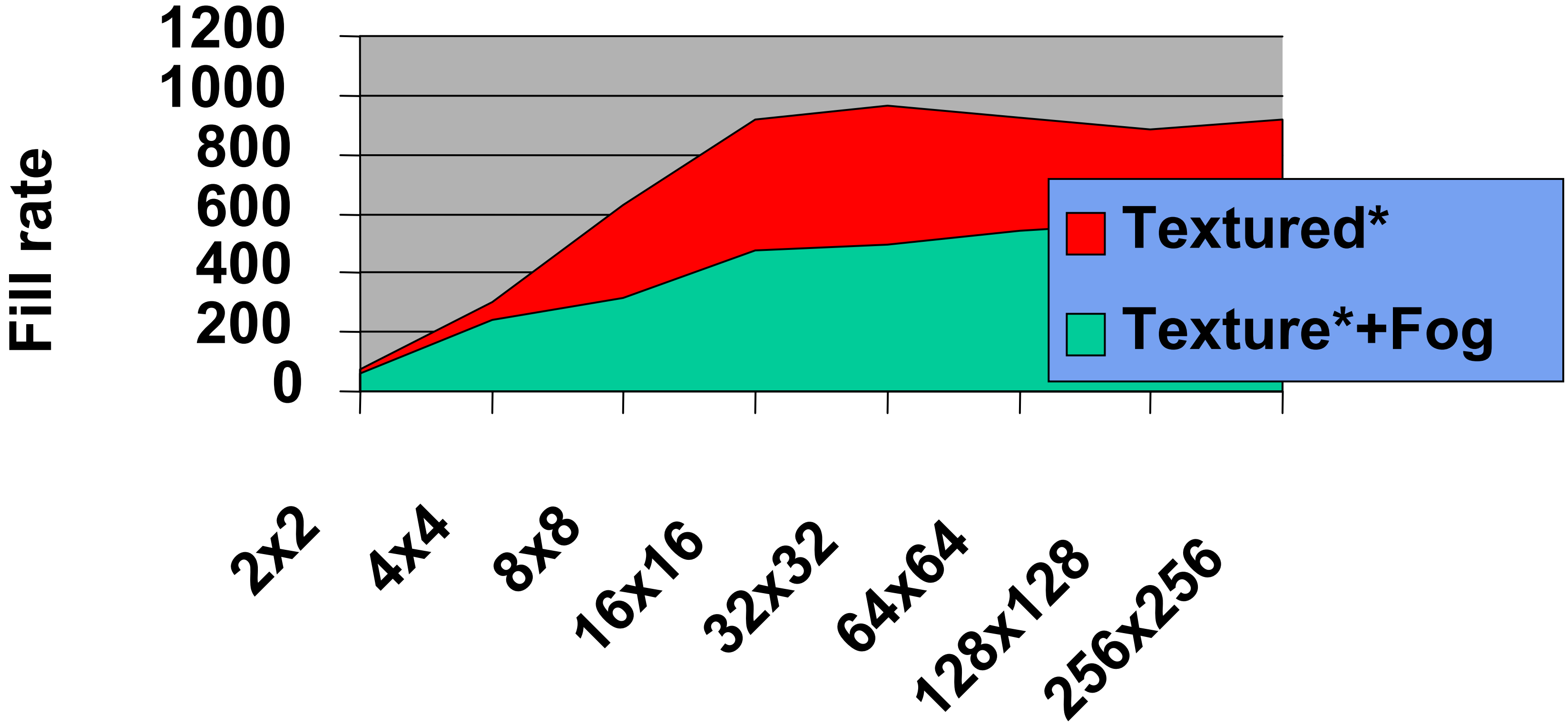
# Level Of Detail

- Make better use of LOD!
  - 5000 polygon model may result in just 50 visible pixels once projected onto the screen
  - there’s also no point having detailed textures that are going to be shrunk so much
- Mip Mapping
  - Improve visual quality
  - Mip maps in different pages can cause multiple texture cache reloads

# Multi-Pass Rendering

- GS Alpha Blend operation is free!
- Maximum textured fill rate is 1.2G Pixels/Sec
  - Limit number of passes (4 passes = 300M P/S)
- Fur rendering
  - Reduce passes when object in distance
- Bump-mapping is possible
  - Technique requires full screen passes
- Back face cull to reduce GS stalls

# GS Fogging



*Texture is on cache without reducing size

# Alternative Fogging

- Technique 1
  - 1st pass draw a textured polygon
  - 2nd pass alpha blend gouraud shaded polygon
- Technique 2
  - Post-process and perspective correct fogging
  - Move bits 8-15 of Z-Buffer into Alpha of Draw Buffer
  - Alpha blend full screen gouraud shaded polygon onto Draw Buffer

# CPU Optimisations

- Emotion Engine Core
  - FPU (Coprocessor 1)
  - Vu0 (Coprocessor 2)
  - 16K Instruction Cache
  - 8K Data Cache
  - 16K Scratch-Pad Memory
- Instruction Set
  - 64Bit MIPS III and some MIPS IV
  - 128Bit Multi-Media

# Multi-Media Instructions

- 128-Bit Multi-Media Instructions
- Parallel Processing
  - 64 bits x2, 32 bits x4, 16 bits x8, 8 bits x16
- Image format conversions
- Sound decompressing
- Pack DMA packets
  - Convert PACKED mode to REGLIST mode
  - Smaller data, faster DMA transfers!

# Use of Data Cache

- Data Suitable for the Data Cache
  - Data that is frequently read or written repeatedly
  - Data with a high degree of locality
- Don’t use Data Cache for
  - Data that gets used only once
  - Big chunks of data larger than 8K

# Reduce Cache Misses

- Prefetch instruction to load data beforehand
- Reduce the size of your code for I$
- Use Uncached memory for data r/w only once
- Performance Counter Lib to measure misses

# Scratchpad Memory

- 16K of high-speed memory (access directly)
- 2 dedicated DMA Channels (toSPR/fromSPR)
- SPR DMA provides best throughput
  - 100% Occupy and 85% Send
- Data Suitable for the SPR
  - Frequently used data where speed is a priority
  - Big chunks of data can be Double Buffered on SPR memory

# CD/DVD Optimisations

- Align destination buffer on 64 Bytes
  - Increase performance by 25%!
- Combine files into a PAK file to reduce files
- Avoid seeking when you could be reading
- Load the most data you can per read
  - Combine IOP modules and load into EE

# Summary

- PA will push developers to the limit!
- Parallel Texture and Geometry Transfer
- DMA is flexible and very powerful!
- Take into consideration GS page sizes
- Vector Unit 0 and Scratchpad memory
- Check assembler output of generated code

# Further Information

- Contact Information
  - SCEE Booth Exhibition Stand #9