Hard-Tuning the PS2

Jason G Doig
Senior Engineer, SCEE R&D

GameDevelopers Conference Europe

What I'll be covering

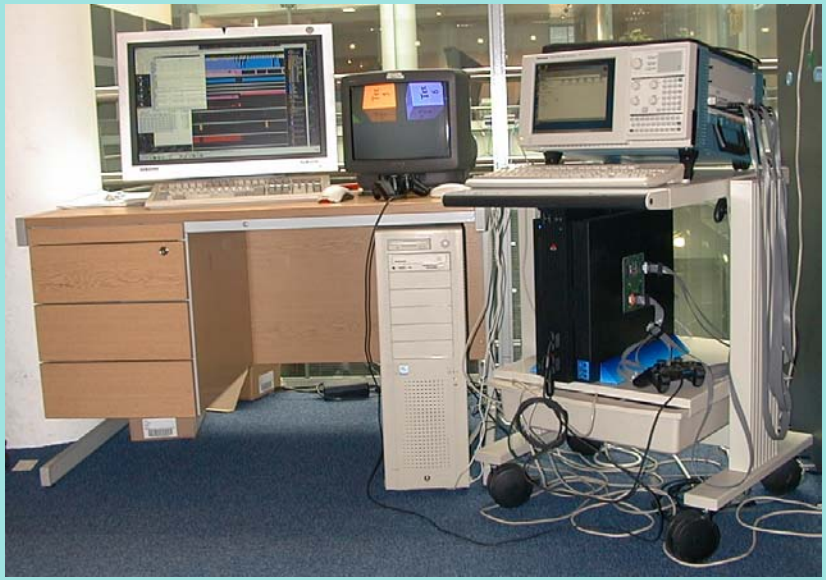
- Me
- "Performance Analyser" Introduction
- Specific PS2 Performance Issues
  - EE Problems
  - GS Problems
  - Case studies

Introduction

- My background
  - 7 years in development
    - Most recent title “Dropship” on PS2
  - >1 year in Sony R&D
    - Support
    - Technology development
    - Consultancy

Performance Analyser

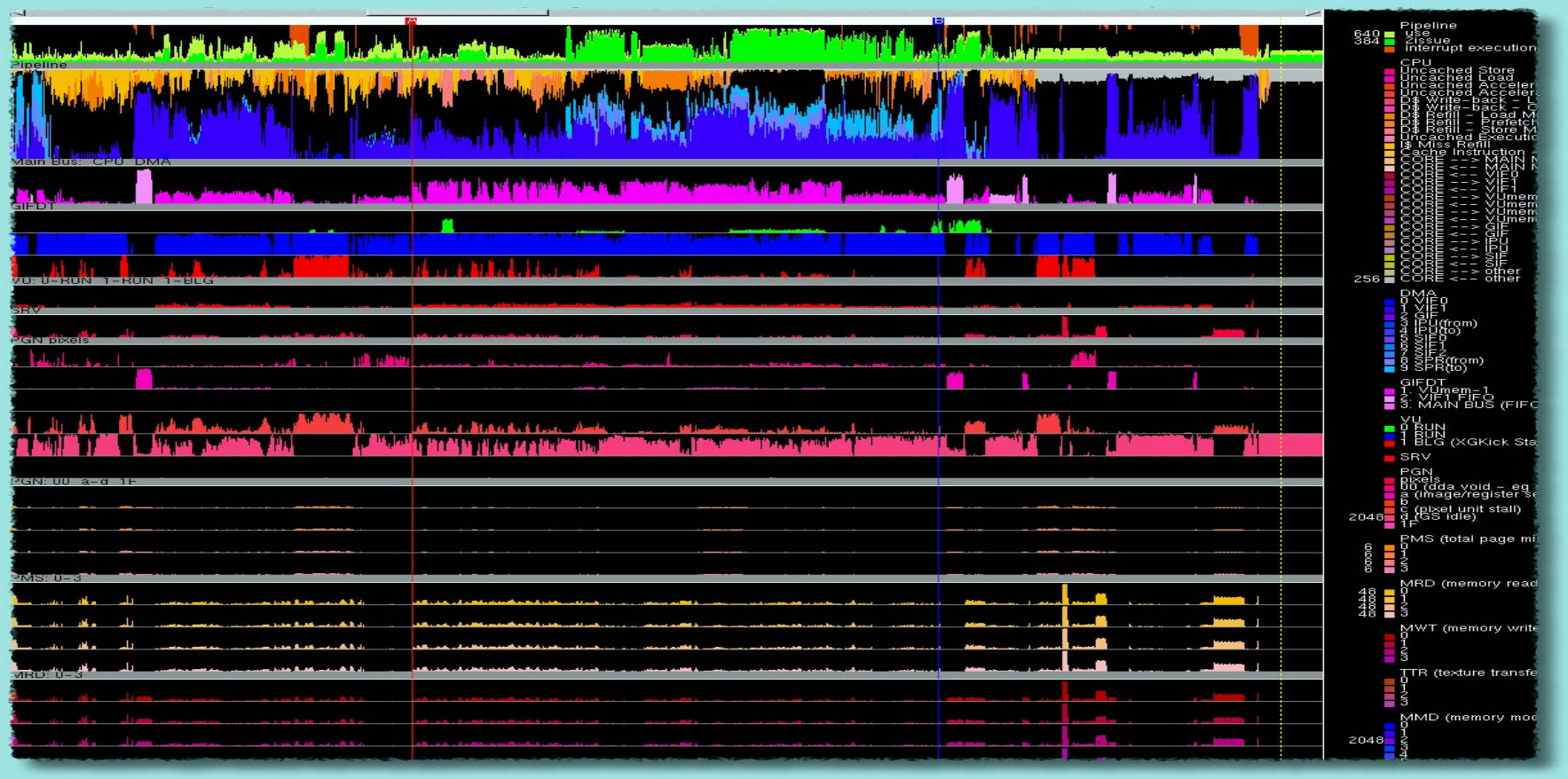
- Customised PS2 devkit
- Captures over 100 signals
  - EE and GS debug output
- Cycle accurate
  - ~100ms capture duration
  - ~7.5 frames at 60Hz
- Software analysis package



What the PA doesn't do

- No data buses captured
  - Internal states only
- No true address bus capture
  - "most" long jumps captured, nothing else
- No IOP information at all
  - At least not in the prototype…

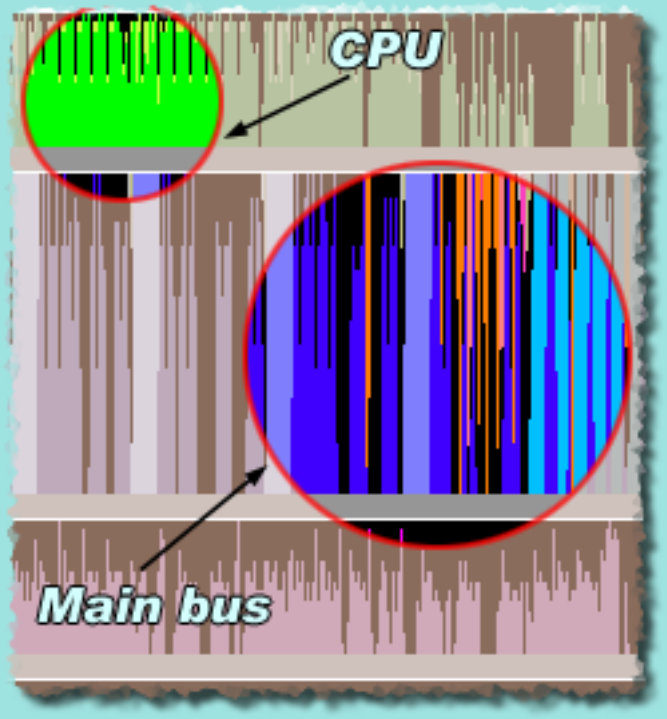
What the PA does



GameDevelopers Conference Europe

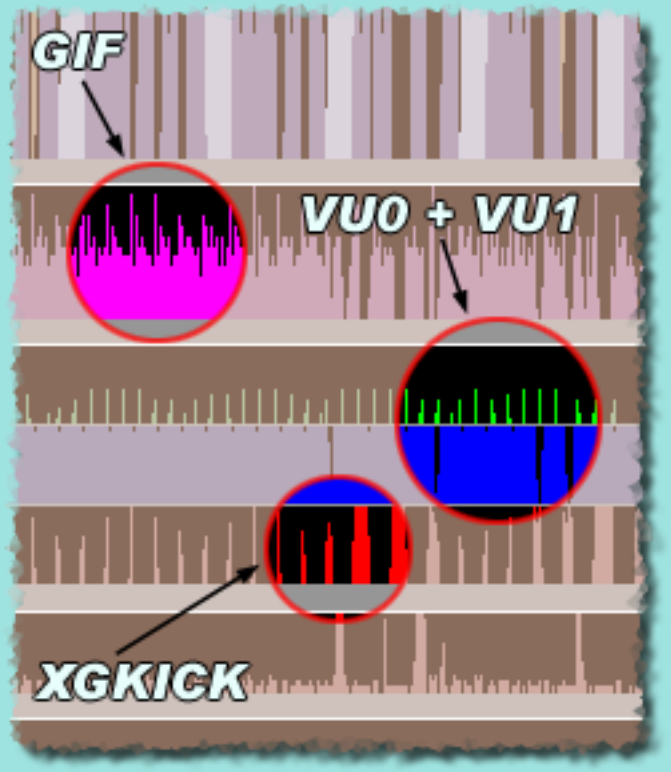
What the PA does

- CPU signal capture
  - CPU pipeline activity
    - Single or dual issue
    - Interrupt display hack
  - Main bus
    - CPU
    - DMA



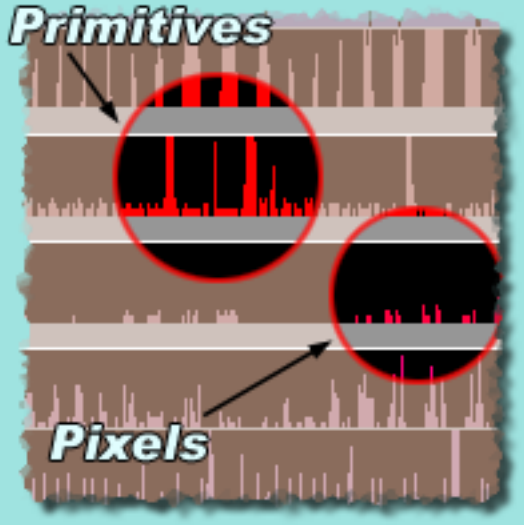
What the PA does (continued…)

- GIF usage
  - 3 Paths
- Vector unit status
  - Running
  - XGKICK stall



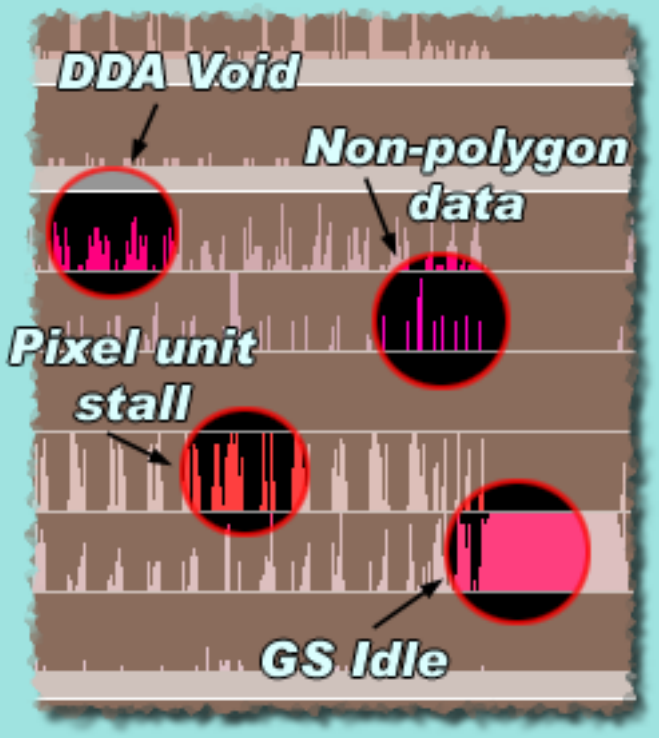
What the PA does (continued…)

- GS signal capture
  - Primitives rendered
    - Doesn’t count zero-area polys
  - Pixels output
    - Half-full for textured ops
    - Only those drawn, not scissored



What the PA does (continued…)

- DDA void
  - Busy, but no pixels
    - Scissor, narrow poly
- Non polygonal data
- Pixel unit stalls
  - Many reasons!
- GS idle time



Balance

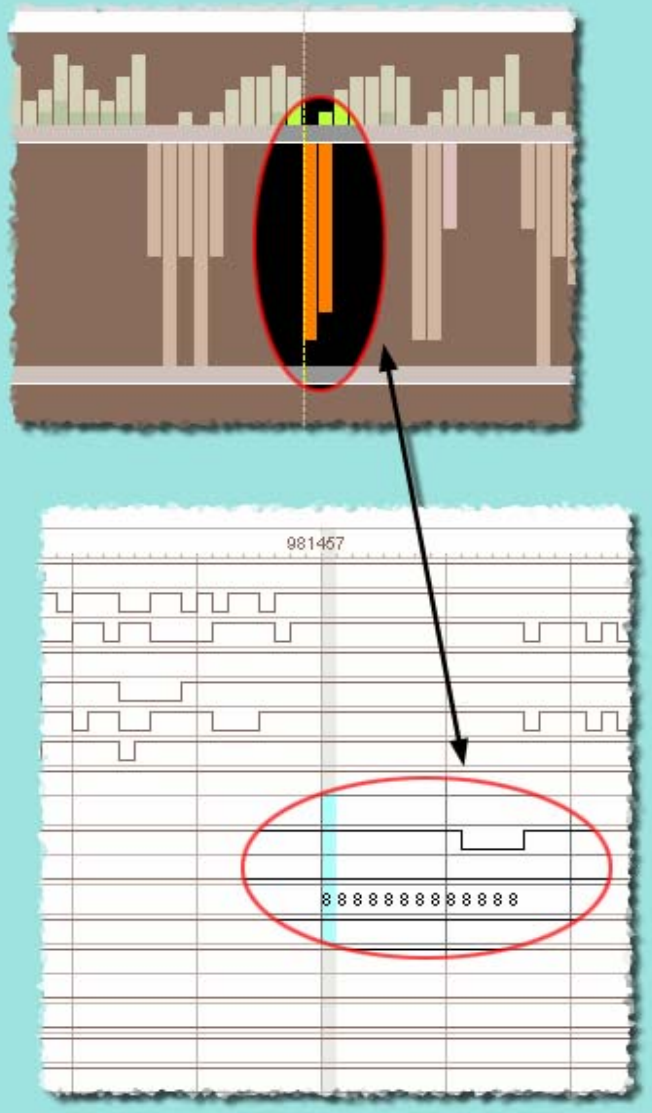
- PS2 designed for parallelism
  - EE and GS
  - CPU and DMA
    - Shared memory bus, VIF, GIF etc.
  - CPU and VUs
    - Simultaneous independent processing units
  - CPU and VU pipelines
    - 2 instructions per cycle

Common Problems

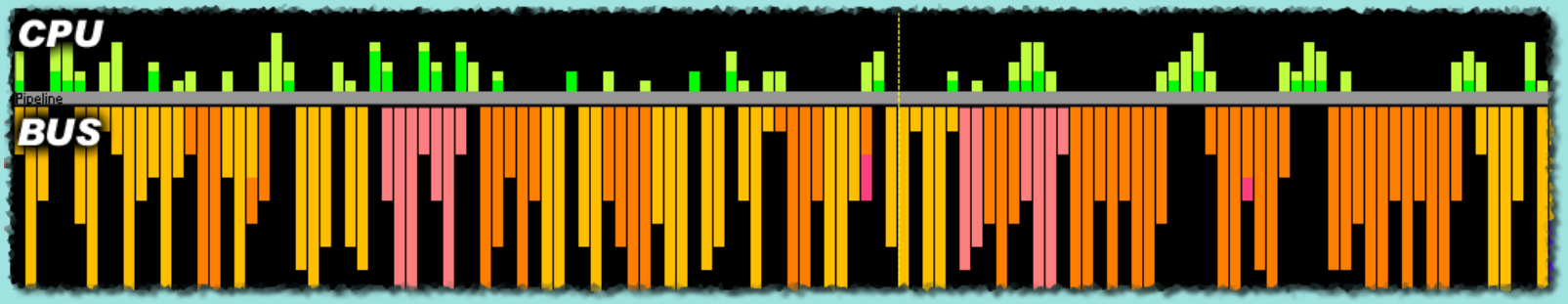
- Poor CPU efficiency
  - Code generation
  - Memory utilisation
- Drawing stalls
  - Bad texture usage
  - Frame buffer page misses

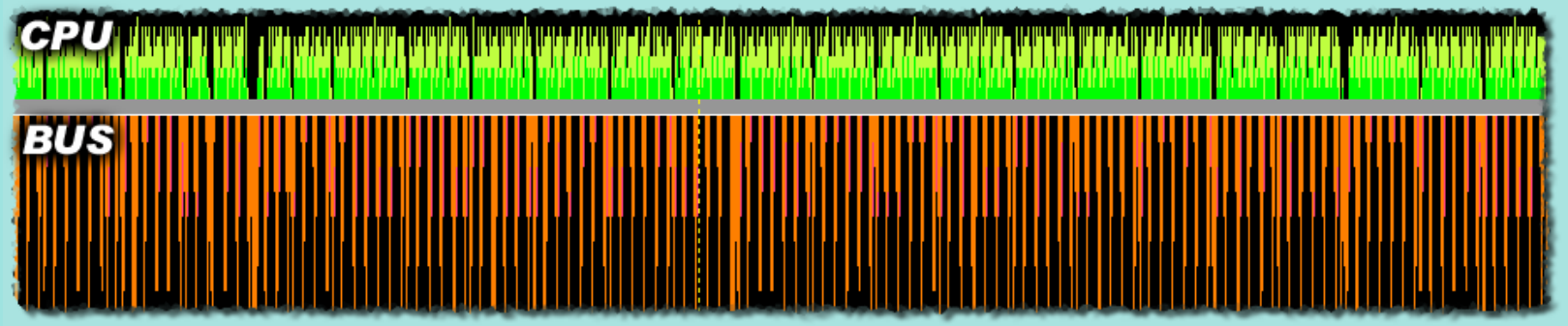
Memory Access

- Memory performance lower than in theory
- Single cache miss
  - 4 QW, should be 4 cycles
  - Actually takes 13!
  - 9 cycles setup cost!



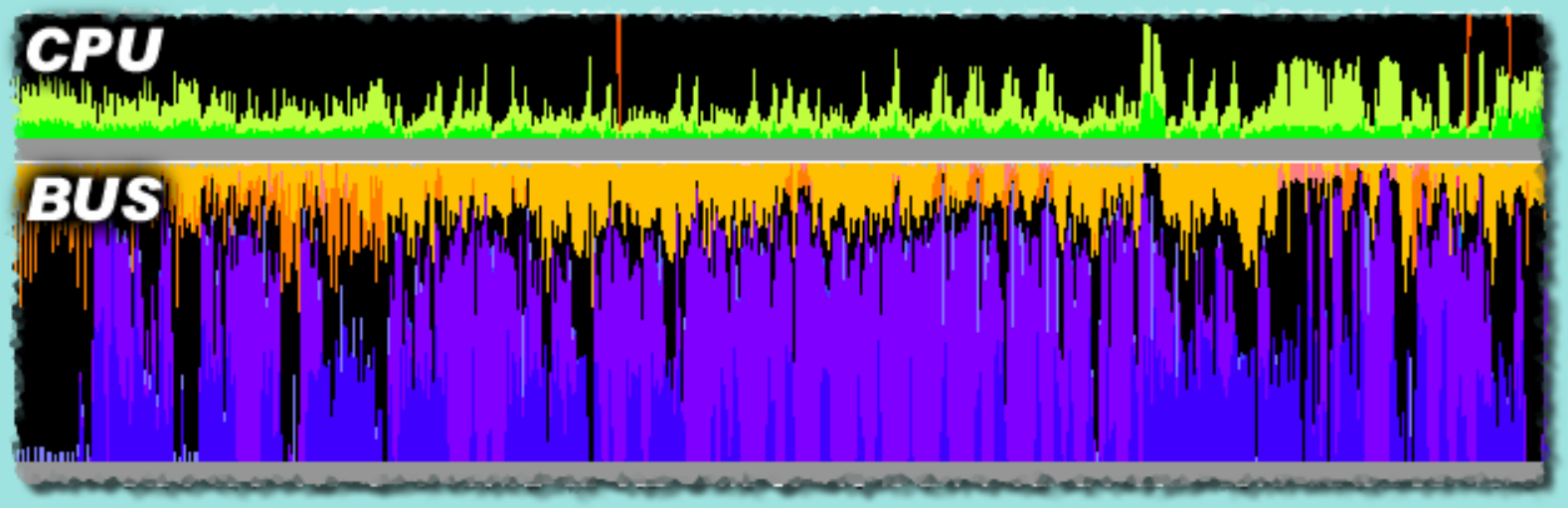
Bad (and good) CPU performance





Code generation

- Compilers do not take advantage of PS2
- Typical CPU performance very low



Dealing With Poor Code

- Look at the output
  - Try to rearrange code to help the compiler
- Investigate other code generators
- Inline assembler
  - Sometimes there is no substitute

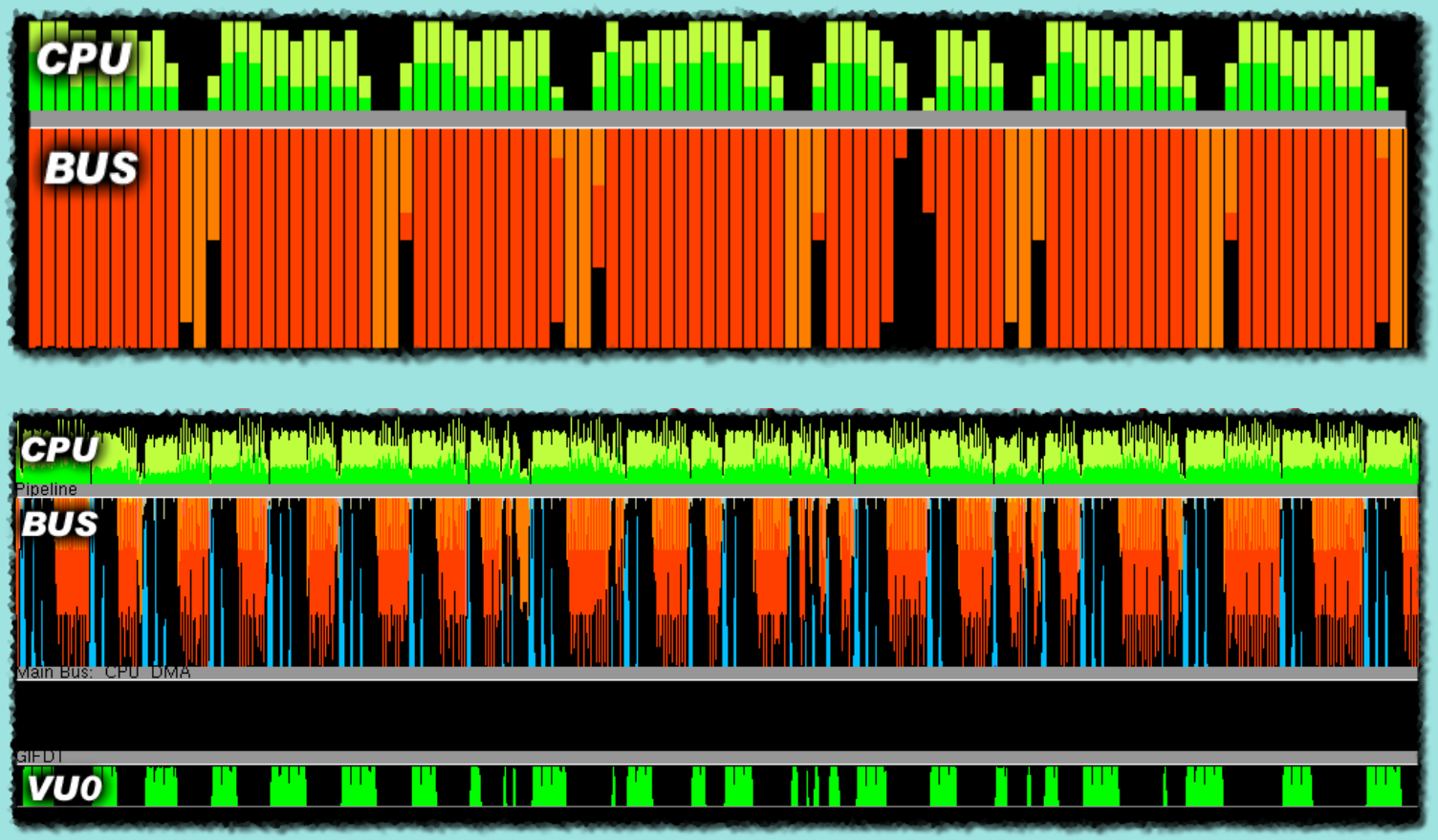
Instruction Cache Problems

- Lots of jumps
  - Call overhead
- Too much inline
- Too much code

Specific Code Generation Problems

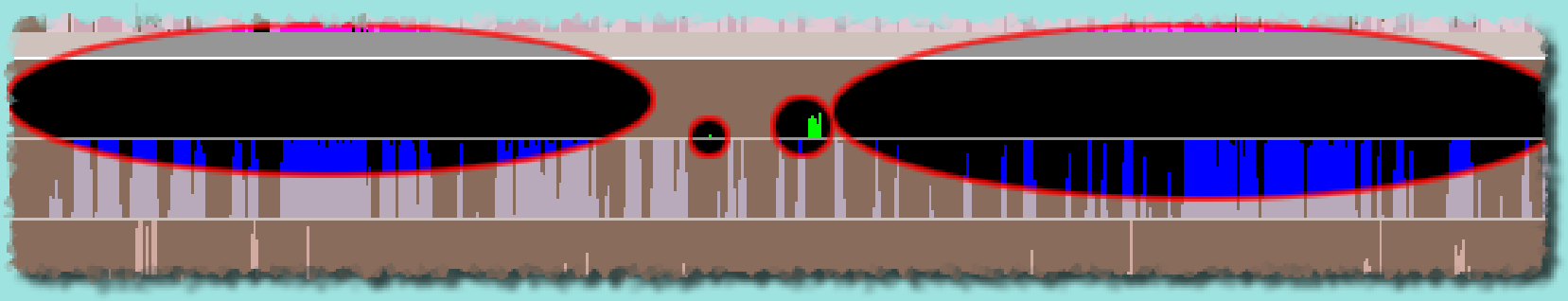
- Data cache thrashing
  - Organise data to minimise random access
  - Frequently accessed memory good
- Alternatives
  - Uncached / Uncached accelerated
  - Scratchpad
    - Only ~100% efficient memory access available!

Uncached Accelerated Usage



VU0 abuse

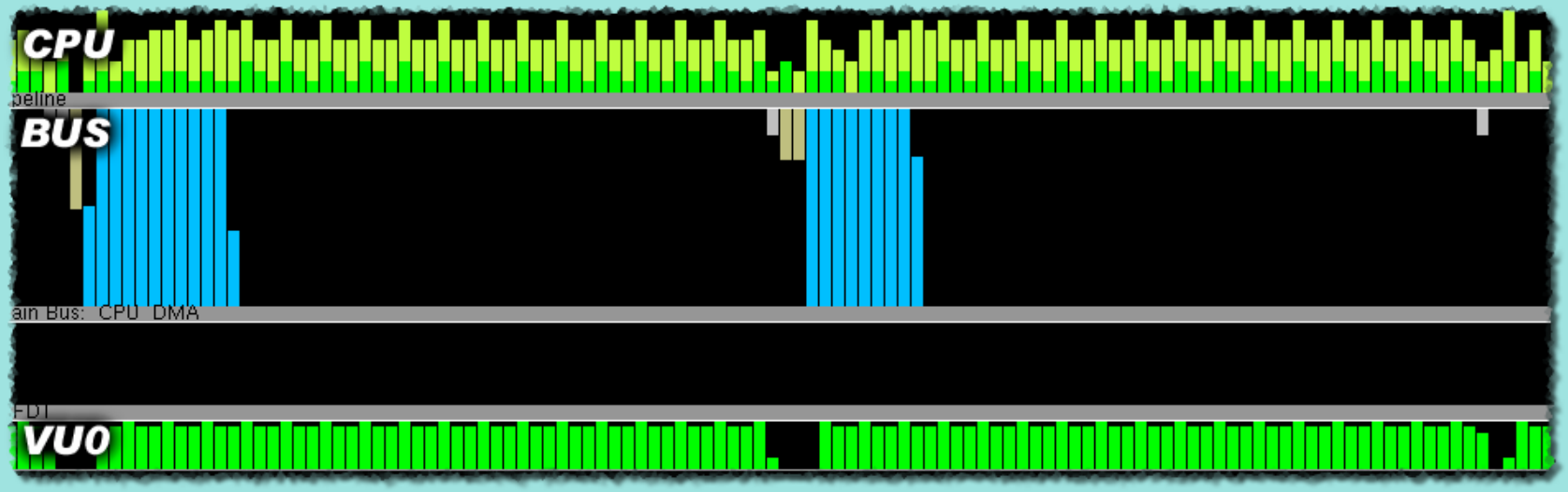
- VU0 heavily underused
- Macro-mode is evil



What can you do with VU0?

- Anything maths related
  - Matrix and vector operations are ideal
  - Consider even scalar operations
    - But think about vectorising them ☺
- Use VU0 memory for lookup tables
  - Or for a matrix stack

Look! VU0!



What Slows Down the GS?

- Polygon Size
- Pixel Engine Stalls
  - Texturing
    - Size
    - Filtering options
  - Vertex fogging

Polygon Size

- Frame buffer “swizzled”
- Arranged in 2d pages
  - Scan converting can cross pages
  - Pages are buffered
- Minimise stalls by “stripping”
  - Use vertical strips 32 pixels wide

Pixel Engine Stalls

- Texture Size
  - Page buffer is 8k
  - Effectively 2D
  - Minimum page size varies per format
    - See manuals!
  - Frequent texture page hits very expensive

More Pixel Engine Stalls

- Bilinear shrink
  - Bilinear expand is “free”
  - Bilinear shrink is not
    - Pixel units arranged 4x2
    - Fetching more than ~16 texels per cycle stalls
    - Rapidly drops to 50% performance
  - Trilinear even worse

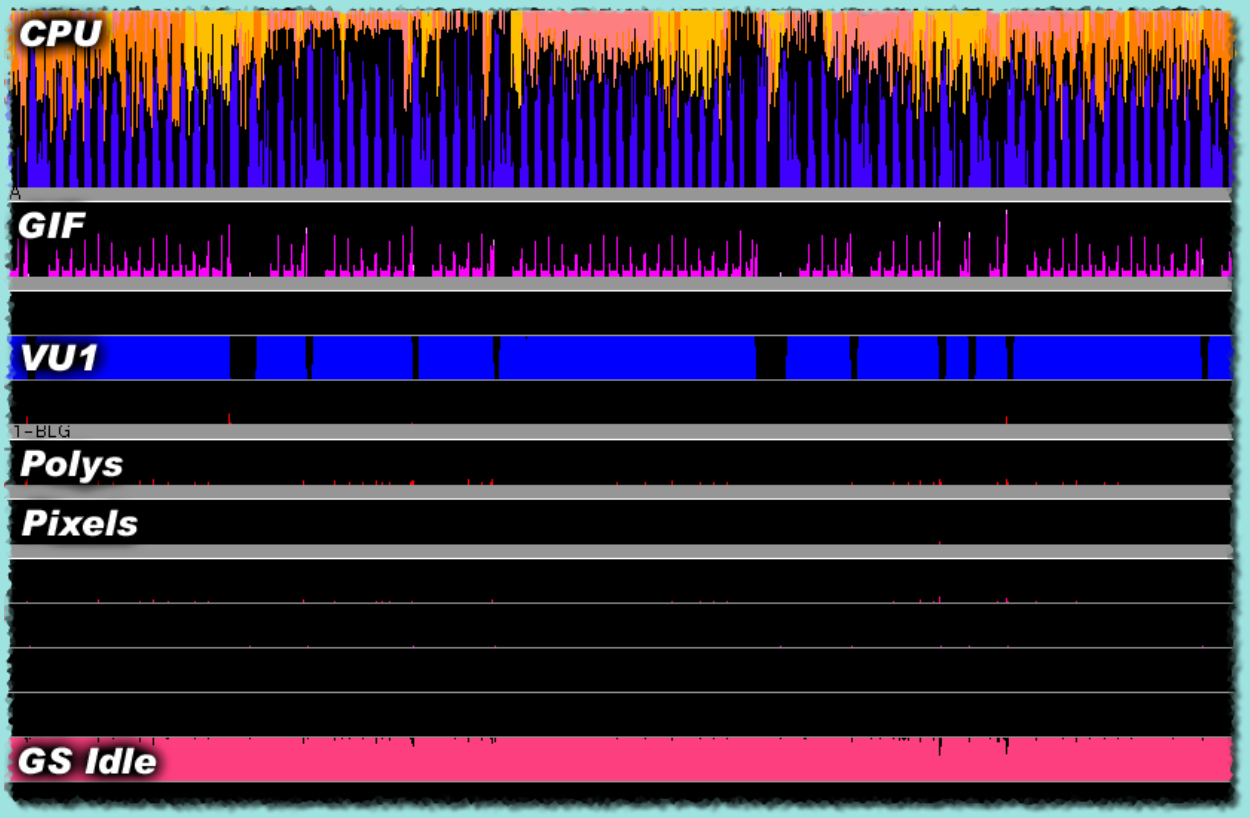
Optimising VU1 Usage

- Optimise VU1 code
  - Obvious, I know…
- Or move work outside VU1
  - VU0
  - EE core
    - If it's not stressed already…

Optimise Your DMA Data

- Compressed data
  - Let the VIF do the work
- Instancing
  - Make artists do the work ☺
  - Can dramatically reduce DMA usage

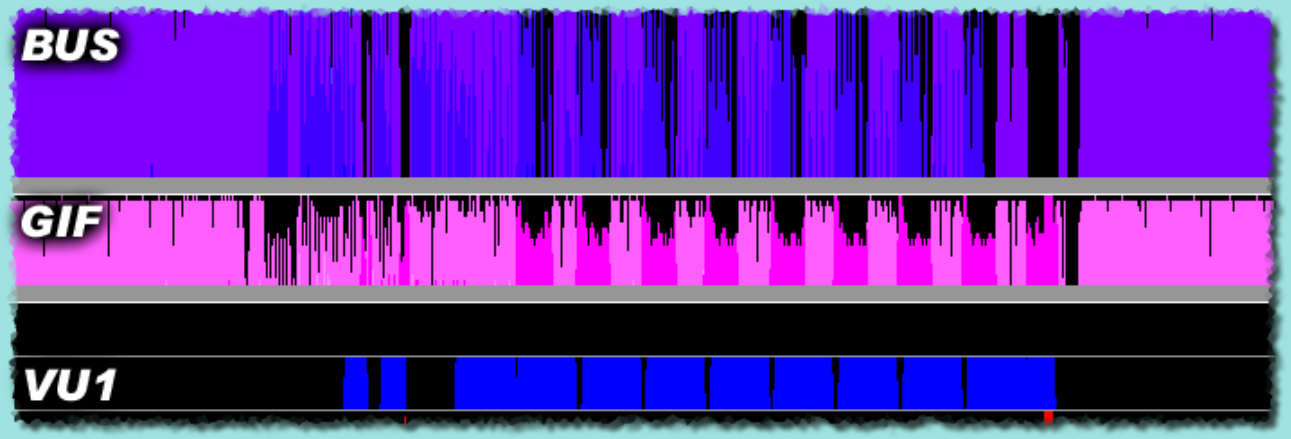
Meet a VU1 Bound Renderer

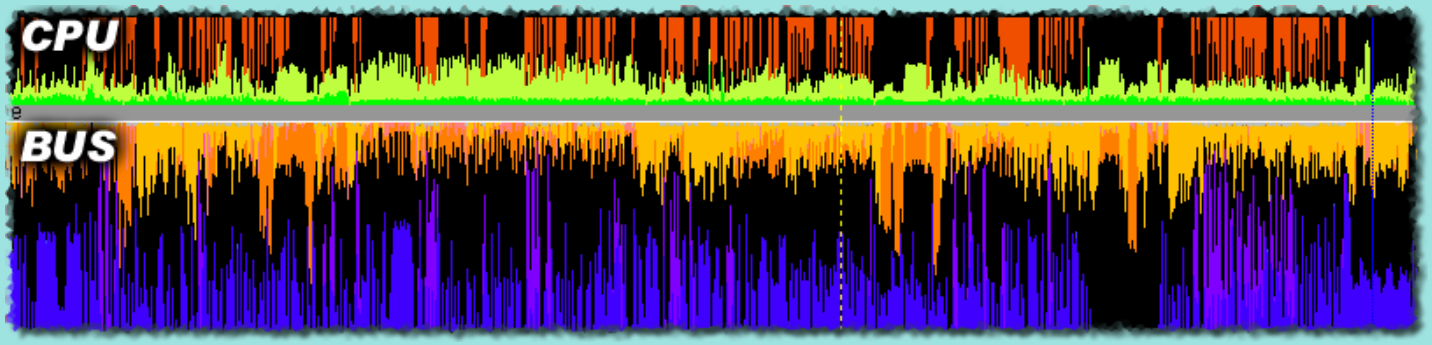


Textures

- PATH 2 easy
  - But rubbish
- PATH 3
  - Mask path 3 syncing
    - Complex, naïve, no CPU
  - Interrupt based syncing
    - Easy, potentially expensive
- GIF PATH1 / PATH3 clashing

Texture Uploading

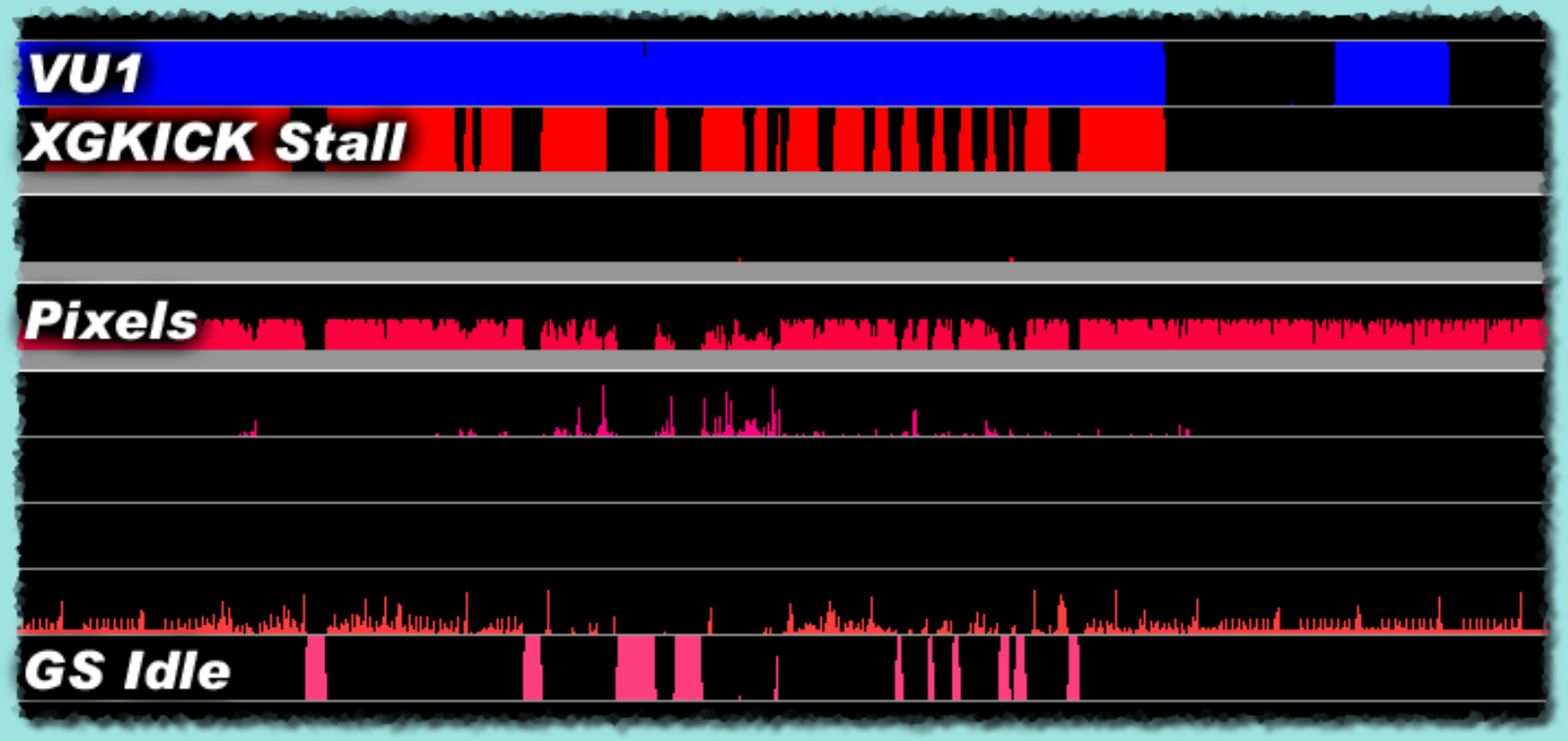




Particle Systems

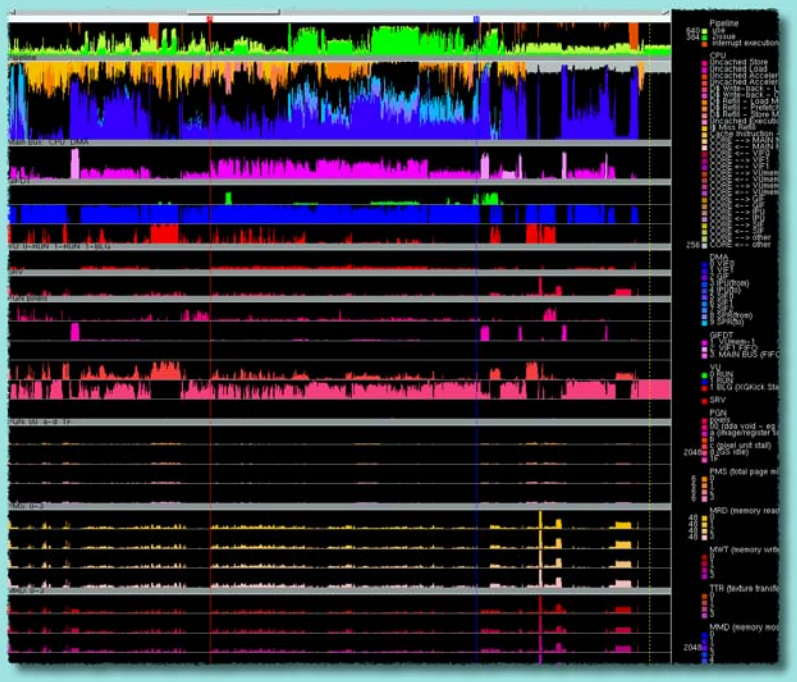
- Mostly simple primitives
  - Sprites, billboard polys
- VU code very basic
- Fine when distant
- Expensive up close
  - GS fill-rate increases dramatically

Particles continued…



It is possible… honest.

- Remember this?
  - 10..20 million polys
  - >50% CPU usage
  - >80% dual issue



Summary

- Balance
  - Get everything working together
- Use more VU0
  - More micro-mode, less macro-mode
- Make better use of the GS
  - Smaller textures, controlled filtering, subdivide
- Make better use of the bus
  - Use scratchpad, instancing, etc.

Questions?