Hardware Overview

SDK Version 1.0

Contents

Figures

Equations

Tables

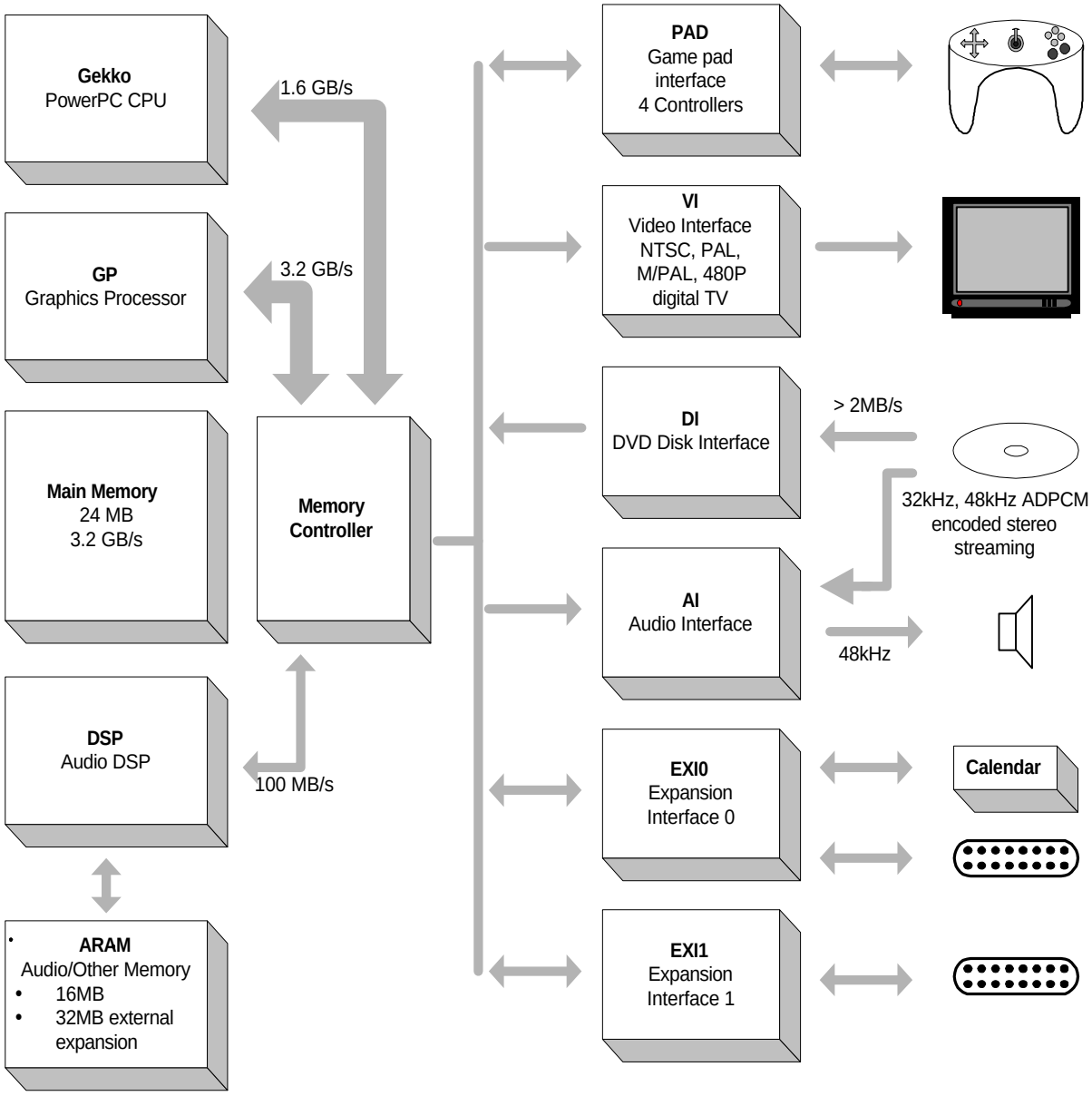

Figure 1 NINTENDO GAMECUBE functional blocks and busses

1 Gekko CPU

The Gekko CPU include a IBM PowerPC 750 processor core with additional functional units for performance enhancement. The Gekko CPU has the following features:

- 400Mhz internal clock operation.
- 200Mhz 64-bit bus to main memory (1.6GB/s peak bandwidth).
- 32KB 8-way set associative L1 Icache.
- 32KB 8-way set associative L1 Dcache (16KB data scratchpad configurable).
- Super-scalar microprocessor with five execution units: 2 integer units, 1 floating point unit, 1 load/store unit and branch unit.
- DMA unit servicing 16KB data scratchpad; 15-entry DMA request queue.
- Write-gather buffer for writing graphics command lists to the graphics chip.
- Embedded 256KB 2-way set-associative L2 unified cache.
- Two (2) 32-bit Integer Units (IU).
- 1 Floating Point Unit (FPU), 32-bit and 64-bit.
- FPU supports Floating Point Paired Singles (FP/PS).
  - FPU supports `ps_madd` (paired-single multiply-add). Most FP/PS instructions can be issued every cycle and complete in three cycles.
  - Simultaneous conversion between fixed and floating point numbers while loading/storing FPU registers without performance penalty.
- Branch Unit offers static and dynamic branch prediction.
- Out-of-order execution; i.e., when an instruction stalls on data, subsequent instructions can continue to issue and execute. All instructions complete in correct program sequence to preserve program logic.

The Gekko CPU has the following features to minimize CPU stalls on data fetching and maximize computational throughput:

- Non-blocking caches.
- Branch prediction.
- 8-way set-associative caches.
- 256KB L2 cache.
- Out-of-order execution feature.

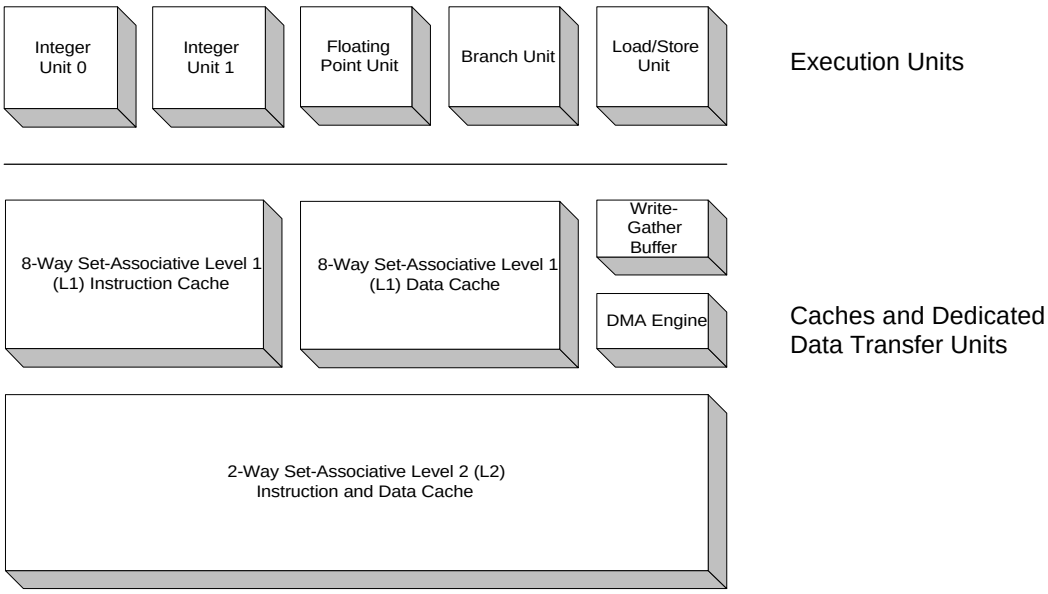

Figure 2 CPU functional blocks

1.1 L1 data cache

The 32KB L1 data cache has two modes of operation:

1. 32KB 8-way set-associative L1 data cache.
2. 16KB 4-way set-associative L1 data cache + 16KB data scratchpad buffer.

The DMA engine transfers data between the 16KB data scratchpad and main memory. The engine can issue up to 15 pending requests. The data scratchpad is directly programmable, so the game programmer can eliminate cache miss unpredictability by fetching the data prior to processing.

The Gekko CPU's 4-way or 8-way set-associative data cache significantly minimizes cache misses and cache thrashing.

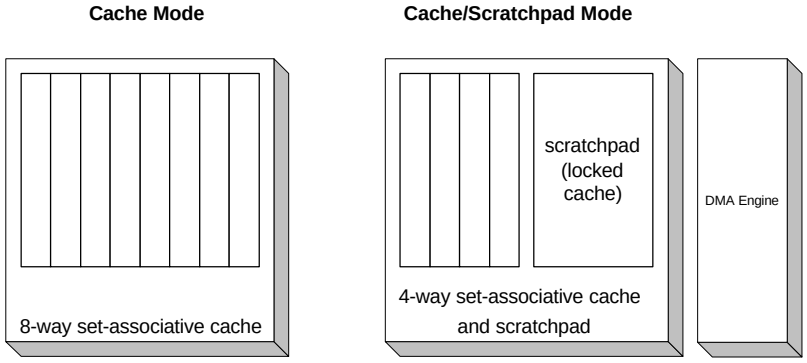

Figure 3 Data cache configuration

1.2 L2 caches

The amount of code and data in a modern console game program is increasing steadily, as is the complexity of its working sets (a working set is defined as the code and data used in one loop of a game event).

The Gekko CPU provides a large 256KB L2 cache to keep more code and data close to the CPU computation units. This features gives the Gekko ability to work continuously through 400 million cycles, rather than losing cycles by waiting frequently for memory.

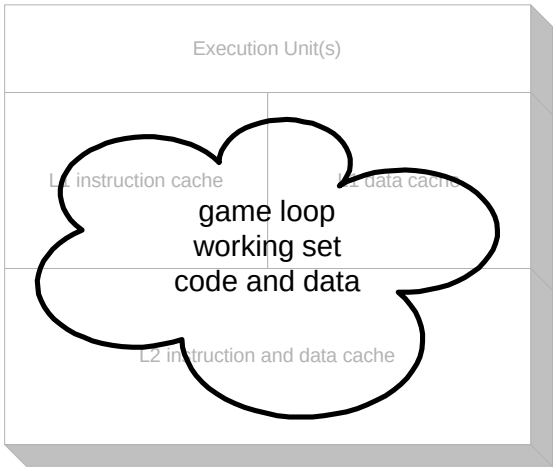

Figure 4 L1 and L2 caches can store a working set's code and data

1.3 FPU performance

Most floating point operations can be issued every cycle and can complete in three cycles. These operations include add, sub, mul, and multiply-add instructions. Reciprocal and reciprocal-square-root estimation instructions are available as well.

- One-cycle execution rate for common floating point instructions.
- Floating point instruction latency completion in three cycles.
- Two-cycle (minimum) execution for floating point reciprocal (1/x) and reciprocal square root (1/sqrt(x)) operations; completion in four cycles.

For more details, please refer to the IBM Gekko RISC Microprocessor User's Manual, Chapter 6.7, "Instruction Latency Summary."

1.3.1 Paired singles

The FPU can also execute paired-single (PS) vector instructions. This means we can perform two floating point instructions per cycle for many instructions, including multiply-add (`ps_madd`); however, the data must be in vector format (i.e., one 64-bit word contains two single-precision floats).

- Two floating point instructions execute per cycle; latency completion in three cycles.
- Two floating point reciprocal (1/x) and/or reciprocal square root (1/sqrt(x)) instructions execute every two cycles (minimum); completion of two instructions in four cycles.
- A single-precision floating point instruction can use the lower single-precision result of a paired singles register. This feature gives the CPU the ability to interchange results between paired singles and single-precision instructions rapidly.

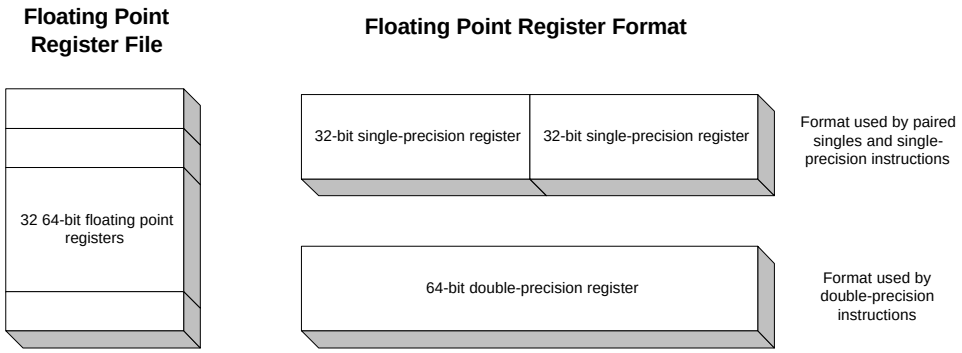

Figure 5 Paired singles register format

1.3.2 Free fixed and floating point conversions

The Gekko CPU includes special fixed point and floating point conversion hardware in its load/store unit. This hardware can perform the following conversions:

- Free fixed point-to-floating point conversions, if loading an FP/PS register.
- Free floating point-to-fixed point conversions, if storing an FP/PS register.
- Programmable decimal point in fixed point format.

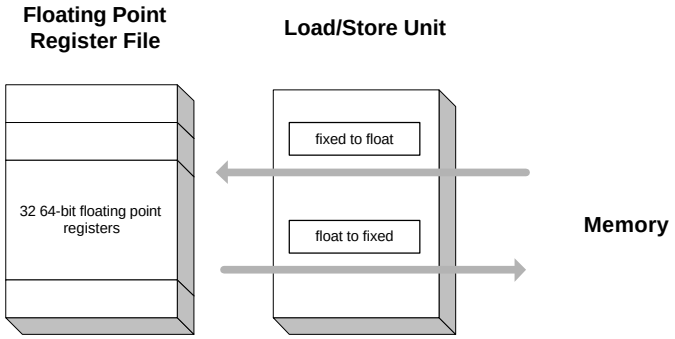

Figure 6 Floating/fixed point conversions

For more information about this Gekko CPU feature, please refer to the IBM Gekko RISC Microprocessor User's Manual, sections 2.1.2.9, "Graphics Quantization Registers (GQRs)"; 1.2.2.4.3, "Load/Store Unit (LSU)"; and 2.3.4.3.12, "Paired Single Load and Store Instructions."

1.4 Out-of-order instruction execution

Modern microprocessors can achieve high speeds by using multiple processor cycle pipeline stages for computations. The Gekko CPU runs at 400Mhz with five different execution units, making it able to complete a lot of work in a very short time. If the CPU had to wait for all instructions to finish sequentially, many computing cycles would be lost; therefore, the Gekko CPU has an out-of-order instruction execution capability to increase the number of operations possible per cycle.

This feature enables instructions to execute while previous instructions are stalled waiting for memory. The following example instruction sequence shows how out-of-order execution can help to increase performance.


```
float   instruction 1 regA          // floating point instruction, output to regA
float   instruction 2 regA          // floating point instruction, input need regA
integer instruction 3 regB          // integer instruction, output is regB
branch  instruction 4 regB          // branch instruction, input is regB
```

Code 1 Out-of-order instruction handling

In this example, **instruction 2** becomes blocked while waiting for **instruction 1** to complete (remember that the typical floating point instruction has a three-cycle latency). However, **instruction 3** has no dependencies on either of the two previous instructions, so this one issues and completes in one cycle (as most integer instructions do). Branch **instruction 4**, which needs the result of **instruction 3** in order to determine branch conditions, can also start execution.

This example shows that you can complete the determination of branch while waiting for independent floating point instructions to complete. Many other code sequences can benefit from out-of-order execution.

An additional hardware "completion unit" guarantees that the code will complete in the correct sequence. For example, **instruction 3** is not allowed to *complete* before **instruction 2** does; however, **instruction 3** still gets all of its work done, in effect, while **instruction 2** is waiting for execution. This feature typically increases performance by 25-30% when compared to a processor without an out-of-order instruction execution feature.

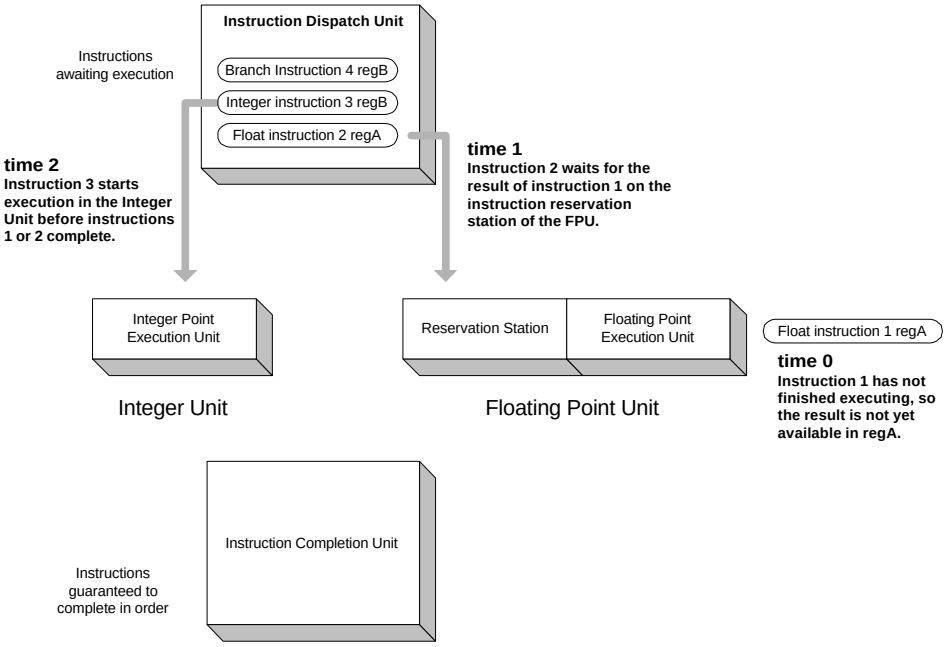

Figure 7 Example of out-of-order execution

1.5 Branch prediction

Pipelining the execution stages allows for high-speed CPU performance. However, this method leaves the system open to the risk of stalling whenever the code branches because the execution pipeline must wait until the branch condition is resolved.

The Gekko CPU provides special branch prediction hardware to significantly minimize such stalling. It keeps a history of recent branches and uses this history to predict code sequence, which gives the CPU the ability to process instructions beyond the current branch before branch conditions have completed. The branch prediction hardware

tracks all but the last loop branch conditions, thus it can eliminate all branch-related stalls in the pipeline except for the last branch out of the loop.

The Gekko CPU has a 512-entry table to contain branch history information, meaning that it can keep up to 512 branch program counter (PC) addresses to record prior branch results. These results act as “hints” that help the Gekko CPU predict which “direction” the next branch(es) will take. The CPU uses this information to continue executing code beyond the current branch. If the prediction turns out to be wrong when the current branch condition resolves, then the Gekko CPU knows to “rewind” all the work and return to execute the correct branch.

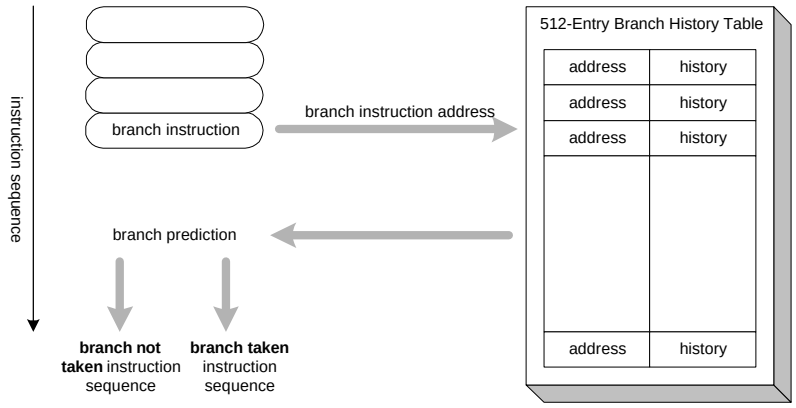

Figure 8 Branch prediction feature

2 1TSRAM main memory

With so many high-speed processors in the NINTENDO GAMECUBE (GCN), we need a high performance memory system that offers both high bandwidth and low latency. (Latency is often overlooked, which can result in many processors "waiting" for data from memory subsystems and thus stalling computation.) The GCN uses high-performance 1TSRAM for main memory, which offers the following advantages:

- Random access with at full peak performance.
- 3.2GB/second peak bandwidth.
- 32-byte transfer packet.
- 24MB total capacity.

To understand 1TSRAM in more detail, we should briefly review DRAM architecture.

2.1 DRAM bank architecture

All modern DRAMs use bank architecture, with two to four banks typically available. Bank architecture is very easy to understand. It functions just like caches.

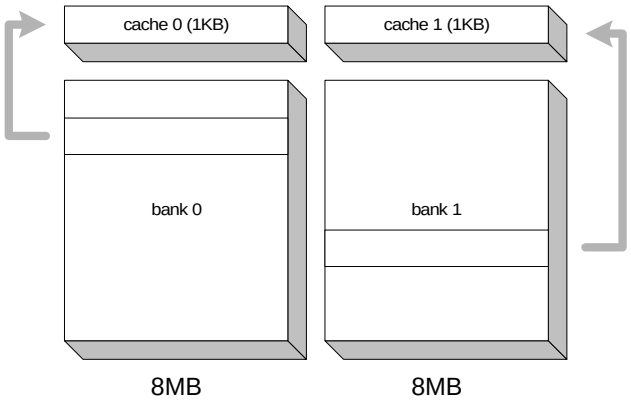

Figure 9 DRAM bank architecture (16MB DRAM with two banks)

The caches for each of the DRAM banks are typically only 1KB of contiguous addressable memory. Each time a memory request falls outside a 1KB cache region, the memory set must signal the processor to wait. It is therefore difficult to minimize bank cache misses during program execution. Code instructions and data are not close to each other in memory, nor are the large amounts of texture and graphical object memory necessary for a visually-complex game environment. This means that a significant amount of processor performance is wasted waiting for memory.

2.2 1TSRAM architecture

1TSRAM has no banks. Every location in memory is randomly accessible, so a processor requesting data does not stall in memory access to any location.

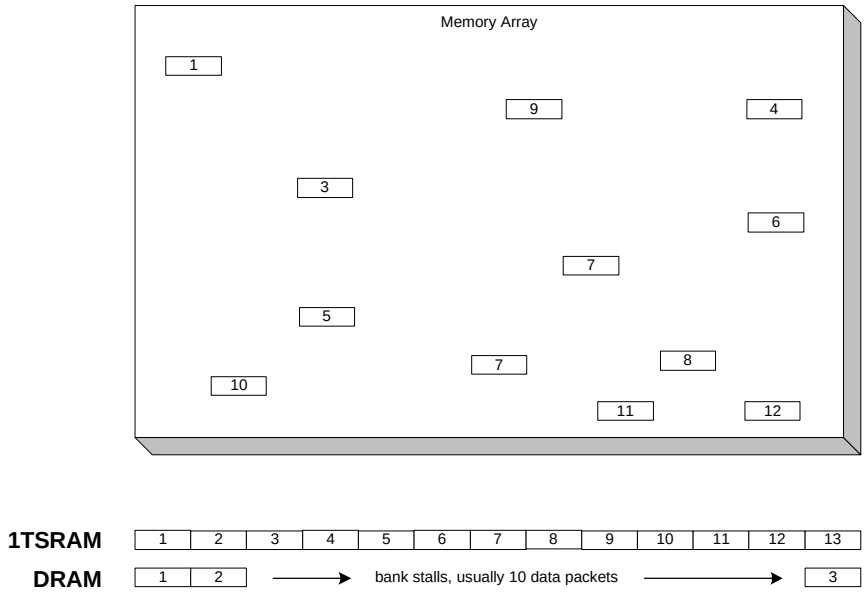

Figure 10 1TSRAM and conventional DRAM comparison

3 Graphics Processor (GP)

The GP has the following basic performance specifications:

- 200MHz operation.
- 25M/33M/40M polygons/second peak, depending on feature selection.
- 800M pixels/second peak.

Many features may be enabled at these peak performance rates.

Features	Performance
1 vertex color + 1 light + 1 texture	25M polygons/second
No vertex color + 1 texture	33M polygons/second
1 vertex color + no texture (Gouraud shading)	40M polygons/second

Table 1 Polygon performance

All of these polygon performance times include fogged, Z-buffered, transparency-blended pixels. For polygons with textures, the following additional operations can be performed at peak fill rate of 800M pixels/second.

- Trilinear mipmap-filtered, perspective-corrected texture.
- S3TC compressed textures.
- Single-texture map.

3.1 Functional units

Here is a brief description of the role played by each functional unit:

- The Command Processor (CP) interprets commands generated by the CPU. It supports vertex array indexing, making it easy to construct geometry using indices for positions, normals, colors, and texture coordinates. Duplication of graphical vertex data is not necessary.

- The Transform Processor (XF) performs Model-to-World, World-to-Viewport, texture coordinate, and normal transformations. The XF can also compute local diffuse lighting, infinite specular lighting, and planar texture projection. Clipping is also performed here.

- The Texture Processor (TX) can interpret many texture formats. It also performs texture caching and texture filtering, and it can apply multiple texture maps per polygon.

- The Texture Environment Processor (TEV) can combine the colors generated by the XF and the multiple textures generated by the TX. The TEV is a more powerful version of the Color Combiner (CC) in the Nintendo 64 (N64) system.

- The Pixel Engine (PE) performs blending, Z-buffering, and antialiasing operations. Z-buffer rejection can be performed prior to texture application, thereby maximizing performance.

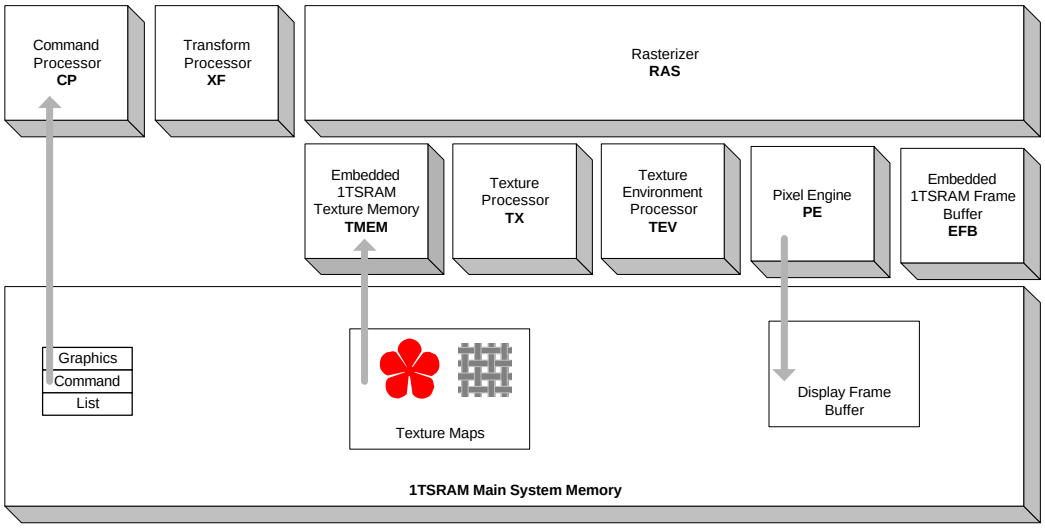

Figure 11 Graphics Processor (GP) blocks

3.1.1 Embedded memory

Embedded memory refers to memory that is included (embedded) on the same silicon chip; in this case, the Graphics Processor. When the GP is applying textures and using bilinear or trilinear mipmap filters, it requires up to eight texels for every pixel rendered. Texture access during rasterization is a non-linear memory access operation, so TMEM uses this embedded memory technology to eliminate performance bottlenecks.

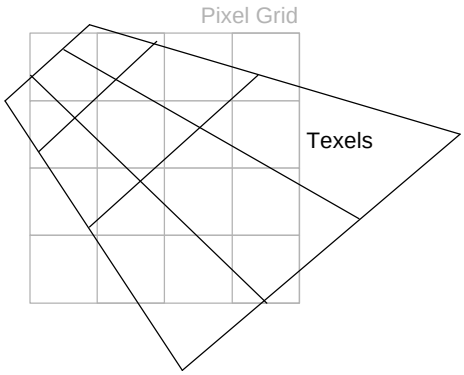

Figure 12 Non-linear data relationship between pixel rasterization and texture memory access

When drawing many polygons, especially small ones such as particle systems, the GP will draw rapidly to different parts of the screen, which results in near random memory access patterns. The embedded frame buffer (EFB) thus uses 1TSRAM to gain high speed random access performance.

3.1.2 Embedded 1TSRAM memory

Traditionally, SRAM offers a 10X performance increase over DRAM; however, SRAM has always been bigger than DRAM—traditional SRAM uses six transistors (6T) per bit of storage—so silicon chips could not contain large amounts of it. The GCN uses a revolutionary one transistor (1T)-per-bit SRAM technology. 1TSRAM gives The GCN the ability to include SRAM performance at the same size as DRAM.

Using embedded 1TSRAM for the frame buffer means that pixel Z-buffer and color blending operations are complete free. Furthermore, the GCN rasterizer can render into the 1TSRAM frame buffer without stalling, whereas DRAM stalls can occur during rendering and thus lower the total performance.

3.2 Command Processor (CP)

The CP handles a wide range of vertex and primitive data structures, from a single stream of vertex data (containing position, normal, texture coordinates, and colors) to fully-indexed arrays. Any vertex component can be index-referenced or inlined directly in the command stream, thus enabling efficient data processing by the CPU. Rather than being restricted to a rigid graphics display data structure which would result in lost performance, the CPU can perform the calculations naturally.

For example, a CPU lighting algorithm must generate only a color array from a list of normals and positions (see Figure 13). The CPU executes a simple `for()` loop to process a list of lighting parameters to generate the color array; there is no need to follow a triangle list display data structure. Likewise, there is no need to format the data for display after finishing. The data can be consumed naturally.

The CP has a one level-deep display list, meaning that the top level command stream can "call" the display list, but only one level down. This is excellent for any pre-computed commands and instancing of geometry.

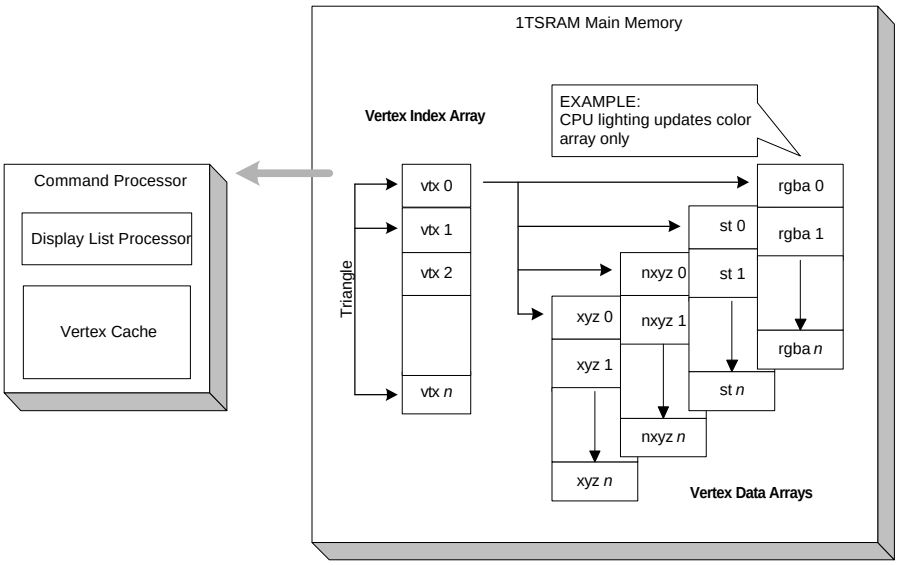

Figure 13 Command Processor (CP) blocks

3.3 Transform Processor (XF)

The XF is a hardware 3D geometry transformation pipeline. It performs typical 3D transform operations at a peak rate of 25M to 40M polygons/second[1]. Such transforms include:

- Model-to-World transform: 10+ matrices can be loaded. This supports stitching, a type of skinning operation.
- World-to-Screen transform.
- View frustum clipping.
- Backface polygon rejection.

[1] This peak rate assumes all polygons are triangles. It also assumes an average rate of one vertex per triangle.

The XF can also perform many useful lighting and texture effects, including:

- One to eight lights. One light can be computed at a peak rate of 25M vertices/second. A light can be local diffuse or infinite specular. These calculations are computed per vertex.
- 2x4 or 3x4 texture transform matrices. For multi-texturing, ten or more 3x4 matrices can be loaded. One matrix transform will be performed at the peak rate of 25M vertices/second. Projected light and shadow textures can be implemented in this way.
- Bump map texture coordinates.

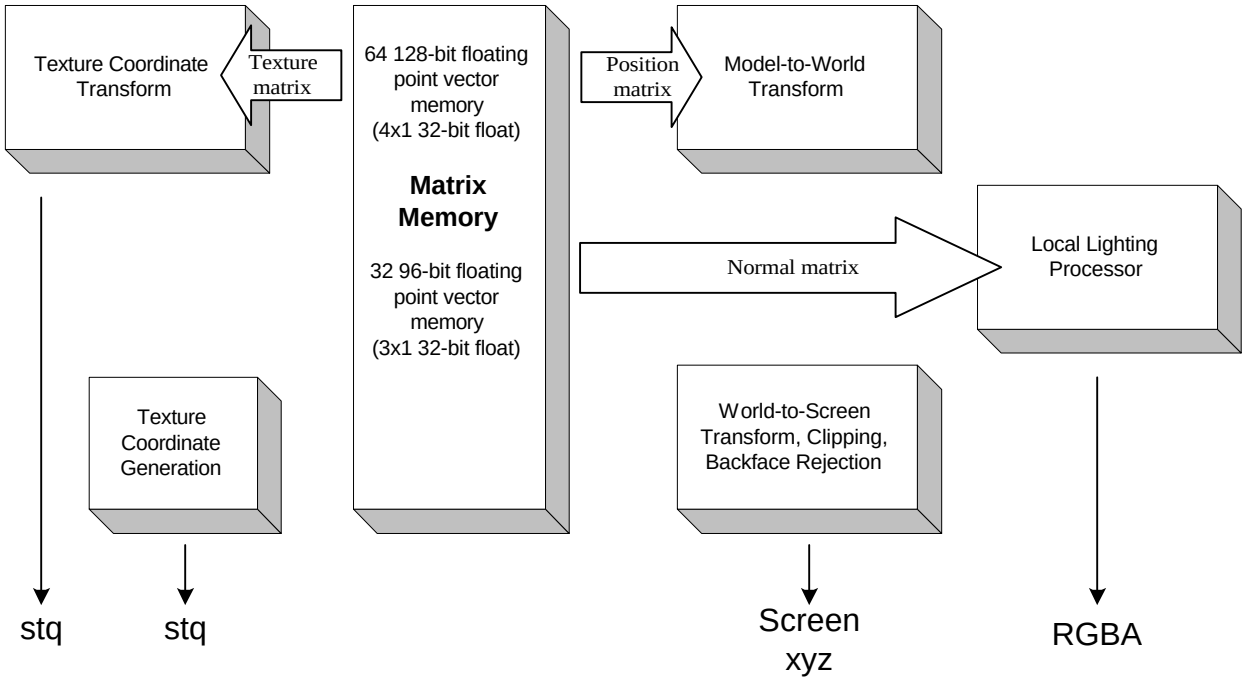

Figure 14 Transform Processor (XF) blocks

3.4 Rasterizer (RAS)

The rasterizer can perform Z tests prior to texture mapping, which can help to increase the fill rate when textured objects are not visible. Only rendered pixels will load texture into the texture cache.

The GP runs at 200Mhz, so RAS can generate four pixels per clock to reach its 800M pixels/second peak performance. The four pixels are arranged in a square pattern, commonly referred to as a “pixel quad.”

The performance of the pixel quad rasterization pattern is identical regardless of whether its rendering tall triangles or wide triangles. The GCN has no preference; they are rendered at the same speed.

We are aware that, with the smaller polygons rendered in the current generation of game consoles, a bigger pixel “footprint” has nearly no performance increase over a pixel quad, and faster clock rates mean more real pixels can be drawn per second. However, we chose this pixel quad “sweet spot” for the GCN in favor of a larger pixel “footprint” because the latter would dramatically increase the size and cost of the silicon chip.

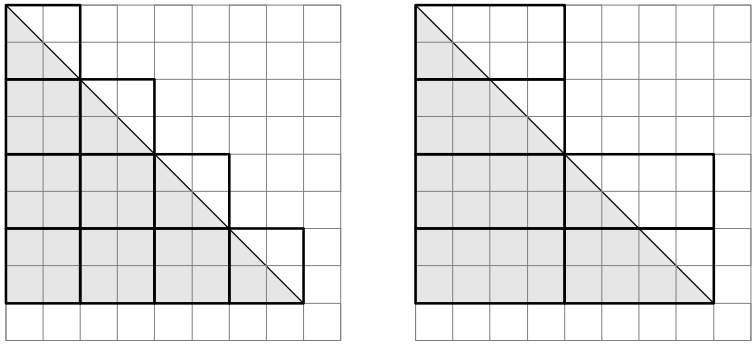

Figure 15 Pixel footprint analysis

The GCN rasterizer renders into the 1TSRAM frame buffer without risk of stalling; please see section 3.1.2 for more information about 1TSRAM.

3.5 Texture Environment Processor (TEV)

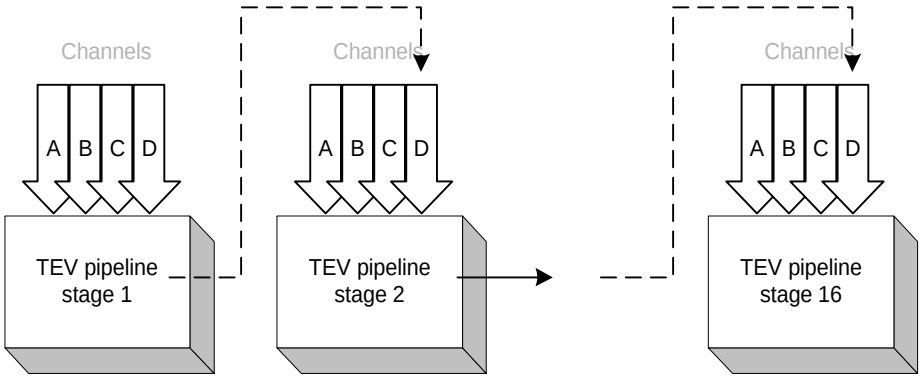

Figure 16 TEV pipeline stages

The TEV blends all of the polygon colors and texture colors together. You can enable up to 16 stages to provide blending for multiple textures. With only one blend stage, the system can perform at a peak rate of 800M pixels/second. Two blends can perform at a peak rate of 400M pixels/second, eight blends can perform at a peak rate of 100M pixels/second, and 16 blends can perform at peak rate of 50M pixels/second. Fill rates for up to eight TEV stages are shown in Table 2.

Number of TEV Stages	Pixel Fill Rate
1	800M pixels/second
2	400M pixels/second
3	267M pixels/second
4	200M pixels/second
5	160M pixels/second
6	125M pixels/second
7	114M pixels/second
8	100M pixels/second

Table 2 TEV stage fill rates

Each blend stage can perform the following equation and operations:

$$R1 = A*(1-C) + B*C$$
$$R2 = (D + sign*R1 + bias)$$
$$result = clamp(R2*shiftfactor)$$

Equation 1 Color blending

- Linear interpolation (LERP). One coefficient, *C*, selects a blend between two others, *A* and *B*.
- Subtraction. *sign* can be +1 or –1. This allows equations such as (*D*-*A*).
- Signed 10-bit (-1024 to +1023) *D* input. Typically, it combines results from other blending stages.
- Additive brightening. *bias* can be 0 or +/-0.5. This added offset brightens or darkens the result.
- Scaled brightening. *shiftfactor* can be 1, 2 or 4 to brighten by a multiply, or to scale *bias* up to 1.0.
- Clamping. The final result of each blend stage can be clamped to unsigned 8-bit or signed 10-bit (-1024 to +1023). No wrapping can occur.
- The TEV clamping modes allow per-pixel if/else evaluation.
- The final alpha output from all the TEV stages can be applied to a two-reference alpha compare circuit. The alpha compare pass/fail can conditionally write color and Z.

Note that *A, B, C,* and *D* can come from 14 possible sources.

3.5.1 Re-ordered blending

The TEV can actually “reorder” blending sources to different blend stages. For example, texture 0 can be “sent” to the TEV15 stage for blending. Please note, however, that this feature is not completely general and certain restrictions exist. Please refer to the Graphics Programmer’s Guide for details.

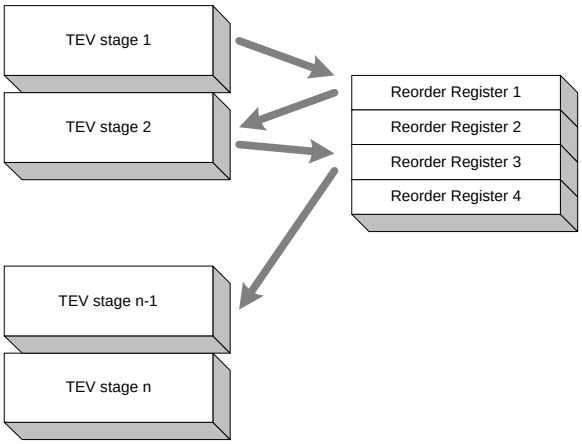

Figure 17 TEV stage result reordering

After blending the stages, the TEV is responsible for calculating and blending fog. For reference, the N64 evaluated the fog equation per vertex and linearly interpolated between vertices. The result was unnatural fog on large polygons. By contrast, the GCN TEV computes the fog equation per quad (group of four pixels), which eliminates "bad fog" on large polygons.

3.6 Pixel Engine (PE)

The PE can perform blending, antialiasing and the motion compensation (MC) stage of video decompression. It can:

- Alpha-blend a pixel into the frame buffer in a variety of ways similar to those described in OpenGL and DX documents.
- Perform the MC portion of video decompression algorithms such as MPEG. (Other parts of the MPEG decompression algorithm, such as Huffman, ZigZag, and DCT^{-1}, must be performed by the CPU.)

3.6.1 Antialiasing

The PE can also perform antialiasing by using a super-sampling technique. For each screen pixel, the PE renders color and Z data for three sub-pixels. After rendering the scene, the three sub-pixels stored in the frame buffer are averaged to compute the final image.

Super-sampling has excellent properties:

- No back-to-front polygon sorting necessary, which would be costly when handling a high number of polygons.
- No aliasing ("jaggies") at the intersection of polygons.
- Better resolution definition than N64-style antialiasing.

However, this technique also has the following restrictions:

- Maximum fill rate is 400M pixels/second; therefore, antialiasing is free if you are using two or more TEV stages.
- Z-buffer precision is reduced from 24-bit to 16-bit. When Z resolution is 16 bits, you can use inverse floating point formats to maximize accuracy and range.
- Rendering resolutions greater than 640 x 240 will not fit into the EFB. Higher resolutions will require two passes.

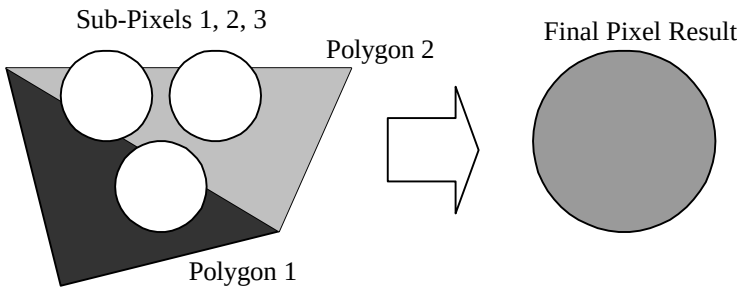

Figure 18 Super-sampling antialiasing

3.7 Internal 1TSRAM memory buffers

The embedded frame buffer (EFB) has enough bandwidth to blend four pixels/clock cycle at the peak fill rate of 800M pixels/second. The maximum EFB size is 640 x 528 x 24-bit color and 24-bit Z (528 lines are needed to support PAL for the European market). The EFB is singled-buffered and will transfer the finished image to the external frame buffer (XFB) for display. Any double-buffering occurs in main memory.

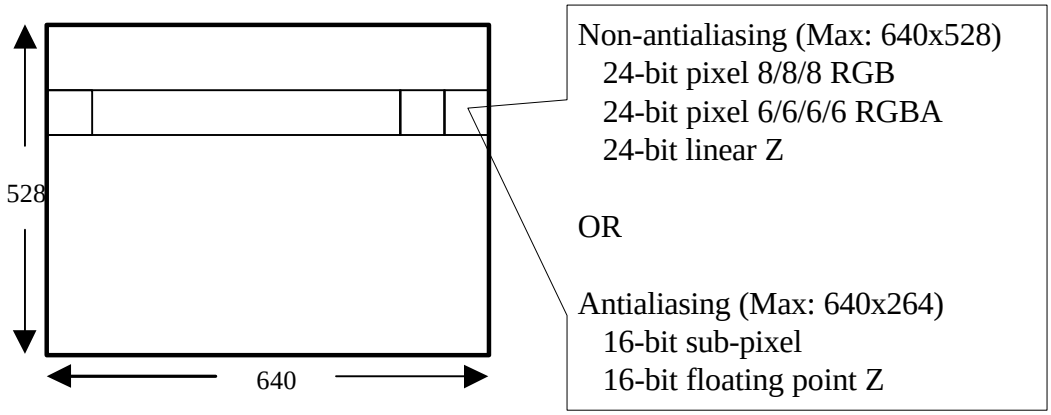

Figure 19 Embedded frame buffer (EFB)

The Texture Processor contains 1MB of local texture memory (TMEM). Entire texture maps can be loaded explicitly into TMEM. Alternatively, textures can be kept in the external 1TSRAM main memory and a portion of the TMEM can be allocated for “texture caching.” Finally, up to 128 color lookup tables (CLUTs) can be loaded into the TMEM.

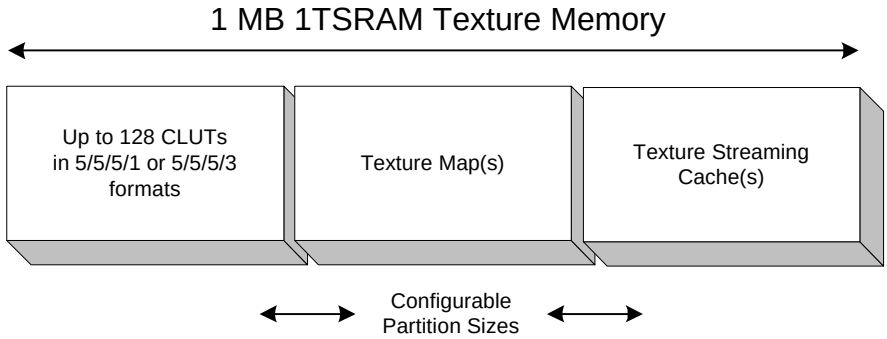

Figure 20 Embedded texture memory (TMEM)

3.7.1 Texture streaming cache

Using a streaming cache eliminates any texture access stalls while rendering pixels.

- The most recently-used texels are cached.
- If not cached, the hardware prefetches texels much earlier than the TEV stages in which they will be used.

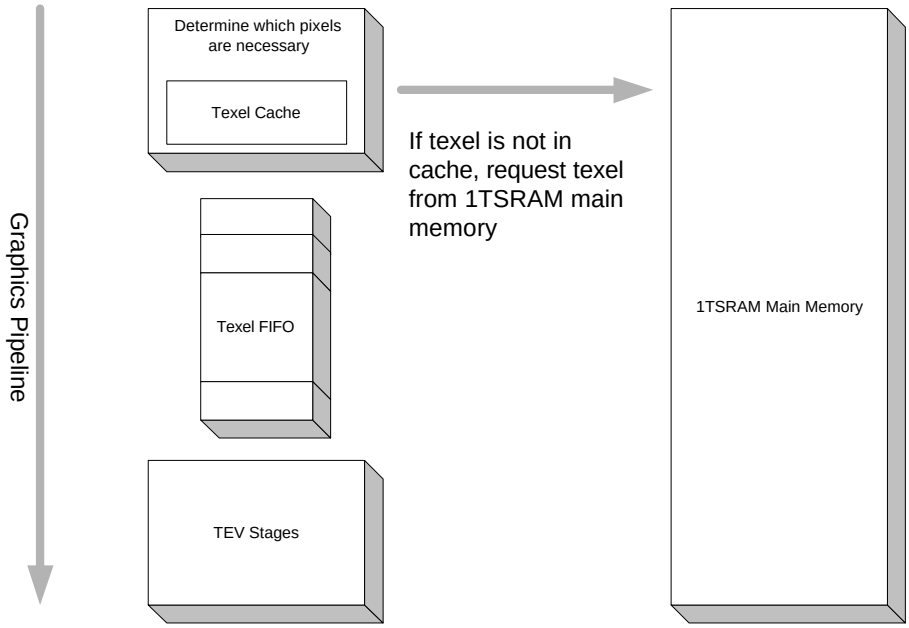

Figure 21 Texture streaming cache

The texture streaming cache gives the game developer the possibility of using the entire 1TSRAM main memory as texture memory with **no** performance penalty to pixel rendering.

3.7.2 Preloaded texture map

You may want to load the texture maps of frequently-used textures into the TMEM to guarantee the best performance. Frequently-used texture maps, such as shadow and light textures, are good candidates. Texture maps with a higher texture cache miss rate, such as reflection maps, can be preloaded effectively as well.

4 The audio DSP

The GCN audio hardware features a custom digital signal processor (DSP) which has the following characteristics.

- 100MHz instruction clock.
- 16-bit data words and addressing.
- 16-bit multiplier, 40-bit accumulator.
- Single cycle add, multiply, subtract, and MAC (multiply-accumulate).
- Single cycle load/store from local RAM.
- Hardware ADPCM decompression (reduces DSP workload by 30%).
- 8KB data RAM.
- 4KB data ROM.
- 8KB instruction RAM.
- 8KB instruction ROM.
- Dual-ported memory for simultaneous reads and writes.
- Parallel ALU and data load/store operations.
- DMA access to main memory.
- Cached interface to ARAM.
- "Mailbox" register interface with CPU.
- Hardware addressing engine for automatic data and instruction loops.

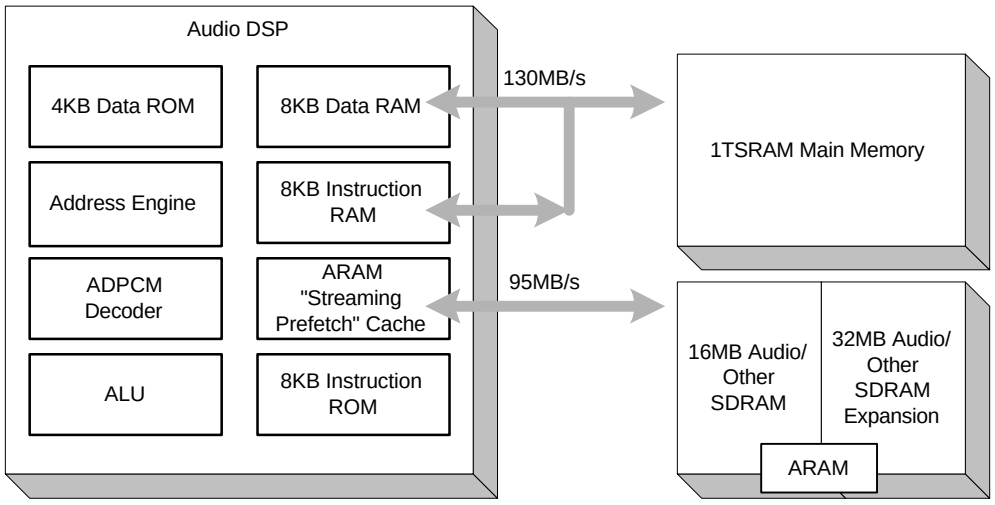

Figure 22 Audio DSP blocks

The DSP relieves the Gekko CPU of the more onerous audio processing tasks. Specifically, the DSP is responsible for:

- ADPCM decompression (hardware accelerated).
- Sample rate conversion.
- Volume envelope articulation.
- Mixing (voices and effects).
- Per-voice filtering.
- Dolby Surround encoding.

4.1 Features and performance

The DSP can generate audio samples with the following specifications:

- Up to 64 stereo Surround voices.
- 32KHz mixing rate, 48KHz output.
- Two dedicated stereo Surround effects busses.

5 Auxiliary audio memory (ARAM)

The DSP's on-chip memory is supplemented by auxiliary RAM (ARAM) with the following characteristics:

- 16MB of internal SDRAM.
- Expandable up to 48MB total with external accessory.
- 8-bit data bus.
- DMA interface to main memory (80MB/second peak rate).
- "Streaming cache" interface to DSP (95MB/second peak rate).

Although ARAM is intended primarily for the storage of audio samples, developers may also place graphics and animation data in it. Such data can be “paged” into main memory with a latency of approximately one video frame, making it ideal for buffering transactions between the very slow DVD drive and very fast main memory. Note, however, that DSP access to ARAM has priority over the main memory DMA.

6 Optical disk drive

The GCN uses optical read-only disk technology, which includes the following features:

- Large capacity.
- Constant Angular Velocity (CAV); i.e., the drive motor spins at the same speed at all times, regardless of optical laser position on an inner or outer track.
- 2MB/second minimum data transfer rate. Note that the transfer rate is different between inner and outer tracks because of CAV (inner tracks have lower transfer rates).
- 512KB data cache in the disk drive. Speculative sequential prefetch algorithm minimizes seek time.
- Separate audio streaming port.

6.1 Speculative prefetch

After the disk drive has completed a requested transfer, the disk drive controller will speculatively retrieve data positioned sequentially after the requested transfer. Since a game will often transfer sequential groups of data, the speculative prefetch mechanism reduces the access latency for subsequent data transfer requests.

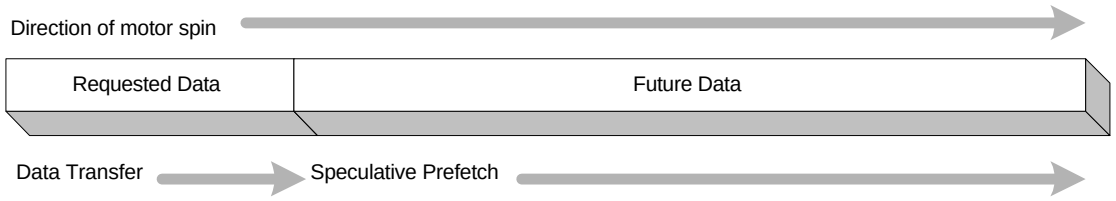

Figure 23 Disk drive speculative prefetch

6.2 Interleaved data access and audio streaming

The DVD drive can also stream 32KHz or 48KHz ADPCM audio data from the disk. This data is transferred directly into the audio interface (AI).

If simultaneous data transfer requests exist, the disk drive controller moves the optical laser reading mechanism intelligently between the data and audio data regions. Audio streaming is guaranteed to never skip; however, the data transfer rate will suffer depending on the distance between the two data regions.

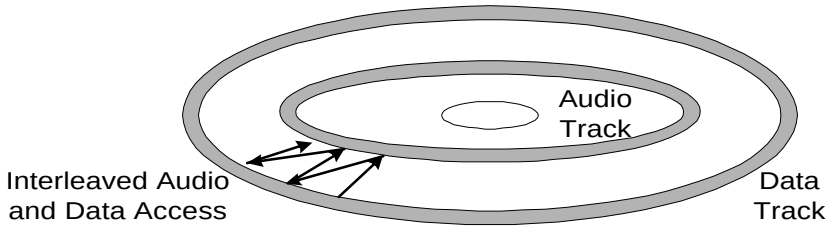

Figure 24 Interleaved data and audio

7 Game Pad (PAD)

The game pad interface (PAD) has the following features:

- Capacity to handle up to four controllers.
- Data transfers to and from the controller. Because the controller wire is long, the CPU may need to perform a checksum and retry if data is corrupt.
- Programmable intervals to retrieve game controller state automatically; no CPU servicing necessary.
- Support for two light guns.

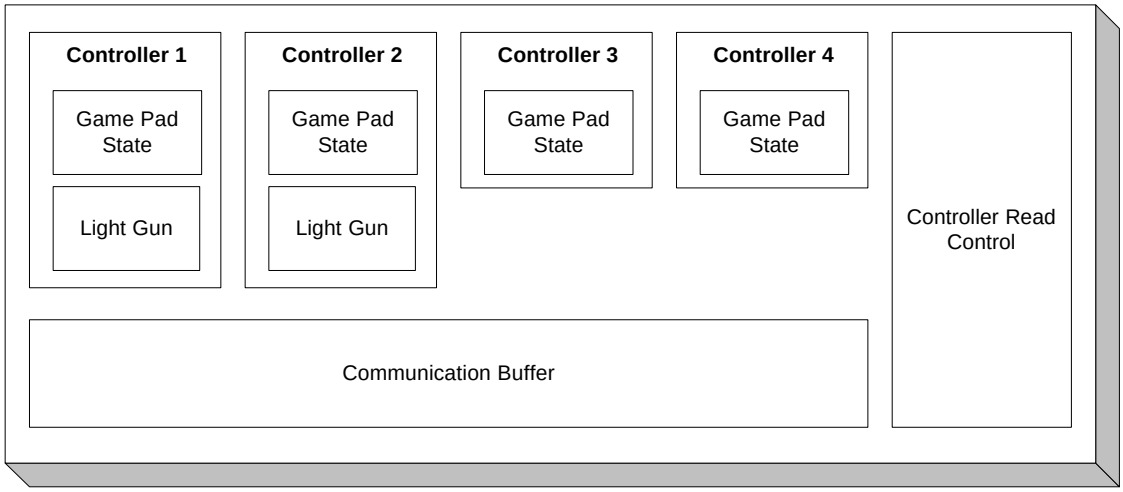

Figure 25 Game pad interface blocks

7.1 Game pad state sampling control

The hardware reads the game pad state automatically at fixed intervals. Game developers can program the interval in order to retrieve the current game pad state prior to computing game logic.

7.2 Communication buffer

This buffer transfers a block of data between main memory and memory-mapped devices, such as a memory pack, in the controller. This transfer occurs over long controller wires; therefore, the CPU has to perform a checksum for data integrity and may request a retry.

7.3 Controller memory save capability

Capacity and other details TBD.

8 Expansion Interface 0 (EXI0)

This is a serial interface port and has the following features:

- 1MHz, 2MHz, 4MHz, 8MHz, 16MHz, 32Mhz selectable clock per device. Maximum transfer rate is 4MB/second.
- Supports real time clock/calendar device (details TBD).
- External expansion port.

9 Expansion Interface 1 (EXI1)

This is serial interface port has the following features:

- 1MHz, 2MHz, 4MHz, 8MHz, 16MHz, 32Mhz selectable clock per device. Maximum transfer rate is 4MB/second.
- External expansion port.

10 Audio Interface (AI)

The Audio Interface (AI) manages the transfer of audio data into the output digital-to-analog converter (DAC). The AI has the following inputs:

- Audio data (32KHz or 48KHz) from the DVD drive.
- Audio data from a buffer in main memory (containing DSP output).

The AI mixes these two inputs together and generates 48KHz stereo samples for the output DAC.

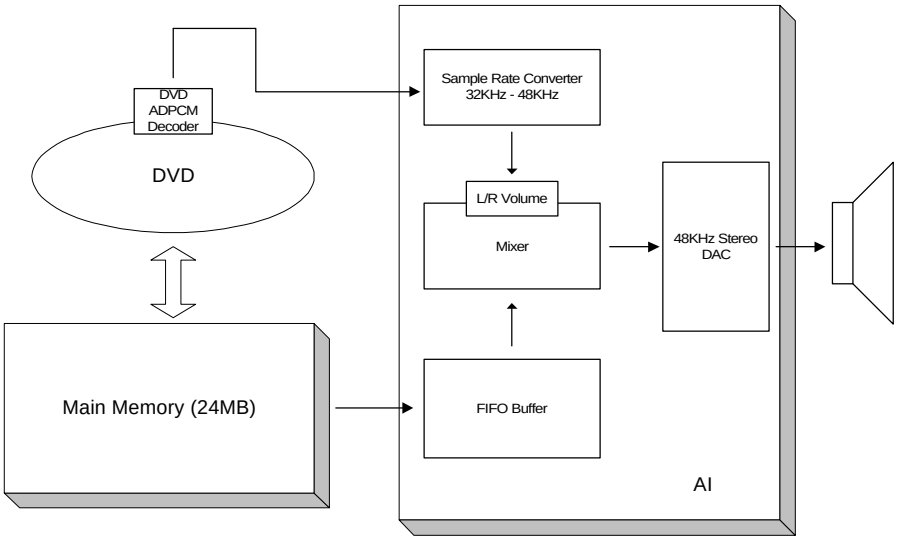

Figure 26 Audio Interface (AI) blocks

The AI provides the following features:

- Automatic conversion of 32KHz audio data from the DVD drive into 48KHz samples.
- Left and right volume controls for audio data from the DVD drive.

11 Video Interface (VI)

The Video Interface (VI) has the following features:

- Full support for NTSC, M/PAL and PAL interlaced and non-interlaced modes.
- 480-line progressive digital television output (480P).

Signal Type	Frame Buffer Size	Mode
NTSC, M/PAL	640x480	• Non-interlaced rendering (<= 30Hz). • Interlaced display.
NTSC, M/PAL	640x240	• Interlaced rendering (60Hz field rendering). • Interlaced display.
NTSC, M/PAL	640x240	• Non-interlaced rendering (60Hz). • Double-strike display (SNES style).
PAL	640x528	• Non-interlaced rendering (<= 25Hz). • Interlaced display.
PAL	640x264	• Interlaced rendering (50Hz field rendering). • Interlaced display.
PAL	640x264	• Non-interlaced rendering (50Hz). • Double-strike display (SNES style).
480P Digital Television	640x480	• 480-line progressive digital television. • Connect to component input and D Terminal of digital television.

Table 3 Video formats

The frame buffer displays from main memory using a YUV format to minimize main memory usage. The YUV color resolution is 16 bits/pixel (YUYV8888); for example, a single 640x480 frame buffer will consume 600KB.

