D/AVE 3D

Data Sheet

D/AVE 3D (V1.0)

Author	:	Christian Sehnke
Document Number	:	TD-070201DH-DS
Security	:	Confidential
Version	:	0.7
Date	:	2009-11-02
Status	:	Proposed
Released From	:	TES Electronic Solutions GmbH

PRELIMINARY REMARK:

This document is the property of the TES Electronic Solutions GmbH.

TABLE OF CONTENTS

TABLE OF CONTENTS **2**

CHANGE HISTORY **4**

1. INTRODUCTION **5**
1.1 Feature Highlights 6
1.2 Definitions and Abbreviations 7

2. INTERFACES **8**
2.1 HW Interfaces 8
2.1.1 Interrupt Request Signal 8
2.1.2 Slave Bus Interface 8
2.1.3 Master Bus Interface 10
2.1.4 Bus Adaptors 12
2.2 SW Interfaces 13
2.2.1 Native D/AVE Driver 13
2.2.2 Generic D/AVE Driver 14
2.2.3 OpenGL 14
2.2.4 OpenVG 15
2.2.5 EGL 15

3. SYSTEM ARCHITECTURE **16**
3.1 Interrupt Controller, Status/Config Controller 17
3.2 Arbiter 17
3.3 Display List Reader 17
3.4 Geometry Setup Processor 17
3.5 Performance Counters 17
3.6 Pixel Pipeline 18
3.6.1 Alpha Color Test Unit 19
3.6.2 Blend Unit 19
3.6.3 Color Unit 19
3.6.4 Color Mask Unit 19
3.6.5 Logic Op 19
3.6.6 Texture Fetch 19
3.6.7 ZSA Unit 22
3.6.8 ZSA Write Unit 22
3.7 Pixel Selection 22
3.8 Stream Controller 22
3.9 Vertex Streaming Processor 23
3.9.1 Aliases 23
3.9.2 Swizzling 24
3.9.3 Masking 24
3.9.4 VSP Example 24
3.10 Vertex Array 24

4. DISPLAY LIST DESCRIPTION **25**

5. REGISTER DESCRIPTION **26**

6. PERFORMANCE **27**
6.1 Fillrate 27
6.2 Vertex Throughput 27

6.3 TRIANGLE THROUGHPUT 27
6.4 BANDWIDTH REQUIREMENTS 28
6.4.1 AVERAGE NUMBER OF BITS REQUIRED PER PIXEL 28
6.4.2 TOTAL AMOUNT OF DATA REQUIRED PER SECOND 29

7. POWER REDUCTION FEATURES 30

7.1 CLOCK GATING 30
7.2 MEMORY BLOCK DISABLING 30
7.3 PREPARATION FOR AUTOMATIC CLOCK GATING 30

8. INTEGRATION OVERVIEW 31

8.1 CONFIGURABLE CONSTANTS 31
8.2 CLOCK 31
8.3 RESET 31
8.4 POWER MANAGEMENT 31
8.5 MEMORY DESCRIPTION 32

9. DEVELOPMENT/VERIFICATION TOOL CHAIN 33

10. RESOURCE USAGE 34

10.1 ASIC GATE COUNT 34
10.1.1 LOGIC GATES 34
10.1.2 MEMORIES 34
10.1.3 FPGA 34

IMAGES 36

TABLES 37

CHANGE HISTORY

Date	Person	Version	Reason / Changes
2008-01-04	BZu	0.1	First draft version.
2008-01-21	BZu	0.2	Add resource and performance estimation.
2008-01-25	BZu	0.3	Update Blend Unit, Color Format and Display List description. Add performance data.
2008-02-21	NSc	0.4	Review and update of SW interfaces.
2008-03-05	CSe	0.5	Included information on VSP multi-threading and corrected performance estimations
2009-01-26	CSe	0.6	Added information on power reduction features, bandwidth requirements and tool chain. Updated resource usage.
2009-10-28	CSe	0.7	Set status to proposed.
2009-11-02	CSe	0.8	Completed list of provided OpenGL ES extensions. Added chapter ‘Productivity Tool Support’.

1. INTRODUCTION

The D/AVE 3D (D3) is a VHDL-RTL-IP developed for vector based graphic applications. D3 perfectly fits to the requirements of embedded systems.

D3 is designed to fulfill OpenGL ES 1.1 (OGLES), OpenVG 1.01 (OVG) requirements and beyond, such as Edge-Based-Filtering and Per-Primitive-Anti-Aliasing.

TES provides the open standard SW APIs OpenGL ES 1.1, EGL 1.3 and OpenVG 1.01 along with its HW-IP.

1.1 Feature Highlights

- Low resource usage
  - Low gate count (~1.2 Mio. Gates)
  - Low internal memory requirements (~200-400 K Memory Bits)

- Offers a wide range of graphical primitives
  - Lines
  - Triangles
  - Quadrangles

- Prepared for easy SoC integration
  - ALTERA AVALON support
  - ARM AMBA-AHP, ARM AMBA-APB Bus support
  - Performance robust to high memory latency
  - Single clock domain architecture

- High Performance
  - 100 Mio. Pixels/Second fill rate (100 MHz).
  - 6,6 Mio. triangles throughput (100 MHz.)
  - Early-Z-Test

- High Quality Rendering
  - Anti-Aliasing and Sub-Pixel accurate rendering
  - Edge-Based-Filtering for image scaling
  - Per-Primitive-Anti-Aliasing (settable per edge)
  - Static dithering at Framebuffer write back to enhance RGB565 output

- Fully programmable Transform and Lighting Engine

- Extensive Software Stack available
  - General Driver
  - EGL 1.3 API
  - OpenGL ES 1.1 API
  - OpenVG 1.01 API

- Textures
  - 2048 x 2048 Texture size
  - Flexible texture color format handling
  - Texture compression
  - Texture offset swizzeling

- Framebuffer
  - 2048 x 2048 supported
  - ARGB8888, ARGB4444, ARGB1555, RGBA5551, RGB565, AL88, AL44, A8
  - Blend Modes - Blend, Multiply, Darken, Lighten

1.2 Definitions and Abbreviations

A	: Alpha
AA	: Anti-Aliasing
ARGB	: Alpha, Red, Green, Blue (color components)
Coverage-Alpha	: Value between 0-1 which indicates how much of the pixel is covered by the primitive.
D3	: D/AVE 3D
DL	: Display-List (Command List, Stream)
DLR	: Display-List-Reader
Driver	: D/AVE 3D Driver
EGL	: Native Platform Graphics Interface 1.3 (http://www.khronos.org)
Framebuffer	: Pixel Color Buffer
GSP	: Geometry Setup Processor
GPU	: Graphic Processing Unit (the D/AVE 3D in the scope of this text)
HW	: Hardware
IP	: Intellectual-Property
IRQ	: Interrupt-Request
IRS	: Interrupt-Request-Signal
MBI	: Master-Bus-Interface
MSB	: Most Significant Bit
N/A	: Not applicable
OS	: Operating System
OGLES	: OpenGL ES 1.1 (http://www.khronos.org)
OVG	: OpenVG 1.01 (http://www.khronos.org)
PP	: Pixel Pipeline
RGB	: Red, Green, Blue
R/W	: Read / Write
SBI	: Slave-Bus-Interface
SIMD	: Single Instruction Multiple Data
SW	: Software
Texel	: Texture Pixel
Tri	: Triangle
UMA	: Unified Memory Architecture
VSP	: Vertex Streaming Processor
ZSA	: Interleaved buffer of Depth, Stencil and Global Alpha Mask attributes

2. INTERFACES

2.1 HW Interfaces

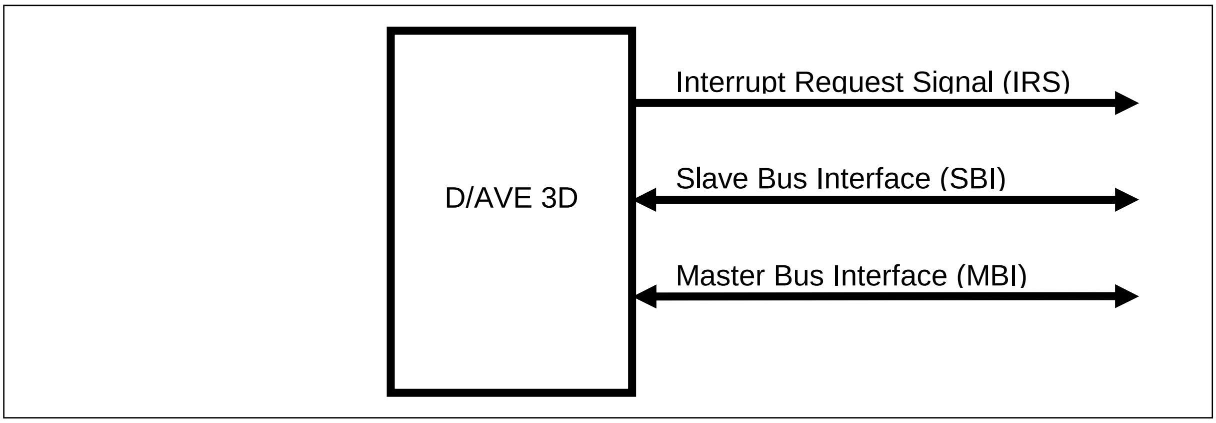

D/AVE 3D
Interrupt Request Signal (IRS)
Slave Bus Interface (SBI)
Master Bus Interface (MBI)

Image 2-1 : Interface Diagram

2.1.1 Interrupt Request Signal

A single signal indicates an interrupt request. There are two interrupt sources:

- DLFINISHED – Indicates that DL has been fully processed
- DLEVENT – Hit event command within DL or indicates an error within DL

In case of a DLEVENT the IP paused until the IRQ has been handled. This interrupt can be used for CPU synchronization at a certain point. This interrupt also indicates an error within the DL itself.

2.1.2 Slave Bus Interface

The Slave-Bus-Interface is a simple 32 Bit slave interface for the following purposes:

- To get general HW capabilities information
- To get status information
- To set display list start address
- Interrupt handling
  - To reset an interrupt request
  - To get active interrupt request bits
  - To get interrupt parameters
- Performance counter handling

2.1.2.1 SBI Specification

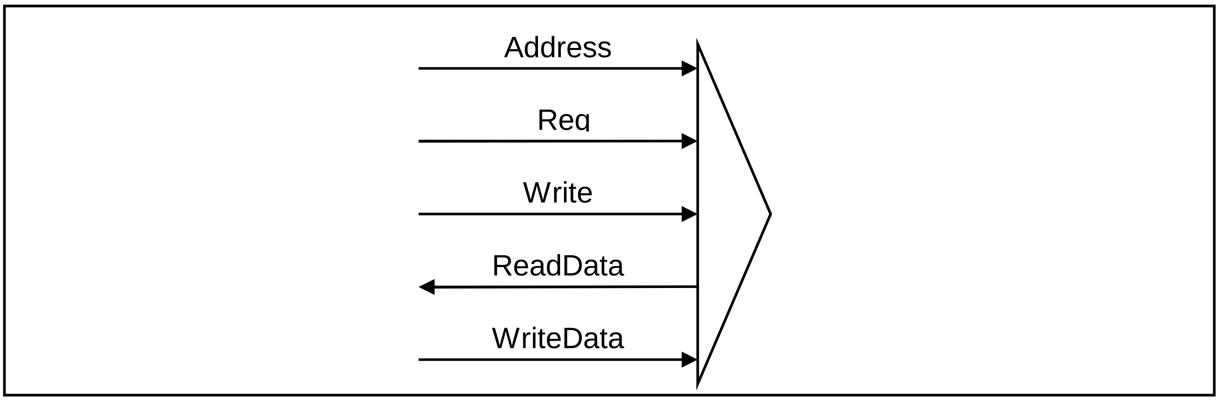


Image 2-2 : Interface Diagram – Slave Bus Interface

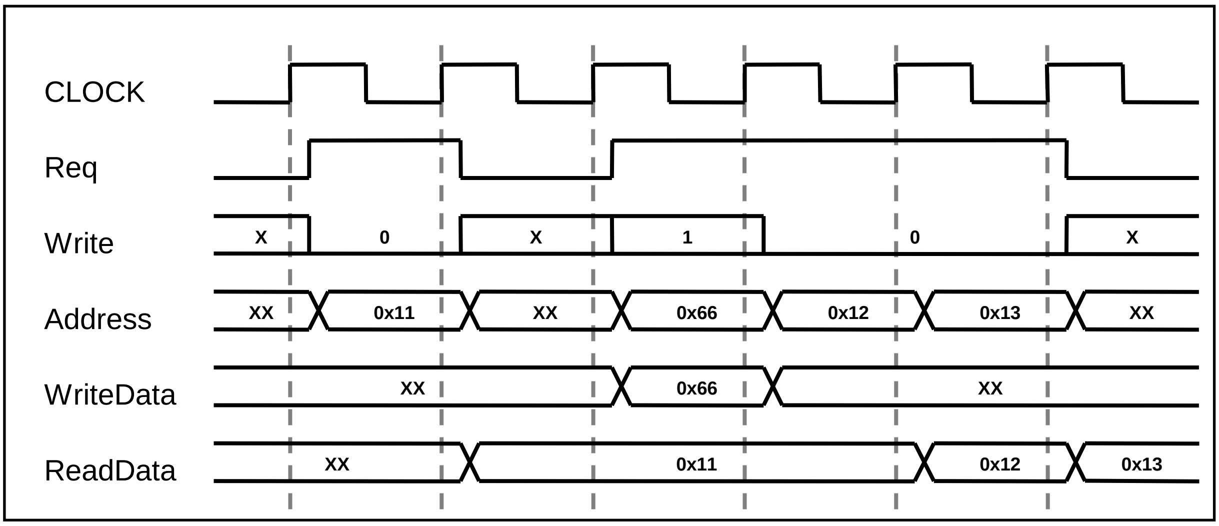


Image 2-3 : Waveform Diagram – Slave Bus Interface

	Inputs	
	• **Address** Target address for the current read or write request. Valid when "Req" is asserted. • **Req** An asserted "Req" indicates a request. • **Write** An asserted "Write" together with an asserted "Req" indicates a write request. Asserted "Req" with an unasserted "Write" indicates a read access. • **WriteData** Data to write. Valid when "Req" and "Write" are asserted.	
	Outputs	
	• **ReadData** Valid one cycle after an asserted "Req". ReadData is a register, which holds/keeps the value of the last read request.	

2.1.3 Master Bus Interface

The generic MBI gives D3 an r/w master access to memory, its main features are:

- Byte enable signals
- Variable burst count
    - max. count configurable
- Word aligned burst access
- Word width configurable (32,64,128)
- A new request can be started before previous read access is fully closed

2.1.3.1 MBI Specification

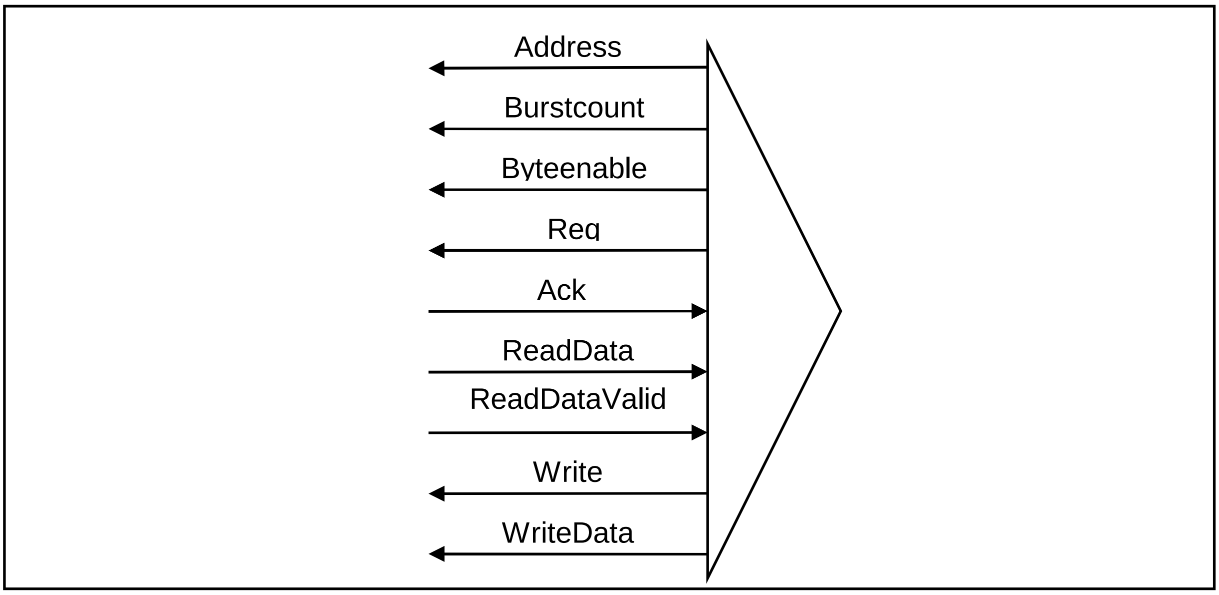


Image 2-4 : Interface Diagram – Master Bus Interface

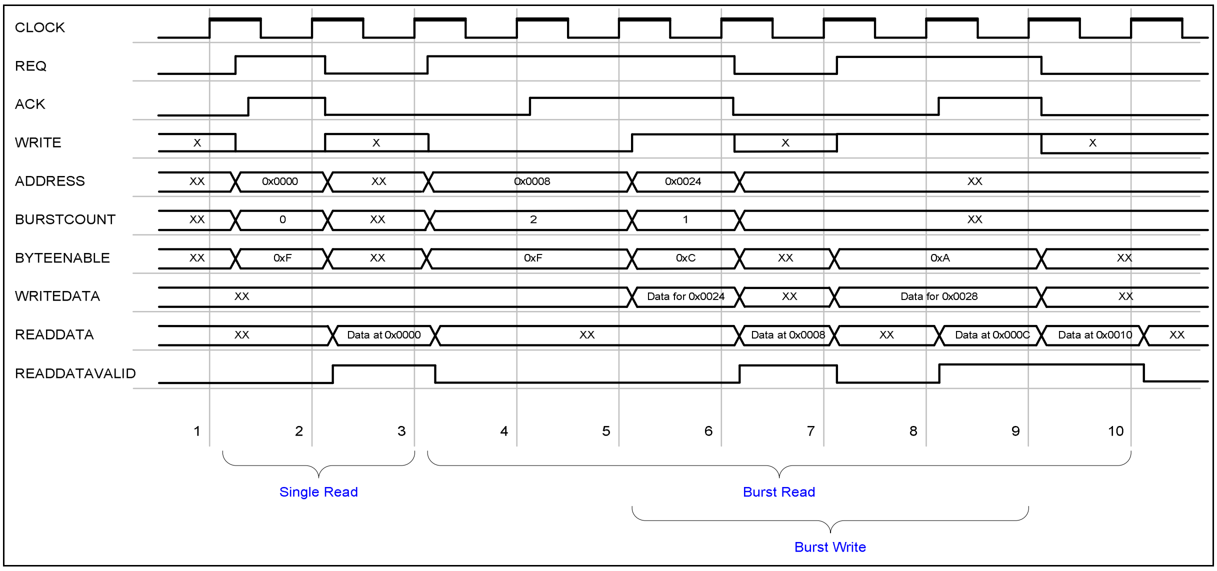


Image 2-5 : Waveform Diagram – Master Bus Interface

The waveform above shows an example for the fastest possible single read, a burst read of length 3 and a burst write of length 2. It is assumed, that a memory word consists of 4 bytes. For simplification, the addresses use only 16 bits.

Inputs

- **Ack**
  Acknowledge for “Req” output.
- **ReadData**
  Data read from the external memory. Valid when “ReadDataValid” is asserted.
- **ReadDataValid**
  Indicates “ReadData” is valid. Asserted for a single clock cycle for every memory word returned from the external memory.

Outputs

- **Req**
  Request to the external memory. Once asserted, it must stay asserted until “Ack” input is high at a rising edge of the clock. This clock edge is the moment, when the command is effectively accepted by the bus.
  “Write”, “Address”, “Burstcount”, “Byteenable” and (in case of a write) “WriteData” have to be stable while “Req” is asserted (exception: “Address” and “Burstcount” are not considered for all but the first request of a write access).
- **Write**
  Indicates, whether a request is a read (0) or a write (1).
- **Address**
  Target address for the current read or write request.
  Byte based, only multiples of memory word size allowed.
- **Burstcount**
  Number of words-1 (0 means one word and so on), for the current read or write request.
  In case of a read request, “Burstcount”+1 words will be returned with “ReadDataValid” being asserted.
  In case of a write request, “Req” and “Write” have to be asserted “Burstcount” times after the initial write request of a burst. “Address” and “Burstcount” are only considered for the initial write request.
- **Byteenable**
  Only considered for a write request: Masks the bytes in current “WriteData”, which have to be written to the external memory.
- **WriteData**
  Current write data. Masked by “Byteenable”.

2.1.4 Bus Adaptors

Due to the fact that MBI and SBI are TES proprietary, glue logic (Bus-Adaptor) is required to connect D3 to certain bus interfaces.

TES has available Bus-Adaptors for ALTERA AVALON™ and ARM AMBA™ (APB for SBI, AHB or AXI for MBI).

Bus-Adaptors for other bus interfaces can be implemented on customer site or can be provided by TES as a design service. Please get the latest list of current supported Bus-Adaptors from TES Sales.

2.2 SW Interfaces

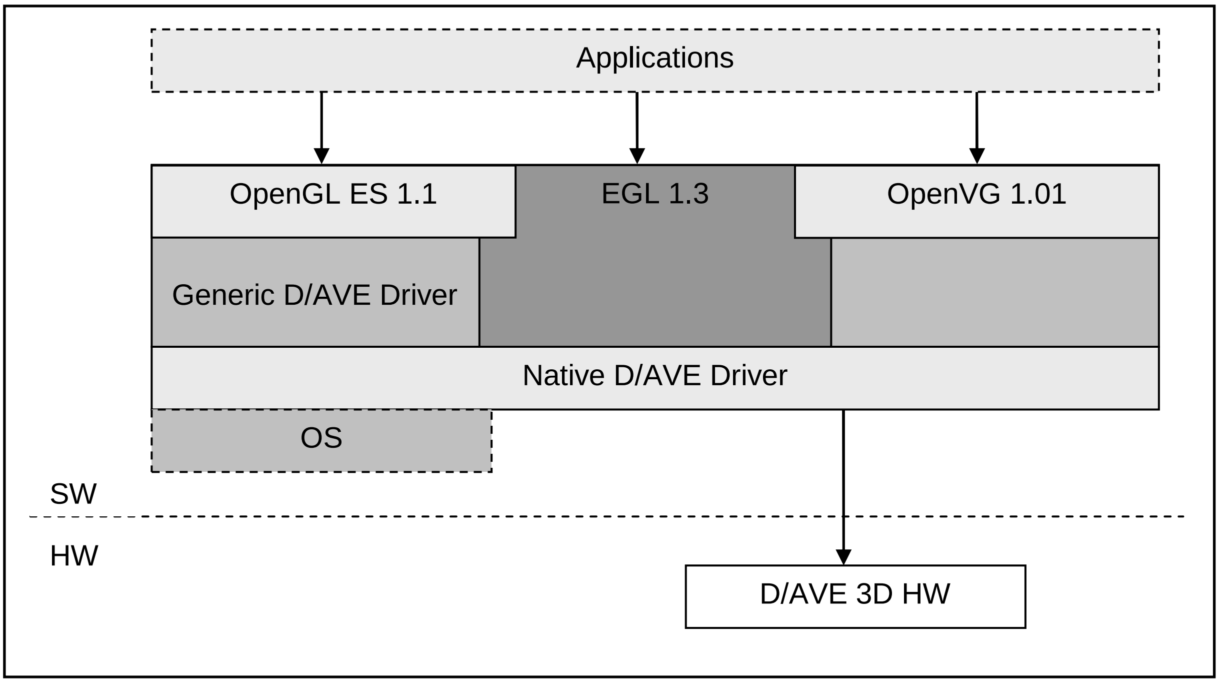


Image 2-6 : SW Interfaces

TES offers the open APIs OGLES, OVG and EGL for the D/AVE 3D IP series.

Please get the list of current supported OS and HW-System configurations from TES Sales.

2.2.1 Native D/AVE Driver

The native driver is a thin layer of OS abstraction. All platform dependant functions are implemented at this level. If the target OS is partitioned into kernel and user space, the native driver is the only element that is running in kernel space.
It provides the following services to the upper software layers :

- Management of GPU memory (if the platform is not UMA).
- Address translation.
- Transfer of data to and from GPU memory.
- Receiving interrupt requests from the HW and raising corresponding events.
- Abstraction of target OS's thread synchronization.
- Abstraction of target OS's CPU memory management.
- Abstraction of target OS's window management (for EGL).
- Accessing D/AVE hardware registers.

2.2.2 Generic D/AVE Driver

The generic driver is written in plain ANSI C and designed to be easily portable. It has no dependency on any standard library or OS function.
All functionality of the D3 HW can be accessed using the generic driver. All functions are fully reentrant and thread safe.
Main services provided by the generic driver are :

- Context management
    - All interaction with the driver is associated with a context.
    - Each process needs to have it's own context.
    - All data and all actions are linked to a context.

- Buffer management
    - All objects the GPU works with are stored in buffers (textures, streams, ZSA buffers, Framebuffers).
    - Buffers can be shared between contexts.
    - Buffer usage by the GPU is tracked using fence sync objects in the stream.
    - Buffers can be in shared memory, cpu side, gpu side or both sides (using a shadow copy) transparent to the application.

- Stream creation
    - All rendering is triggered by the command stream.
    - Adding geometry data, register changes, shader uploads etc to the stream.
    - Quickly add, remove or insert instructions or instruction batches.
    - Flatten all instructions into a Buffer.

- Error handling
    - Every function returns an indication if it succeeded.
    - Errors are stored internally until they are received by the application.
    - Once an unreceived error is in the system no more functions are executed and no more errors are generated.
    - All functions are documented to indicate the errors they can possibly generate.

2.2.3 OpenGL

TES provides an OpenGL ES 1.1 software interface. The OpenGL implementation does conform to the standard according to Khronos OpenGL/ES conformance tests.
All functionality is mapped to the HW, no software rendering is performed.
As OpenGL does not cover all functionality of D3 we will provide custom extensions.

Supported OES extensions:

At least the following extensions are available:

- OES_byte_coordinates
- OES_fixed_point
- OES_single_precision
- OES_read_format

- OES_query_matrix
- OES_blend_equation_separate
- OES_blend_func_separate
- OES_framebuffer_object
- OES_rgb8_rgba8
- OES_texture_env_crossbar

Proprietary extensions to support the following techniques:

- Specular per-pixel lighting using a vector normalization texture
- Spherical environment mapping
- Dual-paraboloid environment mapping

2.2.4 OpenVG

TES provides an OpenVG 1.01 software interface. The OpenVG implementation does conform to the standard according to Khronos OpenVG conformance tests.

2.2.5 EGL

A matching EGL 1.3 software interface is provided to be used in conjunction with both OpenGL ES and OpenVG.
Mixed rendering using both OpenGL ES and OpenVG to the same framebuffer is possible.

3. SYSTEM ARCHITECTURE

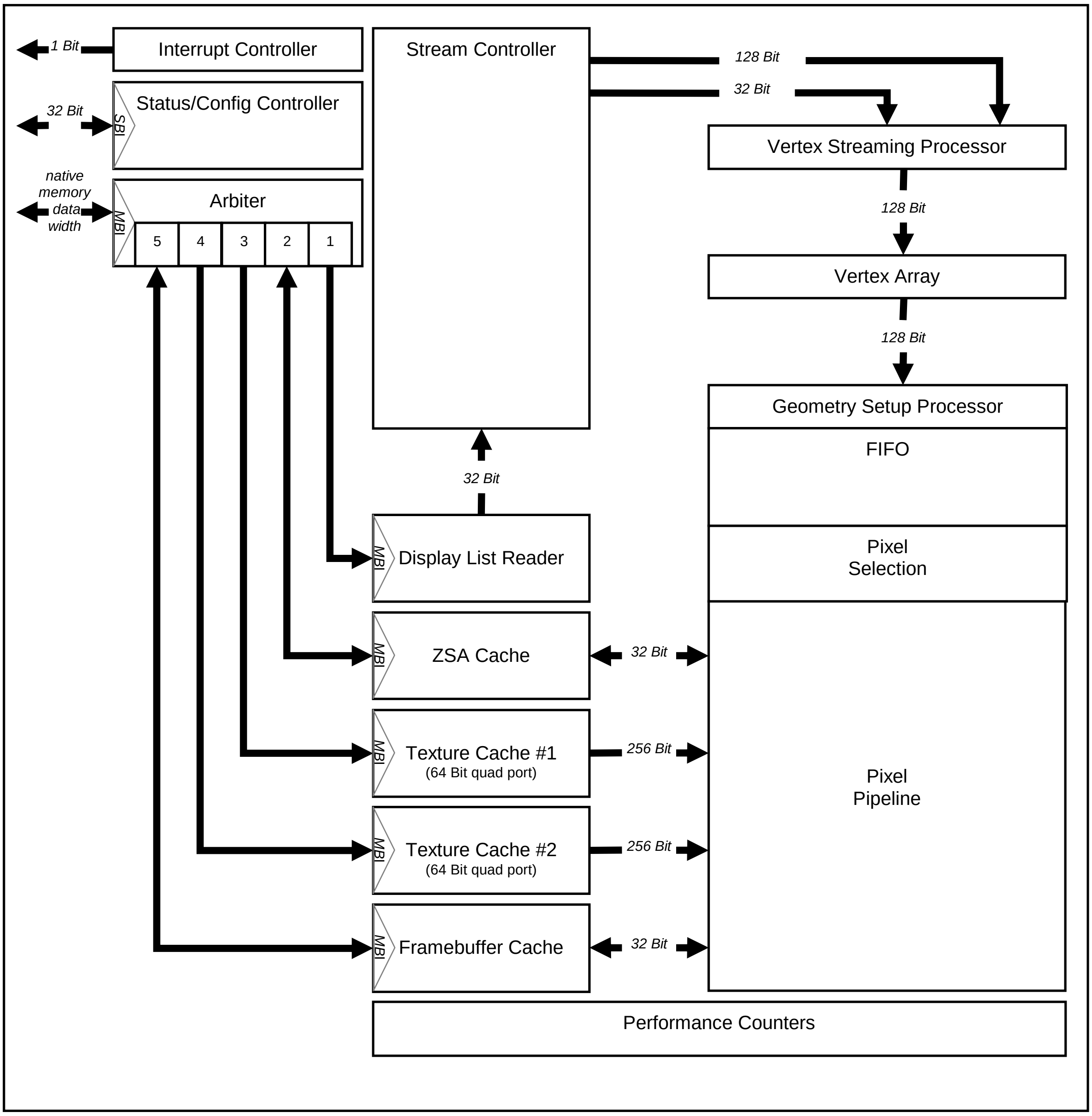


Image 3-1 : Block Diagram – D/AVE 3D Dataflow

After reset the whole IP waits to get configured by the CPU. The CPU can configure or can get status information via the SBI (Slave Bus Interface).

The common approach is that the Driver creates a command list in memory and stores the pointer to it in a dedicated register. The D3 will automatically start processing the command list. A status flag and an interrupt indicate that the command list has been fully processed.

To deal with unpredictable memory latency, FIFOs are added at all crucial places, making D3 more performance robust.

3.1 Interrupt Controller, Status/Config Controller

This module handles interrupts, performance counters, status information etc. For further information see chapter 2.1.1 and chapter 2.1.2.

3.2 Arbiter

This module merges all 5 internal MBIs. The Arbiter works with a fixed prioritization scheme:

- Display List Reader
- ZSA Cache
- Texture Cache #1
- Texture Cache #2
- Framebuffer Cache

Display List Reader is most prior.

3.3 Display List Reader

The DLR reads the display list with fixed bursts into a FIFO and transfers the data in chunky of to the Stream Controller.

3.4 Geometry Setup Processor

This module calculates the input parameters for the Pixel Selection Unit.

3.5 Performance Counters

The D3-IP has four groups of Performance-Counters. Each group has its own trigger events and a specific amount of individual counters.

- 4 Cache-Counters to measure cache performance

- 2 General-Counters to measure overall used cycle for different domains such as Pixel-Pipeline.

- 4 Pixel-Pipeline-Counters to measure performance and to find performance bottlenecks

- 2 VSP-GSP-Counters to measure vertex and primitive performance

3.6 Pixel Pipeline

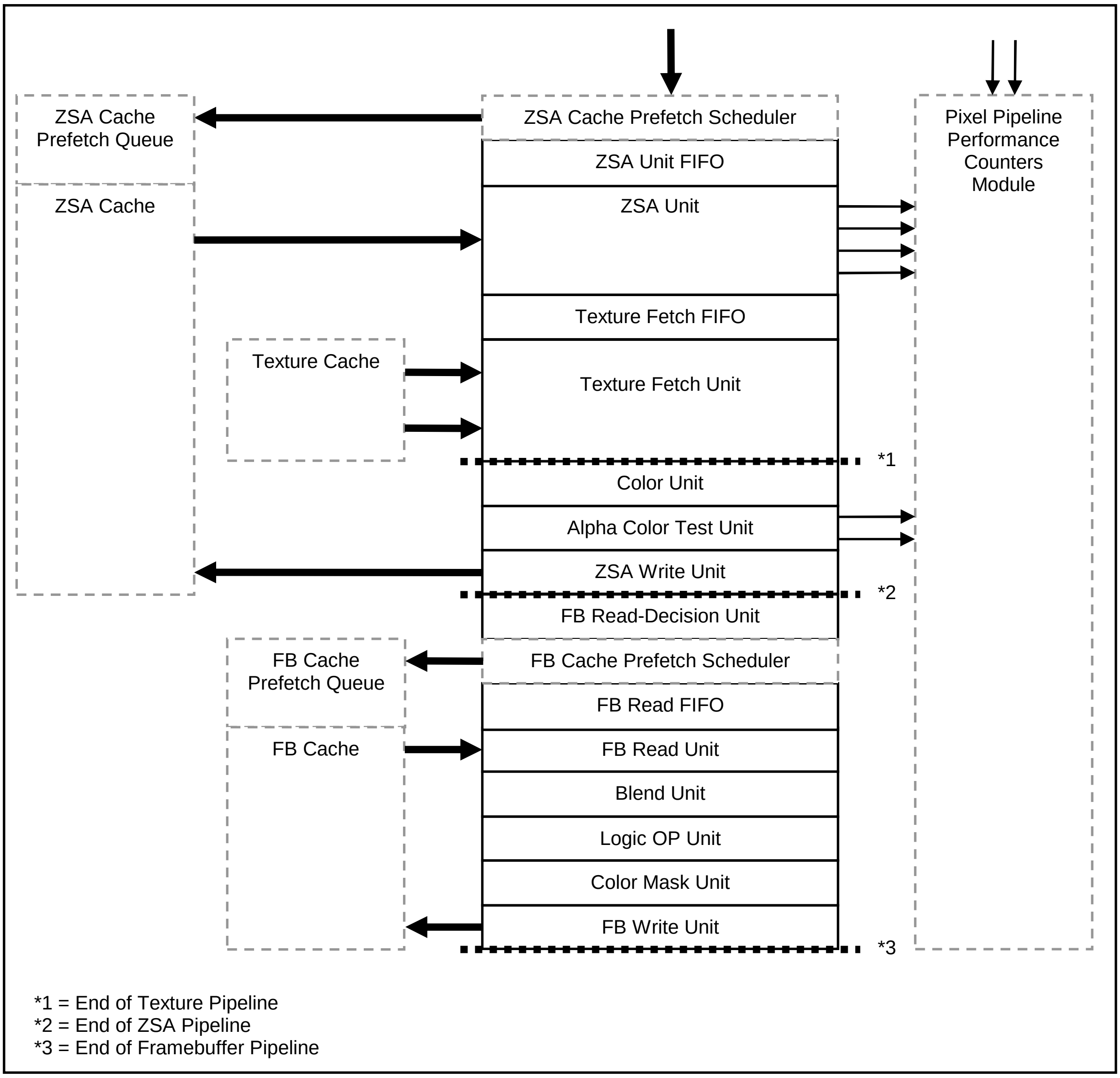


Image 3-2 : Block Diagram - Pixel Pipeline

The actual pixel processing takes place in the Pixel-Pipeline. Last part of the Pixel-Pipeline is the write back to the Framebuffer. In terminology of this document Framebuffer means the alpha and color part only. Other values like Depth, Stencil and Global Alpha Mask are stored interleaved in an extra buffer, called ZSA-Buffer.

The Pixel-Pipeline supports Early-Z-Test. This requires that the ZSA Value write back takes place after Alpha Color Test. If ZSA-Unit marks a pixel as not passed and Alpha-Test is off, no texture fetch will be made for it.

The Pixel-Pipeline has a throughput of 1 pixel per cycle, even in the case that all features are turned on. The Texture-Cache offers 4x64Bit read ports, which enables full speed texture filtering even with compressed textures.

3.6.1 Alpha Color Test Unit

An OGLES conform Alpha-Test and a Color-Key-Test is performed in this module.

3.6.2 Blend Unit

This unit blends the pixel with the Framebuffer. All OGLES and OVG blend modes are supported, such as:

- Blend (F[RGB] = D * A + S *B)
- Multiply (F[RGB] = S * (D + A) + D * B)
- Darken (F[RGB] = MIN (S + D * A , D + S * B))
- Lighten (F[RGB] = MAX (S + D * A , D + S * B))

Additionally dithering a blending with the pixel coverage value is also be performed.

3.6.3 Color Unit

Performs OGLES and OVG conform pixel texturing.

3.6.4 Color Mask Unit

All color channels can be individually turned on or off for the Framebuffer write back.

3.6.5 Logic Op

This module performs OGLES conform logical pixel operations.

3.6.6 Texture Fetch

This module performs the following:

- Texel Fetch (4xTexels per cycle)
- Texture Decompression
- Texture Filtering
  - Bilinear
  - Edge based
  - Next neighbor
- Texture MipMapping
- Texture Offset Swizzeling
  - Linear
  - Full interleaved
  - Interleaved 4
  - Interleaved 16
- Texture Pre-Multiplication
  - texel[RGB] = texel[RGB] * texel[A]
- Texture Wrap (U/V separately)
  - Clamp to Edge
  - No Clamp
  - Repeat

3.6.6.1 Color Formats

The D3-IP supports a wide range of texture color formats, realized by a flexible two step approach. The following formats are supported:

- ARGB8888
- ARGB4444
- ARGB1555
- ARGB5551
- RGB565
- Luminance 8 and Alpha 8
- Luminance 4 and Alpha 4
- Luminance 8 or Alpha 8

And also all possible permutations, such as BGR565 are supported.

3.6.6.2 Edge-Based-Filtering

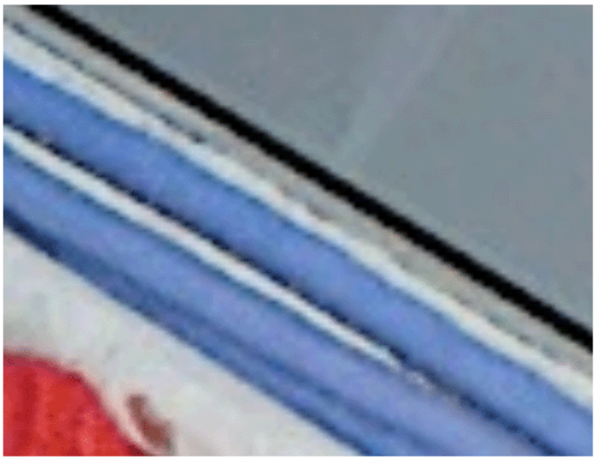
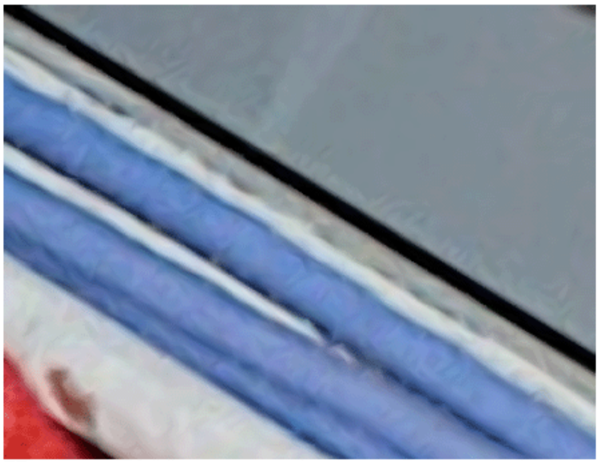
	

Table 3-1: Bilinear (left), Edge-Based (right)

The D3-IP offers with Edge-Based-Filtering a good alternative to the common Bilinear approach. In many cases Edge-Based-Filtering produces far less stairway artifacts compared to Bilinear-Filtering.

3.6.6.3 Texture Compression

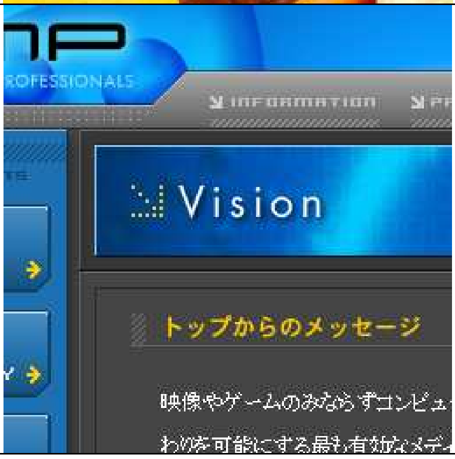
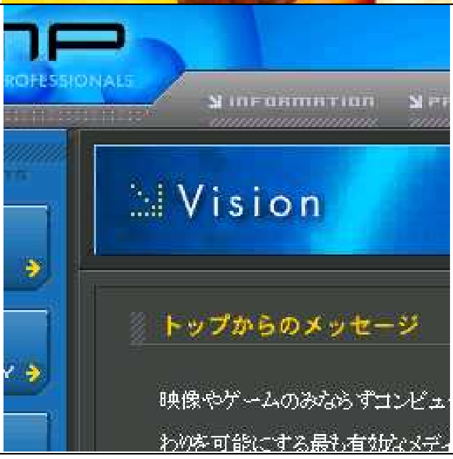
 ROFESSIONALS INFORMATION Vision トップからのメッセージ	 ROFESSIONALS INFORMATION Vision トップからのメッセージ

Table 3-2 : uncompressed 24Bit per Pixel (left), 4Bit per Pixel compressed (right)

D/AVE 3D offers 4Bit per Pixel lossy RGB decompression support. TES provides sources and a Windows based Tool for the compression part.

3.6.7 ZSA Unit

An OGLES conform Depth-Test and Stencil-Test is performed in this module. Also a Global-Alpha-Mask can be optional multiplied either with the pixel color or the coverage alpha. The following ZSA configurations are supported:

- 16 Bit – 1 Bit Stencil, 10 Bit Depth, 5 Bit Global-Alpha-Mask
- 16 Bit – 1 Bit Stencil, 15 Bit Depth
- 32 Bit – 8 Bit Stencil, 16 Bit Depth, 8 Bit Global-Alpha-Mask
- 32 Bit – 8 Bit Stencil, 24 Bit Depth

3.6.8 ZSA Write Unit

This module performs a ZSA-Value write back under certain conditions.

3.7 Pixel Selection

This module determines all pixels in the Framebuffer covered by a primitive (e.g. Triangle) and calculates the following per pixel attributes:

- Color
- Coverage-Alpha
- Depth-Value
- Dither-Pos
- MagFilter
- MipMap-Blend-Factor
- MipMap-Level
- Texture Coordinate #1
- Texture Coordinate #2

3.8 Stream Controller

This module is the primary controller of the whole D3-IP and it handles the actual display list processing. A display list basically contains all information (e.g. full 3D scene) to render a complete frame or parts of a frame. A display list can contains the following data:

- Configuration data
    - Edge Flags
    - Pixel Pipeline parameters
    - Pixel Selection parameters
    - VSP constants
- Program code
    - VSP
    - GSP
- Pointer to a display list (nested display lists supported)
- Custom IRQ + 32 Bit data word
- VSP trigger command
- GSP trigger commands, separate ones for 2,3,4 vertex based primitives
- Vertices and Attributes

3.9 Vertex Streaming Processor

The stream based vertex processing takes place in this module. Execution time is one cycle per instruction, except for the multiply and add instruction (3 cycles), without read data dependency. The VSP enables OpenGL ES 2.0 like Vertex-Shader support.

Brief overview:

- 48 Bit Instruction size
- Internal program memory can handle up to 1022 instructions
- Up to 40 General Purpose Registers (128 Bit, 4x32 Bit Floats, xyzw)
- Up to 384 Constant Registers (128 Bit, 4x32 Bit Floats)
- 22 instructions
    - 13 SIMDs
        - abs – Absolute value
        - add – Add vectors
        - comp – Compare two vectors
        - frac – Fractional part
        - int – Integer part
        - isX(z) – Compare number to reference
        - madd – Multiply Add
        - sub – Subtract
        - max – Maximum of two numbers
        - min – Minimum of two numbers
        - move – Register copy
        - movec – Conditional register copy
        - mul – Multiply
    - 5 vector based instructions
        - dp3 – Three-dimensional dot product
        - dp4 – Four-dimensional dot product
        - xp1 – Cross product part 1
        - xp2 – Cross product part 2
        - msubalt – Alternating product subtraction
    - decompose – Split into sign, exponent, mantissa
    - compose – Counterpart of decompose
    - rsqr – Reciprocal Square Root
    - lutN – Mantissa Lookup
- Multi-threading to reduce effect of pipeline stalls (not yet in version 1.0)

3.9.1 Aliases

The assembler allows the use of aliases for registers. To define an alias simply write ”alias <name> <register>” in the assembly program. Where <register> can be a complete register, or just some components. If <register> is a complete register, swizzle operations are allowed on this alias.

Examples:

```
alias vertex r0 // define vertex as r0
alias ref c0.x  // define ref as c0.x (useful for scalar values)
```

3.9.2 Swizzling

For both input registers the order of the components can be arbitrarily rearranged.

```
r1 . xz // use x and y component for the operation
r1 . xx // replicate x component
r1 . yx // rearrange components
```

3.9.3 Masking

Masking is used for the destination register to determine which components should be written. It works exactly like the swizzle operation, but supports no rearrange and no replicate:

```
r1 . xz //write first resulting x, second resulting z.
```

3.9.4 VSP Example

This example demonstrates a small VSP program which transformed a 4 dimensional vertex and passed the result and one color to the Vertex-Array.

```
stream 2        // initiate 2 reads from inputstream
alias vertex r0 // inputstream is vertex , color
alias color r1

alias M11 c370  // transformation matrix in c370..c373
alias M12 c371
alias M13 c372
alias M14 c373

dp4 r3.x_z, point,M11       //M1*v start
dp4 r3._y_w,point,M12
dp4 r4.x_z, point,M13
dp4 r4._y_w,point,M14

add vertex.xy,r3.xy,r3.zw
add vertex.zw,r4.xy,r4.zw   //M1*v end

// output in order: color,vertex
move sc,color
move sp,vertex
```

3.10 Vertex Array

The Vertex-Array is basically a 16 entry array. Each entry stores the following data:

- vertex - 128 Bit, 4x32 Bit Floats, xyzw
- texture cords #1 - 64 Bit, 2x32 Bit, st
- texture cords #2 - 64 Bit, 2x32 Bit, st
- color - 128 Bit, 4x32 Bit Floats, argb
- valid – 1 Bit, indicates the this entry is contains valid data

4. DISPLAY LIST DESCRIPTION

The stream is a combination of command and data words. Commands are used to load vertices into the VSP, which will transform and store them in the Vertex-Array. The vertex data itself directly follows the command. Other commands will be used to trigger the render unit to draw a line, triangle, quad. These render commands contain direct indices into the Vertex-Array in order to specify the data source. The driver can transform incoming vertex (and index) streams into this interleaved format. Reading and interpreting the resulting stream will create no more memory accesses than working directly with vertex and index data (equal amount of caching assumed).

5. REGISTER DESCRIPTION

N/A.

6. PERFORMANCE

All performance data given are made under the following assumptions:

- 100% cache hit rates
- No latency from external data
- Internal pipeline latency ignored
- 100 MHz.

6.1 Fillrate

The Pixel-Pipeline of the D3-IP can produce 1 pixel per cycle, in any configuration.

6.2 Vertex Throughput

The vertex throughput depends very much on how many light sources are enabled and if transformation is performed. It also depends on the associated vertex attributes, such as color and texture coordinates.

Transformation	Lighting	Shade Model	Texture Coordinates	Performance (Vertices Per Second)
-	-	Flat	-	33 Mio.
4x4 Matrix	-	Flat	-	6,6 Mio.
4x4 Matrix	-	Smooth	-	6,25 Mio.
4x4 Matrix	-	Smooth	S,T	5,8 Mio.
4x4 Matrix	1xDiffuse 1xAmbient	Smooth	S,T	0,6 Mio. *1

*1: Can be doubled at least, once multi-threading is implemented in the VSP

6.3 Triangle Throughput

The triangle throughput depends very much on the number of attributes which have to be interpolated across the primitive.

Z-Buffer	Vertex Colors	Texure Coordinates	Performance (Triangles Per Second)
-	-	-	6,6 Mio.
x	x	-	5,2 Mio.
x	-	S1,T1	6,6 Mio.
x	x	S1,T1	4 Mio.
x	x	S1,T1 / S2, T2	3,5 Mio.

6.4 Bandwidth Requirements

To reach the maximum possible performance, it is necessary to optimize the memory bus interface in terms of both latency and bandwidth.
Latency is not so much an issue, as the core uses prefetching mechanism and FIFOs to cope with a higher latency at the memory interface. Still, it should be clear that the lower the latency the better.

The bandwidth requirements strongly depend on the application. The factors to be considered are:

- Screen size (number of pixels)
- Desired rendering rate (fps)
- Framebuffer format (bits per pixel), framebuffer read required? (blend mode)
- Z buffer used, Z buffer format (bits per pixel)
- Usage of textures, texture format (bits per texel)
- Amount of vertex/geometry data

The following calculation gives an example for a typical 3D use case:

- 800x480 screen → 384000 pixels per frame
- 30 fps target render rate
- 32 bit ARGB framebuffer, alpha blending for 20% of the screen surface
    - Assumption that every pixel on the screen is rendered two times (2x overdraw)
- 32 bit ZSA buffer
    - Assumption that 50% of all pixels in the scene are dropped through the Z buffer and never rendered
- Textures enabled, 4 bit per texel compressed texture format
    - 4 texels read per pixel (bilinear or edge based filtering), 80% cache hit rate
- 10000 vertices (3 floats for position, 2 floats for texture coordinates, 4 bytes for vertex color) → 10000 * (5*4 + 4) = 240000 bytes

There are basically two ways to look at the above calculation:

6.4.1 Average number of bits required per pixel

This point of view allows a short estimation of the minimum necessary data width of the bus:
It is clear that for every single pixel processed by the pipeline, which comes out as being visible, at least a Z read, a Z write and a FB write are necessary.
When it comes to textures, also a certain average amount of texture data needs to be read per pixel, same for the vertex data.
Z read, Z write and FB write already sum up to 96 bits in the above example. This already indicates that 128 bits is a sensible bus width to avoid too much pipeline stalls due to limited bus bandwidth.
Even when using only 16 bit formats for both ZSA and frame buffers, the bus width should not be configured to be less than 64 bits.

6.4.2 Total amount of data required per second

This point of view allows estimating the necessary available bandwidth on the bus, considering the activity of other components in the system.

Calculating the complete bandwidth in terms of MBytes/second for the above case gives the following:

Visible pixels per second: (384000 pixels) * (30 fps) = 11.52 MPixels/second (visible pixels only!!)

Framebuffer data per visible pixel: (4 bytes per pixel) * (2x overdraw) + (4 bytes per pixel) * (20% reads) = 4*2.2 = 8.8 bytes/pixel

Z buffer data read per visible pixel: (4 bytes per pixel) * (2x overdraw) * (2x drop factor) = 16 bytes per pixel

Z buffer data write per visible pixel: (4 bytes per pixel) * (2x overdraw) = 8 bytes per pixel

Texture data: (0.5 bytes per texel) * (4 texels per pixel) * (2x overdraw) * (1-0.8 cache hit rate) = 0.8 bytes per pixel

Vertex data per second: (240000 bytes per frame) * (30 fps) = 7.2 MBytes/second

Total: (384000 * (8.8 + 16 + 8 + 0.8) + 240000) * 30 = 394.272 MBytes/second

7. POWER REDUCTION FEATURES

7.1 Clock Gating

The D/AVE 3D IP uses gated clocks to reduce power consumption.
One global gated clock signal comes from a top-level clock gate. It supplies the whole IP except the register interface and the interrupt controller and is located inside the Status/Configuration Controller.
Two SW accessible register bits (EnableAutoClkGating and DisableClk) inside the Status/Configuration Controller control the enabling of this clock either permanently or dynamically depending on the busy status of the internal modules.

This control works as follows:
When ‘EnableAutoClkGating’ is set to ‘1’, the automatic clock gating is used: The clock is enabled on a write to the stream start address register and disabled when the or’ed sum of all core busy signals goes low again, indicating the end of all action in the core.
When ‘EnableAutoClkGating’ is set to ‘0’, the clock is directly controlled by ‘DisableClk’: The clock is enabled when ‘DisableClk’ is set to ‘0’ and disabled, when ‘DisableClk’ is set to ‘1’.

7.2 Memory Block Disabling

All memory block use read respectively write enable signals. These can easily be used to fully disable the memory blocks during cycles when no accesses are performed.

7.3 Preparation for Automatic Clock Gating

The whole pipeline structure of the D3 IP is designed such that all pipeline registers only get an enable signal, when they need to register valid input data.
This ensures that the performance gain reached by automatic clock gate insertion by means of synthesis tools (e.g. Cadence RTL Compiler, Synopsys Power Compiler) is maximized.

8. INTEGRATION OVERVIEW

This chapter gives a very brief introduction about how to integrate the IP.

8.1 Configurable Constants

The following constants can be configured (default values are denoted in brackets).

DAVE3_MBA_ADDRESSWIDTH	Master address width in bits [32].
DAVE3_MBA_DATAWIDTH	Master data bus width in bits [32].
DAVE3_MBA_BURSTCOUNTWIDTH	Master burst count width in bits [5]. E.g. 5 means 2^5=32 words maximum burst length (must be consistent with cache line length).
DAVE3_FBCACHE_LINE_SEL_WIDTH	Width of selector signal for Framebuffer cache lines [3]. E.g. 3 means 2^3=8 cache lines in Framebuffer cache.
DAVE3_FBCACHE_WORD_SEL_WIDTH	Width of selector signal for Framebuffer cache word inside a line [5]. E.g. 5 means 2^5=32 words per cache line.
DAVE3_TEXCACHE_LINE_SEL_WIDTH	Width of selector signal for texture cache lines [2]. E.g. 2 means 2^2=4 cache lines in texture cache.
DAVE3_TEXCACHE_WORD_SEL_WIDTH	Width of selector signal for texture cache word inside a line [5]. E.g. 5 means 2^5=32 words per cache line.
DAVE3_ZSACACHE_LINE_SEL_WIDTH	Width of selector signal for ZSA cache lines [3]. E.g. 3 means 2^3=8 cache lines in ZSA cache.
DAVE3_ZSACACHE_WORD_SEL_WIDTH	Width of selector signal for ZSA cache word inside a line [5]. E.g. 5 means 2^5=32 words per cache line.

Table 8-1: Configurable constants

8.2 Clock

D/AVE is running in a single clock domain. The active edge is the rising edge. Please note, that a global clock gate is used (see 7.1).
The core does not have any multi cycle paths.

8.3 Reset

The reset is fully asynchronous and active high. When the reset signal goes to low, the IP is ready for work after the next clock cycle.

8.4 Power Management

No known restrictions.

8.5 Memory Description

The D3-IP requires different types of memories with configurable data width.
The design assumes input registers for the memory, so that the memories have an internal latency of one clock cycle.

The different types of memories are:

- Single Port – One Read/Write-Port
- Simple Dual Port – One Read-Port and one Write-Port
- Full Dual Port – Two Read/Write-Ports

9. DEVELOPMENT/VERIFICATION TOOL CHAIN

TES has developed a whole tool chain for both development and verification around the D3 IP. The flow is used under both Windows and Linux. The basic scripting language is Python. The core part of the tool chain is a regression test framework, which runs both the module level and toplevel test cases and compares the results of RTL simulations to the output of an HW exact see model.

See the following list for the details on used tools and versions:

Operating Systems	Microsoft Windows XP SP3 Linux 2.6.18-53.el5xen (CentOS)
Simulators	Mentor Graphics Modelsim-Altera 6.1g Cadence NC-Flow 08.10-s008
Synthesis	Altera Quartus 8.1 Cadence RTL Compiler v08.10-s126_1
Check Tools	Code Coverage → Cadence IUS (part of NC-Flow) LINT checking → Atrenta SpyGlass 4
Build Tools	gcc, CMake, Python

10. RESOURCE USAGE

This chapter gives a first indication of the resource consumption. Please note that the resources usage, especially for the memories, depends very much on the core parameters (see 8.1).

10.1 ASIC Gate Count

Final ASIC area numbers are not yet available. The will be provided in this section as soon as possible.

10.1.1 Logic Gates

The IP requires overall ~1.2 million gates (NAND2 equivalents) without memories.

10.1.2 Memories

Memory blocks inside the D3 IP are used at various locations:

- FIFOs
- Program RAM, register banks and constant tables for the programmable units (VSP and GSP)
- Vertex buffer
- Cache memory

The FIFO sizes and the cache memory sizes are configurable to balance the core performance. In addition, the cache memory size depends on the external data bus width, as the cache memories always need to be as wide as the external bus.

Due to this reason, it's only possible to give a rough overview of the necessary number of memory bits:

- <400 kBits for full performance settings
- <300 kBits for balanced performance settings
- <200 kBits for reduced performance settings (single texture unit)

Most of the internal memories can be implemented as register banks alternatively.

10.1.3 FPGA

Preliminary synthesis results for Altera FPGAs, using the default core parameters (not optimized):

Device	Logic count	Multiplier count	Memory count
Cyclone III (EP3C120)	99.583 LEs	242 9-bit DSP elements	202 M9Ks

11. PRODUCTIVITY TOOL SUPPORT

To ease application development, TES provides the following tools along with the D/AVE 3D HW/SW IP core.

11.1 SoftD/AVE 3D Emulator

An accurate and performance optimized D/AVE 3D emulator provided as dll for Windows based systems. Allows full feature usage and application development without a real target system.

11.2 Texture Compression Tool

A plug-in provided for ATIs 'Compressonator' tool, which allows encoding a given bitmap into the proprietary D/AVE 3D Texture Compression 'D3TC'.

11.3 Environment Map Conversion Tool

A tool to convert between various environment map formats like cube, spherical and dual-paraboloid environment maps.

IMAGES

Image 2-1 : Interface Diagram.. 8
Image 2-2 : Interface Diagram – Slave Bus Interface.. 9
Image 2-3 : Waveform Diagram – Slave Bus Interface 9
Image 2-4 : Interface Diagram – Master Bus Interface..10
Image 2-5 : Waveform Diagram – Master Bus Interface 10
Image 2-6 : SW Interfaces 13
Image 3-1 : Block Diagram – D/AVE 3D Dataflow...16
Image 3-2 : Block Diagram - Pixel Pipeline 18

TABLES

Table 3-1: Bilinear (left), Edge-Based (right) 20
Table 3-2 : uncompressed 24Bit per Pixel (left), 4Bit per Pixel compressed (right) 21
Table 8-1: Configurable constants..31