Wire-Frame 3D Graphics Accelerator IP Core Specification

Kenji Ishimaru <info.wf3d@gmail.com>

Ishimaru Information Engineering Office, All rights reserved.

Revision History

Rev.	Date	Author	Description
1.0	2015/09/30	Kenji Ishimaru	First Release
1.1	2016/08/14	Kenji Ishimaru	Many bugfix, Change email address

Ishimaru Information Engineering Office, All rights reserved.

Contents

Ishimaru Information Engineering Office, All rights reserved.

Ishimaru Information Engineering Office, All rights reserved.

1 Introduction

The “Wire-Frame 3D Graphics Accelerator IP Core” is a real-time 3D graphics rendering IP Core. The IP Core reads 3D triangle vertices from memory, then transforms them into 2D space, and writes 2D triangle’s edge line to memory. Note that this IP Core only supports Wire-Frame 3D graphics. In other words, the IP Core does not have traditional 3D graphics rendering features, such as polygon filling, texture mapping, lighting, etc. Instead of lacking these features, this IP Core has several advantages, such as small logic consumption and low-bandwidth memory access.

1 - 1 Features

RTL Design	
	Small logic consumption
	Single clock operation
	Independent from any vendor specific module
Interface	
	WISHBONE master x 1 (32-bit bus)
	WISHBONE slave x 1 (32-bit bus)
3D Graphics	
	Rendering image size: up to 2014 x 1536(QXGA)
	Matrix transformation
	Polygon Clipping
	Back-face culling
	DMAC for vertex reading
	2D line rasterization

Table 1 IP Core features

This IP Core is NOT including:

- Filled triangle rasterization, texture mapping and lighting.
- DMAC for clearing frame buffer

Ishimaru Information Engineering Office, All rights reserved.

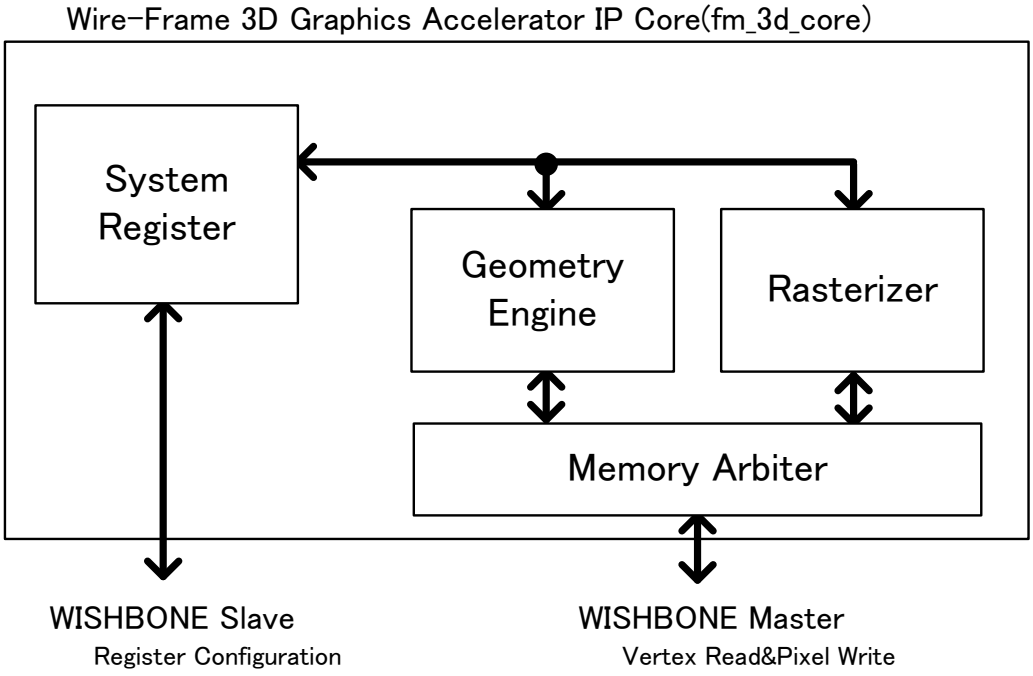


Figure 1 Interface and Internal architecture

1 - 2 System Requirement

This IP Core only generates 2D frame images from 3D triangle data. The real-time 3D graphics system would require additional system resources (Table 2).

Resource	Purpose
Processor	3D scene control, IP Core control(register configuration)
Memory	The storage for 3D scene object and frame buffers
LCD Controller	Displaying rendered images stored in frame buffer
DMAC(or Processor)	Frame buffer initialization

Table 2 Additional system resources

Ishimaru Information Engineering Office, All rights reserved.

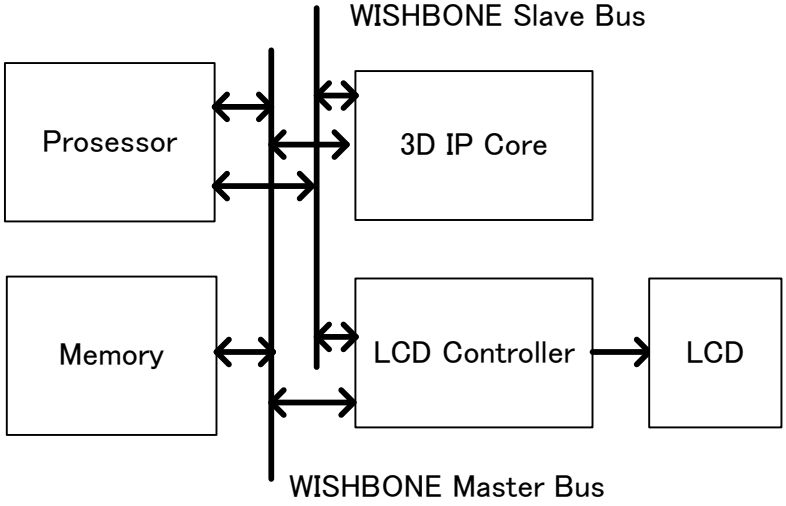


Figure 2 3D System Example

Ishimaru Information Engineering Office, All rights reserved.

2 Architecture

The IP Core contains four sub-modules.

- System Register
- Geometry Engine
- Rasterizer
- Memory Arbiter

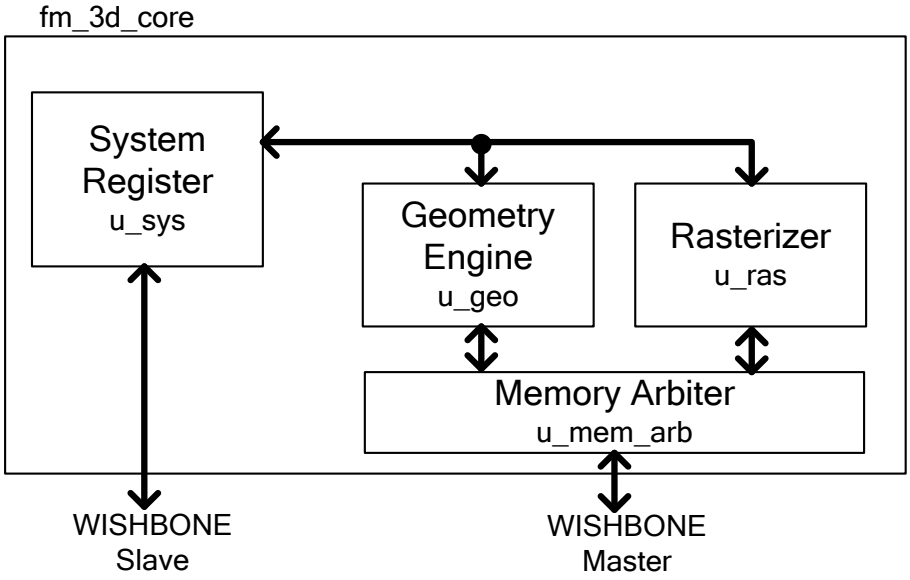


Figure 3 Architecture

2 - 1 System Register

System Register module stores all configuration registers for the IP Core. External master (processor) via WISHBONE slave interface configures all registers. The WISHBONE slave interface is only connected to the System Register module. System Register module distributes registers to other modules as direct signals.

Ishimaru Information Engineering Office, All rights reserved.

2-2 Geometry Engine

2-2-1 Overview

The Geometry Engine contains seven sub-modules (Figure 4). These seven modules are connected serially and construct vertex-processing pipeline. Geometry Engine reads 3D triangles from external memory by using DMAC, and generates 2D projected triangles.

Module Name	Instance Name	Description
fm_geo_mem	u_geo_mem	DMAC for reading triangle vertices
fm_geo_matrix	u_geo_matrix	4x4 matrix calculation
fm_geo_clip	u_geo_clip	Clipping code generation
fm_geo_persdiv	u_geo_persdiv	Perspective division
fm_geo_viewport	u_geo_viewport	Viewport transformation
fm_geo_tri	u_geo_tri	Triangle construction
fm_geo_cull	u_geo_cull	Back-face culling

Table 3 Geometry Engine sub-modules

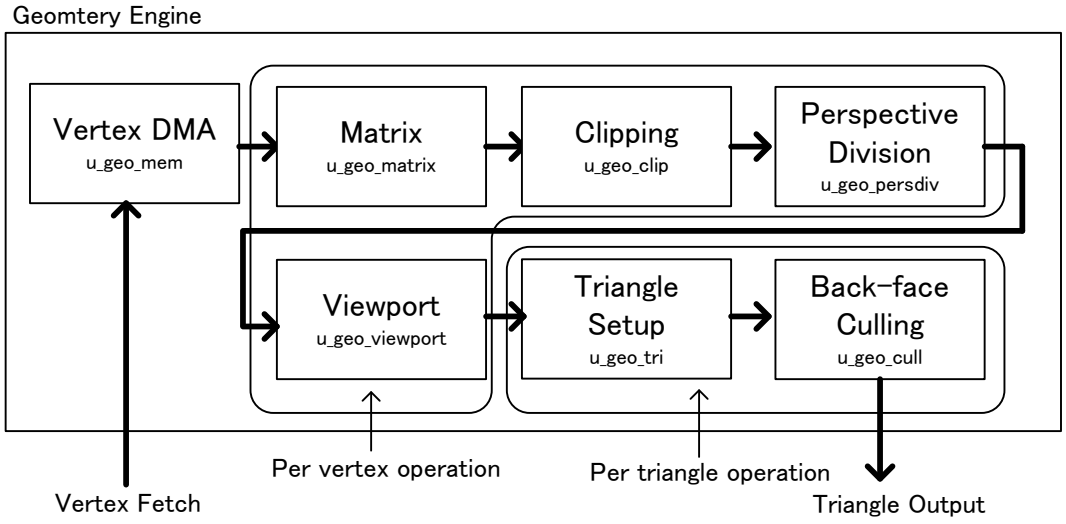


Figure 4 Geometry Engine internal modules

Ishimaru Information Engineering Office, All rights reserved.

2 - 2 - 2 Vertex DMA (fm_geo_mem)

This module reads vertices from main memory via WISHBONE master interface. The main memory access is started from VDMA_ADDR, to the end of the address VDMA_ADDR +(VDMA_SIZE-1)*4.

2 - 2 - 3 Matrix Calculation (fm_geo_matrix)

This module calculates matrix transformation. The input vertex from Vertex DMA is transformed by the 4 x 4 matrix. The matrix transforms incoming vertices from object coordinates to clip coordinates. Before starting Vertex DMA, all matrix elements should be configured to appropriate value. Typically, the matrix elements stores Model-View-Projection matrix elements for current 3D Object.

$$\begin{pmatrix} VC_x \\ VC_y \\ VC_z \\ VC_w \end{pmatrix} = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix} \begin{pmatrix} VO_x \\ VO_y \\ VO_z \\ VO_w \end{pmatrix}$$

VO: object coordinates vertex
VC: clip coordinates vertex

2 - 2 - 4 Clipping (fm_geo_clip)

This module calculates 6-bit clipping code (outcode[5:0]). Each bit of outcode stores the information that current vertex is inside or outside of the clipping plane. Note that this module only calculates clipping code, and does not discard any vertex.

Bit position	Operation	Plane
outcode[0]	Vx – Vw	-X
outcode[1]	Vx + Vw	+X
outcode[2]	Vy – Vw	-Y
outcode[3]	Vy + Vw	+Y
outcode[4]	Vz – Vw	-Z
outcode[5]	Vz + Vw	+Z

Table 4 Clipping Outcode

outcode condition n[0..5]	Vertex position

Ishimaru Information Engineering Office, All rights reserved.

outcode[n] < 0	outside of clipping plane[n]
outcode[n] = 0	on the plane of clipping plane[n]
outcode[n] > 0	inside of clipping plane[n]

Table 5 outcode condition

2-2-5 Perspective Division (fm_geo_persdiv)

This module projects vertices from 3D space to 2D space. The projection is accomplished by dividing incoming vertex's (x,y,z) by w. This module transforms incoming vertices from clip coordinates to normalized device coordinates.

$$\begin{pmatrix} VD_x \\ VD_y \\ VD_z \end{pmatrix} = \begin{pmatrix} \frac{VC_x}{VC_w} \\ \frac{VC_y}{VC_w} \\ \frac{VC_z}{VC_w} \end{pmatrix}$$

VC: clip coordinates vertex
VD: normalized device coordinates vertex

2-2-6 Viewport Mapping (fm_geo_viewport)

This module maps 2D projected vertices to 2D device coordinates. This module transforms incoming vertices from normalized device coordinates to window coordinates.

$$\begin{pmatrix} VW_x \\ VW_y \end{pmatrix} = \begin{pmatrix} \frac{SCREEN\_WIDTH}{2} VD_x \\ \frac{SCREEN\_HEIGHT}{2} VD_y \end{pmatrix}$$

VD: normalized device coordinates vertex
VW: window coordinates vertex

Ishimaru Information Engineering Office, All rights reserved.

2 - 2 - 7 Triangle Construction (fm_geo_tri)

This module generates triangle from adjacent 3 vertices. If all outcodes of a triangle specifies outside of clipping plane, the triangle is rejected and does not output to next module.

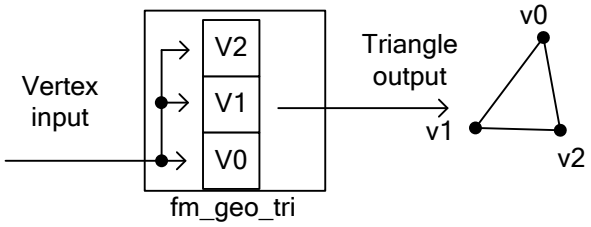


Figure 5 Triangle Construction

2 - 2 - 8 Back-face Culling (fm_geo_cull)

This module discards back-face triangles (if the back-face culling register GEO_CTR[8] is enabled). The back-face is decided by the vertex orientation of a triangle(sum of cross product of the triangle).

$$s = V_{x0}V_{y1} - V_{x1}V_{y0} + V_{x1}V_{y2} - V_{x2}V_{y1} + V_{x2}V_{y0} - V_{x0}V_{y2}$$

The culling operation deplanes on Clock-Wise Mode(GEO_CTR[16]).
If GEO_CTR[16] is 0, the triangle (s <0) is rejected, if GEO_CTR[16] is 1, the triangle (s > 0) is rejected.

2 - 3 Rasterizer

2 - 3 - 1 Overview

The Rasterizer contains 3 sub-modules. The Rasterizer gets 2D triangles from Geometry Engine, then generates triangle's edge line pixels. The Rasterizer only rasterize triangle's edge lines and does not fill triangle's internal pixels.

Module Name	Instance Name	Description
fm_ras_state	u_ras_state	Line distribution
fm_ras_line	u_ras_line	Pixel generation
fm_ras_mem	U_ras_mem	Memory interface

Figure 6 Rasterizer internal modules

Ishimaru Information Engineering Office, All rights reserved.

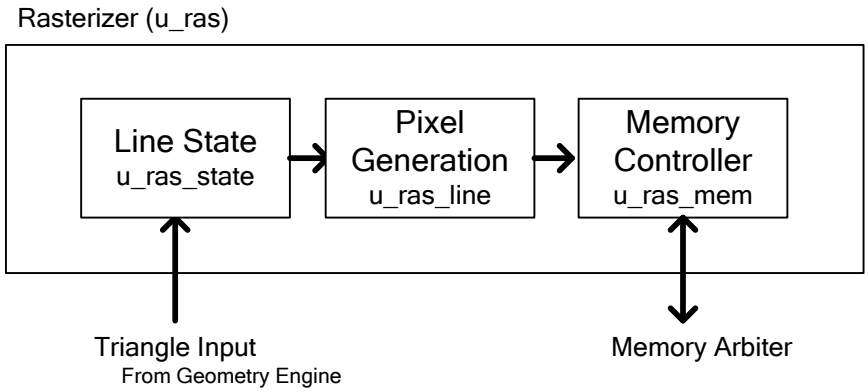


Figure 7 Rasterizer architecture

2-3-2 Line State

This module generates line data (a pair of the line vertices) from triangle. Each line data has start (x0, y0) and end (x1, y1) positions of the line. If both positions are outside of the viewport, the line data is discarded and does not output to the next module.

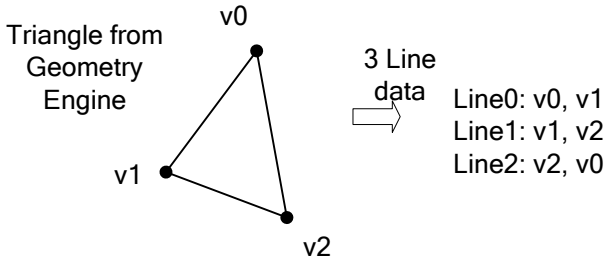


Figure 8 Line data generation

2-3-3 Pixel Generation

This module generates line pixels by using Bresenham's line algorithm. The pixel generation starts from (x0,y0) to (x1,y1). Figure 9 shows the detail of the pixel generation flow.

Ishimaru Information Engineering Office, All rights reserved.

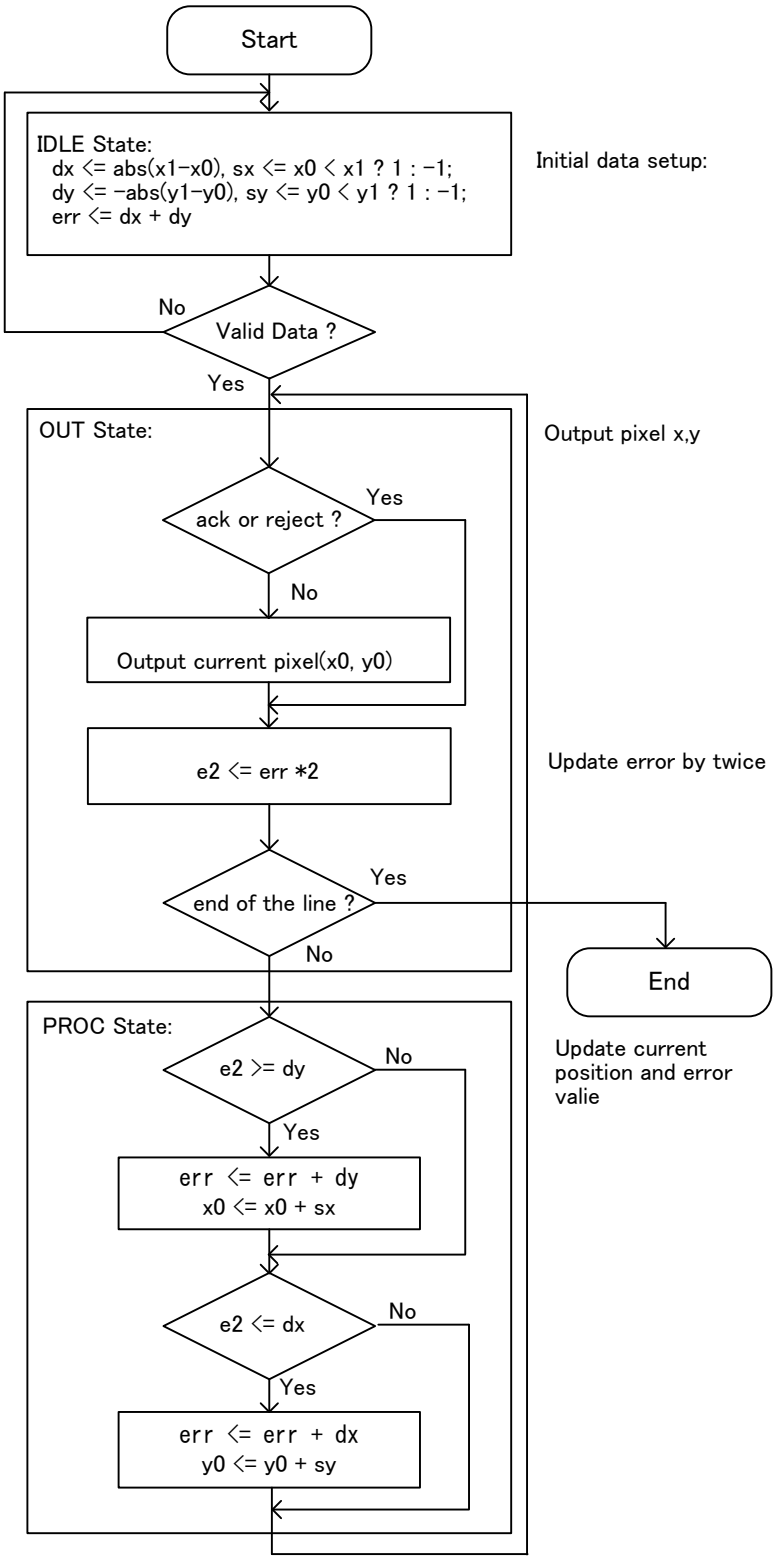


Figure 9 Pixel generation

Ishimaru Information Engineering Office, All rights reserved.

2 - 3 - 4 Memory Controller

This module generates linear memory address from (x, y) pixel position.
The linear address is :

Linear address = FB_ADDR + YF * SCREEN_WIDTH + x

Where:
screen Y-flip register is 0: YF = y
screen Y-flip register is 1: YF = SREEN_HEIGHT -1 – y

If screen Y-flip register is 0, the origin(pixel position (x,y) = (0,0)) is lower-left corner of the frame buffer. If screen Y-flip register is 1, the origin is upper-left corner of the frame buffer.

2 - 4 Memory Arbiter

The 3D Graphics IP Core has two master memory accesses internally. One is read access by Geometry Engine, the other is write access by Rasterizer. If both accesses occur simultaneously, this module always gives the higher priority to Geometry Engine.

Ishimaru Information Engineering Office, All rights reserved.

3 Operation

3 - 1 Overview

From the user side view, a single 3D-object drawing requires the following steps:

1. Vertex generation
2. Register setup
3. Start rendering
4. Interrupt(or status polling)

If there are multiple objects (data arrays) in a 3D scene, repeat the drawing steps from No.1 to No.4 for the number of data arrays.

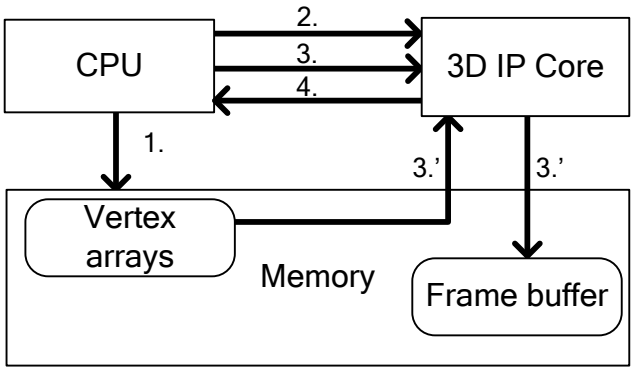


Figure 10 A single draw operations

3 - 2 Vertex generation

The IP Core needs vertex data arrays for the rendering. Typically, the vertex data arrays are stored in main memory. The data format of the vertex is IEEE single precision floating-point number. The IP Core only supports triangle array, and a triangle is constructed with 3 vertices. A vertex has x,y and z elements, and does not have w element.

Ishimaru Information Engineering Office, All rights reserved.

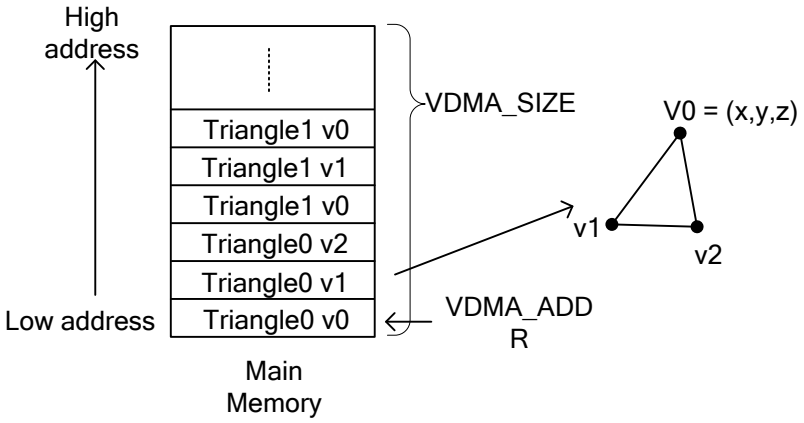


Figure 11 Vertex Array

3 - 3 Register Setup

Before start rendering, the following register configurations are required.

Category	Related Registers	Description
Screen Size	FSCR_W FSCR_H ISCR_W_M1 ISCR_H_M1 ISCR_W	Screen width and height
Vertex DMA	VDMA_ADDR VDMA_SIZE	DMA start address and size
Matrix transformation	MAT_EMT00-33	Model-View-Projection Matrix elements
Culling	GEO_CTR[8] GEO_CTR[16]	On/Off and clock-wise
Frame Buffer	FB_ADDR	Frame buffer address
Line Color	RAS_CTR[7:0]	Rasterize line color
Result Image	RAS_CTR[8]	Screen Y-flip On/Off

Table 6 Register Setup

3 - 4 Start Rendering

To start rendering, write 1 to GET_CTR[0].

After starting rendering, the IP Core reads current draw arrays from VDMA_ADDR, then writes rasterized pixels to FB_ADDR.

Ishimaru Information Engineering Office, All rights reserved.

3-5 Interrupt (or Status Polling)

When the current rendering is finished, INT_CTR[0] status is set to 1.
INT_CTR[0] status shows finish status in the following situation:
- Geometry Engine finished all vertices read
- All state machines in the IP Core modules are in IDLE state.

If INT_CTR[8] is 0, the interrupt also notifies to the processor that current rendering is finished.

Ishimaru Information Engineering Office, All rights reserved.

4 Registers

4 - 1 Overview

Name	Register Address	Description
GEO_CTR	0x00	Geometry Engine Control
INT_CTR	0x04	Interrupt Control
VDMA_ADDR	0x08	Vertex DMA Top Address
VDMA_SIZE	0x0C	Vertex DMA Size
MAT_EMT00	0x10	Matrix Register 00
MAT_EMT 01	0x14	Matrix Register 01
MAT_EMT 02	0x18	Matrix Register 02
MAT_EMT 03	0x1C	Matrix Register 03
MAT_EMT10	0x20	Matrix Register 10
MAT_EMT 11	0x24	Matrix Register 11
MAT_EMT 12	0x28	Matrix Register 12
MAT_EMT 13	0x2C	Matrix Register 13
MAT_EMT20	0x30	Matrix Register 20
MAT_EMT 21	0x34	Matrix Register 21
MAT_EMT 22	0x38	Matrix Register 22
MAT_EMT 23	0x3C	Matrix Register 23
MAT_EMT30	0x40	Matrix Register 30
MAT_EMT 31	0x44	Matrix Register 31
MAT_EMT 32	0x48	Matrix Register 32
MAT_EMT 33	0x4C	Matrix Register 33
FSCR_W	0x50	Screen Width(Floating Point)
FSCR_H	0x54	Screen Height(Floating Point)
ISCR_W_M1	0x58	Screen Width-1(Integer)
ISCR_H_M1	0x5C	Screen Height-1(Integer)
ISCR_W	0x60	Screen Width(Integer)
FB_ADDR	0x64	Frame Buffer Top Address
RAS_CTR	0x68	Rasterizer Control

Ishimaru Information Engineering Office, All rights reserved.

4-2 GEO_CTR

Bits	R/W	POR	Description
31:17	Reserved		
16	RW	1	Clock Wise Mode 0: Front-face is clock wise 1: Front-face is counter clock wise
15:9	Reserved		
8	RW	1	Enable back-face culling. Front face and back-face is controlled by clock Wise Mode 0: disable back-face culling 1: enable back-face culling
7:1	Reserved		
0	RW	0	Vertex DMA start bit. When the DMA is finished, this bit is automatically cleared. 0: IDLE 1: Start DMA, and DMA is working

Figure 12 shows clockwise mode. If back-face culling is enabled(GEO_CTR[8]=1), back-faced triangle is rejected and not rendered.

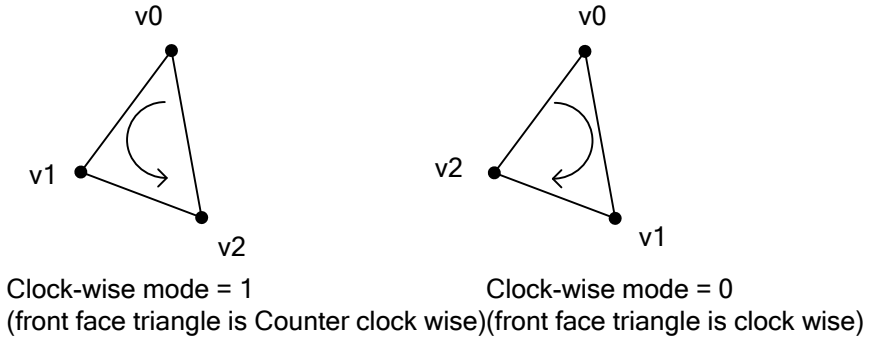


Figure 12 Clock-Wise Mode

4-3 INT_CTR

Bits	R/W	POR	Description

Ishimaru Information Engineering Office, All rights reserved.

31:9	Reserved		
8	RW	1	Interrupt Mask 0: not masked. Interrupt signal is output from o_int 1: Masked.
7:1	Reserved		
0	RW	0	Interrupt Status. The register write to this register clear the status. 0: IDLE state or rendering is not finished. 1: Rendering finished.

4 - 4 VDMA_ADDR

Bits	R/W	POR	Description
31:2	RW	Unknown	DMA Top Address
1:0	R	0	Always 0

VDMA_ADDR is a vertex fetch DMA top address. This is a 32-bit address value, and the lowest 2 bits are not configurable (fixed as 0).

4 - 5 VDMA_SIZE

Bits	R/W	POR	Description
31:16	Reserved		
15:0	RW	Unknown	DMA Size . count word size

VDMA_SIZE is the DMA vertex size. For example, one triangle size is 9(3vertices, each vertex has x,y,z). Max triangles are 65536/9. VDMA_SIZE must be the multiple of 3, because all the triangles construct with 3 floating-point values.

4 - 6 MAT_EMT00 – MAT_EMT33

Write access:

Bits	R/W	POR	Description
31:0	W	Unknown	Matrix Element Register IEEE floating point

Read access:

Bits	R/W	POR	Description
31:22	R	0	Reserved

Ishimaru Information Engineering Office, All rights reserved.

21:0	R	Unknown	Matrix Element Register 22-bits Floating Point

MAT_EMT00 – MAT_EMT33 have different value format between write access and read access. In write access, the write value should be IEEE single floating -point number. In read access, the returned value is 22-bit floating-point number for reducing logic consumption.
MAT_EMT00 – MAT_EMT33 is corresponding to the matrix elements as follows:

$$\begin{pmatrix} VO_x \\ VO_y \\ VO_z \\ VO_w \end{pmatrix} = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix} \begin{pmatrix} VI_x \\ VI_y \\ VI_z \\ VI_w \end{pmatrix}$$

Where:
VI_{xyzw} : input vertex from previous module(model coordinate)
VO_{xyzw} : output vertex to next module(clipping coordinate)

$$\begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix} =$$

$$\begin{pmatrix} MAT\_EMT00 & \text{MAT\_EMT01} & \text{MAT\_EMT02} & \text{MAT\_EMT03} \\ MAT\_EMT10 & \text{MAT\_EMT11} & \text{MAT\_EMT12} & \text{MAT\_EMT13} \\ \text{MAT\_EMT20} & \text{MAT\_EMT21} & \text{MAT\_EMT22} & \text{MAT\_EMT23} \\ \text{MAT\_EMT30} & \text{MAT\_EMT31} & \text{MAT\_EMT32} & \text{MAT\_EMT33} \end{pmatrix}$$

4 - 7 FSCR_W

Write access:

Bits	R/W	POR	Description
31:0	W	VGA:44200000h (640)	Screen Width Register IEEE singe precision floating point format, Default is VGA configuration sample value: XGA: 44800000h(1024) SVGA: 44480000h(800) VGA:44200000h (640) QVGA: 43a00000h(320)

Read access:

Bits	R/W	POR	Description

Ishimaru Information Engineering Office, All rights reserved.

31:22	Reserved		
21:0	R	18a000h	Screen Width Register 22-bits Floating point format, Default is VGA configuration sample value: XGA: 198000h(1024) SVGA: 18c800h(800) VGA: 18a000h(640) QVGA: 17a000h(320)

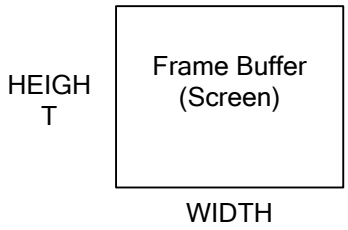


Figure 13 Screen Width and Height

FSCR_W is used for viewport mapping.
FSCR_W has different value format between write access and read access. In write access, the write value should be IEEE single floating -point number. In read access, the returned value is 22-bit floating-point number for reducing logic consumption.

4-8 FSCR_H

Write access:

Bits	R/W	POR	Description
31:0	W		Screen Height Register IEEE singe precision floating point format, Default is VGA configuration sample value: XGA: 44400000h(768) SVGA: 44160000h(600) VGA: 43f00000h(480) QVGA: 43700000h(240)

Read access:

Bits	R/W	POR	Description
21:0	R	17f000h	Screen Height Register 22-bits Floating Point, Default is VGA configuration sample value:

Ishimaru Information Engineering Office, All rights reserved.

			XGA: 18c000h(768) SVGA: 189600h(600) VGA: 17f000h(480) QVGA: 16f000h(240)

FSCR_H is used for viewport mapping.
FSCR_H has different value format between write access and read access. In write access, the write value should be IEEE single floating -point number. In read access, the returned value is 22-bit floating-point number for reducing logic consumption.

4-9 ISCR_W_M1

Bits	R/W	POR	Description
31:16	Reserved		
15:0	R/W	639	Screen Width - 1 Integer value, Default is VGA configuration sample value: XGA:1023 SVGA:799 VGA:639 QVGA:319

ISCR_W_M1 requires screen width -1 value. This value is used for pixel clipping over the screen width.

4-10 ISCR_H_M1

Bits	R/W	POR	Description
31:16	Reserved		
15:0	R/W	479	Screen Height - 1 Integer value, Default is VGA configuration sample value: XGA:767 SVGA:599 VGA:479 QVGA:239

ISCR_H_M1 requires screen height -1 value. This value is used for pixel clipping over the screen height, and used for screen Y-flipping.

Ishimaru Information Engineering Office, All rights reserved.

4-11 ISCR_W

Bits	R/W	POR	Description
31:16	Reserved		
15:0	R/W	640	Screen Width Integer value, Default is VGA configuration sample value: XGA:1024 SVGA:800 VGA:480 QVGA:240

ISCR_W requires screen width value. This value is used for the liner address generation for frame buffer.

4-12 FB_ADDR

Bits	R/W	POR	Description
31:20	RW	Unkown	Frame Buffer top address
19:0	R	0	Always 0

FB_ADDR is a 32-bit byte address value. The address specifies the top of frame buffer address. The lowest 2 bits are not configurable (fixed as 0).

4-13 RAS_CTR

Bits	R/W	POR	Description
31:9	Reserved		
8	RW	0	Screen Y flipping 0: not flip 1: flip
7:0	RW	Unknown	8-bit Pixel Color

RAS_CTR[7:0]:
The rasterized pixel is written to the frame buffer as RAS_CTR[7:0] value. Rasterizer only supports 8-bit pixel color. The mapping to RGB elements deepens on the system color format.

RAS_CTR[8]:
IF external LCD controller only support linear DMA from lower memory address, Screen Y flipping would be required.

Ishimaru Information Engineering Office, All rights reserved.

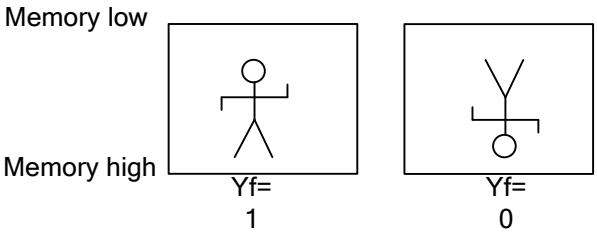


Figure 14 Screen Y-flipping

Ishimaru Information Engineering Office, All rights reserved.

5 IO Ports

5 - 1 Clock&Reset

Name	Bits	Direction	Description
clk_i	1	In	Clock
rst_i	1	In	Sync Reset

5 - 2 Interrupt

Name	Bits	Direction	Description
int_o	1	Out	Interrupt Out, level

5 - 3 WISHBONE Slave

Name	Bits	Direction	Description
s_wb_stb_i	1	In	Strobe input
s_wb_we_i	1	In	Write enable input
s_wb_adr_i[7:2]	6	In	Address input, word address (not including byte, 16-bit address)
s_wb_ack_o	1	Out	Acknowledge output
s_wb_sel_i[3:0]	4	In	Select input
s_wb_dat_i[31:0]	32	In	Data input
s_wb_dat_o[31:0]	32	Out	Data output

5 - 4 WISHBONE Master

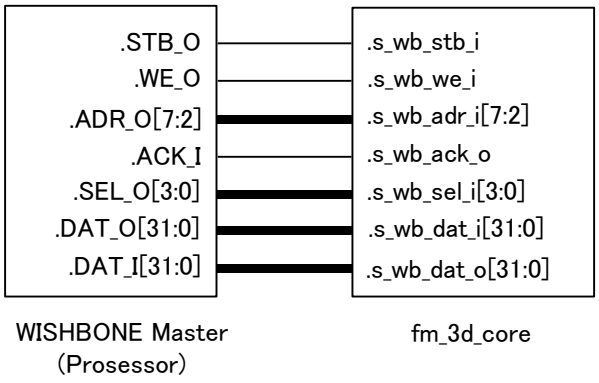
Name	Bits	Direction	Description
m_wb_stb_o	1	Out	Strobe output
m_wb_we_o	1	Out	Write enable output
m_wb_adr_o[31:2]	30	Out	Address output (word address)
m_wb_ack_i	1	In	Acknowledge input
m_wb_sel_o[3:0]	4	Out	Select output
m_wb_dat_o[31:0]	32	Out	Data output
m_wb_dat_i[31:0]	32	In	Data input

Ishimaru Information Engineering Office, All rights reserved.

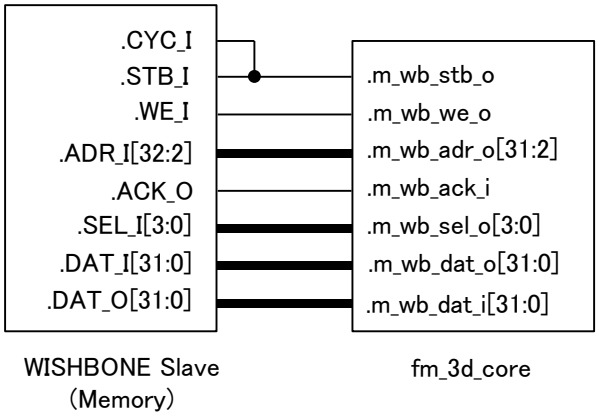
6 Wishbone Datasheet

Connection Example:

6 - 1 WISHBONE Slave



6 - 2 WISHBONE Master



Ishimaru Information Engineering Office, All rights reserved.

7 Floating Point Format

The Geometry Engine uses 22-bit floating-point format number. The format has 1-bit sign, 7-bit exponent and 16-bit mantissa. Note that 16-bit mantissa contains explicit one integer bit. It is not same as IEEE floating format, their format does not have implicit integer bit. The explicit integer bit reduces fraction resolution, but it has several advantages, when the intermediate successive floating point calculation in hardware.

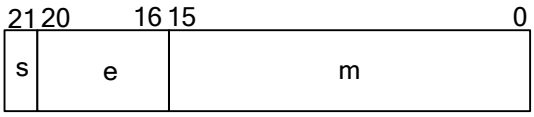


Figure 15 22-bit floating point format

$$\text{Value} = -1^{s} \times \text{m} \times 2^{e-15}$$

Bias = 15(Fh)
m = 1.15(1bit integer, 15bit fraction part)

s	e	m	Value
0	0x0f	0x8000	1.0
0	0x1f	0xffff	131071.0(positive max. number)
0	0x0e	0x8000	0.5
0	0x0e	0xc000	0.75
0	0x0d	0x8000	0.25
0	0x02	0x8000	0.000122
0	0x01	0x8000	0.000061
0	0x00	0x8000	0.000031(denormal)
0	0x00	0x0000	0
1	0xf	0x8000	-1.0
1	0x1f	0xffff	-131071.0.0
1	0xe	0x800	-0.5
1	0xe	0x800	-0.75
1	0x1	0x800	-0.000061
1	0x0	0x800	-0.000031(denormal)

Table 7 22-bit floating-point value example

Ishimaru Information Engineering Office, All rights reserved.

Biased e	Actual exponent	Biased e	Actual exponent
0x1f	2^{16}	0x0f	2^{0}
0x1e	2^{15}	0x0e	2^{-1}
0x1d	2^{14}	0x0d	2^{-2}
0x1c	2^{13}	0x0c	2^{-3}
0x1b	2^{12}	0x0b	2^{-4}
0x1a	2^{11}	0x0a	2^{-5}
0x19	2^{10}	0x09	2^{-6}
0x18	2^{9}	0x08	2^{-7}
0x17	2^{8}	0x07	2^{-8}
0x16	2^{7}	0x06	2^{-9}
0x15	2^{6}	0x05	2^{-10}
0x14	2^{5}	0x04	2^{-11}
0x13	2^{4}	0x03	2^{-12}
0x12	2^{3}	0x02	2^{-13}
0x11	2^{2}	0x01	2^{-14}
0x10	2^{1}	0x00	Denormal Number

Table 8 Biased exponent and actual exponent

Ishimaru Information Engineering Office, All rights reserved.