1 Overview of 3D-RAM and Its Functional Blocks

Overview of 3D-RAM and Its Functional Blocks

Introduction

One of the traditional bottlenecks of 3D graphics hardware has been the rate at which pixels can be rendered into a frame buffer using conventional DRAM or VRAM. The 3D-RAM emerged from a complete rethinking of frame buffer technology and produces an order of magnitude increase in rendering performance. The essence of the 3D-RAM architecture is: (1) an optimized array architecture that minimizes the average memory cycle time when rendering and (2) a selective on-chip logic that converts the interface with the rendering controller from a read-modified-write mode to a write-mostly mode. In addition to the performance boost, the new architecture also significantly reduces the system chip count. In 1994 Mitsubishi pioneered the introduction of the first member of the 3D-RAM family of products. This databook specifies all the features and operations of the third generation product of the 3D-RAM family to further elevate the performance of the 3D-RAM based 3D graphics systems. All references to 3D-RAM means the product M5M410092B, unless otherwise specifically designated.

The factors responsible for the dramatic overall performance improvement include:

- New Memory Architecture
  - 10-Mbits DRAM array supporting 1280 x 1024 x 8 frame buffer
  - Four independent, interleaved DRAM banks
  - 2048-bit SRAM Pixel Buffer as the cache between DRAM and ALU
  - Built-in tile-oriented memory addressing for rendering and scan line-oriented memory addressing for video refresh
  - 256-bit global bus connecting DRAM banks and Pixel Buffer
  - Flexible dual Video Buffer supporting 76-Hz CRT refresh
- Write Mostly Interface
  - On-chip ALU
  - Four ROP units supporting 16 raster operations on byte data
  - Four Blend units blending the old pixel value with new information
  - On-chip hardware acceleration for all OpenGL blending modes *(NEW)*
  - On-chip hardware acceleration for all OpenGL stencil modes *(NEW)*
  - One 32-bit Match Comparator and one 32-bit Magnitude Comparator
  - Concurrent operations of DRAM, Pixel Buffer, ALU and Video Buffer
  - 32-bit synchronous high-bandwidth data bus with rendering controller
  - Blending operations in both (8, 8, 8, 8) and (4 4, 4, 4) color modes *(NEW)*
  - One additional PASS_IN pin for flexible

Frame Buffer Design Example

Figure 1.1 is a simple frame buffer design example showing a 1280 x 1024 x 32 single buffered configuration. The rendering controller writes pixel data across the 128-bit bus to the four 3D-RAMs. The controller commands most of the 3D-RAM operations, including ALU functions, Pixel Buffer addressing, and DRAM operations. The controller can also command video display by setting up the RAMDAC and requesting video transfers from 3D-RAMs.

With the 128-bit pixel data bus shown in Figure 1.1, four pixels can be moved across the bus in one cycle. There are two ways to organize the 3D-RAMs: (1) Each 3D-RAM holds one of the

8-bit color components—R, G, B, or a—for all 1280 x 1024 pixels; (2) Each 3D-RAM holds all 32 bits of a pixel value for 320 x 1024 pixels, allowing fast scrolling in the vertical direction and interleaving four 3D-RAMs in the horizontal direction.

If the width of the data bus from the rendering controller to 3D-RAM is reduced to 64 bits, then two pixels are transferred in one cycle. Similarly, a 32-bit data bus can transfer only one pixel at a time.

Chapter 6 provides more examples of frame buffer organizations using 3D-RAMs, such as 1280 x 1024 x 8, 320 x 1024 x 32, etc.

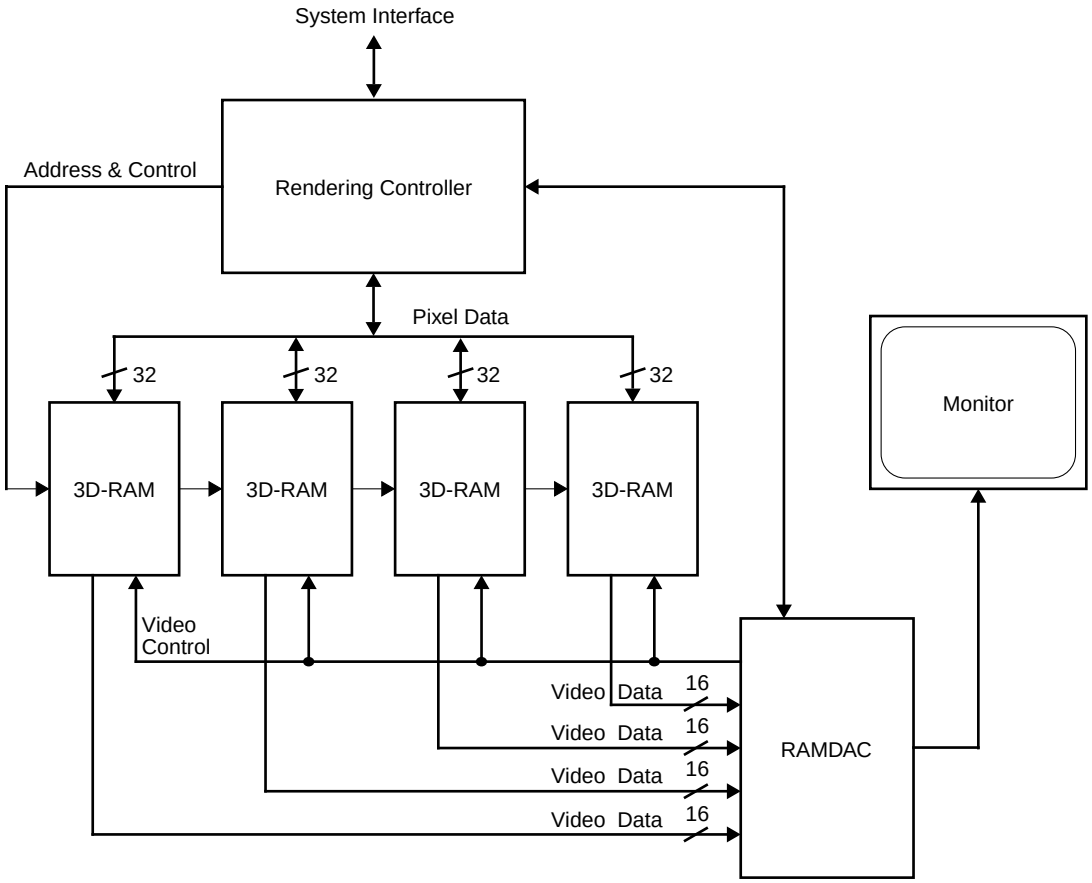


Figure 1.1 1280 x 1024 x 32 frame buffer consisting of four 3D-RAMs, shown together with a rendering controller and a RAMDAC

Simplified 3D-RAM Block Diagram

The 3D-RAM block diagram is shown in Figure 1.2. The DRAM array is partitioned into four independent banks of 2.5 Mbits each. Together, these four banks can support a screen resolution of 1280 x 1024 x 8. The independent banks can be interleaved to facilitate almost uninterrupted frame buffer update and, at the same time, can transfer pixel data to the dual Video Buffer for screen refresh. Data from the DRAM banks is transferred over the 256-bit Global Bus to the triple-ported Pixel Buffer. The Pixel Buffer consists of eight blocks, each of which is 256 bits and is updated in a single transfer on the Global Bus. Hence, the memory size of the Pixel Buffer is 2 Kbits. The ALU uses two of the Pixel Buffer ports to read and write data in the same clock cycle. Each Video Buffer is 80 x 8 bits and is loaded in a single DRAM operation. One Video Buffer can be loaded while the other is sending out video data.

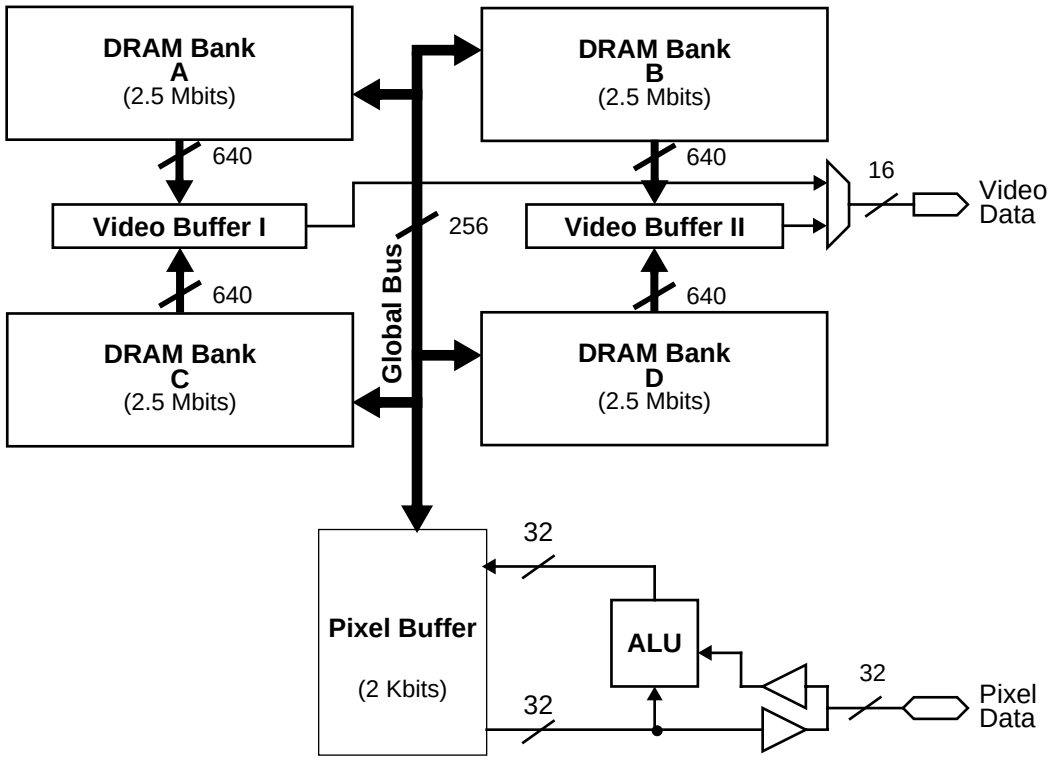


Figure 1.2 Simplified 3D-RAM block diagram

3D-RAM Functional Blocks

The 3D-RAM has five major functional blocks in: DRAM banks, Video Buffers, Pixel Buffer, Global Bus, and Pixel ALU. The following sections provide a quick overview of each of these functional blocks. Chapter 3 describes details of the Pixel ALU operations, Chapter 4 presents specifics of the DRAM operations, and Chapter 5 provides examples of parallelism between the Pixel ALU operations and the DRAM operations. Now, to give readers a better grasp of these functional blocks, we first describe the memory units on which these functional blocks operate.

Block, Page, and Page Group

A word has 32 bits and is the unit of data operations within the Pixel ALU and between the Pixel ALU and Pixel Buffer. When the Pixel ALU accesses the Pixel Buffer, not only a block address needs to be specified but also a word has to be identified. Since there are eight blocks in the Pixel Buffer and eight words in a block, the upper three bits of the input pins PALU_A designate the block, and the lower three bits select the word. The data in a word is directly mapped to $PALU\_DQ_{[31:0]}$ in corresponding order. That is, bit 0 of the word is mapped to PALU_DQ0, bit 1 to PALU_DQ1, and so on.

Although an ALU write operation operates on one word at a time, each of the four bytes in a word may be individually masked. The mapping is also direct and linear: byte 0 is $PALU\_DQ_{[7:0]}$, byte 1 $PALU\_DQ_{[15:8]}$, byte 2 $PALU\_DQ_{[23:16]}$, and byte 3 $PALU\_DQ_{[31:24]}$.

A block has 256 bits and is the unit of memory operations between a DRAM bank and the Pixel Buffer over the Global Bus. The input pins DRAM_A selects a block from the Pixel Buffer and a block from a page of a DRAM bank. The DRAM operations on block data are Unmasked Write Block (UWB), Masked Write Block (MWB), and Read Block (RDB). These operations are described in detail on page 44, “Description of DRAM Operations.”

A page in a DRAM bank is organized into 10 x 4 blocks. Since a block has 256 bits, a page has 10,240 bits. There are four DRAM banks in a 3D-RAM chip, the pages of the same page address from all four DRAM banks compose a page group. Therefore, a page group has 20 x 8 blocks.

Note in Figure 1.3, the block and page are purposely drawn as rectangular shapes. The user may relate these to a tiled frame buffer memory organization. For example, if the display resolution is 1280 x 1024 x 8, then a (32-bit) word contains four pixels. Since a block may be viewed as having 2 x 4 words, it contains 8 x 4 pixels. A page is organized into 10 x 4 blocks, so it contains 80 x 16 pixels, and a page group holds 160 x 32 pixels. Finally, a screen is composed of 8 x 32 page groups. The advantage of such a frame buffer memory organization is the minimization of page miss penalty. 3D objects frequently occupy portions of multiple scan lines. Since in this case a page contains 80 x 16 pixels instead of 10,240 x 1 pixels, page miss is reduced. When an object extends beyond a page boundary, bank interleaving allows hidden precharge and uninterrupted memory access. Details of the various frame buffer memory organizations using 3D-RAMs are discussed in Chapter 6.

On the other hand, to support screen refresh, the Video Buffer must output pixel data one scan line at a time. The internal organization of a page also allows data to be transferred from a page to the Video Buffer, one of the sixteen scan lines of 80 bytes long each at a time. See the section “Video Buffers” on page 7 for a summary and the section “Video Transfer (VDX)” on page 46 for full details.

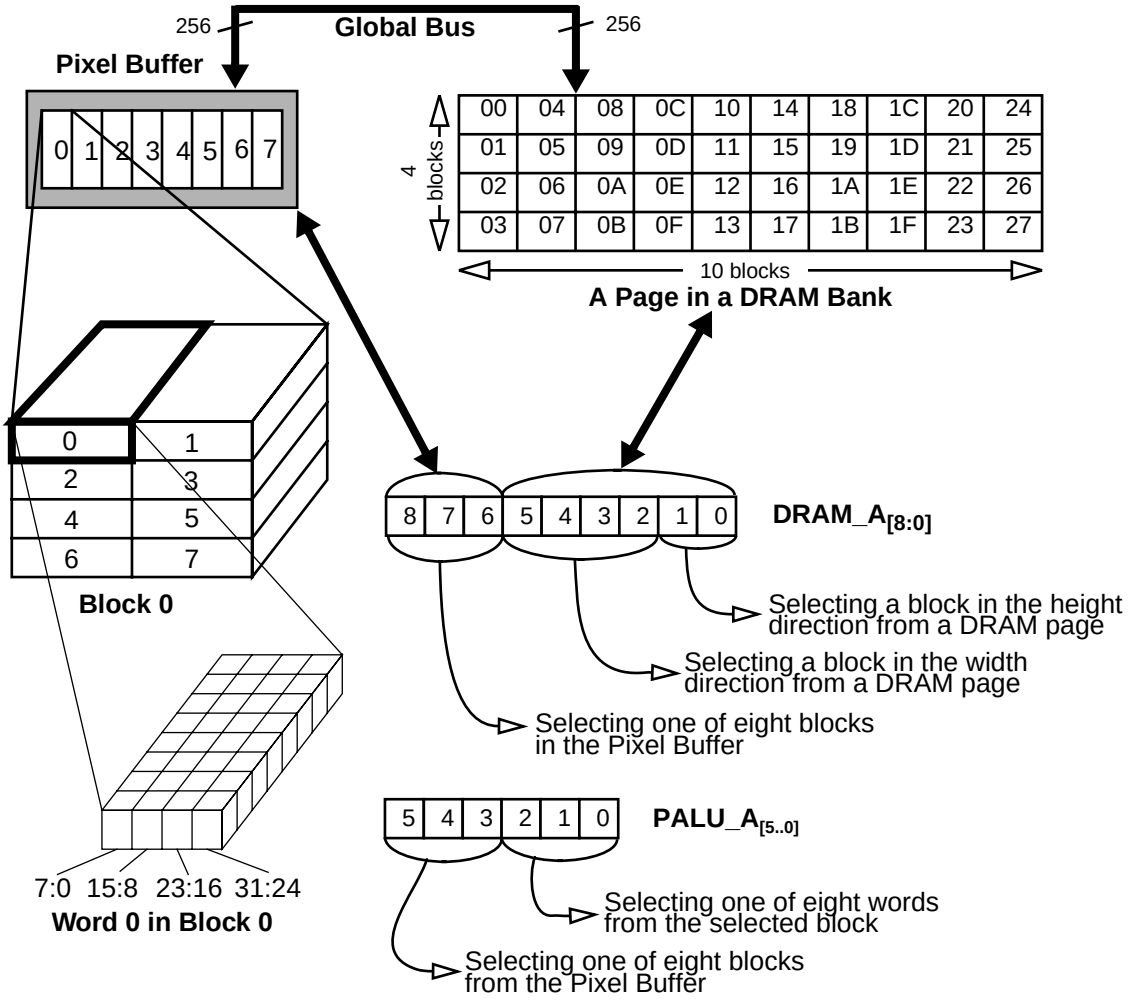


Figure 1.3 Relations and addressing scheme of blocks and words in the Pixel Buffer and in the DRAM page

DRAM Banks and Basic DRAM Operations

The 3D-RAM contains four independent DRAM banks which can be interleaved to facilitate hidden precharge or access in one bank while screen refresh is being performed in another bank. Each DRAM bank has 256 pages with 10,240 bits per page for a total storage of 2,621,440 bits. An additional 257th page can be accessed for special functions or used to hold off-screen data. A row decoder takes 9-bit page address signals to generate 257 word lines, one for each page. The word lines select which page is connected to the sense amplifiers. The sense amplifiers read and write the page selected by the row decoder. Because the sense amplifiers retain data after the read/write operations, they function like a direct-mapped level-two pixel cache. (The Pixel Buffer, which is discussed on page 7, functions as a level-one pixel cache in a frame buffer with 3D-RAMs.)

During an Access Page (ACP) operation, the row decoder selects a page by activating its word line. Activating the word line of a particular page transfers the bit charges of that page to the sense amplifiers. The sense amplifiers amplify the charges. After the sensing and amplification are completed, the sense amplifiers are ready to interface the Global Bus or Video Buffer. In a way, ACP may be viewed as a “write cache” operation on the sense amplifiers as a level-two pixel cache. Because the activated word line remains connected to the sense amplifiers after the ACP operation until the subsequent Precharge Bank operation, when a block of the sense amplifiers is updated by a block write operation (UMB or MWB), the corresponding block in the DRAM array is also updated. Therefore, the sense amplifiers function as a “write-through” cache, and no write back to the DRAM array is necessary. Alternatively, the data in the sense amplifiers can be written to any page in the same bank at this time, simply by selecting a word line without first equalizing the sense amplifiers. This function is called Duplicate Page (DUP). A typical application of this function is copying from the 257th page to one of the 256 normal pages—all 10,240 bits at a time—for fast area fill.

When the sense amplifiers in a DRAM bank completes the read/write operations with the Global Bus or Video Buffer, a Precharge Bank (PRE) operation usually follows. A Precharge Bank cycle simply deactivates the selected word line corresponding to the current page and equalizes the sense amplifiers. The PRE operation may be viewed as the close of a page access or as the preparation for the subsequent page access. The DRAM bank must be precharged prior to accessing a new page.

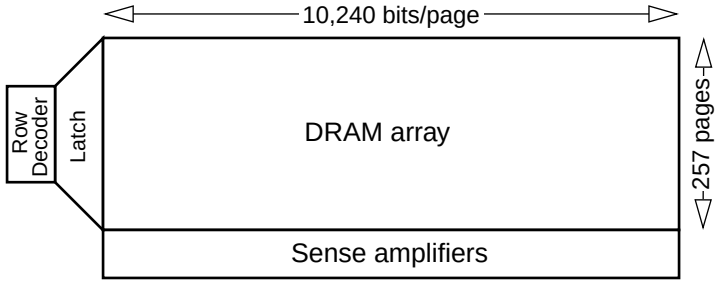


Figure 1.4 DRAM bank consisting of row decoder, address latch, DRAM array, and sense amplifiers

Pixel Buffer

The Pixel Buffer is a 2048-bit SRAM organized into eight 256-bit blocks, as seen in Figure 1.3, and functions as a level-one write-back pixel cache. It has a 256-bit read/write port, a 32-bit read port, and a 32-bit write port. Referring to Figure 1.6, the 256-bit read/write port is connected to the Global Bus via a Write Buffer, and the two 32-bit ports are connected to the Pixel ALU and the pixel data pins. All three ports can be used simultaneously as long as the same memory cell is not accessed. If the two 32-bit ports access the same cell, the write operation will be successful but the read data will be undefined.

A 1-bit Dirty Tag bit is assigned to each byte data in the Pixel Buffer. Therefore, each block in the Pixel Buffer is associated with a 32-bit Dirty Tag in the dual-port Dirty Tag RAM. When a block is transferred from the sense amplifiers to the Pixel Buffer through the 256-bit port, the corresponding 32-bit Dirty Tag is cleared. When a block is transferred from the Pixel Buffer to a DRAM bank, the Dirty Tag determines which bytes are actually written. This feature can save as much as 50% of the power consumed by a 256-bit block write operation without the Dirty Tag.

The cache set associativity is determined external to the 3D-RAM, thereby permitting optimal cache design tailored to the particular graphics system.

Video Buffers

Each video buffer receives 640-bit data at a time from one of the two DRAM banks connected to it. (The reader is reminded of the 3D-RAM block diagram in Figure 1.2.) sixteen bits of data are shifted out onto the video data pins every video clock cycle at 14-ns rate. It takes 40 video clocks to shift all data out of a video buffer. The video counter counts modulo 40 and toggles the buffer select line when the count wraps around to 0. These two video buffers can be alternated to provide a seamless stream of video data.

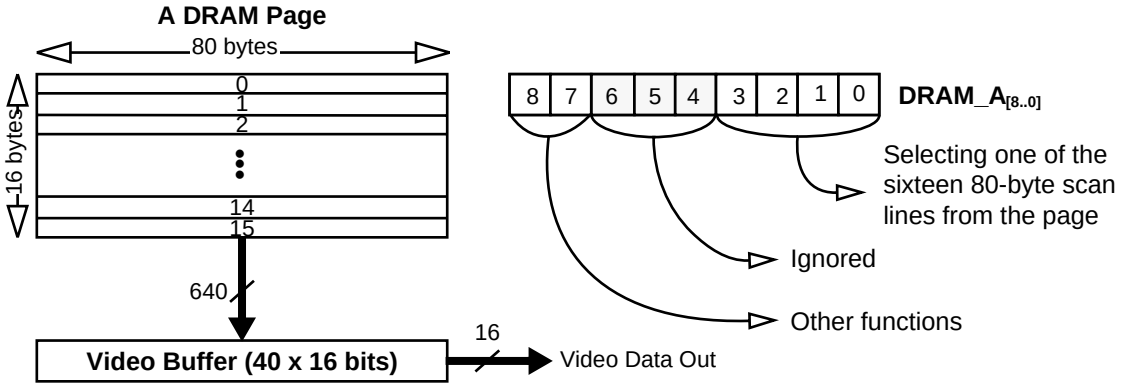


Figure 1.5 Video transfer from a DRAM page to the Video Buffer

Global Bus

The Global Bus connects the Pixel Buffer to the sense amplifiers of all four DRAM banks. The Global Bus consists of 256 data lines. Referring to Figure 1.6, during a transfer from the Pixel Buffer to DRAM, the 256 bits are conditionally written depending on the 32-bit Dirty Tag and the 32-bit Plane Mask. When a data block is transferred from the Pixel Buffer to the sense amplifiers, the Dirty Tag and Plane Mask control which bits of the sense amplifiers are changed via the Write Buffer.

Note that all read/write operations are viewed from the perspective of the rendering controller. In other words, a read operation across the Global Bus always means a read by the Pixel ALU; that is, data is transferred from a DRAM bank into the Pixel Buffer. Similarly, a write operation across the Global Bus means data is updated from the Pixel Buffer to a DRAM bank. This is also specifically noted in Figure 1.6 by the signals Global Bus Write Block Enable and Global Bus Read Block Enable.

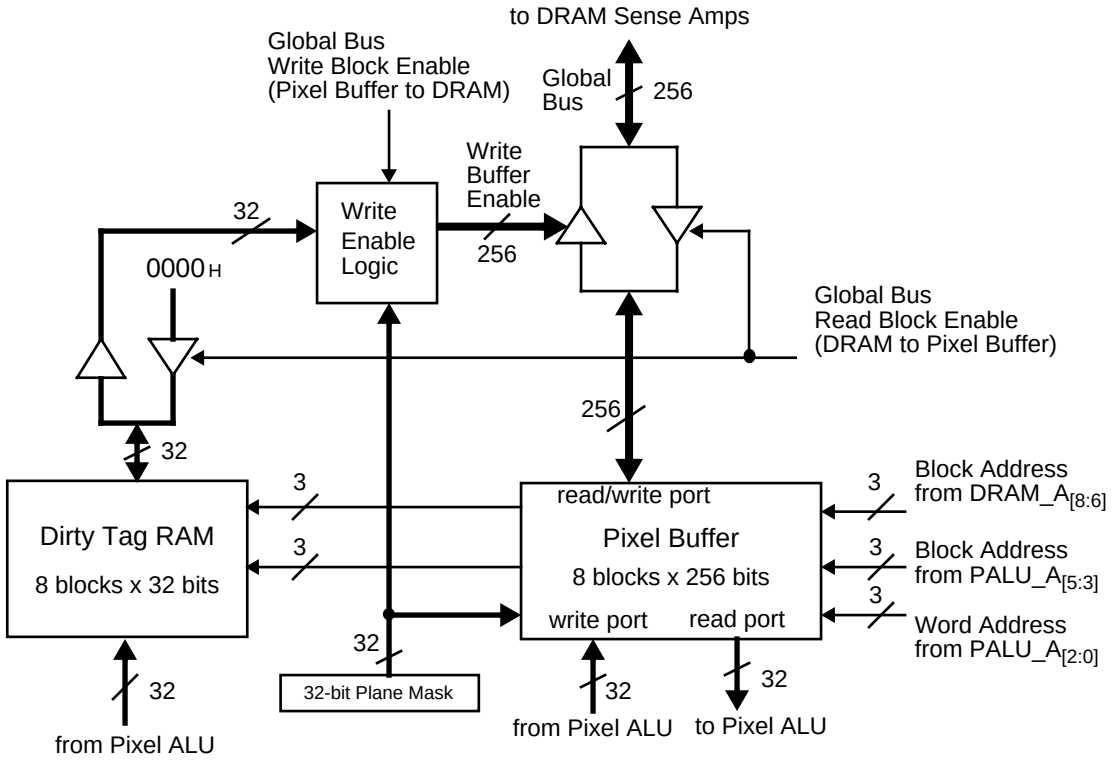


Figure 1.6 Tri-port Pixel Buffer, Global Bus and dual-port Dirty Tag RAM

Pixel ALU Basics

The Pixel ALU consists of four 8-bit ROP/Blend units, which may be independently programed to perform either a raster operation or a blending function, one 32-bit Match Compare unit, and one 32-bit Magnitude Compare unit. The two Compare units are also commonly referred to as the Dual Compare units. The motivation for including the Pixel ALU on chip is to convert the interface from a read-modify-write interface to a write-mostly interface. This logic integration with memory arrays greatly improves rendering throughput by avoiding time consuming reads and direction changes on the data bus.

The ROP/Blend units and the Dual Compare units are highly pipelined. Page 11 contains a brief discussion of the ALU pipeline. The output of a ROP/Blend unit is conditionally written to the Pixel Buffer, depending on the comparison results from the on-chip Dual Compare units and from the Dual Compare units of the preceding 3D-RAM chips. For example, for a 1280 x 1024 x 32 double-buffered graphics system with 32-bit Z buffer, there are effectively 96 bits per pixel. In this case, eight 3D-RAMs are used as color chips and four as Z chips. The Pixel ALUs of the Z chips perform magnitude comparisons and feed the comparison results via their PASS_OUT pins to the

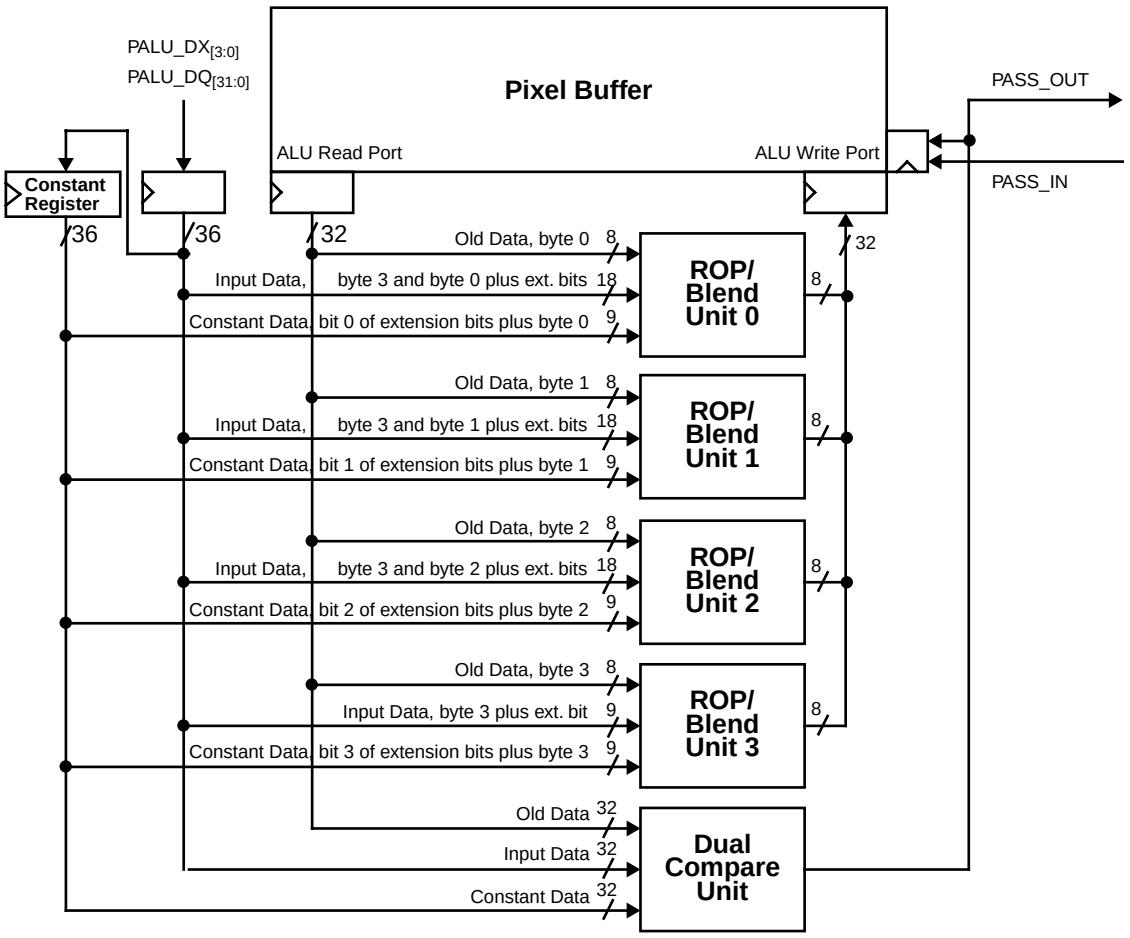


Figure 1.7 Pixel ALU (Pipeline stages are not shown.)

corresponding color chips. It is important to note that due to the pipelining, the color chips do not wait for the magnitude comparison results from the Z chips; rather, the results of the ROP/blending operations and comparison operations on the color chips, and the results of the magnitude comparison on the Z chips all are presented to the Pixel Buffer of the color chips in the same clock cycle. In this sense, the rendering controller can accomplish a pixel blending operation with Z compare and window ID compare all in a single clock cycle. Furthermore, because of the pipelining and the tri-ported architecture of the Pixel Buffer, the read and write operations may be performed on the Pixel Buffer of the 3D-RAM during the same clock cycle.

ROP/Blend Units

The ROP/Blend units can be configured as either a ROP unit or a Blend unit by setting a register bit. Each ROP unit can perform all 16 standard ROP functions. These functions are listed in Chapter 3. One of the operands of the ROP functions is the old data from the Pixel Buffer, and the other operand may be either the data from the primary I/O pins or the data from an internal register (called the Constant register). For the blending operation, the general equation is as follows:

Write data to Pixel Buffer
= New Term + (Old Data x Old Fraction)
= (New Data x New Fraction) +
(Old Data x Old Fraction)

The 3D-RAM Blend units accomplish what is called destination blending in a single MCLK cycle, that is, the addition and the second multiplication in the above equation. In this case, the rendering controller must perform the multiplication of New Data with New Fraction (i.e., the source blending) and present the result as the New Term to 3D-RAM. In addition, 3D-RAM can also accomplish the full blending by taking two MCLK cycles, with a loop back mechanism.

Dual Compare Unit

Physically, the Dual Compare units consist of one 32-bit Match Compare unit and one 32-bit Magnitude Compare unit. Both Match Compare and Magnitude Compare are done in parallel. One of the sources is always the old data from the Pixel Buffer. The other source is independently selectable between the data from the PALU_DQ pins and the data from the Constant source register. There are also two mask registers, namely Match Mask and Magnitude Mask, that define which bits of the 32-bit words will be compared and which will be “don’t care.”

One application of the Match Compare unit is Window ID comparison, and the Magnitude Compare unit is typically used in the depth comparison of a Z-buffer algorithm for hidden surface removal. When these Compare units are used together, the system can achieve hidden surface removal for only a specific window on the display in one cycle. Furthermore, since the data to be written into the Pixel Buffer always comes through the ROP/Blend units, a system with 3D-RAM can achieve a pixel update with a raster or blending operation specifically on only the new objects in the selected window that are closer to the viewer than the existing objects in the frame buffer.

The results of both Match Compare and Magnitude Compare operations are logically ANDed together to generate the PASS_OUT pin. The PASS_IN signal (fed from another 3D-RAM chip) and the internally generated PASS_OUT signal are then logically ANDed together to produce a Write Enable signal to the Pixel Buffer. Thus, the PASS_IN and PASS_OUT pins offer hardware support for display resolutions where multiple 3D-RAM chips are required, such as in the cases of 1280x 1024 x 32 (single color buffer plus Z buffer) and 1280 x 1024 x 96 (double color buffer plus Z buffer).

Pipelining

The 3D-RAM Pixel ALU pipeline is designed so that read and write operations can be performed with minimal delay. This is achieved by having all operations conform to a uniform 7-stage pipeline.

Figure 1.8 is an example that illustrates the efficiency afforded by the pipeline flow of Pixel ALU read/write operations. A pipeline stage begins with a rising edge of MCLK and ends before the next rising edge of MCLK. (As a matter, in the 3D-RAM all references to MCLK are relative to the rising edge except for some boundary scan test operations.) For clarity, separate stage counts are provided for the first read and first write operations and are labeled as R1 through R4 and W1 through W7, respectively. The Read A operation is asserted for two cycles; Read A is first presented in Stage R1 and latched into the 3D-RAM by Clock 1 in Stage R2. Data A is piped out by Clock 2 in Stage R3 and becomes stable for sampling in Stage R4. Between Read B and WC (Write C), two single-cycle NOPs are inserted to guarantee an idle cycle for the data bus to turn around. On the other hand, a read operation can immediately follow a write operation, as shown by Read G following WF. To allow maximum bandwidth for the rendering controller, a write operation may be started everything cycle. In this example, we start with the WC operation. The address and write instruction are presented in Stage W1 and latched into the 3D-RAM by Clock 7 in Stage W2; Data C and WD are presented in Stage W2 and latched into the 3D-RAM by Clock 8 in Stage W3. Then, after three cycles for internal processing, the valid PASS_OUT Pass C is piped out by Clock 11 in Stage W6. The actual updating of the Pixel Buffer takes place in Stage W7. Thus, n consecutive write operations take only 7 + n - 1 = n + 6 cycles to complete, including all internal activities. It is important to point out that the effective write cycle time from the perspective of the rendering controller interface is only n + 1 cycles for n consecutive write operations, as shown by WC through WF.

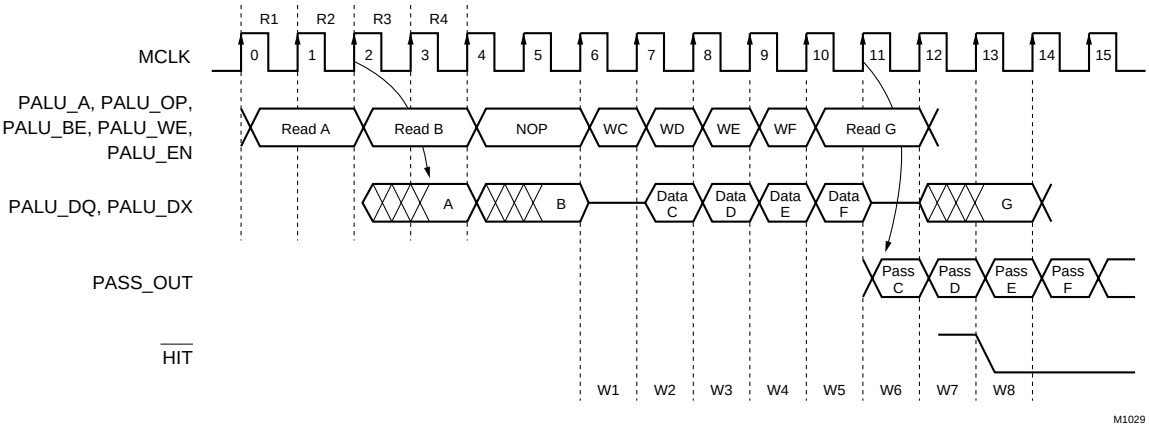


Figure 1.8 Example of Pixel Port read/write operations that satisfy the pipeline flow

The Picking Logic

From the user's view point, a common experience of the picking function in 2D computer graphics may be using the mouse and the associated cursor to select an icon on the display screen, resulting in the selected icon highlighted in a different color. This is a basic function in interactive computer graphics, and 3D-RAM provides the Picking Logic and the HIT pin to support this picking function for selection of objects in a 3D scene.

A picking function may involve redrawing the objects into the frame buffer and returning a list of objects that intersect with some predefined selection volume. When the user uses multiple 3D-RAMs in a frame buffer design to determine if a pixel data is successfully written by any Stateful Write operation (see "Pixel Data Operations" on page 40) during the redraw process, the comparison result on the PASS_OUT pin from each chip must be logically ANDed. If this logical operation is left to off-chip glue logic between the 3D-RAM frame buffer and the rendering controller, excessive delay is unavoidable in this critical timing path. If the rendering controller is to perform this logical operation, extra pins must be provided by both the 3D-RAM and the rendering controller, while delay is still significant. The Picking Logic brings the glue logic on chip and provides an open-drain HIT pin to interface with the rendering controller.

A block diagram of the Picking Logic is shown in Figure 1.9. Initially, the Picking Logic should be enabled and the HIT flag should be cleared, which is done by writing to byte 3 of the Compare Control Register. The HIT pin will be set to high (i.e., not driven low by 3D-RAM) after seven cycles (corresponding to the Pipeline Stage 8). In the figure below, this is indicated by the number 8 in the square box above the HIT pin label. This design of the pipeline flow for the HIT flag and the HIT pin prevents an incorrect HIT value from the Stateful Data Write operations before the Picking Logic is enabled. A sequence of Stateful Data Write operations may be issued immediately after the register writing. A low value on the HIT pin means that at least one of the Stateful Data Writes passed the on-chip and off-chip comparison tests and the pixel data was written to the Pixel Buffer. If the HIT pin is high, none of the Stateful Data Writes passed and no pixel is updated. See Figure 8.6, "Picking Logic Timing," for an illustration of the operations described in this section.

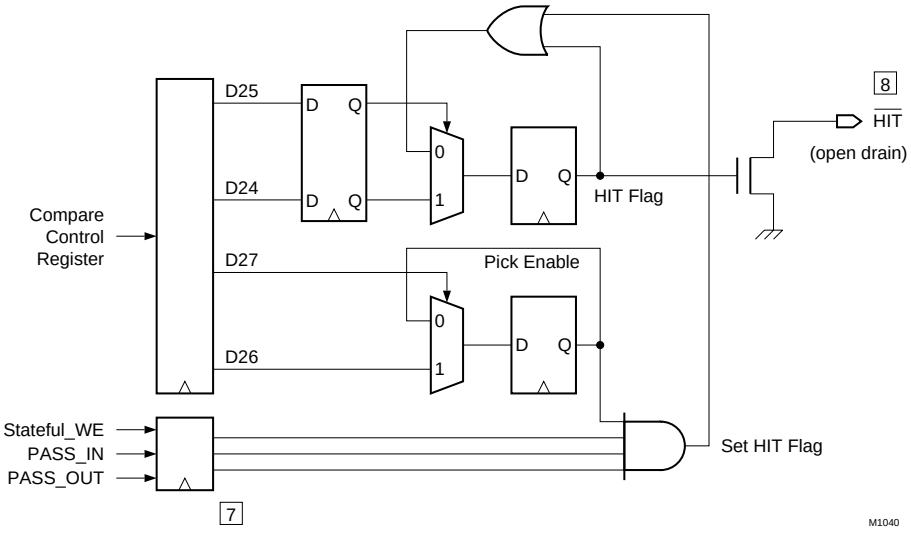


Figure 1.9 Block diagram of the Picking Logic

2 Pin Descriptions and Pinouts

Pin Descriptions and Pinouts

This chapter describes the 3D-RAM pins. Unless otherwise specified, all signals comply with the Low Voltage TTL (LVTTL) standard. The functional block diagram in Figure 2.1 shows all I/O signals on the external pins. The master clock MCLK synchronizes all operations of the Pixel ALU Control and DRAM Covntrol. The Video Control specifies the video interface. The Test Access Port is used for the JTAG (Joint Test Action Group) boundary scan. The following sections describe each signal in detail.

Common Pins

These signals are common to several sections of the 3D-RAM.

Table 2.1 Common control signals

Signal Name	Pin Count	I/O	Description
MCLK	1	I	Master clock
RESET	1	I	Reset
Total	2		

MCLK

The master clock MCLK is used for timing synchronization of internal circuitry. All external timing parameters, except video output operation and boundary scan, are specified with respect to the MCLK rising edge.

RESET

The $\overline{\text{RESET}}$ is an active low asynchronous signal used for power up and restart initialization. During power-up, the $\overline{\text{RESET}}$ signal should be held low for at least 500µs after stable V_{DD}, so that the internal power supply can be stabilized. After the $\overline{\text{RESET}}$ signal goes high, nine idle cycles must elapse before the internal registers can be reset to default values. The power-up reset procedure is illustrated in Figure 8.1. When the $\overline{\text{RESET}}$ signal is asserted low during normal operations, a restart reset sequence begins. The restart reset includes resetting registers in nine idle cycles and initializing DRAM array as in the power-up reset. The restart reset sequence is shown in Figure 8.3. In DRAM array initialization, the Access Page (ACP) operation should be performed on one page for every DRAM bank, followed by the Precharge Bank (PRE) operation for every bank. Figure 8.3 shows two approaches to initializing the DRAM array.

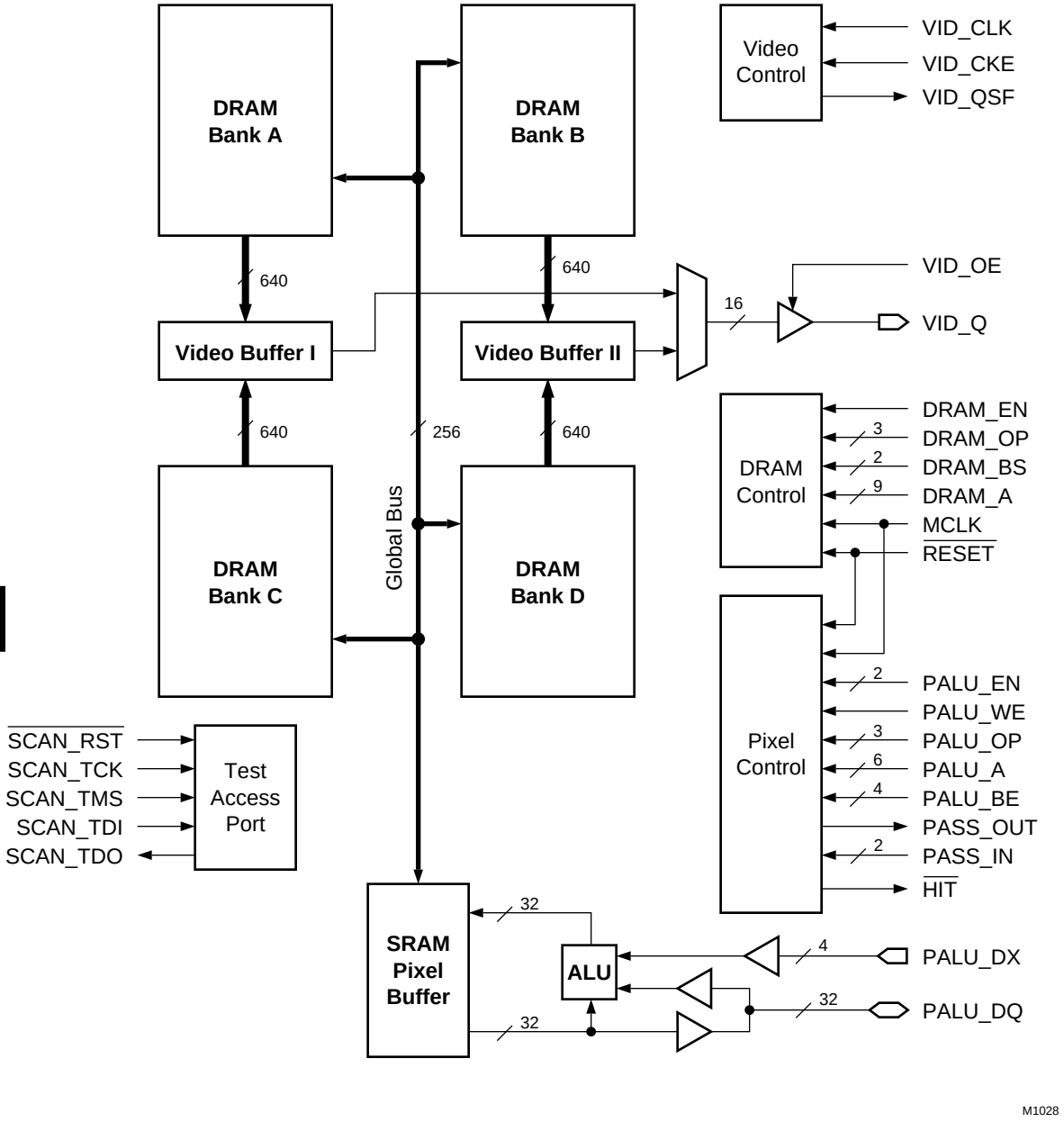


Figure 2.1 3D-RAM functional block diagram with external pins

Pixel ALU Interface

These signals control the Pixel ALU and Pixel Buffer.

Table 2.2 Pixel ALU control signals

Signal Name	Pin Count	I/O	Description
PALU_EN	2	I	Enable Pixel ALU operation starting next cycle
PALU_WE	1	I	Pixel ALU write enable
PALU_OP	3	I	Pixel ALU opcode
PALU_A	6	I	Read/Write address
PALU_BE	4	I	Byte write or output enable
PALU_DQ	32	I/O	Data pins
PALU_DX	4	I	Data extension pins for blending
PASS_OUT	1	O	Compare output (special signal level, see Table 7.2)
PASS_IN	2	I	Compare input (special signal level, see Table 7.2)
$\overline{\text{HIT}}$	1	O	Picking Logic flag output (open-drain, see Table 7.2)
Total	55		

PALU_EN$_{[1:0]}$

The PALU_EN$_{[1:0]}$ pins must be “11” to start a Pixel ALU operation. If either PALU_EN pin is “0”, then all other Pixel ALU pins are ignored.

PALU_WE

The PALU_WE indicates a write operation when high (“1”) and a read operation when low (“0”).

PALU_OP$_{[2:0]}$

The PALU_OP$_{[2:0]}$ pins, together with PALU_WE, specify the operation to be performed. See Table 3.4 for the Pixel ALU operation encoding.

PALU_A$_{[5:0]}$

The PALU_A$_{[5:0]}$ pins provide an address for the specified operation.

PALU_BE$_{[3:0]}$

The PALU_BE$_{[3:0]}$ pins apply to all read and write operations, including register writes and Dirty Tag writes. If PALU_WE is low “0”, indicating a read, the PALU_BE pins are per byte output enables. If PALU_WE is high “1”, indicating a write, the PALU_BE pins are per byte write enables. PALU_BE0 controls PALU_DQ$_{[7:0]}$; PALU_BE1 controls PALU_DQ$_{[15:8]}$; PALU_BE2 controls PALU_DQ$_{[23:16]}$; and PALU_BE3 controls PALU_DQ$_{[31:24]}$.

PALU_DQ$_{[31:0]}$

Data is read from or written to the PALU_DQ[31:0] pins. The write address of Pixel Buffer may be input from PALU_DQ$_{[29:24]}$ in some modes of operation. See “ An Application of the Write Address Control Register” on page 38.

PALU_DX$_{[3:0]}$

Extra high-order bits of PALU_DQ data are provided by PALU_DX$_{[3:0]}$. PALU_DX0 is associated with PALU_DQ$_{[7:0]}$; PALU_DX1 is associated with PALU_DQ$_{[15:8]}$; PALU_DX2 is for PALU_DQ$_{[23:16]}$; and PALU_DX3 is for PALU_DQ$_{[31:24]}$.

PASS_OUT

The comparison result of the Dual Compare unit is output on the PASS_OUT pin. PASS_OUT is low (“0”) only when the Pixel ALU operation during the fifth stage of Pixel ALU pipeline is a Stateful Initial/Normal Data Write operation (see “Pixel Data Operations” on page 40) and when either match comparison or magnitude comparison fails. Otherwise, PASS_OUT is high (“1”), indicating either the Pixel ALU operation is not a Stateful Initial/Normal Data Write or both comparison tests passed during the Stateful Initial/Normal Data Write.

PASS_IN$_{[1:0]}$

When the PASS_IN$_{[1:0]}$ pins are high (“11”) and the internal comparison test also passes (PASS_OUT is high (“1”)), data is written to the Pixel Buffer if the Pixel ALU operation is a Stateful Normal/Initial Data Write. Each of the PASS_IN[1:0] pins may be individually masked by the PASS_INs Select register bits 0 and 8, PINS[0, 8], respectively.

$\overline{\text{HIT}}$

The HIT pin is an open-drain, active low output. This pin reflects the internal status of the HIT flag. See “Compare Control Register (CCR $_{[31:0]}$)” on page 36 for a detailed description.

DRAM Control

These signals command operations on the four DRAM banks, Global Bus and Video Buffer.

Table 2.3 DRAM control signals

Signal Name	Pin Count	I/O	Description
DRAM_EN	1	I	Enable DRAM operation at next cycle
DRAM_OP	3	I	DRAM opcode
DRAM_BS	2	I	DRAM bank select
DRAM_A	9	I	Address for page, block, and video line
Total	15		

DRAM_EN

When DRAM_EN is high (“1”) at the rising edge of MCLK, a DRAM operation is initiated at the next clock cycle. Only the selected DRAM bank is enabled.

DRAM_OP$_{[2:0]}$

The DRAM Opcode DRAM_OP$_{[2:0]}$ specifies the DRAM operation. See Table 4.1 for the DRAM operation encoding.

DRAM_BS$_{[1:0]}$

DRAM_BS$_{[1:0]}$ is used to select one out of four banks. The selection codes are: “00” for Bank A, “01” for Bank B, “10” for Bank C, and “11” for Bank D.

DRAM_A$_{[8:0]}$

The address pins DRAM_A$_{[8:0]}$ are used to select one of the following: (i) a page in a DRAM bank, (ii) a block of data to be transferred between the sense amplifiers of a DRAM bank and the Pixel Buffer over the Global Bus, or (iii) 80 bytes of video data from the sense amplifiers of a DRAM page to a Video Buffer. Details are described in Chapter 4, “DRAM Operations.”

Video Interface

These signals interface with a video RAMDAC chip.

Table 2.4 Video signals

Signal Name	Pin Count	I/O	Description
VID_CLK	1	I	Video clock
VID_CKE	1	I	Video clock enable
VID_OE	1	I	Video output enable
VID_Q	16	O	Video data bus
VID_QSF	1	O	Video buffer indicator
Total	20		

VID_CLK

VID_CLK is a free running or gated video shift clock.

VID_CKE

VID_CKE is a synchronous VID_CLK enable signal. When VID_CKE is high (“1”), the next VID_CLK cycle will be enabled. The video counter will also be enabled in the next cycle.

VID_OE

VID_OE is an asynchronous video output enable for VID_Q. The video data bus is enabled when VID_OE is high (“1”).

VID_Q$_{[15:0]}$

With 16-bit video data bus VID_Q$_{[15:0]}$, two bytes of data can be clocked out on the same cycle. In the 8-bit Video Buffer, the output format is arranged as even bytes on VID_Q$_{[7:0]}$ and odd bytes on VID_Q$_{[15:8]}$. A detailed description of the two output data formats, normal mode and reversed mode, is in “Video Output Operation” on page 48.

VID_QSF

The VID_QSF output indicates which video buffer is currently providing video data. VID_QSF is low (“0”) when Video Buffer I is shifting data out. VID_QSF is high (“1”) when Video Buffer II is shifting data out.

Test Access Port

These signals interface to the Test Access Port for partial compliance with the IEEE Standard 1149.1 Test Access Port and Boundary Scan—Scan Architecture. The three input pins SCAN_RST, SCAN_TMS, and SCAN_TDI should be pulled up to VDD with a 5K resistor. See Chapter 10, “JTAG Boundary Scan,” for more details.

Table 2.5 Serial test signals

Signal Name	Pin Count	I/O	Description
$\overline{\text{SCAN\_RST}}$	1	I	Scan reset
SCAN_TCK	1	I	Scan clock
SCAN_TMS	1	I	Scan test mode select
SCAN_TDI	1	I	Scan test data input
SCAN_TDO	1	O	Scan test data output
Total	5		

Power & Ground

There are 13 Power Supply pins and 17 Ground pins. The NC pin should not be connected.

Table 2.6 Power and Ground

Signal Name	Pin Count	Description
V_{SS}	17	Ground
V_{DD}	13	Power supply
NC	1	No connection
Total	31	

3D-RAM Pinouts

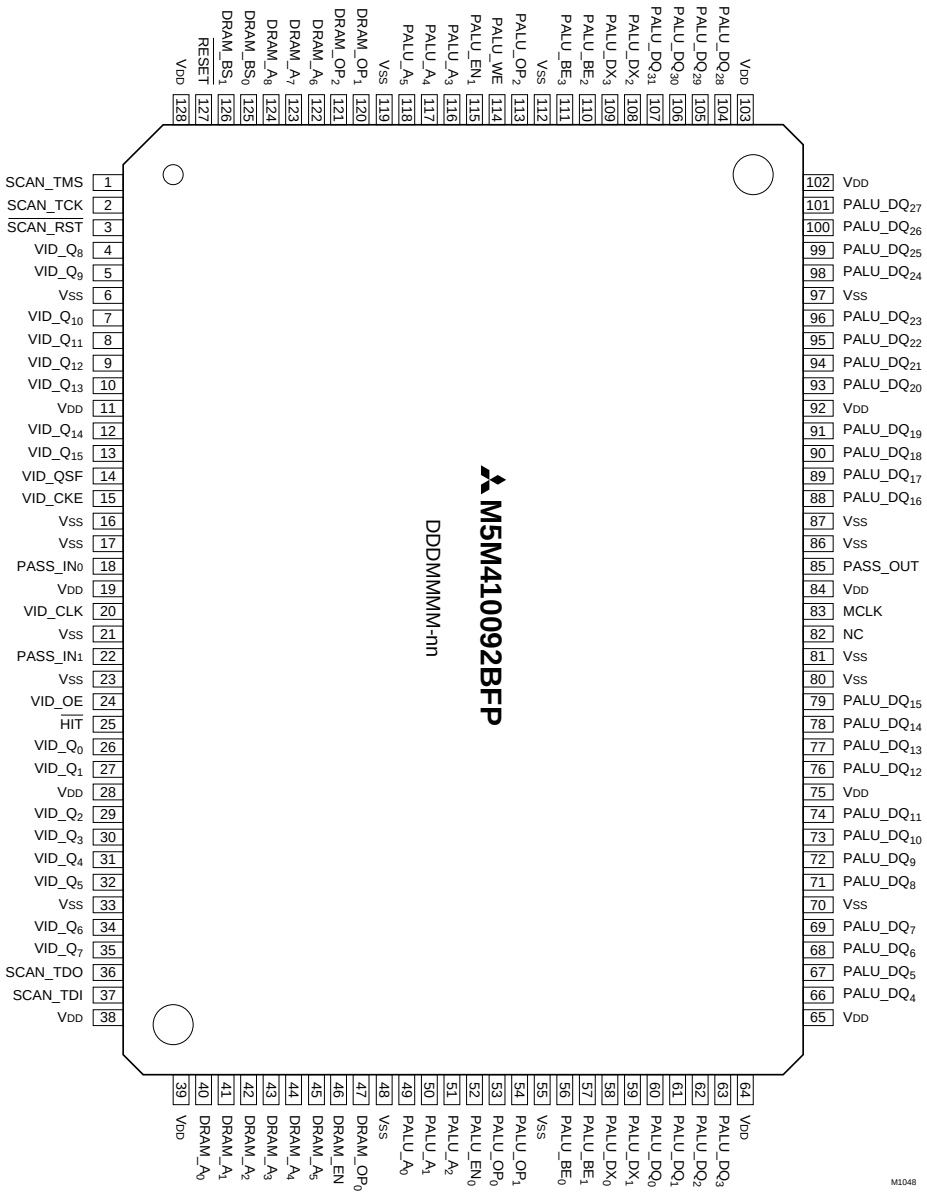
There are two pinouts for 3D-RAM: normal pinout with pin 1 located at the lower left hand corner and specially marked by a small circle; and reverse pinout with pin 1 located at the upper left hand corner and marked by a large circle and a pointing triangle. The device in normal pinout is designated by the letters “FP” in the product number, and the device in reverse pinout by the letters “RF.” In both pinouts, the mapping of pin number with pin name is identical.

Tracking Label

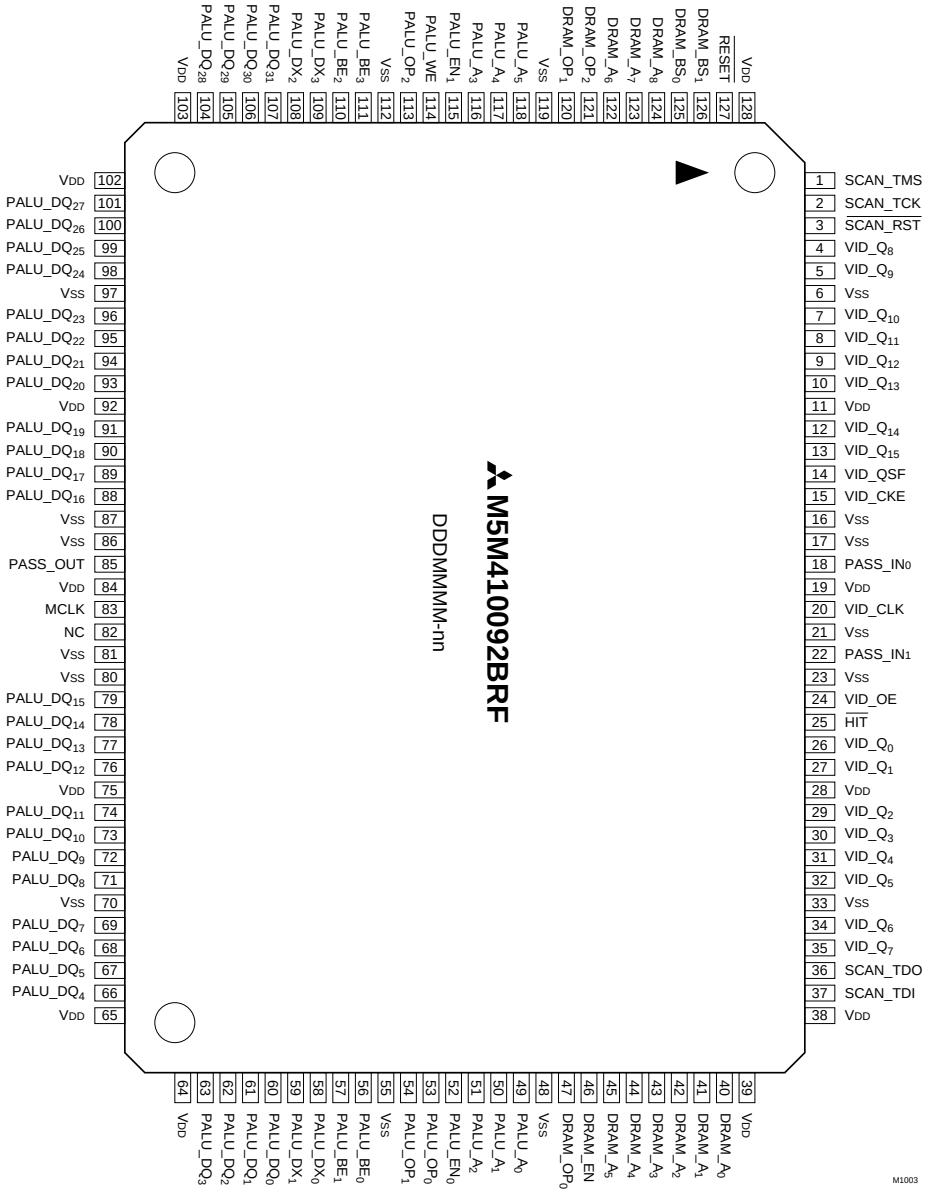
On the top surface of the 3D-RAM package, a tracking label is printed below the Mitsubishi logo and the 3D-RAM product number. The tracking label consists of 7 numbers followed by a dash and a speed/power grade designation and is represented by the mnemonic “DDDMMMM-nn”. This mnemonic is explained below.

DDD: Data code
MMMM: Manufacturing code
nn: One of the following speed/power grade designations:
“10” — t_{CKL} (min) = 10 ns
“13” — t_{CKL} (min) = 13 ns

Normal Pinout Diagram



Reverse Pinout Diagram



3 Pixel ALU Operations

Pixel ALU Operations

This chapter discusses details on the elements and operations of the Pixel Buffer and Pixel ALU in 3D-RAM. An operation that involves only the Pixel ALU and the Pixel Buffer is called a Pixel ALU operation. An operation that involves a DRAM array is categorized as a DRAM operation and is described in Chapter 4, “DRAM Operations.” All registers of the 3D-RAM are defined and explained in this chapter.

Elements of the Pixel Buffer

Block and Word

As stated in Chapter 2, the 2,048-bit Pixel Buffer is organized into eight 256-bit blocks. During a DRAM operation, these blocks can be addressed from the DRAM_A pins for block transfers on the Global Bus. During a Pixel ALU operation, the 32-bit Pixel ALU accesses the Pixel Buffer, requiring not only the block address be specified but also the 32-bit word be identified. This is done via the 6-bit PALU_A pins. The upper three bits select one of eight blocks in the Pixel Buffer, and the lower three bits specifies one of the eight words in the selected block. The availability of both the DRAM_A and PALU_A pins allows concurrent DRAM and Pixel ALU operations. Since a word is mapped directly to $PALU\_DQ_{[31:0]}$, $PALU\_DQ_{[7:0]}$ is byte 0, $PALU\_DQ_{[15:8]}$ is byte 1, $PALU\_DQ_{[23:16]}$ is byte 2, and $PALU\_DQ_{[31:24]}$ is byte 3. Figure 3.1 is a simplified block diagram of these Pixel Buffer elements.

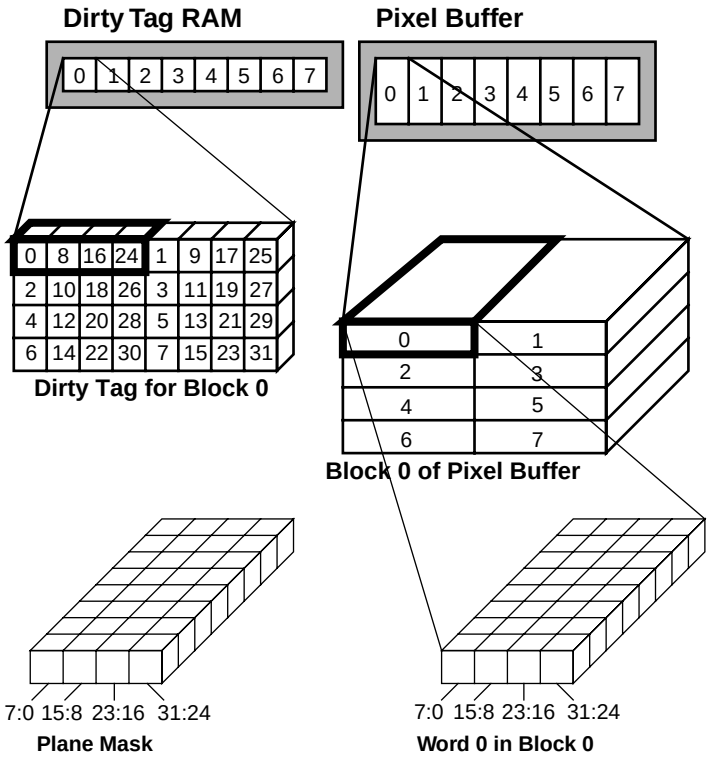


Figure 3.1 Pixel Buffer elements

Dirty Tag

Each data byte of a 256-bit block is associated with a Dirty Tag bit. This means that each ˇ-byte word is associated with four Dirty Tag bits and that a 32-bit Dirty Tag memory controls the corresponding 32-byte block data. The Dirty Tag RAM in the Pixel Buffer contains eight such 32-bit Dirty Tags. There are three aspects of Dirty Tag operations: tag clear, tag set, and tag initialization. In normal operation modes, the clearing and setting of the Dirty Tag by these read and write operations are done by the on-chip logic in the 3D-RAM and are essentially transparent to the rendering controller. The Dirty Tag bits are used by the 3D-RAM internally and are not output to the external pins.

When data is transferred from the sense amplifiers of a DRAM bank to a Pixel Buffer block over the Global Bus (i.e., a Read Block transfer which is a DRAM operation and is described in the next chapter), all 32 Dirty Tag bits associated with the selected Pixel Buffer block are cleared to “0”.

When data is transferred from a Pixel Buffer block to the sense amplifiers of a DRAM bank (i.e., a Write Block transfer, another DRAM operation), the Dirty Tag determines which data bytes can be written into the sense amplifiers. When a Dirty Tag bit is “1”, the corresponding data byte is written under the control of the Plane Mask register (see the following section). When a Dirty Tag bit is “0”, the corresponding byte of data in the DRAM bank is not written and retains its former value. Because the Dirty Tag prevents the unaltered bytes of a 256-bit block from being written into a DRAM bank, the power consumption of a Write Block transfer may be reduced by as much as 50%. This may be a significant power saving when a high-resolution display is constantly redrawn, such as in the case of high-quality full-screen animation.

When a data word is read from the 32-bit ALU port of Pixel Buffer, none of the 32-bit Dirty Tags is affected or has any effect on the out-going data. The setting and initialization of the Dirty Tags are described in the paragraphs below.

Table 3.1 Pixel ALU operations involving Dirty Tags

Pixel Operation	Pixel Data	New Dirty Tag Contents
(Stateful/ Stateless)Normal Data Write	Write bytes 0 to 3 from PALU_DQ pins (per PALU_BE pins)	The four addressed Dirty Tag bits are ORed with $PALU\_BE_{[3:0]}$; the other 28 Dirty Tag bits are unchanged.
(Stateful/ Stateless)Initial Data Write	Write bytes 0 to 3 from PALU_DQ pins (per PALU_BE pins)	$PALU\_BE_{[3:0]}$ is written to the 4 addressed Dirty Tag bits; “0” is written to the 28 unaddressed Dirty Tag bits.
Replace Dirty Tag	Unchanged	$PALU\_DQ_{[31:0]}$ replaces 32 Dirty Tag bits.
OR Dirty Tag	Unchanged	All 32 Dirty Tag bits are ORed with $PALU\_DQ_{[31:0]}$.

The Dirty Tag bits play an important role for all four write operations of the Pixel ALU to the Pixel Buffer: Stateful/Stateless Initial Data Write and Stateful/Stateless Normal Data Write. (These operations are also explained in "Pixel ALU Operations" on page 57.) Since the Pixel ALU operations conform to the 7-stage pipeline, the byte enable $PALU\_BE_{[3:0]}$ data also gets into the pipeline when the operation is issued. At the end of the pipeline, pixel data is written into a Pixel Buffer word and $PALU\_BE_{[3:0]}$ pins can change the four corresponding Dirty Tag bits. In the Initial Data Write operation, the four addressed Dirty Tag bits are replaced with $PALU\_BE_{[3:0]}$, while the other 28 Dirty Tag bits for the same block are cleared to "0". In the Normal Data Write operation, each of the four addressed Dirty Tag bits is set to "1" only when the corresponding PALU_BE pin is "1". An addressed Dirty Tag bit is unchanged if the corresponding PALU_BE pin is "0". The other 28 Dirty Tag bits for the same block are also unchanged.

The 32 Dirty Tag bits for a particular block can all be replaced with the PALU_DQ data through the Pixel ALU operation "Replace Dirty Tag." Another Pixel ALU operation "OR Dirty Tag" changes the Dirty Tag contents for an addressed block with the result of the bitwise "OR" function on the original Dirty Tag data and the $PALU\_DQ_{[31:0]}$ data. The bit mapping between the Dirty Tag and PALU_DQ pins is illustrated in Figure 3.1. For example, to change the Dirty Tag bits for word 0, the data should be placed on PALU_DQ0, PALU_DQ8, PALU_DQ16, and PALU_DQ24. To change the Dirty Tag bits for word 5, the data should be on PALU_DQ5, PALU_DQ13, PALU_DQ21, and PALU_DQ29. The following sub-section provides an application of these "Replace Dirty Tag" and "OR Dirty Tag" operations.

Using Dirty Tag for Color Expansion

Many 2D rendering operations, such as text drawing, involve writing the same color to many pixels. These operations can be greatly accelerated by specifying individual pixels with a single bit and having hardware automatically expand each bit to an entire pixel.

In 3D-RAM Color Expansion is done with the Dirty Tags associated with the Pixel Buffer blocks. The pixel color is written eight times to a Pixel Buffer block so that all of the pixels in the block are the same color. Next, a 32-bit word is written to the Dirty Tag of the associated block. Finally, the block is written to a DRAM bank. The pixel whose corresponding Dirty Tag bit is set is changed to the new color. The other pixels are unaffected.

A new 32-bit word may be written to the Dirty Tag afterwards, and the same Pixel Buffer block may be written to a different part of the DRAM array. Thus, one Pixel Buffer block can be used to hold the foreground color and used repeatedly to write text to the frame buffer.

Plane Mask

The 32-bit Plane Mask register ($PM_{[31:0]}$) is used to qualify two write functions: (1) as per-bit write enables on 32-bit data for a Stateful (Initial/ Normal) Data Write operation from the Pixel ALU to the Pixel Buffer; (2) as per-bit write enables on 256-bit data for a Masked Write Block (MWB) operation from the Pixel Buffer to the sense amplifiers of a DRAM bank over the Global Bus. For a Stateful Data Write, the Plane Mask serves as per-bit write enables over the entering data from the Pixel ALU write port; bit 0 of the Plane Mask enables or disables bit 0 of the incoming 32-bit pixel data, bit 1 of the Plane Mask enables or disables bit 1 of the incoming 32-bit pixel data, and so on. For a Masked Write Block operation on the Global Bus side, when a Pixel Buffer block is transferred out to the DRAM, the 32-bit Plane Mask applies to every 32-bit word as per-bit write enables. In other words, bit 0 of the Plane Mask enables or disables bits 0, 32, 64, 96, 128, 160, 192, and 224 of the 256-bit block; bit 1 of the Plane Mask enables or disables bits 1, 33, 65, 97, 129, 161, 193, and 225.

A particular sense amplifier bit can be written only if both the Dirty Tag bit and the Plane Mask bit are logically “1”. This kind of relationship among multiple enables and block data is illustrated in Figure 3.2 for the first 40 bits (which are Word 0 and byte 0 of Word 1) of the Global Bus.

It is important to note the simultaneous effects of the Plane Mask. Although 3D-RAM allows concurrent operations of Pixel ALU and DRAM, the user is cautioned that there is only one set of Plane Mask bits that can affect both Pixel ALU write and DRAM write operations at the same time. When different plane maskings are required, concurrent Pixel ALU Stateful Data Write operations and DRAM Masked Write Block operations must be avoided.

Once the Plane Mask is written, the new Plane Mask is effective for only the Stateful Data Write operations issued at later cycles, thereby conforming to the uniform 7-stage pipeline rule. The Plane Mask register is loaded through a Pixel ALU “Write Control Register” operation. The mapping of the Plane Mask to the PALU_DQ pins is the same as the Word data to the pins (see also the section on “Block and Word” on page 27).

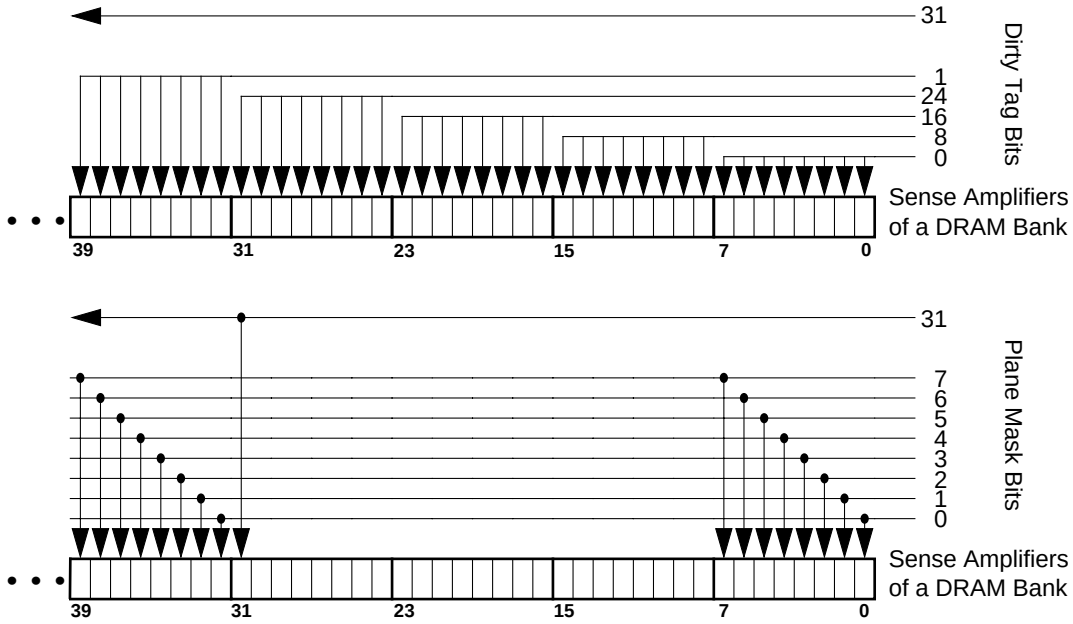


Figure 3-2 The relationship between Dirty Tags and Plane Mask for first 40 bits of the Global Bus. (Both the Dirty Tag bit and the Plane Mask bit must be 1 before a particular Sense Amp bit can be written.)

Elements and Operations of the Pixel ALU

Chapter 2 presented an overview of the Pixel ALU, with an emphasis on the motivation and applications of the elements in the Pixel ALU. In this section, some of the same information is repeated, but the emphasis is on detailed technical specification.

The elements of the Pixel ALU are four 8-bit ROP/Blend units, one 32-bit Match Compare unit, one 32-bit Magnitude Compare unit, and the Picking Logic. Figure 3.3 shows the inputs to the ROP/Blend units and the Dual Compare unit. In the figure, bus “O” is the old data from the Pixel Buffer; “N” and “NX” are from the PALU_DX and PALU_DQ pins, respectively; and finally, buses “KX” and “K” are from the internal 36-bit Constant Source register, with “KX” being the most significant four bits. The inputs to the Dual Compare unit are straightforward. The inputs to the ROP/Blend units are explained in the following sub-sections.

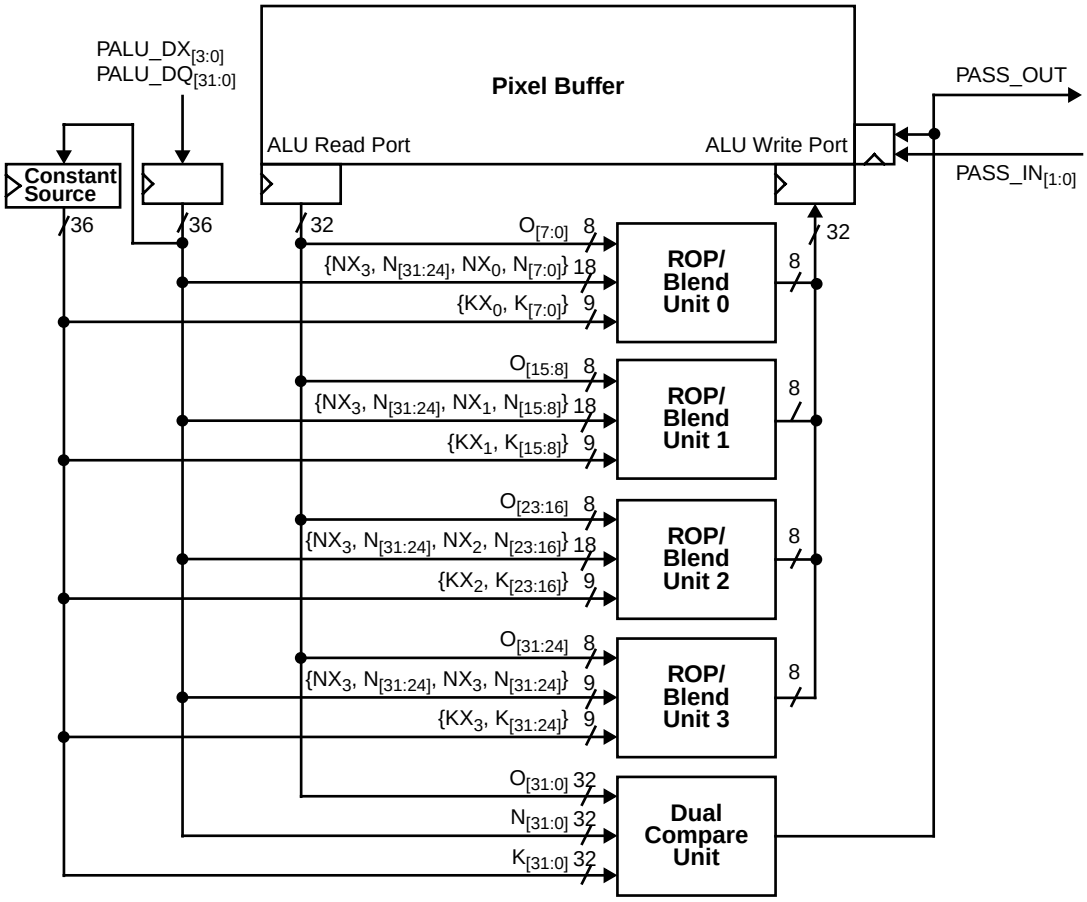


Figure 3.3 Pixel ALU (Pipeline stages are not shown)

ROP/Blend Units

Each ROP/Blend unit can be independently configured as either a ROP unit or a Blend unit through the programming of the ROP/Blend Control register. Each ROP unit can perform all 16 standard ROP functions, which are listed in Table 3.16. ROP functions are performed on a byte of the Old Data (“O”) from the Pixel Buffer and a byte of the New Term, which is either the data from the data pins (“N”) or the data from the Constant Source register (“K”).

Pixel ALU Blend Modes **(NEW)**

For the blending operation, the general equation is as follows:

Write data to Pixel Buffer
= New Term + (Old Data x Old Fraction)
= (New Data x New Fraction) + (Old Data x Old Fraction)

To each Blend unit, an Addend (e.g. the “New Term” or 00h) is input from the PALU_DX and PALU_DQ pins (marked as {“NX”, “N”}), from the Pixel Buffer (marked as “O”), or from the Constant Source register (marked as {“KX”, “K”}). Multiplicand 1 (marked as “MULTP1”) is the fraction term and is from one of five sources; Multiplicand 2 (marked as “MULTP2”) is the data term and is from one of six sources. See Table 3.5 for a complete selection mapping of the Addend and Multiplicands.

In OpenGL terminology (see “The OpenGL Graphics System: A Specification (Version 1.1)”), New Data represents the color values of the source (SRC_Color) which enter the Pixel ALU path from the PALU_DQ pins; Old Data represents the color values of the destination (DST_Color) which are from the Pixel Buffer. New Fraction is known as the source blend factor (sfactor); Old Fraction is known as the destination blend factor (dfactor). The color values, SRC_Color and DST_Color, can be represented in RGBA quadruplets form as (Rs, Gs, Bs, As) and (Rd, Gd, Bd, Ad), respectively. Define sfactor and dfactor as (Sr, Sg, Sb, Sa) and (Dr, Dg, Db, Da), respectively. The blending equation can be rewritten as:

Write data to Pixel Buffer
= (SRC_Color x sfactor)
+ (DST_Color x dfactor)
= (Rs, Gs, Bs, As) x (Sr, Sg, Sb, Sa)
+ (Rd, Gd, Bd, Ad) x (Dr, Dg, Db, Da)
= (RsxSr+RdxDr, GsxSg+GdxDg,
BsxSb+BdxDb, AsxSa+AdxDa)

All the possible values for OpenGL blending factors are listed in Table 3.2. The subtraction of quadruplets means subtracting them componentwise. The column “Relevant Factor” indicates whether the corresponding constant can be used to specify the source or destination blend factor.

The full blending function requires two multiplications and one addition for each of the four components in the quadruplet. Enumreating the eight possible values of destination blend factor and nine possible values of source blend factor, we arrive at the 72 blending modes illustrated in the matrix in Table 3.3. Not all of the combinations make sense. The majority of applications use a small number of combinations. Most of the blending modes with (0,0,0,0) or (1,1,1,1) as the blending factor can be realized with a half blender, meaning that they only require one multiplication and the addition. In addition, if one of the multiplications does not require destination colors or destination alpha from the frame buffer, this multiplication can be performed inside the rendering controller without having to read the destination data out of the frame buffer. Thus, only a half blender is needed inside the 3D-RAM to complete the blending equation in these cases. For the rest of the blending modes, the blending functions can be completed in two consecutive cycles using the 3D-RAM’s Two-Cycle Blend operation by looping back the product term from the first cycle and combining it with the product term from the second cycle.

Table 3.2 Source and Destination Blending Factors

Constant	Relevant Factor	Computed Blend Factor
GL_ZERO	source or destination	(0,0,0,0)
GL_ONE	source or destination	(1,1,1,1)
GL_DST_COLOR	source	(Rd,Gd,Bd,Ad)
GL_SRC_COLOR	destination	(Rs,Gs,Bs,As)
GL_ONE_MINUS_DST_COLOR	source	(1,1,1,1) – (Rd,Gd,Bd,Ad)
GL_ONE_MINUS_SRC_COLOR	destination	(1,1,1,1) – (Rs,Gs,Bs,As)
GL_SRC_ALPHA	source or destination	(As,As,As,As)
GL_ONE_MINUS_SRC_ALPHA	source or destination	(1,1,1,1) – (As,As,As,As)
GL_DST_ALPHA	source or destination	(Ad,Ad,Ad,Ad)
GL_ONE_MINUS_DST_ALPHA	source or destination	(1,1,1,1) – (Ad,Ad,Ad,Ad)
GL_SRC_ALPHA_SATURATE	source	(f,f,f,1); f=min(As, 1–Ad)

Table 3.3 OpenGL blending modes

	Destination Blend Factor							
Source Blend Factor	**ZERO**	**ONE**	**SRC_ COLOR**	**ONE_ MINUS_ SRC_ COLOR**	**SRC_ ALPHA**	**ONE_ MINUS_ SRC_ ALPHA**	**DST_ ALPHA**	**ONE_ MINUS_ DST_ ALPHA**
ZERO	x	x	x	x	x	x	x	x
ONE	x	x	x	o	x	c, o	x	x
DST_COLOR	x	x	o	o	o	o	o	o
ONE_MINUS_ DST_COLOR	x	x	o	o	o	o	o	o
SRC_ALPHA	x	x	o	o	c, o	c, o	x	x
ONE_MINUS_ SRC_ALPHA	x	x	o	o	c, o	c, o	x	x
DST_ALPHA	x	x	o	o	o	o	o	o
ONE_MINUS_ DST_ALPHA	x	x	o	o	o	o	o	o
SRC_ALPHA_ SATURATE	x	x	o	o	o	o	o	o

Legend: “x” = half blending in a single clock cycle; “o” = full blending in two cycles using the Two-Cycle Blend operation; “c” = one cycle blending with alpha blending ignored

3D-RAM accelerates 32 blending modes in single clock cycle throughput by half blending and the other 40 blendng modes in two cycles. In addition, there are five cases, two-cycle blending modes may be accelerated in just one cycle if the alpha blending can be ignored.

Blending Mode Operation (NEW)

The simplified block diagram of the Blending unit is illustrated in Figure 3.4. To execute a single-cycle blending operation, the multiplicands and addend (MULTP1, MULTP2, Addend) must be selected by programming the ROP/Blend and Blend_2 Control registers. Also, the ROP/Blend Control register must be set for blending. Once these registers are set, the blending operation is accomplished by performing a Stateful Write operation. Each Blend unit first performs the multiplication of the data term (MULTP2) and fraction term (MULTP1) and then the addition of the resulting product with the Addend, thereby completing a half blend.

To execute a Two-Cycle Blend operation, it is necessary to program three registers. The ROP/Blend and Blend_2 Control registers are programmed just as they would be for a single-cycle blending operation. The selected MULTP1 and MULTP2 components will apply to the second cycle of the operation. The Addend selected by these two registers is ignored for the Two-Cycle Blend operation. The Preblend Control register selects MULTP2 for the Preblend Cycle (the first cycle during the Two-Cycle Blend operation) and the Addend for the Normal Cycle. MULTP1 and Addend are fixed to the PALU_DQ bus for the Preblend Cycle. Once these three registers have been programmed, an “Initiate Two-Cycle Blending” operation (PALU_OP=110, PALU_WE=1) should be performed and followed by a Stateful Initial/Normal Write operation on the same pixel location (i.e. PALU_A with the same Block:Word address and $PALU\_BE_{[3:0]}$ with the same enble settings). During the Preblend Cycle, MULTP1 and MULTP2 are multiplied and the result is “looped back” one stage in the pipeline. The Addend is also “looped back” one stage. The Addend and the multiplier output are then available as possible Addends for the next cycle. Next, the Stateful Write is issued with the multiplicands selected by the ROP/Blend and Blend_2 Control registers. The Addend selected by the ROP/Blend and Blend_2 registers will be ignored. The blending occurs just as it would for a single-cycle operation except that the Addend source is chosen to be either the previous multiplier output or the previous Addend, based on the settings of the Preblend Control register.

Figure 3.4 illustrates the above description with a simplified block diagram. The blocks labelled “N:NN”, “N:N0”, and “NX:N” on the blending path represent the manner in which the 8-bit Blend units duplicate 4-bit data for the special (4,4,4,4) 16-bit color mode. Specifically, “N:NN” means that the 4-bit data is nibble-wise duplicated to form an 8-bit data; “N:N0” means an 8-bit data is formed by padding the lower nibble with 0000b; and “NX:N” means a 4-bit data is produced by truncating the lower nibble, regardless of its value. More explanations may be found in the section on “4-bit to 8-bit Expansion for Pixel ALU.”

Note that the special OpenGL stencil mode, which will be described in the section on “Stencil Modes,” uses portions of ROP/Blend unit 3 to accomplish its functions. For simplicity, the stencil logic is not shown in Figure 3.4 and, for the most part, can be thought of as a separate unit. It is important to note, however, that the stencil logic uses portions of the blending path and therefore, ROP/Blend unit 3 cannot be used for blending when the OpenGL stencil mode is being used.

ROP/Blend units 0, 1 and 2 are identical, but unit 3 is slightly different because this unit typically handles the Alpha data. The Alpha-Saturate block shown in Figure 3.4 and Figure 3.5 is only present in ROP/Blend unit 3. The result from the Alpha-Saturate block is routed to all four ROP/Blend units as a possible source of MULTP2.

For the specifics of the data multiplexing and selections by the various register bits, refer to Figure 3.6. The timing diagram of an example Two-Cycle Blend operation is presented in Figure 3.7.

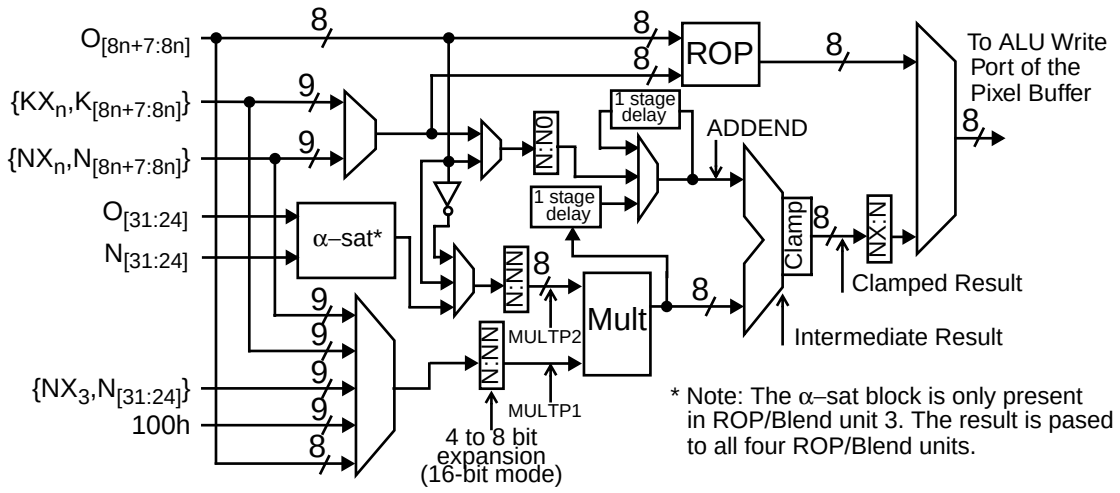


Figure 3.4 ROP/Blend unit n (Pipeline stages ar not shown)

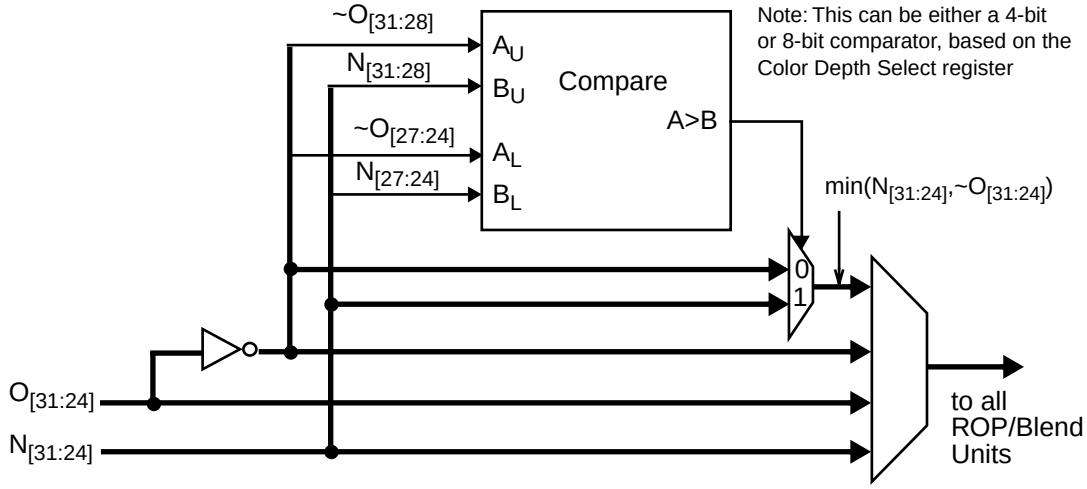


Figure 3.5 Block diagram of the Alpha-Saturate unit

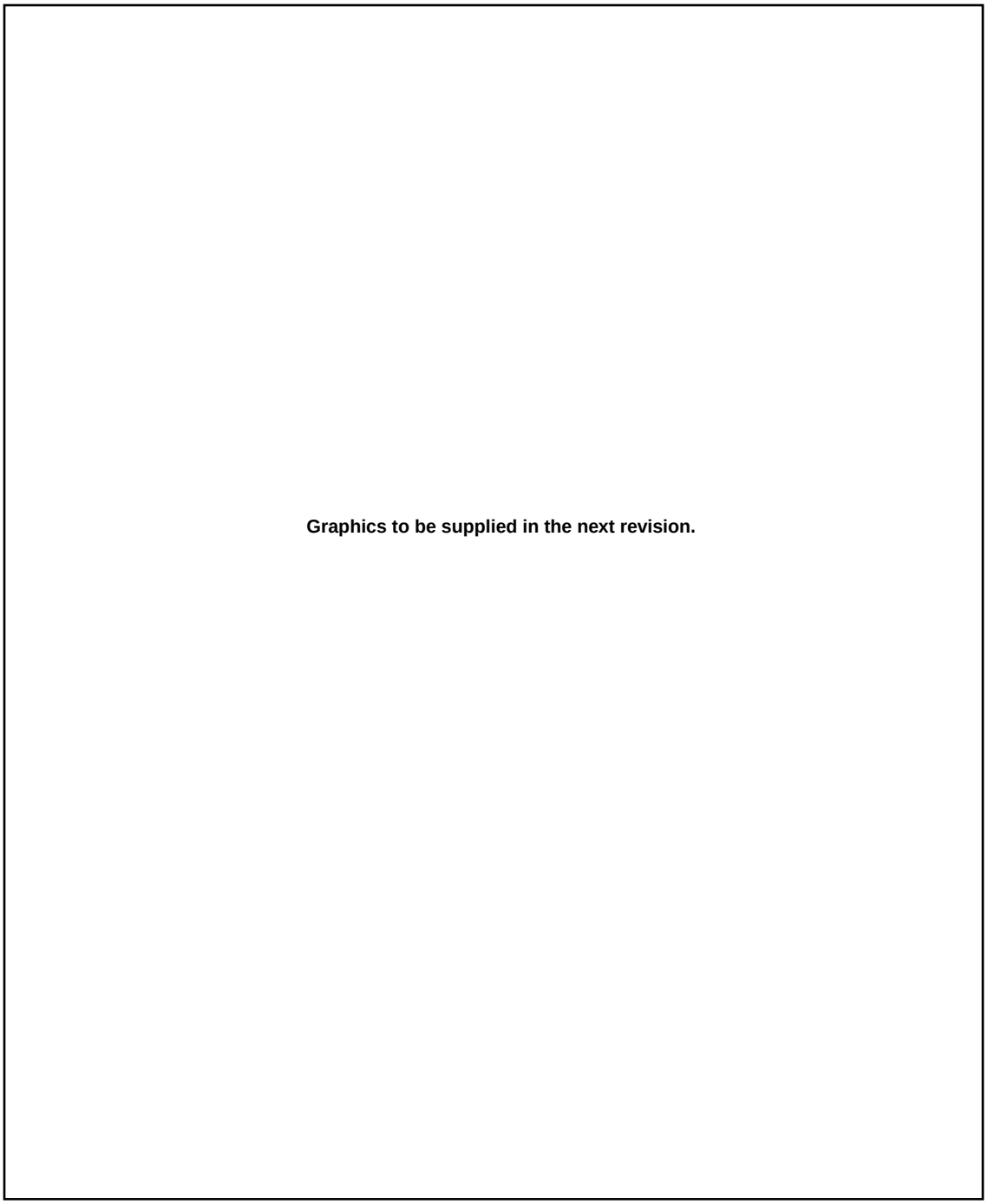


Figure 3.6 Details of data selections in Blend unit n by the various register bits

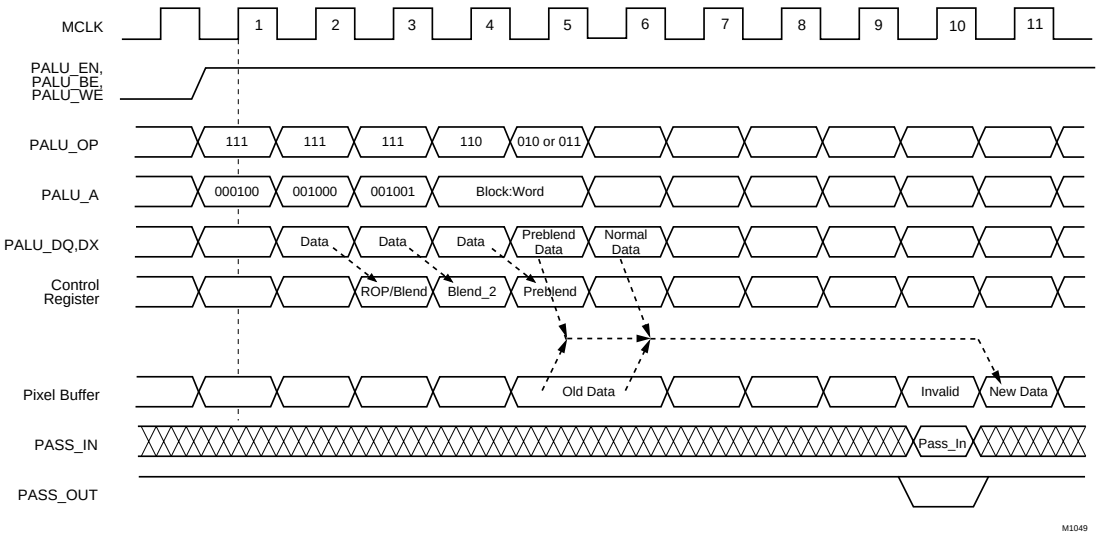


Figure 3.7 An Example of a Two-Cycle Blend operation

The mathematic operations performed in the Blend unit are summarized in Table 3.4. The Clamped Result is written to the Pixel Buffer, depending (1) on the PASS_OUT pin, which is the result of internal Compare units, and (2) on the $PASS\_IN_{[1:0]}$ pins, which is the PASS_OUT signal from the preceding 3D-RAM.

Table 3.4 Mathematical operations in Blend unit n

Operand	Range	Sources	Comments
Multiplicand 1 (Fraction term)	0.00h ~ 0.FFh (8-bit unsigned)	$\{NX_n, N_{[8n+7:8n]}\}$	Source is from $PALU\_DX_n$ and $PALU\_DQ_{[8n+7:8n]}$ pins
	0.00h ~ 0.FFh (8-bit unsigned)	$\{NX_3, N_{[31:24]}\}$	Source is from $PALU\_DX_3$ and $PALU\_DQ_{[31:24]}$ pins
	0.00h ~ 0.FFh (8-bit unsigned)	$O_{[8n+7:8n]}$	Source is from the SRAM Pixel Buffer
	0.00h ~ 0.FFh (8-bit unsigned)	$\{KX_n, K_{[8n+7:8n]}\}$	Source is from the internal Constant Register
	1.00h (9-bit constant 1.00h)	1.00h	Fractions greater than 1.00h are clamped to 1.00h
Multiplicand 2 (Data term)	0 ~ 255 (8-bit unsigned)	$O_{[8n+7:8n]}$	Source is from the SRAM Pixel Buffer
		$\sim O_{[8n+7:8n]}$	Source is inverted from $O_{[8n+7:8n]}$
		$\min\{N_{[31:24]}, \sim O_{[31:24]}\}$ (α-sat)	Source is from the α-saturate block in ROP/Blend Unit 3
		$N_{[31:24]}$	
		$O_{[31:24]}$	
		$\sim O_{[31:24]}$	
Addend	–256 ~ 255 (9-bit signed)	$\{NX_n, N_{[8n+7:8n]}\}$	Source is from $PALU\_DX_n$ and $PALU\_DQ_{[8n+7:8n]}$ pins
	–256 ~ 255 (9-bit signed)	$\{KX_n, K_{[8n+7:8n]}\}$	Source is from the internal Constant Register
	0 ~ 255 (8-bit unsigned)	$O_{[8n+7:8n]}$	Source is from the SRAM Pixel Buffer
	0 ~ 255 (8-bit unsigned)	Previous Addend ($N_{[8n+7:8n]}$)	Source is from the previous stage Addend (Loop Back Blending)
	0 ~ 255 (8-bit unsigned)	MULTP1 x MULTP2	Source is from the previous stage multiplier output (Loop Back Blending)
Intermediate Result	–256 ~ 510 (10-bit signed)	(MULTP1 x MULTP2) + Addend	
Clamped Result	0 ~ 255 (8-bit unsigned)	Intermediate Result	The Clamped Result is written to the Pixel Buffer if the pass condition is valid If source > 255, result = 255 If source < 0, result = 0

The “Alpha” value, denoted as As and Ad, should be placed at the most significant byte of the respective bus, i.e. As at $N_{[31:24]}$, which is from $PALU\_DQ_{[31:24]}$; and Ad at the $O_{[31:24]}$, which is from the Pixel Buffer. Table 3.5 lists possible multiplicand/addend selections for each OpenGL blending mode. Note that the “Preblend Cycle” column only applies to two-cycle blending operations. Each table entry represents data for all four Blending units, and some entries contain two terms. The first term applies to blending units that are designated for blending color data (in Blend units 0, 1, and 2). The second term is for the Blend unit operating on the alpha value (unit 3). The individual color terms have been grouped together in the table for simplicity. For example, the term “Cd, Ad” represents “Rd, Gd, Bd, Ad” and “1–Cd, 1–Ad” represents “1–Rd, 1–Gd, 1–Bd, 1–Ad.” Note that terms such as “1–Cd” and “1–Ad” that are generated inside the 3D-RAM are approximated by the 1’s complement. For example, the term “1–Ad” is actually ~Ad, the bitwise inverse of Ad. Note also that in alpha_saturate blending, the multiplicand selections for color and alpha are different. There are certainly more ways to do the blending operations than those listed in Table 3.5. This list demonstrates that the 3D-RAM does support all OpenGL blending modes.

Table 3.5 Multiplicand/Addend selection for each OpenGL blending mode

Blending Fractions		Preblend Cycle			Normal Cycle		
sfactor	**dfactor**	**MULTP1**	**MULTP2**	**ADDEND**	**MULTP1**	**MULTP2**	**ADDEND**
0, 0	0, 0	na	na	na	0,0 (from K)	Cd, Ad	0,0 (from DQ)
1, 1	0, 0	na	na	na	0,0 (from K)	Cd, Ad	Cs, As
Cd, Ad	0, 0	na	na	na	Cs, As	Cd, Ad	0,0 (from K)
1–Cd, 1–Ad	0, 0	na	na	na	Cs, As	1–Cd, 1–Ad	0,0 (from K)
As, As	0, 0	na	na	na	0,0 (from K)	Cd, Ad	Cs*As, As*As
		na	na	na	Cs, As	As, As	0,0 (from K)
1–As, 1–As	0, 0	na	na	na	0,0 (from K)	Cd, Ad	Cs*(1–As), As*(1–As)
Ad, Ad	0, 0	na	na	na	Cs, As	Ad, Ad	0,0 (from K)
1–Ad, 1–Ad	0, 0	na	na	na	Cs, As	1–Ad, 1–Ad	0,0 (from K)
f, 1	0, 0	na	na	na	Cs, 1	f, As	0,0 (from K)
0, 0	1, 1	na	na	na	1, 1	Cd, Ad	0,0 (from K)
1, 1	1, 1	na	na	na	1, 1	Cd, Ad	Cs, As
Cd, Ad	1, 1	na	na	na	Cs, As	Cd, Ad	Cd, Ad
1–Cd, 1–Ad	1, 1	na	na	na	Cs, As	1–Cd, 1–Ad	Cd, Ad
As, As	1, 1	na	na	na	1, 1	Cd, Ad	Cs*As, As*As
		na	na	na	Cs, As	As, As	Cd, Ad
1–As, 1–As	1, 1	na	na	na	1, 1	Cd, Ad	Cs*(1–As), As*(1–As)
Ad, Ad	1, 1	na	na	na	Cs, As	Ad, Ad	Cd, Ad
1–Ad, 1–Ad	1, 1	na	na	na	Cs, As	1–Ad, 1–Ad	Cd, Ad
f, 1	1, 1	na	na	na	Cs, 1	f, Ad	Cd, As
0, 0	Cs, As	na	na	na	Cs, As	Cd, Ad	0,0 (from K)
1, 1	Cs, As	na	na	na	Cs, As	Cd, Ad	Cs, As
Cd, Ad	Cs, As	Cs, As	Cd, Ad	x	Cs, As	Cd, Ad	Loop Back(MPY)
1–Cd, 1–Ad	Cs, As	Cs, As	1–Cd, 1–Ad	x	Cs, As	Cd, Ad	Loop Back(MPY)
As, As	Cs, As	na	na	Cs*As, As*As	Cs, As	Cd, Ad	Loop Back(ADD)
		Cs, As	As, As	x	Cs, As	Cd, Ad	Loop Back(MPY)

Cs=Rs,Gs,Bs; Cd=Rd,Gd,Bd; x=don’t care; na=not applicable; f=min(As,1–Ad); *=arithmetic multiplication
MPY=multiplier result; ADD=Addend term; K=Constant Source register; DQ=PALU_DQ pins

Table 3.5 Multiplicand/Addend selection for each OpenGL blending mode

Blending Fractions		Preblend Cycle			Normal Cycle		
sfactor	dfactor	MULTP1	MULTP2	ADDEND	MULTP1	MULTP2	ADDEND
1–As, 1–As	Cs, s	na	na	Cs*(1–As), As*(1–As)	Cs, As	Cd, Ad	Loop Back(ADD)
Ad, Ad	Cs, As	Cs, As	Ad, Ad	x	Cs, As	Cd, Ad	Loop Back(MPY)
1–Ad, 1–Ad	Cs, As	Cs, As	1–Ad, 1–Ad	x	Cs, As	Cd, Ad	Loop Back(MPY)
f, 1	Cs, As	Cs, na	f, na	na, As	Cs, As	Cd, Ad	Loop Back(MPY), Loop Back(ADD)
0, 0	1–Cs, 1–As	na	na	na	1–Cs, 1–As	Cd, Ad	0,0 (from K)
1, 1	1–Cs, 1–As	na	na	Cs, As	1–Cs, 1–As	Cd, Ad	Loop Back(ADD)
Cd, Ad	1–Cs, 1–As	Cs, As	Cd, Ad	x	1–Cs, 1–As	Cd, Ad	Loop Back(MPY)
1–Cd, 1–Ad	1–Cs, 1–As	Cs, As	1–Cd, 1–Ad	x	1–Cs, 1–As	Cd, Ad	Loop Back(MPY)
As, As	1–Cs, 1–As	na	na	Cs*As, As*As	1–Cs, 1–As	Cd, Ad	Loop Back(ADD)
		Cs, As	As, As	x	1–Cs, 1–As	Cd, Ad	Loop Back(MPY)
1–As, 1–As	1–Cs, 1–As	na	na	Cs*(1–As), As*(1–As)	1–Cs, 1–As	Cd, Ad	Loop Back(ADD)
Ad, Ad	1–Cs, 1–As	Cs, As	Ad, Ad	x	1–Cs, 1–As	Cd, Ad	Loop Back(MPY)
1–Ad, 1–Ad	1–Cs, 1–As	Cs, As	1–Ad, 1–Ad	x	1–Cs, 1–As	Cd, Ad	Loop Back(MPY)
f, 1	1–Cs, 1–As	Cs, na	f, na	na, As	1–Cs, 1–As	Cd, Ad	Loop Back(MPY), Loop Back(ADD)
0, 0	As, As	na	na	na	As, As	Cd, Ad	0,0 (from K)
1, 1	As, As	na	na	na	As, As	Cd, Ad	Cs, As
Cd, Ad	As, As	Cs, As	Cd, Ad	x	As, As	Cd, Ad	Loop Back(MPY)
1–Cd, 1–Ad	As, As	Cs, As	1–Cd, 1–Ad	x	As, As	Cd, Ad	Loop Back(MPY)
As, As	As, As	na	na	na	As, As	Cd, –	Cs*As, –
		Cs, As	As, As	x	As, As	Cd, Ad	Loop Back(MPY)
1–As, 1–As	As, As	na	na	na	As, –	Cd, –	Cs*(1–As), –
		x	x	Cs*(1–As), As*(1–As)	As, As	Cd, Ad	Loop Back(ADD)
Ad, Ad	As, As	Cs, As	Ad, Ad	x	As, As	Cd, Ad	Loop Back(MPY)
1–Ad, 1–Ad	As, As	Cs, As	1–Ad, 1–Ad	x	As, As	Cd, Ad	Loop Back(MPY)
f, 1	As,As	Cs, na	f, na	na, As	As, As	Cd, Ad	Loop Back(MPY), Loop Back(ADD)
0, 0	1–As, 1–As	na	na	na	1–As, 1–As	Cd, Ad	0,0 (from K)
1, 1	1–As, 1–As	na	na	na	1–As, –	Cd, –	Cs, –
		x	x	Cs, As	1–As, 1–As	Cd, Ad	Loop Back(ADD)
Cd, Ad	1–As, 1–As	Cs, As	Cd, Ad	x	1–As, 1–As	Cd, Ad	Loop Back(MPY)
1–Cd, 1–Ad	1–As, 1–As	Cs, As	1–Cd, 1–Ad	x	1–As, 1–As	Cd, Ad	Loop Back(MPY)
As, As	1–As, 1–As	na	na	na	1–As, 1–As	Cd, –	Cs*As, –
		Cs, As	As, As	x	1–As, 1–As	Cd, Ad	Loop Back(MPY)
1–As, 1–As	1–As, 1–As	na	na	na	1–As, 1–As	Cd, –	Cs*(1–As), –
		x	x	Cs*(1–As), As*(1–As)	1–As, 1–As	Cd, Ad	Loop Back(ADD)
Ad, Ad	1–As, 1–As	Cs, As	Ad, Ad	x	1–As, 1–As	Cd, Ad	Loop Back(MPY)
1–Ad, 1–Ad	1–As, 1–As	Cs, As	1–Ad, 1–Ad	x	1–As, 1–As	Cd, Ad	Loop Back(MPY)
f, 1	1–As, 1–As	Cs, As	f, 1	x	1–As, 1–As	Cd, Ad	Loop Back(MPY)

Cs=Rs,Gs,Bs; Cd=Rd,Gd,Bd; x=don't care; na=not applicable; f=min(As,1–Ad); *=arithmetic multiplication
MPY=multiplier result; ADD=Addend term; K=Constant Source register; DQ=PALU_DQ pins

Table 3.5 Multiplicand/Addend selection for each OpenGL blending mode

Blending Fractions		Preblend Cycle			Normal Cycle		
sfactor	dfactor	MULTP1	MULTP2	ADDEND	MULTP1	MULTP2	ADDEND
0, 0	Ad, Ad	na	na	na	Cd, Ad	Cd, Ad	0,0 (from K)
1, 1	Ad, Ad	na	na	na	Cd, Ad	Cd, Ad	Cs, As
Cd, Ad	Ad, Ad	Cs, As	Cd, Ad	x	Cd, Ad	Ad, Ad	Loop Back(MPY)
1–Cd, 1–Ad	Ad, Ad	Cs, As	1–Cd, 1–Ad	x	Cd, Ad	Ad, Ad	Loop Back(MPY)
As, As	Ad, Ad	na	na	na	Cd, Ad	Ad, Ad	Cs*As, As*As
1–As, 1–As	Ad, Ad	na	na	na	Cd, Ad	Ad, Ad	Cs*(1–As), As*(1–As)
Ad, Ad	Ad, Ad	Cs, As	Ad, Ad	x	Cd, Ad	Ad, Ad	Loop Back(MPY)
1–Ad, 1–Ad	Ad, Ad	Cs, As	1–Ad, 1–Ad	x	Cd, Ad	Ad, Ad	Loop Back(MPY)
f, 1	Ad, Ad	Cs, na	f, na	na, As	Cd, Ad	Ad, Ad	Loop Back(MPY), Loop Back(ADD)
0, 0	1–Ad, 1–Ad	na	na	na	Cd, Ad	1–Ad, 1–Ad	0,0 (from K)
1, 1	1–Ad, 1–Ad	na	na	na	Cd, Ad	1–Ad, 1–Ad	Cs, As
Cd, Ad	1–Ad, 1–Ad	Cs, As	Cd, Ad	x	Cd, Ad	1–Ad, 1–Ad	Loop Back(MPY)
1–Cd, 1–Ad	1–Ad, 1–Ad	Cs, As	1–Cd, 1–Ad	x	Cd, Ad	1–Ad, 1–Ad	Loop Back(MPY)
As, As	1–Ad, 1–Ad	na	na	na	Cd, Ad	1–Ad, 1–Ad	Cs*As, As*As
1–As, 1–As	1–Ad, 1–Ad	na	na	na	Cd, Ad	1–Ad, 1–Ad	Cs*(1–As), As*(1–As)
Ad, Ad	1–Ad, 1–Ad	Cs, As	Ad, Ad	x	Cd, Ad	1–Ad, 1–Ad	Loop Back(MPY)
1–Ad, 1–Ad	1–Ad, 1–Ad	Cs, As	1–Ad, 1–Ad	x	Cd, Ad	1–Ad, 1–Ad	Loop Back(MPY)
f, 1	1–Ad, 1–Ad	Cs, na	f, na	na, As	Cd, Ad	1–Ad, 1–Ad	Loop Back(MPY), Loop Back(ADD)

Cs=Rs,Gs,Bs; Cd=Rd,Gd,Bd; x=don't care; na=not applicable; f=min(As,1–Ad); *=arithmetic multiplication
MPY=multiplier result; ADD=Addend term; K=Constant Source register; DQ=PALU_DQ pins

Stencil Modes *(NEW)*

The stencil buffer in a 3D graphics system may be used to restrict drawing to a certain portion of the screen, just as a cardboard stencil may be used with a can of spray paint to make precise, painted images. 3D-RAM offers a broad range of support for on-chip stencil hardware acceleration. There are two distinct stencil modes supported inside the 3D-RAM. The OpenGL stencil mode is fully compliant with the OpenGL specification that allows for any number of stencil planes from 0 through 8. The 3D-RAM also offers the decal stencil mode, which is compatible with the previous generation of the 3D-RAM, M5M410092A. These two stencil modes should not be used at the same time. If the OpenGL stencil mode is being used, the decal stencil mode should be disabled. Similarly, the OpenGL stencil mode should be disabled when using the decal stencil mode. However, the 3D-RAM chip itself does not check or inhibit such conflict, and it is the controller's responsibility to ensure that such conflict does not occur.

OpenGL Stencil Mode Operations (NEW)

This stencil mode provides fully compliant OpenGL stencil operations. The 3D-RAM stencil features are implemented in ROP/Blend unit 3 so that bits [31:24] of the 32-bit ALU unit are available as stencil planes. These 8 bits are bitwise enabled, so that any number of stencil planes from 0 through 8 may be used.

There are two parts in the data flow of this stencil mode, as illustrated in Figure 3.8, and the paragraphs that follow refer to the blocks in this figure. The first part involves a stencil function which compares the OLD stencil data to the reference data StF.REF which may be either the most significant byte data at the PALU_DQ pins or the most significnat byte in the Constant Source register. The second part involves the execution

of a certain stencil operation on the OLD stencil data or the StF.REF data, based on the results of the stencil and magnitude comparison functions.

The Stencil Function block compares the magnitude of the stencil reference data, StF.REF, to the OLD stencil data that is read from the Pixel Buffer. A mask for the stencil data is available which provides the capability to ignore certain bits in the stencil data comparison. The exact type of comparison executed in the Stencil Function block is defined by $StF.FUNC_{[2:0]}$. The options for this comparison are: GL_ALWAYS, GL_GREATER, GL_EQUAL, GL_GEQUAL, GL_NEVER, GL_LEQUAL, GL_NOTEQUAL, and GL_LESS, as defined in Table 3.29.

The Stencil Operation block considers the results from the Stencil Function and Magnitude Compare and alters the stencil data stored in the Pixel Buffer based on the settings of the Stencil Control register. The Stencil Operation block can be set to ZERO, KEEP, INVERT, REPLACE, INCREMENT, or DECREMENT the stencil planes. The actions taken by the Stencil Operation block for three different cases are defined in Table 3.26 through Table 3.28.

Separately, the SRAM Write Enable Logic block considers the PASS_IN pins, the Match Compare result, the Magnitude Compare result, and the

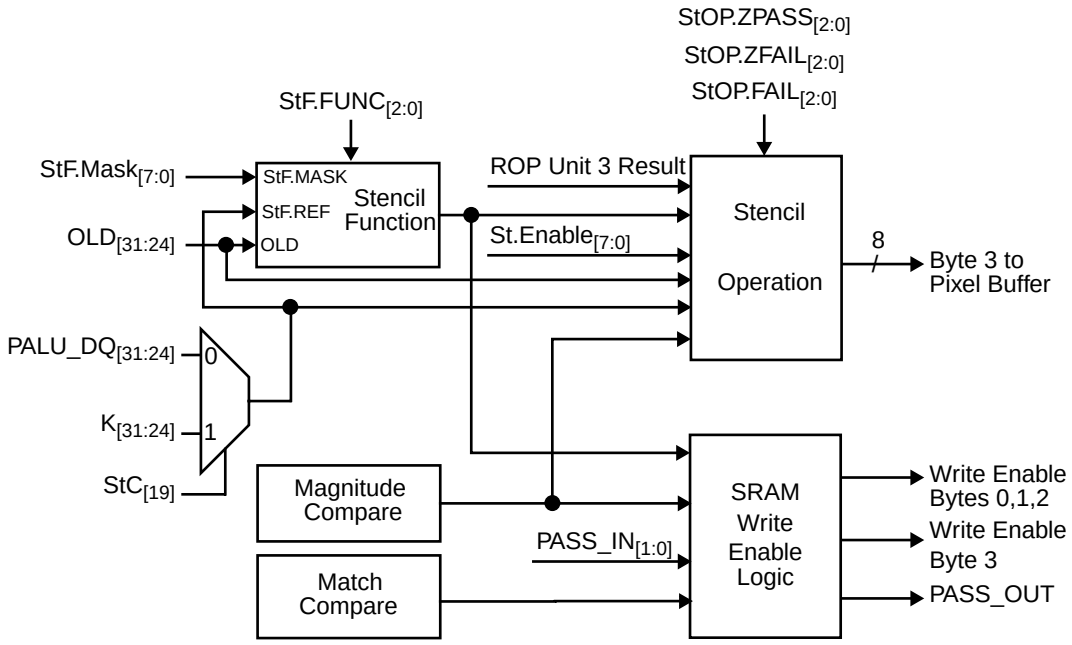


Figure 3.8 Operations of OpenGL stencil mode

Stencil Function result. If either of the $PASS\_IN_{[1:0]}$ pins is “0” or the Match Compare result is “0”, the Pixel Buffer will not be updated and the PASS_OUT pin will be “0”. If both $PASS\_IN_{[1:0]}$ pins are “1” and the Match Compare passes, then depending on the results of the Stencil Function and the Magnitude Compare, the SRAM Write Enable Logic block determines whether to write to the Pixel Buffer and what the state of the PASS_OUT pin should be.

It is helpful to point out that for the stencil support to be useful in the overall graphics processing, the overriding conditions above and beyond the

results of the stencil comparison functions and depth test are the states of two PASS_IN pins and the Match Compare result. For example, it may be desired to restrict stencil functions and stencil operations to only a certain window on the display monitor based on the result of the comparison of window ID bits, which may or may not be stored on the same 3D-RAM chips as the Z buffer and the stencil planes.

The pseudo code below summarizes the above explanations in a concise format. Note that the expression “St.Test(StF)” refers to an OpenGL stencil test based on the stencil function selected in the glStencil_Func. The glStencil_Func should set and reset the bits in the 3D-RAM Stencil Planes register and Stencil Control register. Note also that the above pseudo code assumes that the stencil operations StOP.FAIL, StOP.ZPASS, and StOP.ZFAIL take $St.Enable_{[7:0]}$ into account for proper execution of the GL_DECR and GL_INCR operations, with the correct bit alignment and underflow and overflow clamping. The details of the calculations are not explicitly shown. See also page 67 for the paragraph describing the bit field St.Enable.

```
IF (StC[19]==0)
  StF.REF=PALU_DQ[31:24]
ELSE
  StF.REF=K[31:24]

IF( ((!PINS[0]||PASS_IN[0]) &&
  (!PINS[8]||PASS_IN[1]) && MATCH_COMP)==0 )
  {
  Pixel_Buffer_Write_Enable_Byte[3:0]=0000b
  PASS_OUT=0
  }
ELSE
/* match test passes and both PASS_IN */
/* pins TRUE                          */ {
  IF ( St.Test(StF)==0 )
    {
    Pixel_Buffer_Write_Enable_Byte[3:0]
      =1000b
    For bits disabled by St.Enable[7:0]
      Pixel_Buffer_Data_In[31:24]=OLD[31:24]
    For bits enabled by St.Enable[7:0]
      Pixel_Buffer_Data_In[31:24]
        =StOP.FAIL(StF.REF, OLD[31:24])
    PASS_OUT=0
    }
  ELSE /* stencil test passes */
    {
    IF ( MAGNITUDE_COMP==1 )
      {
      Pixel_Buffer_Write_Enable_Byte[3:0]
        =1111b
      For bits disabled by St.Enable[7:0]
        Pixel_Buffer_Data_In[31:24]
          =ROP(OLD[31:24])
      For bits enabled by St.Enabled[7:0]
        Pixel_Buffer_Data_In[31:24]
          =StOP.ZPASS(StF.REF, OLD[31:24])
      PASS_OUT=1
      }
    ELSE /* depth test fails */
      {
      Pixel_Buffer_Write_Enable_Byte[3:0]
        =1000b
      For bits disabled by St.Enable[7:0]
        Pixel_Buffer_Data_In[31:24]
          =OLD[31:24]
      For bits enabled by St.Enable[7:0]
        Pixel_Buffer_Data_In[31:24]
          =StOP.ZFAIL(StF.REF, OLD[31:24])
      PASS_OUT=0
      }
    } /* depth test */
  } /* stencil test */
} /* PASS_IN/MATCH test */
```

List 3.1 Pseudo code for OpenGL stencil operations

Restrictions on OpenGL Stencil Mode

There are several restrictions that must be met for the OpenGL stencil mode to function correctly. The restrictions are listed below.

- In order for INCREMENT and DECREMENT operations to perform correctly, it is necessary that the enabled stencil bits be in one contiguous group.
- The Z planes must be stored in the same 3D-RAM chip as the stencil planes.
- The Magnitude Compare unit must be used for the depth test.
- If any stencil bits are enabled, the ROP/Blend unit 3 cannot be used for blending, i.e. RBC[27] must be “0”.
- The source for the stencil reference value (StF.REF) is selected by the Stencil Control register bit 19 ($StC_{[19]}$). When $StC_{[19]}$ is 0, $PALU\_DQ_{[31:24]}$ is StF.REF, and when $StC_{[19]}$ is 1, bits 31 through 24 of the Constant Source register ($K_{[31:24]}$) is StF.REF. StF.REF is used in both stencil test and stencil operations at the same time.

Decal Stencil Mode Operations

This decal stencil mode is offered to provide compatibility wiith the stencil mode of the previous generation of the 3D-RAM, M5M410092A. By setting the Match Mask register and the Magnitude Mask register, the user has flexible plane depths for the stencil buffer and the Zbuffer, respectively. The Match Compare unit, as in the normal, non-stencil mode, supports NEVER, ALWAYS, EQUAL, and NOTEQUAL stencil test functions, while the byte-wide ROP units support only the logical stencil update operations, namely KEEP, ZERO, REPLACE, and INVERT, plus the additional ONE operation; the arithmetic stencil update operations INCREMENT and DECREMENT are not implemented in the Decal/Invert Stencil Mode.

The decal stencil mode may be selected by setting the Compare Control register bit 10 to "1". In this mode, the internal Pixel Buffer write enable for a stateful write is no longer solely controlled by $PASS\_IN_{[1:0]}$ and PASS_OUT. The added condition to the write enable signal generation is the output of the Match Compare unit, which now overwrites the PASS_OUT and the inverted output of the Match Compare unit is logically ANDed with $PASS\_IN_{[1:0]}$ to set the Pixel Buffer write enable signal. When the stencil buffer and the Z buffer are placed on the same 3D-RAM chip, the Match Compare unit performs the stencil match test and the Magnitude Compare unit performs the Z (depth) compare test. When both tests pass, the PASS_OUT can enable the update of the color buffer in other 3D-RAM chips through their PASS_IN pins. In the meantime, the stencil and Z buffers are also updated internally. If the stencil match test fails, then only the stencil and Z buffer may be updated internally and the color buffer on other 3D-RAM chips are left unchanged.

Table 3.6 Stateful write Enable in Decal stencil mode

Stencil Mode?	Stateful Write Enable	PASS_OUT
No	$PASS\_IN_{[1:0]}$ && PASS_OUT	MAGPASS && MATCHPASS
YES	$PASS\_IN_{[1:0]}$ && (PASS_OUT + ~MATCHPASS)	MAGPASS && MATCHPASS

Invert Stencil Mode Operations

Warning! The invert stencil mode is removed from this device M5M410092B, and an incompatibility exists between this device M5M410092B and its previous generation, M5M410092A, with respect to this function.

16-bit Color Modes

There are two popular 16-bit color modes: the (4, 4, 4, 4) mode with 4 bits each for Alpha, Red, Green, and Blue; and the (5, 6, 5, 0) mode with 5-bit Red, 6-bit Green, and 5-bit Blue. The 16-bit on-chip blending functions are only for (4, 4, 4, 4) mode. If not invoking any ALU operation, the 3D-RAM can be used to store (5, 6, 5, 0) 16-bit data.

In the normal (8, 8, 8, 8) color mode, Alpha must be placed in the most significant byte $PALU\_DQ_{[31:24]}$ due to the special circuits to handle the case of Alpha-Saturate in Blend unit 3. Red, green, and blue color data may be placed on the other three bytes $PALU\_DQ_{[23:0]}$ in any permutation. The four ROP/Blend units operate on the four color components and write the results back to the Pixel Buffer.

In the (4, 4, 4, 4) 16-bit mode, the 32-bit bus within the 3D-RAM is used for double buffering the 16-bit (4, 4, 4, 4) data. For example, Buffer A data is placed on the upper nibble of each byte: $PALU\_DQ_{[31:28]}$ for Alpha, and $PALU\_DQ_{[23:20]}$, $PALU\_DQ_{[15:12]}$, and $PALU\_DQ_{[7:4]}$ for red, green, and blue; and Buffer B data is placed on the lower nibble of each byte: $PALU\_DQ_{[27:24]}$ for Alpha, and $PALU\_DQ_{[19:16]}$, $PALU\_DQ_{[11:8]}$, and $PALU\_DQ_{[3:0]}$ for red, green, and blue. Note again that the Alpha data of both Buffers A and B can only be stored in the most significant byte.

Byte Enable and Nibble Control

Let NBLn represent the nibble data on the $PALU\_DQ_{[4n+3:4n]}$ pins, for n = 0 through 7. In the (8, 8, 8, 8) mode, $PALU\_BE_{[3]}$ enables NBL7 and NBL6; $PALU\_BE_{[2]}$ enables NBL5 and NBL4; $PALU\_BE_{[1]}$ enables NBL3 and NBL2; $PALU\_BE_{[0]}$ enables NBL1 and NBL0. An example of this arrangement is illustrated in Figure 3.9 below.

3 Pixel ALU Operations

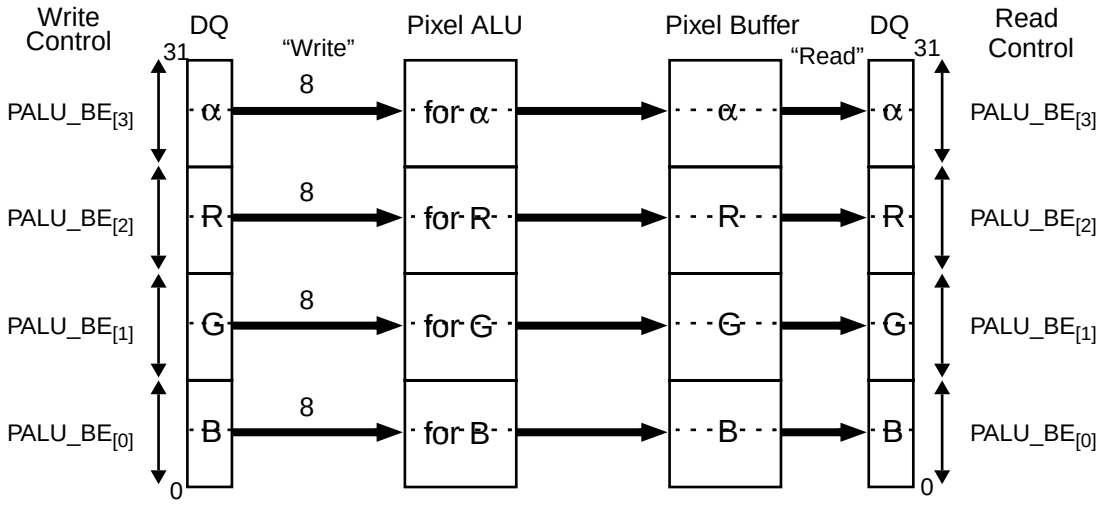


Figure 3.9 Data mapping in the (8, 8, 8, 8) color mode

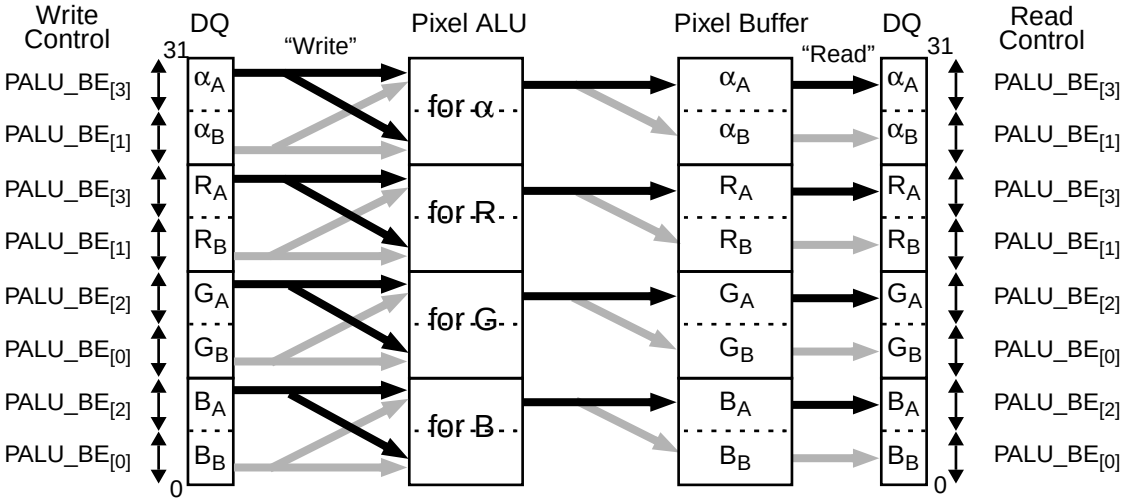


Figure 3.10 Data Mapping in the (4, 4, 4, 4) Color Mode

In the (4, 4, 4, 4) mode, $PALU\_BE_{[3]}$ enables NBL7 and NBL5; $PALU\_BE_{[2]}$ enables NBL3 and NBL1; $PALU\_BE_{[1]}$ enables NBL6 and NBL4; $PALU\_BE_{[0]}$ enables NBL2 and NBL0. From the rendering controller output, $PALU\_BE_{[3,2]}$ controls (A, R, G, B) in Buffer A, while $PALU\_BE_{[1,0]}$ controls (A, R, G, B) in Buffer B. This byte enable assignment allows for support of (4, 4, 4, 4) and

(5, 6, 5, 0) data formats with identical PALU_BE controls from the controller. Figure 3.10 illustrates how data is mapped in the (4, 4, 4, 4) mode.

The solid black arrows correpond to data flow of Buffer A, and the gray arrows correspond to data flow of Buffer B. During write operations, the 4-bit data may be duplicated or padded with zeros in the lower nibble when processed by the byte-wide Pixel ALU; after the Pixel ALU processing, the lower nibble of the resulting 8-bit data is truncated to form a 4-bit data before written into the Pixel Buffer.

To store 16-bit color in (5, 6, 5, 0) data format, the 3D-RAM is programmed to operate in (8, 8, 8, 8) mode. The (R, G, B) data for buffer A is stored in NBL7 to NBL4, which are controlled by $PALU\_BE_{[3:2]}$. The (R, G, B) data for Buffer B is stored in NBL3 to NBL0, which are controlled by $PALU\_BE_{[1:0]}$. The data mapping for the (5, 6, 5, 0) color mode is shown in Figure 3.11 below.

In summary, when the controller is in 16-bit color mode, either (4, 4, 4, 4) or (5, 6, 5, 0), asserting $PALU\_BE_{[3:2]}$ controls Buffer A read and write operations; asserting $PALU\_BE_{[1:0]}$ controls Buffer B read and write operations. Because the same blending circuits for (4, 4, 4, 4) mode are used for both Color Buffers, Buffer A and B Stateful Writes cannot occur on the same clock cycle.

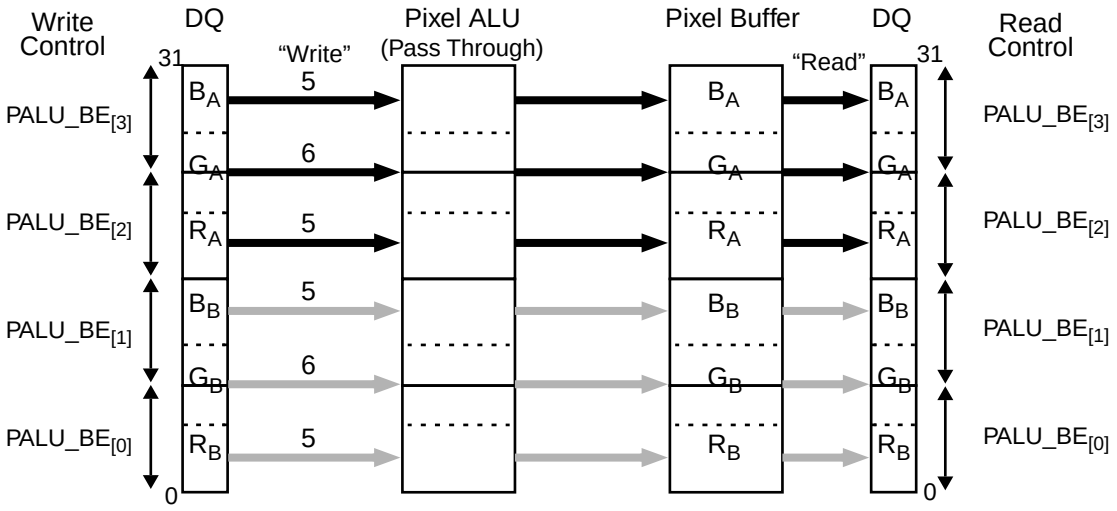


Figure 3.11 Data mapping in the (5, 6, 5, 0) color mode

Table 3.7 Byte Enable controls and color data placement in (4,4,4,4) mode

PALU_DQ	[31:28]	[27:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:4]	[3:0]
NBL	7	6	5	4	3	2	1	0
PALU_BE	3	1	3	1	2	0	2	0
Color Data	α_A	α_B	R_A	R_B	G_A	G_B	B_A	B_B

Table 3.8 Byte Enable controls and color data placement in (5, 6, 5, 0) mode

PALU_DQ	[31:28]	[27:24	[23:20]	[19:16]	[15:12	[11:8]	[7:4]	[3:0]
NBL	7	6	5	4	3	2	1	0
PALU_BE	3		2		1		0	
Color Data	R_A,G_A,B_A				R_B,G_B,B_B			

The following is how $PALU\_BE_{[3:0]}$ can be used in (4, 4, 4, 4) mode:

- In Read Data, Stateless Initial/Normal Write operations
  - $PALU\_BE_{[3:0]}$ may be any combination (decoded as in Table 3.7)
- In Stateful Initial/Normal Write, Initiate Two-Cycle Blending operations
  - $PALU\_BE_{[3:0]}$ controls either Buffer A (as 1100b) data or Buffer B (as 0011b) data, but not both (i.e. not 1111b)

    Warning! During a Stateful Write or Initiate Two-Cycle Blending operation, $PALU\_BE_{[3:0]}$ should be set to only enable either Buffer A or Buffer B, but not both. Enabling both buffers would result in Buffer A and Buffer B datausing the same blending circuits at the same time and therefore in undefined resulting data. In other words, in these operations $PALU\_BE_{[3]}$ and $PALU\_BE_{[1]}$ cannot be enabled at the same time. Similarly, $PALU\_BE_{[2]}$ and $PALU\_BE_{[0]}$ cannot be enabled at the same time.
- PALU_BE mapping to the Dirty Tag in Stateful/Stateless Writes operations
  - Either $PALU\_BE_{[0]}$ or $PALU\_BE_{[2]}$ sets the Dirty_Tag bits corresponding to both Byte 0 and Byte 1 of the addressed data word
  - Either $PALU\_BE_{[1]}$ or $PALU\_BE_{[3]}$ sets the Dirty_Tag bits corresponding to both Byte 2 and Byte 3 of the addressed data word
- Write Control Register(s)
  - Same decoding as in (8, 8, 8, 8) mode
- Read ID Register
  - Same decoding as in (8, 8, 8, 8) mode)
- (4, 4, 4, 4) mode has no effect on the use of the Dirty Tag (Masked/Unmasked Write Block from the Pixel Buffer to a DRAM bank); it only affects the writing of the Dirty Tag during Stateful/Stateless Initial/ Normal Writes operations.

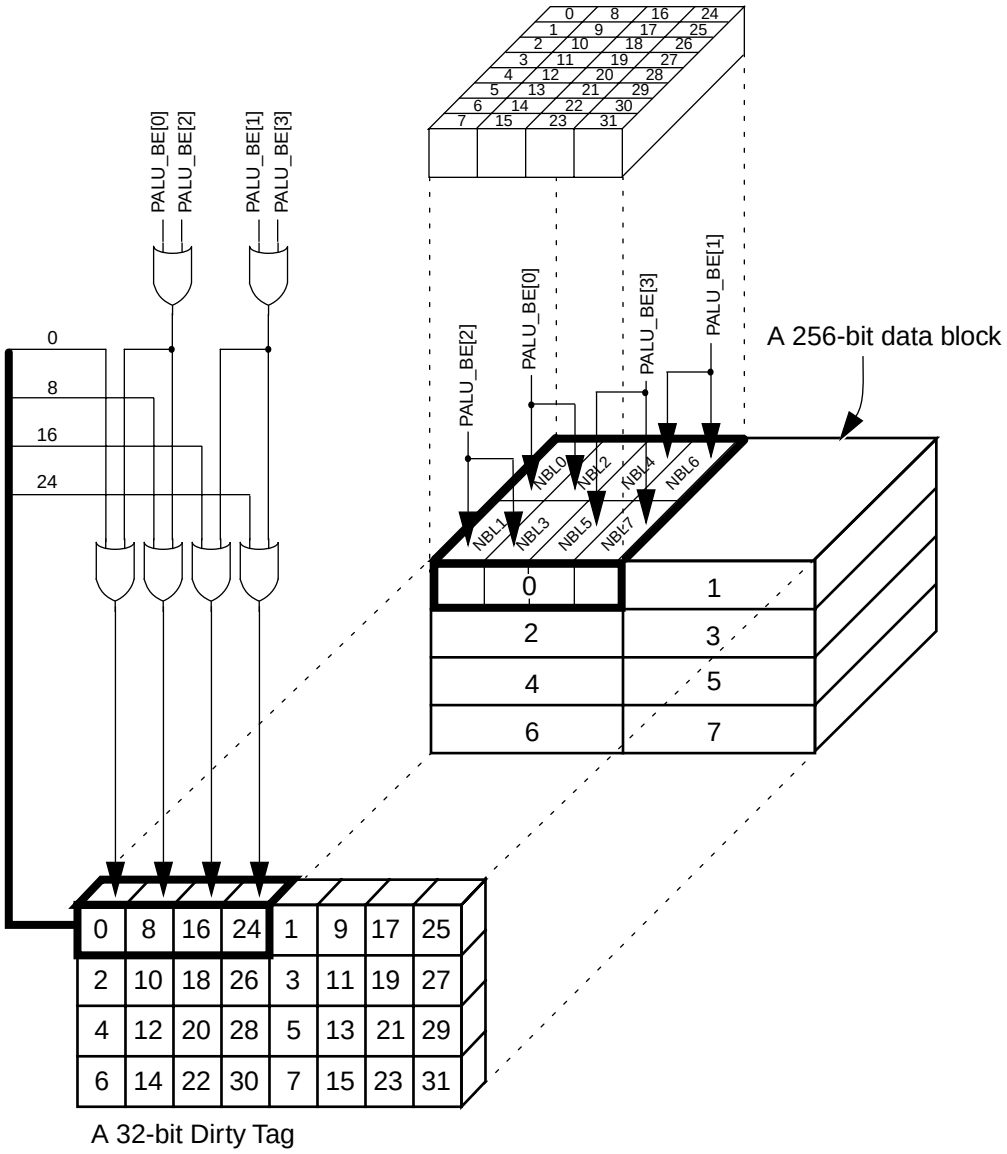
Use of Dirty Tags in (4, 4, 4, 4) 16-bit Mode

The mapping for the “Replace Dirty Tag” and “OR Dirty Tag” operations in (4, 4, 4, 4) mode is the same as in (8, 8, 8, 8) mode, only that the PALU_BE control decoding is for the (4, 4, 4, 4) mode. The Dirty Tag functions in (4, 4, 4, 4) mode are summarized in Table 3.9. The mapping of the PALU_BE pins to the Dirty Tags is illustrated in Figure 3.12.

Table 3.9 Pixel ALU operations involving Dirty Tags in (4, 4, 4, 4) mode

Pixel Operation	Pixel Data	New Dirty Tag Contents
(Stateful/Stateless) Normal Data Write	Write to Buffer A from PALU_DQ pins (per $PALU\_BE_{[3:2]}$ pins)	The Dirty Tag bits for bytes 3 and 2 are ORed with $PALU\_BE_{[3]}$; the Dirty Tag bits for bytes 1 and 0 are ORed with $PALU\_BE_{[2]}$; the other 28 Dirty Tag bits are unchanged.
(Stateful/Stateless) Normal Data Write	Write to Buffer B from PALU_DQ pins (per $PALU\_BE_{[1:0]}$ pins)	The Dirty Tag bits for bytes 3 and 2 are ORed with $PALU\_BE_{[1]}$; the Dirty Tag bits for bytes 1 and 0 are ORed with $PALU\_BE_{[0]}$; the other 28 Dirty Tag bits are unchanged.
(Stateful/Stateless) Initial Data Write	Write to Buffer A from PALU_DQ pins (per $PALU\_BE_{[3:2]}$ pins)	$PALU\_BE_{[3]}$ is written to the Dirty Tag bits for bytes 3 and 2; $PALU\_BE_{[2]}$ is written to the Dirty Tag bits for bytes 1 and 0; “0” is written to the 28 unaddressed Dirty Tag bits.
(Stateful/Stateless) Initial Data Write	Write to Buffer B from PALU_DQ pins (per $PALU\_BE_{[1:0]}$ pins)	$PALU\_BE_{[1]}$ is written to the Dirty Tag bits for bytes 3 and 2; $PALU\_BE_{[0]}$ is written to the Dirty Tag bits for bytes 1 and 0; “0” is written to the 28 unaddressed Dirty Tag bits.
Replace Dirty Tag	Unchanged	$PALU\_DQ_{[31:0]}$ replaces all 32 Dirty Tag bits.
OR Dirty Tag	Unchanged	All 32 Dirty Tag bits are ORed with $PALU\_DQ_{[31:0]}$.

Figure 3.12 PALU_BE Mapping to Dirty Tags for (4,4,4,4) Mode



4-bit to 8-bit Expansion for Pixel ALU Blending

The Pixel ALU blending function operates on 8-bit components. To implement the 8-bit blending operation on a 4-bit color component, it is necessary to expand the 4-bit data (either from PALU_DQ pins or from Pixel Buffer) to an 8-bit operand. For the Addend term, the 4-bit component is mapped to the upper nibble and zeros are padded into the lower nibble. For the multiplicand terms, simply padding zeros in the lower nibble would result in an incorrect pixel value due to computational error from short bit length representation. By duplicating the 4-bit data in both the upper and lower nibbles, we can avoid corrupting the pixel value. This effect can be illustrated with the following two examples, when the blending factor Alpha is equal to “F”.

Table 3.10 Blending color value and Alpha, with Alpha = “F”, zeros padded at lower nibble

Color Value	Blending (ColorxAlpha)	Arithmetic Result	Multiplier Output
E	E0 x F0	D200	D
2	20 x F0	1A00	1

Table 3.11 Blending color value and Alpha, with Alpha = “F”, duplication at lower nibble

Color Value	Blending (ColorxAlpha)	Arithmetic Result	Multiplier Output
E	EE x FF	ED12	E
2	22 x FF	21DE	2

As we can see from the Table 3.11, the duplication of the upper nibble and the lower nibble allows the color data to blend with “1”, without invoking an extra bit in the circuit. After the blending has occurred, the upper nibble is mapped back to the correct nibble in the Pixel Buffer. This mapping is illustrated in Figure 3.10.

On the other hand, in the (4, 4, 4, 4) mode, the 4-bit Addend, whether supplied from the PALU_DQ pins in both the Preblend Cycle and Normal Cycle or looped back from the Multiplier or the Addend in the Preblend Cycle, will always be paddd with 0000b in the least significant bits to minimize the error due to incorrect round-up.

Dual Compare Unit

A functional block diagram of the Dual Compare units is shown in Figure 3.13. Both Match Compare and Magnitude Compare are performed in parallel. The Match Mask and Magnitude Mask define which bits of the 32-bit word will be compared and which will be "don't care." One of the sources is always the Old Data ("O") from the Pixel Buffer. The other source is independently selectable between the New Data ("N") from the PALU_DQ pins and the Register Data ("K") from the Constant register.

The results of both Match Compare and Magnitude Compare operations are logically ANDed together to generate the PASS_OUT pin. The external $PASS\_IN_{[1:0]}$ and internal PASS_OUT are then logically ANDed together to generate the Write Enable signal to the Pixel Buffer.

(NEW) In the normal mode (all stencil planes are disabled, Stencil Function=ALWAYS PASS), the results of Match Compare and Magnitude Compare operations are logically ANDed to generate the PASS_OUT signal. The external $PASS\_IN_{[1:0]}$ and internal PASS_OUT are then logically ANDed together to generate the Write Enable signal to the Pixel Buffer. If OpenGL stencil mode is enabled, the PASS_OUT and Write Enable generation are affected by the stencil function result. Note that the Decal stencil mode logic is not shown below. See the sectoion on "Stencil Modes" on page 41 for more details on how stencil affects PASS_OUT and Pixel Buffer Write Enable

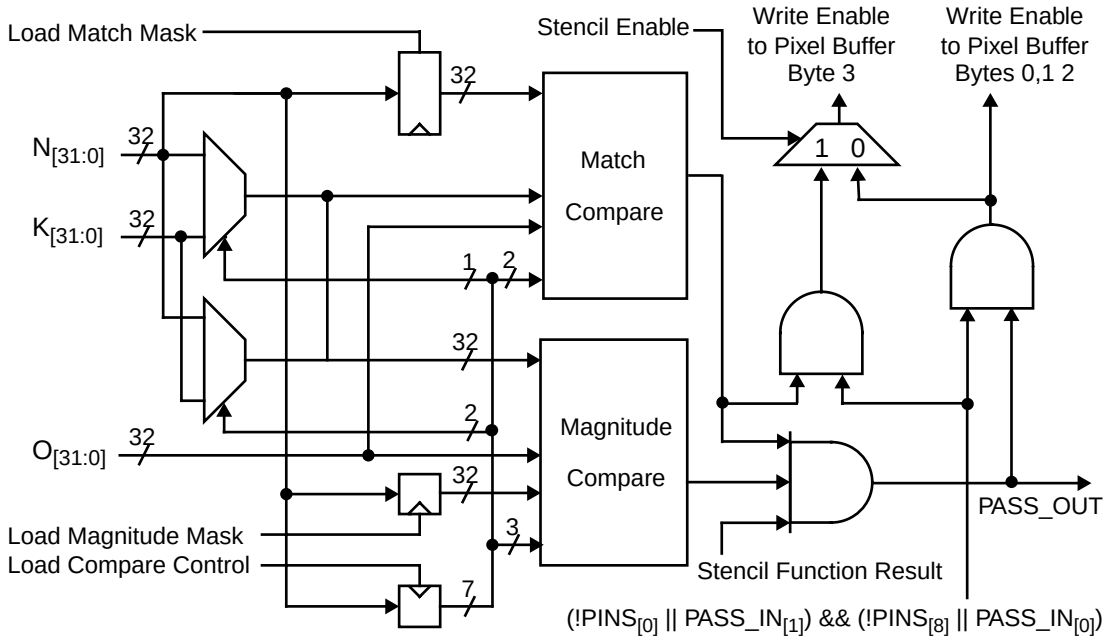


Figure 3.13 Block diagram of the Dual Compare unit (Pipeline stages are not shown)

Pipelining

The 3D-RAM Pixel ALU pipeline is designed so that the rendering controller can issue read and write operations with minimal cycle time. This is achieved by having all operations conform to a uniform 7-stage pipeline.

Table 3.12 Pixel ALU and Pixel Buffer operation pipeline

Stage	External Activities	Internal Activities
1	Operation specified on PALU_EN, PALU_WE, PALU_OP, PALU_A, and PALU_BE pins	
2	Write data on PALU_DQ and PALU_DX pins if write operation	Read Pixel Buffer. Decode operation.
3	Read data on PALU_DQ pins if read operation	First stage of ROP/Blend and Compare Units
4		Second stage of ROP/Blend and Compare Units
5		Third stage of ROP/Blend and Compare Units
6	Compare result transferred from PASS_OUT pin to PASS_IN pin	Fourth stage of ROP/Blend Units; Write Enable generated, if write operation.
7		Write result to Pixel Buffer and Dirty Tags if allowed.
8		$\overline{\text{HIT}}$ pin changes, if the write operation is a Stateful Data Write and it is successful.

A sequence of Pixel ALU write operations may be issued consecutively, one write operation per cycle. The write opcode and address are sampled by the rising edge of MCLK in one cycle, and the write data is loaded by the rising edge of the next MCLK. To the rendering controller, n consecutive writes take only *n+1* cycles. Register writes do not affect operations issued in previous cycles; register writes always affect operations issued in subsequent cycles.

Read operations may be issued consecutively, one read operation per two cycles. Specifically, all read operations require that the same address must be stable for at least two rising edges of MCLK plus the set-up time. (See Figure 8.4 for details.) Due to the pin output delay in the worst case, the read data may be available for sampling by the rendering controller in the second cycle after the read instruction. In other words, to the rendering controller, n consecutive read operations take *2(n+1)* cycles plus access time.

A read operation may be issued immediately after a write operation without any delay cycles. An idle cycle will be automatically generated by the 3D-RAM chip on the PALU_DQ pins between a write operation and the subsequent read operation.

At least two NOP cycles must be inserted between a read operation and a subsequent write operation. The two NOPs are needed to guarantee one idle cycle on the PALU_DQ pins between the read data and subsequent write data.

Figure 3.14 illustrates the above statements on the pipeline flow of Pixel ALU read/write operations.

The pipeline flow of the ROP/Blend units and the Dual Compare units of the Pixel ALU and the Pixel Buffer is illustrated in Figure 3.15. It is helpful to point out that the dotted lines show the boundaries of the pipeline stages and the numbers in the square boxes indicate the pipeline stages. A pipeline stage begins with the rising edge of MCLK and ends just prior to the next rising edge. For example, PALU_BE and PALU_A pins are presented to 3D-RAM by the rendering controller in the pipeline stage 1. On the rising edge of MCLK, they are latched in and the pipeline stage 2 starts. Beginning in the pipeline stage 3, data from the Pixel Buffer becomes available either as output to the PALU_DQ pins during an ALU read operation or as input to the Dual Compare unit and the ROP/Blend units during an ALU write operation. If it is a write, then PALU_DQ must be presented with data from the rendering controller in the pipeline stage 2, to be latched in and used by the Pixel ALU, together with the data from the Pixel Buffer.

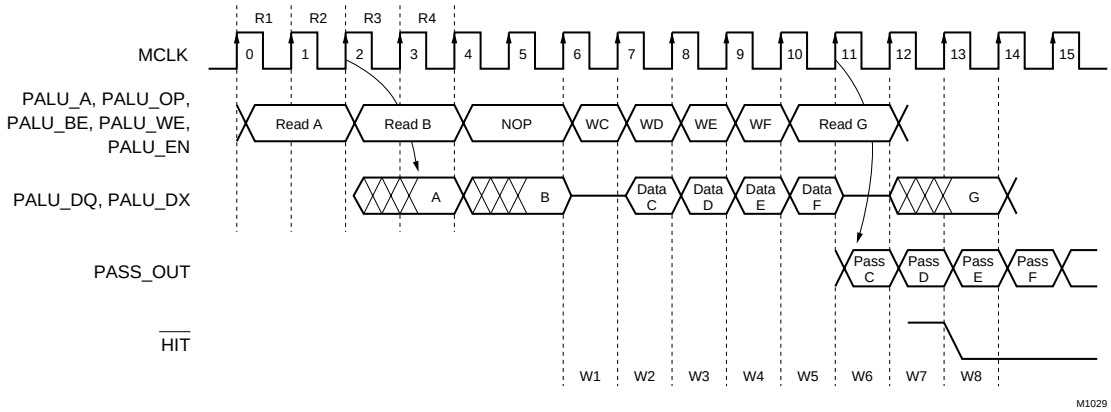


Figure 3.14 Example of Pixel Port read/write operations to satisfy the pipeline flow

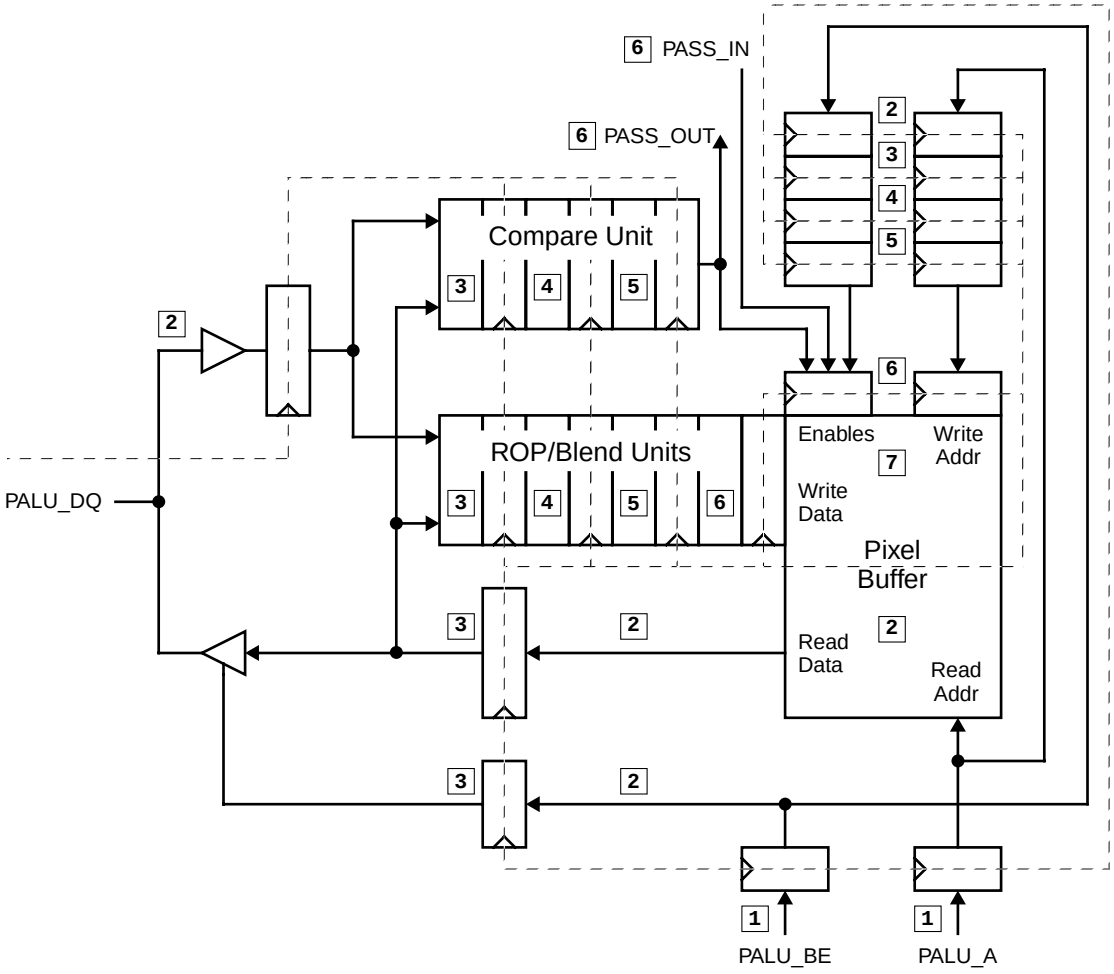


Figure 3.15 Pixel ALU and Pixel Buffer block diagram with pipeline flow

The Picking Logic

The block diagram of the Picking Logic is shown again in Figure 3.16. At the beginning, the Picking Logic should be enabled and the HIT flag should be cleared. This is done either by asserting the $\overline{\text{RESET}}$ pin low or by writing the data Eh into byte 3 of the Compare Control register. It is helpful to note that writing “1” to bits 27 and 25 of the Compare Control register effectively generates one-shots to load the Pick Enable flag and the HIT flag, respectively. The user will not need to perform a second register write operation to reset bits 27 and 25 to “0”.

The $\overline{\text{HIT}}$ pin will be set to high after seven cycles (corresponding to the pipeline stage 8, as in Table 3.12). In the figure below, this is indicated by the number 8 in the square box above the $\overline{\text{HIT}}$ pin label. A sequence of Stateful Data Write operations may be issued immediately after the register writing, and their effects will take place after the $\overline{\text{HIT}}$ pin is set high by this initial register writing. If any of the Stateful Data Writes in the sequence causes the on-chip and off-chip comparison tests to pass ($\text{PASS\_IN}_{[1:0]}$ and PASS_OUT are both high at the pipeline stage 6), the $\overline{\text{HIT}}$ pin is set low until the HIT flag is cleared by writing “10” into bits 25 and 24 of the Compare Control register. See also Figure 8.6, “Pick Logic timing”, for an illustration of the operations described in this section.

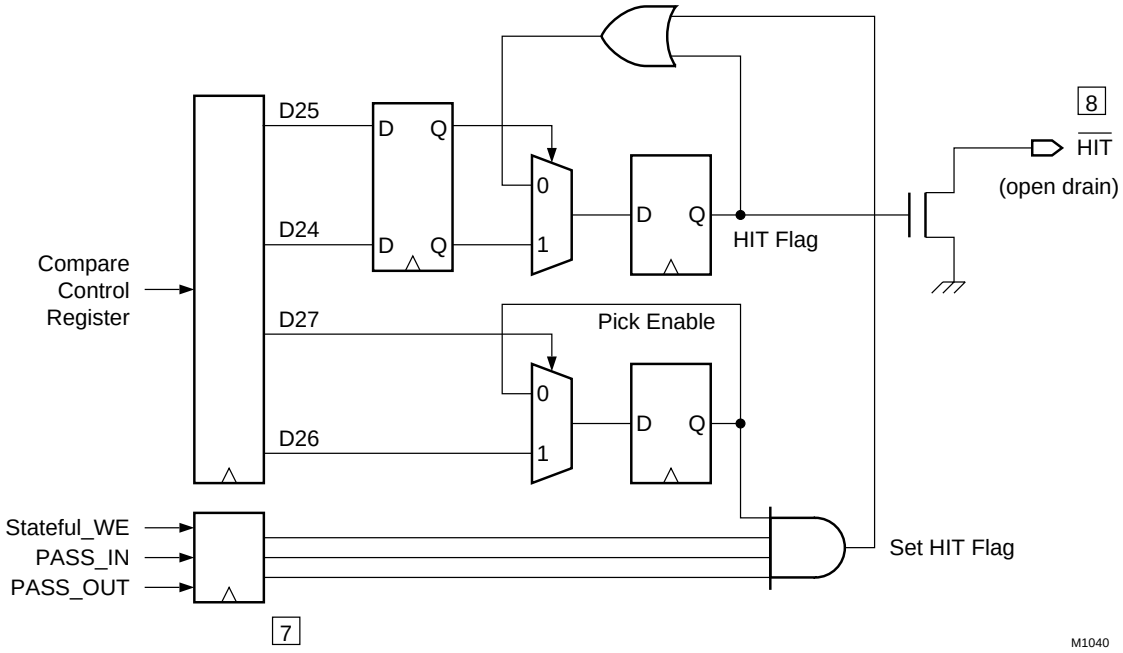


Figure 3.16 Block diagram of the Picking Logic

Pixel ALU Operations

All operations that involve the Pixel ALU and the Pixel Buffer but not the DRAM array are collectively referred to as Pixel ALU operations. Table 3.13 summarizes the Pixel ALU operations. There are two categories of Pixel ALU operations: register operations and pixel data operations. Register operations include reading the Identification register and writing the control registers; in this case, the register is specified by the PALU_A pins. Pixel data operations include reading data from the Pixel Buffer, writing data to the Pixel Buffer in four different modes, replacing Dirty Tag data, and changing Dirty Tag data with OR function; in this case, the block address is assigned by the $PALU\_A_{[5:3]}$ pins, and the word address is assigned by the $PALU\_A_{[2:0]}$ pin.

To support the all blending operations specified in the OpenGL specification version 1.1 (December 21, 1995), a new Pixel ALU command, called Initial Two-Cycle Blending, is added to this implementation of 3D-RAM.

Table 3.13 Pixel ALU operation encoding

PALU_EN	PALU_WE	PALU_OP	PALU_A	Operation
00	—	—	—	NOP
10	—	—	—	NOP
01	—	—	—	NOP
11	0	000	Block:Word	Read Pixel Buffer
11	0	001	—	Reserved
11	0	010	—	Reserved
11	0	011	—	Reserved
11	0	100	—	Reserved
11	0	101	—	Reserved
11	0	110	—	Reserved
11	0	111	000111	Read Identification Register
11	1	000	Block:Word	Stateless Initial Data Write
11	1	001	Block:Word	Stateless Normal Data Write
11	1	010	Block:Word	Stateful Initial Data Write
11	1	011	Block:Word	Stateful Normal Data Write
11	1	100	Block:xxx	Replace Dirty Tag
11	1	101	Block:xxx	OR Dirty Tag
11	1	110	Block:Word	Initiate Two-Cycle Blending (NEW)
11	1	111	Register	Write Control Registers

Register Operations

There are fourteen registers in the 3D-RAM. Their encoding is shown in Table 3.14. Among these registers, the Identification Register is read-only, and all other registers are write-only. All registers are 32 bits wide except the Constant Source register, which is 36 bits. The write-only registers are loaded from the $PALU\_DQ_{[31:0]}$. In the case of the 36-bit Constant Source register, the $PALU\_DX_{[3:0]}$ pins specify the most significant four bits. The operations launched in the previous cycles are never affected by the current register load. The operations launched in the following cycles are always affected by the register load. The $PALU\_BE_{[3..0]}$ pins apply to all register read and write operations. $PALU\_BE_0$ enables writes to bits 7 through 0; $PALU\_BE_1$ enables writes to bits 15 through 8; $PALU\_BE_2$ enables writes to bits 23 through 16; $PALU\_BE_3$ enables writes to bits 31 through 24. Finally, during the Stateful Data Write modes, all bits in these registers are fully effective; during the Stateless Data Write modes, all register bits are ignored except that the Color Depth Select register, the bits controlling the Picking Logic, and the Stencil Control register maintain some of their special functions. Please refer to the description of these bits for more details.

Table 3.14 Register address encoding

PALU_A	Register	Mnemonic	Type	Reset Value	Stateless Mode
000 000	Plane Mask	PM	Write Only	FFFF FFFFh	not applicable
000 001	Constant Source	CSR	Write Only	0 0000 0000h	not applicable
000 010	Match Mask	MatMask	Write Only	0000 0000h	not applicable
000 011	Magnitude Mask	MagMask	Write Only	0000 0000h	not applicable
000 100	ROP/Blend Control	RBC	Write Only	0303 0303h	0303 0303h
000 101	Compare Control	CCR	Write Only	0A00 0000h	0000 0000h
000 110	Write Address Control	WAC	Write Only	0000 0000h	0000 0000h
000 111	Identification*	ID	Read Only	0130 A039h	not applicable
001 000	Blend_2 Control (NEW)	BLD2	Write Only	0000 0000h	0000 0000h
001 001	Preblend Control (NEW)	PBC	Write Only	0000 0000h	0000 0000h
001 010	Stencil Planes (NEW)	StP	Write Only	00FF 0000h	00FF 0000h
001 011	Stencil Control (NEW)	StC	Write Only	3330 0000h	3330 0000h
001 110	PASS_INs Select (NEW)	PINS	Write Only	0000 0100h	not applicable
001 111	Color Depth Select (NEW)	CDS	Write Only	0000 0000h	not applicable

*Note: Reset value for Identification register is for Version 0

Identification Register (ID[31:0])

The read-only Identification register contains the manufacturer identification code (ID), part number code, and version code in the format shown in Figure 3.9. The manufacturer ID is 01Ch for Mitsubishi Electronics. The part number is read as 130Ah for M5M410092B. Bit 0 is always “1”, so for Version 0, this identifi-cation register should be read as 0130 A039h.

Plane Mask Register (PM[31:0])

This register affects both the Stateful Data Writes of the Pixel ALU operations and the Masked Write Block (MWB) of the DRAM operations. The effect is simultaneous on both types of operations. Therefore, the user must exercise caution to ensure the desired plane masking is achieved when such concurrency between the Pixel ALU and the DRAM array is exploited. For the Stateful Data Writes, each bit of the Plane Mask register is a per-bit write enable for the 32-bit data entering the Pixel Buffer. For the MWB operation, each bit of the Plane Mask register serves as a per-bit write enable for the 32-bit word 0 entering the sense amplifiers of the a DRAM bank, and the same write masking mechanism is applied to the upper seven words of the specified Pixel Buffer block. Figure 3.2 provides a clear illustration of this masking relationship between the write data and the bits of the Plane Mask register. The value “1” means write enable; the value “0” means write disable.

This register resets to FFFF FFFFh.

Constant Source Register (CSR[35:0])

This register is used to store 36-bit data that is loaded from the PALU_DQ and PALU_DX pins. (The data extension pins $PALU\_DX_{[3:0]}$ are loaded into the most significant four bits of the Constant Source register.) The bits of this register are commonly referred to as $KX_{[3:0]}$ for the most significant four bits and $K_{[31:0]}$ for the low-order 32 bits. The four ROP/Blend units and the Dual Compare units can individually select this register to provide data.

This register resets to 0000 0000h.

Match Mask Register (MTM[31:0])

This register determines which data bits participate in the match test. Setting the bits of this register to “1” causes the corresponding data bits to be compared by the Match Comparison unit. Setting the bits of this register to “0” causes the corresponding data bits to be ignored in the match test.

This register resets to 0000 0000h.

Magnitude Mask Register (MGM[31:0])

This register determines which data bits participate in the magnitude test. Setting the bits of this register to “1” causes the corresponding data bits to be compared by the Magnitude Comparison unit. Setting the bits of this register to “0” causes the corresponding data bits to be ignored in the magnitude test.

This register resets to 0000 0000h.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Version				Part Number																Manufacturer ID											1

Figure 3.17 Identification register data format

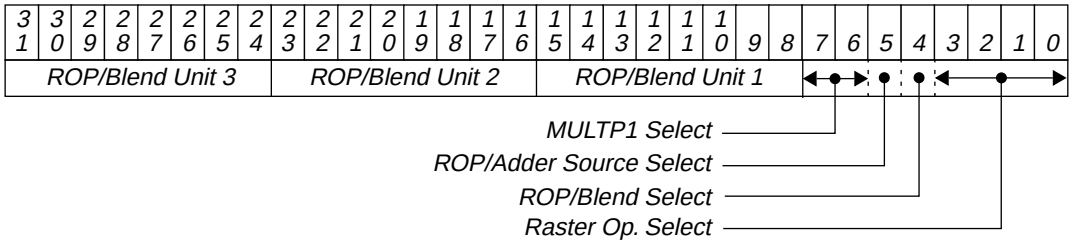


Figure 3.18 ROP/Blend Control register data format

ROP/Blend Control Register (RBC[31:0])

This register controls the operations of the four ROP/Blend units. Each ROP/Blend unit is independently controlled by an 8-bit field of this 32-bit register. Bits 7 through 0 are repeated three more times for Units 1, 2, and 3. That is, bits 15 through 8 for unit 1; bits 23 through 16 for unit 2; and bits 31 through 24 for unit 3.

This register resets to 0303 0303h. This value passes data unchanged from the PALU_DQ pins through all four ROP/Blend units.

During a Stateless Data Write access, the ROP/Blend units behave as if this register were set to 0303 0303h, regardless of its actual value.

The data format of the RBC register is explained below.

- **Bits 8n+7 through 8n+6** select a source for the multiplier fraction, MULTP1 (Table 3.15).

Table 3.15 Fraction source encoding for ROP/Blend unit n

$RBC_{[8n+7:8n+6]}$	**Fraction Source for ROP/ Blend Unit n**
00	100h (1.00)
01	{KX_n, $K_{[8n+7:8n]}$}
10	{$PALU\_DX_n$, $PALU\_DQ_{[8n+7:8n]}$}
11	{$PALU\_DX_3$, $PALU\_DQ_{[31:24]}$}

- **Bit 8n+5** selects a source for ROP unit *n* and for the adder in the Blend unit n. If this bit is “0”, the data from the $PALU\_DQ_{[8n+7:8n]}$ is selected; if this bit is “1”, the Constant Source register bits $K_{[8n+7:8n]}$ are selected.
- **Bit 8n+4** configures ROP/Blend unit *n*, for n = 0 through 2, as either a ROP unit (when bit 8n+4 = “0”) or a blend unit (when bit 8n+4 = “1”) and configures ROP/Blend unit 3 as either ROP unit which bit 27 = “0” and, when bit 27 = “1”, configures ROP/Blend unit 3 as a blend unit if the bit field St.Enable = 00h (in the Stencil Plane register) or as a stencil unit if the bit field St.Enable is non-zero. **(NEW)**
- **Bits 8n+3 through 8n+0** select one of the sixteen possible raster operations for Unit n. In Table 3.16, “NEW” represents the data from the $PALU\_DQ_{[31:0]}$ pins or from the Constant Source register bits $K_{[31:0]}$ (as selected by bit 8n+5), “OLD” represents the 32-bit data from Pixel Buffer, and “~” means logical inversion. All of these operations are bit-wise logical operations.

Table 3.16 Raster operation encoding

$RBC_{[8n+3:8n+0]}$	**Raster Operation**
0000	All bits zero
0001	NEW and OLD
0010	NEW and ~OLD

Table 3.16 Raster operation encoding

$RBC_{[8n+3:8n+0]}$	Raster Operation
0011	NEW
0100	~NEW and OLD
0101	OLD
0110	NEW xor OLD
0111	NEW or OLD
1000	~NEW and ~OLD
1001	~NEW xor OLD
1010	~OLD
1011	NEW or ~OLD
1100	~NEW
1101	~NEW or OLD
1110	~NEW or ~OLD
1111	All bits one

Compare Control Register ($CCR_{[31:0]}$)

This register controls the Picking Logic and the Dual Compare unit, and thereby indirectly influences the status of the PASS_OUT pin. Only 12 bits of this register are currently defined. The other 20 bits are reserved.

- **Bits 31 through 28, 23 through 18, 15 through 11, and 7 through 3** are reserved. They are written as “0” for future compatibility.
- **Bits 27 through 24** control the Picking Logic. Bits 25 and 24 clear or set the HIT flag. Bits 27 and 26 enable or disable the Picking Logic. The encoding tables are in Table 3.17 and Table 3.18. It is helpful to note that after bits 27 and 25 are loaded with “1”, these bits are automatically reset to “0” in the next MCLK cycle, thereby restoring the state machine to the “No Change” state and saving the follow-up register writes. In this sense, writing “1” into bits 27 and 25 generates one-shots at the outputs of $CCR_{[27]}$ and $CCR_{[25]}$.

Table 3.17 Pick Enable encoding

$CCR_{[27:26]}$	Function
0X	No change to Pick Enable flag
10	Disable Picking Logic
11	Enable Picking Logic

Table 3.18 Pick Hit encoding

$CCR_{[25:24]}$	Function
0X	No change to HIT flag
10	Clear HIT flag
11	Set HIT flag

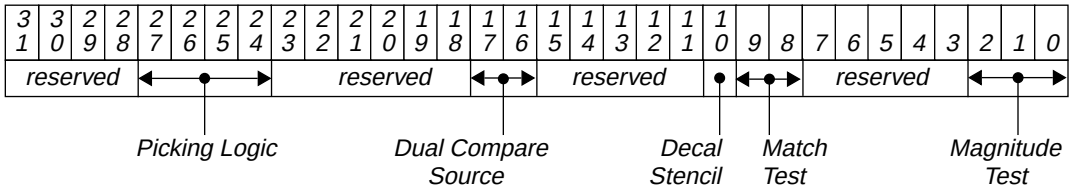


Figure 3.19 Compare Control register data format

- **Bits 17 through 16** select the source for the Dual Compare unit. Bit 16 directly controls the Match Compare source, while the result of Bit 17 XOR Bit 16 controls the Magnitude Compare source. In this way, the first two codes are compatible with the previous generations, M5M410092 and M5M410092A which had bit 17 reserved and always set to “0”. **(NEW)**

Table 3.19 Dual Compare source selection encoding

CCR [17:16]	Magnitude Compare Source	Match Compare Source	Comments
00	PALU_DQ pins	PALU_DQ pins	backward compatible
01	Constant Source register	Constant Source register	backward compatible
10	Constant Source register	PALU_DQ pins	new feature
11	PALU_DQ pins	Constant Source register	new feature

- **Bit 11** previously enables invert stencil mode in M5M410092A but is now reserved and should be always written as “0”.

  Warning! The invert stencil mode is removed from this device M5M410092B, and an incompatibility exists between this device M5M410092B and its previous generation, M5M410092A, with respect to this function.

- **Bit 10** enables the decal stencil mode (see the section on “Stencil Modes” on page 41). “0” selects the normal rendering operation, where stateful write is enabled when $PASS\_IN_{[1:0]}$ and PASS_OUT all are high. “1” selects the decal stencil mode, where the stateful write is enabled in one of the two conditions: (1) $PASS\_IN_{[1:0]}$ and PASS_OUT are all high; or (2) $PASS\_IN_{[1:0]}$ are high and match compare output is low (failing the match test).

- **Bits 9 through 8** select one of four tests for the Match Compare unit (Table 3.20).

Table 3.20 Match test encoding

$CCR_{[9:8]}$	Test condition
00	Always pass
01	Never pass
10	Pass if NEW == OLD
11	Pass if NEW != OLD

- **Bits 2 through 0** select one of eight tests for the Magnitude Compare unit (Table 3.21).

Table 3.21 Magnitude test encoding

$CCR_{[2:0]}$	Test condition
000	Always pass
001	Pass if NEW > OLD
010	Pass if NEW == OLD
011	Pass if NEW >= OLD
100	Never pass
101	Pass if NEW <= OLD
110	Pass if NEW != OLD
111	Pass if NEW < OLD

During a Stateful Data Write to the Pixel Buffer, the pixel data is actually written only when the Magnitude test, the Match test, and the external PASS_IN pin all pass. The PASS_OUT pin is set to pass only when the Magnitude test and the Match test both pass.

While the Picking Logic is enabled, all Stateful Data Writes which pass both compare tests (PASS_OUT high) while PASS_IN is high will set (i.e., OR a “1” into) the HIT flag. The HIT flag will then remain set until cleared by writing “10” into bits 25 and 24. The HIT flag is active high, and when it is “1”, it drives the open-drain $\overline{HIT}$ pin low.

This register resets to 0A00 0000h, which means the HIT flag is cleared and the Picking Logic is disabled.

During a Stateless Data Write access, the Dual Compare unit behaves as if this register were set to 0000 0000h, regardless of its actual value.

Write Address Control Register ($WAC_{[31:0]}$)

Only 1 bit of this register is currently used for the Pixel ALU function. The other 31 bits are reserved.

- **Bit 0** selects the source for the Pixel Buffer write address. “0” selects the Pixel Buffer write address from the $PALU\_A_{[5:0]}$ pins. “1” selects the Pixel Buffer write address from the $PALU\_DQ_{[29:24]}$ pins.
- **Bits 31 through 1** are reserved. They are written as “0” for future compatibility.

This register resets to 0000 0000h.

During a Stateless Data Write, the Write Address Control register behaves as if this register were set to 0000 0000h, regardless of its actual value.

An Application of the Write Address Control Register

The Write Address Control register is used to speed up vertical scroll in screen display. Taking advantage of the pipeline structure, reading data from one Pixel Buffer location and writing into another location can be achieved in one Stateful Data Write.

The 1-bit Write Address Control register selects the Pixel Buffer write address between the $PALU\_A_{[5:0]}$ pins (the normal path) and the $PALU\_DQ_{[29:24]}$ pins (the vertical scroll acceleration path).

For the vertical scroll in a screen as illustrated in Figure 3.20, the data in Pixel A is to be moved to Pixel B. Assume that Pixel data A is stored in Pixel Buffer [Block 3:Word 0] and that Pixel B is in [Block 0:Word 5]. Figure 3.21 shows the pipeline flow for the write address selection, and Figure 3.12 shows the data stream for the example in Figure 3.20. Before the Stateful Data Write to move Pixel A data to Pixel B location is started, four registers should be set as follows:

- Write into Write Address Control Register with “1”.
- Write into ROP/Blend Control Register with 0505 0505h to select old data.

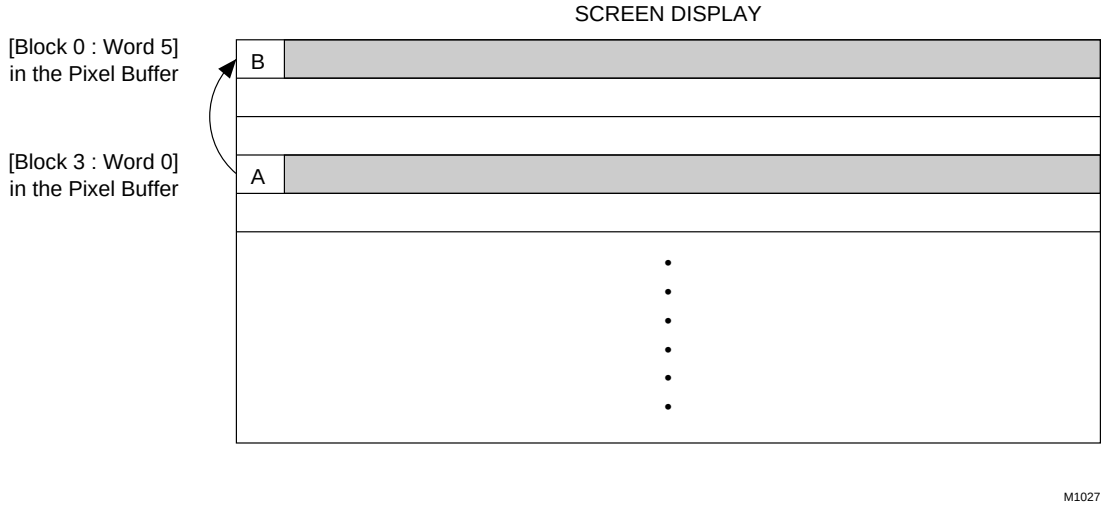


Figure 3.20 Pixel movement in vertical scroll

- Write into Compare Control Register with 0000 0000h to pass data into the the Pixel Buffer.
- Write into Plane Mask Register with FFFF FFFFh to pass every bit into the Pixel Buffer for the Stateful Data Write.

The Stateful Data Write issued later should have the read address asserted on the PALU_A[5:0] pins and the write address on the PALU_DQ[29:24] pins at the next cycle. Seven cycles later in the pipeline path, the Pixel A data will be written into the Pixel B location.

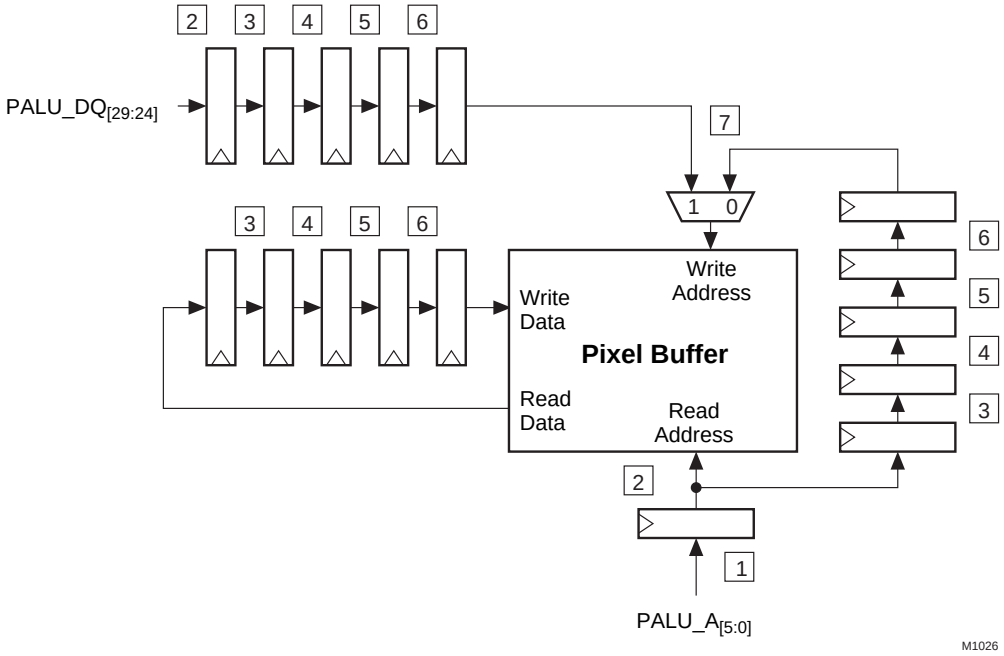


Figure 3.21 Pipeline flow of the Write Address Control

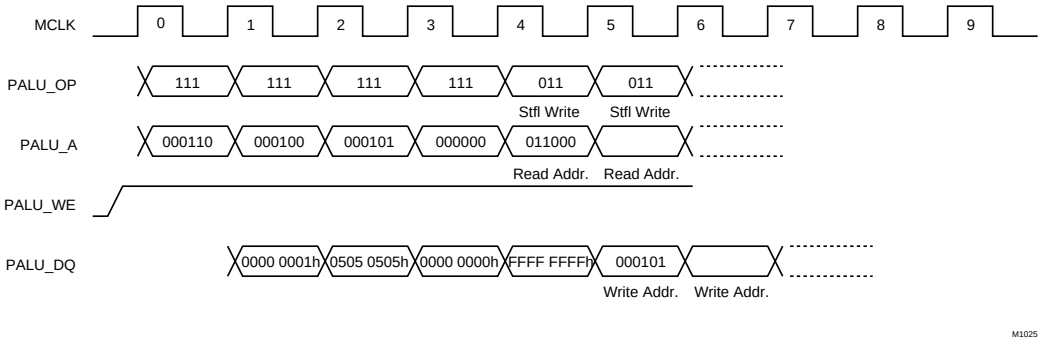


Figure 3.22 Pipeline for performing a vertical scroll

Blend_2 Control Register(BLD2$_{[31:0]}$) (NEW)

This register provides additional control of the multiplicands and addends for the four Blend units. Each Blend unit is independently controlled by a 4-bit field of this 32-bit register. Bits 3 through 0 are repeated three more times on byte boundaries for Units 1, 2, and 3. That is, bits 11 through 8 for unit 1; bits 19 through 16 for unit 2; and bits 27 through 24 for unit 3. In addition, Bits 29 and 28 are used to select the output of the Alpha-Saturate block. This output can then be selected by each of the ROP/Blend units as the source for the second multiplicand, MULTP2. The data format of this register is explained below.

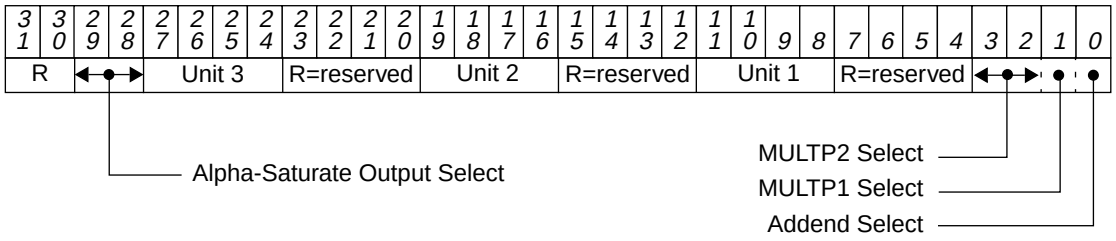


Figure 3.23 Blend_2 Control register data format

- **Bits 31 through 30, 23 through 20, 15 through 12, and 7 through 4** are reserved. They shall be written as “0” for future compatibility.
- **Bits 29 through 28** determine the output of the Alpha-Saturate block (Table 3.22). This output can then be selected by any of the four units as the source of the second multiplicand, MULTP2, using BLD2$_{[8n+3:8n+2]}$.

Table 3.22 Encoding for Output of Alpha-Saturate Block

BLD2$_{[29:28]}$	Output of Alpha-Saturate Block
00	min{NEW$_{[31:24]}$, ~OLD$_{[31:24]}$} (α-sat)
01	NEW$_{[31:24]}$(As)
10	OLD$_{[31:24]}$ (Ad)
11	~OLD$_{[31:24]}$ (~Ad)

- **Bits 8n+3 through 8n+2** select the source for the second multiplicand, MULTP2 (Table 3.23)

Table 3.23 Encoding for MULTP2 Source Selection

BLD2$_{[8n+3:8n+2]}$	Multiplicant 2
00	OLD$_{[8n+7:8n]}$
01	~OLD$_{[8n+7:8n]}$
1X	Data selected by bits [29:28] of this register

- **Bit 8n+1** selects the source for the first multiplicand, MULTP1. If this bit is “0”, the data selected by Bits [8n+7:8n+6] of the ROP/Blend Control Register is used; if this bit is “1”, the OLD$_{[8n+7:8n]}$ data is used.
- **Bit 8n** selects the source for the ADDEND. If this bit is “0”, the data selected by Bit [8n+5] of the ROP/Blend Control Register is used; if this bit is “1”, the OLD$_{[8n+7:8n]}$ data is used.

This register resets to 0000 0000h. This value causes the ROP/Blend units to operate based on the settings in the ROP/Blend Control Register, which assures compatibility with previous generations of the 3D-RAM.

During a Stateless Data Write access, the ROP/Blend units behave as if this register were set to 0000 0000h regardless of its actual value.

Preblend Control Register(PBC$_{[31:0]}$) (NEW)

This register controls the first cycle, known as the Preblend Cycle, of the two-cycle Loop Back operation for the four ROP/Blend units by selecting MULTP2 for the Preblend Cycle and the Addend for the second, or Normal, cycle (see the section on “Pixel ALU Blend Modes” on page 32). MULTP1 and Addend are fixed to PALU_DQ$_{[8n+7:8n]}$ for the Preblend Cycle. Each ROP/Blend unit is independently controlled by an 8-bit field of this 32-bit register. Bits 3 through 0 are repeated three more times on byte boundaries for Units 1, 2, and 3. That is, bits 11 through 8 for unit 1; bits 19 through 16 for unit 2; and bits 27 through 24 for unit 3. However, since there is only one Alpha-Saturate block (located in ROP/Blend Unit 3), the selection made by Bits 29 and 28 applies to all four Blend units.

The data format of this register is explained below.

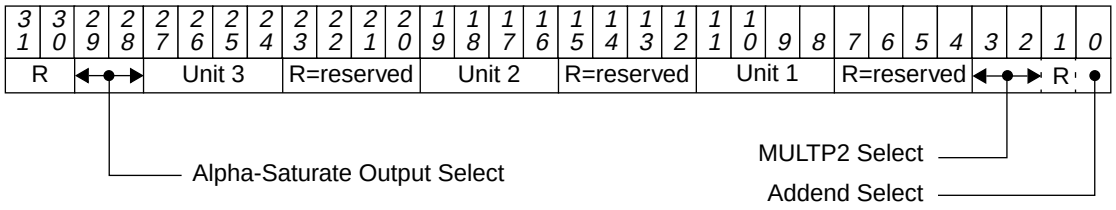


Figure 3.24 Preblend Control register data format

- **Bits 31, 30, 25, 23 through 20, 17, 15 through 12, 9, 7 through 4, and 1** are reserved. They shall be written as “0” for future compatibility.
- **Bits 29 through 28** determine the output of the Alpha-Saturate block (Table 3.24). This output can then be selected by any of the four units as the source of the second multiplicand, MULTP2, using PCR$_{[8n+3:8n+2]}$.

Table 3.24 Encoding for Output of Alpha-Saturate Block

PCR$_{[29:28]}$	**Output of Alpha-Saturate Block**
00	min{NEW$_{[31:24]}$, ~OLD$_{[31:24]}$} (α-sat)
01	NEW$_{[31:24]}$ (As)
10	OLD$_{[31:24]}$ (Ad)
11	~OLD$_{[31:24]}$ (~Ad)

- **Bits 8n+3 through 8n+2** select the source for the second multiplicand, MULTP2, for the Preblend Cycle(Table 3.25)

Table 3.25 Encoding for MULTP2 Source Selection for Preblend Cycle

$PCR_{[8n+3:8n+2]}$	**MULTP2 Source for Preblend Cycle**
00	$OLD_{[8n+7:8n]}$
01	$\sim OLD_{[8n+7:8n]}$
1X	Data selected by bits [29:28] of this register

- **Bit 8n** selects the source for the Addend for the second, or Normal cycle. If this bit is “0”, the multiplier output from the Preblend Cycle is “looped back” to become the Addend for the Normal Cycle; if this bit is “1”, the Addend from the Preblend Cycle is “looped back” to become the Addend for the Normal Cycle.

This register resets to 0000 0000h. This value causes the ROP/Blend units to operate based on the settings in the ROP/Blend Control Register, which assures compatibility with previous generations of the 3D-RAM.

During a Stateless Data Write access, the ROP/Blend units behave as if this register were set to 0000 0000h regardless of its actual value.

Stencil Planes Register(StPl$_{[31:0]}$) (NEW)

This register defines the bits allocated for stencil planes in OpenGL stencil mode and the value masking of these stencil planes when the selected stencil comparison function is performed against the stencil reference value. The on-chip stencil hardware acceleration features are implemented in ROP/Blend unit 3. Therefore, only bits 31 through 24 of the 32-bit ALU unit are available for use as stencil planes. Any number of bits from 0 through 8 may be allocated as stencil planes. In order for the GL_INCREMENT and GL_DECREMENT functions to work properly, it is necessary to keep the stencil planes in one contiguous group.

31	*30*	*29*	*28*	*27*	*26*	*25*	*24*	*23*	*22*	*21*	*20*	*19*	*18*	*17*	*16*	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
$St.Enable_{[7:0]}$								*$StF.MASK_{[7:0]}$*								*reserved*															

Figure 3.25 Stencil Planes register data format

- **Bits 31 through 24 = $St.Enable_{[7:0]}$** provide a bitwise selection of which bits are allocated as stencil planes in OpenGL mode. Bits that are enabled are subject to the controls of the Stencil Control register. Bits that are disabled are available for use by ROP Unit 3 or the Dual Compare unit. For each bit, a “0” disables that bit for OpenGL stencil mode. A “1” enables that bit for OpenGL stencil mode. Again, for the INCREMENT and DECREMENT stencil functions to work properly, it is necessary that the enabled stencil planes be in one contiguous group. For example, if $St.Enable_{[7:0]}$ is set to the value 0011 1100, then bits 29 through 26 of the ALU unit are used for stencil planes and bits 31, 30, 25, and 24 may be used for ROP or Compare functions. Note that the blending function in ROP/Blend Unit 3 cannot be used if any bits in this unit are used for OpenGL stencil.
- **Bits 23 through 16 = $StF.MASK_{[7:0]}$** provide a bitwise stencil value mask for stencil comparison functions. This mask

maps directly to bits 31 through 24 of the 32-bit bus. That is, bit 23 of this register will mask/unmask bit 31 of the ALU bus; bit 22 will mask/unmask bit 30 of the ALU bus, and so on. For each bit, a “0” will cause the corresponding bit to be ignored in stencil comparison functions. A “1” will cause the corresponding bit to be used in the stencil comparison functions. For convenience and clarity, the mnemonic used here corresponds to the OpenGL terminology. Specifically, “StF” refers to the OpenGL command “glStencilFunc” with “MASK” referring to the parameter “*mask*” of this command. Thus, this bit field also corresponds to the symbolic parameter “GL_STENCIL_VALUE_MASK”.

- **Bits 15 through 0** are reserved. They shall be written as “0” for future compatibility.

This register resets to 00FF 0000h, which means that all stencil planes are disabled.

During a Stateless Data Write access, the Pixel ALU behaves as if this register were set to 00FF 0000h regardless of its actual value.

Stencil Control Register(StC$_{[31:0]}$) (NEW)

This register defines the stencil comparison function and the stencil operations that will be performed for the OpenGL stencil mode. Only 12 bits of this register are currently used for stencil hardware acceleration, and the other 20 bits are reserved. For convenience and clarity, the mnemonics used here correspond to the OpenGL terminology. Specifically, “StOP” refers to the OpenGL command “glStencilOp” with “ZPASS”, “ZFAIL”, and FAIL” referring to the parameters of this command; “StF” refers to “glStencilFunc” with “FUNC” referring to the parameter “*func*” of this command. Also, it is important to note that the bits programmed in this register are effective only in the case when both PASS_IN pins are “1” and the Match Compare passes, since otherwise no data will be written into the Pixel Buffer and the PASS_OUT pin will be “0”.

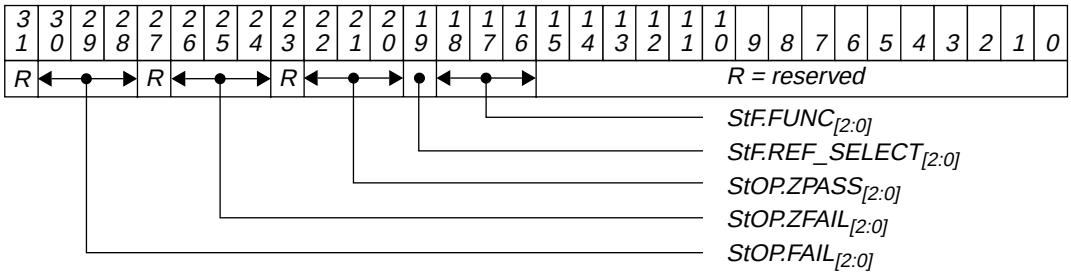


Figure 3.26 Stencil Control register data format

- **Bits 31, 27, 23, 19, and 15 through 0** are reserved. They shall be written as “0” for future compatibility.
- **Bits 30 through 28 = StOP.FAIL$_{[2:0]}$** define the stencil operation to be executed in the case of GL_STENCIL_FAIL. That is, these bits determine which one of the stencil operations listed in Table 3.26 will be performed when the Stencil Compare function fails. Disabled bits keep their OLD data. In the GL_INCR operation, the maximum value is defined as 2^n-1, where n=the number of bits enabled by St.Enable$_{[7:0]}$.

Table 3.26 Stencil Operation for StOP.FAIL

$StOP.FAIL_{[2:0]}$	Stencil Operation	Definition
000	GL_ZERO	Enabled bits cleared to zero
001	GL_KEEP	Enabled bits remain OLD data
010	GL_INVERT	Enabled bits are inverted OLD
011	GL_REPLACE	Enabled bits are replaced by StF.REF
100 110	GL_INCR	Enabled bits are incremented by 1 (clamped to max. value)
101 111	GL_DECR	Enabled bits are decremented by 1 (clamped to zero value)

- **Bits 26 through 24 = $StOP.ZFAIL_{[2:0]}$** define the stencil operation to be executed in the case of GL_STENCIL_PASS_DEPTH_FAIL. That is, these bits determine which one of the stencil operations listed in Table 3.27 will be performed when the Stencil Compare function passes, but the MAGNITUDE Compare function fails. Disabled bits keep their OLD data In the GL_INCR operation, the maximum value is defined as 2^n-1, where n=the number of bits enabled by $St.Enable_{[7:0]}$.

Table 3.27 Stencil Operation for StOP.ZFAIL

$StOP.ZFAIL_{[2:0]}$	Stencil Operation	Definition
000	GL_ZERO	Enabled bits cleared to zero
001	GL_KEEP	Enabled bits remain OLD data
010	GL_INVERT	Enabled bits are inverted OLD
011	GL_REPLACE	Enabled bits are replaced by StF.REF
100 110	GL_INCR	Enabled bits are incremented by 1 (clamped to max. value)
101 111	GL_DECR	Enabled bits are decremented by 1 (clamped to zero value)

- **Bits 22 through 20 = $StOP.ZPASS_{[2:0]}$** define the stencil operation to be executed for the case of GL_STENCIL_PASS_DEPTH_PASS. That is, these bits determine which one of the stencil operations listed in Table 3.28 will be performed when the MAGNITUDE compare and Stencil compare functions both pass. Disabled bits use the ROP unit results. In the GL_INCR operation, the maximum value is defined as 2^n-1, where n=the number of bits enabled by $St.Enable_{[7:0]}$.

Table 3.28 Stencil Operation for StOP.ZPASS

$StOP.ZPASS_{[2:0]}$	Stencil Operation	Definition
000	GL_ZERO	Enabled bits cleared to zero
001	GL_KEEP	Enabled bits remain OLD data
010	GL_INVERT	Enabled bits are inverted OLD
011	GL_REPLACE	Enabled bits are replaced by StF.REF
100 110	GL_INCR	Enabled bits are incremented by 1 (clamped to max. value)
101 111	GL_DECR	Enabled bits are decremented by 1 (clamped to zero value)

- **Bits 19 = StF.REF_SELECT** defines the source of the StF.REF stencil data. Setting this bit to 0 selects the StF.REF from the pins $PALU\_DQ_{[31:24]}$; setting this bit to 1, the bits 31 through 24 in the Constant Source register will be used as the StF.REF.
- **Bits 18 through 16 = $StF.FUNC_{[2:0]}$** define the stencil comparison function. The StF.REF stencil data (from the pins $PALU\_DQ_{[31:24]}$) is compared with the OLD stencil data based on the settings of these register bits. The mnemonic “StF” refers to the OpenGL command “glStencilFunc” with “REF” referring to the parameter “*ref*” of this command. Table 3.29 defines which test will be used for the stencil comparison.

This register resets to 3330 0000h, which means that the stencil test always passes.

During a Stateless Data Write access, the Pixel ALU behaves as if this register were set to 3330 0000h regardless of its actual value.

Table 3.29 Stencil Comparison Functions

$StF.FUNC_{[2:0]}$	Stencil Test	Definition
000	GL_ALWAYS	Always pass stencil test
001	GL_GREATER	Pass stencil test if (StF.REF && StF.Mask) > (OLD && StF.Mask)
010	GL_EQUAL	Pass stencil test if (StF.REF && StF.Mask) == (OLD && StF.Mask)
011	GL_GEQUAL	Pass stencil test if (StF.REF && StF.Mask) >= (OLD && StF.Mask)
100	GL_NEVER	Always fail stencil test
101	GL_LEQUAL	Pass stencil test if (StF.REF && StF.Mask) <= (OLD && StF.Mask)
110	GL_NOTEQUAL	Pass stencil test if (StF.REF && StF.Mask) != (OLD && StF.Mask)
111	GL_LESS	Pass stencil test if (StF.REF && StF.Mask) < (OLD && StF.Mask)

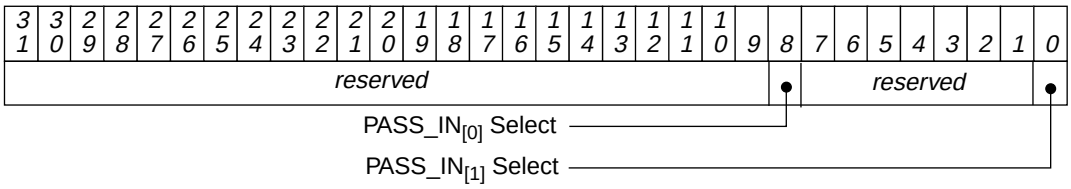


Figure 3.27 PASS_IN Select register data format

PASS_INs Select Register (PINS[31:0]) (NEW)

Only 2 bits of this register are currently used for the Pixel ALU function, and the other 30 bits are reserved.

- **Bit 0** enables the $PASS\_IN_{[1]}$ pin to participate in the internal write enable logic of the Pixel Buffer. Setting this bit to “0” disables the $PASS\_IN_{[1]}$ which is then internally set to “1” and has no effect on the operation of the Pixel ALU. Setting this bit to “1” enables and passes through the $PASS\_IN_{[1]}$ signal to be ANDed with the $PASS\_IN_{[0]}$ signal. See also Figure 3.13 for illustration.
- **Bit 8** enables the $PASS\_IN_{[0]}$ pin to participate in the internal write enable logic of the Pixel Buffer. Setting this bit to “0” disables the $PASS\_IN_{[0]}$ which is then internally set to “1” and has no effect on the operation of the Pixel ALU. Setting this bit to “1” enables and passes through the $PASS\_IN_{[0]}$ signal to be ANDed with the $PASS\_IN_{[1]}$ signal. See also Figure 3.13 for illustration.
- **Bits 31 through 1** are reserved. They shall be written as “0” for future compatibility.

This register resets to 0000 0100h. This value assures compatibility with previous generations of the 3D-RAM.

Color Depth Select Register(CDS[31:0]) (NEW)

Only 1 bit of this register is currently used for the Pixel ALU function, and the other 31 bits are reserved.

- **Bit 0** selects the color depth for Pixel ALU operations. “0” selects the normal (8,8,8,8) 32-bit blending mode. Also, with this setting, color data can be stored in the (5,6,5,0) 16-bit mode. “1” selects the (4,4,4,4) 16-bit blending mode. For details on the 16-bit color modes, refer to the section titled “16-bit Color Modes” in this chapter.
- **Bits 31 through 1** are reserved. They shall be written as “0” for future compatibility.

Note that an ALU NOP must be inserted between (a) the write to the Color Depth Select register and (b) the following ALU operations: Read Pixel Buffer, Stateful Initial Data Write, Stateful Normal Data Write, and Initiate Two-Cycle Blending.

This register resets to 0000 0000h. This value assures compatibility with previous generations of the 3D-RAM.

Note During a Stateless Data Write, the Color Depth Select register still behaves based on its current state. In other words, if bit 0 of the register is set to “1”, the Stateless Write will be done in the (4,4,4,4) mode of operation.

Pixel Data Operations

There are six pixel data operations: Stateless Initial Data Write, Stateless Normal Data Write, Stateful Initial Data Write, Stateful Normal Data Write, Replace Dirty Tag, and OR Dirty Tag. Simply put, Stateless Data Writes refer to the condition that the states of the Pixel ALU units are entirely ignored and that the write data is passed to the Pixel Buffer unaffected, whereas in Stateful Data Writes, the settings of the various registers in the Pixel ALU, the results of the compare tests, and the state of the PASS_IN pin all affect whether the bits of the pixel data will be written into the Pixel Buffer. Initial and Normal Data Writes refer to the manner in which the Dirty Tag is updated. In an Initial Data Write, the bits of the Dirty Tag are selectively set and cleared. In a Normal Data Write, the bits of the Dirty Tag associated with the addressed block and word are inclusive ORed with the PALU_BE pins, and the other bits of the Dirty Tag are unchanged. The following sections describe these operations in details.

Stateless Initial Data Write

The Stateless Initial Data Write operation writes 32-bit data to the addressed block and word in the Pixel Buffer. No register values affect this operation.

The ROP/Blend units simply pass the write data through without affecting it. The Dual Compare unit is ignored and does not inhibit the writing of data to Pixel Buffer. The PASS_OUT pin is forced to “1” for this operation. The PASS_IN pin has no effect.

The corresponding four Dirty Tag bits for the addressed word are set to the respective $PALU\_BE_{[3:0]}$ value of the 32-bit data. The other 28 Dirty Tag bits corresponding to the addressed block are cleared to “0”.

Stateless Normal Data Write

The Stateless Normal Data Write operation writes 32-bit data to the addressed block and word in the Pixel Buffer. No register values affect this operation.

The ROP/Blend units simply pass the write data through without affecting it. The Dual Compare unit is ignored and does not inhibit the writing of data to Pixel Buffer. The PASS_OUT pin is forced to “1” for this operation. The PASS_IN pin has no effect.

The four Dirty Tag bits corresponding to the addressed block and word are inclusive ORed with the $PALU\_BE_{[3:0]}$ pin. The other 28 Dirty Tag bits corresponding to the addressed block are unchanged.

Stateful Initial Data Write

The Stateful Initial Data Write operation writes 32-bit data to the addressed block and word. The new data may be combined with the existing destination data. The conditional write enable applies. All register values can affect this operation.

The four Dirty Tag bits corresponding to the addressed block and word are set to the $PALU\_BE_{[3:0]}$ value. The other 28 Dirty Tag bits corresponding to the addressed block are cleared to “0”.

Both the writing to the Pixel Buffer and the updating of the Dirty Tag can be inhibited by a compare test failure (which means that either PASS_IN or PASS_OUT is low).

Stateful Normal Data Write

The Stateful Normal Data Write operation writes 32-bit data to the addressed block and word. The new data may be combined with the existing destination data. The conditional write enable applies. All register values can affect this operation.

The four Dirty Tag bits corresponding to the addressed block and word are inclusive ORed with the $PALU\_BE_{[3:0]}$ value. The other 28 Dirty Tag bits corresponding to the addressed block are unchanged.

Both the writing to the Pixel Buffer and the updating of the Dirty Tag can be inhibited by a compare test failure (which means that either PASS_IN or PASS_OUT is low).

Replace Dirty Tag

The 32-bit data on the $PALU\_DQ_{[31:0]}$ pins replaces the Dirty Tag of the addressed block. The bit mapping between the Dirty Tag and PALU_DQ pins is explained on pages 22 and 42. The $PALU\_BE_{[3:0]}$ pins determine which byte of the $PALU\_DQ_{[31:0]}$ data gets written into the Dirty Tag RAM. The Dirty Tag data passes through the ROP portion of the ROP/Blend units. All of the registers behave the same way they would during a Stateless Data Write.

OR Dirty Tag

The 32-bit data on the $PALU\_DQ_{[31:0]}$ pins is inclusive ORed with the Dirty Tag of the addressed block. The bit mapping between the Dirty Tag and PALU_DQ pins is explained on pages 22 and 42. The $PALU\_BE_{[3:0]}$ pins determine which byte of the $PALU\_DQ_{[31:0]}$ data gets written into the Dirty Tag RAM. The Dirty Tag data passes through the ROP portion of the ROP/Blend units. All of the registers behave the same way they would during a Stateless Data Write.

Initiate Two-Cycle Blending

This operation initiates a Preblend Cycle, which is the first cycle of the Two-Cycle Blend operation. The Preblend Cycle is similar to a Stateful Write, except that the Preblend Cycle does not actually write the data back to the Pixel Buffer or affect the Dirty Tags in any way. When this operation is issued, the ROP/Blend units begin blending the data based on the settings of three registers: ROP/Blend Control, Blend_2 Control, and Preblend Control. After the multiplier stage of the Preblend Cycle, the multiplier output and the Addend are looped back as possible addends for the next cycle, which is called the Normal Cycle. The Preblend Cycle must always be followed by a Stateful Initial/Normal Write with the same Pixel Buffer Address on the PALU_A pins. This operation is only for blending. The ROP/Blend Control register must be set to perform blending for all ROP/Blend units or this operation will not function correctly. See the paragraphs on “Pixel ALU Blend Modes” starting on page 32 for further details.

4 DRAM Operations

DRAM Operations

This chapter discusses the 3D-RAM operations involving the DRAM arrays. These include the data transfers between a DRAM bank and the Pixel Buffer and between a DRAM bank and a Video Buffer.

An Overview of DRAM Operations

Depending on the DRAM_OP code, the DRAM_A address pins may be interpreted in three different ways: (1) page access (for Access Page and Duplicate Page operations), (2) block access (for Read Block, Unmasked Write Block, and Masked Write Block operations), and (3) scan line access (for Video Transfer operation).

A page access selects one page out of 257 pages (256 normal pages plus one extra page). DRAM_A8 is used to select the extra page—"1" is for the extra page, "0" is for choosing one of the 256 normal pages from a given bank. When DRAM_A8 is equal to "1", the lower eight address pins $DRAM\_A_{[7:0]}$ should still be driven to stable states although they are not decoded internally.

The position and orientation of all pages displayed on the screen are controlled by the user. However, the mapping of data within a given page to a Pixel Buffer block is fixed and is shown in Figure 4.1. (More precisely from the perspective of DRAM operations, we should speak of the mapping between the sense amplifiers of a selected DRAM bank with a Pixel Buffer block. However, since the page-wide sense amplifiers act as a direct-mapped write-through pixel cache for a DRAM bank, the mapping between the sense amplifiers of a DRAM bank and the Pixel Buffer is the same as the mapping between a DRAM page and the Pixel Buffer. For convenience, the latter reference is used liberally in this document.) A DRAM page is always organized as 10 blocks wide and 4 blocks high; this is fixed. A block always contains eight 32-bit words, for a total of 256 bits. In the case of 8 bits per pixel, the eight words in a given block may be viewed as 8 pixels wide by 4 pixels high. Thus, a DRAM page would be mapped to the screen as 80 pixels wide by 16 pixels high with 8 bits per pixel. In the case of 32 bits per pixel, the eight words in a given block may be viewed as 2 pixels wide by 4 pixels high. Thus, a DRAM page would be mapped to the screen as 20 pixels wide by 16 pixels with 32 bits per pixel. For simplicity, we represent these frame buffer organizations with the short hand notations 80W x 16H x 8 and 20W x 16H x 32, respectively. Several frame buffer organization examples are shown in Chapter 6.

During a data transfer between a DRAM page and the Pixel Buffer, both the block location in the Pixel Buffer and the block location in the DRAM page must be specified. In the Pixel Buffer, the selection of one of eight blocks is through the $DRAM\_A_{[8:6]}$ pins. In the height direction of a DRAM page, the $DRAM\_A_{[1:0]}$ pins select one of four block rows. In the width direction of the page, a block is selected from the ten block columns through the $DRAM\_A_{[5:2]}$ pins. Figure 4.1 illustrates the addressing scheme for block transfer with a block configured as 8W x 4H x 8. The hexadecimal number written on every block of the DRAM page corresponds to the six address pins $DRAM\_A_{[5:0]}$. Similarly, the number on the Pixel Buffer block is from the other address bits $DRAM\_A_{[8:6]}$.

Description of DRAM Operations

Table 4.1 DRAM operation encoding

Operation	DRAM_OP	DRAM_BS	DRAM_A
Unmasked Write Block (UWB)	000	Bank	Pixel Buffer Block(3 pins), DRAM Block(6 pins)
Masked Write Block (MWB)	001	Bank	Pixel Buffer Block(3 pins), DRAM Block(6 pins)
Precharge Bank (PRE)	010	Bank	—
Video Transfer (VDX)	011	Bank	Control (2pins), Line (4pins)
Duplicate Page (DUP)	100	Bank	Page (9 pins)
Read Block (RDB)	101	Bank	Pixel Buffer Block(3 pins), DRAM Block(6 pins)
Access Page (ACP)	110	Bank	Page (9 pins)
No Operation (NOP)	111	—	—

Table 4.1 lists all of the DRAM operations. One operation can be launched in every cycle. However, the sequence of these DRAM operations is bounded by the resource interlocks. The Access Page can only be issued after Precharge Bank, and the only operation after Precharge Bank is Access Page. Tables 7.6 and 7.7 contain the specific timing interlocks for DRAM operations in the same bank and between different banks.

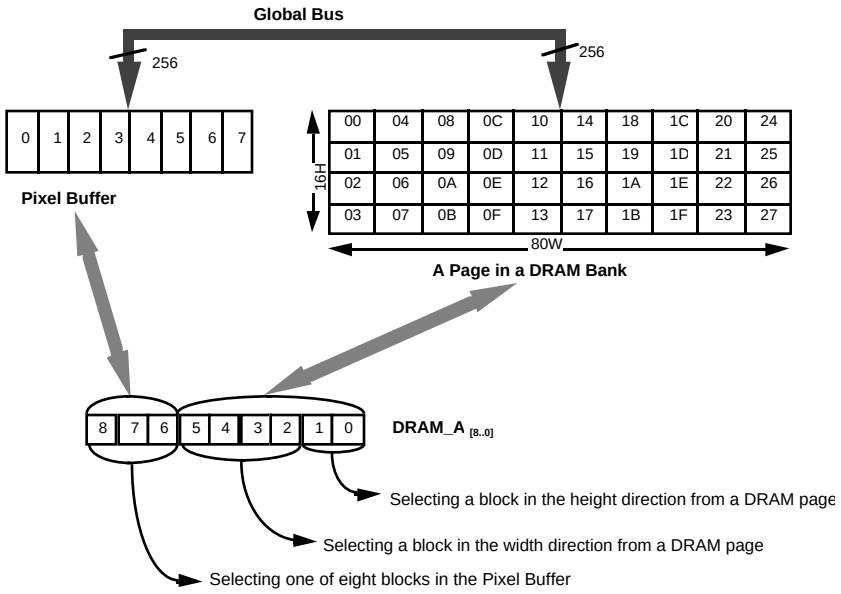


Figure 4.1 Addressing scheme for block transfer on the Global Bus, for a block size of 8W x 4H x 8 (or 2W x 4H x 32). The blocks in the DRAM page are numbered with hexadecimal values and selected by $DRAM\_A_{[5:0]}$.

Unmasked Write Block (UWB)

The UWB operation copies 32 bytes from the specified Pixel Buffer block over the Global Bus to the specified block in the sense amplifiers and the DRAM page of a selected DRAM bank. The $DRAM\_A_{[5:0]}$ pins select one of the 40 blocks in a DRAM page. The DRAM_A[8:6] pins select one of the eight Pixel Buffer blocks. The 32-bit Plane Mask register has no effect on Unmasked Write Block operation. The 32-bit Dirty Tag still controls which bytes of the block are updated.

Masked Write Block (MWB)

The MWB operation copies 32 bytes from the specified Pixel Buffer block over the Global Bus to the specified block in the sense amplifier and the DRAM page of a selected DRAM bank. The $DRAM\_A_{[5:0]}$ pins select one of the 40 blocks in a DRAM page. The $DRAM\_A_{[8:6]}$ pins select one of the eight Pixel Buffer blocks. Both the 32-bit Dirty Tag and the 32-bit Plane Mask register control which bytes of the block are updated.

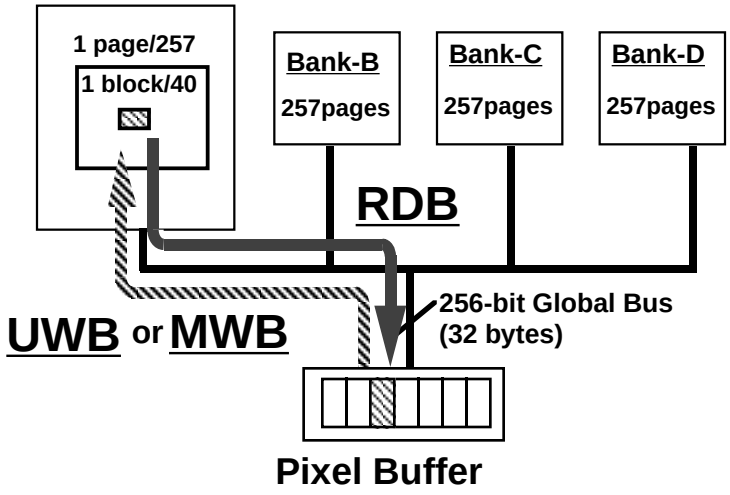


Figure 4.2 Unmasked Write Block, Masked Write Block, and Read Block on the Global Bus

Precharge Bank (PRE)

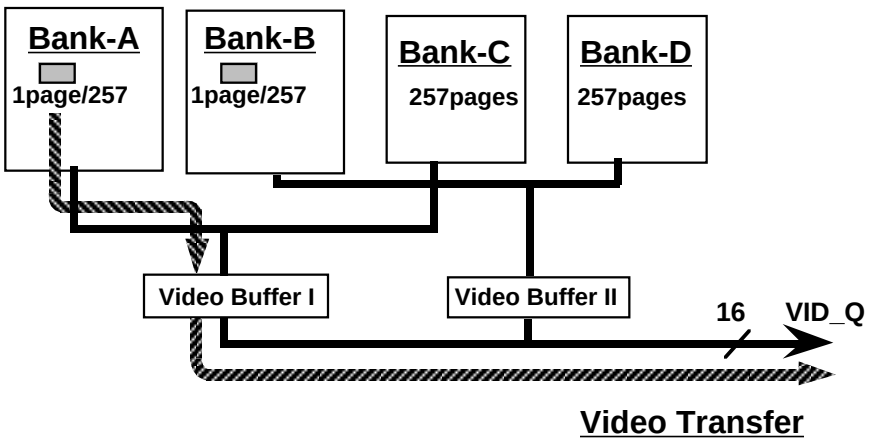
The PRE operation first deactivates the word line corresponding to the most recently accessed DRAM page of a selected DRAM bank and then equalizes the bit lines of the sense amplifiers for a subsequent Access Page operation. After a Precharge Bank operation has been performed on a certain DRAM bank, the operations that can be performed on that DRAM bank are Access Page, Precharge Bank, and NOP. Other operations after a Precharge Bank operation are illegal, and the resulting data is undefined.

Video Transfer (VDX)

There are two parts to the VDX operation: video buffer load and video output. Video Buffer load relates to the transfer from the sense amplifiers of a selected DRAM bank to a corresponding Video Buffer. Video output relates to the transfer from a Video Buffer to the VID_Q pins.

Video Buffer Load

There are two video buffers available for interleave transfer. Video Buffer I is for Bank A and Bank C. Video Buffer II is for Bank B and Bank D. Figure 4.3 illustrates a Video Transfer example from a page in Bank A to Video Buffer I.



Fiugre 4.3 Video Transfer from a Bank A page to Video Buffer I

This paragraph describes the addressing scheme for the Video Transfer operation in detail. A DRAM page has a fixed organization of 10 blocks wide by 4 blocks high. For VDX operation, a 32-byte block is always considered as being 4 rows high (either 8W x 4H x 8 or 2W x 4H x 32). That is, for VDX operation, a DRAM page is always viewed as containing 16 rows of 80 bytes each. In the case of 8 bits per pixel, the Video Transfer operation transfers a 80W x 1H x 8 line of pixel data from the sense amplifiers of a DRAM page to the corresponding Video Buffer. The DRAM_$A_{[3:0]}$ pins are used to select one of the 16 rows in a DRAM page. Since there are 16 VID_Q pins, one may think of the Video Buffer as 40 double-bytes. The DRAM_$A_{[6:4]}$ pins are ignored in this operation but should still be driven to stable states.

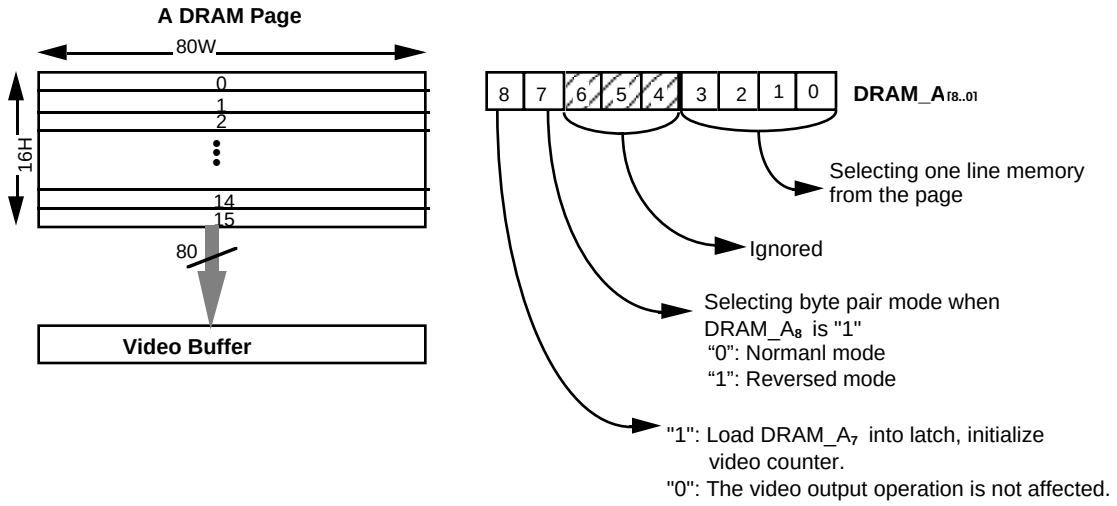


Figure 4.4 Addressing scheme for video transfer

Video Output Operation

There are two byte order formats for the VID_Q video output pins: normal mode and reversed mode. This byte ordering is selected by an internal byte pair mode latch, which is loaded from the DRAM_A7 pin when the DRAM_A8 pin is equal to "1". If the latched data is "0", the normal video output mode is applied to the VID_Q bus. If the latch data is "1", the reversed video output mode is selected.

Since a Video Buffer holds 640 bits, we may number the bytes in the Video Buffer from byte 0 through 79. In both normal and reversed modes, even bytes always appear on the $VID\_Q_{[7:0]}$ pins, and odd bytes on $VID\_Q_{[15:8]}$ pins. In normal mode, the byte data is shifted out to the VID_Q pins in normal sequence as in [byte 0, byte 1] at video clock 0, [byte 2, byte 3] at video clock 1 … [byte 78, byte 79] at video clock 39. However, in many systems, byte 3 contains control bits, while bytes 0, 1, and 2 are the RGB data. Therefore, it may be desirable to make byte 3 available on the first video clock to allow the RAMDAC chip the maximum time to make use of the control bits. The reversed mode is designed for this purpose, where the byte data is shifted in reversed sequence as in [byte 2, byte 3] at video clock 0, [byte 0, byte 1] at video clock 1 … [byte 78, byte 79] at video clock 38, and finally [byte 76, byte 77] at video clock 39. In summary, the 16-bit VID_Q bus output scheme is illustrated in Figure 4.5 for a 80W x 1H x 8 Video Buffer.

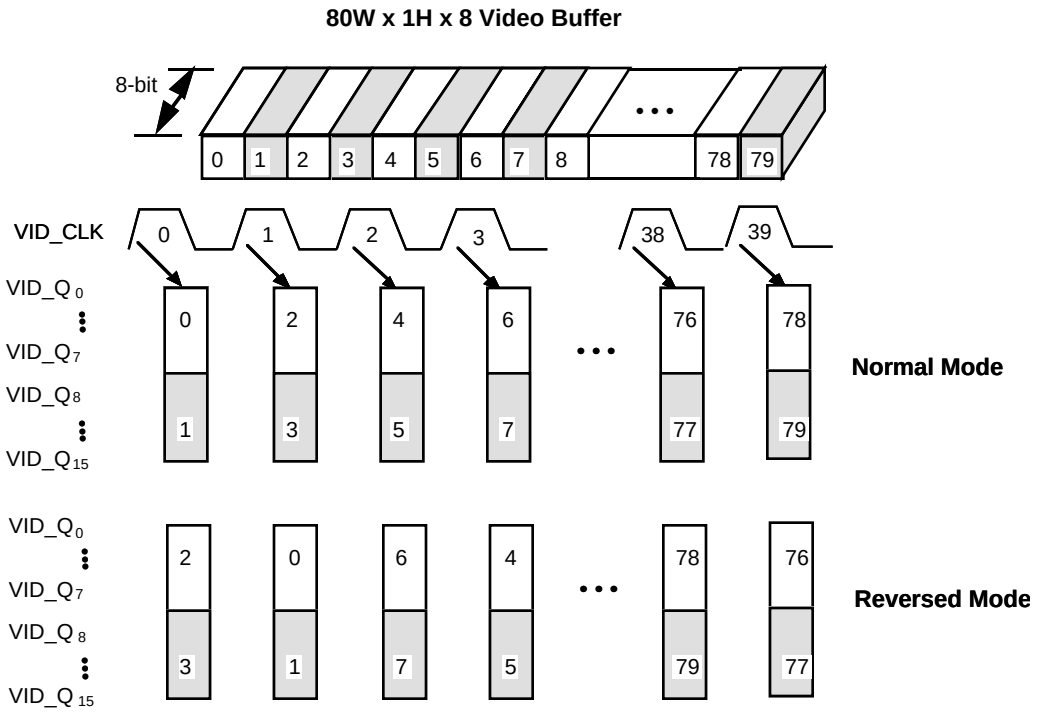


Figure 4.5 16-bit VID_Q bus output scheme for a 80W x 1H x 8 Video Buffer

Initialize and Abort Video Output

When the DRAM_A_8 pin is "1", the byte pair mode latch is loaded, and the current Video Buffer output operation is aborted. The VID_Q bus is driven starting from the Video Buffer indicated by the DRAM_BS_0 pin. Also, the modulo-40 Video Counter is initialized. If DRAM_A_8 is "0", the Video Counter is not affected. The video output from the current Video Buffer continues until this buffer is exhausted. Then, the Video Buffer is automatically switched and the Video Counter is initialized.

To avoid data corruption in the Video Buffer, the user should not start a Video Transfer operation to the Video Buffer that is outputting data to the VID_Q bus.

Figure 8.15 shows an example of initiating a video output process. It begins with commanding a Video Transfer from the DRAM control port, while holding VID_CKE signal low to disable internal VID_CLK until VID_QSF changes. When VID_QSF indicates the specific Video Buffer is ready, the clock enable VID_CKE is asserted to allow video output. For a normal mode video output, Figure 8.16 illustrates an example of continuous video output from both Video Buffers by issuing consecutive Video Transfer operations on the four DRAM banks.

Note that VID_QSF settles from an unknown state to a known state after the initial Video Transfer with DRAM_A_8 = 1. Except for this initial Video Transfer, the clean edge transition on VID_QSF is guaranteed for every occurrence of Video Buffer interleave.

Duplicate Page (DUP)

All 10,240 bits of the data in the sense amplifiers of a selected DRAM bank can be transferred to any specified page in the same bank within one Duplicate Page operation. The data in the sense amplifiers is not affected by this operation. If the DRAM_A_8 pin is 0, then the DRAM_$A_{[7:0]}$ pins select one of the 256 normal pages. If DRAM_A_8 is 1, then the DRAM_$A_{[7:0]}$ pins are ignored and the extra page is written. The Plane Mask register does not apply to this operation.

It may be helpful to point out that it is not necessary to use the DUP operation to write back the data in the sense amplifiers, because they function as a level-two write-through pixel cache. DUP is a special performance function that offers ultra-fast data movement in a frame buffer. Consider the task of clearing the entire frame buffer of 1280 x 1024 x 32. Using only the MWB operations for this task, the 256-bit Global Bus and four bank interleaving plus parallel operations to the four 3D-RAM chips offer very good bandwidth. The data rate is 5.8 GB/s for the -10 grade of 3D-RAM, and the entire screen is cleared in 860 μs, without considering the interruptions of video refresh. However, with the DUP performance function, the data rate increases ten-fold to 58.6 GB/s, and the entire screen is cleared in only 85 μs, with the same -10 grade of 3D RAM.

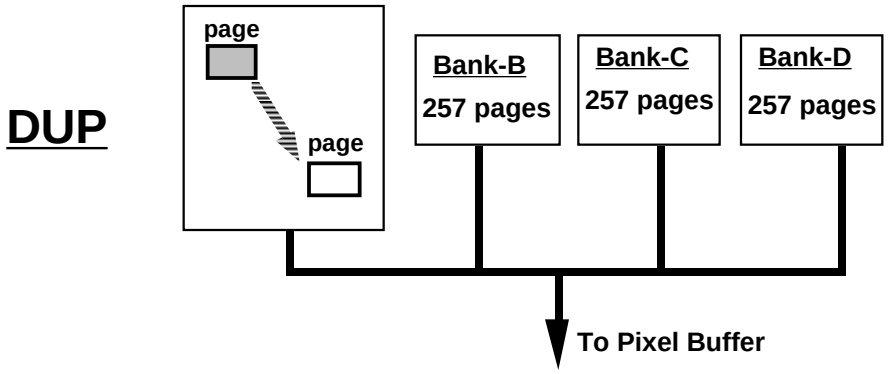


Figure 4.6 Duplicate Page in DRAM Bank A

Read Block (RDB)

The RDB operation copies 32 bytes from the sense amplifiers of a selected DRAM bank over the Global Bus to the specified block in the Pixel Buffer. The corresponding 32-bit Dirty Tag is cleared. The DRAM_$A_{[5:0]}$ pins select one of the 40 blocks in a DRAM page. The DRAM_$A_{[8:6]}$ pins select one of the eight Pixel Buffer blocks. The Read Block operation is also illustrated in Figure 4.2.

Access Page (ACP)

The ACP operation activates the word line corresponding to the specified DRAM page of a selected DRAM bank and transfers the data in the DRAM array to the sense amplifiers. If the DRAM_A8 pin is “0”, then the DRAM_$A_{[7:0]}$ pins select one of the 256 normal pages. If the DRAM_A8 pin is “1”, then the DRAM_$A_{[7:0]}$ pins are ignored and the extra page is transferred.

Before an Access Page operation can be performed on a certain DRAM bank, a Precharge Bank operation must have been performed for that DRAM bank. After an Access Page operation, several DRAM read and write operations, such as UWB, MWB, RDB, DUP, and VDX, may be performed.

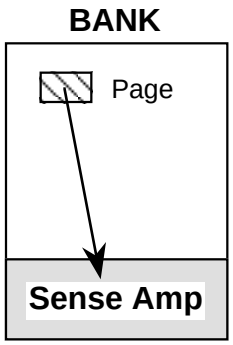


Figure 4.7 Access Page means transferring a specified page to the sense amplifiers.

No Operation (NOP)

The NOP operation may be freely inserted between the ACP operation and the PRE operation on the same bank. NOPs are issued when the DRAM arrays are idle, no read or write is required by the Pixel Buffer, and no Video Buffer load is necessary. More importantly, NOPs are required to satisfy the timing interlocks of the various DRAM operations, as listed in Tables 7.6 and 7.7; for this application, each NOP operation simply takes one clock period.

5 Pixel ALU Pipelines and DRAM Activities

Pixel ALU Pipelines and DRAM Activities

This chapter inculdes some pipeline examples of the interaction between the Global Bus and the Pixel ALU, as well as some typical sequences of DRAM operations on the same bank. For DRAM operations, we assume that the clock cycle time equals to the minimum requirements of the specification—10 ns or 13 ns, depending on the speed grade of the parts. If the 3D-RAM is not running at the minimum MCLK cycle time, then the DRAM operations shown in the tables of this chapter do not govern the cycles of operations. The interlocks listed in Tables 7.6 and 7.7 are always the governing parameters that determine the cycles of DRAM operations. These interlocks specify time durations only and are independent of the clock period and the number of clock cycles, unless specifically noted otherwise.

DRAM and Pixel ALU Interactions

The Global Bus and the Pixel ALU interact as shown in the following tables. A word on the notation may be in order here. Braces are used to enclose more than one specifications that can qualify the operation outside the braces. For example, Stateful {Initial, Normal} Data Write means either Stateful Initial Data Write or Stateful Normal Data Write may be applied. In fact, the table entries shorten this notation to simply Stateful Data Write.

Table 5.1 shows a 2-cycle Read Block operation immediately followed by a Read Data operation that uses data from the Read Block.

Table 5.1 Read Block on Global Bus to Read Data on Pixel ALU

Cycle	DRAM Activities	Pixel ALU Activities
n	Read Block op specified on DRAM_EN, DRAM_OP …	
n+1	Read Block on Global Bus	
n+2	Read Block on Global Bus	Read Data op specified on PALU_EN, PALU_WE, PALU_OP …
n+3		Read Data op specified on PALU_EN, PALU_WE, PALU_OP … Data read from Pixel Buffer
n+4		Data read from Pixel Buffer Data on PALU_DQ
n+5		Data on PALU_DQ

Table 5.2 shows a 3-cycle Read Block operation immediately followed by a Read Data operation that uses data from the Read Block.

Table 5.2 Read Block on Global Bus to Read Data Pixel ALU

Cycle	DRAM Activities	Pixel ALU Activities
n	Read Block op specified on DRAM_EN, DRAM_OP …	
n+1	Read Block on Global Bus	
n+2	Read Block on Global Bus	
n+3	Read Block on Global Bus	Read Data op specified on PALU_EN, PALU_WE, PALU_OP …
n+4		Read Data op specified on PALU_EN, PALU_WE, PALU_OP … Data read from Pixel Buffer
n+5		Data read from Pixel Buffer Data on PALU_DQ
n+6		Data on PALU_DQ

Table 5.3 shows a 2-cycle Read Block operation immediately followed by a Stateful {Initial, Normal} Data Write operation that performs a Read Modify Write on the data from the Read Block.

Table 5.3 Read Block on Global Bus to Stateful {Initial, Normal} Data Write on Pixel ALU

Cycle	DRAM Activities	Pixel ALU Activities
n	Read Block op specified on DRAM_EN, DRAM_OP …	
n+1	Read Block on Global Bus	
n+2	Read Block on Global Bus	Stateful Data Write op specified on PALU_EN, PALU_WE, PALU_OP …
n+3		Old data read from Pixel Buffer New data read from PALU_DQ, PALU_DX
n+4		ROP/Blend 1
n+5		ROP/Blend 2
n+6		ROP/Blend 3
n+7		ROP/Blend 4
n+8		Result data written to Pixel Buffer

Table 5.4 shows a 3-cycle Read Block operation immediately followed by a Stateful {Initial, Normal} Data Write operation that performs a Read Modify Write on the data from the Read Block.

Table 5.4 Read Block on Global Bus to Stateful {Initial, Normal} Data Write on Pixel ALU

Cycle	DRAM Activities	Pixel ALU Activities
n	Read Block op specified on DRAM_EN, DRAM_OP …	
n+1	Read Block on Global Bus	
n+2	Read Block on Global Bus	
n+3	Read Block on Global Bus	Stateful Data Write op specified on PALU_EN, PALU_WE, PALU_OP …
n+4		Old data read from Pixel Buffer New data read from PALU_DQ, PALU_DX
n+5		ROP/Blend 1
n+6		ROP/Blend 2
n+7		ROP/Blend 3
n+8		ROP/Blend 4
n+9		Result data written to Pixel Buffer

Table 5.5 shows a {Stateless, Stateful} {Initial, Normal} Data Write operation immediately followed by a 2-cycle {Masked, Unmasked} Write Block operation.

Table 5.5 {Stateless, Stateful} {Initial, Normal} Data Write on Pixel ALU {Masked, Unmasked} Write Block on Global Bus

Cycle	DRAM Activities	Pixel ALU Activities
n		Data Write op specified on PALU_EN, PALU_WE, PALU_OP …
n+1		Old data read from Pixel Buffer New data on PALU_DQ, PALU_DX
n+2		ROP/Blend 1
n+3		ROP/Blend 2
n+4		ROP/Blend 3
n+5		ROP/Blend 4
n+6	{Masked, Unmasked} Write Block op specified on DRAM_EN, DRAM_OP …	Result data written to Pixel Buffer
n+7	{Masked, Unmasked} Write Block on Global Bus	
n+8	{Masked, Unmasked} Write Block on Global Bus	

Table 5.6 shows a {Stateless, Stateful} {Initial, Normal} Data Write operation immediately followed by a 3-cycle {Masked, Unmasked} Write Block operation.

Table 5.6 {Stateless, Stateful} {Initial, Normal} Data Write on Pixel ALU to {Masked, Unmasked} Write Block on Global Bus

Cycle	DRAM Activities	Pixel ALU Activities
n		Data Write op specified on PALU_EN, PALU_WE, PALU_OP …
n+1		Old data read from Pixel Buffer New data on PALU_DQ, PALU_DX
n+2		ROP/Blend 1
n+3		ROP/Blend 2
n+4		ROP/Blend 3
n+5		ROP/Blend 4
n+6	{Masked, Unmasked} Write Block op specified on DRAM_EN, DRAM_OP …	Result data written to Pixel Buffer
n+7	{Masked, Unmasked} Write Block on Global Bus	
n+8	{Masked, Unmasked} Write Block on Global Bus	
n+9	{Masked, Unmasked} Write Block on Global Bus	

Table 5.7 shows a {Replace, OR} Dirty Tag operation immediately followed by a 2-cycle {Masked, Unmasked} Write Block operation.

Table 5.7 {Replace, OR} Dirty Tag on Pixel ALU to {Masked, Unmasked} Write Block on Global Bus

Cycle	DRAM Activities	Pixel ALU Activities
n		Dirty Tag op specified on PALU_EN, PALU_WE, PALU_OP …
n+1		Dirty Tag data on PALU_DQ,
n+2		Pass through ROP/Blend 1
n+3		Pass through ROP/Blend 2
n+4		Pass through ROP/Blend 3
n+5		Pass through ROP/Blend 4
n+6	{Masked, Unmasked} Write Block op specified on DRAM_EN, DRAM_OP…	Data written to Dirty Tag
n+7	{Masked, Unmasked} Write Block on Global Bus	
n+8	{Masked, Unmasked} Write Block on Global Bus	

Table 5.8 shows a {Replace, OR} Dirty Tag operation immediately followed by a 3-cycle {Masked, Unmasked} Write Block operation.

Table 5.8 {Replace, OR} Dirty Tag on Pixel ALU to {Masked, Unmasked} Write Block on Global Bus

Cycle	DRAM Activities	Pixel ALU Activities
n		Dirty Tag op specified on PALU_EN, PALU_WE, PALU_OP …
n+1		Dirty Tag data on PALU_DQ,
n+2		Pass through ROP/Blend 1
n+3		Pass through ROP/Blend 2
n+4		Pass through ROP/Blend 3
n+5		Pass through ROP/Blend 4
n+6	{Masked, Unmasked} Write Block op specified on DRAM_EN, DRAM_OP …	Data written to Dirty Tag
n+7	{Masked, Unmasked} Write Block on Global Bus	
n+8	{Masked, Unmasked} Write Block on Global Bus	
n+9	{Masked, Unmasked} Write Block on Global Bus	

Table 5.9 shows a Write Register operation to the Plane Mask register followed by the latest 2-cycle Masked Write Block operation that can use the previous contents of the Plane Mask register.

Table 5.9 Write Register (Plane Mask) on Pixel ALU to Masked Write Block on Global Bus

Cycle	DRAM Activities	Pixel ALU Activities
n		Write Register op specified on PALU_EN, PALU_WE, PALU_OP …
n+1		Plane Mask data on PALU_DQ
n+2		
n+3	Masked Write Block op specified on DRAM_EN, DRAM_OP …	
n+4	Masked Write Block on Global Bus (uses old Plane Mask value)	
n+5	Masked Write Block on Global Bus (uses old Plane Mask value)	
n+6		Plane Mask register loaded
n+7		
n+8		
n+9		

Table 5.10 shows a Write Register operation to the Plane Mask register followed by the latest 3-cycle Masked Write Block operation that can use the previous contents of the Plane Mask register.

Table 5.10 Write Register (Plane Mask) on Pixel ALU to Masked Write Block on Global Bus

Cycle	DRAM Activities	Pixel ALU Activities
n		Write Register op specified on PALU_EN, PALU_WE, PALU_OP …
n+1		Plane Mask data on PALU_DQ
n+2	Masked Write Block op specified on DRAM_EN, DRAM_OP …	
n+3	Masked Write Block on Global Bus (uses old Plane Mask value)	
n+4	Masked Write Block on Global Bus (uses old Plane Mask value)	
n+5	Masked Write Block on Global Bus (uses old Plane Mask value)	
n+6		Plane Mask register loaded
n+7		
n+8		
n+9		

Table 5.11 shows a Write Register operation to the Plane Mask register followed by the earliest 2-cycle Masked Write Block operation that can use the new Plane Mask.

Table 5.11 Write Register (Plane Mask) on Pixel ALU to Masked Write Block on Global Bus

Cycle	DRAM Activities	Pixel ALU Activities
n		Write Register op specified on PALU_EN, PALU_WE, PALU_OP …
n+1		Plane Mask data on PALU_DQ
n+2		
n+3		
n+4		
n+5		
n+6	Masked Write Block op specified on DRAM_EN, DRAM_OP …	Plane Mask register loaded
n+7	Masked Write Block on Global Bus (uses new Plane Mask value)	
n+8	Masked Write Block on Global Bus (uses new Plane Mask value)	
n+9		

Table 5.12 shows a Write Register to the Plane Mask register followed by the earliest 3-cycle Masked Write Block operation that can use the new Plane Mask.

Table 5.12 Write Register (Plane Mask) on Pixel ALU to Masked Write Block on Global Bus

Cycle	DRAM Activities	Pixel ALU Activities
n		Write Register op specified on PALU_EN, PALU_WE, PALU_OP …
n+1		Plane Mask data on PALU_DQ
n+2		
n+3		
n+4		
n+5		
n+6	Masked Write Block op specified on DRAM_EN, DRAM_OP …	Plane Mask register loaded
n+7	Masked Write Block on Global Bus (uses new Plane Mask value)	
n+8	Masked Write Block on Global Bus (uses new Plane Mask value)	
n+9	Masked Write Block on Global Bus (uses new Plane Mask value)	

DRAM Activities

This section discusses consecutive DRAM operations on the same bank. To illustrate interlock timing for DRAM activities, we assume that the clock cycle time equals the minimum specification requirements—10ns or 13ns — depending on the speed grade of the parts. The interlock timing restrictions are listed in Table 7.6 for the DRAM operations on the same bank and in Table 7.7 for the DRAM operations on different banks.

Table 5.13 shows a minimal length video refresh sequence.

Table 5.13 Video refresh sequence

Cycle	External Activities	Internal Activities
n	Access Page specified	
n+1		Access Page
n+2		Access Page
n+3		Access Page
n+4	Video Transfer specified	Access Page
n+5		Video Transfer
n+6		Video Transfer
n+7		Video Transfer
n+8	Precharge Bank specified	Video Transfer
n+9		Precharge Bank
n+10		Precharge Bank
n+11		Precharge Bank
n+12		Precharge Bank

Table 5.14 shows minimal length DRAM refresh sequence.

Table 5.14 DRAM refresh sequence

Cycle	External Activities	Internal Activities
n	Access Page specified	
n+1		Access Page
n+2		Access Page
n+3		Access Page
n+4		Access Page
n+5		
n+6		
n+7		
n+8	Precharge Bank specified	
n+9		Precharge Bank
n+10		Precharge Bank
n+11		Precharge Bank
n+12		Precharge Bank

Table 5.15 shows a sequence of Read Block and {Masked, Unmasked} Write Block operations.

Table 5.15 Sequence of Read Block and {Masked, Unmasked} Write Block operations

Cycle	External Activities	Internal Activities
n	Access Page specified	
n+1		Access Page
n+2		Access Page
n+3		Access Page
n+4	1st Read Block specified	Access Page
n+5		1st Read Block
n+6	2nd Read Block specified	1st Read Block
n+7		2nd Read Block
n+8	1st Write Block specified	2nd Read Block
n+9		1st Write Block
n+10	2nd Write Block specified	1st Write Block
n+11		2nd Write Block
n+12	Precharge Bank specified	2nd Write Block
n+13		Precharge Bank
n+14		Precharge Bank
n+15		Precharge Bank
n+16		Precharge Bank

Table 5.16 shows Duplicate Page sequence.

Table 5.16 Duplicate Page sequence

Cycle	External Activities	Internal Activities
n	Access Page specified	
n+1		Access Page
n+2		Access Page
n+3		Access Page
n+4	Duplicate Page specified	Access Page
n+5		Duplicate Page
n+6		Duplicate Page
n+7		Duplicate Page
n+8		Duplicate Page
n+9		Duplicate Page
n+10		Duplicate Page
n+11		Duplicate Page
n+12	Precharge Bank specified	Duplicate Page
n+13		Precharge Bank
n+14		Precharge Bank
n+15		Precharge Bank
n+16		Precharge Bank

6 Frame Buffer Organizations

Frame Buffer Organizations

Introduction

There are many ways to use the 3D-RAM to implement frame buffers of various resolutions and depths. This section describes the following frame buffer organizations:

- 1280 x 1024 x 8 organization in single chip
- 1280 x 1024 x 32, organized as four 1280 x 1024 x 8 or 320 x 1024 x 32
- 1280 x 1024 x 32 double buffered organization with 32-bit Z
- 640 x 512 x 8 double buffered organization with 16-bit Z in single chip

1280 x 1024 x 8 Organization

In this organization, the screen display is made up of an 8W x 32H array of page groups (that is, 8 page groups wide by 32 page groups high). A page group is 160-pixel wide by 32-pixel high and consists of the same page from all four DRAM banks (A, B, C, D). The four independent DRAM banks can be interleaved to allow pages to be prefetched as images are drawn. Each page within a page group is 80-pixel wide by 16-pixel high. Pages are either sliced into sixteen 80-pixel wide scan lines when sending data to its Video Buffer or they are diced into a 10W x 4H array of 256-bit blocks when dealing with the Global Bus. Two pixels are shifted out of the Video Buffer every video clock.

Blocks are 8-pixel wide by 4-pixel high and can be transferred to and from one of the Pixel Buffer blocks via the Global Bus. The Pixel ALU and data pins access four pixels of a Pixel Buffer block at a time. The Dirty Tag for an entire Pixel Buffer block can be written in a single cycle from the data pins.

The following formulas determine which bank, page, etc. a given pixel is in, given the x and y coordinates of the pixel. The formulas use C syntax where the percent sign (“%”) indicates integer modulus operation and the slash sign (“/”) indicates integer division. These formulas are valid only when $0 \le x < 1280$ and $0 \le y < 1024$.

- bank = 2*((y%32)/16) + (x%160)/80, [0 = Bank A, 1 = Bank B, 2 = Bank C, 3 = Bank D]
- page = 8*(y/32) + x/160
- scan line within page = y%16
- block within page = (y%16)/4 + 4* ((x%80)/8)
- word within block = 2*(y%4) + (x%8)/4
- pixel (byte) within word = x%4

The mapping of page groups to the display screen is completely user definable. The following mappings are hardwired inside the 3D-RAM: blocks to pages, scan lines to pages, words to Pixel Buffer blocks, and Dirty Tags to Pixel Buffer blocks.

1280 x 1024 x 32 Single Buffered Organization

A frame buffer of this size requires four 3D-RAMs; however, there are two recommended ways of organizing the 3D-RAMs which trade off 2D color expansion rendering performance with pixel oriented rendering performance.

- Each of the four components of a pixel (R, G, B, a) are in separate 3D-RAMs. Thus, each 3D-RAM supports 1280 x 1024 x 8. This section describes this implementation.
- All four components of a pixel reside in the same 3D-RAMs. The four 3D-RAMs are interleaved on a pixel by pixel basis in a scan line. Thus, each 3D-RAM supports 320 x 1024 x 32. Page 107 describes this implementation.

The 320 x 1024 x 32 mode is nearly the same as 1280 x 1024 x 8 except that the pixels are four times as deep and the widths of the screen, page groups, pages, and blocks are one fourth as wide. One pixel is shifted out of the Video Buffer every two video clocks. The Pixel ALU and PALU_DQ pins access one pixel of a Pixel Buffer block. The

Dirty Tag for an entire Pixel Buffer block can be written in a single cycle from the PALU_DQ pins. The Dirty Tag controls the four bytes of 32-bit pixel independently.

The following formulas determine which bank, page etc. a given pixel is in, given the x and y coordinates of the pixel.

- bank = 2*((y%32)/16) + (x%40)/20, [0 = Bank A, 1= Bank B, 2 = Bank C, 3 = Bank D]
- page = 8*(y/32) + x/40
- scan line within page = y%16
- block within page = (y%16)/4 + 4* ((x%20) /2)
- pixel (word) within block = 2*(y%4) + (x%2)

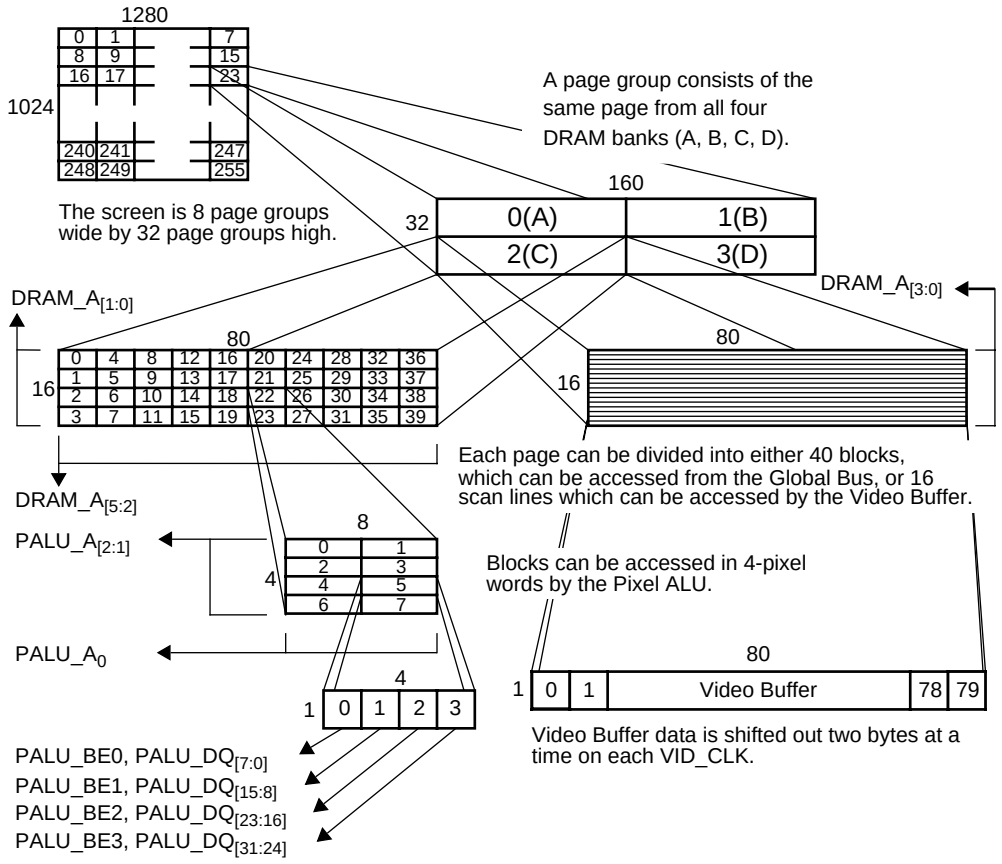


Figure 6.1 This diagram shows how 3D-RAM maps to pixels in a single-chip 1280x1024x8 frame buffer. The numbers outside each rectangle show its dimensions in pixels.

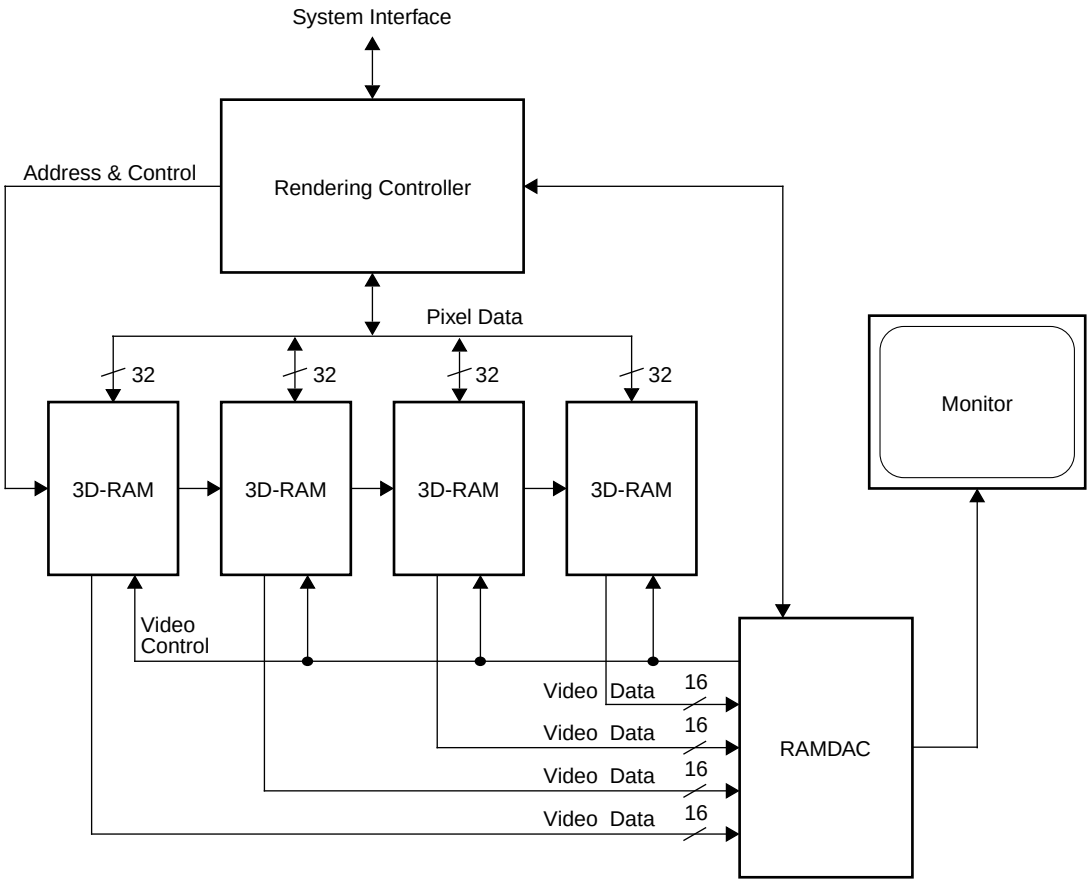


Figure 6.2 1280 x 1024 x 32 Single Buffer 3D-RAM System

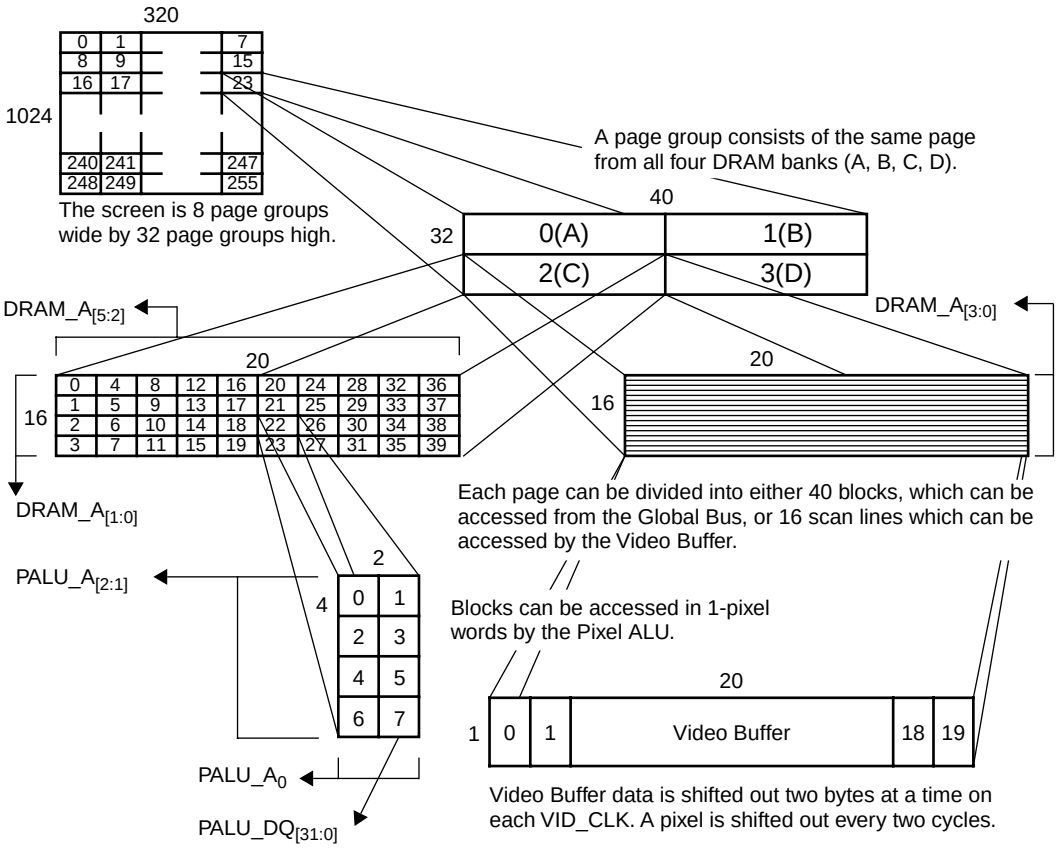


Figure 6.3 This diagram shows how 3D-RAM maps to pixels in a single chip 320x1024x32 frame buffer. The numbers outside each rectangle show its dimensions in pixels.

1280 x 1024 x 32 Double Buffered Organization with Z

The basic configuration for a 1280 x 1024 x 32 double buffered organization with Z Buffer is shown in Figure 6.4. This configuration uses only twelve 3D-RAMs. In this example each 3D-RAM (for Buffers A, B, and Z) covers a 320 x 1024 portion of the 1280 x 1024 displayed image. The interleave is in the x direction. This implies that vertical scrolling can take place at a very high speed because all data movement occurs within the 3D-RAM chips rather than across chips. Horizontal scrolling would require 3D-RAM to 3D-RAM data transfers. Each of Buffers A, B, and Z is 32 bits in pixel depth. This allows 8 bits each for R, G, B, and 8 bits for alpah or overlays, and we refer to the eight 3D-RAMs containing these data as the Color Buffer 3D-RAMs. In the case of Z Buffer, 24 bits can be used for depth and 8 bits for a combination of stencil pattern ID and window ID, and we refer to these four 3D-RAMs as the Z Buffer 3D-RAMs.

The rendering controller is shown with a 256-bit interface for maximum performance. The three 3D-RAMs (one from each of Buffers A, B, and Z) that hold the data for the same pixels share a 64-bit bus. More specifically, the two 3D-RAM chips in the Buffers A and B share the same 32-bit data bus because only one of them is active for rendering and the other outputs display data throught the video port, while the 3D-RAM chip in the Z Buffer requires its own 32-bit data bus. A 64-bit or 128-bit bus between the rendering controller and the 3D-RAMs could be used but with some loss of performance due to more restricted bandwidth and higher bus loading (implying lower maximum clock frequency).

The Z Buffer 3D-RAM utilize their Compare units to check depth, stencil and window ID, and supply the result to the PASS_OUT pins. The PASS_OUT pins of Z Buffer are connected to the PASS_IN pins of the corresponding Color Buffer 3D-RAMs. The results of all ALU operations are conditionally written to the Pixel Buffer, depending on the states of the PASS_IN pins (and on the states of the PASS_OUT pins of the Color Buffer 3D-RAMs themselves if they also perform compare tests).

Both Buffers A and B are connected to the RAMDAC chip using a 128-bit bus. Buffers A and B can be selected on a pixel-by-pixel basis, alternating between the two buffers.

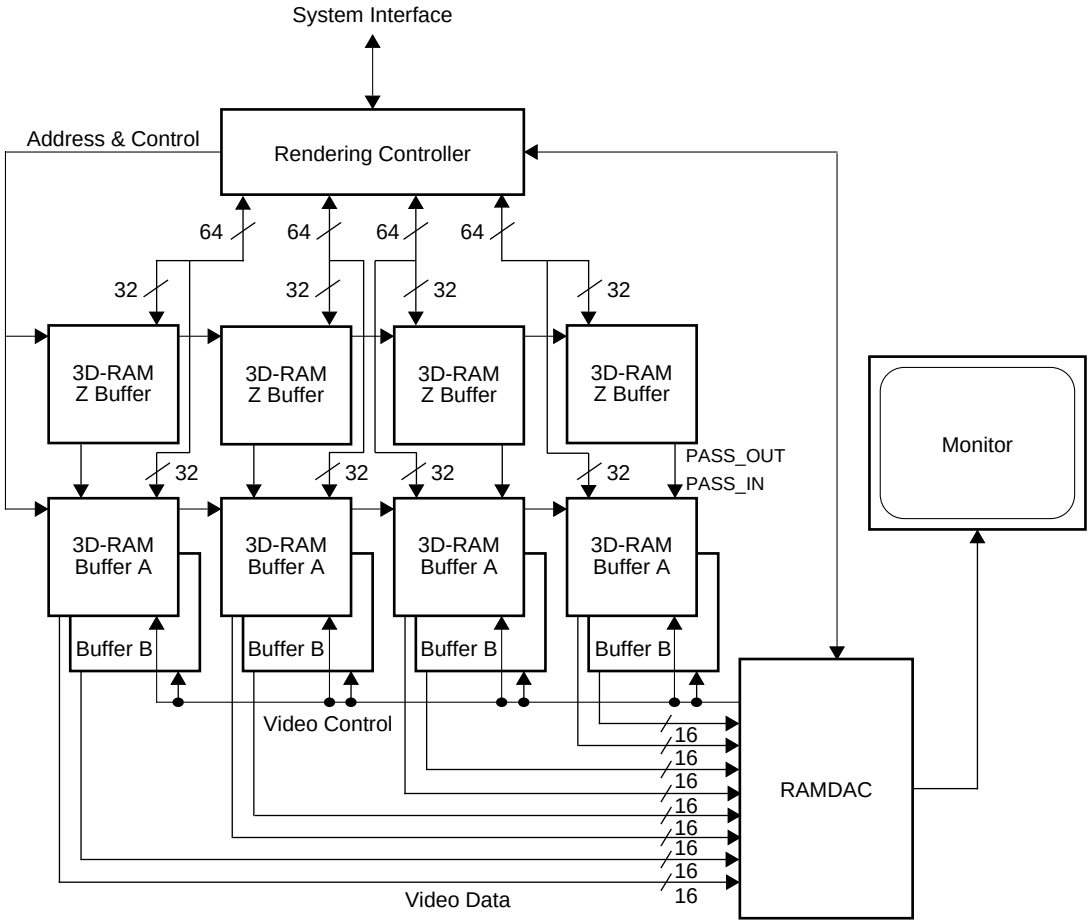


Figure 6.4 1280 x 1024 x 32 double buffered organization with 32-bit Z Buffer

640 x 512 x 8 Double Buffered Organization With Z

A single 3D-RAM chip can be configured to support 640 x 512 x 8 double buffered organization with 16-bit Z. This configuration might be suitable for a very high performance, low cost consumer home or arcade game application.

The basic allocation of memory can be seen in Figure 6.6. One fourth of the 3D-RAM serves as Buffer A, one fourth as Buffer B, and the rest as the 16-bit Z Buffer. All Z compares and ROP/ Blend functions are done on the same 3D-RAM. A 32-bit Pixel and Z data bus is provided to the rendering controller. A 16-bit bus interfaces to the RAMDAC.

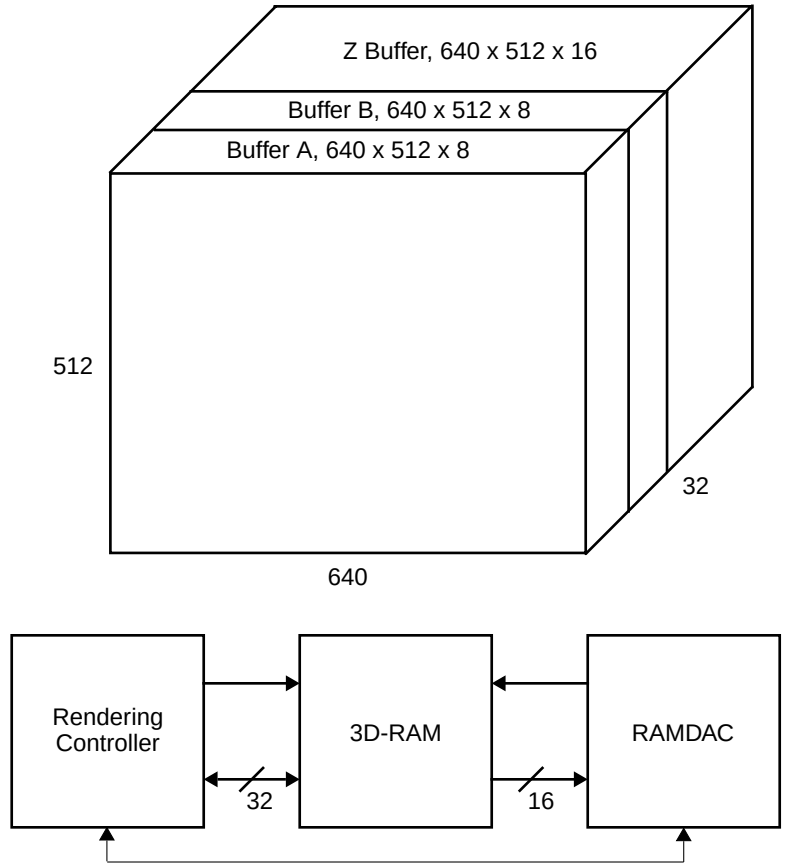


Figure 6.5 Using single 3D-RAM to configure double buffered 640 x 512 x 8 with 16-bit Z

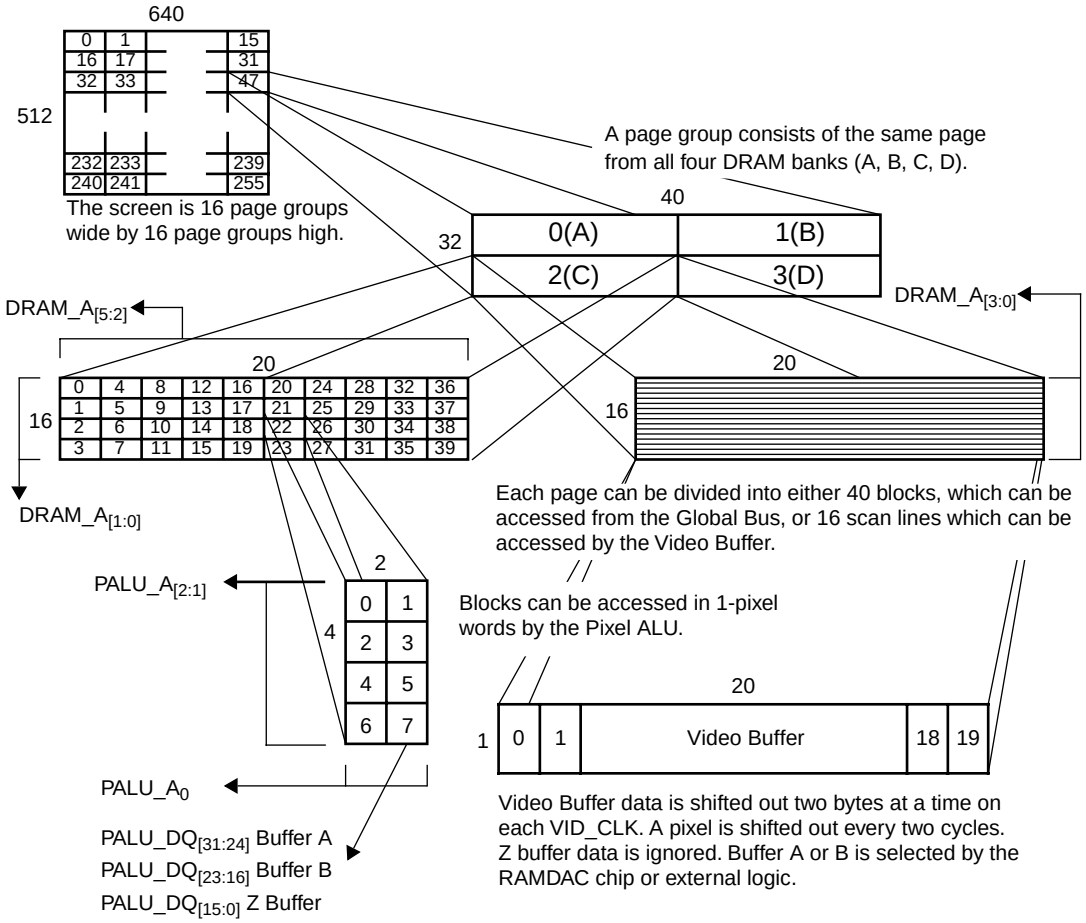


Figure 6.6 This diagram shows how 3D-RAM maps to pixels in a single chip 640x512x8 frame buffer. The numbers outside each rectangle show its dimensions in pixels.

7 Electrical Specifications

Electrical Specifications

Absolute Maximum Ratings

Table 7.1 Absolute Maximum Ratings

Symbol	Parameter	Conditions	Ratings	Unit
V_{DD}	Supply Voltage	with respect to Vss	– 0.5 to 4.6	V
V_I	Input Voltage		– 0.5 to 4.6	V
V_O	Output Voltage		– 0.5 to 4.6	V
I_O	Output Current	—	50	mA
T_j	Maximum Junction Temperature	—	125	°C
T_{opr}	Operation Temperature	—	0 to 70	°C
T_{stg}	Storage Temperature	—	– 65 to 150	°C

Testing Conditions

The supply voltage VDD and ambient temperature Ta for testing are as follows:

VDD = 3.3 V ± 5%, Ta = 0 °C to 70 °C

Figure 7.1 shows the output test load for the PALU_DQ, PASS_OUT, VID_Q, and VID_QSF pins. The capacitive loading CL is 60 pF for the PALU_DQ pins, 30 pF for the PASS_OUT pin, and 20 pF for both the VID_Q pins and the VID_QSF pin. Figure 7.2 is the output test load for the open-drain HIT pin, with Rpu = 330 W for pull-up and CL = 75 pF.

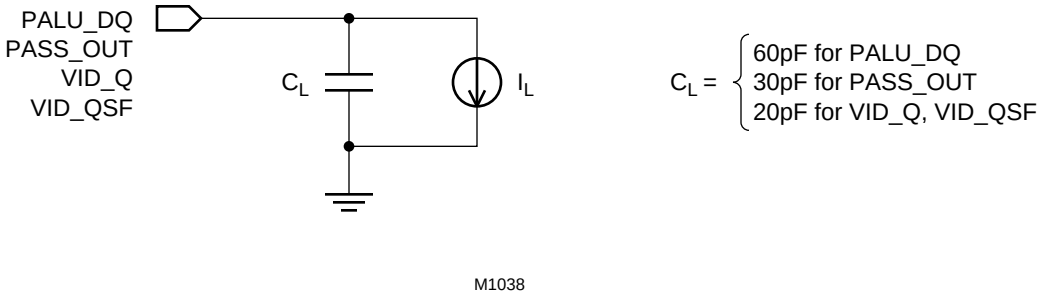


Figure 7.1 Output test load for the PALU_DQ, PASS_OUT, VID_Q, AND VID_QSF pins

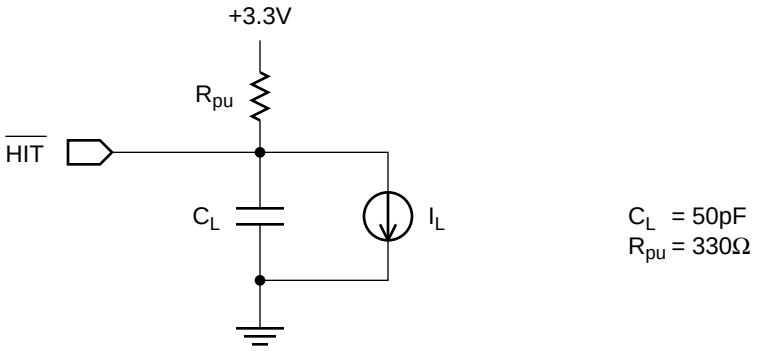
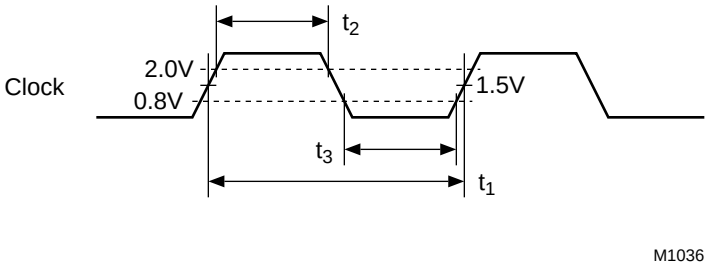


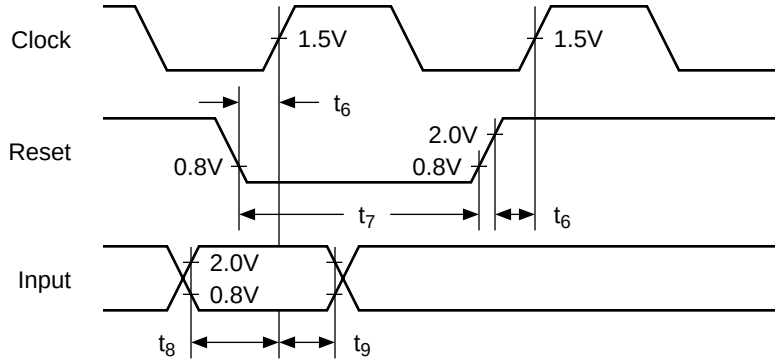
Figure 7.2 Output test load for the HIT pin

The AC timing measurements are summarized in Figure 7.3 through Figure 7.6. The clock waveform measurements are shown in Figure 7.3. The input and output timing measurements are shown in Figure 7.4 and Figure 7.5, respectively. Figure 7.6 shows the asynchronous output enable timing measurements.



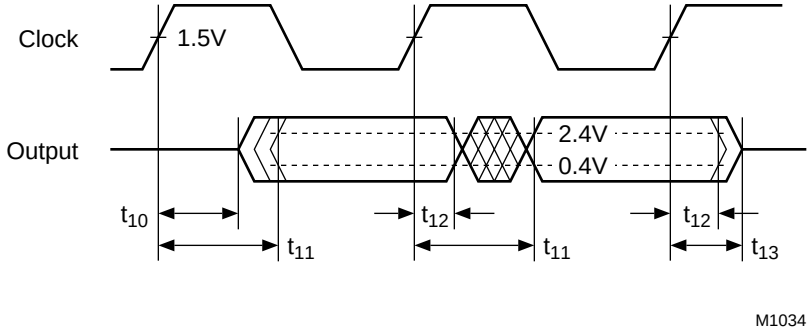
t_1 : Clock cycle time (minimum)
t_2 : Clock high pulse width (minimum)
t_3 : Clock low pulse width (minimum)

Figure 7.3 Clock waveform measurement



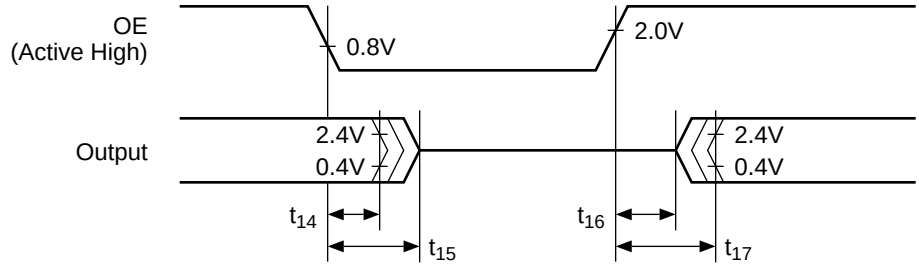
t_6 : Reset setup time (minimum)
t_7 : Reset pulse width (minimum)
t_8 : Input setup time (minimum)
t_9 : Input hold time (minimum)

Figure 7.4 Input timing measurement



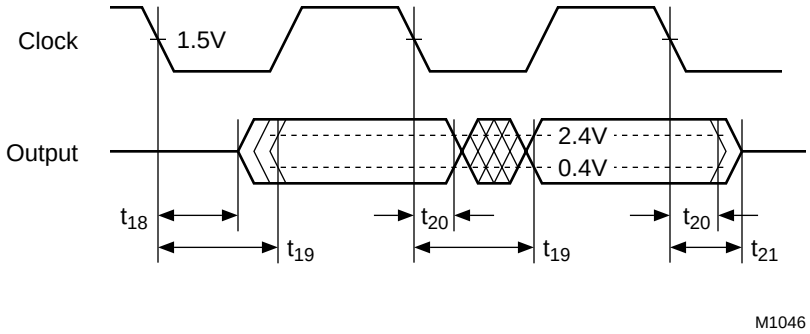
t_{10} : Clock to output low impedance, $I_O > 2 * I_{OZ}$ (minimum)
t_{11} : Output access time from clock (maximum)
t_{12} : Output valid time after clock (minimum)
t_{13} : Clock to output high impedance, $I_O < I_{OZ}$ (maximum)

Figure 7.5 Output timing measurement



t_{14} : Valid output after OE low (minimum)
t_{15} : Output high impedance, $I_O < I_{OZ}$, after OE low (maximum)
t_{16} : Output low impedance, $I_O > 2 \times I_{OZ}$, after OE high (minimum)
t_{17} : Valid output after OE high (maximum)

Figure 7.6 Asynchronous output enable timing measurement



t_{18} : Clock to output low impedance, $I_O > 2 \times I_{OZ}$ (minimum)
t_{19} : Output access time from clock (maximum)
t_{20} : Output valid time after clock (minimum)
t_{21} : Clock to output high impedance, $I_O < I_{OZ}$ (maximum)

Figure 7.7 SCAN_TDO timing measurement

DC Specifications

Table 7.2 lists the DC characteristics and the operation conditions. Table 7.3 lists the average supply current according to the operations, which include the Pixel ALU, DRAM and video operations.

Table 7.2 DC characteristics

V_{DD} = 3.3V ± 5%, T_a = 0°C~70°C				
Symbol	**Parameter**	**Min**	**Max**	**Unit**
V_{IH}[a]	Input High Voltage	2.0	V_{DD} + 0.3	V
V_{IL}[b]	Input Low Voltage	– 0.3	0.8	V
V_{IH} (PASS_IN$_{[1:0]}$)	PASS_IN$_{[1:0]}$ High Voltage	1.5	V_{DD} + 0.3	V
V_{IL} (PASS_IN$_{[1:0]}$)	PASS_IN$_{[1:0]}$ Low Voltage	– 0.3	0.9	V
V_{OH}[c]	Output High Voltage, I_L = – 0.2 mA	2.4	—	V
V_{OL}[d]	Output Low Voltage, I_L = 0.2 mA	0	0.4	V
V_{OH} (PASS_OUT)	PASS_OUT High Voltage, I_L=-0.1 mA	1.9	—	V
V_{OL} (PASS_OUT)	PASS_OUT Low Voltage, I_L = 0.1 mA	—	0.5	V
V_{OH} ($\overline{HIT}$)	$\overline{HIT}$ High Voltage	—	—	V
V_{OL} ($\overline{HIT}$)	$\overline{HIT}$ Low Voltage	—	0.8	V
I_{OZ}	Output Leakage Current in Tri-state	– 10	10	µA
I_{IL}	Input Leakage Current	– 10	10	µA
C_{IN}	Input Capacitance	—	5	pF
C_{CLK}	CLK Input Capacitance	—	7	pF
$C_{I/O}$	I/O Capacitance	—	7	pF

a. This parameter applies to every input pin except PASS_IN$_{[1:0]}$.

b. This parameter applies to every input pin except PASS_IN$_{[1:0]}$.

c. This parameter applies to all output pins except PASS_OUT and $\overline{HIT}$.

d. This parameter applies to all output pins except PASS_OUT and $\overline{HIT}$.

Table 7.3 Average supply current by function

Symbol	Parameter	M5M410092B		Unit
		-10	-13	
$I_{CC<ALU>}$	Average supply current for ALU operation t_{CLK} = min, (1 MCLK cycle)	260	200	mA
$I_{CC<NOP>}$	Average standby current t_{CLK} = ∞	20	20	mA
$I_{CC<ACP>}$	Average supply current for DRAM operation ACP t_{CLK} = min, (4 MCLK cycles)	105	80	mA
$I_{CC<PRE>}$	Average supply current for DRAM operation PRE t_{CLK} = min, (4 MCLK cycles)	55	40	mA
$I_{CC<DUP>}$	Average supply current for DRAM operation DUP t_{CLK} = min, (8 MCLK cycles)	55	40	mA
$I_{CC<RDB>}$	Average supply current for DRAM operation RDB t_{CLK} = min, (2 or 3 MCLK cycles)	160	120	mA
$I_{CC<UWB>}$	Average supply current for DRAM operation UWB t_{CLK} = min, (2 or 3 MCLK cycles)	160	120	mA
$I_{CC<VDX>}$	Average supply current for DRAM operation VDX t_{CLK} = min, (4 MCLK cycles)	80	60	mA
$I_{CC<VID>}$	Average supply current for video output t_{VCLK} = min	60	60	mA

AC Specifications

Every AC timing parameter is illustrated in at least one of the timing figures in Chapter 8. The “Refer” column in each timing table refers to the exact measurement levels illustrated in Figures 7.3 through 7.6.

Pixel ALU Timing Parameters

The timing parameters of M5M410092B-10 and -13 are presented in Table 7.4.

Table 7.4 Pixel ALU timing parameters

Symbol	Parameter	M5M410092B				Unit	Refer	Ch. 8 Figure
		-10		-13				
		Min	Max	Min	Max			
t_{CLK}	Master clock MCLK cycle time	10	16000	13	16000	ns	t_1	4
t_{CLKH}	MCLK high pulse width	4	—	5	—	ns	t_2	4
t_{CLKL}	MCLK low pulse width	4	—	5	—	ns	t_3	4
t_{RSS}	$\overline{\text{RESET}}$ setup time	0	—	0	—	ns	t_6	1
t_{RSP}	$\overline{\text{RESET}}$ pulse width	40	—	52	—	ns	t_7	2
t_{ENS}	PALU_EN setup time	3	—	4	—	ns	t_8	4
t_{ENH}	PALU_EN hold time	0	—	0	—	ns	t_9	4
t_{OPS}	PALU_OP setup time	3	—	4	—	ns	t_8	4
t_{OPH}	PALU_OP hold time	0	—	0	—	ns	t_9	4
t_{ADS}	PALU_A setup time	3	—	4	—	ns	t_8	4
t_{ADH}	PALU_A hold time	0	—	0	—	ns	t_9	4
t_{DQS}	PALU_DQ, PALU_DX setup time	3	—	4	—	ns	t_8	5
t_{DQH}	PALU_DQ, PALU_DX hold time	0	—	0	—	ns	t_9	5

Table 7.4 Pixel ALU timing parameters (cont.)

Symbol	Parameter	M5M410092B				Unit	Refer	Ch. 8 Figure
		-10		-13				
		Min	Max	Min	Max			
t_{WES}	PALU_WE setup time	3	—	4	—	ns	t_8	4
t_{WEH}	PALU_WE hold time	0	—	0	—	ns	t_9	4
t_{BES}	PALU_BE setup time	3	—	4	—	ns	t_8	4
t_{BEH}	PALU_BE hold time	0	—	0	—	ns	t_9	4
t_{CLZ}	MCLK to PALU_DQ low impedance	4	—	5	—	ns	t_{10}	4
t_{CQ}	PALU_DQ access time	—	14	—	20	ns	t_{11}	4
t_{CVD}	PALU_DQ data valid time	3	—	3	—	ns	t_{12}	4
t_{CHZ}	MCLK to PALU_DQ high impedance	—	3	—	3	ns	t_{13}	4
t_{PSS}	PASS_IN setup time	2	—	3	—	ns	t_8	5
t_{PSH}	PASS_IN hold time	0	—	0	—	ns	t_9	5
t_{CPS}	MCLK to valid PASS_OUT	—	6	—	8	ns	t_{11}	5
t_{CPSV}	PASS_OUT data valid time	3	—	3	—	ns	t_{12}	5
t_{CHT}	MCLK to valid $\overline{HIT}$	—	35	—	35	ns	t_{11}	6

DRAM Timing Parameters

The measurements of the DRAM interlock timings in Tables 7.6 and 7.7 are from the MCLK rising edge of the first operation to the MCLK rising edge of the second operation. Both MCLK edges are measured at 1.5 V.

Table 7.5 Minimum requirements of the DRAM timing parameters

Symbol	Parameter	M5M410092B		Unit	Refer	Ch. 8 Figure
		-10	-13			
t_{REF}	Refresh interval for array	17	17	ms	—	—
t_{DENS}	DRAM_EN setup time	3	4	ns	t_8	7
t_{DENH}	DRAM_EN hold time	0	0	ns	t_9	7
t_{DOPS}	DRAM_OP setup time	3	4	ns	t_8	7
t_{DOPH}	DRAM_OP hold time	0	0	ns	t_9	7
t_{DBKS}	DRAM_BS setup time	3	4	ns	t_8	7
t_{DBKH}	DRAM_BS hold time	0	0	ns	t_9	7
t_{DADS}	DRAM_A setup time	3	4	ns	t_8	7
t_{DADH}	DRAM_A hold time	0	0	ns	t_9	7

Table 7.6 Minimum requirements of the DRAM interlock timings for operations on same bank

Symbol	Parameter	M5M410092B		Unit	Ch. 8 Timing Figure
		-10	-13		
t_{dABS}	Access Page to Block Transfer	40	40	ns	7
t_{dAPS}[a]	Access Page to Precharge Bank	80	104	ns	7
t_{dADS}	Access Page to Duplicate Page	40	52	ns	8
t_{dAVS}	Access Page to Video Transfer	40	52	ns	9
t_{dBBS}	Block Transfer to Block Transfer	20	26	ns	7
t_{dBPS}	Block Transfer to Precharge Bank	20	26	ns	7
t_{dBDS}	Block Transfer to Duplicate Page	20	26	ns	8
t_{dBVS}	Block Transfer to Video Transfer	20	26	ns	10
t_{dPAS}[b]	Precharge Bank to Access Page	40	52	ns	7
t_{dPPS}	Precharge Bank to Precharge Bank	40	52	ns	10
t_{dDBS}	Duplicate Page to Block Transfer	80	104	ns	8
t_{dDPS}	Duplicate Page to Precharge Bank	80	104	ns	9
t_{dDDS}	Duplicate Page to Duplicate Page	80	104	ns	8
t_{dDVS}	Duplicate Page to Video Transfer	80	104	ns	9
t_{dVBS}	Video Transfer to Block Transfer	40	52	ns	10
t_{dVPS}	Video Transfer to Precharge Bank	40	52	ns	9
t_{dVDS}	Video Transfer to Duplicate Page	40	52	ns	9
t_{dVVS}	Video Transfer to Video Transfer	80	104	ns	9

a. The maximum timing limit from ACP to PRE is 100,000 ns.

b. The operation from PRE to ACP requires at least two clock cycles. At the first clock rising edge, PRE starts. At the second clock rising edge, the preparation for ACP starts.

Table 7.7 Minimum requirement of the DRAM interlock timings for operations between two different banks

Symbol	Parameter	M5M410092B		Unit	Ch. 8 Timing Figure
		-10	-13		
t_{dAAD}	Access Page to Access Page	40	52	ns	11
t_{dABD}	Access Page to Block Transfer	10	13	ns	11
t_{dAPD}	Access Page to Precharge Bank	40	52	ns	11
t_{dADD}	Access Page to Duplicate Page	40	52	ns	12
t_{dAVD}	Access Page to Video Transfer	40	52	ns	13
t_{dBAD}	Block Transfer to Access Page	10	13	ns	11
t_{dBBD}	Block Transfer to Block Transfer	20	26	ns	11
t_{dBPD}	Block Transfer to Precharge Bank	10	13	ns	11
t_{dBDD}	Block Transfer to Duplicate Page	10	13	ns	11
t_{dBVD}	Block Transfer to Video Transfer	10	13	ns	13
t_{dPAD}	Precharge Bank to Access Page	10	13	ns	11
t_{dPBD}	Precharge Bank to Block Transfer	10	13	ns	11
t_{dPPD}	Precharge Bank to Precharge Bank	10	13	ns	11
t_{dPDD}	Precharge Bank to Duplicate Page	10	13	ns	11
t_{dPVD}	Precharge Bank to Video Transfer	10	13	ns	13
t_{dDAD}	Duplicate Page to Access Page	80	104	ns	12
t_{dDBD}	Duplicate Page to Block Transfer	10	13	ns	12
t_{dDPD}	Duplicate Page to Precharge Bank	40	52	ns	12
t_{dDDD}	Duplicate Page to Duplicate Page	80	104	ns	12
t_{dDVD}	Duplicate Page to Video Transfer	80	104	ns	13
t_{dVAD}	Video Transfer to Access Page	40	52	ns	13
t_{dVBD}	Video Transfer to Block Transfer	10	13	ns	13
t_{dVPD}	Video Transfer to Precharge Bank	40	52	ns	13
t_{dVDD}	Video Transfer to Duplicate Page	40	52	ns	13
t_{dVVD}	Video Transfer to Video Transfer	80	104	ns	13

Video Buffer Timing Parameters

Table 7.8 Video Buffer timing parameters

Symbol	Parameter	M5M410092B				Unit	Refer	Ch. 8 Figure
		-10		-13				
		Min	Max	Min	Max			
t_{VCLK}	VID_CLK cycle time	12	—	14	—	ns	t_1	14
t_{VCLKH}	VID_CLK high pulse width	5	—	6	—	ns	t_2	14
t_{VCLKL}	VID_CLK low pulse width	5	—	6	—	ns	t_3	14
t_{VCES}	VID_CKE setup time	4	—	4	—	ns	t_8	14
t_{VCEH}	VID_CKE hold time	0	—	0	—	ns	t_9	14
t_{VQ}	VID_Q access time from VID_CLK	—	8	—	10	ns	t_{11}	14
t_{VQVC}	VID_Q valid after VID_CLK	3	—	3	—	ns	t_{12}	14
t_{VLZ}	VID_Q output low impedance	3	—	3	—	ns	t_{16}	14
t_{VHZ}	VID_Q output high impedance	—	3	—	3	ns	t_{15}	14
t_{VQE}	VID_Q access time from VID_OE high	—	9	—	11	ns	t_{17}	14
t_{VQVE}	VID_Q valid after VID_OE low	—	—	—	—	ns	t_{14}	14
t_{VXCI1}	Initial VDX after last internal VID_CLK	14	—	14	—	ns	—	15
t_{VXCI2}	Initial VDX before next internal VID_CLK	80	—	80	—	ns	—	15
t_{VXQFI}	VID_QSF delay time after initial VDX	—	80	—	80	ns	t_{11}	15
t_{QSF}	VID_QSF delay time from internal VID_CLK 38	—	25	—	25	ns	t_{11}	16
t_{VXC1}	Normal VDX after internal VID_CLK 38	20	—	20	—	ns	—	16
t_{VXC2}	Normal VDX before next internal VID_CLK 38	60	—	60	—	ns	—	16

Boundary-Scan Timing Parameters

Table 7.9 Boundary-Scan timing parameters

Symbol	Parameter	M5M410092B -10, -13		Unit	Refer	Ch. 8 Figure
		Min	Max			
t_{SCLK}	SCAN_TCK cycle time	100	—	ns	t_1	17
t_{SCLKH}	SCAN_TCK high pulse width	40	—	ns	t_2	17
t_{SCLKL}	SCAN_TCK low pulse width	40	—	ns	t_3	17
t_{SCNTS}	SCAN_TMS setup time	8	—	ns	t_8	17
t_{SCNTH}	SCAN_TMS hold time	26	—	ns	t_9	17
t_{SCNIS}	SCAN_TDI setup time	8	—	ns	t_8	17
t_{SCNIH}	SCAN_TDI hold time	26	—	ns	t_9	17
t_{SLZ}	SCAN_TCK to SCAN_TDO low impedance	—	20	ns	t_{18}	17
t_{SQ}	SCAN_TDO access time	—	26	ns	t_{19}	17
t_{SVD}	SCAN_TDO data valid time	8	—	ns	t_{20}	17
t_{SHZ}	SCAN_TCK to SCAN_TDO high impedance	—	20	ns	t_{21}	17
t_{SCNRS}	$\overline{\text{SCAN\_RST}}$ setup time	8	—	ns	t_6	18
t_{SCNRP}	$\overline{\text{SCAN\_RST}}$ pulse width	30	—	ns	t_7	18

7 Electrical Specifications

8 Timing Diagrams

Timing Diagrams

This chapter shows 18 timing diagrams, which are summarized in Table 8.1. These diagrams show the gross timing specifications. Refer to Chapter 7 for exact timing measurements. Also, the entries of parameter values from Tables 7.4 through 7.9 are repeated here for convenient reference.

Table 8.30 Timing diagram figures

Figure	Description
8.1	Power on reset
8.2	Restart reset
8.3	DRAM array initialization
8.4	Pixel port read
8.5	Pixel port write
8.6	Pick logic timing
8.7	DRAM operations on the same bank (1)
8.8	DRAM operations on the same bank (2)
8.9	DRAM operations on the same bank (3)
8.10	DRAM operations on the same bank (4)
8.11	DRAM operations between two different banks (1)
8.12	DRAM operations between two different banks (2)
8.13	DRAM operations between two different banks (3)
8.14	Internal VID_CLK and video output timing
8.15	Video output sequence from initial VDX for normal and reversed modes
8.16	Continuous video output sequence in normal mode during display
8.17	Boundary scan
8.18	Boundary scan reset

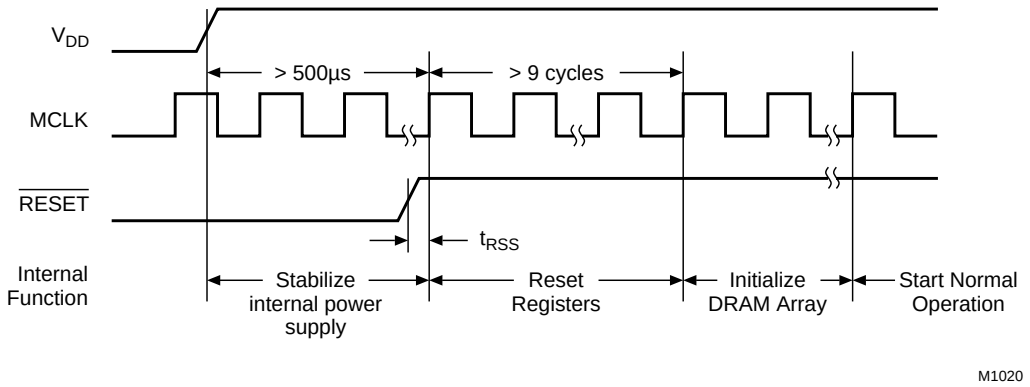


Figure 8.28 Power on reset

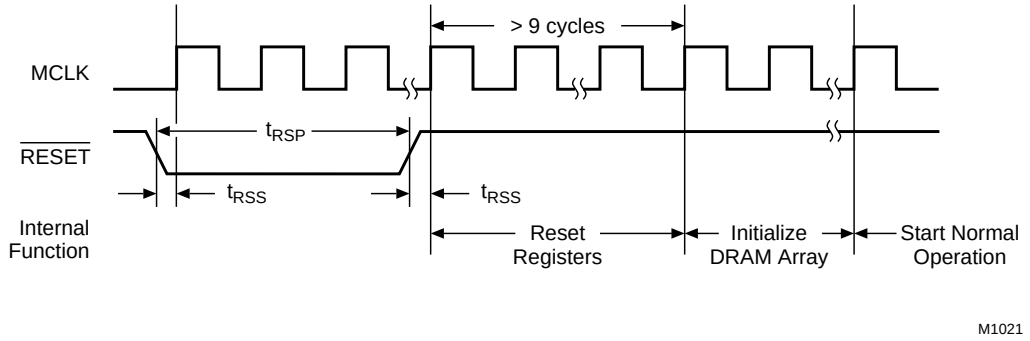


Figure 8.29 Restart reset

Table 8.31 Reset timing parameters

Symbol	Parameter	M5M410092B				Unit	Refer
		-10		-13			
		Min	Max	Min	Max		
t_{RSS}	$\overline{RESET}$ setup time	0	—	0	—	ns	t_6
t_{RSP}	$\overline{RESET}$ pulse width	40	—	52	—	ns	t_7

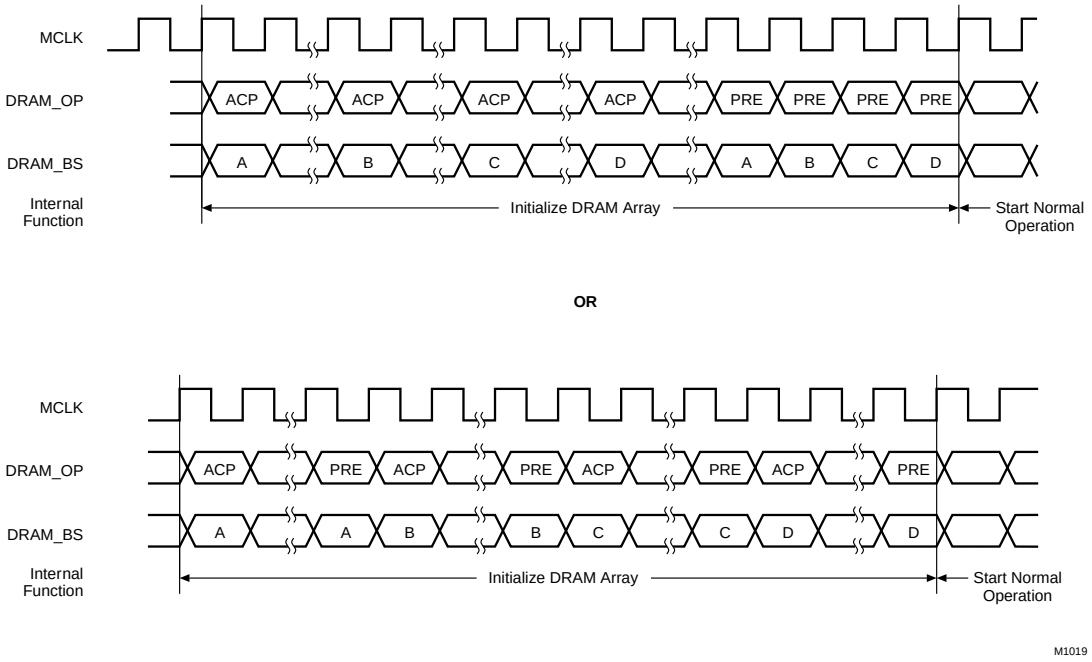
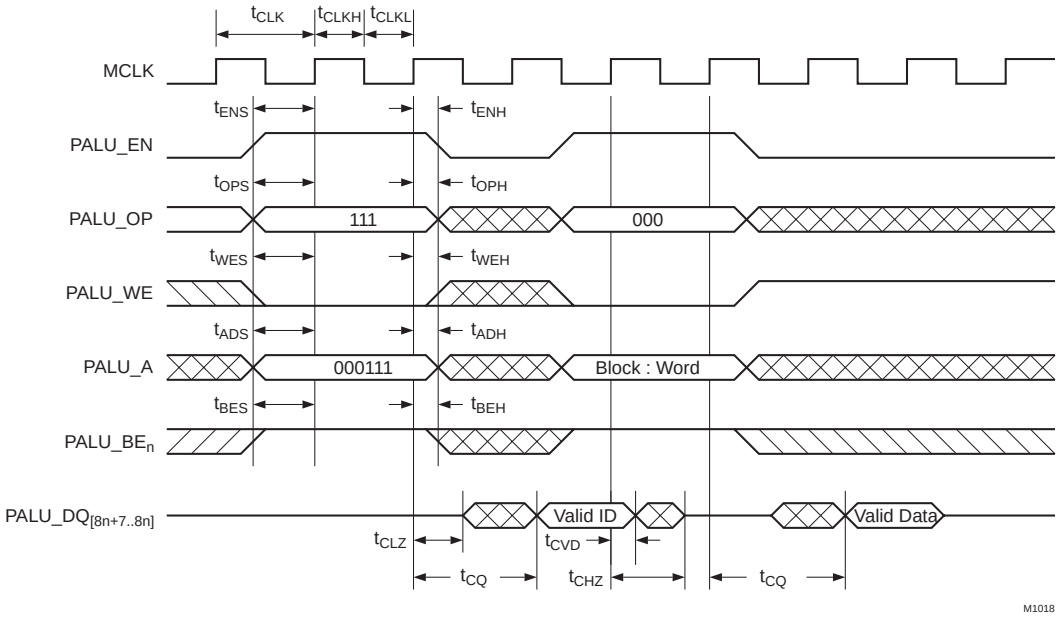


Figure 8.30 DRAM array initialization



Note: Refer to Figure 2.6 for an example of combined operations of Pixel ALU read and write.

Figure 8.31 Pixel port read

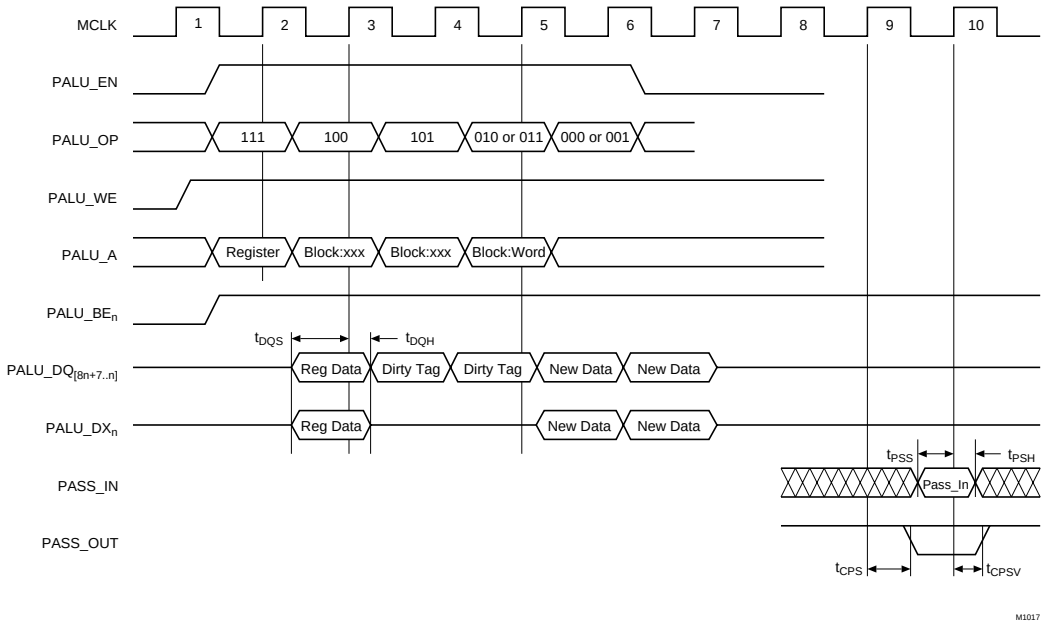
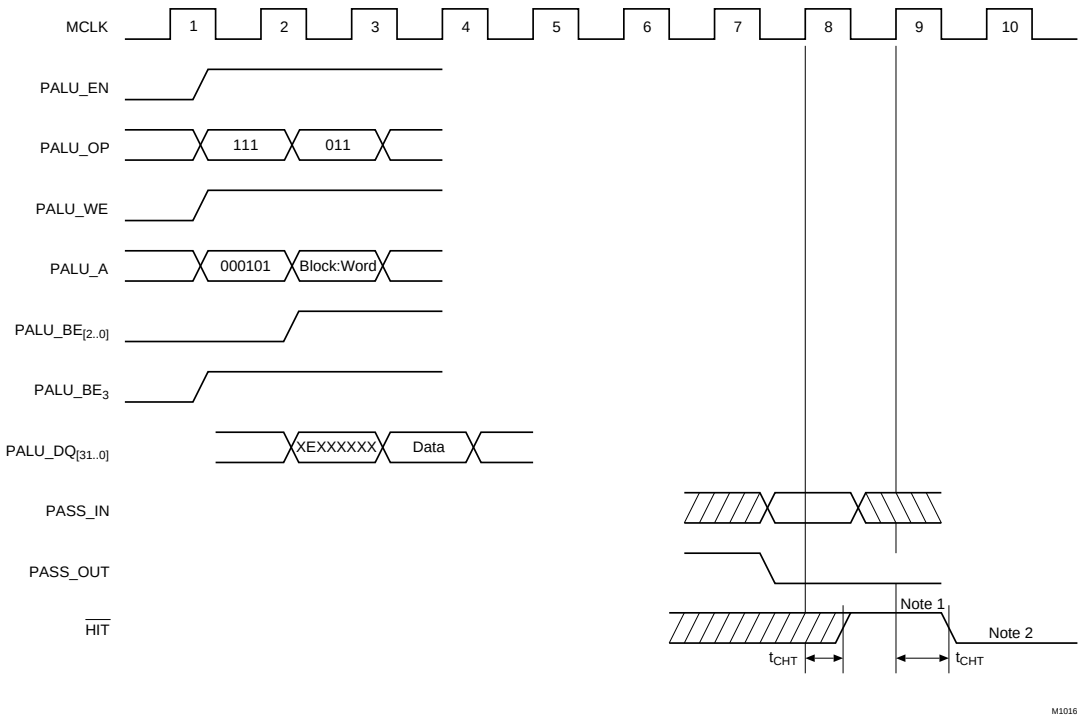


Figure 8.32 Pixel port write

Table 8.32 Pixel ALU timing parameters

Symbol	Parameter	M5M410092B				Unit	Refer
		-10		-13			
		Min	Max	Min	Max		
t_{CLK}	Master clock MCLK cycle time	10	16000	13	16000	ns	t_1
t_{CLKH}	MCLK high pulse width	4	—	5	—	ns	t_2
t_{CLKL}	MCLK low pulse width	4	—	5	—	ns	t_3
t_{ENS}	PALU_EN setup time	3	—	4	—	ns	t_8
t_{ENH}	PALU_EN hold time	0	—	0	—	ns	t_9
t_{OPS}	PALU_OP setup time	3	—	4	—	ns	t_8
t_{OPH}	PALU_OP hold time	0	—	0	—	ns	t_9
t_{ADS}	PALU_A setup time	3	—	4	—	ns	t_8
t_{ADH}	PALU_A hold time	0	—	0	—	ns	t_9
t_{WES}	PALU_WE setup time	3	—	4	—	ns	t_8
t_{WEH}	PALU_WE hold time	0	—	0	—	ns	t_9
t_{BES}	PALU_BE setup time	3	—	4	—	ns	t_8
t_{BEH}	PALU_BE hold time	0	—	0	—	ns	t_9
t_{CLZ}	MCLK to PALU_DQ low impedance	4	—	5	—	ns	t_{10}
t_{CQ}	PALU_DQ access time	—	14	—	20	ns	t_{11}
t_{CVD}	PALU_DQ data valid time	3	—	3	—	ns	t_{12}
t_{CHZ}	MCLK to PALU_DQ high impedance	—	3	—	3	ns	t_{13}
t_{DQS}	PALU_DQ, PALU_DX setup time	3	—	4	—	ns	t_8
t_{DQH}	PALU_DQ, PALU_DX hold time	0	—	0	—	ns	t_9
t_{PSS}	PASS_IN setup time	2	—	3	—	ns	t_8
t_{PSH}	PASS_IN hold time	0	—	0	—	ns	t_9
t_{CPS}	MCLK to valid PASS_OUT	—	6	—	8	ns	t_{11}
t_{CPSV}	PASS_OUT data valid time	3	—	3	—	ns	t_{12}



Note: 1. The HIT signal is cleared by writing to the Compare Control register.

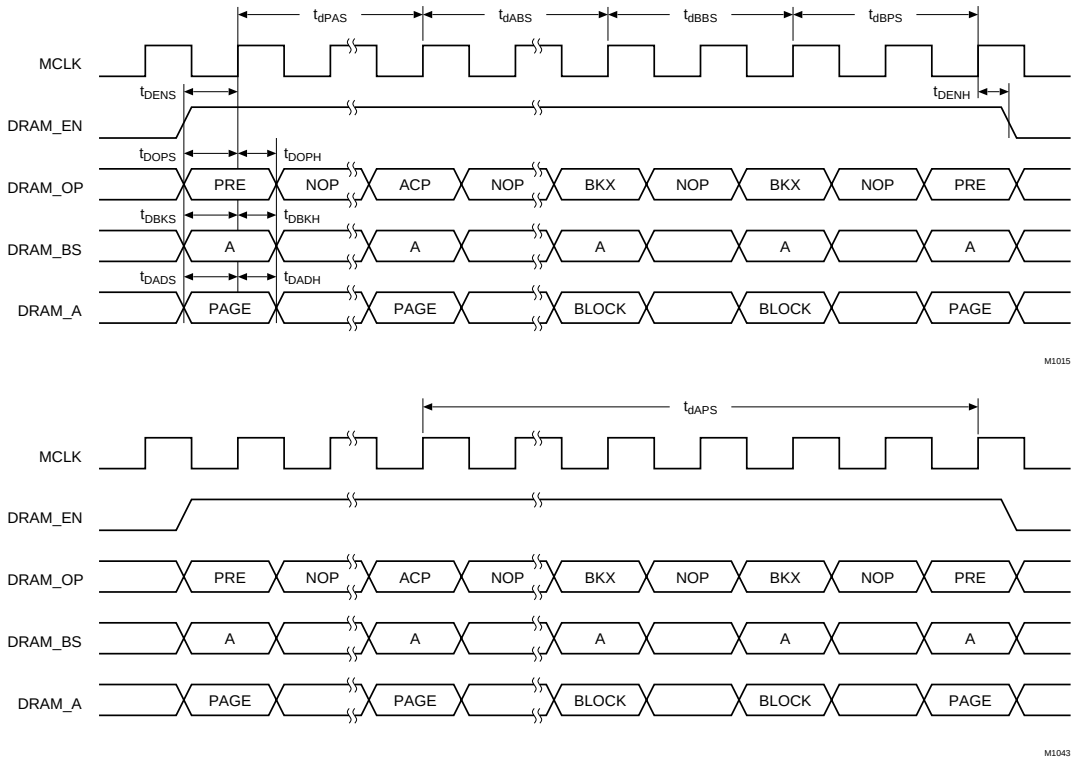
2. The HIT signal can be set by the comparison result from the PASS_IN and PASS_OUT pins, which are generated two cycles before the HIT signal is.

Figure 8.33 Picking Logic timing

Table 8.33 Picking Logic timing parameter

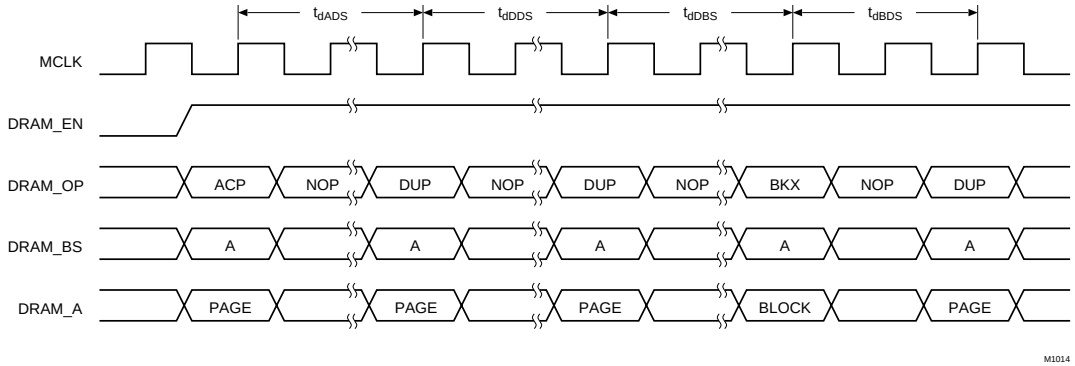
Symbol	Parameter	M5M410092B				Unit	Refer
		-10		-13			
		Min	Max	Min	Max		
t_{CHT}	MCLK to valid $\overline{HIT}$	—	35	—	35	ns	t_{11}

This page is intentionally left blank.



Note: BKX means any block transfer operation, such as UWB, MWB, or RDB.

Figure 8.34 DRAM operations on the same bank (1)



Note: BKX means any block transfer operation, such as UWB, MWB, or RDB.

Figure 8.35 DRAM operations on the same bank (2)

Table 8.34 Minimum requirements of the DRAM port interface timing parameters

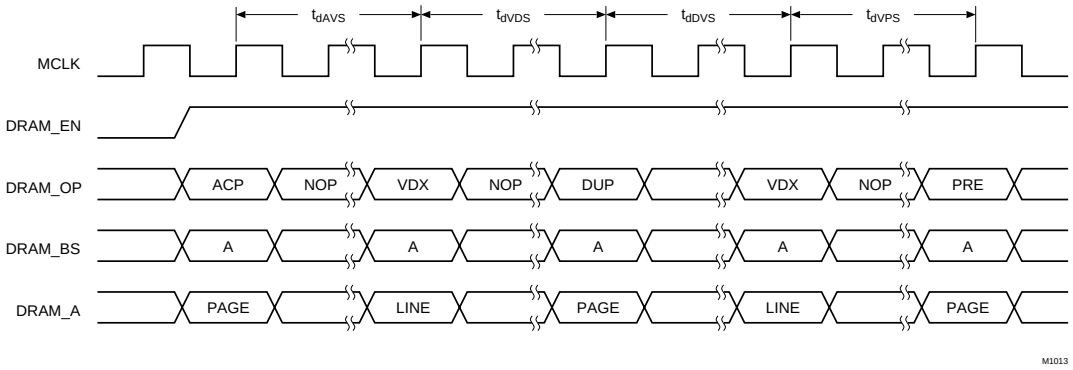
Symbol	Parameter	M5M410092B		Unit	Refer	Timing Figure
		-10	-13			
t_{DENS}	DRAM_EN setup time	3	4	ns	t_8	8.7
t_{DENH}	DRAM_EN hold time	0	0	ns	t_9	8.7
t_{DOPS}	DRAM_OP setup time	3	4	ns	t_8	8.7
t_{DOPH}	DRAM_OP hold time	0	0	ns	t_9	8.7
t_{DBKS}	DRAM_BS setup time	3	4	ns	t_8	8.7
t_{DBKH}	DRAM_BS hold time	0	0	ns	t_9	8.7
t_{DADS}	DRAM_A setup time	3	4	ns	t_8	8.7
t_{DADH}	DRAM_A hold time	0	0	ns	t_9	8.7

Table 8.35 Minimum requirements of the DRAM interlock timings for operations on same bank

Symbol	Parameter	M5M410092B		Unit	Timing Figure
		-10	-13		
t_{dABS}	Access Page to Block Transfer	40	40	ns	8.7
t_{dAPS}[a]	Access Page to Precharge Bank	80	104	ns	8.7
t_{dADS}	Access Page to Duplicate Page	40	52	ns	8.8
t_{dBBS}	Block Transfer to Block Transfer	20	26	ns	8.7
t_{dBPS}	Block Transfer to Precharge Bank	20	26	ns	8.7
t_{dBDS}	Block Transfer to Duplicate Page	20	26	ns	8.8
t_{dPAS}[b]	Precharge Bank to Access Page	40	52	ns	8.7
t_{dDBS}	Duplicate Page to Block Transfer	80	104	ns	8.8
t_{dDDS}	Duplicate Page to Duplicate Page	80	104	ns	8.8

a. The maximum timing limit from ACP to PRE is 100,000 ns.

b. The operation from PRE to ACP requires at least two clock cycles. At the first clock rising edge, PRE starts. At the second clock rising edge, the preparation for ACP starts.



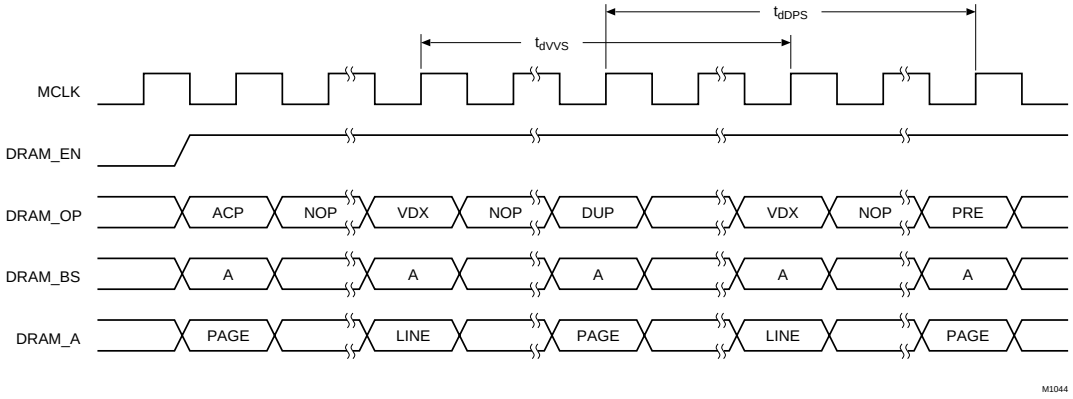
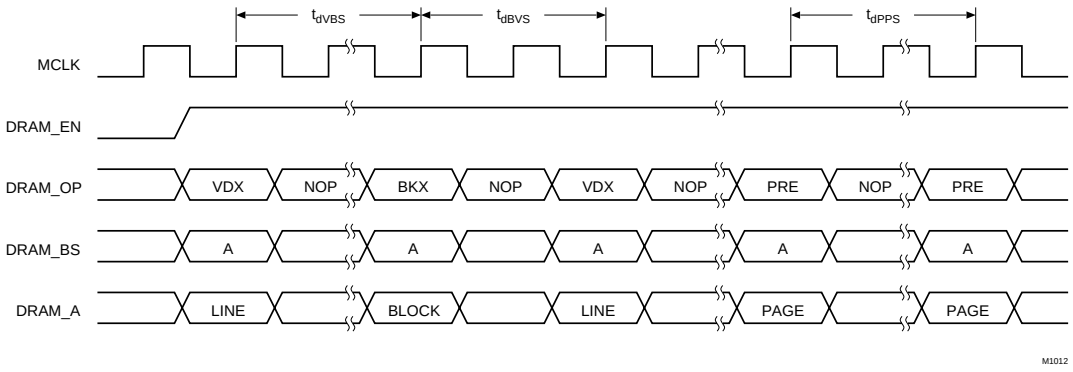


Figure 8.36 DRAM operations on the same bank (3)

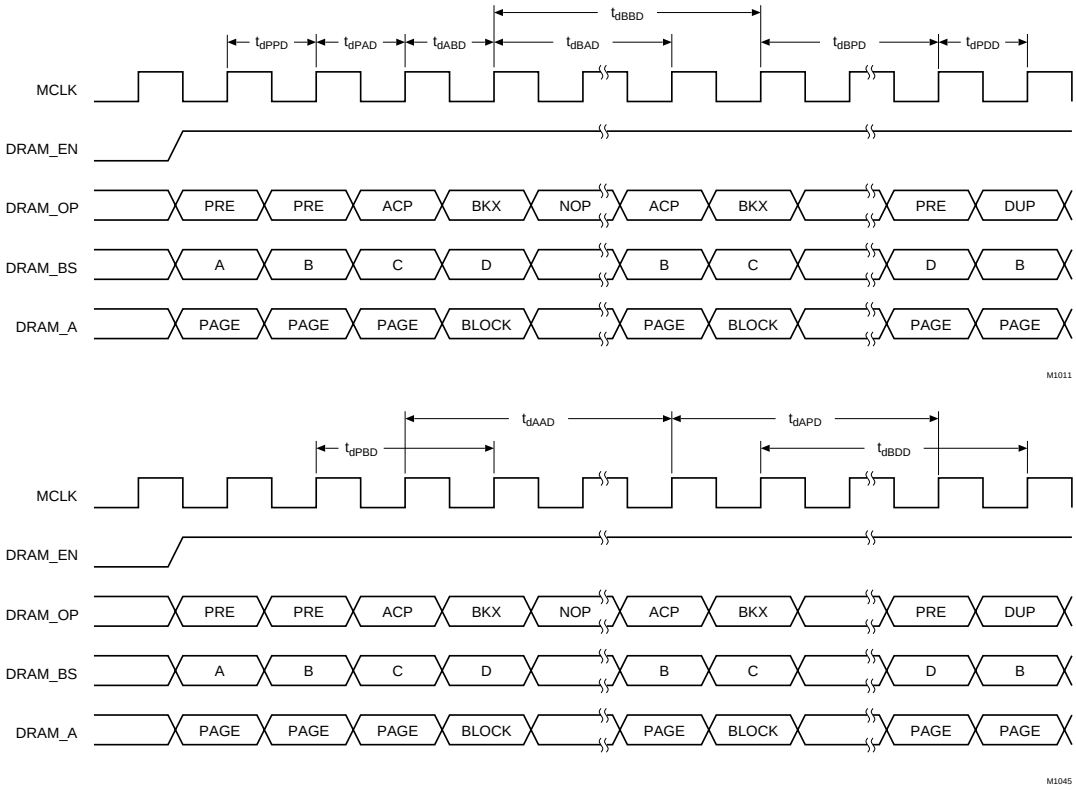


Note: BKX means any block transfer operation, such as UWB, MWB, or RDB.

Figure 8.37 DRAM operations on the same bank (4)

Table 8.36 Minimum requirements of the DRAM interlock timings for operations on same bank

Symbol	Parameter	M5M410092B		Unit	Timing Figure
		-10	-13		
t_{dAVS}	Access Page to Video Transfer	40	52	ns	8.9
t_{dBVS}	Block Transfer to Video Transfer	20	26	ns	8.10
t_{dPPS}	Precharge Bank to Precharge Bank	40	52	ns	8.10
t_{dDPS}	Duplicate Page to Precharge Bank	80	104	ns	8.9
t_{dDVS}	Duplicate Page to Video Transfer	80	104	ns	8.9
t_{dVBS}	Video Transfer to Block Transfer	40	52	ns	8.10
t_{dVPS}	Video Transfer to Precharge Bank	40	52	ns	8.9
t_{dVDS}	Video Transfer to Duplicate Page	40	52	ns	8.9
t_{dVVS}	Video Transfer to Video Transfer	80	104	ns	8.9



Note: BKX means any block transfer operation, such as UWB, MWB, or RDB.

Figure 8.38 DRAM operations between two different banks (1)

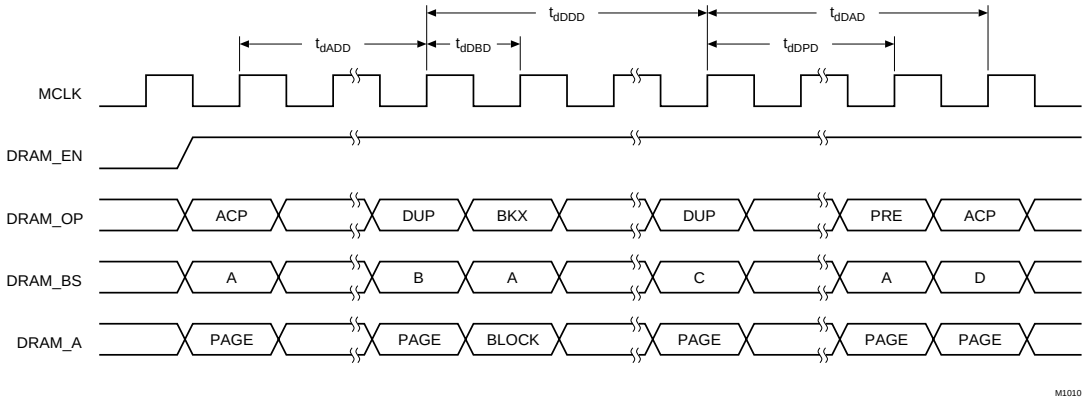


Figure 8.39 DRAM operations between two different banks (2)

Table 8.37 Minimum requirement of the DRAM interlock timings for operations between two different banks

Symbol	Parameter	M5M410092B		Unit	Timing Figure
		-10	-13		
t_{dAAD}	Access Page to Access Page	40	52	ns	8.11
t_{dABD}	Access Page to Block Transfer	10	13	ns	8.11
t_{dAPD}	Access Page to Precharge Bank	40	52	ns	8.11
t_{dADD}	Access Page to Duplicate Page	40	52	ns	8.12
t_{dBAD}	Block Transfer to Access Page	10	13	ns	8.11
t_{dBBD}	Block Transfer to Block Transfer	20	26	ns	8.11
t_{dBPD}	Block Transfer to Precharge Bank	10	13	ns	8.11
t_{dBDD}	Block Transfer to Duplicate Page	10	13	ns	8.11
t_{dPAD}	Precharge Bank to Access Page	10	13	ns	8.11
t_{dPBD}	Precharge Bank to Block Transfer	10	13	ns	8.11
t_{dPPD}	Precharge Bank to Precharge Bank	10	13	ns	8.11
t_{dPDD}	Precharge Bank to Duplicate Page	10	13	ns	8.11
t_{dDAD}	Duplicate Page to Access Page	80	104	ns	8.12
t_{dDBD}	Duplicate Page to Block Transfer	10	13	ns	8.12
t_{dDPD}	Duplicate Page to Precharge Bank	40	52	ns	8.12
t_{dDDD}	Duplicate Page to Duplicate Page	80	104	ns	8.12

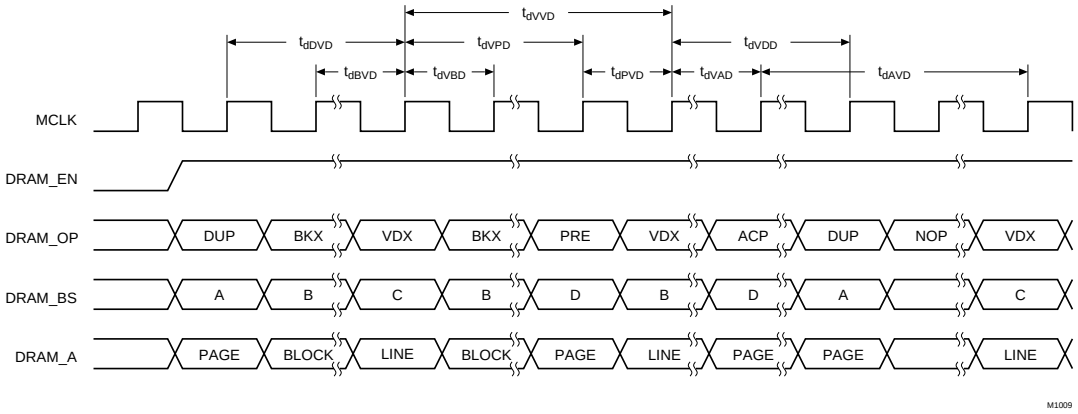
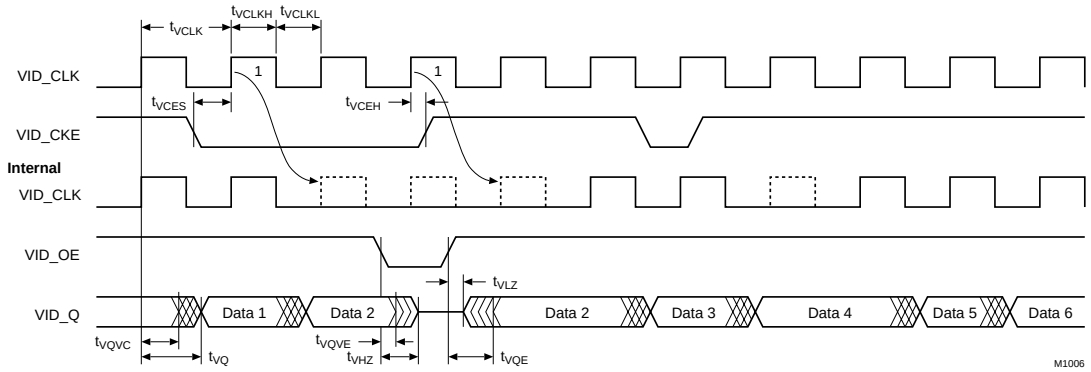


Figure 8.40 DRAM operations between two different banks (3)

Table 8.38 Minimum requirement of the DRAM interlock timings for operations between two different banks

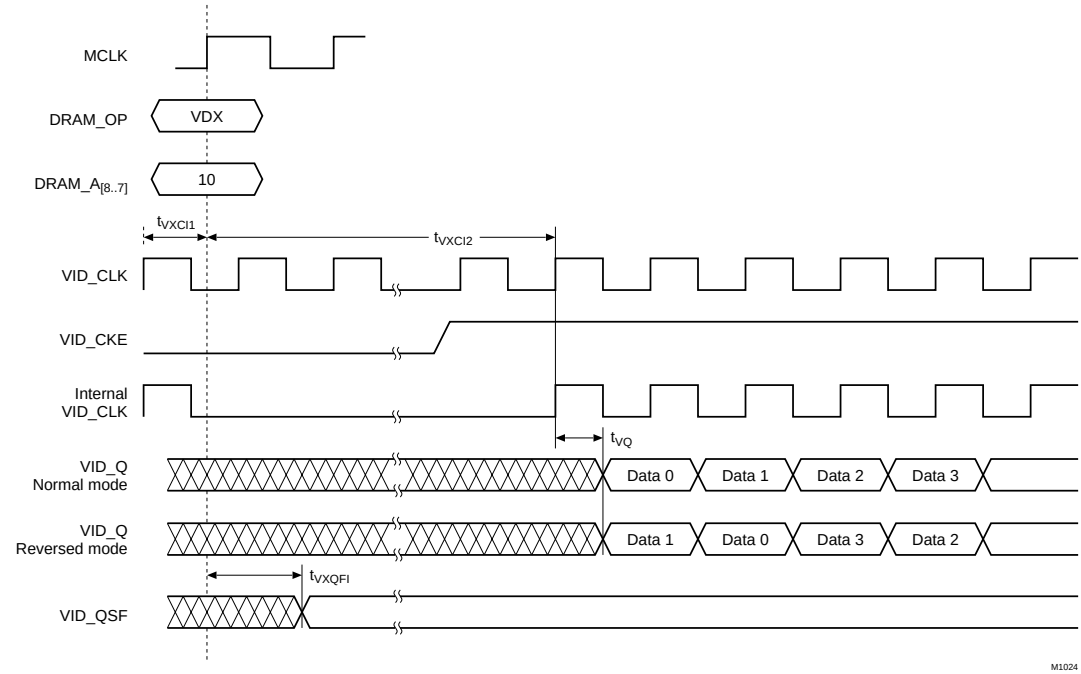
Symbol	Parameter	M5M410092B		Unit
		-10	-13	
t_{dAVD}	Access Page to Video Transfer	40	52	ns
t_{dBVD}	Block Transfer to Video Transfer	10	13	ns
t_{dPVD}	Precharge Bank to Video Transfer	10	13	ns
t_{dDVD}	Duplicate Page to Video Transfer	80	104	ns
t_{dVAD}	Video Transfer to Access Page	40	52	ns
t_{dVBD}	Video Transfer to Block Transfer	10	13	ns
t_{dVPD}	Video Transfer to Precharge Bank	40	52	ns
t_{dVDD}	Video Transfer to Duplicate Page	40	52	ns
t_{dVVD}	Video Transfer to Video Transfer	80	104	ns

This page is intentionally left blank.



Note: 1. The deassertion of VID_CKE at the current VID_CLK rising edge will mask out the next internal VID_CLK cycle.
2. Timings are measured from the VID_CLK pin or from the VID_OE pin.

Figure 8.41 Internal VID_CLK and video output timing

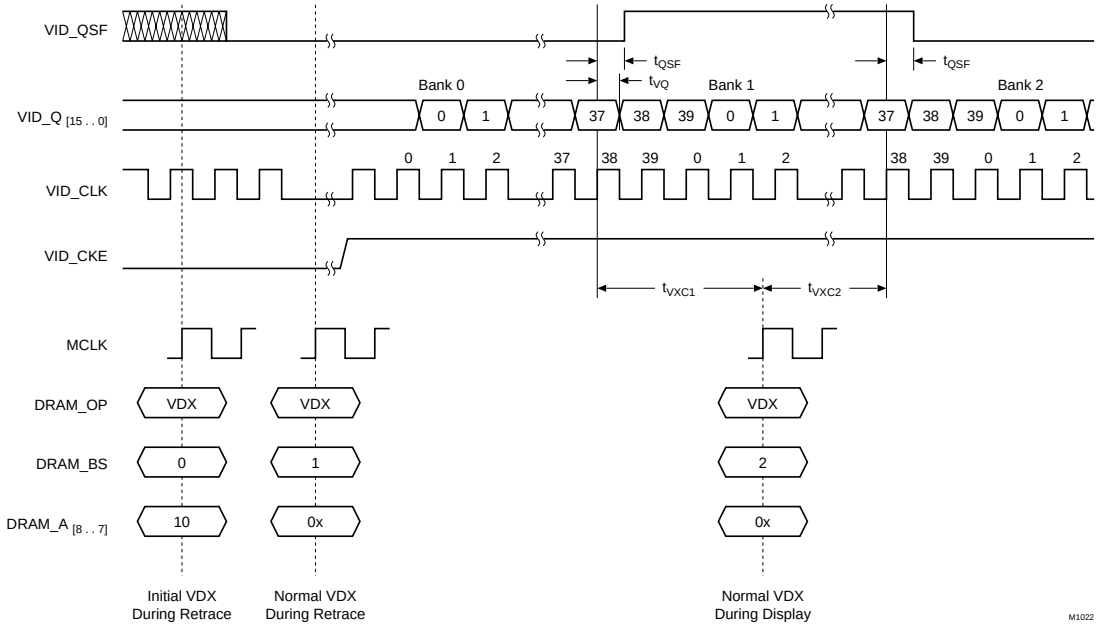


Note: 1. Note that the VID_OE is always “1” for video output enable is assumed.
2. Timings are measured from the MCLK pin or from the VID_CLK pin.

Figure 8.42 Video output sequence from intial VDX for normal and reversed modes

Table 8.39 Video Buffer timing parameters

Symbol	Parameter	M5M410092B				Unit	Refer	Timing Figure
		-10		-13				
		Min	Max	Min	Max			
t_{VCLK}	VID_CLK cycle time	12	—	14	—	ns	t_1	8.14
t_{VCLKH}	VID_CLK high pulse width	5	—	6	—	ns	t_2	8.14
t_{VCLKL}	VID_CLK low pulse width	5	—	6	—	ns	t_3	8.14
t_{VCES}	VID_CKE setup time	4	—	4	—	ns	t_8	8.14
t_{VCEH}	VID_CKE hold time	0	—	0	—	ns	t_9	8.14
t_{VQ}	VID_Q access time from VID_CLK	—	8	—	10	ns	t_{11}	8.14
t_{VQVC}	VID_Q valid after VID_CLK	3	—	3	—	ns	t_{12}	8.14
t_{VLZ}	VID_Q output low impedance	3	—	3	—	ns	t_{16}	8.14
t_{VHZ}	VID_Q output high impedance	—	3	—	3	ns	t_{15}	8.14
t_{VQE}	VID_Q access time from VID_OE high	—	9	—	11	ns	t_{17}	8.14
t_{VQVE}	VID_Q valid after VID_OE low	—	—	—	—	ns	t_{14}	8.14
t_{VXCI1}	Initial VDX after last internal VID_CLK	14	—	14	—	ns	—	8.15
t_{VXCI2}	Initial VDX before next internal VID_CLK	80	—	80	—	ns	—	8.15
t_{VXQFI}	VID_QSF delay time after initial VDX	—	80	—	80	ns	t_{11}	8.15



Note: 1. t_{VXC1} specifies the earliest allowed normal VDX.
2. t_{VXC2} specifies the latest allowed normal VDX.

Figure 8.43 Continuous video output sequence in normal mode during display

Table 8.40 Video Buffer timing parameters

Symbol	Parameter	M5M410092B				Unit	Refer
		-10		-13			
		Min	Max	Min	Max		
t_{QSF}	VID_QSF delay time from internal VID_CLK 38	—	25	—	25	ns	t_{11}
t_{VXC1}	Normal VDX after internal VID_CLK 38	20	—	20	—	ns	—
t_{VXC2}	Normal VDX before next internal VID_CLK 38	60	—	60	—	ns	—

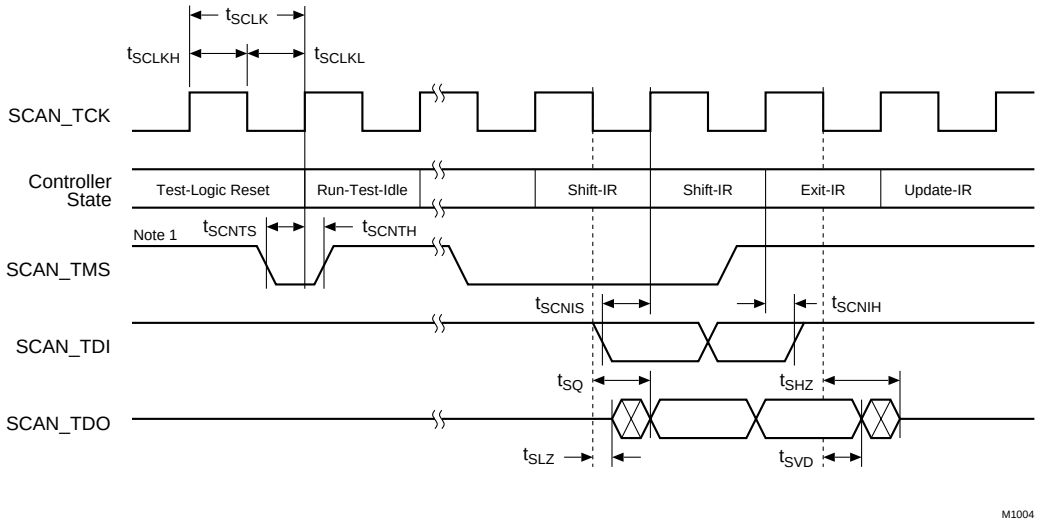


Figure 8.44 Boundary scan

Table 8.41 Boundary-Scan timing parameters

Symbol	Parameter	M5M410092B -10, -13		Unit	Refer
		Min	Max		
t_{SCLK}	SCAN_TCK cycle time	100	—	ns	t_1
t_{SCLKH}	SCAN_TCK high pulse width	40	—	ns	t_2
t_{SCLKL}	SCAN_TCK low pulse width	40	—	ns	t_3
t_{SCNTS}	SCAN_TMS setup time	8	—	ns	t_8
t_{SCNTH}	SCAN_TMS hold time	26	—	ns	t_9
t_{SCNIS}	SCAN_TDI setup time	8	—	ns	t_8
t_{SCNIH}	SCAN_TDI hold time	26	—	ns	t_9
t_{SLZ}	SCAN_TCK to SCAN_TDO low impedance	—	20	ns	t_{18}
t_{SQ}	SCAN_TDO access time	—	26	ns	t_{19}
t_{SVD}	SCAN_TDO data valid time	8	—	ns	t_{20}
t_{SHZ}	SCAN_TCK to SCAN_TDO high impedance	—	20	ns	t_{21}

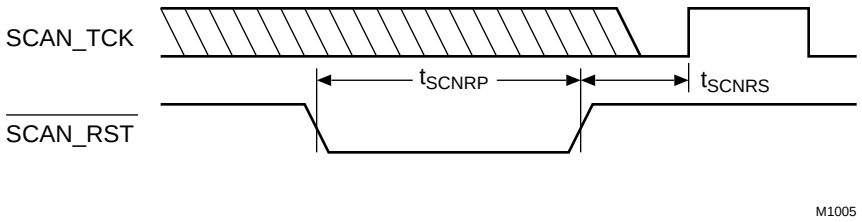


Figure 8.45 Boundary scan reset

Table 8.42 Boundary-Scan reset timing parameters

Symbol	Parameter	M5M410092B -10, -13		Unit	Refer
		Min	Max		
t_{SCNRS}	$\overline{SCAN\_RST}$ setup time	8	—	ns	t_6
t_{SCNRP}	$\overline{SCAN\_RST}$ pulse width	30	—	ns	t_7

9 Packaging

Packaging

The 3D-RAM is housed in 128-pin QFP(FP) and reverse QFP(RF) packages.

For the purpose of convenient reference, the pinout diagrams for the 3D-RAM are repeated on pages 153 and 154.

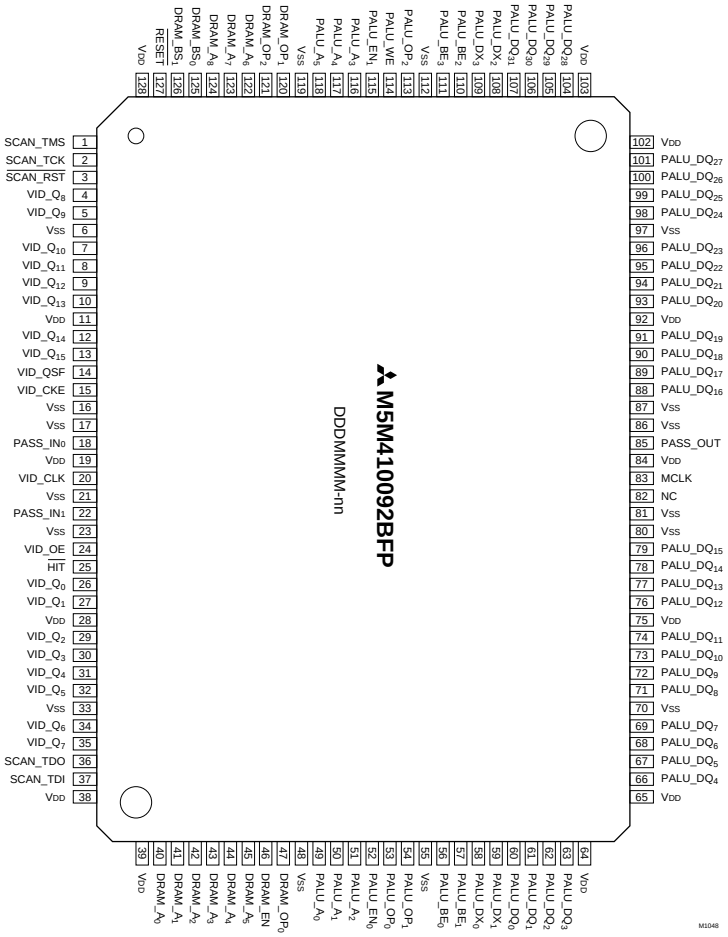
Page 103 contains the mechanical specification for the FP and RF packages.

The thermal characteristics data for both packages is on page 158.

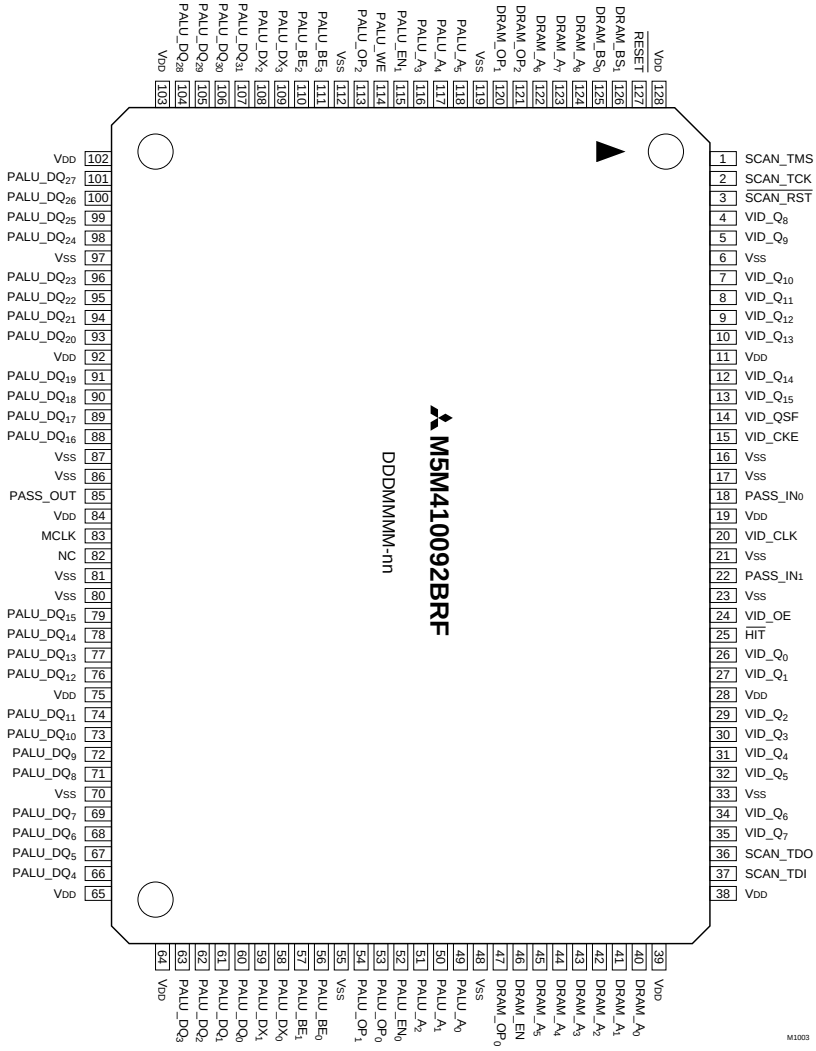
3D-RAM Pinouts

There are two pinouts for 3D-RAM: normal pinout with pin 1 located at the lower left hand corner and specially marked by a small circle; and reverse pinout with pin 1 located at the upper left hand corner and marked by a large circle and a pointing triangle. The device in normal pinout is designated by the letters “FP” in the product number, and the device in reverse pinout by the letters “RF.” In both pinouts, the mapping of pin numbers with pin names is identical.

Normal Pinout Diagram



Reverse Pinout Diagram



Tracking Label

On the top surface of the 3D-RAM package, a tracking label is printed below the Mitsubishi logo and the 3D-RAM product number. The tracking label consists of 7 numbers followed by a dash and a speed/power grade designation, and is represented by the mnemonic “DDDMMMM-nn”. This mnemonic is explained as below:

DDD: Date code
MMMM: Manufacturing code
nn: One of the following speed designations:
“10” — t_{CLK} (min) = 10 ns
“13” — t_{CLK} (min) = 13 ns

Mechanical Drawing for 128-pin FP and RF Packages

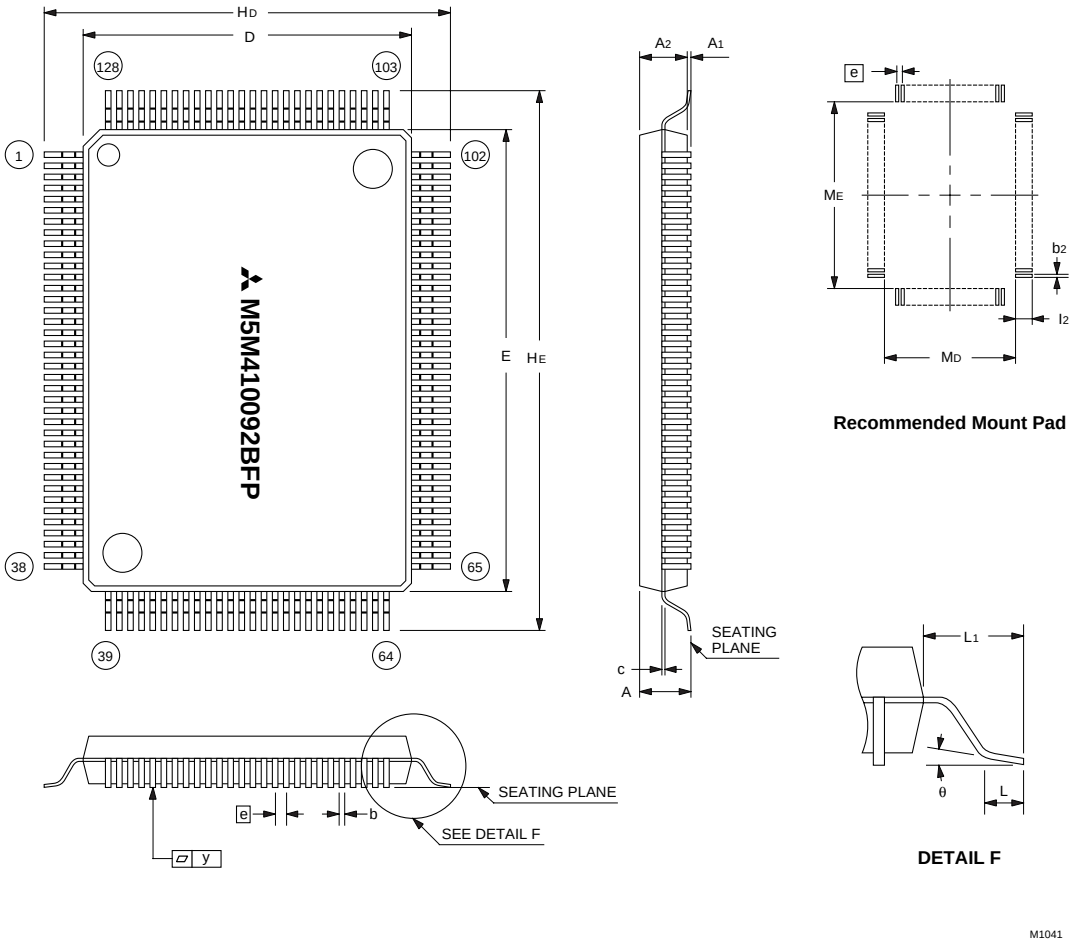


Figure 9.46 128-pin FP package drawing

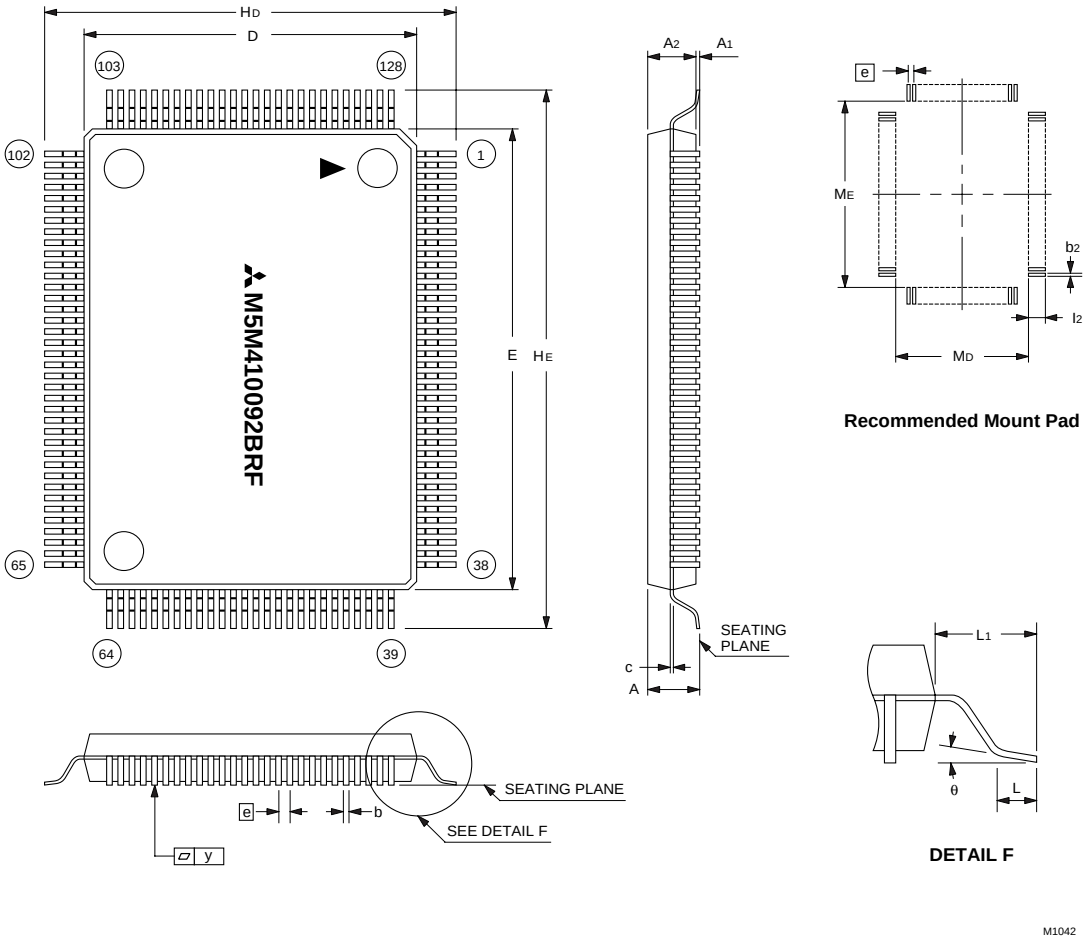


Figure 9.47 128-pin RF package drawing

Table 9.43 Package drawing parameters in millimeters

Description	Symbol	Dimension in Millimeters		
		Min	Nominal	Max
Mounting height	A	—	—	1.7
Stand-off height	A1	0.05	0.15	0.25
Package height	A2	—	1.4	—
Terminal width	b	0.13	0.18	0.28
Terminal thickness	c	0.105	0.125	0.175
Package length	D	13.9	14.0	14.1
Package width	E	19.9	20.0	20.1
Linear spacing between terminals	e	—	0.5	—
Over length	HD	15.8	16.0	16.2
Over width	HE	21.8	22.0	22.2
Length of the flat portion of terminal	L	0.3	0.3	0.7
Terminal length	L1	—	1.0	—
Flatness of terminal	y	—	—	0.1
Terminal angle	q	0°	—	10°
Mount pad dimensions	b2	—	0.225	—
Mount pad dimensions	I2	—	1.0	—
Mount pad dimensions	MD	—	14.4	—
Mount pad dimensions	ME	—	20.4	—

Thermal Characteristics

The maximum junction temperature (T_{jc}) is set to 125 °C. The junction temperature can be calculated with the following equation:

$$T_{jc}\ (°C) = \theta_{ja}\ (°C/W) \times P\ (W) + T_a\ (°C)$$

where θ_{ja} is the junction-to-ambient thermal resistance, P is the whole chip power dissipation, and T_a is the ambient temperature.

Thermal Resistance for Single Package

Four cases of the thermal resistance for single package are listed in Table 9.2. These four cases are package only, package on PCB, package on PCB with thermal compound, and package on PCB with fin. Figures 9.3 through 9.7 show the detailed conditions of these four cases.

Table 9.44 Thermal resistance for single package

Airflow	Case 1 Package only θja (°C/W)	Case 2 Package on PCB θja (°C/W)	Case 3 Package on PCB with compound θja (°C/W)	Case 4 Package on PCB with fin θja (°C/W)
0; or natural convection	151.0	73.0	62.3	41.2
100 ft/min or 0.5 m/s	89.0	55.0	45.6	30.6
200 ft/min or 1.0 m/s	70.5	48.2	39.7	25.7
400 ft/min or 2.0 m/s	52.0	40.1	33.1	20.2
1000 ft/min or 5.0 m/s	36.1	31.6	26.1	15.3

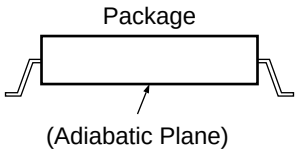


Figure 9.48 Case 1 condition: package only

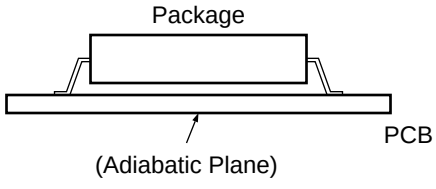


Figure 9.49 Case 2 condition: package on PCB

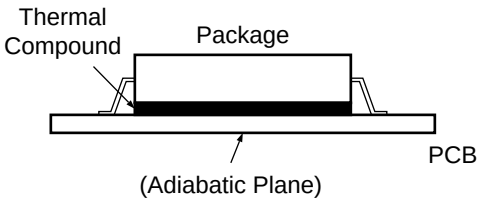


Figure 9.50Case 3 condition: package on PCB with thermal compound

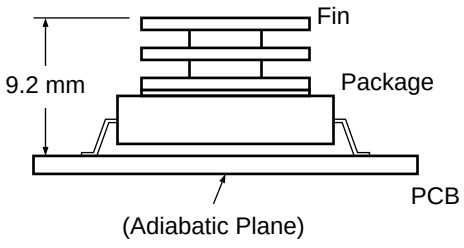
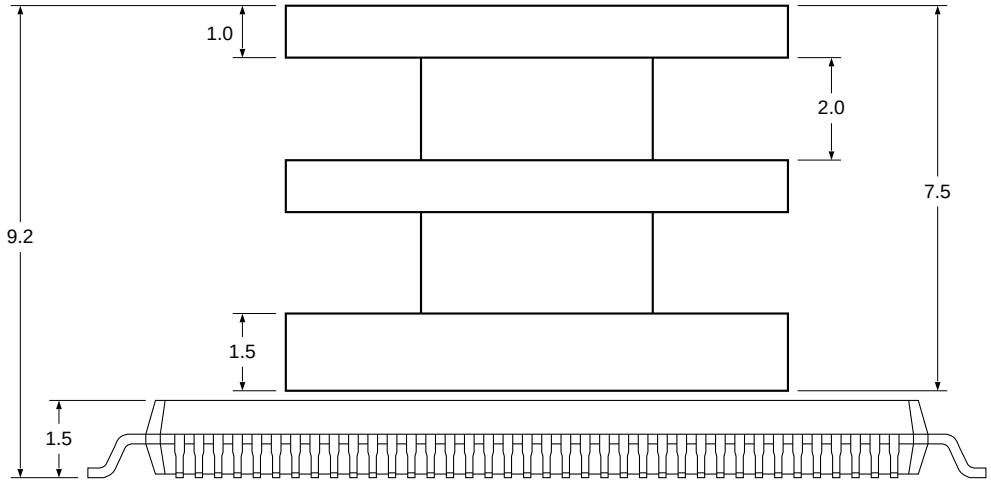


Figure 9.51Case 4 condition: package on PCB with fin



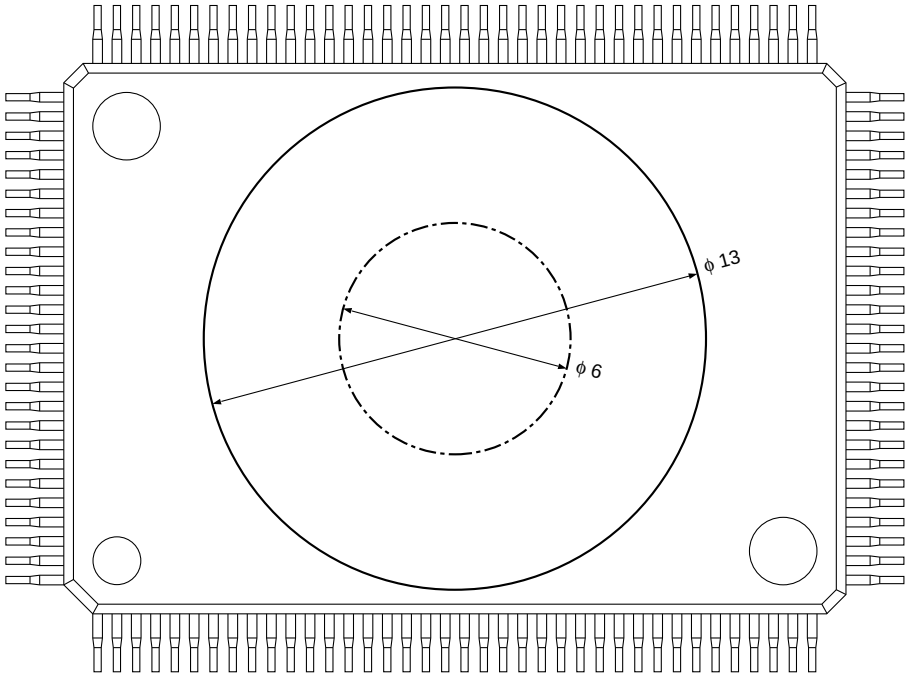


Figure 9.52 Mechanical drawing of the fin

Thermal Resistance for Twelve Packages Mounted on PCB

Table 9.3 lists the thermal resistance for 12 packages mounted on two sides of a PCB. Two cases are listed: without heat sink and with aluminum plate for heat sink.

Table 9.45 Thermal resistance for 12 packages mounting double-sided on PCB

Airflow	Without Heat Sink θja (°C/W)	With Heat Sink θja (°C/W)
0 m/s or natural convection	73.3	38.3
100 ft/min or 0.5 m/s	62.5	24.2
200 ft/min or 1.0 m/s	50.7	19.8
400 ft/min or 2.0 m/s	37.5	15.1
1000 ft/min or 5.0 m/s	26.3	11.4

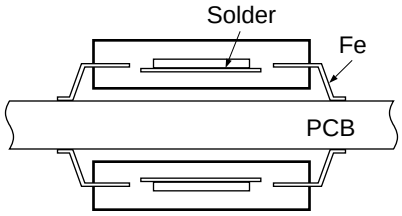


Figure 9.53 Multiple packages double-side mounted on PCB, without heat sink

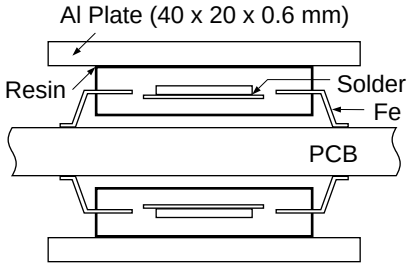


Figure 9.54 Multiple packages double-side mounted on PCB, with heat sink

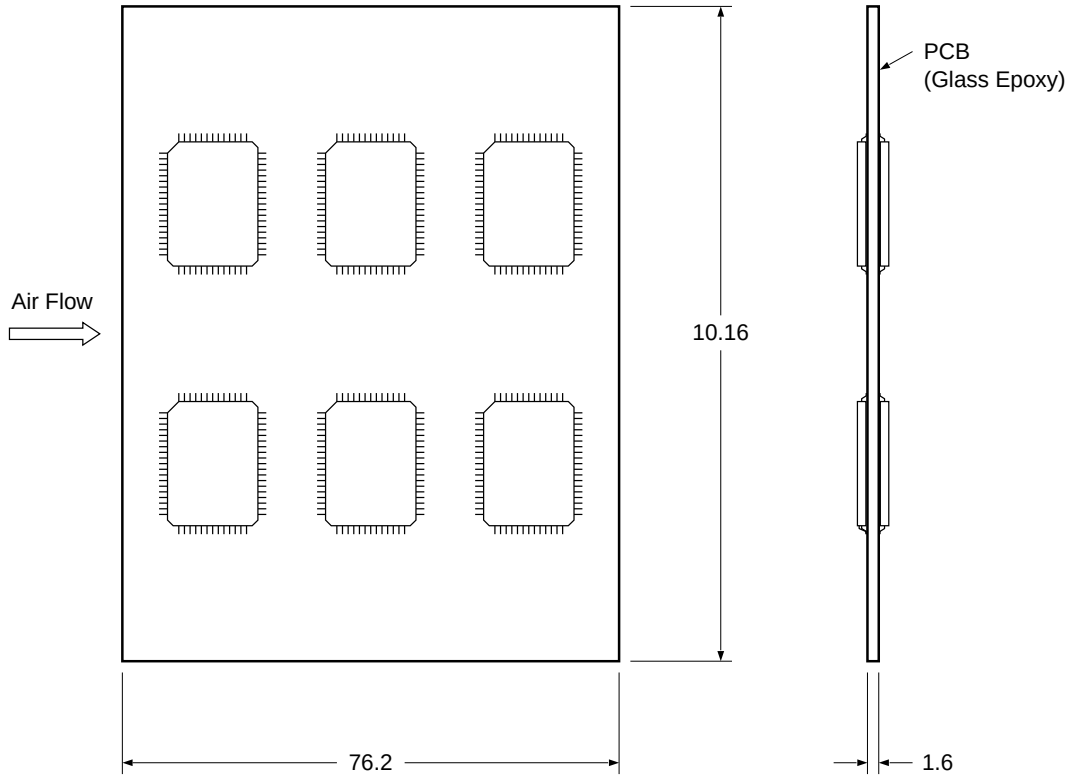


Figure 9.55 Mechanical drawing of the mounting, without heat sink

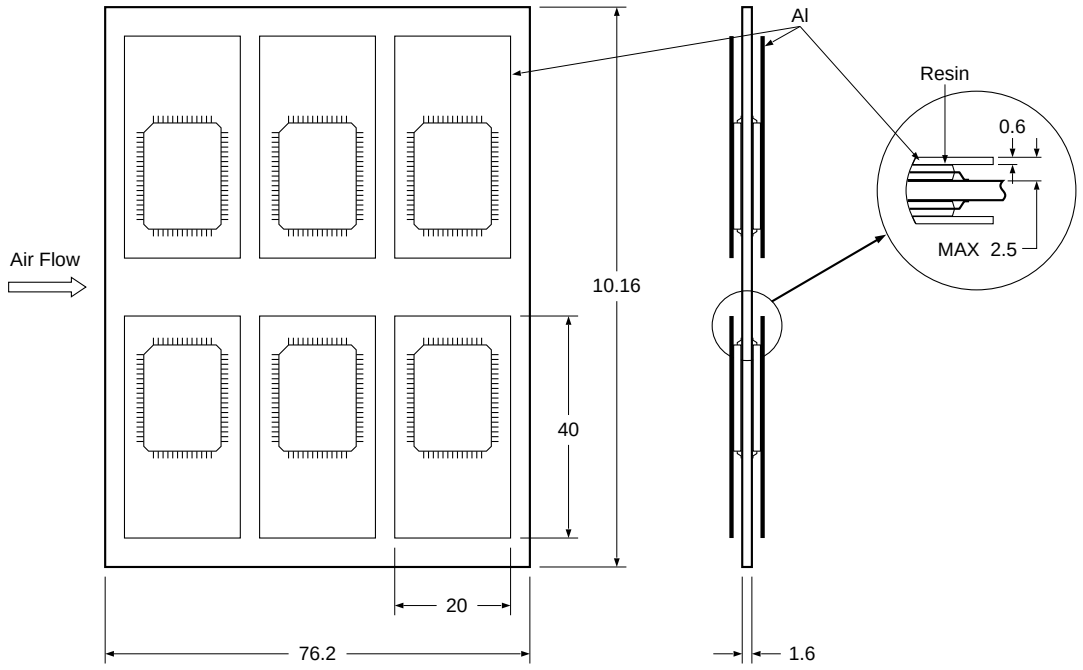


Figure 9.56 Mechanical drawing of the mounting, with heat sink

10 JTAG Boundary Scan

JTAG Boundary Scan

Boundary-Scan Architecture

The 3D-RAM provides test features that are partially in compliance with the IEEE Standard 1149.1 Test Access Port and Boundary-Scan Architecture. The on-chip test logic provides a standardized approach for checking the interconnections between different components on the same printed circuit board.

The boundary-scan test logic consists of the boundary-scan register and support logic. The test function is accessed through the Test Access Port (TAP). The TAP provides a simple serial interface that allows testing of all signal traces with only five pins in the Serial Test Port.

Inside the 3D-RAM, the boundary-scan cells for the signal pins are interconnected to form a shift-register chain around the pads. This path has serial input and output connections with scan clock and control signals. On a printed circuit board, the boundary-scan registers for the individual components can be connected in series to form a single path through the whole board, as illustrated in Figure 10.1. Alternatively, a board design could contain several independent boundary-scan paths.

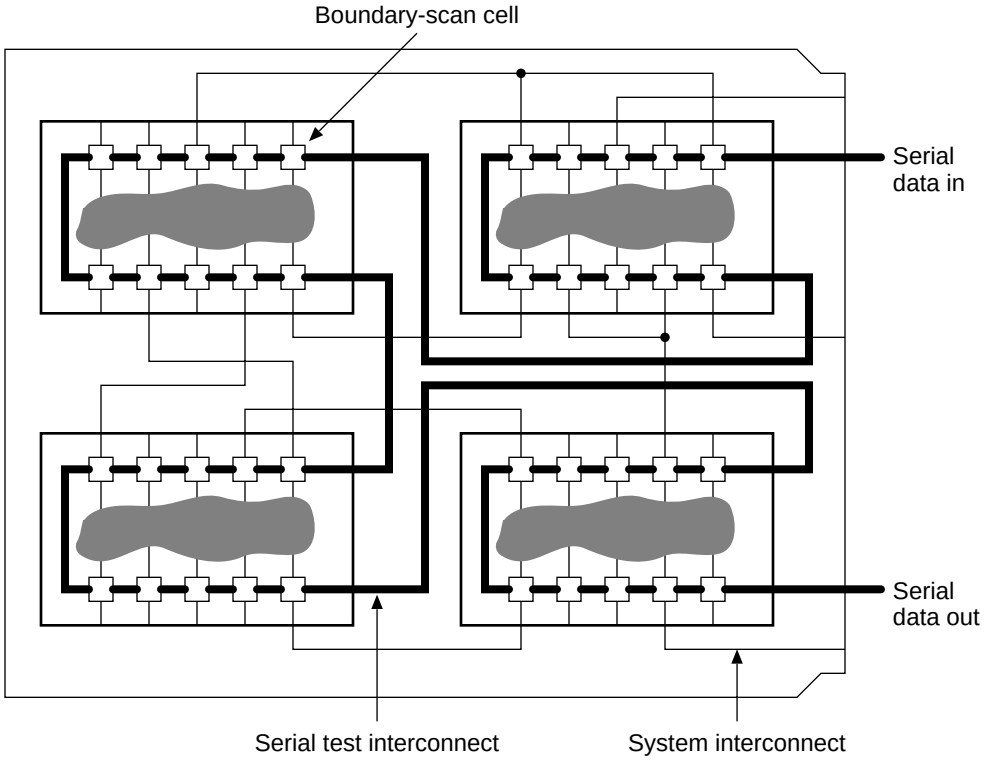


Figure 10.1 A boundary-scannable board design

The boundary-scan test logic contains the following elements:

- Test Access Port (TAP), consisting of input pins SCAN_TMS, SCAN_TCK, SCAN_RST and SCAN_TDI, and an output pin SCAN_TDO
- TAP Controller, which interprets the inputs on the test mode select line (SCAN_TMS) and performs the corresponding operations, such as controlling the scan instruction and data registers within the 3D-RAM
- Instruction Register (IR), which accepts instruction codes shifted through the test data input (SCAN_TDI) pin
- Two test Data Registers: Bypass Register (BPR) and Boundary-Scan Register (BSR)

The instruction and test data registers are separate shift-register paths connected in parallel and have a common serial data input (SCAN_TDI) and a common serial data output (SCAN_TDO). The data flow is controlled by the TAP controller signals. A block diagram of the boundary-scan architecture is shown in Figure 10.2.

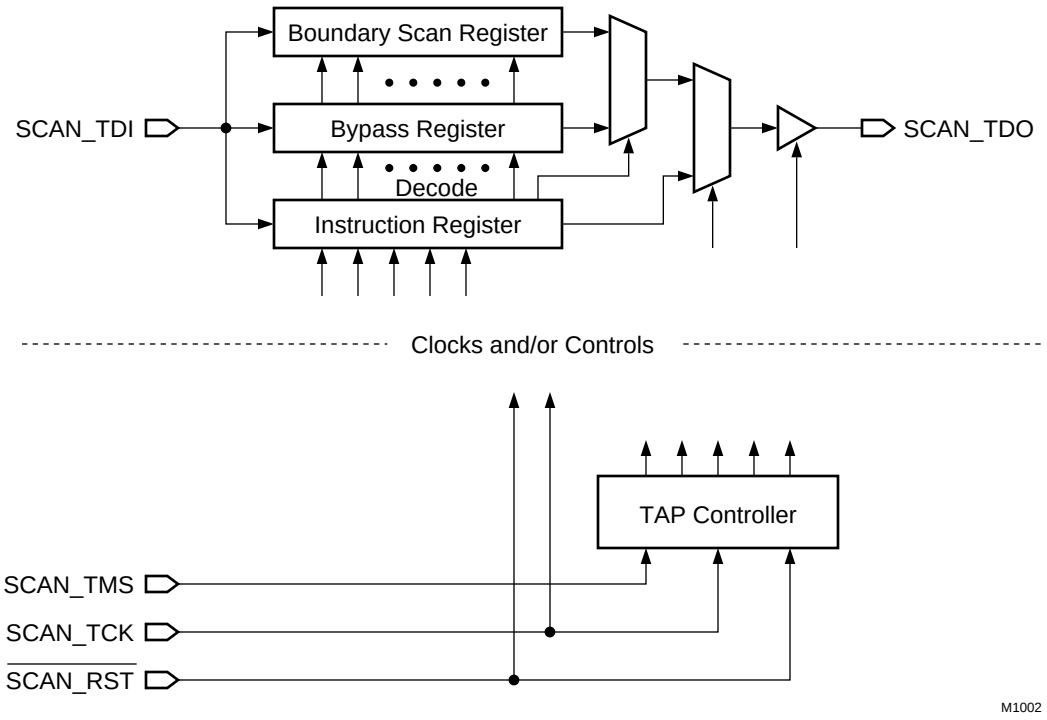


Figure 10.2 Block diagram of the boundary scan architecture

The TAP Controller

The TAP controller is a synchronous finite state machine controlling the sequence of test logic operations. The TAP controller changes state at the rising edge of the SCAN_TCK pin. The SCAN_TMS pin controls the sequence of the state changes. A state diagram for the TAP controller is shown in Figure 10.3.

The TAP controller is initialized either after power up or when SCAN_RST is asserted low. In addition, it can be initialized by applying a high signal level on the SCAN_TMS input for five SCAN_TCK cycles.

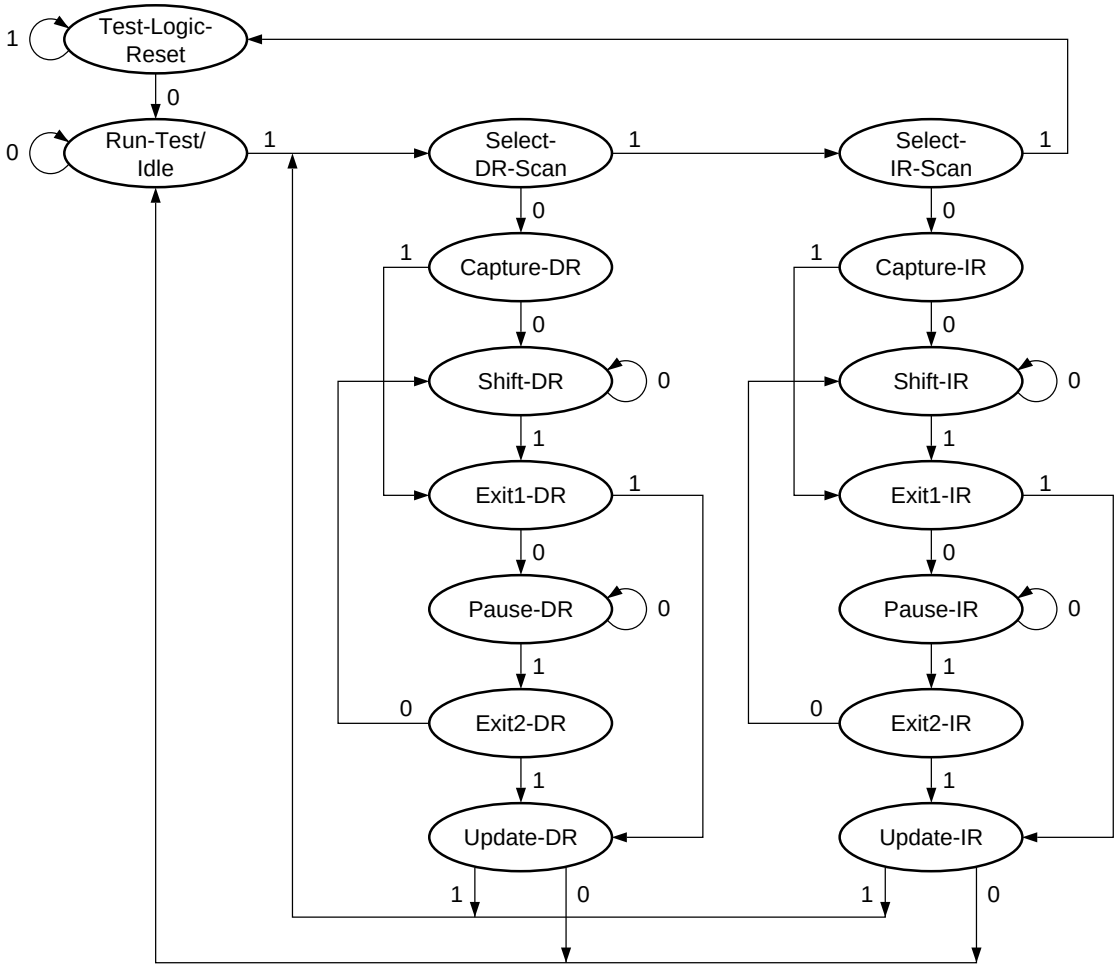


Figure 10.3 TAP controller state diagram

Test-Logic-Reset State

The Instruction Register is set to the default Bypass instruction, so that normal 3D-RAM operations can proceed without interference. The TAP controller enters this state when it is initialized after power-up or by the reset signal $\overline{\text{SCAN\_RST}}$. Regardless of the original state, the controller enters this state when the SCAN_TMS input is held high for at least five rising SCAN_TCK cycles.

Run-Test/Idle State

This is an idle state. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Select-DR-Scan State

This is a temporary controller state. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Capture-DR State

The Boundary-Scan Register captures input data from the SCAN_TDI pin if the current instruction is Extest or Sample/Preload. The Bypass Register does not change. The current instruction does not change in this state.

Shift-DR State

The test data register selected by the current instruction shifts data one stage toward SCAN_TDO on each rising edge of SCAN_TCK. The current instruction does not change in this state.

Exit1-DR State

This is a temporary controller state. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Pause-DR State

The Pause-DR state allows the data shifting through the test data register to be temporarily halted. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Exit2-DR State

This is a temporary controller state. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Update-DR State

The boundary-scan cell for output pad has a latch to prevent changes at the parallel output while data is shifting along the boundary-scan chain. When the TAP controller is in this state and the Boundary-Scan Register is selected, data is latched from the shift-register path on the falling edge of SCAN_TCK. The data held at the latch does not change other than in this state. All shift-register stages in selected test data register retain their previous values during this state. The current instruction does not change in this state.

Select-IR-Scan State

This is a temporary controller state. The test data register selected by the current instruction retains its previous state. The current instruction does not change in this state.

Capture-IR State

The shift-register contained in the Instruction Register is loaded with the fixed value “1001” on the rising edge of SCAN_TCK. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Shift-IR State

The shift-register contained in the Instruction Register is connected between the SCAN_TDI and SCAN_TDO pins. The shift-register shifts data one stage towards its serial output on each rising edge of SCAN_TCK. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Exit1-IR State

This is a temporary state. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Pause-IR State

The Pause-IR state allows the data shifting through the Instruction Register to be temporarily halted. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Exit2-IR State

This is a temporary state. The test data register selected by the current instruction retains its previous value during this state. The current instruction does not change in this state.

Update-IR State

The instruction shifted into the Instruction Register is latched from the shift-register path on the falling edge of SCAN_TCK. The test data register selected by the current instruction retains its previous value during this state. Once the new instruction has been latched, it becomes the current instruction.

Test Data Register

The 3D-RAM contains the two required test data registers: Bypass Register and Boundary-Scan Register. Both registers are connected to the SCAN_TDI and SCAN_TDO pins. When a register is selected by the current instruction, the data in its shift-register is shifted one stage towards the SCAN_TDO output pin on each rising edge of the SCAN_TCK pin.

Bypass Register

The Bypass Register is a one-bit shift-register that provides the minimal length path between the SCAN_TDI and SCAN_TDO pins. When the 3D-RAM is not required to perform scan test operation, this path can be selected to allow rapid movement of test data to and from other components on the board.

Boundary-Scan Register

The Boundary-Scan Register is a shift-register path containing the boundary-scan cells that are connected to all input/output signal pins of the 3D-RAM except the following pins: MCLK, PASS_IN, PASS_OUT, and VID_CLK. The boundary-scan cells of the $PALU\_DQ_{[31:0]}$ pins are implemented as input pins only. Figure 10.4 shows the logical structure of the Boundary-Scan Register. While output cells determine the value of the signal driven on the corresponding pin, input cells only capture data. The output cell has a latch connected to the shift-register for latching the data in Update-DR state. These operations do not affect the normal operations of the device. Data is shifted from the SCAN_TDI pin to the SCAN_TDO pin through the Boundary-Scan Register during scanning. The Boundary-Scan Register can be operated by the Extest and Sample/Preload instructions.

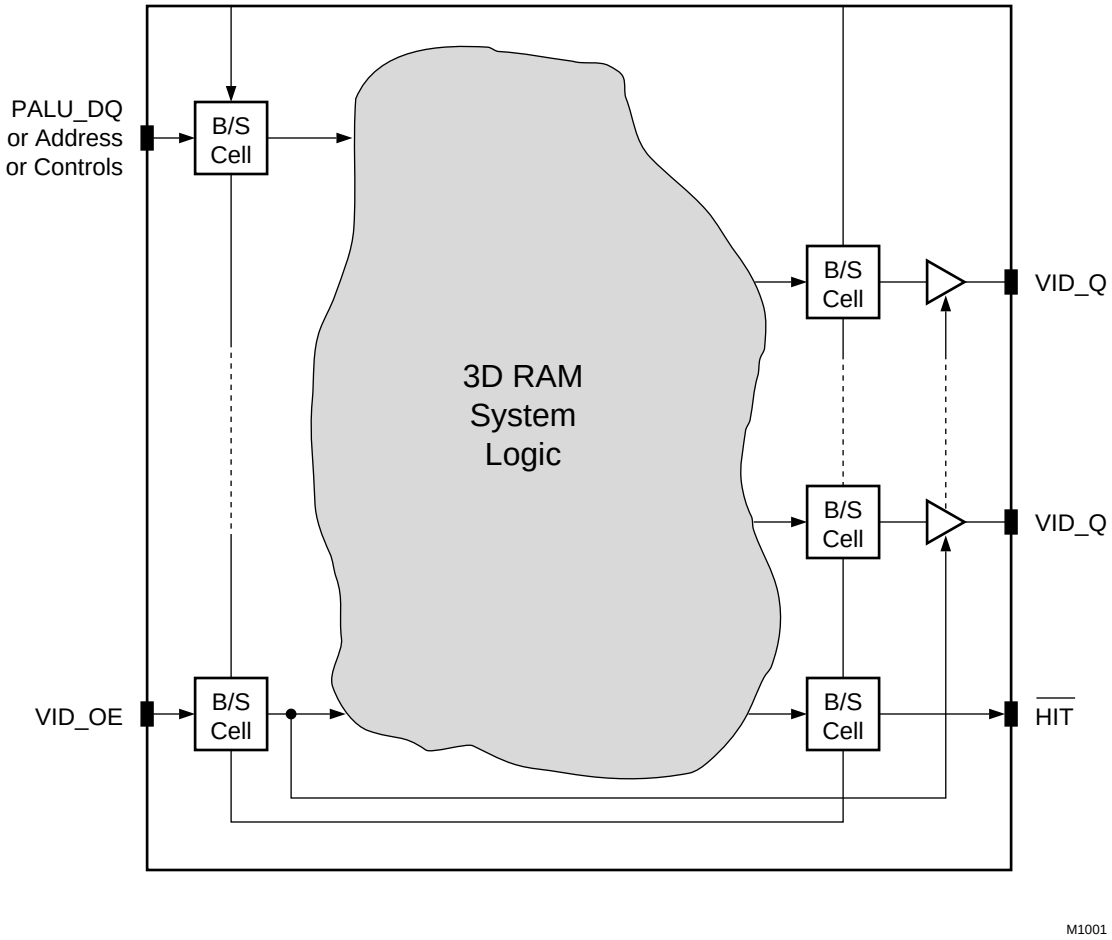


Figure 10.4 Logical structure of the Boundary-Scan Register

The boundary-scan cells inside the Boundary-Scan Register are organized in the following order:

SCAN_TDI → $DRAM\_A_0$ → $DRAM\_A_1$ → $DRAM\_A_2$ →
$DRAM\_A_3$ → $DRAM\_A_4$ → $DRAM\_A_5$ → DRAM_EN →
$DRAM\_OP_0$ → $PALU\_A_0$ → $PALU\_A_1$ → $PALU\_A_2$ →
$PALU\_EN_0$ → $PALU\_OP_0$ → $PALU\_OP_1$ → $PALU\_BE_0$ →
$PALU\_BE_1$ → $PALU\_DX_0$ → $PALU\_DX_1$ → $PALU\_DQ_0$ →
$PALU\_DQ_1$ → $PALU\_DQ_2$ → $PALU\_DQ_3$ → $PALU\_DQ_4$ →
$PALU\_DQ_5$ → $PALU\_DQ_6$ → $PALU\_DQ_7$ → $PALU\_DQ_8$ →
$PALU\_DQ_9$ → $PALU\_DQ_{10}$ → $PALU\_DQ_{11}$ → $PALU\_DQ_{12}$ →
$PALU\_DQ_{13}$ → $PALU\_DQ_{14}$ → $PALU\_DQ_{15}$ → $PALU\_DQ_{16}$ →
$PALU\_DQ_{17}$ → $PALU\_DQ_{18}$ → $PALU\_DQ_{19}$ → $PALU\_DQ_{20}$ →
$PALU\_DQ_{21}$ → $PALU\_DQ_{22}$ → $PALU\_DQ_{23}$ → $PALU\_DQ_{24}$ →
$PALU\_DQ_{25}$ → $PALU\_DQ_{26}$ → $PALU\_DQ_{27}$ → $PALU\_DQ_{28}$ →
$PALU\_DQ_{29}$ → $PALU\_DQ_{30}$ → $PALU\_DQ_{31}$ → $PALU\_DX_2$ →
$PALU\_DX_3$ → $PALU\_BE_2$ → $PALU\_BE_3$ → $PALU\_OP_2$ →
PALU_WE → $PALU\_EN_1$ → $PALU\_A_3$ → $PALU\_A_4$ →
$PALU\_A_5$ → $DRAM\_OP_1$ → $DRAM\_OP_2$ → $DRAM\_A_6$ →
$DRAM\_A_7$ → $DRAM\_A_8$ → $DRAM\_BS_0$ → $DRAM\_BS_1$ →
$\overline{RESET}$ → VID_Q_8 → VID_Q_9 → VID_Q_{10} →
VID_Q_{11} → VID_Q_{12} → VID_Q_{13} → VID_Q_{14} →
VID_Q_{15} → VID_QSF → VID_CKE → VID_OE →
$\overline{HIT}$ → VID_Q_0 → VID_Q_1 → VID_Q_2 →
VID_Q_3 → VID_Q_4 → VID_Q_5 → VID_Q_6 →
VID_Q_7 → SCAN_TDO

Instruction Register

The Instruction Register (IR) allows instructions to be serially shifted into the 3D-RAM through the SCAN_TDI pin. The instruction selects the particular test to be performed, the test data register to be accessed, or both. The instruction register is a four-bit wide shift-register with a parallel latch. The most significant bit is connected to the SCAN_TDI pin and the least significant bit is connected to the SCAN_TDO pin. On entering the Capture-IR controller state, the Instruction Register is loaded with the default instruction “1001”, which is the Bypass instruction. Instructions are shifted into the Instruction Register on the rising edge of the SCAN_TCK pin while the TAP controller is in the Shift-IR state.

The 3D-RAM supports all three mandatory boundary-scan instructions, namely Bypass, Sample/Preload, and Extest. Table 10.1 lists the 3D-RAM boundary-scan instruction codes.

Table 10.1 Boundary-Scan instruction codes

Instruction Code	Instruction Name
0000	Extest
0001	Bypass
0010	Bypass
0011	Bypass
0100	Sample/Preload
0101	Bypass
0110	Bypass
0111	Bypass
1000	Bypass
1001	Bypass
1010	Bypass
1011	Bypass
1100	Bypass
1101	Bypass
1110	Bypass
1111	Bypass

Bypass Instruction

The instruction codes for the Bypass instruction are any codes except “0000” (for Extest) and “0100” (for Sample/Preload). The Bypass instruction selects the Bypass Register to be connected to the SCAN_TDI or SCAN_TDO pin. The Bypass Register contains a single shift-register stage and is used to provide a minimum length serial path between the SCAN_TDI and the SCAN_TDO pins when no scan test operation of the 3D-RAM is required. This allows more rapid movement of test data to and from other components on the board.

Due to the pull-up resistor on the SCAN_TDI input, an open circuit fault in the board level test data path will cause the Bypass Register to be selected following an instruction scan cycle. This was done to prevent any unwanted interference with the normal operation of the 3D-RAM.

Sample/Preload Instruction

The instruction code is “0100”. The Sample/Preload instruction allows the scanning of the Boundary-Scan Register without interference to the normal operation of the 3D-RAM. As suggested by the instruction name, the Sample/Preload instruction can be used to perform two functions:

- SAMPLE is performed in the Capture-DR controller state. All signals received at the 3D-RAM input pins are loaded into the Boundary-Scan Register on the rising edge of the SCAN_TCK pin.
- PRELOAD is performed in the Update-DR controller state. The data held in the shift-register stage of the output cell is latched on the falling edge of the SCAN_TCK pin.

Extest Instruction

The instruction code is “0000”. The Extest instruction allows testing of board interconnections. The Extest instruction selects the Boundary-Scan Register to be connected between the SCAN_TDI and SCAN_TDO pins. Two functions are performed when the Extest instruction is selected:

- In the Capture-DR controller state, all signals received at the 3D-RAM input pins are loaded into the Boundary-Scan Register on the rising edge of the SCAN_TCK pin. This is equivalent to the Sample operation in the Sample/Preload instruction.
- In the Update-DR controller state, the data held in the shift-register stage of the output cell is latched and driven to the 3D-RAM output pins, on the falling edge of the SCAN_TCK.

VID_OE Boundary-Scan Cell

The VID_OE pin controls the tri-state buffer of the VID_Q bus. Therefore, its boundary-scan cell configuration is different from a normal input pin. The functions performed on this cell for the Sample/Preload and Extest instructions are summarized below.

- In the Capture-DR controller state, the signal received at the VID_OE pin is loaded into the shift-register on the rising edge of the SCAN_TCK pin.
- In the Update-DR controller state, the data held in the shift-register is latched on the falling edge of the SCAN_TCK pin. If the instruction is Extest, the latched VID_OE data will control the tri-state buffer of the VID_Q bus.

10 JTAG Boundary Scan

11 Formal Specification of Operations

Formal Specification of Operations

This chapter specifies exactly which bits are moved for many types of operations. It uses a syntax derived from C and Verilog to specify exactly which bits are copied for each operation.

Elements

```
bit DA[4][257][10240]   DRAM Array
bit SA[4][10240]        Sense Amplifiers
bit RAL[4][9]           Row Address Latch
bit VB[2][640]          Video Buffer
bit VD[16]              Video Data pins
bit VC[7]               Video Counter
bit VM[1]               Video Mode
bit SRAM[8][256]        Static RAM
bit DT[8][32]           Dirty Tag
bit PM[32]              Plane Mask register
bit DQ[32]              Pixel ALU data pins
bit BE[4]               Byte Enable pins
bit daddr[9]            DRAM address
bit paddr[6]            Pixel ALU address
```

Bit Ordering of Elements

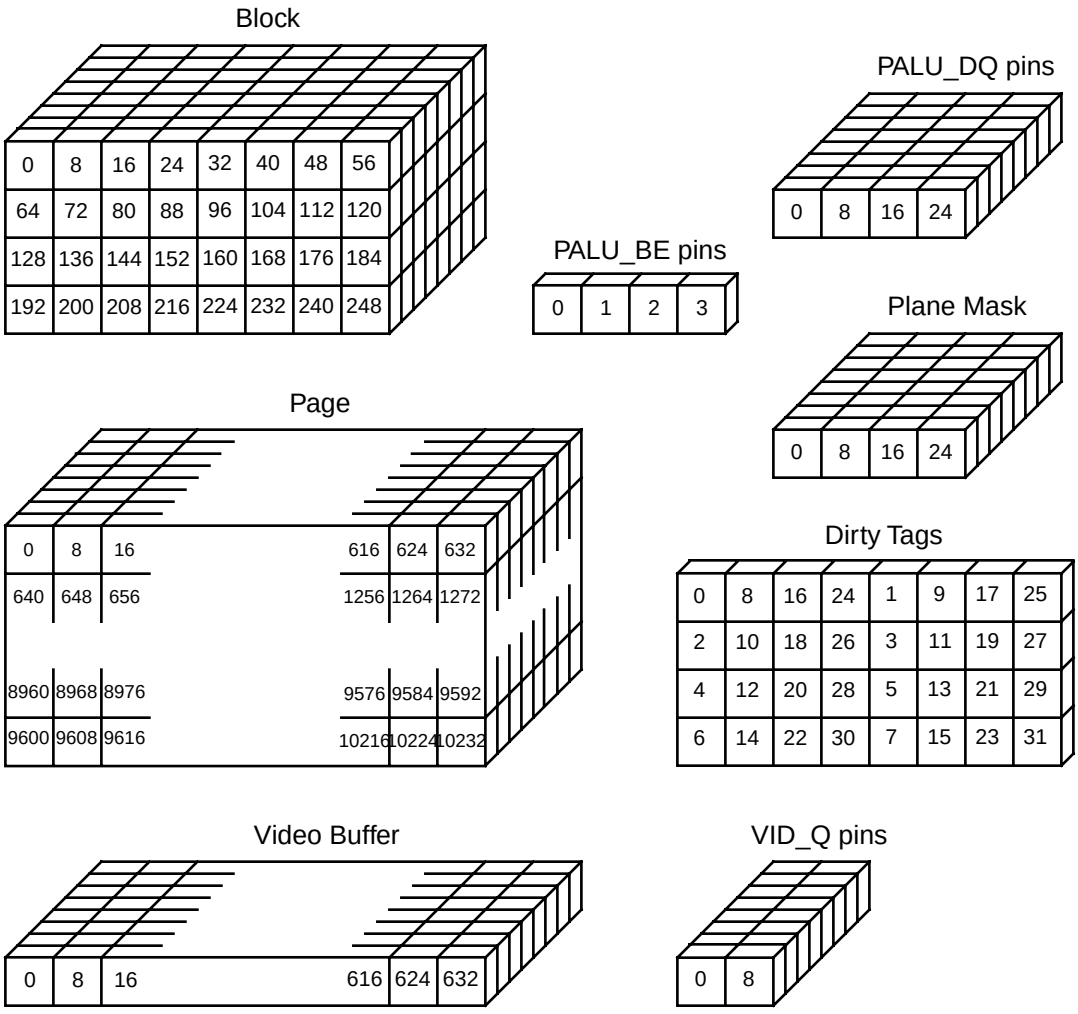


Figure 11-1 Bit orderings of several elements

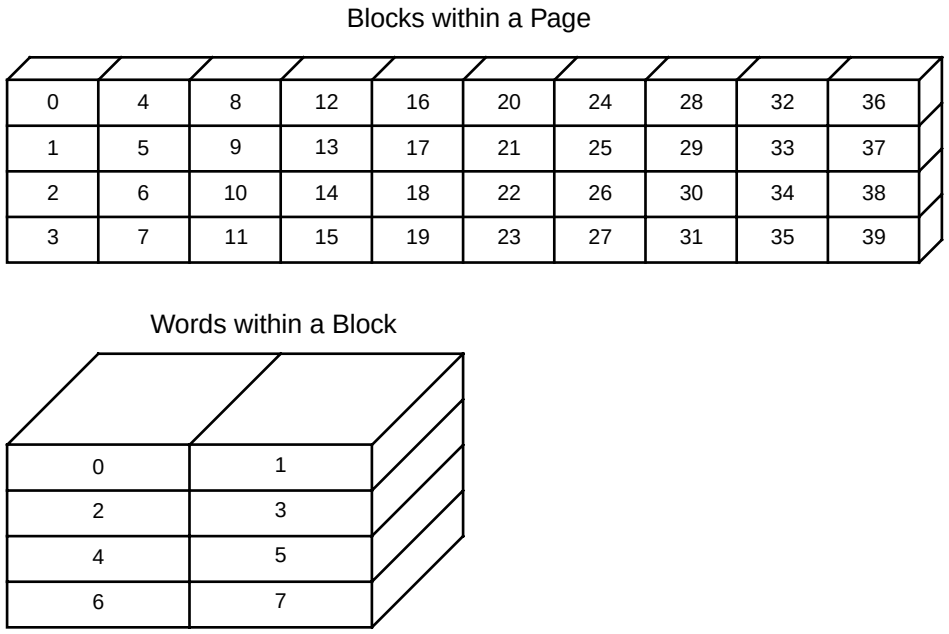


Figure 11.2 Orderings of words and blocks in a page

Access Page

```
ACCESS_PAGE(bit bank[2], bit daddr[9]) {
    bit i[4];
    for(i = 0; i < 9; i++)
        RAL[bank][i] <- daddr[i];
    bit j[14];
    for(j = 0;j < 10240; j++)
        SA[bank][j] <- DA[bank][daddr][j];
}
```

Duplicate Page

```
DUPLICATE_PAGE(bit bank[2], bit daddr[9]) {
    bit i[4];
    for(i = 0; i < 9; i++)
        RAL[bank][i] <- daddr[i];
    bit j[14];
    for(j = 0;j < 10240; j++)
        DA[bank][daddr][j] <- SA[bank][j];
}
```

Precharge Bank

```
PRECHARGE(bit bank[2]) {}
```

Read Block

```
READ_BLOCK(bit bank[2], bit daddr[9]) {
    bit i[8];
    for(i = 0; i < 256; i++)
        SRAM[daddr[8..6]][i] <- SA[bank][daddr[1..0]*2560 +
                i[7..6]*640 + daddr[5..2]*64 + i[5..0]];
    bit j[5];
    for(j = 0;j < 32; j++)
        DT[daddr[8..6]][j] <- 0;
}
```

Masked Write Block

```
MASKED_WRITE_BLOCK(bit bank[2], bit daddr[9]) {
    bit i[8];
    for(i = 0; i < 256; i++)
        if(PM[i[4..0]] && DT[daddr[8..6]][{i[4..3],i[7..5]}]) {
            SA[bank][daddr[1..0]*2560 + i[7..6]*640 +
                    daddr[5..2]*64 + i[5..0]] <-
                    SRAM[daddr[8..6]][i];
            DA[bank][RAL][daddr[1..0]*2560 + i[7..6]*640 +
                    daddr[5..2]*64 + i[5..0]] <-
                    SRAM[daddr[8..6]][i];
        }
}
```

Unmasked Write Block

```
UNMASKED_WRITE_BLOCK(bit bank[2], bit daddr[9]) {
    bit i[8];
    for(i = 0; i < 256; i++)
        if(DT[daddr[8..6]][{i[4..3],i[7..5]}]) {
            SA[bank][daddr[1..0]*2560 + i[7..6]*640 +
                    daddr[5..2]*64 + i[5..0]] <-
                    SRAM[daddr[8..6]][i];
            DA[bank][RAL][daddr[1..0]*2560 + i[7..6]*640 +
                    daddr[5..2]*64 + i[5..0]] <-
                    SRAM[daddr[8..6]][i];
        }
}
```

Video Transfer

```
VIDEO_TRANSFER(bit bank[2], bit daddr[9]) {
    bit i[10];
    for(i = 0; i < 640; i++)
        VB[bank[0]][i] <- SA[bank][640*daddr[3..0] + i];
    if(daddr[8]) {
        VC[5..0] <- 0;
        VC[6] <- bank[0];
        VM[0] <- daddr[7];
    }
}
```

Video Cycle

```
VIDEO_CYCLE(bit enable[1], bit voe[1]) {
    if(voe) {
        bit i[4];
        for(i = 0; i < 16; i++)
            VD[i] <- VB[VC[6]][{VC[5..1],(VC[0]^VM[0])}*16 + i];
    }
    if(enable) {
        if(VC[5..0] == 39) {
            VC[5..0] <- 0;
            VC[6] <- ~VC[6];
        }
        else {
            VC[5..0] <- VC[5..0] + 1;
        }
    }
}
```

Data Read

```
DATA_READ(bit paddr[6]) {
    bit i[5];
    for(i = 0; i < 32; i++)
        if(BE[i[4..3]])
            DQ[i] <- SRAM[paddr  [5..3]][paddr[2..0]*32 + i];
}
```

Stateless Initial Data Write

```
STATELESS_INITIAL_DATA_WRITE(bit paddr[6]) {
    bit i[5];
    for(i = 0; i < 32; i++)
        if(BE[i[4..3]])
            SRAM[paddr[5..3]][paddr[2..0]*32 + i] <- DQ[i];
    bit j[5];
    for(j = 0;j < 32; j++)
        if (CDS[0] == 0) /* (8,8,8,8) normal mode */
            if((paddr[2..0] == j[2..0]) && BE[j[4..3]])
                DT[paddr[5..3]][j] <- 1;
             else
                DT[paddr[5..3]][j] <- 0;
        else /* (4,4,4,4) 16-bit color mode */
            if(((BE[3] || BE[2]) == 1) && ((BE[1] || BE[0]) == 1))
                ERROR("illegal byte enable combination");
            else
                if((paddr[2..0] == j[2..0]) && j[4])
                    DT[paddr[5..3]][j] <- (BE[3] || BE[1]);
                else if((paddr[2..0] == j[2..0] && !j[4])
                    DT[paddr[5..3]][j] <- (BE[2] || BE[0]);
                else
                    DT[paddr[5..3]][j] <- 0;)
}
```

Stateless Normal Data Write

```
STATELESS_NORMAL_DATA_WRITE(bit paddr[6]) {
    bit i[5];
    for(i = 0; i < 32; i++)
        if(BE[i[4..3]])
            SRAM[paddr[5..3]][paddr[2..0]*32 + i] <- DQ[i];
    bit j[5];
    for(j = 0; j < 32; j++)
        if (CDS[0] == 0) /* (8,8,8,8) normal mode */
            if((paddr[2..0] == j[2..0]) && BE[j[4..3]])
                DT[paddr[5..3]][j] <- 1;
        else /* (4,4,4,4) 16-bit color mode */
            if(((BE[3] || BE[2]) == 1) && ((BE[1] || BE[0]) == 1))
                ERROR("illegal byte enable combination");
            else
                if((paddr[2..0] == j[2..0]) && j[4])
                    DT[paddr[5..3]][j] <- (BE[3] || BE[1]);
                else if((paddr[2..0] == j[2..0] && !j[4])
                    DT[paddr[5..3]][j] <- (BE[2] || BE[0]);
}
```

Replace Dirty Tag

```
REPLACE_DIRTY_TAG(bit paddr[6]) {
    bit i[5];
    for(i = 0; i < 32; i++)
        if(BE[i[4..3]])
            if (i == paddr[2..0] + 8*i[4..3])
                DT[paddr[5..3]][i] <- DQ[i];
}
```

OR Dirty Tag

```
OR_DIRTY_TAG(bit paddr[6]) {
    bit i[5];
    for(i = 0; i < 32; i++)
        if(BE[i[4..3]] && DQ[i])
            if (i == paddr[2..0] + 8*i[4..3])
                DT[paddr[5..3]][i] <- DQ[i];
}
```

Write Plane Mask Register

```
WRITE_PLANE_MASK_REGISTER() {
    bit i[5];
    for(i = 0; i < 32; i++)
        if(BE[i[4..3]])
            PM[i] <- DQ[i];
}
```

12 Appendix A

Appendix A

Glossary

Some of the terms used in this document may be unfamiliar to the reader, and some may have a specific meaning in the context of this document. For convenience, these terms are collected here and provided with a brief definition and a reference to the paragraphs where the terms are explained in greater detail. This glossary list is not intended to be exhaustive.

3D-RAM

An innovative 10-Mbit cached dual-port CMOS memory device that dramatically improves the performance of a three-dimensional computer graphics system with on-chip support for Z-buffer hidden surface removal algorithm and for full blending and logical raster operations, achieving a peak bandwidth of 14.6 Gbytes/s and a sustained bandwidth of 400 Mbytes/s (for the -10 speed grade).

Blending

A computer graphics operation for simulating the visual effect of overlapping objects with the foreground objects being partially transparent. An example of blending equations is that overall color = (a) x (color of foreground object) + (1 - a) x (color of background object), where a is the percentage of light transmitted through the medium of the foreground object. Each of the four 8-bit Blend units in the Pixel ALU can perform one of the two multiplications and then the addition, provided that the product of the other multiplication is supplied by the rendering controller. (Pages 10 and 26)

Block

A unit of memory organized into eight 32-bit words. This is the unit of data movement between a DRAM bank and the Pixel Buffer. (Pages 4 and 21)

Byte

A unit of memory containing 8 bits of data. This is the unit of data operation for the four Blend units in the Pixel ALU. The rendering controller can enable or disable the writing (to 3D-RAM) or reading (from 3D-RAM) of the individual bytes in a word. (Pages 10, 15, and 26)

Color Buffer

The collection of memory that contains all color bits of all pixels to be displayed on the screen. Because the alpha information is needed for blending operations in 3D-RAM, the alpha data should also be stored together with the color data. It is also popular to have overlay information stored together with the color information to allow fast display of 2D objects by data multiplexing in a RADMAC chip. (Chapter 6)

Dirty Tag

A 32-bit memory in the Pixel Buffer, indicating which of the 32 bytes in the corresponding 256-bit block in the SRAM cache have been updated by the Pixel ALU since the data was transferred from the DRAM array. A “1” in a bit of Dirty Tag indicates the corresponding byte in the SRAM cache is newer than the data in the DRAM array. There are eight such Dirty Tags in the Pixel Buffer. (Page 22)

In 2D rendering, this feature may also be used for color expansion from a bit to a byte to accelerate drawing of many pixels of the same color. Two Pixel ALU operations, namely “Replace Dirty Tags” and “Or Dirty Tags,” may be used to facilitate this application of the Dirty Tags. (Page 23)

In the (4, 4, 4, 4) 16-bit color mode, the setting of the dirty tag bits is same as the (8, 8, 8, 8) 32-bit color mode in the sense that if a byte of data is updated, then the corresponding dirty tag bit is set. However, since the $PALU\_BE_{[3:0]}$ pins have different meanings in the (4, 4, 4, 4) 16-bit color mode from the (8, 8, 8, 8) 32-bit color mode, the specific dirty tag bits that are set are different in the two color modes. The mapping for the “Replace Dirty Tag” and “OR Dirty Tag” are the same in both modes. (Page 42)

Double Buffer

Two color buffers of identical size, with one of them shifting out video data toward the display screen (usually through a RAMDAC chip), while the other being updated with new pixel data by the rendering controller. (Pages 109 and 111)

DRAM Bank

One of the four 2.5-Mbit DRAM banks in a 3D-RAM chip. Each banks has 10,240 sense amplifiers, which function as a level-two pixel cache, and 257 pages of 10,240 bits each. All four DRAM banks are connected with a common 256-bit Global Bus to interface with the Pixel Buffer. (Page 6)

Frame Buffer

A collection of memory that contain all bits of data for all pixels in the display screen and, in some systems where the frame buffer is large enough to hold more than the pixel data for the specific display resolutions, extra pixel data temporarily stored outside the memory area for display pixels. A pixel data may include bits for the color value for each of the R, G, and B color component, bits for the Z (or depth) value, bits for window ID, and bits for some other auxiliary functions, such as overlay; only the color bits are actually displayed on the screen, while the other bits are stored with the color bits for faster graphic processing. (Chapter 6)

Global Bus

A 50-MHz (for the -10 speed grade) 256-bit data bus connecting between the four DRAM banks and the Pixel Buffer. (Page 8)

Magitude Compare

Pixel ALU operation that compares the incoming 32-bit data with the 32-bit data stored in the frame buffer. Each bit of the 32-bit magnitude comparison may be masked by setting a “1” in the corresponding bit of the Magnitude Mask register. The result of one of eight possible magnitude comparison tests can govern whether the new data is written into the frame buffer. Most commonly, as part of the Z-buffer hidden surface removal algorithm, the magnitude comparison tests are performed on the Z value of the existing pixel against that of a new pixel intended for the same screen location. (Pages 10, 46 and 55)

Match Compare

A Pixel ALU operation that compares the incoming 32-bit data with the 32-bit data stored in the frame buffer. Each bit of the 32-bit match comparison may be masked by setting a “1” in the corresponding bit of the Match Mask register. The result of one of four possible match comparison tests can govern whether the new data is written into the frame buffer. (Pages 10, 46 and 55)

Page

A unit of memory contains forty 256-bit blocks. Three DRAM operations (PRE, ACP, and DUP) operate on the entire 10,240-bit page in one command. (Pages 4 and 71)

Picking

A computer graphics operation to select a drawn object on the display screen. A familiar example in 2D graphics is selecting an icon by clicking a mouse button. In 3D graphics, the selection is complicated by the Z values of objects and the size and locations (X, Y, Z) of the 3D selection cursor, and is supported by the on-chip Picking Logic. (Pages 12 and 50)

Picking Logic

A Pixel ALU function that provides to the rendering controller a HIT flag, indicating if a pixel falls within the 3D selection cursor. (Pages 12 and 50)

Pixel ALU

An on-chip 100-MHz (for the -10 speed grade) processing unit with a 7-stage pipeline that performs destination blending/logical raster operation, magnitude comparison, and match comparison all in parallel, thus converting the read-modify-write interface with the rendering controller to a write-mostly one. (Pages 9, 25 and 51)

Pixel Buffer

The triple-port 2,048-bit SRAM as a level-one pixel cache in 3D-RAM with two 100-MHz (for the -10 speed grade) 32-bit buses interfacing with the Pixel ALU and a 50-MHz (for the -10 speed grade) 256-bit bus interfacing with the DRAM arrays. Also included in the Pixel Buffer are eight 32-bit Dirty Tags and a 32-bit Plane Mask. (Pages 7 and 21)

Plane Mask

A 32-bit register that affects both the Stateful Data Writes to the Pixel Buffer and the Masked Block Writes (MBW) to the DRAM arrays. The effect is simultaneous on both types of operations when they are perfomed concurrently by the Pixel ALU port and the DRAM port, respectively. With respect to the Pixel Buffer, the Plane Mask facilitates the “write-per-bit” function on the 32-bit resultant of the ROP/Blend units. For MBW, the effect of the Plane Mask on the (32-bit) word 0 is simply extended to words 1 through 7. (Pages 24 and 53)

ROP

An abbreviation for raster operation. There is a total of sixteen standard logical raster operations, including AND, OR, and XOR. (Pages 10, 26 and 54)

Stateful Data Write

A pixel data operation of the Pixel ALU. The word “stateful” refers to the controls of the Dual Compare units and the ROP/Blend units of the Pixel ALU on whether and what data is to be written into the Pixel Buffer. There are two types of Stateful Data Write: namely, Stateful Initial Data Write and Stateful Normal Data Write. (Page 66)

Stateless Data Write

A pixel data operation of the Pixel ALU. The word “stateless” refers to the straight pass-through of pixel data from the 3D-RAM inputs to the Pixel Buffer, with the data write unaffected by the ROP/Blend units and the Dual Compare units of the Pixel ALU. There are two types of Stateless Data Write: namely, Stateless Initial Data Write and Stateless Normal Data Write. (Page 66)

Stencil or Stenciling

Stenciling applies a test that compares a reference value with the value stored at a pixel in the stencil buffer and then performs two tasks based on the results of this stencil test and the depth test: (1) operates on the update of the stencil data and (2) controls the write enable of the color buffer and the depth buffer. The most common use of stencil function is to generate an irregular shaped region of some desirable color pattern, such as a decal. Stencil may also be applied to produce stipples or screened door

patterns which are sometimes employed to achieve transparency effect without resorting to the expensive multipliers and adders in the blending function. Stencil is also useful in hidden surface removal. (Pages 35 and 61)

Video Buffer

One of the two serial access video buffers of 40x16 bits, which alternates every forty VCLK cycles to shift video data out. One video buffer is connected with two DRAM banks, and all 640 bits of a video buffer are loaded by a single DRAM command (VDX). The video data output rate is 16 bits per 14-ns cycle. (Pages 7 and 74)

Word

A unit of memory representing four bytes. This is the unit of data movement between the Pixel ALU and the Pixel Buffer. (Pages 4 and 21)

Z Buffer

The collection of memory that contains all Z values of all pixels to be displayed on the screen. (Pages 103 and 105)