ZiiLabs Inc., Ltd v. Samsung Electronics Co. Ltd. et al Doc. 273

**IN THE UNITED STATES DISTRICT COURT**
**FOR THE EASTERN DISTRICT OF TEXAS**
**MARSHALL DIVISION**

| | | |
|---|---|---|
| ZIILABS INC., LTD.,<br><br>*Plaintiff,*<br><br>v.<br><br>SAMSUNG ELECTRONICS CO. LTD.; SAMSUNG ELECTRONICS AMERICA, INC.; SAMSUNG TELECOMMUNICATIONS, AMERICA, LLC; SAMSUNG AUSTIN SEMICONDUCTOR, LLC; and APPLE INC.,<br><br>*Defendants.* | §<br>§<br>§<br>§<br>§<br>§<br>§<br>§<br>§<br>§<br>§<br>§<br>§<br>§<br>§ | Case No. 2:14-cv-00203-JRG-RSP |

**MEMORANDUM OPINION AND ORDER**

On March 24, 2015, the Court held a hearing to determine the proper construction of the disputed claim terms in United States Patent Nos. 5,831,637 ("the '637 Patent"); 5,835,096 ("the '096 Patent"); 6,111,584 ("the '584 Patent"); 6,683,615 ("the '615 Patent"); 6,977,649 ("the '649 Patent"); 7,050,061 ("the '061 Patent"); 7,187,383 ("the '383 Patent"); 7,710,425 ("the '425 Patent"); and 8,144,156 ("the '156 Patent") (collectively, the "Asserted Patents"). After considering the arguments made by the parties at the hearing and in the parties' claim construction briefing (Dkt. Nos. 144, 151, 153, 155, and 158), the Court issues this Claim Construction Memorandum and Order.[1]

[1] On March 4, 2015, the Court ordered supplemental claim construction briefing based on Defendants' Joint Notice of Additional Claim Terms Requiring Construction. (Dkt. No. 150.) During the March 24, 2015 hearing, the Court entertained arguments on the originally briefed terms and the nine supplemental terms as presented by the parties.

Dockets.Justia.com

TABLE OF CONTENTS

## I. BACKGROUND

### A. The ‘615, ‘061, and ‘425 Patents

The ‘615, ‘061, and ‘425 Patents generally share a common specification. The patents relate to computer graphics rendering systems and methods, with a particular focus on the “handling of texture data used by rendering accelerators for 3D graphics.” *See, e.g.*, ‘615 Patent at 1:10–13. In the Background Section of the ‘615 Patent, the specification states that “[f]or some years the most critical area of graphics development has been in three-dimensional (‘3D’) graphics.” *Id.* at 1:20–21. The specification adds that the “peculiar demands of 3D graphics are driven by the need to present a realistic view, on a computer monitor, of a three-dimensional scene.” *Id.* at 1:21–23. The specification states that “[t]his requires extensive computation to obtain the correct image for display, taking account of surface textures, lighting, shadowing, and other characteristics.” *Id.* at 1:29–31.

The specification then states that the “visual appeal of computer graphics rendering is greatly enhanced by the use of ‘textures.’” *Id.* at 2:24–25. The specification defines a “texture” as “a two-dimensional image which is mapped into the data to be rendered.” *Id.* at 2:25–27. The specification further states that “[t]extures provide a very efficient way to generate the level of minor surface detail which makes synthetic images realistic, without requiring transfer of immense amounts of data.” *Id.* at 2:28–30.

The specification then describes virtual memory management and virtual texture memory. Regarding virtual memory management, the specification states it “is a technique which allows application software to use a very large range of memory addresses, without knowing how much physical memory is actually present on the computer, nor how the virtual addresses correspond to the physical addresses which are actually used to address the physical

memory chips (or other memory devices) over a bus." *Id.* at 3:8–15. Regarding virtual texture memory, the specification states that "like virtualization of host memory, [virtualization of texture memory] gives the user the impression of a memory space which is larger than can be physically accommodated in real memory." *Id.* at 3:60–63. The specification adds that "[t]his is achieved by partitioning the memory space into a small physical working set and a large virtual set with dynamic swapping between the two." *Id.* at 3:63–64. The specification then describes "doubly-virtualized texture memory" and states that "[t]he present application claims the use of a two- or three-level memory hierarchy for texture memory." *Id.* at 4:64–65. With this general background, the Court now turns to an exemplary claim from each of the patents.

The '615 Patent is titled Doubly-virtualized Texture Memory. It was filed on June 9, 2000, and issued on January 27, 2004. The Abstract of the '615 Patent states the following:

> A graphics system in which the dedicated graphics memory is doubly virtualized: it can be paged into host physical memory, and also, beyond that, into host bulk storage. Portions of host physical memory which are needed to support the graphics memory management process can be locked down.

'615 Patent at Abstract. Claim 3 of the '615 Patent is representative of the asserted claims and recites the following elements (disputed terms in italics):

> 3. A computer system, comprising:
> at least one main memory;
> at least one bulk storage unit;
> specialzed graphics-processing logic, which performs rendering functions on graphics data;
> *dedicated graphics memory* which is used by said graphics-processing logic as primary memory;
> first *memory management logic* which *virtualizes* said graphics memory into a first portion of said main memory; and
> second *memory management logic* which *virtualizes* said main memory, including said first portion thereof, into said bulk storage unit.

The '425 Patent is titled Graphic Memory Management with Invisible Hardware-managed Page Faulting. It was filed on June 9, 2000, and issued on May 4, 2010. The

specification of the ‘425 Patent further describes “a computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor.” ‘425 Patent at 5:60–6:33. The specification states that “[w]hen a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host’s physical memory), it will be fetched in automatically by the graphics memory manager, and the host is not aware anything has happened.” *Id.* at 6:4–8. The specification notes that “if the faulting logical page identifies a page in the third level memory the host” determines where the page is located in host physical memory. *Id.* at 6:10–30. The Abstract of the ‘425 Patent states the following:

> A computer system in which a graphics accelerator unit manages page faulting of texture data invisibly to the host processor.

‘425 Patent at Abstract. Claim 6 of the ‘425 Patent is representative of the asserted claims and recites the following elements (disputed terms in italics):

> 6. A computer system, comprising:
> at least one CPU, operatively connected to have read/write access to a main memory;
> first *memory management logic*, which *virtualizes* said main memory with reference to at least one bulk storage unit; and
> a graphics accelerator unit, comprising rendering accelerator logic, *dedicated graphics memory*, and a second *memory management unit* which manages texture data for said accelerator logic and *performs page faulting of said texture data, invisibly to said CPU.*

The ‘061 Patent is titled Autonomous Address Translation in Graphic Subsystem. It was filed on June 9, 2000, and issued on May 23, 2006. The specification of the ‘061 Patent further states that “[s]ince rendering is a computationally intensive operation, numerous designs have offloaded it from the main CPU. An example of this is the GLINT chip described below.” ‘061 Patent at 2:18–20. The specification states that “FIG. 3 shows a block diagram of a graphics processor which can incorporate the disclosed embodiments of the read-modify-write solutions

in its rendering subsystem." *Id.* at 10:39–41.

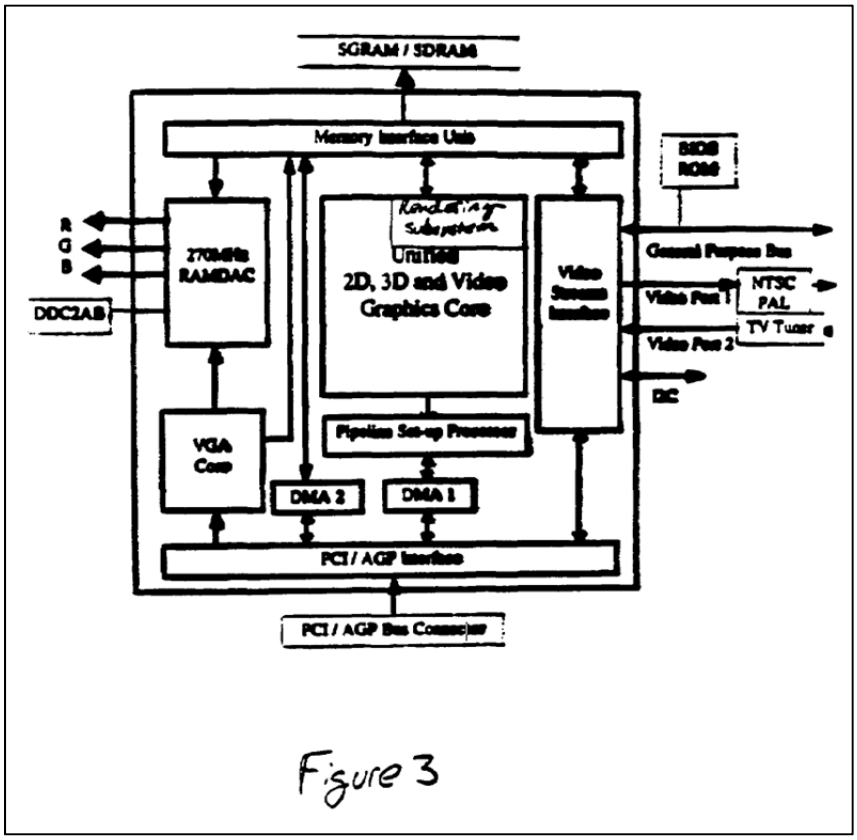


Figure 3

'061 Patent, Figure 3.The Abstract of the '061 Patent states the following:

> A texture caching controller, located on the graphics card, handles address logical-to-physical translation for texture addresses which are not downloaded to level-1 memory due to low use or dynamically changing values. This offloads texture memory management duties from the host.

'061 Patent at Abstract. Claim 1 of the '061 Patent is representative of the asserted claims and recites the following elements (disputed terms in italics):

> 1. A *graphics processing chip*, comprising:
> a graphics accelerator comprising rendering acceleration logic; and
> a *texture memory management function*, integrated on said chip, which manages both texture storage in host memory and also texture storage in *normal texture memory*.

**B. The '156 Patent**

The '156 Patent is titled Sequencer with Async SIMD Array. It was filed on September

28, 2004, and issued on March 27, 2012. The '156 Patent generally relates to computer graphics rendering architecture. *See* '156 Patent at Abstract.[2] The specification states that "[a] conventional SIMD processor or microcode CPU is designed with the sequencer and arithmetic logic unit (ALU) running in lock step." '156 Patent at 3:15–17. The specification further states that "[t]he sequencer is responsible for calculating the address of the next instruction and fetching it." *Id.* at 3:17–18.

In contrast to the "conventional SIMD processor," the specification states "[t]he sequencer and PEs are not designed to run in lock step: instead the sequencer and PEs are decoupled to allow the PEs, which form the SIMD array, to run at 100% efficiency even when the sequencer is switching between threads and performing other flow control operations." *Id.* at 3:60–65. The specification further states that "[t]he present application describes a 3D graphics architecture in which a FIFO buffer is placed between the sequencer and the processing elements (PEs)." *Id.* at 3:58–60. Figure 1 illustrates sequencer FIFO buffer 120 placed between sequencer 130 and processing elements 110 (labeled "PE0, PE1, . . . PE15"). *Id.* at 22:11–23.

[2] The Abstract of the '156 Patent recites the following:

> A 3D graphics architecture in which a buffer is placed between the sequencer and the processing element (PE) array. The sequencer and PE array are not designed to run in lock step: instead the sequencer and PE array are decoupled to allow the PEs to run at 100% efficiency even when the sequencer is switching between threads and performing other flow control operations. Thus, the rate of instruction processing in the PE array is not coupled to the rate of instruction processing in the sequencer.

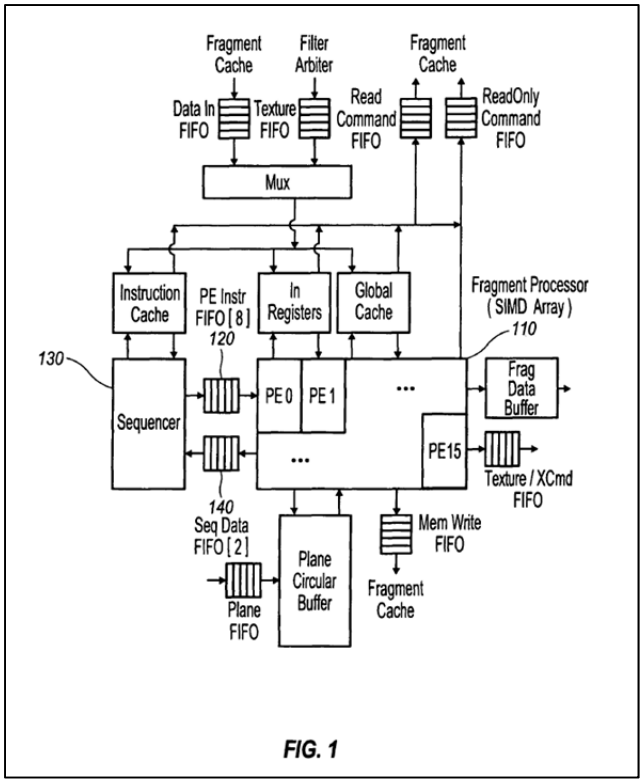


FIG. 1

'156 Patent, Figure 1. The specification states that "the rate of instruction processing in the PE is not coupled to the rate of instruction processing in the sequencer. As a result, the PE and sequencer can be asynchronous." *Id.* at 3:65–4:1. In other words, "FIFO 120 is used to hold the PE instructions generated by sequencer 130, thereby allowing sequencer 130 to run ahead of fragment processor 110 at least until FIFO 120 is full." *Id.* at 22:14–18. The specification also notes that "[t]he PE and sequencer are logically asynchronous, i.e. decoupled, in that they do not have to work on the same instruction at the same time, but as an implementation convenience, they run in the same clock domain so technically they are physically synchronous." *Id.* at 4:1–4.

Claim 6 of the '156 Patent is representative of the asserted claims and recites the following elements (disputed terms in italics):

> 6. A parallel rendering architecture in 3D graphics, comprising:
> at least one *sequencer* which decodes 3D rendering commands for a first plurality of data items and generates fragment-processing commands therefrom;

a plurality of SIMD processing elements, each *operating asynchronously to* and configured to receive processing instructions from said *sequencer* and also separately connected to send and receive pixel data, the processing elements *automatically transferring processing* to another plurality of data items when processing on said first plurality of data items stalls; and

at least one buffer operatively interposed between said *sequencer* and said plurality of processing elements, into which the *sequencer* deposits said fragment-processing commands and from which said processing elements receive said fragment-processing commands, that *decouples operations* between the *sequencer* and the processing elements,

wherein said *sequencer* is *logically asynchronous to* said processing elements and does not have to work on the same instructions at the same time, and

wherein said *sequencer* and said processing elements are physically synchronous, and said *sequencer* and said processing elements operate at different rates,

whereby the likelihood of stalling in said *sequencer* is reduced

### C. The '383 Patent

The '383 Patent is titled Yield Enhancement of Complex Chips. It was filed on March 1, 2002, and issued on March 6, 2007. The '383 Patent generally relates to complex integrated circuits, particularly parallellized graphics accelerators. *See* '383 Patent at Abstract.[3] The specification states that "[t]he only practical way to design high performance graphics chips involves replicating part of the design (i.e. multiple texture pipes) so that multiple operations can be carried out in parallel." '383 Patent at 2:14–17. The specification further states that "[a]s the size of the silicon die is large the expected yield from manufacturing is less than desirable." *Id.* at 2:17–20. The specification continues that "by internally reconfiguring the chip we can still make

[3] The Abstract of the '383 Patent recites the following:

> A graphics processing chip which includes parallel texturing pipelines, with task allocation units which can bypass inoperative ones of said pipelines. Chips which have some but not all pipelines operative can still have full functionality, although performance is reduced.

use of a die with one or more failing texture pipes, for example." *Id.* at 2:27–29. According to the specification, "[t]his allows us to take die which would otherwise be classified as scrap and use them in a lower performance product." *Id.* at 2:29–31.

The specification states that "FIG. 1 shows an example of the physical layout of a graphics processing integrated circuit, wherein a subset of working parts can define a fully operable lower performance secondary part." *Id.* at 3:31–34.

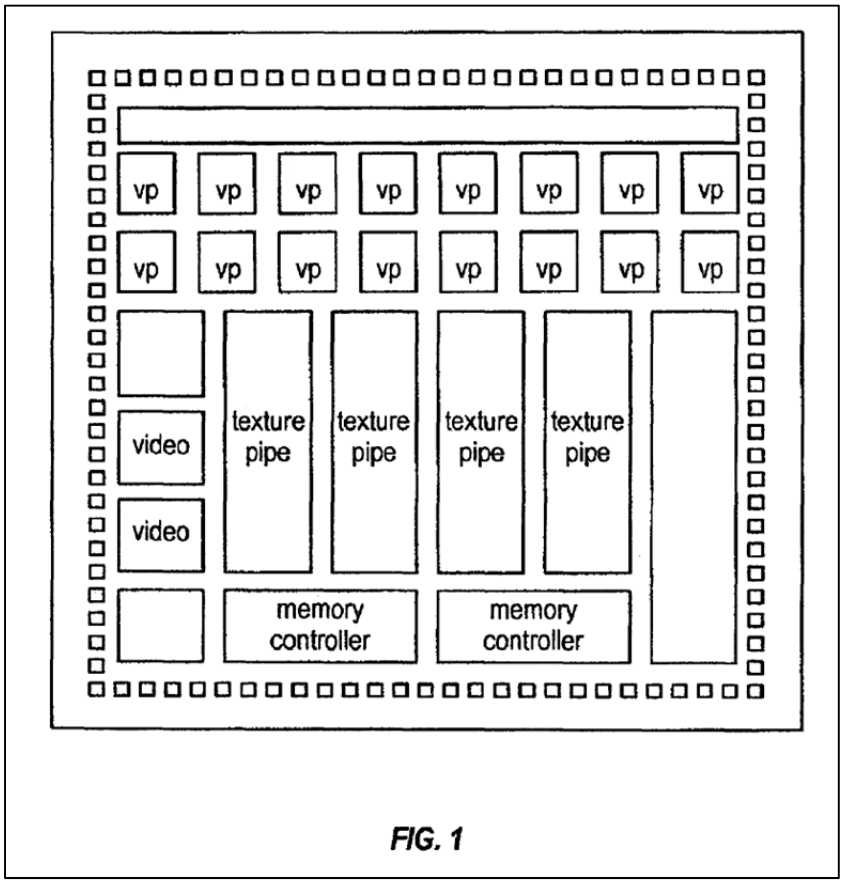


FIG. 1

'383 Patent, Figure 1. The specification states that "[i]n this example, if all components are functional, the chip will operate with: 16 vertex processors, 4 texture pipes, 2 memory controllers, and 2 RAMDACs. (The operation of these components is described below.)" *Id.* at 3:34–37. The specification further states that "if some of these components are defective, the same chip can be specified under a secondary part specification, with e.g. as little as 4 vertex

processors, 1 texture pipe, 1 memory controller, and 1 RAMDAC." *Id.* at 3:38–41. The specification states that "[t]he results of testing at manufacture can be recorded in any of a variety of known ways, e.g. by blowing fuses or by selective laser heating to cause diode punchthrough." *Id.* at 3:42–44.

Claim 1 of the '383 Patent is representative of the asserted claims and recites the following elements (disputed terms in italics):

> 1. A *graphics processor*, comprising:
> a plurality of parallellized *graphics computational units*; and
> one or more *task allocation units programmed to bypass defective ones* of said units within said groups, and to distribute incoming tasks only among operative ones of said units.

**D. The '637 Patent**

The '637 Patent is titled Video Stream Data Mixing for 3D Graphics Systems. It was filed on May 1, 1995, and issued on November 3, 1998. The '637 Patent generally relates to video controllers, and more particularly to graphics processors. *See* '637 Patent at Abstract.[4] The specification states that "[t]he system has a graphics engine, for processing graphics request code and data, in communication with a host computer over a data bus and also in communication

[4] The Abstract of the '637 Patent recites the following:

> A 3D graphics processing system in a preferred embodiment has an input for a digital video data stream. The system has a graphics engine, for processing graphics request code and data, in communication with a host computer over a data bus and also in communication with a frame buffer. It also has an input for a digital video data stream, and the input is in communication with the graphics engine. A control arrangement interrupts processing by the graphics engine of conventional graphics request code and data to permit priority processing of the digital video data stream. In this manner, an image associated with the digital video data stream may be displayed in real time in a desired plane that may be accessed and processed by the graphics processing system as a graphic image. In a further embodiment, the control arrangement is operative to interrupt processing by the graphics engine at a span break and the image associated with the digital video data stream and processed by the graphics engine is stored in the frame buffer.

with a frame buffer." '637 Patent at 1:19–22. The specification adds that the system "also has an input for a digital video data stream, and the input is in communication with the graphics engine." *Id.* at 1:22–24. The specification further states that "[a] control arrangement interrupts processing by the graphics engine of conventional graphics request code and data to permit priority processing of the digital video data stream." *Id.* at 1:24–27. The specification states that "[i]n this manner, an image associated with the digital video data stream may be displayed in real time in a desired plane that may be accessed and processed by the graphics processing system as a graphic image." *Id.* at 1:27–31.

The specification describes a further embodiment where "the control arrangement is operative to interrupt processing by the graphics engine at a span break and the image associated with the digital video data stream and processed by the graphics engine is stored in the frame buffer." *Id.* at 1:31–36. The specification states that "[a]s part of this embodiment, a video buffer is disposed in association with the input for feeding the digital video data stream through the input, and the control arrangement includes a means for preventing the interruption of processing by the graphics engine of conventional graphics request code unless it is determined that the video buffer is filled to a desired extent." *Id.* at 1:36–42.

Claim 1 of the '637 Patent is representative of the asserted claims and recites the following elements (disputed terms in italics):

> 1. A 3D graphics processing system capable of processing a digital video data stream and *3D graphics request code*, the system comprising:
> a graphics engine for processing the *3D graphics request code* and providing output data to a frame buffer, the graphics engine having a request code input for receiving the *3D graphics request code*;
> a video input for receiving the digital video data stream, the video input being in communication with the graphics engine; and

a *graphics engine controller* for i*nterrupting processing by the graphics engine of the 3D graphics request code* when the digital video data stream is received by the video input, the *graphics engine controller* permitting priority processing of the digital video data stream *when the processing of 3D graphics request code is interrupted*,

*when interrupted*, the graphics engine responsively producing an image associated with the digital video data stream, the image being displayed in real time in a preselected plane and being accessed and processed by the graphics processing system as a graphic image.

**E. The ‘096 Patent**

The ‘096 Patent is titled Rendering System Using 3D Texture-processing Hardware for Accelerated 2D Rendering. It was filed on May 1, 1996, and issued on November 10, 1998. The ‘096 Patent generally relates to computer graphics and animation systems, and particularly to 3D graphics rendering hardware. *See* ‘096 Patent at Abstract.[5] The specification states that “there is a risk that the use of three-dimensional graphics, while highly attractive, may jeopardize an even more basic element of user satisfaction, if they slow down 2D rendering tasks.” ‘096 Patent at 5:40–44. The specification continues that “[s]ince the 2D rendering tasks may be generated by the operating system (or other user-interfacing software), it is highly desirable not to interfere with rapid performance of 2D rendering, even if substantial 3D rendering tasks have been loaded into the pipeline.” *Id.* at 5:44–48.

To address this issue, the specification states that “[t]he present application discloses a

[5] The Abstract of the ‘096 Patent recites the following:

> A 3D rendering accelerator, in which the hardware texturing capability is also used to provide enhanced 2D rendering. Texturing units, when operating in a 2D mode, are available for use for storing icons and characters locally to avoid the expense of doing a lookup from the host system. Texturing units are also used as storage for pattern data for performing a tiled fill of a graphical object and for defining arbitrarily large stipple patterns. Color index dither patterns may also be stored in the texture units to avoid the necessity of doing a texture download from the host system.

high-performance 3D rendering accelerator, in which the hardware texturing capability is also used to provide enhanced 2D rendering." *Id.* at 5:50–52. The specification further states that "[t]exturing units, when operating in a 2D mode, are available for use for storing icons and characters locally to avoid the expense of doing a lookup from the host system." *Id.* at 5:55–59. The specification adds that the "[t]exturing units are also used as storage for pattern data for performing a tiled fill of a graphical object and for defining arbitrarily large stipple patterns." *Id.* at 5:58–61. The specification also states that "[c]olor index dither patterns may also be stored in the texture units to avoid the necessity of doing a multi-color fill from the host to simulate the color index dither process." *Id.* at 5:61–63

Claim 5 of the '096 Patent is representative of the asserted claims and recites the following elements (disputed terms in italics):

> 5. A rendering system for processing 2D and 3D graphics, comprising:
> a data bus;
> a memory connected to said data bus;
> a rasterizer connected to said data bus;
> a localbuffer unit connected to said data bus;
> a texturing unit connected to said data bus; and
> a framebuffer unit connected to said data bus;
> wherein said texturing unit processes conventional texture data when in a 3D graphics mode; and
> wherein said texturing unit stores *character data* when in a *2D mode*.

**F. The '584 Patent**

The '584 Patent is titled Rendering System with Mini-patch Retrieval from Local Texture storage. It was filed on June 4, 1996, and issued on August 29, 2000. The '584 Patent generally relates to computer graphics and animation systems, and particularly to 3D graphics rendering

hardware. *See* '584 Patent at Abstract.[6] The specification states that "it is always preferable to organize accesses, if possible, to minimize the number of accesses and page breaks." '584 Patent at 5:29–31. The specification describes that "[a]s each page of texel data is read in a conventional sequential, or one-dimensional, manner, the number of useful texels read will typically be relatively small, e.g. only one of the read pixels will actually required for the texture, and another page must be read for the next required texel." *Id.* at 5:55–59.

The specification states that the solution to this problem is using a wide word "so that fetches from the texture memory retrieve enough data for several pixels." *Id.* at 5:62–64. The specification states that "[t]he multiple pixels in each word are aligned in a 'mini-patch' which includes both vertical and horizontal neighbors." *Id.* at 5:64–66. The specification further states that "[b]y reading the texture data in a two dimensional patch, the number of useful texels per read is increased." *Id.* at 6:2–4. According to the specification, "[t]his naturally results in more efficient memory access, with fewer accesses required for each texture pattern." *Id.* at 6:4–6.

Claim 1 of the '584 Patent is representative of the asserted claims and recites the following elements (disputed terms in italics):

> 1. A rendering method, comprising the steps of:
> decomposing primitives into fragments to be rendered;
> computing depth and color values for individual ones of said fragments;
> retrieving at least four pixels of a stored texture from a single memory location, using a word width which is more than three times the number of bits per pixel in said stored texture; *said pixels of each said word being grouped in said stored texture in a pattern which is more than one pixel wide and more than one pixel high*;

[6] The Abstract of the '584 Patent recites the following:

> A rendering processor with texture processing capability, in which textures are retrieved from local storage as nxn patches. By retrieving a multi-pixel patch on each memory read, the frequency of memory accesses and page breaks in particular are reduced, resulting in a lower memory access time overhead.

performing a texturing operation upon said color values using said stored texture to generate primitive data;
rendering a primitive using said primitive data; and
displaying said rendered primitive.

## II. APPLICABLE LAW

### *A. Claim Construction*

"It is a 'bedrock principle' of patent law that 'the claims of a patent define the invention to which the patentee is entitled the right to exclude.'" *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc) (quoting *Innova/Pure Water Inc. v. Safari Water Filtration Sys., Inc.*, 381 F.3d 1111, 1115 (Fed. Cir. 2004)). To determine the meaning of the claims, courts start by considering the intrinsic evidence. *See id.* at 1313. *C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 861 (Fed. Cir. 2004); *Bell Atl. Network Servs., Inc. v. Covad Commc'ns Group, Inc.*, 262 F.3d 1258, 1267 (Fed. Cir. 2001). The intrinsic evidence includes the claims themselves, the specification, and the prosecution history. *See Phillips*, 415 F.3d at 1314; *C.R. Bard, Inc.*, 388 F.3d at 861. Courts give claim terms their ordinary and accustomed meaning as understood by one of ordinary skill in the art at the time of the invention in the context of the entire patent. *Phillips*, 415 F.3d at 1312–13; *Alloc, Inc. v. Int'l Trade Comm'n*, 342 F.3d 1361, 1368 (Fed. Cir. 2003).

The claims themselves provide substantial guidance in determining the meaning of particular claim terms. *Phillips*, 415 F.3d at 1314. First, a term's context in the asserted claim can be very instructive. *Id*. Other asserted or unasserted claims can also aid in determining the claim's meaning because claim terms are typically used consistently throughout the patent. *Id*. Differences among the claim terms can also assist in understanding a term's meaning. *Id*. For example, when a dependent claim adds a limitation to an independent claim, it is presumed that the independent claim does not include the limitation. *Id*. at 1314–15.

"[C]laims 'must be read in view of the specification, of which they are a part.'" *Id*. (quoting *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995) (en banc)). "[T]he specification 'is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term.'" *Id*. (quoting *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996)); *Teleflex, Inc. v. Ficosa N. Am. Corp.*, 299 F.3d 1313, 1325 (Fed. Cir. 2002). This is true because a patentee may define his own terms, give a claim term a different meaning than the term would otherwise possess, or disclaim or disavow the claim scope. *Phillips*, 415 F.3d at 1316. In these situations, the inventor's lexicography governs. *Id*. The specification may also resolve ambiguous claim terms "where the ordinary and accustomed meaning of the words used in the claims lack sufficient clarity to permit the scope of the claim to be ascertained from the words alone." *Teleflex, Inc.*, 299 F.3d at 1325. But, "'[a]lthough the specification may aid the court in interpreting the meaning of disputed claim language, particular embodiments and examples appearing in the specification will not generally be read into the claims.'" *Comark Commc'ns, Inc. v. Harris Corp.*, 156 F.3d 1182, 1187 (Fed. Cir. 1998) (quoting *Constant v. Advanced Micro-Devices, Inc.*, 848 F.2d 1560, 1571 (Fed. Cir. 1988)); *see also Phillips*, 415 F.3d at 1323. The prosecution history is another tool to supply the proper context for claim construction because a patent applicant may also define a term in prosecuting the patent. *Home Diagnostics, Inc., v. Lifescan, Inc.*, 381 F.3d 1352, 1356 (Fed. Cir. 2004) ("As in the case of the specification, a patent applicant may define a term in prosecuting a patent.").

Although extrinsic evidence can be useful, it is "'less significant than the intrinsic record in determining the legally operative meaning of claim language.'" *Phillips*, 415 F.3d at 1317 (quoting *C.R. Bard, Inc.*, 388 F.3d at 862). Technical dictionaries and treatises may help a court

understand the underlying technology and the manner in which one skilled in the art might use claim terms, but technical dictionaries and treatises may provide definitions that are too broad or may not be indicative of how the term is used in the patent. *Id*. at 1318. Similarly, expert testimony may aid a court in understanding the underlying technology and determining the particular meaning of a term in the pertinent field, but an expert's conclusory, unsupported assertions as to a term's definition are entirely unhelpful to a court. *Id*. Generally, extrinsic evidence is "less reliable than the patent and its prosecution history in determining how to read claim terms." *Id*.

### *B. Means-plus-function Limitations*

The asserted patents also contain means-plus-function limitations that require construction. Where a claim limitation is expressed in "means plus function" language and does not recite definite structure in support of its function, the limitation is subject to 35 U.S.C. § 112, ¶ 6. *Braun Med., Inc. v. Abbott Labs.*, 124 F.3d 1419, 1424 (Fed. Cir. 1997). In relevant part, 35 U.S.C. § 112, ¶ 6 mandates that "such a claim limitation 'be construed to cover the corresponding structure . . . described in the specification and equivalents thereof.'" *Id.* (citing 35 U.S.C. § 112, ¶ 6). Accordingly, when faced with means-plus-function limitations, courts "must turn to the written description of the patent to find the structure that corresponds to the means recited in the [limitations]." *Id*.

Construing a means-plus-function limitation involves multiple steps. "The first step in construing [a means-plus-function] limitation is a determination of the function of the means-plus-function limitation." *Medtronic, Inc. v. Advanced Cardiovascular Sys., Inc.*, 248 F.3d 1303, 1311 (Fed. Cir. 2001). Once a court has determined the limitation's function, "the next step is to determine the corresponding structure disclosed in the specification and equivalents thereof." *Id.*

A "structure disclosed in the specification is 'corresponding' structure only if the specification or prosecution history clearly links or associates that structure to the function recited in the claim." *Id.* Moreover, the focus of the "corresponding structure" inquiry is not merely whether a structure is capable of performing the recited function, but rather whether the corresponding structure is "clearly linked or associated with the [recited] function." *Id.*

## III. CONSTRUCTION OF AGREED TERMS

The parties agreed to the construction of the following term:

| Claim Term/Phrase | Agreed Construction |
|---|---|
| rendering functions on graphics data<br><br>('615 Patent, claims 1, 3) | functions that produce values for each displayed pixel of a two-dimensional representation of a three-dimensional scene |

(Dkt. No. 122-1, Joint Claim Construction and Prehearing Statement). In view of the parties' agreement on the proper construction of the identified term, the Court hereby adopts the parties' agreed construction.

During the claim construction hearing, the Court provided the parties with preliminary constructions for the disputed terms/phrases. The parties agreed to the Court's preliminary construction for the following terms:

| Claim Term/Phrase | Agreed Construction |
|---|---|
| "memory management logic/unit"<br><br>('615 Patent, claims 3, 5, 6, 8)<br>('425 Patent, claims 6, 7, 10, 11) | "a hardware device or circuit that manages virtual memory and paging" |
| "when interrupted"<br><br>('637 Patent, claim 1) | "when suspended" |
| "interrupting processing by the graphics engine of the 3D graphics request code"<br><br>('637 Patent, claim 1) | "suspending processing of 3D graphics request code" |
| "when the processing of 3D graphics request code is interrupted" | "when the processing of 3D graphics request code is suspended" |

| | |
|---|---|
| (‘637 Patent, claim 1) | |
| “2D mode”<br><br>(‘096 Patent, claim 5) | “mode for processing two-dimensional graphics” |
| “character data”<br><br>(‘096 Patent, claims 2, 5) | “data representing an individual letter, digit, punctuation mark, or symbol” |

Regarding the term **“memory management logic/unit,”** the Court finds that the term appears in claims 3, 5, 6, and 8 of the ‘615 Patent, and claims 6, 7, 10, and 11 of the ‘425 Patent. The Court further finds that the terms “memory management logic” and “memory management unit” are used interchangeably and are synonymous. The Court also finds that the terms are used consistently in the claims and are intended to have the same meaning in each claim.

The Court further finds that the intrinsic evidence indicates that the recited “memory management logic/unit” is “a hardware device or circuit that manages virtual memory and paging.” For example, claim 3 of the ‘615 Patent recites that the first memory management logic “virtualizes said graphics memory into a first portion of said main memory.” Likewise, claim 6 of the ‘425 Patent recites that the second memory management unit “manages texture data for said accelerator logic and performs page faulting of said texture data.” The parties agree that “memory management logic/unit” is a “hardware device or circuit,” and this claim language indicates that this circuit manages virtual memory and paging.

Turning to the constructions proposed by the parties, the Court rejects Plaintiff’s construction because it is limited to certain aspects of virtual memory management. The Court agrees that the evidence indicates that logical or virtual addresses have to be translated into physical addresses as part of managing virtual memory. However, the intrinsic evidence also indicates that this is not the only aspect of virtual memory schemes, and the Court finds that the claim language is broader than just the translation aspect. For example, the specification

discusses "dynamic swapping," as well as "mapping an address." '615 Patent at 3:63–65, 19:63–21:30. Indeed, the applicant argued during prosecution of the '615 Patent that "virtualization necessarily involves address remapping." (Dkt. No. 151-18 at 7 ('615 Patent FH, April 21, 2003 Office Action Response)). Thus, the Court finds that Plaintiff's construction is incomplete and unnecessary.

Turning to Defendants' construction, the Court finds that it is too broad because it requires only "support" of virtual memory and paging. As discussed above, the intrinsic evidence indicates that the memory is being managed, and not merely supported. Accordingly, the Court construes the term **"memory management logic/unit"** to mean **"a hardware device or circuit that manages virtual memory and paging."**

Regarding the phrases **"when interrupted," "interrupting processing by the graphics engine of the 3D graphics request code,"** and **"when the processing of 3D graphics request code is interrupted,"** the Court finds that these phrases appear in claim 1 of the '637 Patent. The Court further finds that the terms "interrupting" and "interrupted" are used consistently in the claims and are intended to have the same meaning in each claim. The Court also finds that the claim language indicates that it is the "3D graphics request code" that is "interrupted." Two of the disputed phrases explicitly recite that it is "3D graphics request code" that is "interrupted." Accordingly, the Court rejects Plaintiff's proposed "when one routine on a process" language. Furthermore, the Court will construe the term "3D graphics request code." Thus, any further dispute on what is being interrupted will be resolved by the Court's construction for that term.

Regarding the term "interrupted," the Court finds that the intrinsic and extrinsic evidence indicates that it means "suspended." The specification states that any processing of 3D graphics request code is only "stalled . . . and then . . . resumed" or "temporarily preempted" to allow for

the priority processing of an incoming video data stream, after which the processing of the normal request stream resumes:

> The critical point is that the Graphics Engine 42 does not have to complete the normal requests which have been partially processed before reading the Video FIFO. When data is available to be read from the Video FIFO, the [Graphics Engine] looks for an interruptible point at which to stall the normal request stream. *When that point is reached, the normal request stream is stalled, JPEG is processed, and then processing is resumed in the normal request stream.*

'637 Patent at 19:28–35 (emphasis added). Consistent with the idea of temporarily suspending and resuming, the applicant argued during prosecution that "[t]he claim also requires that the graphics request processing be temporarily preempted by real time processing of video stream data. . . . This requires a graphics processing arrangement that may be arbitrarily interrupted in the midst of such processing, so as to immediately begin processing of incoming video frame data." (Dkt 144-11 at 122 ('637 Patent FH, July 16, 1997 Office Action Response)). Similarly, the applicant argued that "the graphics engine of the present invention can be interrupted such that it stops the scan line at an interruptible point and resumes rendering of the scan line at a later point in time after it has processed the video data stream." (Dkt 151-2 at 11 ('637 Patent FH, January 17, 1997 Office Action Response)). Thus, the intrinsic evidence indicates "interrupted" means "suspended."

Turning to the extrinsic evidence, both sides cite to the McGraw-Hill Dictionary of Scientific and Technical Terms, which defines "interrupt" to mean "[t]o stop a running program in such a way that it can be resumed at a later time, and in the meanwhile permit some other action to be performed." (Dkt 151-5 at 6 (McGraw-Hill Dictionary of Scientific and Technical Terms)). Similarly, the IEEE Standard Dictionary of Electrical and Electronic Terms defines "interrupt" as "(1)(A) (software): The suspension of a process to handle an event external to the process." (Dkt 145-12 at 6 (IEEE Standard Dictionary of Electrical and Electronic Terms, Sixth

Ed. 1996)). Consistent with the intrinsic evidence, these definitions indicate that a person of ordinary skill in the art would understand "interrupted" to mean "suspended."

Defendants contend that their proposed construction does not dictate that processing cannot be resumed at a later time. (Dkt. No. 151 at 7) But this appears to be the crux of the parties' dispute. Although the Court agrees that the intrinsic evidence indicates that before the 3D graphics processing can be resumed, it must be stopped, the Court finds that Defendants' construction fails to capture the resuming aspect that was intended by the applicant's use of the term "interrupted." As Plaintiff notes, Defendants' citation to the McGraw-Hill Dictionary of Scientific and Technical Terms, omits the portion that states that the program is stopped "in such a way that it can be resumed at a later time." (Dkt. No. 151 at 7) (citing Dkt 151-5 at 6 (McGraw-Hill Dictionary of Scientific and Technical Terms)). Accordingly, the Court construes the phrase **"when interrupted"** to mean "**when suspended,"** the phrase **"interrupting processing by the graphics engine of the 3D graphics request code"** to mean "**suspending processing of 3D graphics request code,"** and the phrase **"when the processing of 3D graphics request code is interrupted"** to mean **"when the processing of 3D graphics request code is suspended."**

Regarding the term **"2D mode,"** the Court finds that the term appears in claim 5 of the '096 Patent. The Court further finds that neither party's construction is necessary or required. The claim language is straightforward and plainly states that the "texturing unit stores character data when in a 2D mode." Defendants contend that Plaintiff is trying to rewrite the claim by arguing that the 2D mode is not a mode of the texturing unit.

If this indeed is Plaintiff's position, it would be contrary to the intrinsic evidence, because the claims, specification, and the prosecution history all discuss the texturing unit storing

character data when in a 2D mode. For example, the specification states that "[t]exturing units, when operating in a 2D mode, are available for use for storing icons and characters locally to avoid the expense of doing a lookup from the host system." '096 Patent at 5:55–58. Likewise, during prosecution, the applicant argued that "nothing in [the prior art] appears to teach or suggest storing conventional texture data in a texturing unit when in a 3D mode, and storing character data in that texturing unit when in a 2D mode, as required by claim 5." (Dkt 151-7 at 3 ('096 Patent FH, February 16, 1998 Office Action Response)). The examiner agreed with the applicant, and stated that the claim was allowable because "none of the cited prior art shows a rendering method and system for processing 2D and 3D graphics comprising a texturing unit in which [sic] processes conventional texture data when in a 3D graphics mode; and stores character data when in a 2D mode as now claimed." (Dkt 151-8 at 3 ('096 Patent FH, May 12, 1998 Notice of Allowability)).

Notwithstanding, the Court is not persuaded that it needs to redraft "2D mode" as "a mode of the texturing unit," as Defendants proposed. The plain language of the claims provides an indication of what "mode" the texturing unit is in. In a 2D mode, the claim requires storing character data. In 3D graphics mode, the claim requires processing conventional texture data. Accordingly, the Court finds that it does needs to redraft the claim language as the parties propose. The Court notes that the parties appear to agree that "2D" means two-dimensional graphics in this context. Given this, and to clarify the term for the jury, the Court construes the term **"2D mode"** to mean **"mode for processing two-dimensional graphics."**

Regarding the term **"character data,"** the Court finds that the term appears in claims 2 and 5 of the '096 Patent. The Court further finds that the term is used consistently in the claims and is intended to have the same meaning in each claim. The Court also finds that the claims

differentiate between different types of data. For example, claim 2 recites "character data," claim 3 recites "tile pattern data," claim 5 recites "texture data," claim 6 recites "pattern data," and claim 7 recites "stipple pattern data." Thus, the Court agrees with Defendants that the applicant intended to differentiate between different types of data. Specifically, claim 5 requires that the texturing unit stores "character data."

The specification states that "[i]ndividual characters or character sets may be modified as they are mapped, e.g. to provide scaled or skewed local character sets." '096 Patent at 61:39–42. Accordingly, the Court finds that "character data" may be an individual letter. Likewise, the extrinsic evidence submitted by the parties generally defines a character as a "letter, digit, punctuation mark, or symbol." For example, the New World Dictionary of Computer Terms defines "character" as "Any symbol, digit, letter, or punctuation mark—including the blank character—stored or processed by computing equipment." (Dkt. No. 158-6 at 4 (New World Dictionary of Computer Terms)). Similarly, the IEEE dictionary defines "character" in the computer context as "[a] letter, digit, or other symbol that is used to represent information." (Dkt. No. 158-5 at 4 ((IEEE Standard Dictionary of Electrical and Electronic Terms, Sixth Ed. 1996)). Finally, Webster's Ninth New Collegiate Dictionary defines "character" as "a graphic symbol (as a hieroglyph or alphabet letter) used in writing or printing." (Dkt. No. 158-7 at 4 (Webster's Ninth New Collegiate Dictionary)). Accordingly, the Court finds that a person of ordinary skill in the art would understand "character data" to mean "data representing an individual letter, digit, punctuation mark, or symbol."

Turning to the parties' proposed constructions, the Court finds that Plaintiff's proposed "image data" language is too broad and could encompass all of the different types of data covered in the claims. As discussed, the claims indicate that the applicant intended to

differentiate between different types of data. Regarding Defendants' construction, the Court finds that it fails to capture the different types of character data, and instead is limited to text characters. As discussed, the extrinsic evidence indicates that "character" includes "letters, digits, punctuation marks, or symbols." Accordingly, the Court construes the term **"character data"** to mean **"data representing an individual letter, digit, punctuation mark, or symbol."**

## IV. CONSTRUCTION OF DISPUTED TERMS

### A. The '615, '061, '425 Patents

The parties' dispute focuses on the meaning and scope of six groups of terms/phrases in the '615 Patent, the '061 Patent, and the '425 Patent.

#### 1. "virtualizes/virtualizing" and "virtualizing"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "virtualizes" | "translates a large address space onto a smaller physical address space, and retrieves addressed data into memory that is not physically present" | "remaps a smaller physical address space into a larger virtual address space and swaps pages of" |
| "virtualizing" | "translating a large address space onto a smaller physical address space, and retrieving addressed data into memory that is not physically present" | "a larger virtual address space with pages of memory for swapping and remapping with a smaller physical address space" |

##### a) The Parties' Positions

The parties agree that the virtualize terms include a large address space and a smaller physical address space. The parties dispute whether the virtualize terms require a bidirectional swapping in and out of data and remapping of addresses, as Defendants propose. Plaintiff contends that virtual memory involves translation of a large virtual address space onto a smaller physical address space, and when a page is not physically present in the smaller physical memory, retrieving that addressed data into memory. (Dkt. No. 144 at 18) (citing Dkt. No. 145-5

at 63 (‘615 Patent FH, April 21, 2003 Office Action Response)). Plaintiff argues that Defendants’ construction of “swapping” implies a two-way data transaction, instead of one-way data transaction. (*Id.* at 19.) Plaintiff contends that this imposes a “swapping” requirement not normally found in virtualizing memory. (*Id.*)

Plaintiff further argues that the ‘615 Patent provides numerous examples of handling a page fault in virtual memory without any “swapping.” (*Id.*) (citing ‘615 Patent at 5:6–12, 7:43–65, 20:23–36). Plaintiff contends that these examples show that when a page is not present in the working set, it is loaded into the working set from the host memory. (*Id.* at 20) According to Plaintiff, handling the page fault is a one-way transaction in which data is loaded from the backing store into the working set. (*Id.*) Plaintiff argues that Defendants’ “swapping” requirement would exclude numerous disclosed embodiments in the specification. (*Id.*)

Plaintiff further argues that the Encyclopedia of Computer Science and Engineering, quoted during prosecution of the ‘615 Patent, makes clear that the page fault handling is generally a one-way transaction. (*Id.* at 20-21) (quoting Dkt. No. 145-5 at 66-67 (‘615 Patent FH, April 21, 2003 Office Action Response)). Plaintiff contends that the Encyclopedia states that moving the displaced data in the working set back to the backing store is not required to manage a page fault, but is an option. (*Id.* at 21.) Plaintiff agrees that the ‘615 Patent does refer to “swapping” in the context of virtual memory, but argues that it cannot be read to be a requirement in every circumstance. (*Id.*) (citing ‘615 Patent at 3:60–65, 4:1–3). Plaintiff contends that Defendants attempt to read a limitation only discussed peripherally in the specification into the claims, and would potentially exclude every example in the specification. (*Id.* at 22.)

Defendants respond that the intrinsic evidence makes clear that “virtualizing” is a

bidirectional concept encompassing the swapping in and out of data, and that remapping addresses is a necessary component of this concept. (Dkt. No. 151 at 19.) Defendants argue that Plaintiff's construction ignores these required operations and instead requires "translating addresses," which is only a part of the required remapping operation; and "retrieving data," which is only a part of the required swapping functionality. (*Id.*) Defendants contend that claims 3 and 5 of the '615 Patent involve an action of moving something from the closer (smaller) memory into a larger remote memory, i.e., swapping or paging out data. (*Id.*) Defendants further argue that this is the opposite of retrieving data from the remote memory locations. According to Defendants, Plaintiff's construction not only seeks to impose a unidirectional restriction on a bidirectional concept, but it also excludes the directional flow of data that the claims explicitly contemplate. (*Id.* at 20.)

Defendants further argue that the specification and prosecution history also confirm that virtualization requires "swapping." (*Id.* at 20) (citing '615 Patent at 3:60–65, 4:1); (citing Dkt. No. 151-23 at 31 ('425 Patent FH, March 2, 2004 Office Action Response)); (citing Dkt. No. 151-24 at 17 ('425 Patent FH, April 25, 2005 Appeal Brief)). Defendants contend that the inventor reiterated that "dynamic swapping is part of the virtualization process" (Dkt. No. 151-9, Baldwin Tr. 85:19-21; 100:6-12; 155:19-25). Defendants further argue that the prosecution history confirms that virtualization requires "address remapping." (Dkt. No. 151 at 20.) Defendants contend that to distinguish the Peddada reference, the applicants cited a dictionary and encyclopedia, and argued that "virtualization necessarily involves address remapping," and that "Peddada does not appear to teach . . . virtualization at all." (*Id.*) (quoting 151-18 at 7 ('615 Patent FH, April 21, 2003 Office Action Response)). Defendants further contend that remapping allows revising and updating these assignments in order to allow the virtual memory process to

work without using contiguous physical memory. (*Id.* at 21) (citing '615 Patent at 29:41–44).

Defendants further argue that the "translating" and "retrieving" functions recited in Plaintiff's construction fail to capture the entirety of the "remapping" and "swapping" operations that are required for virtualization. (*Id.*) Defendants contend that the intrinsic record clearly shows that virtualization requires a "remapping" and "swapping" operation. (*Id.*) Defendants argue that Plaintiff's construction seeks to include cache as a level of the virtualized memory. (*Id.*) Defendants contend that unlike virtual memory, retrieving data to and from a cache does not involve "remapping" or "swapping" operations. (*Id.*) Defendants further argue that Plaintiff's attempt to capture caches through the construction of virtualization is at odds with the applicant's repeated statements distinguishing virtualization from caching. (*Id.* at 22) (citing Dkt. No. 151-18 at 7–8 ('615 Patent FH, April 21, 2003 Office Action Response)); (citing Dkt. No. 151-31 at 5 ('425 FH, September 3, 2003 Office Action Response)). Defendants contend that these prosecution remarks follow the specification and the inventor's description of "virtualization." (*Id.*) (citing '615 Patent at 5:1–3); (citing Dkt. No. 151-9, Baldwin Tr. 95:14-18).

Plaintiff replies that its construction in no way restricts virtualizing to a unidirectional concept, and that Defendants point to nothing in Plaintiff's construction that imposes a unidirectional concept. (Dkt. No. 155 at 9.) Plaintiff contends that Defendants' construction imposes a bidirectional requirement through the term "swapping" that would exclude the preferred embodiment, and is contradicted by the inventor testimony. (*Id.*) Plaintiff argues that with respect to "memory management logic/unit," the preferred embodiment does not include swapping, and that the inventor repeatedly testified that swapping was not required for virtual memory. (*Id.*) Plaintiff argues that the portion of claim 5 of the '615 Patent referencing paging

out is entirely consistent with its position that claims 1 and 3 do not require "swapping." (*Id.*)

Plaintiff further argues that Defendants' undefined distinction between "address translation" and "address remapping" lacks merit. (*Id.*) Plaintiff argues that the file history for the '615 Patent uses both address translation and address remapping. (*Id.*) Plaintiff also contends that the extrinsic dictionary quoted by Defendants requires no more of "mapping" than "translating." (*Id.*) Plaintiff further argues that Defendants' reliance on inventor testimony is misleading because it is incomplete. (*Id.*) According to Plaintiff, Defendants are trying to use the term "remapping" to narrow the claims in some undefined way that neither the intrinsic evidence nor the extrinsic evidence supports. (*Id.*)

For the following reasons, the Court finds that the term **"virtualizes"** should be construed to mean **"translates a larger logical address space onto a smaller physical address space and enables the capability of swapping and paging."** Similarly, the Court finds that the term **"virtualizing"** should be construed to mean **"translating a larger logical address space onto a smaller physical address space and enabling the capability of swapping and paging."**

**b) Analysis**

The term "virtualizes" appears in claim 3 of the '615 Patent, and claims 6, 10, and 11 of the '425 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. The term "virtualizing" appears in claim 1 of the '615 Patent. The Court finds that the claim language indicates that "virtualizing" relates to translating a larger logical address space onto a smaller physical address space. For example, claim 3 recites "first memory management logic which virtualizes said graphics memory into a first portion of said main memory; and second memory management logic which virtualizes said main memory, including said first portion thereof, into said bulk storage unit." This is consistent with the

specification that states the following:

> Virtualization of texture memory, like virtualization of host memory, gives the user the impression of a memory space which is larger than can be physically accommodated in real memory. This is achieved by partitioning the memory space into a small physical working set and a large virtual set with dynamic swapping between the two.
>
> The swapping required for virtual memory management is normally done automatically (as far as the application software is concerned). There is a vast amount of literature concerning CPU based virtual memory systems and their management.

'615 Patent at 3:59–4:5. Likewise, the extrinsic evidence states that "[v]irtual memory is a way to provide the image of a large linear address space that is mapped onto a small physical address space." (Dkt. No. 151-29 at 12 (Efficient Memory Programming (1999)). Accordingly, the Court finds that "virtualizing" includes translating a larger logical address space onto a smaller physical address space.

The Court further finds that "virtualizing" also includes enabling the capability of swapping and paging. As indicated in the passage above, the specification states that "virtualization" is achieved "by partitioning the memory space into a small physical working set and a large virtual set with dynamic swapping between the two." '615 Patent at 3:63–65. The specification adds that "the swapping required for virtual memory management is normally done automatically (as far as the application software is concerned)." *Id.* at 4:1–3. Consistent with this statement, the specification further states that treating texture addresses as logical or virtual addresses "allows the dynamic paging of textures out of host or system memory with or without any assistance from the host CPU." *Id.* at 19:46–53. Moreover, during prosecution, the applicants argued that virtual memory allows swapping pages into memory. Specifically, the applicant stated the following:

> Virtual memory uses disk storage space to make the computer work as if it had more memory. When a file or program is too big for the computer to work with in

> its memory, part of the data is stored on disk. This virtual storage is divided into segments called pages; each page is correlated with a location in physical memory, or RAM. When an address is referenced, *the page is swapped into memory*; *it is sent back to disk when other pages must be called*. This allows the program to run as if all the data is in memory.

(Dkt. No. 151-23 at 31 ('425 Patent FH, March 2, 2004 Office Action Response)) (emphasis added). Accordingly, the intrinsic evidence indicates that "virtualizing" includes enabling the capability of swapping or paging.

The Court notes that this construction does not impose a "bidirectional" requirement. In other words, it does not require an action of moving something from the closer (smaller) memory into a larger remote memory, as Defendants argue. Instead, it only requires that the system be capable of this type of swapping. The Court agrees with Plaintiff; to find otherwise could exclude embodiments disclosed in the specification. Accordingly, the Court rejects this aspect of Defendants' construction, but agrees that the intrinsic evidence indicates that "virtualizing" includes providing the capability of swapping and paging. To that end, the Court disagrees with Plaintiff that requiring this capability automatically excludes disclosed embodiments. Specifically, the embodiments described in the '425 Patent do not state what happens to the "page just bumped out of memory." '425 Patent at 6:4–24. Moreover, this embodiment is not excluded by requiring the capability of swapping and paging. Therefore, the Court rejects Plaintiff's argument to the extent it contends that "virtualizing" does not require being capable of moving something from the closer (smaller) memory into the larger remote memory.

Regarding Defendants' "remapping" language, it is unclear if Defendants intend to argue that "remapping" is not the same as "mapping." The Court agrees that the applicants argued that "virtualization necessarily involves address remapping, whereas caching does not." (Dkt. No. 151-18 at 7 ('615 Patent FH, April 21, 2003 Office Action Response)). The Court also agrees

that the applicant argued that the prior art "DOES NOT HAVE ANYTHING TO DO WITH VIRTUAL MEMORY MANAGEMENT. . . . Peddada does discuss texture caching, but this is not the same." (Dkt. No. 151-31 at 5 ('425 FH, September 3, 2003 Office Action Response)) (emphasis in original). However, Defendants have not convinced the Court that this requires reading in a "remapping" requirement. Instead, it indicates that "virtualizing" is not the same as caching. Moreover, the Court finds that for a system to be capable of swapping or paging, it would necessarily have to be capable of mapping or translating addresses. In other words, the requirement of mapping, and potentially "remapping," is captured by the inclusion of swapping and paging in the Court's construction. Finally, the Court has considered the remaining extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court construes the term **"virtualizes"** to mean **"translates a larger logical address space onto a smaller physical address space and enables the capability of swapping and paging."** Similarly, the Court construes **"virtualizing"** to mean **"translating a larger logical address space onto a smaller physical address space and enabling the capability of swapping and paging."**

**2. "texture memory management function"**

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "texture memory management function" | "functions related to virtual memory management and cache management" | "function of the graphics accelerator that supports virtual memory and paging" |

**a) The Parties' Positions**

The parties dispute two issues: (1) whether "texture memory management function" is "of the graphics accelerator," as Defendants propose, and (2) whether "texture memory management function" necessarily includes "cache management," as Plaintiff proposes. Plaintiff

contends that the file history of the '061 Patent provides a clear definition of "texture memory management function." (Dkt. No. 144 at 16-17.) Regarding Defendant's construction, Plaintiff argues that it is too broad and only includes one component of the definition in the file history. (*Id.*) Specifically, Plaintiff argues that Defendants' reference to supporting "virtual memory and paging" only covers a part of the "virtual memory management" aspect of the Board's construction, and ignores the cache management aspect. (*Id.* at 18.) According to Plaintiff, memory management encompasses virtual memory, paging, and managing page faults. (*Id.*)

Defendants respond that the specification makes clear that the "texture memory management function" pertains to a function of the graphics accelerator (i.e., not a central processing unit ("CPU")). (Dkt. No. 151 at 16) (citing '061 Patent at Abstract, 9:23–24, 9:58–10:34, 2:18–21, FIG. 3). Defendants also cite to inventor testimony to support their construction. (*Id.* at 16) (citing Dkt. No. 151-9, Baldwin Tr. 159:5-12, 109:5-10, 140:7-14). Defendants further argue that during prosecution, the applicant repeatedly distinguished memory management functions requiring host (i.e., CPU) intervention from the purported invention that offloads that functionality from the host onto the "graphics accelerator." (*Id.* at 16-17) (citing Dkt. No. 151-20 at 13, 24 (('061 Patent FH, February 4, 2004 Appeal Brief)); (citing Dkt. No. 151-21 at 3 ('061 Patent FH, April 21, 2003 Office Action Response)).

Defendants also argue that the intrinsic evidence and inventor testimony demonstrate that the claimed "texture memory management function" need not include "cache management." (*Id.* at 17-18) (citing Dkt. No. 151-20 at 24-25 ('061 Patent FH, February 4, 2004 Appeal Brief)); (citing Dkt. No. 151-9, Baldwin Tr. 123:4-23, 140:7-14); (citing Dkt. No. 151-22 at 5 (Provisional App. 60/138,248)). Defendants contend that Plaintiff's construction misuses a statement made by the PTO relating memory management to cache management. (Dkt. No. 151

at 18.) Defendants argue that the PTO's statement referenced the specification which clarified that "'memory management' logic often performs functions related to virtual memory management as well as to cache management." (*Id.* at 18) (citing '061 Patent at 3:52–56). Defendants contend that this does not mean that it must perform both these functions, or that it necessarily does so when managing non-cache memory. (*Id.*) Finally, Defendants argue that the inventor testified that "cache management" in this context refers to a caching mechanism (i.e., the TLB) within the virtual memory system as opposed to the management of on-chip cache memory. (*Id.* at 18–19) (citing Dkt. No. 151-9, Baldwin Tr. 436:13–437:23).

Plaintiff replies that there is no need for a separate "of the graphics accelerator" requirement in "texture memory management function" because the claim language expressly requires it. (Dkt. No. 155 at 8.) Regarding the inclusion of "cache management," Plaintiff argues that Defendants' reliance on extrinsic inventor testimony cannot overcome the strong intrinsic evidence. (*Id.*)

For the following reasons, the Court finds that the phrase **"texture memory management function"** should be construed to mean **"functions related to virtual memory management and cache management."**

**b) Analysis**

The phrase "texture memory management function" appears in claims 1 and 2 of the '061 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. The claim language further indicates that the "texture memory management function" is "of the graphics accelerator." Claim 1 recites "[a] graphics processing chip, comprising: a graphics accelerator comprising rendering acceleration logic; and a texture memory management function, integrated on said chip." Plaintiff argues that including "of the

graphics accelerator" is unnecessary, given the claim language. The Court agrees.

However, the Court rejects Plaintiff's argument to the extent that it contends that memory management function requires host (i.e., CPU) intervention. During prosecution, the applicant distinguished memory management functions requiring host (i.e., CPU) intervention from one that offloads that functionality from the host onto the "graphics accelerator." For example, the applicants argued that "Peddada . . . does not disclose allowing the graphics accelerator itself direct access to the main memory or any other innovations with regard to graphics accelerators." (Dkt. No. 151-20 at 13 ('061 Patent FH, February 4, 2004 Appeal Brief)). When distinguishing another reference, the applicants argued that "Porterfield does teach virtual memory operations . . . but does not suggest that these operations are managed by the graphics accelerator. Indeed it appears that Porterfield only contemplates virtual memory management on the host side, NOT by the graphics accelerator."[7] (Dkt. No. 151-21 at 3 ('061 Patent FH, April 21, 2003 Office Action Response)) (emphasis in original).

The applicants later distinguished Porterfield again on the same basis, arguing that "Appellant is unable to find any mention of memory management by the graphics accelerator disclosed in Porterfield. Accordingly, Porterfield does not support this limitation of Claim 3 [which issued as Claim 1]." (Dkt. No. 151-20 at 24 ('061 Patent FH, February 4, 2004 Appeal Brief)). The applicants further argued that an advantage of the application was that "no assistance from the host CPU is required." (*Id.* at 18–19.) Based on the clear and unambiguous arguments made by the applicant during prosecution, the Court rejects any argument that memory management functions require host (i.e., CPU) intervention.

[7] The Court notes that this argument pertained to original Claim 2, and finds that this claim tracks original Claim 3 (*i.e.*, issued Claim 1) in significant respects, such as by reciting a "graphics processing chip, comprising: a graphics accelerator . . . comprising rendering acceleration logic."

Regarding the "cache management" issue, the Court finds that the intrinsic evidence indicates that "texture memory management function" necessarily includes "cache management." The specification explicitly states that the "[v]irtual memory management, like cache management, is an important architectural design choice, and 'memory management' logic often performs functions related to virtual memory management as well as to cache management." '061 Patent at 3:51–56. Likewise, during an appeal of an obviousness rejection to the Board of Patent Appeals and Interferences ("Board"), the applicants asserted that "[t]he present application also states, 'Virtual memory management, like cache management is an important architectural design choice, and 'memory management' logic often performs functions related to virtual memory management as well as to cache management.'" (Dkt. No. 145-6 at 323 ('061 Patent FH, May 7, 2004 Reply Brief)).

In response, the Board held that the term "texture memory management function" in claim 3, which issued as claim 1, "requires functions related to virtual memory management and cache management." (Dkt. No. 145-6 at 350 ('061 Patent FH, June 22, 2005 Board Decision)). The intrinsic evidence further indicates that the Board overruled the examiner's rejection based on that construction. (*Id.* at 351.) Finally, the examiner acknowledged the Board's ruling allowing claim 3, and explicitly referenced the Board's construction in the reasons for allowance. Specifically, the examiner stated:

> The following is an examiner's statement of reasons for allowance:
> The claims 3 and 6 are allowable over the prior art in record in view of BPAI Decision dated 6/22/05. The Board considered or read "the texture memory management function" in the claim performs functions related to virtual memory management as well as to cache management as disclosed in applicant's specification.

(Dkt. No. 145-6 at 367 ('061 Patent FH, Notice of Allowability)). Accordingly, the Court finds that the intrinsic evidence indicates that "texture memory management function" includes "cache

management."

Defendants argue that the PTO's statement during prosecution is taken out of context. Focusing on the word "often," Defendants argue that the PTO's statement referenced the specification, which clarified that "'memory management' logic often performs functions related to virtual memory management as well as to cache management." (Dkt. No. 151 at 18) (quoting '061 Patent at 3:52–56). The Court is not persuaded by Defendants' argument. The Board reversed the examiner, and the examiner later allowed the claims based on "texture memory management function" including "cache management." Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"texture memory management function"** to mean **"functions related to virtual memory management and cache management."**

**3. "manages page faulting of texture data" and "performs page faulting of said texture data"**

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "manages page faulting of texture data" | "fetching a virtually addressed page of texture data from physically addressed main memory" | "fetches a page of texture data and updates a page table when the page is not present" |
| "performs page faulting of said texture data" | "fetching a virtually addressed page of texture data from physically addressed main memory" | "fetches a page of said texture data and updates a page table when the page is not present" |

**a) The Parties' Positions**

The parties agree that the managing or performing "page faulting" involves fetching a page of texture data. The parties dispute whether managing or performing "page faulting"

involves "update[ing] a page table when the page is not present," as Defendants propose. Plaintiff contends that its construction fits exactly with what happens when a page fault occurs (i.e., a virtually addressed page is fetched from memory). (Dkt. No. 144 at 22.) Plaintiff argues that a page fault occurs when a table showing correspondence between virtual addresses and physical addresses is missing a physical address entry for a specific virtual address. (*Id.* at 23) (citing Dkt. No. 145-5 at 66–67 ('615 Patent FH, April 21, 2003 Office Action Response)). Plaintiff contends that during the course of fetching the missing page, the table will be updated to reflect the fact that there is now a physical address that corresponds to the virtual address. (*Id.*)

Regarding Defendants' construction, Plaintiff argues that Defendants may try to argue that updating a local cached version of a page table, such as a translation look aside buffer, does not satisfy this limitation. (*Id.*) Plaintiff contends that the translation look aside buffer is in effect a cache of a page table, and that updating such a buffer has the same effect as updating the page table itself. (*Id.*) According to Plaintiff, this would be regarded as managing page faulting of texture data in the same manner as updating the page table itself. (*Id.*)

Defendants respond that a "page fault" is commonly understood by persons of ordinary skill to mean an error condition that occurs when a unit of data, called a "page," is not present in physical memory. (Dkt. No. 151 at 23) (citing Dkt 151-27 at 9 (Computer Dictionary (1991)). According to Defendants, this meaning of page fault is reflected in their constructions by "when the page is not present" language. (*Id.*) Defendants further argue that the specification and file history explain that managing page faults requires a number of "automatic mechanisms," foremost of which is "updat[ing] the page tables." (*Id.*) (citing '425 Patent at 6:4–24); (citing Dkt. No. 151-24 at 3–8, 11–12 ('425 Patent FH, April 25, 2005 Appeal Brief)). According to Defendants, Plaintiff even admits that "[d]uring the course of fetching the missing page, the table

will be updated." (*Id.*) (quoting Dkt. No. 144 at 23).

Defendants further argue that Plaintiff excludes this essential requirement because it is contrary to its infringement theory, which is based on a different concept called "texture caching." (*Id.*) Defendants contend that texture caching (or managing "cache misses") and managing page faults are very distinct concepts. (*Id.* at 24.) Defendants further argue that this distinction was drawn very clearly in prosecution during discussion of the Peddada prior art. (*Id.*) (citing Dkt. No. 151-31 at 5 ('425 FH, September 3, 2003 Office Action Response)).

Defendants appear to agree that their construction may exclude "updating . . . a translation look aside buffer [TLB]" from the scope of the disputed term. (*Id.* at 24.) However, Defendants contend that this exclusion naturally follows from the applicants' own arguments distinguishing management of page faults from updating a TLB. (*Id.*) Defendants argue that the applicant argued that updating a TLB "only means that the address translation is missing [and] does not mean that the data is not present, it is not a page fault." (*Id.*) (citing Dkt. No. 151-35 at 3, 5–6 ('425 Patent FH, December 19, 2006 Office Action Response)); (citing Dkt. No. 151-36 at 5–9 ('425 Patent FH, August 27, 2009 Board Decision)). Finally, Defendants argue that the specification never references a page fault in relation to a TLB. (*Id.*) (citing '425 Patent at 9:35, 24:32–38, 26:40–41, 32:18–20, 44:49–51, 46:23–27; and 22:37–58).

Plaintiff replies that Defendants agree that their construction would exclude updating an entry in a TLB. (Dkt. No. 155 at 10.) Plaintiff argues that the applicant never distinguished the management of page faults from updating a TLB, but instead distinguished it from a TLB miss. (*Id.*) Plaintiff argues that a TLB miss is not "updating a TLB," as Defendants assert. (*Id.*) According to Plaintiff, Defendants' construction should be rejected because they agree that it will exclude updating a TLB, which Plaintiff contends is a way to handle a page fault. (*Id.*)

For the following reasons, the Court finds that the phrase **"manages page faulting of texture data"** should be construed to mean **"fetches a page of texture data and updates a page table when the page of texture data is in the second level of memory (the host's physical memory)."** Similarly, the Court finds that the phrase **"performs page faulting of said texture data"** should be construed to mean **"fetches a page of texture data and updates a page table when the page of texture data is in the second level of memory (the host's physical memory)."**

**b) Analysis**

The phrase "manages page faulting of texture data" appears in claims 1 and 2 of the '425 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. The phrase "performs page faulting of said texture data" appears in claims 6, 10, and 11 of the '425 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. The Court further finds that the intrinsic evidence indicates that managing or performing "page faulting of texture data" requires fetching a page of texture data and updating a page table when the page of texture data is in the second level of memory (the host's physical memory). Specifically, the specification states:

> In particular, the present application discloses a computer system in which a graphics accelerator unit *manages page faulting of texture data* invisibly to the host processor.
>
> *When a logical page fault occurs* and *the page of texture is in the second level of memory (i.e. the host's physical memory), it will be fetched in automatically* by the graphics memory manager, and the host is not aware anything has happened. In a preferred embodiment, *a number of automatic mechanisms* would be in place for this to happen:
>
> a. Determine where the page is located *in host physical memory*.

> b. Determine which page out of the working set (in level 1 memory) to use. In a sample embodiment, this determination uses the least recently used algorithm.
>
> c. Make this page the most recently used page (as well as continuing to keep the least-recently-used list up to date as other pages are used).
>
> d. *Update the page tables for the new page and remove any reference to the page just bumped out of memory (if any).*
>
> e. Download the page.
>
> f. Restart texture processing.
>
> Note that if the faulting logical page identifies a page in the third level memory the host does (a) (after having made the page available), but the hardware carries on and does b, c, d, e and f.

'425 Patent at 6:1–24 (emphasis added). The specification indicates that this process occurs "automatically" by the graphics memory manager and ensures that the "host is not aware anything has happened." *Id.*

During prosecution, the applicant reiterated that managing or performing "page faulting of texture data" requires automatically locating and fetching a page of texture data, and then updating a page table when a logical page fault occurs. (Dkt. No. 151-24 at 3–8, 11–12 ('425 Patent FH, April 25, 2005 Appeal Brief)). In addition, the Board noted that "Appellant also contends (App. Br. 13) that Kaiser's 'definition of a page fault is common and well established, and like Appellant's Specification, involves a 'second level of memory (i.e. the host's physical memory)." (Dkt. No. 151-36 at 7 ('425 Patent FH, August 27, 2009 Board Decision)). Finally, Plaintiff stated in its brief that "[d]uring the course of fetching the missing page, the table will be updated." (Dkt. No. 144 at 23.) Accordingly, the Court finds that managing or performing "page faulting of texture data" requires fetching a page of texture data and updating a page table when the page of texture data is in the second level of memory (the host's physical memory).

In support of their construction, Defendants argue that Plaintiff offers alternative

language to advance its infringement theory by asserting that a concept called "texture caching" infringes. (Dkt. No. 151 at 23.) Defendants argue that "texture caching" and managing page faults are very distinct concepts. (*Id.*) The Court does not necessarily agree that the two are completely unrelated, but does agree that "texture caching" by itself would not be considered managing page faults. As the applicant clearly stated during prosecution, "Peddada DOES NOT HAVE ANYTHING TO DO WITH VIRTUAL MEMORY MANAGEMENT. It is not correct that Peddada shows 'page faulting,' and indeed Peddada does not appear to refer anywhere to 'virtual' memory nor to 'page fault' nor 'page faulting.' Peddada does discuss texture caching, but this is not the same." (Dkt. No. 151-31 at 5 ('425 FH, September 3, 2003 Office Action Response)) (emphasis in original). Therefore, the Court rejects Plaintiff's argument to the extent that it contends that managing or performing "page faulting of texture data" only requires "texture caching."

However, this is not to conclude that managing or performing "page faulting of texture data" excludes or cannot include aspects of "texture caching," (*e.g.*, updating a local cached version of a page table, such as a translation look aside buffer), it just requires more than this alone. Indeed, the specification states the following:

> If the logical page is not resident in the working set then details about the page (its host address, target memory pool, etc.) is made available to the host or DMA controller. (The DMA controller is in Gamma for RXs or is integrated into P3.) Sometime later the working set has been updated with the new page of texture data and … [t]he TLB is updated so the next time this logical page is accessed the physical page is to hand.

'425 Patent at 21:30-39. Finally, Defendants contend that the applicant distinguished management of page faults from updating a TLB, and therefore updating a TLB should be excluded from the scope of the disputed term. (Dkt. No. 151 at 24.) The Court disagrees. The applicant did not distinguish updating a TLB from a page fault, but instead distinguished a TLB

miss from a page fault. Specifically, the applicant argued:

> It is respectfully submitted that Kaiser's teaching does not teach the limitations in the claims of the present application. Specifically, Applicant respectfully submits that what Examiner refers to as a "less extreme page fault" (for example, at page 2 of the present Office action) is not a page fault, *but is instead a situation where the address translation of command data is not found in the TLB*. This only means that the address translation is missing. It does not mean that the data is not present, and it is not a page fault.
>
> . . . .
>
> Hence, what Examiner refers to as a "less extreme page fault" is not a page fault at all. It appears that Examiner is referring to the fact that there is no TLB entry that gives a translation for the virtual address sent to the graphics processor. When no such translation exists, a translation is needed. Hence, the memory controller performs a page walk and "fetches cache lines form the system page table on an as-needed basis to find an effective real address translation." This activity is, respectfully, not a page fault.

(Dkt. No. 151-35 at 3–4 (‘425 Patent FH, December 19, 2006 Office Action Response)) (emphasis added). As indicated, the applicant clearly and unambiguously argued that a TLB miss is not the recited “page fault.” The Board of Patent Appeals agreed and stated that “[t]herefore, Appellant has established that the Examiner erred in determining that skilled artisans, in light of the Specification, would have understood Kaiser’s TLB miss to include a page fault as required by claim 1.” (Dkt. No. 151-36 at 8 (‘425 Patent FH, August 27, 2009 Board Decision)).

This understanding is consistent with the specification because it does not reference a page fault in relation to a TLB. Instead the specification always refers to a “TLB miss,” not a “page fault.” *See, e.g.*, ‘425 Patent at 9:35, 24:32–38, 26:40–41, 32:18–20, 44:49–51, and 46:23–27; *cf. id.* at 22:37–58 (detailing page faults). Accordingly, the Court rejects Defendant’s argument to the extent that it contends that a TLB miss, by itself, would be understood to be the recited page fault. As the Board of Patent Appeals stated, “Appellant's arguments demonstrate that while the skilled artisans may interpret a TLB miss with host processor involvement as a page fault, the Examiner has not established that without such involvement (as called for in the claims and disclosed by Appellant), skilled artisans would so interpret a TLB miss.” (Dkt. No.

151-36 at 7 ('425 Patent FH, August 27, 2009 Board Decision)).

During the claim construction hearing, Plaintiff expressed concerns that the Court's construction would exclude the CPU's involvement from the scope of the claims. Plaintiff's concerns are unfounded because the claim language itself explicitly states when the CPU is involved. For example, claim 2 recites "a graphics accelerator unit which manages page faulting of texture data, from main memory used by at least one host processor into a dedicated graphics memory, *invisibly to the host processor, except when said graphics accelerator unit calls for data which has not recently been present in said main memory.*" '425 Patent at claim 2 (emphasis added). Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"manages page faulting of texture data"** to mean **"fetches a page of texture data and updates a page table when the page of texture data is in the second level of memory (the host's physical memory)."** Similarly, the Court construes the phrase **"performs page faulting of said texture data"** to mean **"fetches a page of texture data and updates a page table when the page of texture data is in the second level of memory (the host's physical memory)."**

**4. "dedicated graphics memory," "specialized graphics memory," and "normal texture memory"**

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "dedicated graphics memory" | "memory that is local to the graphics processor" | '615 Patent claim 3<br>"primary memory for storing graphics data"<br><br>'425 Patent claims 2, 6, 11<br>"primary memory of the graphics accelerator unit for storing graphics data" |

| | | |
|---|---|---|
| "specialized graphics memory" | "memory that is local to the graphics processor" | "primary memory for storing graphics data" |
| "normal texture memory" | "memory that is local to the graphics processor" | "primary memory of the graphics accelerator for storing texture data" |

**a) The Parties' Positions**

The parties dispute whether the "dedicated graphics memory" is a particular memory, *i.e.*, the "primary memory" of the graphics processor "for storing graphics data," as Defendants propose, or if it can be any memory local to the graphics processor, as Plaintiff proposes. Plaintiff contends that its construction better captures what is intended by the '615 Patent and is less confusing to the jury. (Dkt. No. 144 at 24.) Plaintiff argues that the Provisional Application is incorporated by reference in both the '425 Patent and the '615 Patent, and that it describes the dedicated graphics memory as "local memory." (*Id.*) (quoting Dkt 145-15 at 33–34 (Provisional Application)). Plaintiff contends that Defendants' constructions are confusing because it is unclear what is meant by primary memory in this context. (*Id.*) Plaintiff argues that Defendants attempt to broaden the limitation by defining it by the function it performs in an effort to read dedicated graphics memory on asserted prior art. (*Id.*) Finally, Plaintiff argues that the claims of the '425 Patent contain no similar limitations referring to "primary," and that construing the common term from both patents using a limitation from a single claim of the '615 Patent is improper. (*Id.*)

Defendants respond that given the many memory types, Defendants seek to clarify that "dedicated graphics memory" refers to the primary or principal memory of the graphics accelerator. (Dkt. No. 151 at 25.) Defendants argue that their construction follows from the plain language of claim 3 that equates the disputed "dedicated graphics memory" with "primary memory." (*Id.*) Defendants contend that the characterization of this memory as "primary" is consistent with the specification and inventor testimony. (*Id.* at 25–26) (citing '615 Patent at

4:66–5:1); (citing Dkt. No. 151-9, Baldwin Tr. 71:5–9).

Defendants also contend that Plaintiff's construction disregards this evidence and seeks instead to broaden the meaning of "dedicated graphics memory" by adopting a vague unsupported phrase "local to the graphics processor." (*Id.* at 26.) Defendants argue that Plaintiff's construction might permit the "dedicated graphics memory" to encompass any of the recited memories in the claims, including the "on-chip cache" in dependent claim 9. (*Id.*) Defendants argue that this should be rejected because claim 3 of the '615 Patent recites that the "dedicated graphics memory" must be part of a virtual memory system. (*Id.*) Defendants further argue that the prosecution history, specification, and inventor testimony illustrate that Plaintiff is attempting to stretch "dedicated graphics memory" to cover concepts that were never intended and expressly distinguished. (*Id.*) (citing Dkt. No. 151-18 at 7 ('615 Patent FH, April 21, 2003 Office Action Response)); (citing '615 Patent at 5:1–3); (citing Dkt. No. 151-9, Baldwin Tr. 75:5–8, 80:24–81:3; 81:19–23; and 150:9–13). Finally, Defendants argue that during prosecution the applicant stated that "dedicated graphics memory . . . stores data associated with graphics processing, for example, texture data, or other data used by graphics processor." (*Id.* at 26–27) (citing Dkt. No. 151-39 at 21 ('425 Patent FH, July, 19, 2007 Appeal Brief)).

Plaintiff replies that Defendants did not address the intrinsic evidence cited by Plaintiff in its Opening Brief. (Dkt. No. 155 at 10.) Plaintiff also argues that Defendants did not address the distinction between their proposed construction and the express limitation of claim 3, or how the claims of the '425 Patent do not contain a "primary" limitation. (*Id.*)

For the following reasons, the Court finds that the terms **"dedicated graphics memory," "specialized graphics memory,"** and **"normal texture memory"** should be construed to mean **"memory used primarily by a graphics processor for storing data associated with graphics**

**processing."**

**b) Analysis**

The term "dedicated graphics memory" appears in claim 3 of the '615 Patent and claims 2, 4, 6, 10, and 11 of the '425 Patent. The Court finds that the term is used consistently in the claims and is intended to have the same meaning in each claim. The term "specialized graphics memory" appears in claim 1 of the '615 Patent. The term "normal texture memory" appears in claim 1 of the '061 Patent and claim 11 of the '425 Patent. The Court finds that the term is used consistently in the claims and is intended to have the same meaning in each claim. The Court notes that the parties generally contend that the terms should be construed the same.

Turning to the claim language, the Court finds that the claims indicate that the "dedicated graphics memory" is not just any memory, but instead is virtualized memory. For example, claim 3 of the '615 Patent recites that the first memory management logic "virtualizes said graphics memory into a first portion of said main memory." Likewise, claim 9 differentiates the recited "dedicated graphics memory" from "on-chip cache memory." Furthermore, the specification states that, unlike the recited "dedicated graphic memory," the "on-chip cache . . . is not particularly involved with the virtual memory system." '615 Patent at 5:2–4. However, all of the claims do not require that the recited "dedicated graphics memory" be the "primary memory," as Defendants contend.

Instead, consistent with the prosecution history, the claims indicate that "the recited "dedicated graphics memory" is "memory used primarily by a graphics processor for storing data associated with graphics processing." The plain language of the disputed terms implies this conclusion by reciting the words "dedicated," "specialized," and "normal." Moreover, the applicants argued that "[a] dedicated graphics memory, on the other hand, stores data associated

with graphics processing, for example, texture data, or other data used by a graphics processor. A dedicated graphics memory is used primarily or only by a graphics processor." (Dkt. No. 151-39 at 20–21 ('425 Patent FH, July, 19, 2007 Appeal Brief)). Accordingly, the Court finds that the terms should be construed as "memory used primarily by a graphics processor for storing data associated with graphics processing."

Turning to the parties' construction, the Court is not persuaded by Plaintiff's argument that the intrinsic evidence indicates that the terms refer to "memory that is local to the graphic processor." The intrinsic evidence cited by Plaintiff refers to on-chip cache, and does not state that it is the "dedicated graphics memory." Instead, the provisional application states that "[i]n the AGP execute model, PERMEDIA 3 uses both the local memory and the system memory as primary graphics memory," and "[t]hat the two memory systems are logically equivalent and textures in system memory are no longer copied to local memory but executed directly from system memory." (Dkt. No. 145-15 at 34 (Provisional Application)). Consistent with the claims, this indicates that the on-chip cache memory (i.e., local memory) is distinct from the "dedicated graphics memory" (*i.e.*, the system memory). Thus, the Court rejects Plaintiff's argument to the extent that it contends that the "on-chip cache memory" can be the "dedicated graphics memory," the "specialized graphics memory," or the "normal texture memory."

However, this does not completely exclude all involvement of the "on-chip cache memory," instead it only excludes the "on-chip cache memory" from being the memory that is primarily used by the graphics processor for storing data associated with graphics processing. Indeed, dependent claim 9 of the '615 Patent explicitly recites that the "specialized graphics-processing logic comprises an on-chip cache memory." This indicates that the "on-chip cache memory" is distinct from the "dedicated graphics memory."

Regarding Defendants' construction, the Court finds that the phrase "primary memory" is confusing because it is unclear what the term means. As discussed above, the provisional application states that both "PERMEDIA 3 uses both the local memory and the system memory as primary graphics memory." (Dkt. No 145-15 at 34 (Provisional Application)). Moreover, only the claims in the '615 Patent use the term "primary memory." The claims in the '425 Patent and the '061 Patent do not recite "primary memory." The Court is not persuaded that this language should be applied across all of the patents for these disputed terms. Instead, as discussed above, the intrinsic evidence indicates that the "dedicated graphics memory is used primarily by a graphics processor." (Dkt. No. 151-39 at 20–21 ('425 Patent FH, July, 19, 2007 Appeal Brief)). Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

#### c) Court's Construction

In light of the intrinsic and extrinsic evidence, the Court construes the terms **"dedicated graphics memory," "specialized graphics memory,"** and **"normal texture memory"** to mean **"memory used primarily by a graphics processor for storing data associated with graphics processing."**

### 5. "graphics processing chip"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "graphics processing chip" | "an integrated circuit containing a rendering accelerator for 3D graphics processing" | "an integrated circuit for generating graphical images independently of a central processing unit" |

#### a) The Parties' Positions

The parties agree that the recited "graphics processing chip" is an integrated circuit. The parties dispute two issues: (1) whether the circuit must generate graphical images independently of a central processing unit, and (2) whether the circuit is limited to 3D graphics processing.

Plaintiff contends that Defendants' construction would exclude all of the graphics processing chips described in the '061 Patent by incorporating the requirement that a graphics processing chip "generat[e] graphical images independently of a central processing unit." (Dkt. No. 144 at 25.) According to Plaintiff, this is incorrect because graphics images are not generated independently of the central processing unit on the graphics processors described in the '061 Patent. (*Id.*) Plaintiff contends that the '061 Patent provides many examples of host processor or central processing unit involvement in generating graphical images. (*Id.*) (citing '061 Patent at 2:65–3:1, 11:19–26, 23:28–40, 25:33–38, 27:59–64, 33:40–49, 51:9–18, and 51:29–52:26).

Defendants respond that the claims and specification confirm that the claimed "graphics processing chip" refers to a specialized chip that is separate and distinct from the CPU or "host." (Dkt. No. 151 at 27) (citing '061 Patent at Abstract, 2:18–21). Defendants contend that claim 1 indicates that the "graphics processing chip" cannot be the CPU, because that would contradict the claimed invention's purpose to not have the CPU perform texture memory management. (*Id.* at 28.) Defendants further argue that their construction is also consistent with how persons of ordinary skill understood the term "graphics processing chip" at the time of the invention. (*Id.*) (citing Dkt 151-40 at 6 (Dictionary of Computing & Comm'n (2003)); (citing Dkt 151-28 at 6 (Computer Dictionary (1997)).

Defendants also contend that their construction does not preclude the CPU from having some secondary "involvement" in generating graphical images. (*Id.*) Defendants argue that it instead requires that the responsibility for generating graphical images be offloaded to a circuit that is separate and independent from the CPU. (*Id.*) Finally, Defendants argue that Plaintiff improperly limits the claim term "graphics processing chip" to "3D graphics processing." (*Id.* at 29.) Defendants contend that neither the disputed term nor the rest of the claim refers to 3D

graphics, and that Figure 3 illustrates "a block diagram of a graphics processor which can incorporate the disclosed embodiments" that includes "2D, 3D and Video Graphics Core." (*Id.*) (quoting '061 Patent at 10:39–40).

Plaintiff replies that Defendants essentially concede that the '061 Patent does not describe the graphics processing chip as generating graphical images "independently of a central processing unit," as Defendants propose. (Dkt. No. 155 at 11.) Plaintiff argues that Defendants' "secondary involvement" assertion would be difficult for a jury to understand. (*Id.*) Plaintiff further contends that the reason "3D" is included in its construction is because that is how the Board characterized its inventions. (*Id.*)

For the following reasons, the Court finds that the term **"graphics processing chip"** should be construed to mean **"an integrated circuit used to offload rendering and texture memory management duties from the central processing unit."**

**b) Analysis**

The phrase "graphics processing chip" appears in the preamble of claim 1 of the '061 Patent. The parties agree that the graphics processing chip is an integrated circuit. The claim language further indicates that the graphics processing chip is used for rendering and texture memory management function. Claim 1 recites "a graphics processing chip, comprising: a graphics accelerator comprising rendering acceleration logic; and a texture memory management function, integrated on said chip, which manages both texture storage in host memory and also texture storage in normal texture memory." The specification further indicates that by performing these duties, the graphics processor offloads them from the central processing unit.

Specifically, the specification states that "[s]ince rendering is a computationally intensive operation, numerous designs have offloaded it from the main CPU. An example of this is the

GLINT chip described below." '061 Patent at 2:17–20. The Abstract also describes a "text caching controller, located on the graphics card" that "offloads texture memory management duties from the host." '061 Patent at Abstract. The specification further states that "the virtual texture mapping architecture described in the present application include at least the following: A single chip solution is provided . . . [t]he page faulting is all done in hardware with no host intervention." '061 Patent at 9:1–6.

In addition, the applicant distinguished prior art memory management features as being on the host and not the graphics accelerator. Specifically, the applicant argued that "Porterfield does teach virtual memory operations . . . but does not suggest that these operations are managed by the graphics accelerator. Indeed it appears that Porterfield only contemplates virtual memory management on the host side, NOT by the graphics accelerator." (Dkt. No. 151-21 at 3 ('061 Patent FH, April 21, 2003 Office Action Response)) (emphasis in original). Accordingly, the Court finds that "graphics processing chip" is "an integrated circuit used to offload rendering and texture memory management duties from the central processing unit."

Regarding Plaintiff's construction, the specification indicates that the graphics processor is not limited to 3D processing. For example, the specification states that Figure 3 illustrates "a block diagram of a graphics processor which can incorporate the disclosed embodiments" that includes "2D, 3D and Video Graphics Core." '061 Patent at 10:39–40, FIG. 3. The Court is not persuaded that the Board's general description of "3D" in a Background section requires limiting the claims to 3D. In addition, the claim language recites "a graphics accelerator comprising rendering acceleration logic," and there is no reason to repeat this claim language in the construction.

Regarding Defendants' construction, the Court finds that the '061 Patent does not

describe the graphics processing chip as generating graphical images "independently" of a central processing unit, as Defendants propose. In fact, the '061 Patent provides a number examples of host processor or central processing unit involvement in generating graphical images. '061 Patent at 2:65–3:1, 11:19–26, 23:28–40, 25:33–38, 27:59–64, 33:40–49, 51:9–18, and 51:29–52:26. Furthermore, the Court is not convinced that a jury would understand how a CPU can have "secondary involvement," if the Court's construction requires "generating graphical images independently of a central processing unit," as Defendants propose. Instead, the Court finds that the intrinsic evidence indicates that the recited "graphics processing chip" is used to offload rendering and texture memory management duties from the central processing unit. Accordingly, the Court does not adopt either parties' construction.

During the claim construction hearing, Plaintiff proposed to alter the Court's construction to state that the integrated circuit "partially" offloads the duties from the CPU. The Court rejects Plaintiff's argument that the "graphics processing chip" is only "partially" involved in performing the recited duties. Indeed, this would be inconsistent with the intrinsic evidence discussed above. However, this does not mean that the CPU cannot have any involvement. As Defendants stated in their briefing and represented to the Court during the claim construction hearing, they are not taking the position that there is no host CPU involvement. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court construes the term **"graphics processing chip"** to mean **"an integrated circuit used to offload rendering and texture memory management duties from the central processing unit."**

### 6. "invisibly to the host processor" and "invisibly to said CPU"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "invisibly to the host processor" | "without intervention of the CPU" | "without awareness of said CPU" |
| "invisibly to said CPU" | "without intervention of the CPU" | "without awareness of said CPU" |

#### a) The Parties' Positions

The parties dispute whether "invisibly" means without awareness by the host/CPU, as Defendants propose, or merely that there is lack of intervention by the host/CPU, as Plaintiff proposes. Defendants contend that the plain meaning, specification, and prosecution history all support Defendants' construction that "invisibly" means "without awareness." (Dkt. No. 153 at 4) (quoting '425 Patent at 6:1–8); (citing Dkt. No. 151-24 at 14 ('425 Patent FH, April 25, 2005 Appeal Brief)). Defendants argue that Plaintiff's construction conflicts with the plain meaning of the term "invisibly." (*Id.* at 5.) Defendants argue that Plaintiff's construction would improperly cover a system where the host does not intervene even though it is aware something has happened. (*Id.*) Defendants argue that this is inconsistent with the preferred embodiment of the '425 Patent, in which "the page of texture . . . will be fetched in automatically by the graphics memory manager, and the host is not aware anything has happened." (*Id.*) (quoting '425 Patent at 6:4–8).

Plaintiff responds that the passage quoted by Defendants states that the fault will be handled "automatically by the graphics memory manager." (Dkt. No. 158 at 4) (quoting '425 Patent at 6:1–8). Plaintiff contends that Defendants ignore this language and incorrectly focus on the "host is not aware anything has happened" language in this passage. (*Id.*) Plaintiff further argues that during oral argument before the Board on appeal, applicant's counsel equated the

term “invisibly” and “automatically,” and said that they mean “without intervention by the CPU.” (*Id.* at 4–5) (quoting Dkt. 145-8 at 314–16 (Hearing Tr. at 17:6-19:2)). According to Plaintiff, the intrinsic evidence supports its construction. (*Id.* at 5.)

For the following reasons, the Court finds that the phrase **“invisibly to the host processor”** should be construed to mean **“without involvement of the host processor,”** and that the phrase **“invisibly to said CPU”** should be construed to mean **“without involvement of the CPU.”**

**b) Analysis**

The phrase “invisibly to the host processor” appears in claims 1 and 2 of the ‘425 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. The phrase “invisibly to said CPU” appears in claims 6, 10, and 11 of the ‘425 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. The claim language further indicates that “invisibly to the host processor/CPU” means without involvement of the host processor/CPU. Claim 2 recites “a graphics accelerator unit which manages page faulting of texture data, . . . invisibly to the host processor, except when said graphics accelerator unit calls for data which has not recently been present in said main memory.” A person of ordinary skill in the art would understand that this language indicates that managing page faults is done “invisibly” or without involvement of the host processor, unless the host processor has to locate data which is not present in the main memory. This is consistent with the specification, which states:

> In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data *invisibly to the host processor*.
>
> When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host’s physical memory), *it will be fetched in automatically by the graphics memory manager*, and the host is not aware anything has happened.

‘425 Patent at 6:1–8 (emphases added); *see also* 6:37–38 (“The page faulting is all done in hardware with no host intervention.”). As indicated, fetching a page of texture “automatically” means without involvement of the host. Likewise, in discussing the format of the entries in the Logical Page Table, the specification states that “[s]etting [Bit No. 43] will generate an interrupt and *involve the host* in providing this page of texture data. When this bit is 0 the HostPage is the physical page and will be read directly with no host intervention. This field is maintained by the host.” ‘425 Patent at 25:1–20 (emphasis added). Similar to claim 2, this discloses a scenario where the host may or may not be involved.

This is consistent with the Board’s statement that the applicant argued that “invisibly to the host processor” meant “no host processor involvement.” Specifically, the Board stated “while Appellant asserts (App. Br. 14) that ‘[v]arious claims of the present application claim *no host processor involvement* in a page fault (e.g., texture data is fetched ‘automatically’ or ‘invisibly’ to the host processor),’ claims 4, 5, 7, and 16-20 do not recite any such ‘page fault’ limitation, nor any management thereof as invisible to the host processor.” (Dkt. No. 151-36 at 8 (‘425 Patent FH, August 27, 2009 Board Decision)). Accordingly, the Court finds that “invisibly to the host processor/CPU” means without involvement of the host processor/CPU.

Regarding Defendants’ construction, the Court is not persuaded that the intrinsic evidence indicates that “invisibly” means “without awareness.” Defendants are correct that the specification states that “[w]hen a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host’s physical memory), it will be fetched in automatically by the graphics memory manager, and the host is not aware anything has happened.” ‘425 Patent at 3:4–9. Defendants place emphasis on “the host is not aware anything has happened” portion of this passage. However, unlike Defendants, the Court finds that this statement emphasizes that the

page of texture will be fetched automatically. In other words, the page of texture will be fetched without involvement of the host processor/CPU. This is consistent with the other intrinsic evidence discussed above.

Regarding Plaintiff's construction, the Court does not adopt it because "intervention" could be interpreted to mean that the host processor/CPU is involved, but decides not to intervene. As discussed, the claims require the graphic accelerator unit to fetch the data automatically and without involvement of the host processor/CPU. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

#### c) Court's Construction

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"invisibly to the host processor"** to mean **"without involvement of the host processor,"** and that the phrase **"invisibly to said CPU"** to mean **"without involvement of the CPU."**

### B. The '156 Patent

The parties' dispute focuses on the meaning and scope of three groups of terms/phrases in the '156 Patent.

#### 1. "is logically asynchronous," "asynchronous operations thereof," "operating asynchronously to," and "decouples operations"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "is logically asynchronous" | "does not have to work on the same instruction at the same time" | "not having to run at a fixed delay to" |
| "asynchronous operations thereof" | Plain and ordinary meaning<br>Alternatively, "said sequencer and said plurality of processing elements to operate asynchronously" | "the sequencer and the plurality of SIMD processing elements to not run at a fixed delay" |

| | | |
|---|---|---|
| "operating asynchronously to" | Plain and ordinary meaning<br>Alternatively, "does not have to work on the same instruction at the same time" | "not having to run at a fixed delay to" |
| "decouples operations" | Plain and ordinary meaning<br>Alternatively, "allows asynchronous operations" | "allows operations not to run at a fixed delay" |

**a) The Parties' Positions**

The parties dispute whether "asynchronous" means that the sequencer and the processing elements "do not have to work on the same instruction at the same time," as Plaintiff proposes, or if it means that the sequencer and the processing elements are "not having to run at a fixed delay," as Defendants propose. Plaintiff contends that the specification indicates that instruction processing at the sequencer is decoupled from instruction processing at the processing elements. (Dkt. No. 144 at 28–29) (citing at '156 Patent at Abstract, 3:58–4:5, 21:53–22:2, 3:34–56, 16:47–64, 21:54–22:2, 3:41–49, 16:55–64, 21:54–22:2, 22:3–10, 22:11–23, and 3:37–56). Plaintiff argues that this means that the rate of instruction processing typically is faster in the sequencer than it is in the processing elements. (*Id.* at 29.) According to Plaintiff, the concept of asynchronous operation is focused on the ability of the sequencer to work on different instructions than the processing elements at the same time. (*Id.*)

Plaintiff further argues that the specification states that the processing elements and the sequencer are "logically asynchronous," and then defines "logically asynchronous" as "i.e. decoupled, in that they do not have to work on the same instruction at the same time." (*Id.* at 29–30) (citing '156 Patent at 4:1–3, 21:66–22:1). Plaintiff contends that its construction is consistent with the claim language that was amended to include "does not have to work on the same instructions at the same time." (*Id.* at 30) (citing '156 Patent at claims 6 and 11). Plaintiff argues that this amendment clarified the asynchronous nature of the sequencer and processing elements by specifically reciting that the sequencer "does not have to work on the same instruction at the

same time" as the processing elements. Plaintiff contends that Defendants' construction has no support in the specification, and fails to accord proper weight to the decoupling of the instruction processing that is the focus of the specification. (*Id.*)

Defendants respond that Plaintiff's proposal is inconsistent with the claim language because the claims already include the limitation of not having "to work on the same instruction at the same time." (Dkt. No. 151 at 31.) Defendants argue that the phrases "logically asynchronous to" and "does not have to work on the same instructions at the same time," are separated by "and," indicating that they are separate requirements. (*Id.*) Defendants also argue that the specification distinguishes prior art in which the sequencer and processing elements "run[] in lock step" and "operate in a fixed-phase relationship." (*Id.*) (citing '156 Patent at 3:15–22). Defendants contend that the claimed invention is contrasted by the "sequencer and PEs are not designed to run in lock step: instead the sequencer and PEs are decoupled . . . [a]s a result, the PE and sequencer can be asynchronous." (*Id.*) (citing '156 Patent at 3:60–4:1). According to Defendants, the specification equates "logically asynchronous" with not having to run in "lock step" or at a fixed delay. (*Id.*)

Defendants further argue that this construction is confirmed by the prosecution history. (*Id.*) Defendants contend that the prior art operated the sequencer and processing elements at a "fixed delay," and was distinguished precisely because this meant that these two units were not "logically asynchronous" to one another. (*Id.*) (citing Dkt 145-10 at 192–93 ('156 Patent FH, January 31, 2008 Office Action Response)). Defendants argue that Plaintiff's reliance on one sentence from the specification conflicts with the remainder of the intrinsic evidence and the inventor's testimony. (*Id.* at 32.) Defendants contend that the "i.e." provides a definition of "logically asynchronous," and that definition is the word that follows in the sentence—

"decoupled"—which is consistent with Defendants' construction. (*Id.*) Defendants argue that the remainder of the sentence goes on further to explain a mere result of this decoupling, which is that the two units do not have to work on the same instruction at the same time. (*Id.*)

Plaintiff replies that the '156 Patent twice provides an express definition of "logically asynchronous." (Dkt. No. 155 at 11) (citing '156 Pat. at 4:1–3; 21:66–22:1). Plaintiff contends that the first part of the definition treats "logically asynchronous" and "decoupled" as synonymous, and that the rest of the definition explains how the sequencer and PEs are "logically asynchronous" or "decoupled" (*i.e.*, "in that they do not have to work on the same instruction at the same time.") (*Id.*) Plaintiff further argues that the redundancy created in the claims by its proposed construction was required by the Examiner to clarify the meaning of "asynchronous." (*Id.* at 12) (citing Dkt. No. 145-11 at 280–87). Plaintiff also contends that its construction relates to the decoupling of instruction processing at the sequencer and processing elements, which allows the sequencer to run ahead of the processing elements to switch program threads. (*Id.*) (citing '156 Patent at 21:62–64; 22:14–23).

Plaintiff further argues that nothing in the specification prohibits "logically asynchronous" from including fixed delays. (*Id.*) Plaintiff argues that the prosecution history cited by Defendants does not support a disclaimer of a system that operates with a fixed delay, even if the system includes a decoupling buffer. (*Id.*) Finally, Plaintiff contends that the inventor also testified that the sequencer and the processing elements are "processing their own instructions at different rates, [so] they can . . . lose the synchronicity between them at the instruction level." (*Id.*) (citing Dkt. No. 155-2 at 8 (Baldwin Tr.)).

For the following reasons, the Court finds that the phrase **"is logically asynchronous to"** should be construed to mean **"has a rate of instruction processing decoupled from,"** the

phrase **"asynchronous operations thereof"** should be construed to mean **"operating with a decoupled rate of instruction processing,"** the phrase **"operating asynchronously to"** should be construed to mean **"operating with a decoupled rate of instruction processing,"** and the phrase **"decouples operations"** should be construed to mean **"allows a decoupled rate of instruction processing."**

**b) Analysis**

The phrase "is logically asynchronous to" appears in claims 1, 6, 11, and 16 of the '156 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. The phrase "asynchronous operations thereof" appears in claim 11 of the '156 Patent. The phrase "operating asynchronously to" appears in claim 6 of the '156 Patent. The phrase "decouples operations" appears in claims 1 and 6 of the '156 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. As an initial matter, the Court finds, and the parties agree, that the phrases should generally be construed the same.

Turning to the claim language, the Court finds that the claims provide a distinction between "physically synchronous" and "logically asynchronous." Claim 1 recites that the "sequencer is logically asynchronous to said processing elements," and that the "sequencer and said processing elements are physically synchronous." The specification clarifies this distinction:

> The present application describes a 3D graphics architecture in which a FIFO buffer is placed between the sequencer and the processing elements (PEs). The sequencer and PEs are not designed to run in lock step: *instead the sequencer and PEs are decoupled* to allow the PEs, which form the SIMD array, to run at 100% efficiency even when the sequencer is switching between threads and performing other flow control operations. Thus, *the rate of instruction processing in the PE is not coupled to the rate of instruction processing in the sequencer*. *As a result, the PE and sequencer can be asynchronous*. The PE and sequencer are logically asynchronous, i.e. decoupled, in that they do not have to work on the same instruction at the same time, but as an implementation convenience, they *run in*

*the same clock domain so technically they are physically synchronous*.

'156 Patent at 3:57–4:5 (emphasis added), *see also* Abstract. Here, the specification indicates that "physically synchronous" means that the sequencer and processing elements run in the same clock domain, while "logically asynchronous" means that the sequencer and processing elements have a decoupled rate of instruction processing. In other words, the sequencer and the processing elements run off the same clock, but are logically decoupled because that can have a different rate of instruction processing. Accordingly, the Court finds that a person of ordinary skill in the art would understand "logically asynchronous" to mean having a decoupled rate of instruction processing.

Regarding Plaintiff's construction, the Court finds that it is redundant and would conflate the two claim elements: "logically asynchronous to" and "does not have to work on the same instructions at the same time." Substituting Plaintiff's construction into claim 6, the claim would read "wherein said sequencer [*does not have to work on the same instruction at the same time*] to said processing elements and [*does not have to work on the same instructions at the same time*] as said processing elements." '156 Patent claim 6 (emphasis added). Contrary to Plaintiff's contention, this does not clarify the phrase "is logically asynchronous." In contrast, the Court's construction provides that the sequencer and processing elements have both a decoupled rate of instruction processing and do not have to work on the same instruction at the same time.

Regarding Defendants' construction, the Court disagrees that the applicant clearly and unambiguously disclaimed all systems that include a fixed delay, as Defendants' construction requires. Instead, the applicant distinguished the prior art because the calculated delay period did not have the effect of decoupling the operation of the sequencer and the processor elements. Specifically, the applicant argued that a "fixed delay does not decouple a sequencer and a processor, the operations of each of them are still logically tied together although one is some

preset time later/earlier than the other. The operations of them are synchronized in preset manner." (Dkt 145-10 at 193 ('156 Patent FH, January 31, 2008 Office Action Response)).

In other words, a fixed delay does not disclose a system where the sequencer and processing elements have a different rate of instruction processing. That said, to the extent that Plaintiff argues that a fixed delay by itself meets the "logically asynchronous" limitation, the Court rejects this argument. The claims require that the sequencer and processing elements have a decoupled rate of instruction processing so that they do not necessarily have to work on the same instructions at the same time. Accordingly, the Court does not adopt either parties' construction.

During the claim construction hearing, Plaintiff expressed concern that the Court's preliminary construction did not allow for the possibility that the sequencer and processing elements may operate at the same rate. Defendants represented to the Court that they were not taking the position that decoupling the sequencer from the processing elements prevented the two from possibly running at the same rate. The Court agrees and has modified its preliminary construction to indicate that "logically asynchronous" means having a decoupled rate of instruction processing, and not just a different rate of instruction. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"is logically asynchronous to"** to mean **"has a rate of instruction processing decoupled from,"** the phrase **"asynchronous operations thereof"** to mean **"operating with a decoupled rate of instruction processing,"** the phrase **"operating asynchronously to"** to mean **"operating with**

**a decoupled rate of instruction processing,"** and the phrase **"decouples operations"** to mean **"allows a decoupled rate of instruction processing."**

### 2. "automatically transferring processing" and "automatically transfers processing"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "automatically transferring processing" | "switching processing to a different thread without the sequencer receiving a yield instruction" | "without receiving an instruction transferring processing" |
| "automatically transfers processing" | "transfers processing without involvement of the host processor" | "without receiving an instruction transferring processing" |

#### a) The Parties' Positions

The parties dispute whether the term "automatically" indicates that processing transfers to a different program thread without the sequencer receiving an explicit yield instruction from the program, as Plaintiff proposes, or if the term "automatically" means that the processing elements transfer processing without any instruction being received, as Defendants propose. Plaintiff contends that the specification indicates that the sequencer "detects if the current thread is about to access a register with an outstanding load from memory (or an instruction or global cache miss) and will switch to running another thread that can do useful work." (Dkt. No. 144 at 31) (quoting '156 Patent at 3:45–49). Plaintiff argues that in the preferred embodiment, the sequencer determines if the processing elements should switch to processing a different thread, and if so, the sequencer provides the processing elements with instructions from the new thread. (*Id.*) (citing '156 Patent at 3:45–49, 22:11–23, Figure 1).

Plaintiff further argues that claim 11 specifies that the sequencer is the device that "automatically transfers processing." (*Id.*) Plaintiff also contends that the inventor submitted a

declaration during prosecution that stated that the claimed sequencer automatically switches threads when it detects that a stall condition is going to arise without involvement of the program. (*Id.* at 32) (citing Dkt. No. 145-10 at 312 ('156 Patent FH, Declaration of David Baldwin)). Plaintiff further argues that Defendants' construction is inconsistent with the prosecution history because it seeks to introduce a restriction that was removed from the claims after the examiner specifically objected to the terminology as unsupported. (*Id.*) (citing Dkt. No. 145-10 at 296, 321, and 350 ('156 Patent FH)).

Defendants respond that the prosecution history demonstrates that "automatically transferring processing" should be construed to require transfer without receipt of an instruction. (Dkt. No. 151 at 33.) Defendants contend the applicant replaced "without receiving an instruction" with "automatically" in claim 6 in response to a written description rejection. (*Id.*) (citing Dkt. No. 151-42 at 5, 11 ('156 Patent FH, April 2, 2009 Office Action Response)). Defendants argue that this illustrates that the applicant equated "automatically" with "without receiving an instruction."

Defendants further argue that Plaintiff's construction should be rejected because it defines the term by introducing technical limitations of "thread" and "yield instruction" that would confuse the jury. (*Id.*) Defendants also argue that Plaintiff's construction limits "transferring processing" to "switching processing to a different thread," even though the patent, inventor's declaration, and deposition testimony support other embodiments such as "jump" instructions and "subroutines." (*Id.*) (citing '156 Patent at claim 1, 22:37); (citing Dkt. No. 151-43 at 2–3 ('156 Patent FH, Declaration of David Baldwin)); (citing Dkt. No. 151-9, Baldwin Tr. 201:9–20)). Finally, Defendants argue that while the applicant may have distinguished "yield instructions" in prior art, they never limited "automatically" to exclude receipt of only "yield

instructions." (*Id.*) (citing Dkt. No. 151-44 at 14-15 ('156 Patent FH, August 8, 2008 Office Action Response)); (citing Dkt. No. 151-42 at 16 ('156 Patent FH, April 2, 2009 Office Action Response)).

Plaintiff replies that neither the specification nor the claims ever suggest autonomously transferring processing to another thread without any instructions or commands. (Dkt. No. 155 at 13.) Plaintiff contends that processing is transferred to another thread based on the instructions that the sequencer loads into the instruction buffer. (*Id.*) (citing '156 Patent at 22:14–18, FIG. 1). Plaintiff argues that the only relevant discussion in the intrinsic evidence refers to switching threads without including yield instructions in the thread's program. (*Id.*) (citing Dkt. No. 145-10 at 308, 312 ('156 Patent FH)). Plaintiff contends that Defendants' construction reverts to the claim language that was rejected by the PTO, and replaced with "automatically." (*Id.*) (citing Dkt. No. 145-10 at 321, 350–56 ('156 Patent FH)).

For the following reasons, the Court finds that the phrase **"automatically transferring processing"** should be construed to mean **"transferring processing based on the sequencer detecting or predicting a stall,"** and that the phrase **"automatically transfers processing"** should be construed to mean **"transfers processing based on the sequencer detecting or predicting a stall."**

**b) Analysis**

The phrase "automatically transferring processing" appears in claim 6 of the '156 Patent. The phrase "automatically transfers processing" appears in claim 11 of the '156 Patent. Claims 6 and 11 recite "at least one sequencer," "a plurality of SIMD processing elements," and a "plurality of data items." Claim 6 further recites that the "plurality of SIMD processing elements . . . [are] configured to receive processing instructions from said sequencer . . . , the

processing elements automatically transferring processing to another plurality of data items when processing on said first plurality of data items stalls." Similarly, claim 11 recites that the "sequencer . . . generates fragment-processing commands therefrom, and automatically transfers processing to another plurality of 3D graphics data items when processing on said first plurality of 3D graphics data items stalls or is predicted to stall." Thus, the Court finds that the claim language indicates that "automatically transfers processing" relates to transferring processing based on the sequencer detecting or predicting a stall.

Consistent with the claim language, the specification indicates that it is the recited sequencer that determines if the processing elements should switch to processing another plurality of data items. Specifically, the specification states "[t]he sequencer detects if the current thread is about to access a register with an outstanding load from memory (or an instruction or global cache miss) and will switch to running another thread that can do useful work." '156 Patent at 3:45–49. The specification adds that the "FIFO instruction buffer 120 holds the instructions generated by the sequencer that are processed by the processing elements." '156 Patent at 22:11–23.

A declaration submitted by the inventor during prosecution is consistent with the claims and specification, and further confirms that "automatically transferring processing" means transferring processing based on the sequencer detecting or predicting a stall. The declaration states that the "sequencer is responsible for controlling the flow of a program through sequential instruction and jumps and subroutines." (Dkt. No. 145-10 at 312 ('156 Patent FH, Declaration of David Baldwin)). The declaration further states that one embodiment of the invention automatically switches threads "when it detects that a stall condition is going to rise." (*Id.* at 313.) The declaration explains that "[t]his testing, scheduling and switching are done in the

sequencer and the cost of this is hidden by running the sequencer ahead of the computation units with a[] FIFO buffer to provide some elasticity." (*Id.*)

Accordingly, with regard to claim 6, the Court finds that "automatically transferring processing" means "transferring processing based on the sequencer detecting or predicting a stall." This is done "automatically" by allowing the sequencer to "run ahead of fragment processor" to detect or predict a stall. '156 Patent at 22:18–19. Similarly, with regard to claim 11, the Court finds that the disputed "automatically transfers processing" means "transfers processing based on the sequencer detecting or predicting a stall."

Regarding Defendants' construction, the Court finds that it is not consistent with the prosecution history. During prosecution, the applicant amended the independent claims to include Defendants' proposed "without receiving an instruction" construction. (Dkt 145-10 at 296 ('156 Patent FH)). In response, the examiner rejected the amended claims, finding that there was no section 112 support for the "without receiving an instruction" limitation. (*Id.* at 321). The examiner further noted that there was support for "automatically switching threads" in paragraph [0019] of the specification, and that "automatically" did not have the same meaning as "without receiving an instruction."[8] (*Id.*) In fact, the examiner stated that the two phrases differed because "automatically" does not "limit the exclusion of providing instructions (e.g., the next instruction may [be] automatically delivered to the system . . . ." (*Id.*) The applicant subsequently deleted the phrase "without receiving an instruction" and inserted the word "automatically." (*Id.* at 352.)

Defendants argue that this required changing the amended "automatically" back to "without receiving an instruction," because when it was made the applicant stated that "[n]o new matter is added" and "[n]o substantive amendment" was made. (Dkt. No. 151 at 33) (citing Dkt.

[8] Paragraph 19 of the application appears in the issued '156 Patent at 3:37–56.

No. 151-42 at 11 (‘156 Patent FH, April 2, 2009 Office Action Response)). Defendants further argue that the Court should give the examiner’s statements less weight, because this amendment shows that the applicant equated “automatically” with “without receiving an instruction.” (Dkt. No. 151 at 33 n.15).

The Court disagrees with Defendants’ characterization of the prosecution history. As discussed above, the examiner explicitly stated that there was no section 112 support for the “without receiving an instruction” limitation, and found that there was support for “automatically switching threads” in the specification. The section of the specification referenced by the examiner explicitly states that “[t]he sequencer detects if the current thread is about to access a register with an outstanding load from memory (or an instruction or global cache miss) and will switch to running another thread that can do useful work.” ‘156 Patent at 3:45–49. This is consistent with the examiner’s additional statement that this interpretation of “automatically” is what “allows the system to function.” (Dkt 145-10 at 321.) A person of ordinary skill in the art would not interpret this as Defendants propose. Furthermore, the Court is not persuaded that it should be given little weight, or effectively ignored, as Defendants contend.

Regarding Plaintiff’s construction, the Court agrees that it would be confusing to a jury because it introduces the term “yield instruction,” a term not found in the claims or the specification. Moreover, the Court agrees with Defendants that the applicant did not limit “automatically” to exclude receipt of only “yield instruction,” as Plaintiff proposes. Finally, claims 6 and 11 do not recite the word “thread,” and construing the disputed phrase to require “switching processing to a different thread” would be confusing in the context of the surrounding claim language. Accordingly, the Court does not adopt either parties’ construction. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its

proper weight in light of the intrinsic evidence.

#### c) Court's Construction

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"automatically transferring processing"** to mean **"transferring processing based on the sequencer detecting or predicting a stall,"** and the phrase **"automatically transfers processing"** to mean **"transfers processing based on the sequencer detecting or predicting a stall."**

### 3. "sequencer"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "sequencer" | "functional unit that decodes commands and performs flow control operations" | "unit that controls instruction flow and calculates the address of and fetches the next instruction" |

#### a) The Parties' Positions

The parties agree that the recited "sequencer" performs flow control operations or controls instruction flow. The parties dispute whether the sequencer is also required to "calculate the address of and fetch the next instruction," as Defendants propose, and whether the sequencer also "decodes commands," as Plaintiff proposes. Plaintiff contends that the specification repeatedly identifies the primary functions performed by the sequencer as "switching between threads and performing other flow control operations." (Dkt. No. 144 at 33) (citing '156 Patent at 3:63–65, 21:61–62, Abstract, 3:45–49). Plaintiff also contends that the prosecution history focuses on flow control operations as the primary function of the sequencer. (*Id.*) (citing Dkt. No. 145-10 at 306, 311–12 ('156 Patent FH)). Plaintiff further argues that the asserted claims require the sequencer to decode rendering commands, and thus includes the decoding requirement for completeness. (*Id.*)

Regarding Defendants' construction, Plaintiff argues that the only suggestion that a sequencer performs the functions of calculating the address of and fetching the next instruction is in the specification's discussion of conventional SIMD processors. (*Id.*) ('156 Patent at 3:15–18). Plaintiff contends that the subsequent discussion, however, notes that the sequencer operations used in these designs may cause stalls and prevent the ALU from running efficiently. (*Id.* at 33–34) (citing '156 Patent at 3:23–25). Plaintiff argues that the specification then discusses the configuration of the fragment processor of Figure 1, and outlines the operation of the sequence. (*Id.* at 34) (citing '156 Patent at 3:34–4:5). Plaintiff further argues that the specification states that the sequencer switches between program threads and performs other flow control operations. (*Id.*) (citing '156 Patent at 3:34–4:5). According to Plaintiff, nothing in the claims or specification requires the claimed sequencer to perform address calculations and fetch instructions. (*Id.*)

Defendants respond that Plaintiff's construction adds a redundant limitation about "decoding commands" and fails to capture basic functionality of the sequencer. (Dkt. No. 151 at 34.) Defendants argue the specification states that "[t]he sequencer is responsible for calculating the address of the next instruction and fetching it." (*Id.*) (citing '156 Patent at 3:17–18). Defendants also argue that the specification also explains that the sequencer is responsible for "switching between threads and other flow control operations." (*Id.*) (citing '156 Patent at 3:63–65). According to Defendants, their construction simply combines these functions enumerated in the specification into a straightforward construction. (*Id.*)

Responding to Plaintiff's argument that portions of the specification are not applicable because they describe a "conventional processor," Defendants argue that the inventor confirmed that the claimed invention does not have "anything to do with changing the specific operation of

the sequencer," and that the claimed invention's "sequencer" is also "responsible for calculating the address of the next instruction and fetching it." (*Id.*) (citing Dkt. No. 151-9, Baldwin Tr. 198:12–199:3, 211:24–212:4). Defendants further argue that during prosecution, the applicants explained that "a sequencer . . . functions for instruction flow control." (*Id.*) (citing Dkt. No. 151-44 at 13 ('156 Patent FH, August 8, 2008 Office Action Response)). Defendants also argue that the inventor's declaration explains that "a sequencer is responsible for controlling the flow of a program through sequential instruction" and that "instruction scheduling . . . [is] done in the sequencer." (*Id.*) (citing Dkt. No. 151-43 at 2, 4 ('156 Patent FH, Declaration of David Baldwin)).

Plaintiff replies that the intrinsic evidence does not require the sequencer to calculate the address of and fetch the next instruction. (Dkt. No. 155 at 13.) Plaintiff argues that the specification indicates the primary function of the sequencer is to perform flow control operations. (*Id.*) (citing '156 Patent at 3:63–65, 21:61–62, Abstract, 3:45–49). According to Plaintiff, there is no basis in the record to limit the construction as Defendants propose. (*Id.*)

For the following reasons, the Court finds that the term **"sequencer"** should be construed to mean **"unit that performs flow control operations."**

**b) Analysis**

The term "sequencer" appears in claims 1, 6, 11, and 16 of the '156 Patent. The Court finds that the term is used consistently in the claims and is intended to have the same meaning in each claim. The Court further finds that the intrinsic record indicates that the sequencer is a unit that performs flow control operations. The specification states that the sequencer "switch[es] between threads and perform[s] other flow control operations." '156 Patent at 3:63–65. Likewise, during prosecution, the applicant argued that the sequencer "functions for instruction

flow control." (Dkt. No. 145-10 at 306 ('156 Patent FH, Declaration of David Baldwin)). The inventor also stated that "a sequencer is responsible for controlling the flow of a program through sequential instruction." (*Id.* at 311–12.) Regarding this aspect, the Court notes that both parties cite to the same intrinsic evidence, thus it does not appear that the parties dispute that the sequencer is a unit that performs flow control operations.

Regarding Defendants' construction, there is only one line in the intrinsic record that states that a sequencer calculates the address of and fetches the next instruction. Specifically, the specification states that "[a] conventional SIMD processor or microcode CPU is designed with the sequencer and arithmetic logic unit (ALU) running in lock step. The sequencer is responsible for calculating the address of the next instruction and fetching it." '156 Patent at 3:15–18. The Court finds that this statement is made in the context of a "conventional SIMD processor." The specification then states that in "the present application . . . [t]he sequencer and PEs are not designed to run in lock step," and that the sequencer switches between threads and performs other flow control operations. '156 Patent at 3:58–65. The Court finds that the distinction between a "conventional SIMD processor" and "the present application" indicates that Defendants' language of calculating and fetching the address of the next instruction is unwarranted. Accordingly, the Court is not convinced that a person of ordinary skill would understand that the claims were intended to be limited to a "conventional SIMD processor," as Defendants propose.

Regarding Plaintiff's construction, claims 1, 6, and 11 recite that the sequencer decodes rendering commands. Thus, the Court agrees with Defendants that this language is unnecessary. Accordingly, the Court does not adopt either parties' construction. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light

of the intrinsic evidence.

#### c) Court's Construction

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"sequencer"** to mean **"unit that performs flow control operations."**

### C. The '383 Patent

The parties' dispute focuses on the meaning and scope of three terms/phrases in the '383 Patent.

#### 1. "graphics processor"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "graphics processor" | "a specialized processor that accelerates creation of images intended for output to a display" | Plain and ordinary meaning<br>Alternatively, "a processor that creates images" |

#### a) The Parties' Positions

The parties dispute whether the term "graphics processor," which appears in the preamble of claim 1, is a limitation that must be construed. Plaintiff contends that the term is limiting and must be construed to distinguish the specialized graphics processor described in the '383 Patent, from a general purpose processor such as a CPU. (Dkt. No. 144 at 26.) Plaintiff argues that the specification indicates that the graphics processor context is an important aspect of the invention. (*Id.*) (citing '383 Patent at Abstract, 1:7–8, 2:14–17). Plaintiff also argues that during prosecution, the applicant summarized the invention as relating to "complex integrated circuits, particularly parallelized graphics accelerators." (*Id.* at 27) (Dkt. No. 145-9 at 232 ('383 Patent FH)).

Plaintiff further argues that claim 1 specifically recites "a graphics processor" that includes "a plurality of parallelized graphics computational units." (*Id.*) (citing '383 Patent at

claim 1). Plaintiff contends that the preamble of claim 1 provides the necessary context for the claim, making it evident that the claim is directed to a graphics processor and not to a generic processing element. (*Id.*) Plaintiff also argues that its construction is consistent with the IEEE's definition of "graphics processor": "a hardware device that executes a sequence of display commands to create a display image." (*Id.* at 28) (quoting Dkt 145-12 at 4 (IEEE Standard Dictionary of Electrical and Electronic Terms, Sixth Ed. 1996)).

Finally, Plaintiff argues that Defendants' construction could be interpreted to cover a vast array of processors that people of ordinary skill would not consider "graphics" processors, but rather just "processors." (*Id.*) Plaintiff argues that any general purpose processor could be considered a processor that creates images. (*Id.*) According to Plaintiff, Defendants' construction would eliminate the necessary context of the claim, which is a graphics processor that includes graphics computational units. (*Id.*)

Defendants respond that "graphics processor" is a non-limiting preamble term that needs no construction. (Dkt. No. 151 at 29.) Defendants argue that the term "graphics processor" does not provide antecedent basis for any claim limitations, does not itself connote any structure, and was not used to distinguish prior art during prosecution. (*Id.*) Defendants contend that Plaintiff's cite to the file history only shows the applicants used "graphics" to explain the context of the invention to the PTO, not to distinguish prior art. (*Id.* at 30.) Defendants further argue that the specification explains that the claimed invention "can be applied not only to complex chips in the field of graphics, but to any market where large parts of the design are replicated to give scalable performance." (*Id.*) (citing '383 Patent at 2:36–38).

Defendants further argue that if "graphics processor" is limiting, it is a straightforward term that requires no construction or can be defined simply as "a processor that creates images."

(*Id.*) Defendants contend that their alternative construction is consistent with the IEEE definition cited by Plaintiff. (*Id.*) Finally, Defendants argue that Plaintiff's proposal redefines "graphics processor" with complex, ambiguous limitations of "specialized," "accelerates," and "intended for output" that are not supported by any evidence. (*Id.*)

Plaintiff replies that the preamble term "graphics processor" is limiting because it recites structure emphasized as important in the specification. (Dkt. No. 155 at 11.) Plaintiff argues that the portion of the specification cited by Defendants supports its construction because although the "idea" of the invention can be applied to markets beyond graphics processors, claim 1 is specifically limited to "graphics" processors. (*Id.*) According to Plaintiff, if the claims were intended to cover all general purpose processors that "create[d] images," the term "graphics" would be superfluous. (*Id.*)

For the following reasons, the Court finds that the phrase **"graphics processor"** should be construed to mean **"a processor that executes commands to create a display image."**

**b) Analysis**

The term "graphics processor" appears in claims 1–4 and 6–9 of the '383 Patent. The Court finds that the term is used consistently in the claims and is intended to have the same meaning in each claim. The Court further finds that the intrinsic evidence indicates that the preamble is limiting. Claim 1 recites "a *graphics* processor" that includes "a plurality of parallelized *graphics* computational units." '383 Patent at Claim 1 (emphasis added). Thus, the preamble of claim 1 is "necessary to give life, meaning, and vitality" to the claim because it provides context for the claim. *Pitney Bowes, Inc. v. Hewlett-Packard Co.*, 182 F.3d 1298, 1305 (Fed. Cir. 1999). Specifically, it indicates to a person of ordinary skill in the art that the claim is directed to a graphics processor, and not to a generic processor.

Likewise, the specification indicates that the claims are focused on graphic processors. For example, the specification states that "[t]he present invention relates to complex integrated circuits, particularly parallelized graphics accelerators." '383 Patent at 1:7–8. In addition, the specifications states that the "idea" of the present invention "can be applied not only to complex chips in the field of graphics, but to any market where large parts of the design are replicated to give scalable performance." '383 Patent at 2:36–38. This indicates that claim 1 applies the idea to a "graphics processor," as recited in the preamble, and is not so broad as to apply to "any market." Accordingly, the Court finds that the preamble is limiting and that the claim is directed to a graphics processor, and not to a generic processor.

Turning now to the construction of "graphics processor," the Court notes that both parties cite to the IEEE definition, which defines a "display processor" as "a hardware device that executes a sequence of display commands to create a display image. Synonym: graphics processor." (Dkt 145-12 at 4 (IEEE Standard Dictionary of Electrical and Electronic Terms, Sixth Ed. 1996)). Defendants contend that their construction is consistent with the "create a display image" language of this definition. The Court agrees, but finds that Defendants' construction leaves out the portion of executing commands. Furthermore, the parties agree that the term should be construed as a "processor." Accordingly, the Court finds that the term "graphics processor" should be construed to mean "a processor that executes commands to create a display image."

Regarding Plaintiff's construction, the Court does not adopt it because it uses ambiguous language that is not found in the specification. For example, the word "specialized" does not appear in the intrinsic evidence, and it is not clear what determines when a processor is "specialized." Accordingly, the Court finds that Plaintiff's construction includes language that

could be confusing to a jury.

#### c) Court's Construction

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"graphics processor"** to mean **"a processor that executes commands to create a display image."**

### 2. "graphics computational units"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "graphics computational units" | "computational units that form part of a graphics processor" | "computational subunits within a single graphics processor" |

#### a) The Parties' Positions

The parties agree that the graphics processor is made up of computational units. The parties dispute two issues: (1) whether the claims require a single graphics processor, as Defendants propose, and (2) whether the single graphics processor is made up of multiple computational "subunits," as Defendants propose. Defendants contend that the claim language, specification, and figures describe a single graphics processor with multiple computational units. (Dkt. No. 153 at 12) (citing '383 Patent at 2:25–35, 2:55–58, 3:31–41, and FIG. 1). Defendants also argue that during prosecution, the applicants explained that the invention relates to a single processor as opposed to multiple processors. (*Id.*) (citing Dkt. No. 153-9 at 14–16, 24 ('383 Patent FH, March 4, 2005 Appeal Brief)). According to Defendants, the intrinsic evidence unambiguously dictates a construction requiring multiple computational subunits within a single graphics processor. (*Id.* at 13.)

Regarding their "subunit" language, Defendants argue that they include "subunits" in their construction to clarify that each "computational unit" cannot itself be a standalone unit like a processor, but instead only exists as a subunit within a single graphics processor. (*Id.*)

Defendants contend that the claim language and specification describe subunits (e.g., texture pipes, vertex processors, etc.) as components within a single graphics processor. (*Id.*) (citing '383 Patent at claim 1, Figure 1, 2:25–35, 2:55–58, and 3:31–41). Defendants also argue that during prosecution, the applicants argued that "processor" and "unit" are "not interchangeable." (*Id.*) (citing Dkt 153-10 at 13 ('383 Patent FH, October 5, 2004 Office Action Response)). Defendants further note that claim 1 originally recited "subunits" rather than "computational units," and that even after making this amendment, the applicants still referred to the computational units as "subunits." (*Id.*) (citing Dkt. No. 153-11 at 50 ('383 Patent FH, Original Application)); (citing Dkt. No. 153-12 at 13 ('383 Patent FH, May 4, 2004 Office Action Response)); (citing Dkt. No. 153-9 at 23 ('383 Patent FH, March 4, 2005 Appeal Brief)). According to Defendants, the term "subunit" appropriately clarifies the distinction between the single graphics processor and the computational units within it. (*Id.*)

Plaintiff responds that there is no support for replacing the word "units" with the word "subunits." (Dkt. No. 158 at 10.) Plaintiff notes that during prosecution, the applicant amended claim 1 to replace "subunits" with the word "units." (*Id.*) (Dkt. No. 145-9 at 173 ('383 Patent FH)). Plaintiff argues that Defendants' requested alteration would undo an amendment made during prosecution. (*Id.*) Plaintiff further argues that the applicant described the change from "subunits" to "units" as "purely formal," and noted that the amendment was "believed not to change the scope of the[] claims." (*Id.*) (Dkt. No. 145-9 at 162 ('383 Patent FH)). Plaintiff contends that there is no reason for the Court to undo a claim amendment entered by the PTO, especially when the applicant did not believe the amendment changed the scope of the claims. (*Id.*)

Regarding Defendants' "within a single graphics processor" language, Plaintiff argues

that claim 1 teaches that "a graphics processor" comprises "a plurality of parallelized graphics computational units." (*Id.* at 11.) Plaintiff contends that there is no restriction on the number of graphics processors, and that Defendants' construction violates the "repeatedly emphasized" rule that "a" in a "comprising" claim means "one or more." (*Id.*) (quoting *Baldwin Graphic Sys., Inc. v. Siebert, Inc.*, 512 F.3d 1338, 1342 (Fed. Cir. 2008)).

Plaintiff further argues that Defendants' citation to the prosecution history has nothing to do with the number of graphics processors allowed in a system. (*Id.*) Plaintiff contends that the prior art ("Brent") taught that if a texture pipeline within a particular processor was defective, the entire processor was disabled, and tasks were allocated to other fully operational processors. (*Id.*) (citing Dkt. No. 145-9 at 240 ('383 Patent FH)). According to Plaintiff, Brent was an entirely different invention than the '383 Patent because the two systems operate at different levels of granularity. (*Id.*) Plaintiff contends that Brent operates at the processor level, and if a single pipeline within a processor is defective, the entire processor is disabled. (*Id.*) Plaintiff further argues that the '383 Patent operates at the pipeline level, and if a pipeline is defective, the '383 Patent teaches that only the defective pipeline is bypassed, rather than disabling the entire processor. (*Id.*) (citing Dkt. No. 145-9 at 241 ('383 Patent FH)). Plaintiff argues that every processor in the system can remain operational, even if certain pipelines within the processors are defective. (*Id.*) (citing Dkt. No. 145-9 at 240 ('383 Patent FH)).

Plaintiff further argues that Defendants improperly argue that all computational units must be located within a single graphics processor, which implies there can only be one graphics processor in the system. (*Id.*) Plaintiff contends that this is not how the applicant characterized the invention, and there is nothing in the intrinsic record restricting the invention to a one-processor system. (*Id.*) According to Plaintiff, the '383 Patent does not require a plurality of

graphics processors to function, but it can still function with a plurality of graphics processors. (*Id.*)

For the following reasons, the Court finds that the term **"graphics computational units"** should be construed to mean **"computational units within a single graphics processor."**

**b) Analysis**

The term "graphics computational units" appears in claims 1–7, 10–15, and 18–23 of the '383 Patent. The Court finds that the term is used consistently in the claims and is intended to have the same meaning in each claim. The claim language further recites that "graphical computational units" are a subunit, or contained within the "graphical processor." For example, claim 1 recites "a graphic processor" that includes "a plurality of parallelized graphical computational units." The claim language further indicates that the "graphical computational units" may include "multiple vertex processors," "texturing pipelines," and "memory controllers." '383 Patent at Claims 2, 3, and 4. Thus, the claim language indicates that a "graphical computational unit" is a unit within a graphics processor.

This is consistent with the specification and prosecution history. For example, the specification states:

> FIG. 1 shows an example of the physical layout of a graphics processing integrated circuit wherein a subset of working parts can define a secondary lower performance part. In this example, if all components are functional, the chip will operate with: 16 vertex processors, 4 texture pipes, 2 memory controllers, and 2 RAMDACs. (The operation of these components is described below.)
>
> However, if some of these components are defective, the same chip can be specified under a secondary part specification, with e.g. as little as 4 vertex processors, 1 texture pipe, 1 memory controller, and 1 RAMDAC.

'383 Patent at 3:31–41. In this passage, the specification states that the vertex processors, texture pipes, and memory controllers are within "the physical layout of a graphics processing integrated circuit." Likewise, during prosecution, the applicant stated that "[t]he claimed subject matter of

independent claim 1 can be summarized as a graphics processor (described repeatedly throughout the specification . . . ) with more than one graphics computational units, such as vertex processors, texture pipes, memory controllers, and RAMDACs." (Dkt. No. 145-9 at 303 ('383 Patent FH, November 21, 2005 Appeal Brief)). This further indicates that that a "graphical computational unit" is a unit within a graphics processor.

Moreover, the applicant clearly argued that "graphical computational units" must be within a single graphics processor. In distinguishing the prior art, the applicant argued that "[Brent] teaches away from the present inventions as it not only operates on a completely different scale, but also requires a plurality of processors in order to function. The Brent patent requires a multiple of separate, fully functional processors in order to function." Dkt. No. 153-9 at 16 ('383 Patent FH, March 4, 2005 Appeal Brief)). Focusing on the different scale aspect, the applicant argued that "[t]he entire present application would be placed within any one of the CP0, CP1, etc. elements [processor] within the Figure 2 of the Brent Application." (*Id.* at 17.) The applicant concluded that "[t]his application deals with the ability to bypass a defective pipeline within a SINGLE processor ('A GRAPHICS PROCESSOR') as opposed to the MULTIPLE ('*PLURAL QUEUE PROCESSORS*') processors alluded to in the prior art." (*Id.* at 24) (emphasis in original). Thus, the applicant clearly limited the claims "to bypass[ing] a defective pipeline within a SINGLE processor," thereby requiring the recited "graphical computational units" to be within a single graphics processor. (*Id*.) (emphasis in original).

That said, the Court's construction does not exclude from the scope of the claims systems that include multiple graphics processors. Instead, it only finds that the claims require the recited bypassing to occur at the computational units level or pipeline level, not the processor level. Moreover, the intrinsic evidence requires the recited bypassing to occur within each processor,

respectively. Indeed, the applicant made this clear by arguing:

> "[t]he Brent solution to a failed pipeline is to disable the entire processor, whereas in the present invention the solution is to disable only the pipeline leaving the remainder of the processor to function. The Brent solution is like amputating limb in the case of broken artery in order to save the body wherein the present application can close the artery and save the limb."

(*Id.* at 17.) Accordingly, to the extent that Defendants argue that multiple graphics processors systems were disclaimed or disavowed during prosecution, the Court rejects that argument.

Regarding Defendants' "subunit" language, the Court is not persuaded that "unit" needs to be redrafted as "subunit." The applicant amended Claim 1 to replace "subunits" with the word "units." (Dkt. No. 145-9 at 173 ('383 Patent)). The applicant described the change from "subunits" to "units" as "purely formal" and noted that the amendment was "believed not to change the scope of the[] claims." (*Id.* at 162.) Defendants contend that the Court should undo this amendment and redraft the claim to clarify that each "computational unit" cannot itself be a standalone unit like a processor. The Court disagrees that redrafting this language is necessary. However, as discussed above, the Court finds that the recited "graphical computational unit" is a unit within a graphics processor, and is not a standalone unit like a processor. The applicant argued that the terms "processors" and "units" are "indeed not interchangeable and that a graphics computational unit is not the same thing as an I/O processor." (Dkt 153-10 at 13 ('383 Patent FH, October 5, 2004 Office Action Response)).

Accordingly, although the Court does not adopt Defendants' "subunit" language, the Court finds that the recited "graphical computational unit" is a unit within a graphics processor, and is not a standalone unit like a processor. Therefore, the Court rejects Plaintiff's argument to the extent it contends that a computational unit discloses a "graphics processor." Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

### c) Court's Construction

In light of the intrinsic and extrinsic evidence, the Court construes the term **"graphics computational units"** to mean **"computational units within a single graphics processor."**

## 3. "task allocation units programmed to bypass defective ones"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "task allocation units programmed to bypass defective ones of said units" | "functional units programmed to distribute tasks only to operative graphics computational units" | "units running software to recognize and avoid allocating tasks to defective graphics computational units" |

### a) The Parties' Positions

The parties agree that the disputed phrase refers to "units" that allocate or distribute tasks to certain computational units. The parties dispute two issues: (1) whether the units must run software, as Defendants propose, and (2) whether the units are programmed to distribute tasks only to operative graphics units, as Plaintiff proposes. Regarding the software issue, Defendants argue that the specification explains that "programmed to bypass" means running software that recognizes defective units and avoids allocating tasks to those units. (Dkt. No. 153 at 14) (citing '383 Patent at 3:48–50, 14:33–40).

Regarding the operative graphics units issue, Defendants argue that Plaintiff changes the claim language from reciting which units are bypassed or avoided (i.e., the "defective" units) to reciting which units are affirmatively used (*i.e.*, "operative" units). (*Id.* at 15.) Defendants argue that this is incorrect because units may be inoperative for many reasons—such as being intentionally turned off—even if they have no manufacturing defects and thus are not

"defective." (*Id.*) Defendants argue that Plaintiff's construction is contrary to the specification that repeatedly describes the claimed invention using the specific terms of avoiding or overcoming "defective" units or units with manufacturing "defects" or faults. (*Id.*) (citing '383 Patent at 2:42–44, 14:34–40, and 2:33). Defendants also contend that Plaintiff improperly construes the disputed term to repeat language already present in the very next limitation in the claim, which recites "distribute incoming tasks only among operative ones of said units." (*Id.*)

Plaintiff responds that by requiring the task allocation units to "run[] software," Defendants are limiting the claim to a preferred embodiment. (Dkt. No. 158 at 12.) Plaintiff argues that the specification discloses several different techniques for bypassing defective components that do not involve "running software," including "blowing fuses," "selectively laser heating to cause diode punch through," and using "an off-chip configuration memory which the chip reads when coming out of reset." (*Id.*) (quoting '383 Patent at 3:42–47). Plaintiff also argues that the specification refers to running software as an embodiment, not a requirement. (*Id.* at 12-13.)

Plaintiff further argues that Defendants' construction adds a requirement that task allocation units contemporaneously recognize defective components. (*Id.* at 13.) Plaintiff contends that the specification teaches multiple embodiments where a defect is recognized prior to operation of the GPU. (*Id.*) ('383 Patent at 3:42–47). Plaintiff also contends that the applicant stated during prosecution that, in the "current invention," a "diagnostic can be [sic] preformed to detect the error, and the failure can be prevented prior to the processor even leaving the factory." (*Id.*) (quoting Dkt. No. 145-9 at 312 ('383 Patent FH)). Finally, Plaintiff argues that its construction more accurately describes what it means to "bypass" defective computational units. (*Id.*) (citing Dkt. No. 145-9 at 303 ('383 Patent FH)).

For the following reasons, the Court finds that the phrase **"task allocation units programmed to bypass defective ones of said units"** should be construed to mean **"units programmed to avoid allocating tasks to defective graphics computational units."**

**b) Analysis**

The phrase "task allocation units programmed to bypass defective ones of said units" appears in claim 1 of the '383 Patent. The claim language indicates that "one of said units" refers to the "graphics computational units." The parties agree on this point, and the Court finds that it would be helpful to clarify this language for the jury. The claim language also explicitly states that the "graphics computational units" that are bypassed are "defective ones." The Court is not persuaded that Plaintiff's redrafting of this language is necessary or warranted. Therefore, the Court rejects Plaintiff's argument to the extent it contends that the scope of the claim includes units that are otherwise operational, but are intentionally turned off.

The specification emphasizes that an advantage of the invention is the "ability to use partially defective die as fully functioning parts with lower performance" '383 Patent at 2:42–44, *see also* 3:23–25 ("This allows us to take die which would otherwise be classified as scrap and use them in a lower performance product."). In addition, Plaintiff's construction repeats language recited in other limitations in the claim. Specifically, claim 1 recites distributing "incoming tasks only among operative ones of said units." The Court finds that there is nothing confusing or ambiguous about "defective unit." Accordingly, the Court rejects this part of Plaintiff's construction.

Turning to Defendants' language of "running software to recognize and avoid allocating," the Court finds that this would improperly limit the task allocation units to contemporaneously recognized defective components. This is not required by the specification,

and is only included as a preferred embodiment. Plaintiff is correct that the specification teaches multiple embodiments where a defect is recognized prior to operation of the GPU. For example, the specification states that "[t]he results of testing at manufacture can be recorded in any of a variety of known ways, e.g. by blowing fuses or by selective laser heating to cause diode punch through. Alternatively and less preferably, it is also possible to use an off-chip configuration memory which the chip reads when coming out of reset (or power-up)." '383 Patent at 3:42–47.

Likewise, during prosecution the applicant stated that in the "current invention," a "diagnostic can be [sic] preformed to detect the error, and the failure can be prevented prior to the processor even leaving the factory." (Dkt. No. 145-9 at 312 ('383 Patent FH)). Moreover, as the embodiments above illustrate, the claims are not limited to only embodiments where the unit "runs software," as Defendants propose. Instead, consistent with the intrinsic evidence, the Court finds that the units are programmed to avoid allocating tasks to defective units. For example, the specification states that the "Texture Switch Unit can avoid using texture pipes with manufacturing defects." '383 Patent at 14:34–35. Similarly, the specification states that "by internally reconfiguring the chip we can still make use of a die with one or more failing texture pipes . . . ." *Id.* at 3:21–23. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"task allocation units programmed to bypass defective ones of said units"** to mean **"units programmed to avoid allocating tasks to defective graphics computational units."**

**D. The '637 Patent**

The parties' dispute focuses on the meaning and scope of four terms/phrases in the '637

Patent.

### 1. "means for storing the associated image in the frame buffer"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "means for storing the associated image in the frame buffer" | **Governed by § 112 (6)**<br>**Function:** storing the associated image in the frame buffer.<br>**Structure:** Bus and one or more resolvers, or equivalents thereof. | **Governed by § 112 (6)**<br>**Function:** storing the associated image in the frame buffer<br>**Structure:** IZ bus and ASICs that implement the IZ bus protocol |

#### a) The Parties' Positions

The parties agree that the "means for storing the associated image in the frame buffer" should be construed according to 35 U.S.C. § 112 ¶ 6, and that the function is "storing the associated image in the frame buffer." The parties dispute whether the structure that performs the function is IZ bus interface connected to an IZ bus, as Defendants propose, or a bus and resolvers, as Plaintiff proposes. Plaintiff contends that Figure 1 in the '637 Patent depicts the preferred embodiment of the invention, in which the graphics engine communicates with the frame buffer using a high speed bus and one or more resolvers. (Dkt. No. 144 at 7.) Plaintiff argues that Figure 4 illustrates the video data stream feeds through the graphics engine, over the high speed bus (referred to in the preferred embodiment as the "Image/Z" or "IZ" bus). (*Id.*) (citing '637 Patent at 3:8–10).

Plaintiff contends that in the preferred embodiment, "[t]he IZ bus 27 is a 64 bit, 256 MB per second bus with a pixel span protocol to facilitate high data throughput." (*Id.* at 8) (citing '637 Patent at 3:45–47.) Plaintiff argues that both graphics requests and video data requests travel the same path upon exiting the graphics engine and "are written over a wide high speed bus 17 to a set of resolvers 13, which in turn control reading and writing of the frame buffer 14." (*Id.*) (quoting '637 Patent at 2:35–40, 18:49–58). Plaintiff further argues that the resolvers are

implemented using application-specific integrated circuits ("ASICs") in the preferred embodiment. (*Id.*) (citing '637 Patent at 4:1–16).

Regarding Defendants' construction, Plaintiff argues that it includes the specific name of the high speed bus in the preferred embodiment, and "ASICs that implement the IZ bus protocol." (*Id.*) Plaintiff argues that the particular name "IZ bus" does not itself connote any particular structure relative to the "storing" function, and should not be included in the construction. (*Id.*) Plaintiff further argues that the "IZ bus protocol" included in Defendants' identification of structure is neither structure nor an algorithm necessary to the "storing" function. (*Id.*) According to Plaintiff, the IZ bus protocol is a data format used for communications over the IZ bus in the preferred embodiment. (*Id.*) Plaintiff further argues that Defendants' identification of ASICs is deficient because it does not explain that the resolver ASICs are specific structures for "control[ling] the flow of data to and from the frame buffer memory." (*Id.* at 9.)

Defendants respond that a means-plus-function term must be limited to the structure in the specification that is linked to the claimed function. (Dkt. No. 151 at 8.) Defendants argue that claim 6 requires that the "graphics engine includ[e the] means for storing the associated image in the frame buffer." (*Id.*) Defendants contend that Figure 5 shows that the graphics engine 42 includes IZ bus interface 55, which connects to the IZ bus 47. (*Id.*) Defendants argue that this is the only disclosed structure by which the graphics engine stores data to the frame buffer. (*Id.*) (citing '637 Patent at 3:8–10, 18:56–58, and 4:10–12). Defendants further argue that the patentees could have, but did not disclose a generic bus, and instead disclosed only an "IZ bus." (*Id.*) Defendants also argue that the specification describes the IZ bus as a specially-designed bus for handling both graphics and video data. (*Id.*) (citing '637 Patent at 5:1–25, 19:65–21:64).

Defendants further contend that the "resolvers" identified by Plaintiff cannot be a part of the structure because they are not a part of the graphics engine as required by claim 6. (*Id.*) Defendants argue that Figure 1 shows that the resolvers identified by Plaintiff are not in the graphics engine. (*Id.* at 9.) According to Defendants, Figure 5 shows that the IZ bus interface (55) is connected to the IZ bus (47) and is a part of the graphics engine. (*Id.*)

Plaintiff replies that in describing Figure 1, the specification describes both the structure and the claimed function by explaining that the graphics engine writes pixel requests "over a wide high speed bus 17 to a set of resolvers, which in turn control reading and writing of the frame buffer 14." (Dkt. No. 155 at 4) (quoting '637 Patent at 2:37–40). Plaintiff further argues that Defendants' argument that the "resolvers" cannot be corresponding structure because they are not "in the graphics engine" would also exclude any connected bus. (*Id.* at 5.) Plaintiff contends that the bus is connected to the graphics engine in Figures 1, 2, 4 and 5, but is not "included in" the graphics engine under Defendants' narrow interpretation. (*Id.*)

For the following reasons, the Court finds that the phrase is governed by 35 U.S.C. § 112, ¶ 6, and construes the term as follows.

**b) Analysis**

The phrase "means for storing the associated image in the frame buffer" appears in claim 6 of '637 Patent. Having reviewed the claims, the Court finds that the phrase is governed by 35 U.S.C. § 112, ¶ 6. The disputed phrase uses the words "means" and specifies a function, thus the Court presumes that the patentees intended to invoke the statutory mandates for means-plus-function clauses. *York Prods. v. Central Tractor Farm & Family Ctr.*, 99 F.3d 1568, 1574 (Fed. Cir. 1996) ("In determining whether to apply the statutory procedures of section 112, ¶ 6, the use of the word 'means' triggers a presumption that the inventor used this term advisedly to invoke

the statutory mandates for means-plus-function clauses.").

"The first step in construing [a means-plus-function] limitation is a determination of the function of the means-plus-function limitation." *Medtronic*, 248 F.3d at 1311. Having reviewed the intrinsic evidence, the Court agrees that the recited function is "storing the associated image in the frame buffer." Having determined the limitation's function, "the next step is to determine the corresponding structure disclosed in the specification and equivalents thereof." *Medtronic*, 248 F.3d at 1311. The specification states that the corresponding structure that performs the recited function of "storing the associated image in the frame buffer" includes a wide, high-speed bus, one or more resolvers, and frame buffer. Specifically, the specification states the following:

> The architecture can be broken down into six basic subsystems. Of these two are optional. The basic system has a graphics engine 12 that accepts requests via a FIFO buffer 11. *These requests are broken down by the graphics engine 12 into pixel requests, which are written over a wide high speed bus 17 to a set of resolvers 13, which in turn control reading and writing of the frame buffer 14.* The back end subsystem 18 reads the frame buffer and displays it on the screen of a suitable monitor.

'637 Patent at 2:34–43 (emphasis added). In describing the preferred embodiment, the specification identifies that the "wide high speed bus" may include "IZ bus 27," which "is a 64 bit, 256 MB per second bus with a pixel span protocol to facilitate high data throughput." *Id.* at 3:44–46. The specification further describes the resolver and frame buffer as follows:

> *The four resolver ASICs 331-334 on the graphics processor board control the flow of data to and from the frame buffer memory*. These resolvers provide a 256-bit frame buffer interface. This wide interface allows the embodiment to achieve a very high drawing bandwidth. *The resolvers thus coordinate the movement of data from the frame buffer memory to the video selector and mapper (VSM) ASIC chips 381-388 and DAC 39 (digital to analog converter) in the display subsystem.* The graphics engine ASIC 22 sends interpolated pixel data over the IZ bus 27 to the resolver ASIC's IZ input FIFO. The pixel data consists of color (Red, Green, Blue, and Alpha--RGBA)and depth (Z) data. The resolver IZ input FIFO is 128 words deep, so that the resolver can accommodate bursts of pixel data without slowing overall system performance.

> *The resolver examines pixel data from the input FIFO and determines if it should write the pixel data to the VRAM on the frame buffer board.* The resolver performs Z, mask and alpha tests to determine if data should be written.
>
> . . .
>
> The resolvers 331-334 are thus a set of identical ASICs that accept span requests over the IZ bus 27 *and read from or write to the frame buffer formed by VRAM memory chips 341-314*.

*Id.* at 4:2–35 (emphasis added). Accordingly, the Court finds that the corresponding structure that performs the function of "storing the associated image in the frame buffer" is a wide high speed bus, one or more resolvers, and frame buffer.

Defendants argue that the "resolvers" cannot be a part of the structure because they are not a part of the graphics engine. (Dkt. No. 151 at 8.) The Court disagrees and finds that this is an overly narrow reading of claim 6 in light of the specification. For example, the specification explicitly states that "the four resolver ASICs 331-334 on *the graphics processor board* control the flow of data to and from the frame buffer memory." '637 Patent at 4:2–4 (emphasis added). Likewise, the specification states a number of times that it is the resolvers that perform the function of "storing the associated image in the frame buffer." For example, the specification states, "a set of resolvers 13, which in turn control reading and writing of the frame buffer 14." *Id.* at 2:35–40; *see also* 4:2–4 ("[T]he four resolver ASICs 331-334 on *the graphics processor board* control the flow of data to and from the frame buffer memory.") (emphasis added); 4:33–36 ("The resolvers 331-334 are thus a set of identical ASICs that accept span requests over the IZ bus 27 *and read from or write to the frame buffer formed by VRAM memory chips 341-314*") (emphasis added); 6:48–50 ("Finally, the texture processors route the textured pixels to the resolvers 331-334 and the resolvers 331-334 write the textured pixels into the frame buffer 341-344."). Moreover, Defendants' argument is inconsistent with its proposed "IZ bus" language, because the connected "IZ bus" is also not in the "graphics engine."

Regarding Defendants' "IZ bus" and the "IZ bus protocol" language, the Court finds that

the "IZ bus" is identified as one possible structure that can perform the recited function. However, the description of the "IZ bus" and "IZ bus protocol" is in the context of the preferred embodiment. And, as discussed above, the specification indicates that the "IZ bus" is not the only disclosed bus that can perform the recited function. *Micro Chemical, Inc. v. Great Plains*, 194 F.3d 1250, 1258 (Fed. Cir. 1999) ("When multiple embodiments in the specification correspond to the claimed function, proper application of § 112, ¶ 6 generally reads the claim element to embrace each of those embodiments.").

Specifically, the Court finds that a wide, high-speed bus would embrace each of the disclosed embodiments. As an example of a wide high speed bus, the specification describes the "IZ bus" as a 64 bit, 256 MB per second bus. '638 Patent at 3:44–47. In sum, the Court finds that the claims must include each of the embodiments described in the specification. That said, the Court agrees with Defendants that Plaintiff's proposed "bus" language is too broad, and could read on structures not identified in the specification. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court finds that the phrase **"means for storing the associated image in the frame buffer"** is governed by 35 U.S.C. § 112, ¶ 6, and construes the phrase as follows:

**Function: storing the associated image in the frame buffer.**

**Corresponding Structure: a wide, high-speed bus, one or more resolvers, and frame buffer, and equivalents thereof.**

### 2. "3D graphics request code"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "3D graphics request code" | "3D graphics commands sent to a graphics engine" | "3D drawing commands from a host processor" |

#### a) The Parties' Positions

The parties dispute two issues: (1) whether the claimed 3D graphics request code originates from a host processor, as Defendants propose, and (2) whether the request is for 3D drawing commands, as Defendants propose. Defendants contend that it is known in the art that a request is a "transaction that is generated by a requester, to initiate an action on a responder." (Dkt. No. 153 at 6) (citing Dkt. No. 153-3 at 4 (IEEE Standard Dictionary of Electrical and Electronic Terms, Sixth Ed. 1996)). Defendants argue that the specification repeatedly confirms that the external source for these requests is a host processor. (*Id.*) (citing 637 Patent at 3:50–56, 6:63–65, Abstract; 1:19–23; and 2:56–3:3). Defendants further argue that during prosecution, the applicants emphasized the graphics request that is passed over the host processor to the graphic board. (*Id.*) (citing Dkt. No. 151-3 at 8 ('637 Patent FH, July 16, 1997 Office Action Response)). Defendants also contend that one of the inventors confirmed that the claimed request code is "received from a host processor." (*Id.* at 7) (citing Dkt. No. 153-4, Young Tr. at 98:4–7).

Regarding Plaintiff's construction, Defendants argue that it effectively vitiates the word "request" and repeats language already present in the claim. (*Id.*) Defendants argue that claim 1 already identifies the graphics engine as the recipient of the 3D graphic request code. (*Id.*) According to Defendants, Plaintiff merely repeats that the graphics engine receives the request code rather than saying anything about the request. (*Id.*)

Plaintiff responds that Defendants have not articulated any reason to change "graphics" to "drawing." (Dkt No. 158 at 6.) Plaintiff argues that the term "graphics" is what is used in the claims and is easily understood. (*Id.*) Plaintiff further argues that the specification explains "3D

graphics request code" is not limited to "drawing commands," but may include graphics requests such as "draw, data movement, and control operations affecting the frame buffer." (*Id.*) (quoting '637 Patent at 6:63–65).

Plaintiff also argues that while the claimed graphics processing system may operate in conjunction with other components or systems, the claims do not set any specific details for those components or systems. (*Id.*) Plaintiff contends that the lack of any such requirement in the claims is highlighted by Defendants' attempt to embed a structural detail from the preferred embodiment ("host processor") into a non-structural claim term ("request code"). (*Id.*) Plaintiff also contends that when the applicant intended to specify a source or destination outside of the claimed 3D graphics processing system, he specifically did so. (*Id.*) (citing '637 Patent at 22:22–24). According to Plaintiff, Defendants' "host processor" limitation improperly imports a detail from the preferred embodiment that has no basis in the claim. (*Id.*)

For the following reasons, the Court finds that the phrase **"3D graphics request code"** should be construed to mean **"a request to perform a 3D graphics operation."**

**b) Analysis**

The phrase "3D graphics request code" appears in claims 1 and 3 of the '637 Patent. The Court finds that the phrase is used consistently in the claims and is intended to have the same meaning in each claim. The claim language indicates that the 3D graphics request code is a request to perform an operation. Specifically, the claims recite that the graphics engine processes the request unless it is interrupted. Likewise, dependent claim 3 recites that "3D graphics request code includes a plurality of requests." The specification also indicates that the request relates to 3D graphic operations by stating that "[t]he graphics engine 22 is request-based, in that it receives requests from a host processor to perform draw, data movement, and control operations

affecting the frame buffer." '637 Patent at 6:63–65. Similarly, the specification states that the "[r]equests include graphic primitives (points, lines and triangles), rectangular fill, get/put pixel data, blits, and control requests." *Id.* at 3:53–55. Accordingly, the Court finds that a person of ordinary skill in the art would understand "3D graphics request code" to mean "a request to perform a 3D graphics operation."

Turning to Defendants' construction, the Court is not persuaded that it should read into this claim a requirement that the request must originate from a "host processor." The Court agrees that the preferred embodiments state that "[t]he graphics engine 22 is request-based, in that it receives requests from a host processor." However, there is nothing in the intrinsic record that would require specifying the source of the request. Instead, the claims require that the graphics engine receive the request and then perform the requested operation.

Moreover, the Court agrees with Plaintiff that there is no reason to read the structural limitation of "host processor" into the non-structural term "request code." Defendants have failed to show that the applicants distinguished the prior art based on the host sending the request. Instead, the Court finds that in the prosecution history cited by Defendants, the applicants distinguished the prior art based on the prior art's inability to allow "the graphics engine to look for an interruptible point (i.e. to break up large requests) so as to prevent stalling." (Dkt. No. 151-3 at 9 ('637 Patent FH, July 16, 1997 Office Action Response)). Contrary to Defendants' contention, the portion of the prosecution history they cite relates to a general description of the idea that "[g]raphics requests presented to a graphics engine for processing generally have several stages." *Id.* at 8.

Regarding Plaintiff's construction, the Court agrees with Defendants that it appears to remove the idea of a "request" and repeats language present in the claims. As discussed, the

specification makes clear that the "[t]he graphics engine 22 is request-based, in that it receives requests from a host processor to perform draw, data movement, and control operations affecting the frame buffer." '637 Patent at 6:63–65. Plaintiff's construction fails to capture this aspect because sending a command is not necessarily the same as receiving a request. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

#### c) Court's Construction

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **"3D graphics request code"** to mean **"a request to perform a 3D graphics operation."**

### 3. "graphics engine controller"

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "graphics engine controller" | Plain and ordinary meaning. In the alternative "logic that directs or regulates the graphics engine." | "controller component within the graphics engine" |

#### a) The Parties' Positions

The parties dispute whether the graphics engine controller must be a component within the graphics engine, as Defendants propose. Defendants contend that claim 1 requires that the "graphics engine controller" is responsible for interrupting processing of 3D graphics request code. (Dkt. No. 153 at 8.) Defendants argue that the specification states that the graphics engine ("GE") performs this claimed interrupting, requiring that this functionality exists within the graphics engine. (*Id.*) (citing '637 Patent at 19:30–35). According to Defendants, because the graphics engine finds interruptible points, and the "graphics engine controller" performs this function in the claim, it follows that the graphics engine controller must be located within the graphics engine. (*Id.*)

Defendants also contend that the inventors confirmed that the "multiplexer is what looks for the interruptible point," "the multiplexer is within the graphics engine," and "the graphics engine would have control circuitry for handling the interrupting mechanism provided by the multiplexer." (*Id.*) (citing Dkt. No. 153-5, Deming Tr. at 66:16–68:12). Defendants also argue that Plaintiff's construction fails because it contradicts the plain meaning and applicants' statements to the PTO. (*Id.*) (citing Dkt. No. 151-3 at 8–10 ('637 Patent FH, July 16, 1997 Office Action Response)).

Plaintiff responds that the "graphics engine controller" is "for interrupting processing by the graphics engine," and need not "interrupt[] processing" itself. (Dkt. No. 158 at 7.) Plaintiff also argues that the "graphics engine" and the "graphics engine controller" are separate claim limitations, and the claims do not require the controller to be wholly "within the graphics engine," as Defendants propose. (*Id.*) Plaintiff contends that Figure 4 shows that the "controller" in the preferred embodiment is shared between the graphics engine and the JPEG card. (*Id.*) Plaintiff argues that the specification describes shared protocol, or "logic," used in the preferred embodiment, which begins with a control signal generated by the video controller. (*Id.*) ('637 Patent at 19:31–32, 21:45–55).

Plaintiff further contends that dependent claim 2 also contradicts Defendants' construction by adding a limitation that the "graphics engine controller includes a video input controller." (*Id.* at 8.) According to Plaintiff, the video input FIFO and the video input controller are located on the JPEG card, not in the graphics engine. (*Id.*) (citing '637 Patent at 18:61–65). Plaintiff also argues that Defendants are correct that the intrinsic evidence states that the graphics engine identifies an interruptible point in the request stream where it can suspend processing. (*Id.*) However, Plaintiff argues that does not require the "controller" to be "within the graphics

engine." (*Id.*) Plaintiff contends that the graphics engine processes the request stream, and when it receives a control signal from the controller, the graphics engine looks for an interruptible point in the request stream. (*Id.*) (citing '637 Patent at 3:57–59, 19:28–35, 19:30–33, and 21:45–55). According to Plaintiff, none of this requires the controller to be "within the graphics engine."

For the following reasons, the Court finds that the term **"graphics engine controller"** should be given its plain and ordinary meaning.

**b) Analysis**

The term "graphics engine controller" appears in claims 1, 2, and 5 of the '637 Patent. The Court finds that the term is used consistently in the claims and is intended to have the same meaning in each claim. The Court further finds that claim 1 recites two other elements that interact with the "graphics engine controller." These elements are "a graphics engine" and "a video input." The claim language further recites how the "graphics engine controller" interacts with the "graphics engine" and the "video input." Specifically, claim 1 recites that the graphics engine controller interrupts "processing by the graphics engine . . . when the digital video data stream is received by the video input," and that the graphics engine controller permits "priority processing of the digital video data stream when the processing of 3D graphics request code is interrupted." The specification further describes the interaction between these elements. Specifically, the specification states:

> On the graphics processor board 56 is located registered transceiver 413 on the JPEGDT bus 413 to facilitate communication of the video data stream between the processor board 56 and the JPEG board 412. The graphics engine transceiver 51 handles its interface with both types of data. The normal pipe 52 in the GE 42 can be bypassed by the JPEG pipe 53 in the case of video data requests. The two data data streams are combined by multiplexer 541 before going to the IZ bus interface 55 and then onto the IZ bus 47.

'637 Patent at 18:41–58. Thus, a person of ordinary skill in the art would understand what the

"graphics engine controller" is by referring to the surrounding claim language. Moreover, in the context of the claim language, the Court finds the term "graphics engine controller" is unambiguous, and is easily understandable by a jury, and should be given its plain and ordinary meaning.

Turning to the parties' construction, Plaintiff argues that the "graphics engine" and the "graphics engine controller" are separate claim limitations, and that the claims do not require the controller to be wholly "within the graphics engine." The Court agrees. For example, Figure 5 illustrates that the graphics engine may include functional components of the graphics engine controller, but does not require the graphics engine controller to be wholly within the graphics engine. Indeed, the intrinsic evidence indicates that the "graphics engine controller" is located on the graphics processing board 56, and illustrates components of the graphics engine controller that are not within the graphics engine 42. Specifically, the specification states that "[o]n the graphics processor board 56 is located registered transceiver 413 on the JPEGDT bus 413 to facilitate communication of the video data stream between the processor board 56 and the JPEG board 412." '637 Patent at 18:49–54. Figures 4 and 5 illustrate the registered transceiver 413 outside of the "graphics engine" block. Thus, although the intrinsic evidence indicates that there may be shared logic or functionality between the graphics engine controller and the graphics engine, there is no requirement that the "graphics engine controller" be "within the graphics engine," as Defendants propose. Accordingly, the Court rejects Defendants' argument that the controller must be located entirely within the graphics engine.

Regarding Plaintiff's alternate construction, the Court agrees with Defendants that it is too broad and would allow the controller logic that directs the graphics engine to come from anywhere. As discussed above, the intrinsic evidence indicates that the recited "graphics engine

controller" is located on the graphics processing board. Indeed, during prosecution the applicants argued that that the prior art (Guttag) did not teach "how to modify a *graphics processor board*" to support "a control arrangement for allowing such preemptive processing of incoming video data." (Dkt. No. 151-3 at 12 ('637 Patent FH, July 16. 1997 Office Action Response) (emphasis added). Accordingly, the Court rejects Plaintiff's argument that the controller logic that directs the graphics engine is not required to be on the graphics processor board. Finally, the Court has considered the extrinsic evidence submitted by the parties, and given it its proper weight in light of the intrinsic evidence.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court finds that the term **"graphics engine controller"** should be given its **plain and ordinary meaning.**

**4. "span break"**

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "span break" | "an interruptible point in the request stream" | "end of a horizontal sequence of adjacent pixels in a graphics primitive" |

**a) The Parties' Positions**

The parties dispute whether a span break refers to the end of a horizontal sequence of adjacent pixels in a graphics primitive, as Defendants propose, or if it refers to an interruptible point in the request stream, as Plaintiff proposes. Defendants contend that the specification confirms that a "span" means a sequence of horizontally adjacent pixels, and that these pixels are part of a graphics primitive. (Dkt. No. 153 at 9) (citing '637 Patent at 3:57–59). Defendants argue that the specification also unambiguously shows that the sequence of adjacent pixels in a span is "in a graphics primitive." (*Id.*) (citing '637 Patent at 3:31–33, 3:51–56, and 6:29–34). Defendants further argue that the purpose of a span is widely understood in the art to define the

pixels of a scan line that fall within a primitive. (*Id.*) (Dkt. No. 153-6 at 14 (Computer Graphics: Principles and Practice (1990))).

Defendants also argue that the intrinsic evidence indicates that a "break" is the "end of a" sequence of adjacent pixels. (*Id.* at 10) (citing '637 Patent at Claim 5, 19:21–35). Defendants contend that when the graphics engine locates a "span break" it can interrupt processing. (*Id.*) Defendants further argue that the prosecution history makes clear that the applicants intended "break" to mean the end of a particular pixel sequence. (*Id.*) (citing Dkt. No. 151-3 at 9 ('637 Patent FH, July 16, 1997 Office Action Response)); (citing Dkt. No. 153-7 at 9 ('637 Patent FH, March 26, 1998 Office Action Response)); (citing Dkt. No. 153-8 at 5 ('637 Patent FH, April 28, 1998 Office Action Response)). According to Defendants, the graphics engine uses a "span boundary" (the end of a span) to signal that processing a graphics primitive has reached an interruptible point where a request has been broken into a smaller request. (*Id.* at 11) (citing '637 Patent at 19:44–45).

Defendants also argue that Plaintiff's construction effectively reads the term "span" out of the claim and contradicts the intrinsic record. (*Id.*) Defendants argue that the specification discloses breaking spans into subspans and further describes that graphics processing is interrupted at these span breaks. (*Id.*) (citing '637 Patent at 19:44–45). Defendants contend that a span break is not just any point at which graphics processing is interrupted. (*Id.*) Defendants also argue that during prosecution, the applicant noted that if graphics processing was "arbitrarily interrupted in favor of video data," then "video data commonly would be lost." (*Id.*) (quoting Dkt. No. 153-7 at 9 ('637 Patent FH, March 26, 1998 Office Action Response)). Defendants contend that Plaintiff's construction contradicts the intrinsic record's explicit teaching that to properly process graphics and video data, graphics processing is specifically interrupted at

breaks in spans rather than at any arbitrary point. (*Id.*)

Plaintiff responds that its construction comes straight from the specification. (Dkt. No. 158 at 9) (citing '637 Patent at 19:21–35). Plaintiff argues that Defendants focus much of their discussion on the meaning of "span," and miss the point that the '637 Patent does not have to process an entire span, but can break the span into subspans as necessary to process video data. (*Id.*) Plaintiff contends that Defendants' construction would eliminate this critical point from claim 5. (*Id.*)

Plaintiff also argues that the specification explains that the graphics engine in the preferred embodiment breaks graphics requests into "'span' requests—requests to read or write a horizontal sequence of adjacent pixels." (*Id.*) (quoting '637 Patent at 3:57–59). According to Plaintiff, the '637 Patent describes that "span requests" may be broken into "subspans" when necessary to process incoming video data. (*Id.*) ('637 Patent at 19:21–23, 19:28–30). Plaintiff argues that the critical point is that the interruptible point need not be the "end of a horizontal sequence of adjacent pixels in a graphics primitive," but is determined by the graphics engine itself. (*Id.* at 10–11.) Plaintiff contends that the prosecution history makes this same point. (*Id.* at 10) (citing Dkt. No. 144-11 at 72 ('637 Patent FH, January 17, 1997 Office Action Response)).

For the following reasons, the Court finds that the term **"span break"** should be construed to mean **"an interruptible point in a scan line and indicated by a span boundary."**

**b) Analysis**

The term "span break" appears in claim 5 of the '637 Patent. The Court finds that the plain language of the claims indicates that "span break" is a break or interruptible point in a scan line. The parties appear to agree that a span is a horizontal sequence of adjacent pixels. (*See, e.g.*, Dkt. No. 153 at 9); (Dkt. No. 158 at 9). Both parties cite to the specification, which states that

"requests are broken down into 'span' requests—requests to read or write a horizontal sequence of adjacent pixels." '637 Patent at 3:57–59. Likewise, both parties cite to the Computer Graphics: Principles and Practice text, which states that "[a] contiguous sequence of pixels on a scan line . . . is called a span." (Dkt. No. 153-6 at 13 (Computer Graphics: Principles and Practice (1990))). Accordingly, a person of ordinary skill in the art would understand that a "span break" is a break in an interruptible point in a scan line.

The specification further states that "the GE 42 has the capability of breaking spans into subspans." '637 Patent at 19:22–23. The specification adds that "[t]he critical point is that the Graphics Engine 42 does not have to complete the normal requests which have been partially processed before reading the Video FIFO." *Id.* at 19:28–30. To accomplish this, "the GE looks for an interruptible point at which to stall the normal request stream. When that point is reached, the normal request stream is stalled, JPEG is processed, and then processing is resumed in the normal request stream." *Id.* at 19:31–35. Importantly, the specification indicates that the interruptible point the GE is looking for is a span boundary. *Id.* at 19:44–45 ("The output FIFO level must show that the GE will reach an interruptible point in the get (*i.e. a span boundary*) before filling the output FIFO.") (emphasis added).

In other words, a person of ordinary skill in the art would understand that the graphics engine does not arbitrarily interrupt the request stream, but instead looks for an interruptible point to avoid losing video data. Indeed, during prosecution the applicant argued that that if graphics processing was "arbitrarily interrupted in favor of video data," then "video data commonly would be lost," (Dkt. No. 153-7 at 9 ('637 Patent FH, March 26, 1998 Office Action Response)). Accordingly, the Court finds that "span break" should be construed as "an interruptible point in a scan line and indicated by a span boundary"

Regarding the parties' constructions, the Court does not adopt Defendants' language of "end of a horizontal sequence of adjacent pixels" because it implies that there is only one interruptible point in a scan line. This is inconsistent with the specification, which states that "the GE 42 has the capability of breaking spans into subspans." '637 Patent at 19:22–23. Therefore, the Court rejects Defendants' argument to the extent that they contend that the interruptible point can only be at the end of a scan line. As indicated in the Computer Graphics: Principles and Practice text, a scan line may include multiple spans of pixels. (Dkt. No. 153-6 at 9) ("In scan line 8 of Fig. 3.22, for instance, there are two spans of pixels within the polygon.").

Turning to Plaintiff's construction, the Court does not adopt it because it reads "span" out of the claims. It also would contradict the intrinsic evidence by allowing the "span break" to take place at an arbitrary point in the request stream. As discussed above, the specification indicates that the interruptible point is indicated by a span boundary. '637 Patent at 19:44–45. A span boundary is the end of a span or subspan, which may or may not be the end of a scan line.

**c) Court's Construction**

In light of the intrinsic and extrinsic evidence, the Court construes the term **"span break"** to mean **"an interruptible point in a scan line and indicated by a span boundary."**

**E. The '584 Patent**

The parties' dispute focuses on the meaning and scope of two phrases in the '584 Patent.

1. **"said pixels of each said word being grouped in said stored texture in a pattern which is more than one pixel wide and more than one pixel high," "said pixels of each said word being grouped in said stored texture in a pattern which extends over more than one pixel in multiple orthogonal directions"**

| Disputed Term | Plaintiff's Proposal | Defendants' Proposal |
|---|---|---|
| "said pixels of each said word being grouped in said stored texture in a pattern which is more than one pixel wide and more than one pixel high" | "texture pixels in a word whereby the texture pixels form a set that is more than one pixel wide and more than one pixel high" | "said pixels of each word being physically grouped in a pattern that extends over more than one row and more than one column in memory hardware" |
| "said pixels of each said word being grouped in said stored texture in a pattern which extends over more than one pixel in multiple orthogonal directions" | "texture pixels in a word whereby the texture pixels form a set that is more than one pixel wide and more than one pixel high" | "said pixels of each word being physically grouped in a pattern that extends over more than one row and more than one column in memory hardware" |

**a) The Parties' Positions**

The parties agree these phrases require a two-dimensional (2D) pattern or set of pixels that is more than one row or pixel high and more than one column or pixel wide. The parties dispute whether the 2D pattern must exist physically in memory hardware, as Defendants propose, or if the 2D pattern instead refers to a 2D set of pixels in an image, as Plaintiff proposes. Plaintiff contends that claim 1 requires retrieving pixels grouped in a stored texture in a "pattern which is more than one pixel wide and more than one pixel high." (Dkt. No. 144 at 11.) Plaintiff argues that Defendants misread the claim to require a physical grouping of the pixels across "more than one row and more than one column in memory hardware." (*Id.*) Plaintiff contends that nowhere does the claim or specification require that the texture pixels must be physically stored in memory in the same two-dimensional pattern as their location in a texture map. (*Id.* at 11–12.) According to Plaintiff, such a construction would eliminate the benefit the claimed invention seeks to achieve. (*Id.* at 12.)

Plaintiff further argues that the specification describes retrieving a multi-pixel patch on memory reads so that "the frequency of memory accesses and page breaks in particular are reduced." (*Id.*) (citing '584 Patent at Abstract). Plaintiff notes that the specification states that "[a]s the memory accesses are made in a two-dimensional patch and, as described above, tend to

contain a greater number of useful texels on each access, the number of total memory accesses across memory pages is reduced." (*Id.*) (quoting '584 Patent at 6:16–20). Plaintiff argues that the result is fewer page breaks to retrieve a given set of texture pixels. (*Id.*) ('584 Patent at 59:43–44, 70:65–67). According to Plaintiff, the specification makes clear that the point is to retrieve a two-dimensional patch of texture pixels in a single read of a memory location without having to read multiple rows and columns of memory hardware. (*Id.*)

Plaintiff also argues that the specification indicates that the two-dimensional texture patch coordinates must be converted to a linear memory address as part of the texture retrieval and storage process. (*Id.* at 13) (citing '584 Patent at 62:32–34, 67:59–61, 70:43–46). Plaintiff contends that this makes clear that the 2D patch of texture pixels does not correspond to a 2D set of storage locations in physical memory. (*Id.*)

Plaintiff further argues that the file history teaches against Defendants' construction. (*Id.*) (citing Dkt. No. 145-1 at 163–164 ('584 Patent FH)). Plaintiff contends that the applicants' reference to "more than one pixel in multiple orthogonal directions," says nothing about more than one row or more than one column in physical memory. (*Id.*) Plaintiff contends that neither the patent specification nor the file history ever discusses storing the 2D patch of pixels across multiple rows and columns in physical memory hardware as Defendants propose. (*Id.*)

Plaintiff further argues that the specification's discussion of the multiple 2D pixel patching arrangements for textures is followed by a discussion about storing the entire patch in a single page in memory. (*Id.* at 14.) According to Plaintiff, a page in physical memory is in a single row. (*Id.*) (citing Dkt. No. 145-1 at 25 ('584 Patent FH)). Plaintiff argues that file history notes the "specific advantages of this approach, e.g. in terms of page breaks on memory access." (*Id.*) (citing Dkt. No. 145-1 at 164 ('584 Patent FH)). Plaintiff contends that an advantage of the

claimed invention discussed in the file history was to retrieve a two dimensional patch of pixels in a single read for more efficient memory access. (*Id.*) Plaintiff argues that this cannot occur if the patch stretches across multiple rows of physical memory, as Defendants' construction would require. (*Id.*) According to Plaintiff, Defendants' construction would exclude the preferred embodiment.

Defendants argue that during prosecution, the PTO rejected the pending claims five times over two similar references, U.S. Patent Nos. 5,495,563 ("Winser '563") and 5,544,292 ("Winser '292"). (Dkt. No. 151 at 13.) Defendants contend that Winser '563 discloses a "texture address converter" that converts "2-D texture coordinate pair[s] . . . into a linear physical texture memory address" or a "linear (one-dimensional) address." (*Id.*) (quoting Dkt. No. 151-12 (Winser '563 at 2:50–55, 2:59–62)). Defendants argue that the applicants repeatedly distinguished the Winser references as disclosing 2D patterns in texture space stored in one row of memory, arguing that the claims instead require 2D patterns that are physically grouped in memory hardware. (*Id.*) (citing Dkt. No. 151-14 at 3–4 ('584 Patent FH, August 17, 1998 Office Action Response)); (citing Dkt. No. 151–15 at 5-6 ('584 Patent FH, January 24, 2000 Office Action Response)).

Defendants contend that the applicants explained that the "plain meaning" of Claim 1—which recites "stored texture" data in "memory" and pixels "grouped" in a 2D pattern—refers to a "physical grouping of . . . pixels in memory" or a "patch as it is stored in memory hardware," not a "patch of texture space" as in the Winser references. (*Id.* at 14.) Defendants further argue that even if the plain meaning could be broader, the applicants unequivocally disavowed storing a 2D pixel pattern in one row of memory or converting 2D texture coordinates into linear memory addresses. (*Id.*)

Plaintiff replies that the '584 Patent is directed to a 2D texture patch arrangement stored

linearly in memory. (Dkt. No. 155 at 5.) Plaintiff argues that this is the purpose of the invention—to maximize the amount of relevant data captured in a single memory read. (*Id.*) (citing '584 Patent at Abstract). Plaintiff argues that requiring multiple reads to multiple rows (or pages) of memory would eliminate the advantage realized by the patent. (*Id.* at 6.)

Regarding the prosecution history, Plaintiff contends that the applicants distinguished Winser by arguing that it requires four separate reads for each 2x2 texture patch. (*Id.*) (citing Dkt. No. 145-1 at 164 ('584 Patent FH)). Plaintiff contends that this is in contrast to the '584 Patent, which groups pixels together in a single memory location accessible by a single read from a single page in memory. (*Id.*) (citing '584 Patent at Claim 1, 62:9–17). Plaintiff argues that in this configuration, the patch of pixels is physically grouped, sequentially in a single page in memory. (*Id.*) Plaintiff contends that the 2x2 language refers exclusively to the configuration in the texture map, not physical memory. (*Id.*) (citing'584 Patent at 61:63–62:8, 70:43–46).

Finally, Plaintiff argues that Defendants' disclaimer argument fails because it completely contradicts the understanding of a skilled artisan. (*Id.*) Plaintiff contends that Defendants asked the inventor sixteen different times whether the claims require data to be stored in multiple rows of memory hardware, and all sixteen times he said "no." (*Id.*) (citing Dkt. No. 155-3, Murphy Tr. at 51:14–20, 51:23–52:6, 53:24–54:11, 70:25–71:2, 70:4–7, 106:18–24, 123:6–12, 126:16–24, 129:21–24, 130:3–6, 130:7–19, 131:3–7, 135:12–18, 136:2–5, 137:14–22, and 193:2–18).

For the following reasons, the Court finds that the phrase **"said pixels of each said word being grouped in said stored texture in a pattern which is more than one pixel wide and more than one pixel high"** should be construed to mean **"texture pixels in a word whereby the texture pixels form a set that is more than one pixel wide and more than one pixel high and stored in a single memory location,"** and that the phrase **"said pixels of each said word**

**being grouped in said stored texture in a pattern which extends over more than one pixel in multiple orthogonal directions"** should be construed to mean **"texture pixels in a word whereby the texture pixels form a set that is more than one pixel wide and more than one pixel high and stored in a single memory location."**

**b) Analysis**

The phrase "said pixels of each said word being grouped in said stored texture in a pattern which is more than one pixel wide and more than one pixel high" appears in claim 1 of the '584 Patent. The phrase "said pixels of each said word being grouped in said stored texture in a pattern which extends over more than one pixel in multiple orthogonal directions" appears in claim 4 of the '584 Patent. The Court finds that the claim language itself contradicts Defendants' construction. Specifically, claim 1 recites "retrieving at least four pixels of a stored texture from a single memory location." In contrast to a single memory location, Defendants' construction requires the pixels to be "physically grouped in a pattern that extends over more than one row and more than one column in memory hardware." This is inconsistent with not only the claim language, but also the stated goal of '584 Patent. Specifically, the '584 Patent states that the innovative approach is to reduce the frequency of memory accesses and page breaks to achieve a lower memory access time overhead. '584 Patent at Abstract, 6:16–20.

Turning to Plaintiff's construction, the Court finds that it is consistent with the intrinsic evidence. The specification describes retrieving a patch of pixels from "texture memory" as a wide word containing a "mini-patch" of pixels. '584 Patent at 5:61–6:6. By retrieving a "wide word," each word fetched from texture memory can contain multiple pixels. *Id.* The specification states that these multiple pixels are a "mini-patch" of texture coordinates (i.e., pixel positions in a texture map or image) that includes pixels that are "vertical and horizontal

neighbors." *Id.* The specification adds that "[a]s the memory accesses are made in a two-dimensional patch and, as described above, tend to contain a greater number of useful texels on each access, the number of total memory accesses across memory pages is reduced." *Id.* at 6:16–20. Accordingly, the Court finds that when the specification addresses storage in memory, it is discussing how a word containing a 2D patch of pixels is stored in a single memory location. *Id*.

Indeed, under the heading "Texture Read," the specification states that "[t]he texture maps can be organized as a patch to reduce the frequency of page-breaks." *Id.* at 59:43–44. In another example, the specification states, "At 4 bits per texel it [sic] a 4x4 patch fits in a 64-bit word; at 8 bits a 4x2 patch fits in 64 bits, and at 16 bits a 2x2 patch fits in 64 bits." *Id.* at 70:65–67. As these passages indicate, the specification discloses a system that can retrieve a two-dimensional patch of texture pixels in a single read of a memory location without having to read multiple rows and columns of memory hardware. This is consistent with Plaintiff's construction and inconsistent with Defendants' construction. Accordingly, the Court rejects Defendants' construction because it would exclude the preferred embodiment. The Court finds that this is not the "rare case in which such an interpretation is compelled." *Elekta Instrument S.A. v. O.U.R. Sci. Int'l*, 214 F.3d 1302, 1308 (Fed. Cir. 2000); *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1583–84 (Fed. Cir. 1996) ("A claim construction that excludes the preferred embodiment is rarely, if ever, correct and would require highly persuasive evidentiary support.").

Turning to the prosecution history, the Court finds that the applicant did not clearly and unambiguously disclaim the preferred embodiment. Consistent with the disclosure in the specification, the applicant argued Winser stores texture patches in four separate addresses, then retrieves all four addresses in parallel. (Dkt. No. 145-1 at 164 ('584 Patent FH)) ("Winser uses four distinct memory blocks to allow a logical 2x2 patch to be stored."). Thus, Winser requires

four separate reads for each 2x2 texture patch. (Dkt. No. 145-2 at 44 (‘584 Patent FH)). In contrast, the ‘584 Patent discloses grouping pixels together in a single memory location accessible by a single read from a single page in memory. *See, e.g.*, ‘584 Patent at Claim 1 (“retrieving . . . from a single memory location”); 62:9–17 (discussing maximizing the amount of data stored in “one page in memory”).

In this configuration, the patch of pixels is physically grouped sequentially in a single page in memory. As the applicant argued during prosecution, “[t]he application clearly describes the actual physical grouping of the pixels in memory, and discusses specific advantages of this approach, e.g., in terms of page breaks on memory access.” (Dkt. No. 145-1 at 164 (‘584 Patent FH)). Accordingly, the Court finds that the 2x2 language refers to the configuration in the texture map, not physical memory. *See, e.g.*, ‘584 Patent at 61:63–62:8, 70:43–46. Accordingly, the Court finds that the applicant did not clearly and unambiguously disclaim grouping pixels together in a single memory location accessible by a single read from a single page in memory. Thus, the Court generally adopts Plaintiff’s construction, but adds that the texture pixel set is stored in a single memory location.

**c) Court’s Construction**

In light of the intrinsic and extrinsic evidence, the Court construes the phrase **“said pixels of each said word being grouped in said stored texture in a pattern which is more than one pixel wide and more than one pixel high”** to mean **“texture pixels in a word whereby the texture pixels form a set that is more than one pixel wide and more than one pixel high and stored in a single memory location,”** and the phrase **“said pixels of each said word being grouped in said stored texture in a pattern which extends over more than one pixel in multiple orthogonal directions”** to mean **“texture pixels in a word whereby the**

**texture pixels form a set that is more than one pixel wide and more than one pixel high and stored in a single memory location"**

## V. CONCLUSION

The Court adopts the above constructions. The parties are ordered that they may not refer, directly or indirectly, to each other's claim construction positions in the presence of the jury. Likewise, the parties are ordered to refrain from mentioning any portion of this opinion, other than the actual definitions adopted by the Court, in the presence of the jury. Any reference to claim construction proceedings is limited to informing the jury of the definitions adopted by the Court.

**It is SO ORDERED.**

**SIGNED this 25th day of June, 2015.**

ROY S. PAYNE
UNITED STATES MAGISTRATE JUDGE