(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 436 959 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **90125789.9**

(51) Int. Cl.5: **G09G 1/28**, G09G 5/14

(22) Date of filing: **28.12.90**

(30) Priority: **08.01.90 US 459537**

(43) Date of publication of application:
17.07.91 Bulletin 91/29

(84) Designated Contracting States:
DE GB NL

(71) Applicant: **Intergraph Corporation**
One Madison Industrial Park
Huntsville, Alabama 35807-4201(US)

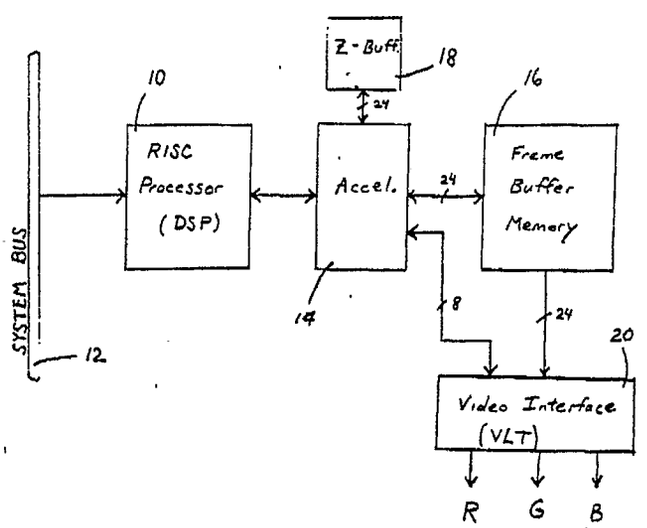
(72) Inventor: **Imsand, Bruce E.**
2627 Trailway Road
Huntsville, Alabama 35801(US)
Inventor: **Terry, Joseph Clayton**
3517 Wildwood Drive
Huntsville, Alabama 35801(US)
Inventor: **Pesto, William Steve, Jr.**
114 Greenwood Drive
Madison, Alabama 35758(US)

(74) Representative: **von Bülow, Tam, Dipl.-Ing.**
Dipl.-Wirtsch. et al
Patentanwälte Samson & Bülow,
Widenmayerstrasse5
W-8000 München 22(DE)

(54) Digital color video system using dirty bits to provide high speed pixel data for raster displays.

(57) A digital color video system provides a "Dirty Bit" in its pixel data to provide rapid throughput in the reading and writing of the pixel data to a raster display screen for multi-windowed environments having z-buffered images. The dirty bit combines the Flash Erase function of DRAMs with the referencing of each pixel by its associated property or context to provide direct mapping to a predetermined null context when the pixel is responsive to the dirty bit, and to enable rapid writing of multiple model components in 3-D space into new image data for the pixel frame buffer memory and z buffer memory without first clearing the previous data therein.

FIGURE 1



EP 0 436 959 A2

DIGITAL COLOR VIDEO SYSTEM USING DIRTY BITS TO PROVIDE HIGH SPEED PIXEL DATA FOR RASTER DISPLAYS

BACKGROUND OF THE INVENTION

Field of the Invention:

The present invention relates to raster display video systems generating high speed pixel data for displays utilized in multi-window environment workstations.

Description of the Related Art:

Recently, inexpensive video RAMs, fast digital signal processing devices and RISC processors have been employed to drive high resolution raster displays. Nevertheless, the raster display drivers have been unable to output more complex images, such as a window having an image in motion, without having a delay in the output of the image. Since the pixel data is written to and read from a frame buffer memory as the image changes, the key to providing a distortion-free moving image is a fast throughput in the reading and writing of the pixel data.

The need for fast throughput is especially acute when using three-dimensional video processing techniques such as Gouraud shading and z-buffering. Gouraud shading performs a bilinear interpolation to determine the red, green and blue values of the pixels which display the model component of the image. The use of z-buffering positions arbitrary model components of the display image in the 3-D model space with respect to each other such that a pixel for one model component will be written over a pixel for a second model component of the same display image only if the first model component is logically determined to be in front of the second model component in the 3-D model space. The z-buffering technique is especially important to position arbitrary model components of the display image in 3-D space where there is no prior information of the relative location of the model components. The need for fast throughput is found when a picture perspective changes during animated rendering, whereby a display driver will erase the z-buffer as the perspective of the model components in the window changes. However, a major obstacle to fast throughput is the time needed to erase the z-buffer and the working buffer of a frame buffer memory before writing the next image, while currently displaying the present image of the display buffer of the frame buffer memory; the time needed to erase the buffers one pixel at a time may take as long as 16 milliseconds. Thus, distortions or time lags may be present during animated rendering of the windowed images.

The need for fast throughput is also found in the generation of true color and pseudo oolor images. A multi-windowed system may use both pseudo color and true color images in the respective windows; for example, if a raster display includes three windows, each having its respective application program, window A and B may show pseudo-color images, whereby window C may show a true color image. Pseudo color images are user-defined and may be generated by using 8 bits, for example, from the frame buffer memory to map 2^8 simultaneously displayable colors from a palette of 16 million ($2^8 \times 2^8 \times 2^8 = 2^{24} =$ 16,777,216); the assignment of the eight bit index to the palette of 16 million is done through a video look-up table (VLT), which, through D/A converters, drives each of the red, green and blue guns in the CRT. True color images generally need more than 2^8 actual colors, so 24 bits (e.g., planes) of the frame buffer memory are used to generate the colors; however, since the RGB electron guns, the associated amplifiers and the human eye are all nonlinear, look-up tables are still needed to provide correction factors for the true color images.

Attempts have been made to increase throughput speed with limited success. For example, dynamic random access memories (DRAMs) have been developed for video applications that have a bit map commonly arranged with rows by columns corresponding to pixels in the display (e.g., 512 x 512), whereby a particular memory cell is accessed by specifying a row address and a column address. A Flash Erase (or Fill) function has been developed in the DRAMs, whereby upon the receipt of a Flash Erase command either a 0 or 1 is forced into every location along a particular row within one memory cycle. In a windowed environment, however, where a memory cell corresponds to a pixel, conventional frame buffer memories cannot use the Flash Erase function, since the Flash Erase function would operate on every pixel within the particular row, independent of the conditions of the windowed environment. Thus, since the Flash Erase of a given row of memory cannot be controlled with respect to the characteristics of a window, the problems associated with implementing Flash Erase in a frame buffer memory have induced some DRAM manufacturers to eliminate entirely the use of Flash Erase in video memory architectures. Also, there is no Flash Erase in conventional DRAMS which are typically used for the 2-buffer memory.

SUMMARY OF THE INVENTION

The display color video system of the present invention provides fast throughput in the reading and writing of the pixel data to minimize distortions, especially during animated rendering of windowed images. This fast throughput is realized by a novel combination of the Flash Erase function of DRAMs with successive referencing of each pixel by its associated property or context.

Since each window of a multi-window applications environment will display certain images in accordance with its respective applications program, it is desirable to consider each window as having its own set of properties or contexts of interpretation, so that all pixels of the respective window are identified by the properties corresponding to the window. Thus, each pixel has an associated ID properties bundle which enables the pixel to be mapped through a look-up table in order to generate the appropriate data for the RGB digital-to-analog (D/A) converters associated with the cathode ray tube (CRT). In addition, each pixel has a second properties bundle which defines the context of interpretation of the respective ID properties bundle. This second properties bundle enables rapid operations via a secondary look-up table on all pixels whose ID properties bundles have a given property defined within the predetermined context specified by the secondary properties bundle. Thus, on a pixel-by-pixel basis, a given context of a window can be quickly modified with respect to the properties defined within that context.

The present invention takes advantage of the Flash Erase function of DRAM technology by defining at least one of the bits of the second properties bundle as a "Dirty Bit", capable of being Flash Erased by the DRAM. The display color video system of the present invention has a memory organization defining each pixel to include frame buffer memory data for real or pseudo colors, z-buffer data, and pixel tag data. The frame buffer memory data and the z-buffer data are stored in memory that does not use the Flash Erase function. The pixel tag data for each pixel, however, which includes the ID properties bundle and the second properties bundle, is stored in a DRAM having the Flash Erase function. At least one of the second properties bundle bits is reserved as a dirty bit, which utilizes the Flash Erase function; by defining at least one bit of the pixel data as a dirty bit, a Flash Erase command will set all the dirty bits for all pixels of the raster display screen to either a 0 or a 1, whichever is preferred. The display of the pixel, in light of the set dirty bit, can then be determined through a look up table in accordance with the remaining pixel tag data.

Since the pixel tag data serves to define a specific context of a given window, once the dirty bit for all pixels is set, assuming that there is no animation occurring except for the given window, a context may be defined through a look up table such that the dirty bit invalidates the properties of the pixels of the given window, and redirects the pixels within that window to a predetermined null context; as a result, the pixel color data and z data for the redirected pixels are invalidated by the predetermined null context. Alternatively, another context may be defined to disregard the dirty bit, so that the pixels associated with the alternate context are not redirected to the predetermined null context when their dirty bit is set. Thus, on a pixel by pixel basis, the display color video system of the present invention can selectively switch between different contexts by assigning a bit of the pixel ID to be a dirty bit, flash filling all the dirty bits of all the pixels, and then determining whether the pixel associated with a particular context is defined to respond to or disregard the dirty bit. If the pixel is responsive to the dirty bit, the look up table directly maps the pixel to the null context; if, however, the pixel is to disregard the dirty bit, the pixel is output in accordance with its existing context.

The dirty bits of the pixel data are also used during the pixel-by-pixel image construction stage to more quickly construct in memory the model components of the next windowed image to be displayed. Those pixels that are responsive to the setting of dirty bits enter into a null context, such that the corresponding z value is irrelevant; since the z value of any pixel responding to the dirty bit is defined to be irrelevant whenever a pixel is written whose dirty bit is set, the z value of the existing image is ignored and the pixel data for the next model component is automatically written into the frame buffer memory and the z buffer memory, and the dirty bit for the pixel is reset. For example, in a double buffered frame buffer memory, before drawing a new model component in a windowed image, a working buffer and the z buffer in a conventional system would be erased on a pixel-by-pixel basis at locations corresponding to the windowed image while the display buffer was outputting the display data for the current image. In the present invention, however, the erase can be achieved much more quickly by flash setting the dirty bits for all pixels of the display image in response to a flash set signal. The working buffer is then drawn along with the z buffer memory using standard z-buffering techniques when drawing multiple model components of the display image (e.g., comparing the z value of the existing pixel of the existing model component with the z value of the corresponding pixel of the next model component), except that whenever a pixel is written whose dirty

bit is set, the z value is ignored and the new pixel data is written into the working buffer memory.

The reading of the dirty bit during the construction of the image is done in parallel with the reading of the z value to maintain speed performance. By repeating the process whenever the frame buffer memory needs to be updated due to animated rendering in the window, the new image can be written into the frame buffer memory, without ever needing to clear the frame buffer memory or the z buffer memory. The result is the savings of a substantial amount of time.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the invention will be described in detail with reference to the drawings wherein like reference numerals denote like or corresponding parts throughout, wherein:

FIGURE 1 is a system block diagram of a digital color video system according to a preferred embodiment of the present invention;

FIGURE 2 is an exemplary block diagram of the data associated with each pixel in accordance with the present invention;

FIGURE 3 is a block diagram of the frame buffer memory and the video interface of the digital color video system of FIGURE 1;

FIGURE 4 is an exemplary schematic of the organization of a video look up table of the video interface of FIGURE 3;

FIGURE 5A shows an arrangement of a pixel tag look up table of the video interface of FIGURE 3;

FIGURE 5B shows exemplary outputs for three display applications from the pixel tag look up table of FIGURE 5A;

FIGURE 5C shows exemplary inputs corresponding to the outputs of FIGURE 5B; and

FIGURE 6 is a block diagram of an image construction driver, within the accelerator of FIGURE 1, according to the preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIGURE 1 is a system block diagram of a digital color video system for a raster display according to the present invention. A RISC processor 10 having digital signal processing (DSP) capabilities receives basic image and control data from a system bus 12. The system bus 12 may be an I/O bus which passes the basic image and control data from, for example, a central computer. The RISC processor 10 performs all necessary DSP functions before outputting video data to a hardware accelerator 14. The accelerator 14 constructs the image to be displayed by writing the appropriate pixel data to a frame buffer memory 16, a z buffer memory 18 and a video interface 20; the accelerator 14 in constructing the image to be displayed may also perform functions such as Gouraud shading, drawing anti-aliased vectors, etc.

The z-buffer memory 18 stores the z value for each respective pixel. The z value includes, for example, 24 bits for a possible 2^{24} gradations in depth as positioned in the 3-D model space. The frame buffer memory 16 stores the color data for each pixel; if the pixel color data is to be displayed as true color data, the frame buffer memory 16 is arranged to provide, for example, 24 planes (e.g., bits) of actual frame buffer memory; if, however, the pixel color data is to be displayed as pseudo color data, the frame buffer memory 16 is arranged to provide 12 planes of frame buffer memory for each of a front buffer and a back buffer, whereby the front buffer acts as a display buffer by providing the pixel color data for the display image while the back buffer acts as a working buffer, e.g., the pixel color data for the next image is written into the back buffer by the accelerator 14, and vice versa.

The video interface 20 contains video look up tables (VLT) which contain color code data for the RGB electron guns; the VLTs map the pixel color data on a pixel-by-pixel basis from the frame buffer memory 16 to a predetermined context in accordance with corresponding pixel tag data (described in detail later). The color code data stored in the VLTs is preloaded before each new image is to be output.

FIGURE 2 is an exemplary schematic of the data associated with each pixel, in accordance with the present invention. Bits 0-23 for each pixel are designated for z values, and are stored in the z buffer memory 18. Bits 24-47 designate the color data for the respective pixel and are stored in the frame buffer memory 16; if pseudo color is desired, the frame buffer memory 16 is divided into a front and back buffer, such that bits 24-35 are reserved for the pseudo color code stored in the front buffer, and bits 36-47 are reserved for the pseudo color code stored in the back buffer. Bits 48-55 designate the pixel tag data for each pixel. At least one of the bits of the pixel tag data is designated a dirty bit.

The pixel tag data for each pixel is used to characterize the pixel with respect to the context or the properties the pixel is intended to represent. For example, in a multi-window applications environment, each window will display certain images in accordance with its respective applications program; those images of the window can be characterized as having a unique set of properties or contexts of interpretation. Since each pixel represents a color corresponding to one of the contexts,

the pixel can be identified by the property or context it represents, as opposed to its location on the raster display. Thus, by identifying each pixel by the context in which the pixel is represented, a given context in a window can be quickly modified with respect to the properties defined within the context. Therefore, the value of the pixel tag data indicates the context in which the pixel is intended to be represented.

The pixel tag data for each pixel includes an ID properties bundle and a second properties bundle. The ID properties bundle indicates the context which the pixel is intended to represent, and maps the pixel data through the VLT of the video interface 20 to generate the appropriate data for the RGB D/A converters of the CRT (not shown). The second properties bundle include at least one bit reserved as a "Dirty Bit", which is used to provide rapid writing of the pixel data in the frame buffer memory, and to map pixels whose ID properties bundles have a specific property to a predetermined null context. The second properties bundle characterizes the ID properties bundle in order to define a context of interpretation of the respective ID properties bundle (e.g., even the properties represented by the ID properties bundle have their own properties). Thus, the accelerator 14 can identify all ID properties bundles having a specific property by their corresponding second properties bundle, and can modify through a secondary look up table the pixels whose ID properties bundle have a specific property in accordance with the second properties bundle.

The features and advantages associated with using a dirty bit in the pixel tag data will become more readily apparent in the following description. FIGURE 3 is a block diagram of the frame buffer memory 16 and the video interface 20 of FIGURE 1. The video interface 20 receives pixel color data from the frame buffer memory 16 and outputs digital color code data to D/A converters 22R, 22G and 22B in accordance with the corresponding pixel tag data. The pixel tag data for each pixel is stored in a pixel tag memory 24, which is connected to the accelerator 14 via a bi-directional bus. The pixel tag memory 24 is a DRAM having a Flash Erase function. In this embodiment, the DRAM of the pixel tag memory 24 is arranged so that two bits of the 8 bit pixel tag, DIRTY FB and DIRTY BB, are reserved as dirty bits which can be set to 0 in response to an external Flash Erase command. Generation of the Flash Erase command may be application specific, and may, for example, arise from an external command from the system bus 12. The external Flash Erase command has two bits to selectively flash set to 0 either DIRTY FB, DIRTY BB, or both. When the pixel tag memory 24 receives a Flash Erase command, the appropriate dirty bit is simultaneously set for all pixels of the raster display by flash setting the particular column of the DRAM having the appropriate dirty bit. The pixel tag data also includes HIGHLIGHT FB and HIGHLIGHT BB bits, and a 4-bit ID properties bundle. Thus, the second properties bundle of the pixel tag includes the bits DIRTY FB, DIRTY BB, HIGHLIGHT FB and HIGHLIGHT BB.

The pixel tag memory 24 outputs the pixel tag data, in response to a corresponding address signal (not shown), to a pixel tag look up table (LUT) 26. The pixel tag LUT 26 maps the pixel color data from the frame buffer memory 16 to the appropriate color contexts stored in VLTs 28R, 28G and 28B; thus, the pixel tag LUT 26 maps the pixel color data to a particular context of interpretation according to the properties of the respective ID properties bundle by selecting a Major Context, and a Minor Context of the VLTs 28R, 28G and 28B (described later). The pixel tag LUT 26 also arranges the pixel color data via switching devices 30R, 30G and 30B, depending on whether the pixel color data represents pseudo or true color data. If the pixel color data represents pseudo color data, the pixel tag LUT 26 outputs a buffer select signal (FB/BB) to a switching device 32. The switching device 32 then outputs the pseudo color data from either the front buffer or back buffer of the frame buffer memory 16, in accordance with the buffer select signal.

The switching devices 30R, 30G and 30B provide the appropriate pixel color data to the VLTs 28R, 28G and 28B, each of which are 16k x 8 bit RAMs, in response to a Pseudo Select signal from the pixel tag LUT 26. If the Pseudo Select signal is 0, indicating true color data, the 4 bits of Minor Context data are combined with each of the 8 bits of the appropriate R,G and B true color data from the frame buffer memory 16, thus providing 12 bits of pixel color data output to the respective VLTs 28; if the Pseudo Select signal is 1, indicating pseudo color data, the same 12 bits of pseudo color data are output to the VLTs 28R, 28G and 28B. Two bits of Major Context data are added to each of the 12 bits of color data from the switch devices 30R, 30G and 30B, thereby providing a total of 14 bits input to the VLTs 24R, 28G and 28B. Thus, in true color mode, the data input to each VLT 28 includes 8 bits of the corresponding true color data from the frame buffer memory 16, 4 bits of Minor Context data, and 2 bits of Major Context data; in pseudo color mode, however, the data input to each VLT 28 includes 12 bits of pseudo color data from the frame buffer memory 16, and 2 bits of Major Context data. Hence, the minor context data is not used for mapping in the VLT 28 during pseudo color mode.

FIGURE 4 is an exemplary block diagram of

one of the VLTs 28 of the video interface 20 in FIGURE 3. The VLT 28 is a 16k x 8 bit RAM subdivided into four major contexts. The 14 bit data input into the VLT 28 serves as an address input, ranging from hexadecimal values 0-3FFF, to access the appropriate color code representing a particular color context. The VLT 28 is updated with the color code data each time a new window is rendered active in the multi-windowed environment.

The VLT 28 is subdivided into four major contexts: reference 40 designates Major Context 0; reference 42 designates Major Context 1; reference 44 designates Major Context 2; and reference 46 Major Context 3. The desired Major Context is addressed by the 2 bits of Major Context data from the pixel tag LUT 26. Major Contexts 0, 1 and 2 are pseudo color contexts of 4k regions, each region being addressable by the 12 bits of pseudo color data from the frame buffer memory 16.

Major Context 3 is a true color context of a 4k region which is further subdivided into 16 minor contexts. Of the 16 minor contexts, true color contexts 0-13 contain conventional contexts for true color mapping; thus, the multiple true color contexts enable up to fourteen separate correction factors. Each minor context has 256 locations, addressable by the 8 bits of true color data from the frame buffer memory 16.

Two of the minor contexts of Major Context 3 are reserved for HIGHLIGHT and DIRTY functions. As mentioned earlier, the function of Dirty Bits is to redirect a pixel designated responsive to its dirty bit to a predetermined null context. That predetermined null context is defined in the DIRTY minor context: all 256 locations of the DIRTY minor context are loaded with the identical color code value. Thus, when a pixel is mapped to Major Context 3, Minor Context 1, (Minor Context 0 being the HIGHLIGHT function), the pixel color data is disregarded and the VLT 28 outputs from the DIRTY minor context the color code value indicating the predetermined null context. Thus, the use of the dirty bit enables the nullification of video data from frame buffer memory 16 without erasing the frame buffer memory 16 or the z buffer memory 18.

The HIGHLIGHT minor context works in a similar manner: a pixel is mapped to the HIGHLIGHT minor context if, in response to the pixel tag bits HIGHLIGHT FB or HIGHLIGHT BB being a value of 1, the pixel tag LUT 26 outputs a Major Context of 3 and a Minor Context of 0. The HIGHLIGHT function can be used to highlight a given pixel to a predetermined highlight color code. This allows a highlighting of the pixel without erasing the true color data in the frame buffer memory 16. After the HIGHLIGHT FB or HIGHLIGHT BB pixel tag bits are reset to 0, the pixel can be displayed in accordance with the original pixel data.

FIGURE 5A shows an arrangement of the pixel tag LUT 26. The pixel tag LUT 26 is segmented with respect to applications being simultaneously displayed on the raster display. Each ID segment is addressed by the ID properties bundle output from the pixel tag memory 24.

FIGURE 5B shows exemplary outputs for three display applications from the pixel tag LUT 26. The output data for Application "A" is stored in ID segment 2 of the pixel tag LUT 26, the output data for Application "B" is stored in ID segment 14, and the output data for Application "C" is stored in ID segment 8. FIGURE 5C shows the inputs to the pixel tag LUT 26 corresponding to the output data described in FIGURE 5B.

The pixel tag LUT 26 will map the pixel color data to either one of two generic contexts if a dirty bit is set: a first context, which responds to the dirty bit, or a second context, which disregards the dirty bit. As shown in FIGURE 5B, Application "A" is a single buffered, 12 bit pseudo color application using highlight and which disregards the dirty bits. A window is created by writing the 12 bits of pixel pseudo color data into the front buffer of the frame buffer memory 16. The pixel tag memory 24 is also written to with the appropriate pixel tag data. When HIGHLIGHT FB is set to 1, the pixel tag LUT 26 outputs a Pseudo Select signal of 0 to the switch devices 30R, 30G and 30B, and outputs a Major Context of 3 and a Minor Context of 0, which maps the pixel color data to the HIGHLIGHT Minor Context of the VLT 28, as shown in FIGURE 4. The output data for Application "A" has redundant entries with respect to the input values of the dirty bits DIRTY FB and DIRTY BB; thus, Application "A" disregards the dirty bits.

As shown in FIGURE 5B, Application "B" is a single buffered, 24 bit true color application since the Pseudo Select signal is 0 for all inputs. While Application "B" uses the HIGHLIGHT function, since it has redundant entries for the different dirty bit values, it disregards the dirty bits. Since the entire 24 bits of the frame buffer memory 16 are used for the true color data, the FB/BB output is disregarded; also, all data is mapped to Major Context 3 in the VLT 28, which contains all the true color contexts. If the HIGHLIGHT FB bit is 1, the HIGHLIGHT minor context is selected (Minor Context = 0); otherwise, Application "B" uses true color context 7 (Minor Context = 9) for the appropriate true color correction.

Application "C" is a double buffered, 12 bit pseudo-color which is responsive to the dirty bits, and which also uses the HIGHLIGHT function. As shown in FIGURE 5B, Application "C" displays the current display image from the back buffer of the frame buffer memory 16 (BB=1). During normal

display of pseudo-color pixels, the pixel tag LUT 26 outputs a Major Context of 1; since the 12 bits of pseudo-color data from the frame buffer memory 16 address the data in Major Context 1 of VLT 28, the Minor Context output by the pixel tag LUT 26 is disregarded, as indicated by an "X" value in FIGURE 5B. If, however, the HIGHLIGHT BB bit is set to 1, the pixel tag LUT 26 outputs a Major Context of 3, Minor Context of 0, to map the pixel to the HIGHLIGHT context. If the DIRTY BB bit is set to 1, the pixel tag LUT 26 outputs a Major Context of 3, Minor Context of 1, thus mapping the pixel to the DIRTY context, which contains the color code data for the null context. Note that the HIGHLIGHT FB and DIRTY FB bits are disregarded while the back buffer is in use (BB=1); if the front buffer were in use, such that BB=0, Application "C" would respond to the HIGHLIGHT FB and DIRTY FB bits and disregard the HIGHLIGHT BB and DIRTY BB bits.

Thus, the pixel tag LUT 26 directs the pixel color data to a predetermined context in response to the ID properties bundle of the pixel and the second properties bundle which includes the front and back buffer bits for the HIGHLIGHT and DIRTY functions.

FIGURE 6 is a block diagram of an image construction driver portion of the display color video system of the present invention. The image construction driver portion 48 is located within the accelerator 14, which constructs the image to be displayed. The accelerator 14 may also include a processor to address the appropriate pixel to be written into the frame buffer memory 16, the z buffer 18 and the pixel tag memory 24. The image construction driver portion 48 enables the rapid drawing of multiple model components of the display image in the 3-D model space of the windowed environment, whereby a first model component will not be displayed in the windowed image if it is logically determined that a second model obstructs the first model component of the display image.

The image construction driver portion 48 has a write decision logic circuit 50 which determines whether new pixel data for a new model component of the display image from the RISC processor 10 should be written over the existing pixel data; the write decision logic circuit 50 examines both the dirty bits of the existing pixel tag data, and compares the z value of the existing pixel in memory with the z value of the corresponding pixel of the new model component. The dirty bits of the existing pixel in memory are temporarily stored in a register 52, and the z value of the existing pixel in memory is temporarily stored in a register 54. The write decision logic circuit 50 outputs a write signal to a memory controller 56 if the dirty bit associated with the current working buffer of the existing pixel in memory is set, or if the z value of the pixel of the new model component of the display image is greater than the z value of the corresponding existing pixel in memory. The memory controller 56 will output a write enable signal to the frame buffer memory 16, the z buffer memory 18 and the pixel tag memory 24 in response to the write signal, thus causing the new pixel data for the new model component of the display image to be written into the appropriate memory. As the new pixel data is written into memory, the dirty bit of the pixel is cleared in order to enable the pixel to be displayed; the dirty bit is cleared by the writing of the new pixel tag data into the pixel tag memory 24. The image construction driver portion 48 repeats the writing procedure until all model components included in the display image have been written, even though portions thereof may be obstructed by other model components that are closer to the display in the 3-D space.

As mentioned earlier, the setting of the dirty bit redirects the pixel to a predetermined null context if the pixel tag data in the pixel tag LUT 26 is defined to be responsive to the dirty bit. When the pixel is redirected to a null context, the z value of the pixel is considered to be at the lowest possible value (e.g., the farthest depth from the front of the raster display screen in the 3-D model space), and is therefore considered irrelevant; as a result, any other context must have a z value greater than that of the null context. However, the actual z value need not be changed. Thus, all images by definition will write over any pixel representing the null context. As a result, the image construction driver portion 48 can output a write signal whenever the dirty bit is set. This use of the dirty bit enables rapid erasures and rapid writing of pixel data into memory.

The procedure of using the flash fill and the dirty bits when writing will be described with respect to Application "C" in FIGURE 5B. While the use of flash fill and the dirty bits is described in context of double-buffered memory using pseudo color images, it is to be understood that the dirty bits can be used in the same manner in the construction of single or double buffered true color images. A raster display initially shows a display image in accordance with pixel data from the front buffer of the frame buffer memory 16. A window showing the image from Application "C" is created in the back buffer of the frame buffer memory 16 by first setting the dirty bit DIRTY BB in the pixel tag memory 24 for all pixels of the entire screen by sending the Flash Erase command to the pixel tag memory 24. The 4 bits of pixel ID data in the pixel tag memory 24 for all of the pixels in the area defining the window are then written with a unique

ID, in this case "8".

While the video interface 20 displays the existing display image data from the front buffer, the new display image for the window in Application "C" is now written into the back buffer of the frame buffer memory 16 and the z buffer memory 18 by sequentially drawing each of the model components included in the display image in accordance with their respective locations in the 3-D model space. Since all pixels in the new window have DIRTY BB set, the first model component of the display image will be automatically written; while standard z-buffering techniques are still used for subsequent model components of the same display image by comparing the z value of the existing model component with the z value of the corresponding pixel of the next model component of the same display image, if a pixel at an intended location of a model component has its dirty bit DIRTY BB set, the z value of the pixel of the new model component of the display image is ignored and the new pixel data for the new model component is automatically written, the dirty bit DIRTY BB being cleared thereafter. The pixel tag LUT 26 is then reloaded during the vertical blanking interval of the raster display to display the back buffer pixels having an ID properties bundle value of 8.

This procedure repeats itself each time the image in the window changes, alternating between the front and back buffers without ever needing to clear the display memory or the z-value memory. Thus, a substantial amount of time is saved.

With conventional graphics architectures the back buffer would need to be first erased pixel-by-pixel along with the z-buffer memory for the window in use before the new image could be drawn. This erasure of the frame buffer and z-buffer memory by conventional means for the new window data is very time consuming because it must be done one pixel at a time. In the present invention, however, the erasure step is bypassed by setting the back buffer dirty bit DIRTY BB in the pixel tag memory 24 for all pixels of the display.

As mentioned earlier, it is generally assumed that there is no other animation occurring on the raster display except in the active window. Thus, only the active window would be responsive to the dirty bits. If animation is desired in another window, it is necessary to clear the dirty bits of the pixels corresponding to that window if the pixel tag LUT 26 is to be updated to make the pixels corresponding to the new active window responsive to the dirty bits.

While in the present invention the concept of using dirty bits in constructing and displaying images has been described with respect to detailed applications in what is presently considered to be the most practical and preferred embodiment, it should be readily apparent that the features of the present invention may be implemented while making numerous modifications as needed for various display applications and raster display systems. Thus, it is to be understood that the invention is not limited to the disclosed embodiment, but, on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

Claims

1. A digital color video system for generating color codes of a display image to a raster display comprising:
a frame buffer memory for storing pixel color data for each of a plurality of pixels;
memory means for storing a value for each of said pixels corresponding to one of a plurality of predetermined contexts in which said pixel color data is to be interpreted, the value having a dirty bit, the memory means flash setting the dirty bits for all pixels of an existing image in response to a flash set signal, the value selectively defining a null context among said plurality of contexts when said dirty bit is set;
a z-buffer memory for storing z-buffer data for each of the pixels;
image construction means for writing said pixel color data of said display image to the frame buffer memory and the corresponding z-buffer data to the z-buffer memory, said image construction means writing said data of each said pixel of said display image over a corresponding pixel of said existing image if:
the dirty bit of said pixel of the existing image is set, or
a z-buffer value for said pixel of said display image is greater than a z-buffer value for the corresponding pixel of said existing image; and
means for generating said color codes in response to said pixel color data for said pixels and said context.

2. An apparatus as recited in claim 1, wherein said means for generating said color codes comprises:
a video look-up table for generating said color codes, said video look-up table storing said plurality of predetermined contexts including said null context; and
a pixel tag look-up table for mapping each pixel into one of said plurality of predetermined contexts in response to said corresponding value, said pixel tag look-up table mapping said pixel to said null context when said dirty bit is flash set if said value indicates a state whereby the pixel is predetermined to be re-

sponsive to said dirty bit.

3. A graphics driver for providing pixel data of a display image to a digital color video display driver, the graphics driver comprising:
a frame buffer memory for storing pixel color data for each pixel, the display driver reading the pixel color data from the frame buffer memory;
memory means for storing a tag value for each of said pixels, the tag value having a dirty bit, the memory means flash setting the dirty bits for all pixels of an existing image in response to a flash set signal;
a z-buffer memory for storing z-buffer data for each pixel; and
image construction means for writing said pixel color data of said display image to the frame buffer memory and the corresponding z-buffer data to the z-buffer memory, said image construction means writing said data of each said pixel of said display image over a corresponding pixel of said existing image if:
the dirty bit of said pixel of the existing image is set, or
a z-buffer value for said pixel of said display image is greater than a z-buffer value for the corresponding pixel of said existing image.

4. An apparatus as recited in claim 3, wherein the frame buffer memory has display buffer means for storing the pixel color data for the existing image and working buffer means for storing pixel color data for a next image, the image construction means writing from an external source the pixel data of the display image to the working buffer means while the display driver reads the existing image data from the display buffer means and the z-buffer memory.

5. An apparatus as recited in claim 3, wherein the image construction means comprises:
write decision means for outputting a write signal if said dirty bit of said pixel of the existing image is set, or if said z-buffer value for said pixel of said display image is greater than the z-buffer value for the corresponding pixel of said existing image; and
memory control means for writing the z-buffer value, the tag value and the pixel color data of the pixel of the display image into the z-buffer, the memory means and the frame buffer memory, respectively, in response to the write signal.

6. An apparatus as recited in claim 3, wherein the dirty bit of said pixel of the existing image is cleared if the image construction means writes the data for the corresponding pixel of the display image.

7. A digital color video display driver, responsive to pixel data of a display image, for generating color codes of the display image to a raster display, the display driver comprising:
a frame buffer memory for storing pixel color data for each of a plurality of pixels;
memory means for storing a value for each of said pixels corresponding to one of a plurality of predetermined contexts in which said pixel color data is to be interpreted, the value having a dirty bit, the memory means flash setting the dirty bits for all pixels of the display image in response to a flash set signal; and
means for generating said color codes in response to said pixel color data for said pixels and said context.

8. An apparatus as recited in claim 7, wherein said means for generating said color codes comprises:
a video look-up table for generating said color codes, said video look-up table storing said plurality of predetermined contexts including said null context; and
a pixel tag look-up table for mapping each pixel into one of said plurality of predetermined contexts in response to said corresponding value, said pixel tag look-up table mapping said pixel to said null context when said dirty bit is flash set if said value indicates the pixel is predetermined to be responsive to said dirty bit.

9. An apparatus as recited in claim 7, wherein said memory means is updated for said next image during a vertical blanking interval of said display.

10. An apparatus as recited in claim 7, wherein the frame buffer memory has a front buffer for storing the pixel color data for the display image and a back buffer for storing pixel color data for a next image.

11. An apparatus as recited in claim 10, wherein:
the apparatus further comprises switching means for selecting the front or back buffer of the frame buffer memory to output the pixel color data to the video look-up table, the switching means selecting the front or back buffer in response to a buffer select signal; and
the pixel tag look-up table outputs the buffer select signal to said switching means in response to the value corresponding to one of said predetermined contexts.

12. An apparatus as recited in claim 7, wherein the memory means simultaneously flash sets the dirty bits for all pixels of the raster display in response to the flash set signal.

13. An apparatus as recited in claim 7, wherein the plurality of predetermined contexts includes a pseudo-color context and a true color context.

14. An apparatus as recited in claim 13, wherein the true color context includes said null context.

15. A method for writing pixel data of a plurality of pixels of a display image to a frame buffer memory and a z-buffer memory, the display image pixel data being written over an existing image, the method comprising the steps of:

(a) flash setting simultaneously a dirty bit for each respective pixel of the existing image in response to a flash set signal;

(b) writing on a pixel-by-pixel basis pixel data of a model component of the display image, said writing step for said each pixel of the model component including the steps of:

(i) writing pixel color data and a z value of the pixel of the model component into the frame buffer memory and the z-buffer memory, respectively, if the dirty bit of a corresponding pixel of the existing image is set,

(ii) writing the pixel color data and the z value of the pixel of the model component into the frame buffer memory and z-buffer memory, respectively, if the dirty bit of the corresponding pixel is not set and the z value of said pixel of the model component is greater than a corresponding z value stored in said z-buffer memory, and

(iii) clearing the dirty bit of the corresponding pixel of the existing image; and

(c) repeating step (b) until all model components of the display image have been written.

16. A method for outputting color codes of a plurality of pixels of a display image to a raster display in response to data for a plurality of model components of the display image, the method comprising the steps of:

(a) flash setting simultaneously a dirty bit for each respective pixel of an existing image in response to a flash set signal;

(b) writing on a pixel-by-pixel basis the data of each of said model components of the display image, said writing step for said each pixel of said each model component including the steps of:

(i) writing pixel color data and a z value of the pixel of the model component into a frame buffer memory and a z-buffer memory, respectively, if the dirty bit of a corresponding pixel of the existing image is set,

(ii) writing the pixel color data and the z value of the pixel of the model component into the frame buffer memory and the z-buffer memory, respectively, if the dirty bit of the corresponding pixel is not set and the z value of said pixel of the model component is greater than a corresponding z value stored in said z-buffer memory, and

(iii) clearing the dirty bit of the corresponding pixel of the existing image;

(c) repeating step (b) until all model components of the display image have been written;

(d) storing a value for each of said pixels of said display image corresponding to one of a plurality of contexts in which said pixel color data of said pixels of said display image is to be interpreted, the value having the dirty bit, the value selectively defining a null context among the plurality of contexts when the dirty bit is set; and

(e) generating the color codes in response to said pixel color data for said pixels and said context.

FIGURE 1

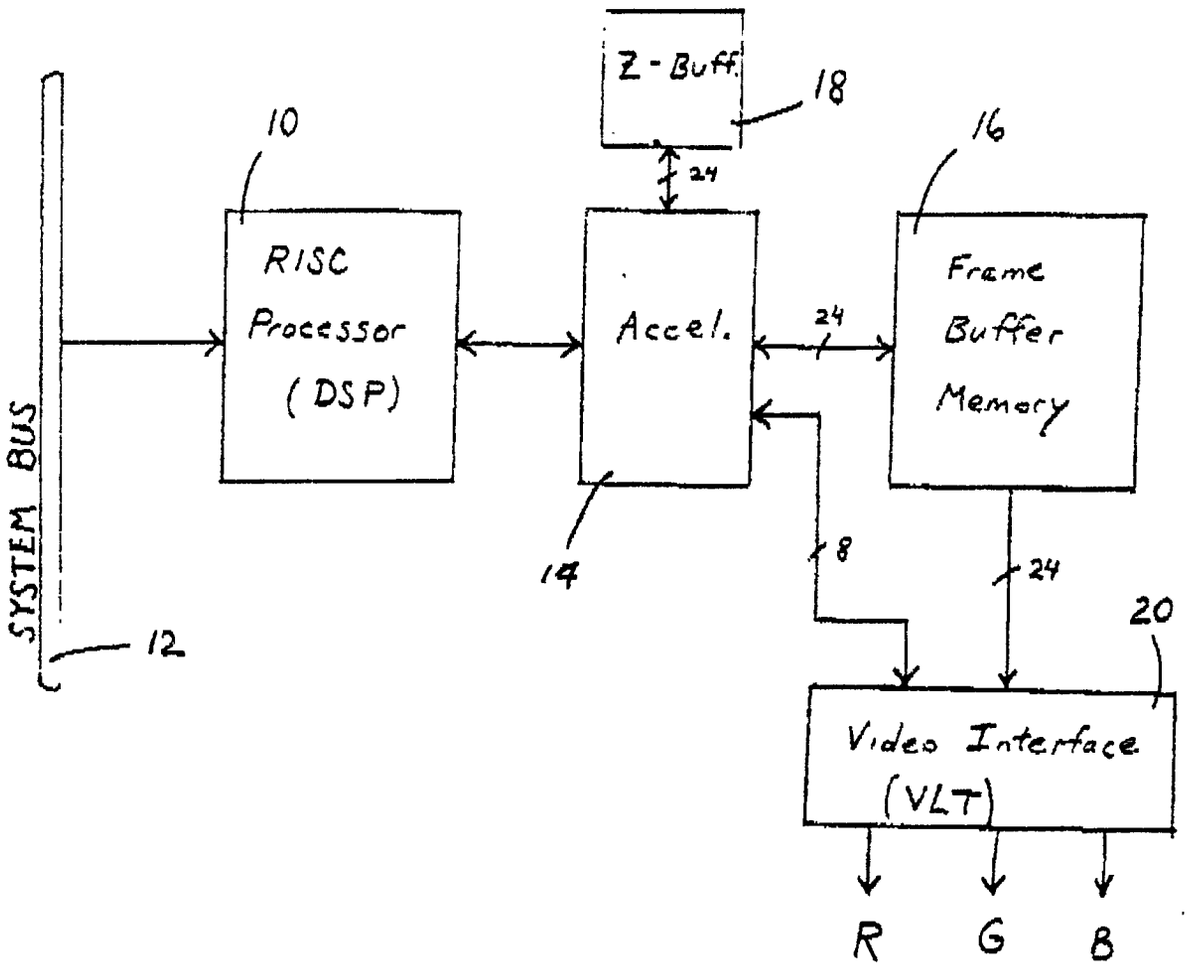
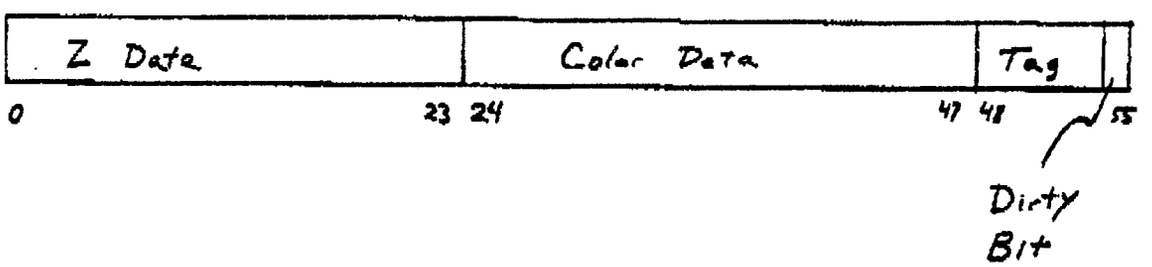


FIGURE 2



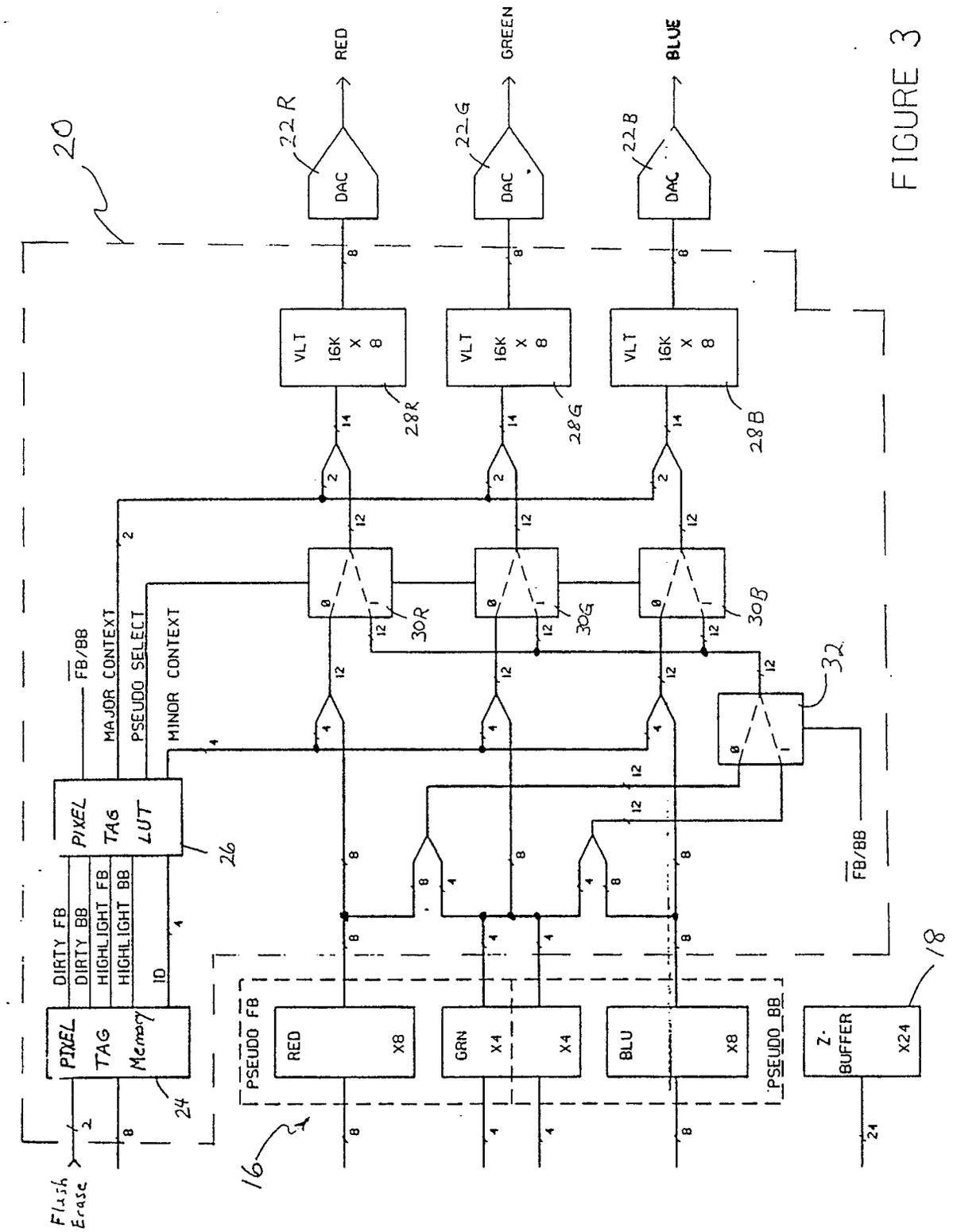
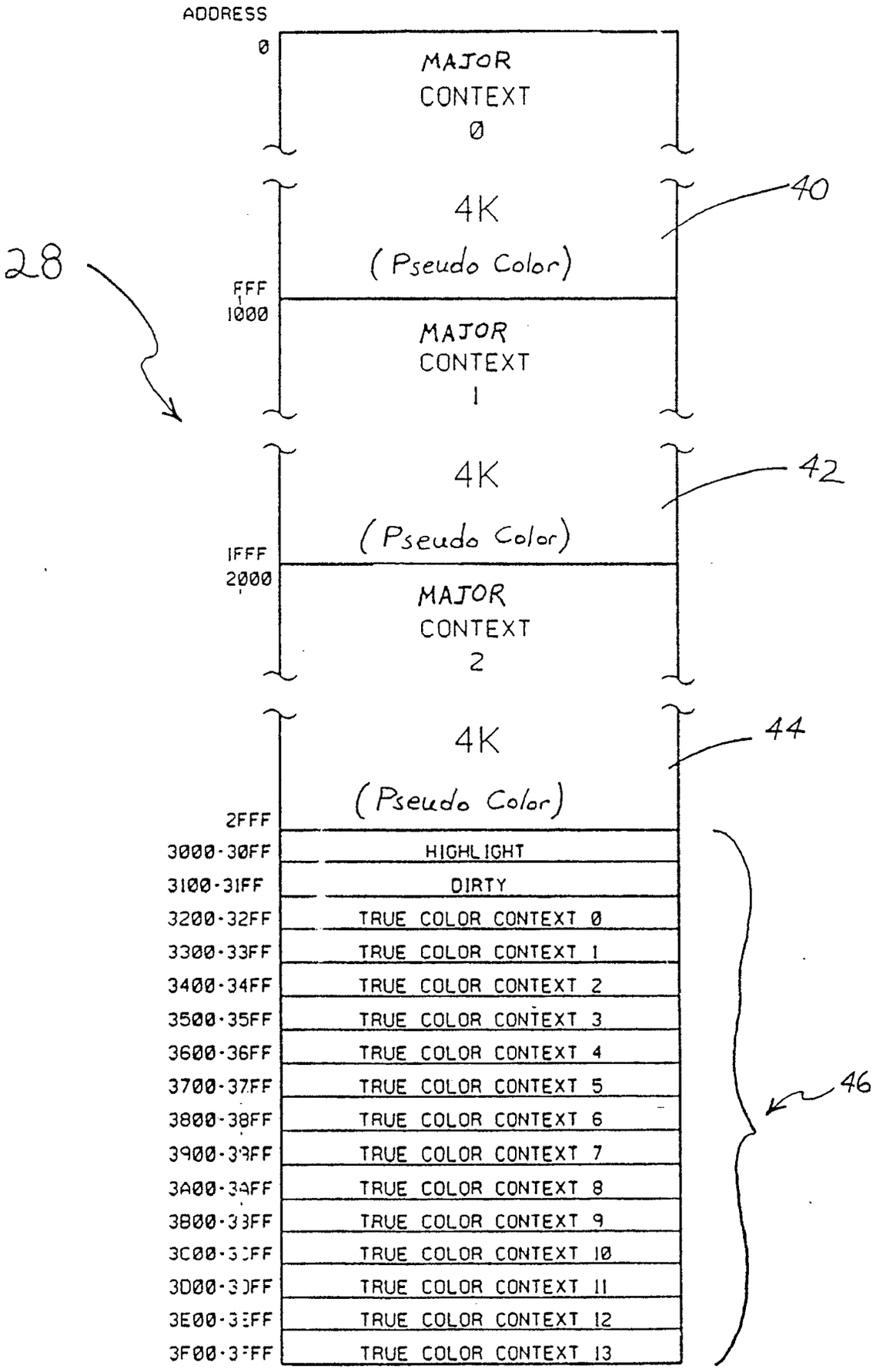

20
22R
22G
22B
DAC
RED
GREEN
BLUE
VLT
16K
X
8
28R
28G
28B
30R
30G
30B
32
FB/BB
MAJOR CONTEXT
PSEUDO SELECT
MINOR CONTEXT
PIXEL
TAG
LUT
26
DIRTY FB
DIRTY BB
HIGHLIGHT FB
HIGHLIGHT BB
ID
PIXEL
TAG
Memory
24
Flash Erase
16
PSEUDO FB
PSEUDO BB
RED
GRN
BLU
X8
X4
Z-
BUFFER
X24
18

FIGURE 3

FIGURE 4


ADDRESS
0
MAJOR CONTEXT 0
4K (Pseudo Color)
40
28
FFF
1000
MAJOR CONTEXT 1
4K (Pseudo Color)
42
1FFF
2000
MAJOR CONTEXT 2
4K (Pseudo Color)
44
2FFF
3000-30FF HIGHLIGHT
3100-31FF DIRTY
3200-32FF TRUE COLOR CONTEXT 0
3300-33FF TRUE COLOR CONTEXT 1
3400-34FF TRUE COLOR CONTEXT 2
3500-35FF TRUE COLOR CONTEXT 3
3600-36FF TRUE COLOR CONTEXT 4
3700-37FF TRUE COLOR CONTEXT 5
3800-38FF TRUE COLOR CONTEXT 6
3900-39FF TRUE COLOR CONTEXT 7
3A00-3AFF TRUE COLOR CONTEXT 8
3B00-3BFF TRUE COLOR CONTEXT 9
3C00-3CFF TRUE COLOR CONTEXT 10
3D00-3DFF TRUE COLOR CONTEXT 11
3E00-3EFF TRUE COLOR CONTEXT 12
3F00-3FFF TRUE COLOR CONTEXT 13
46

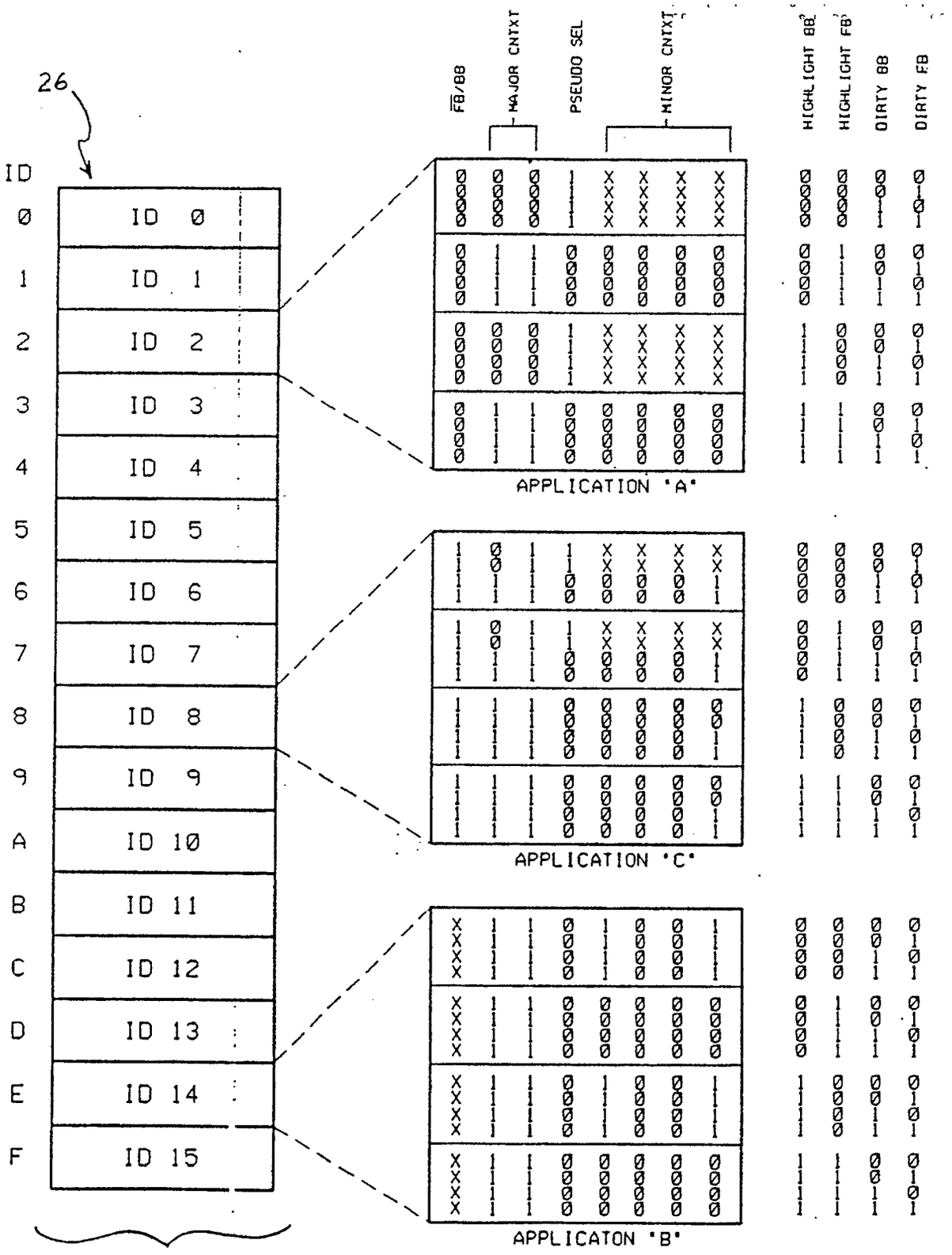


FIGURE 5A

FIGURE 5B

FIGURE 5C

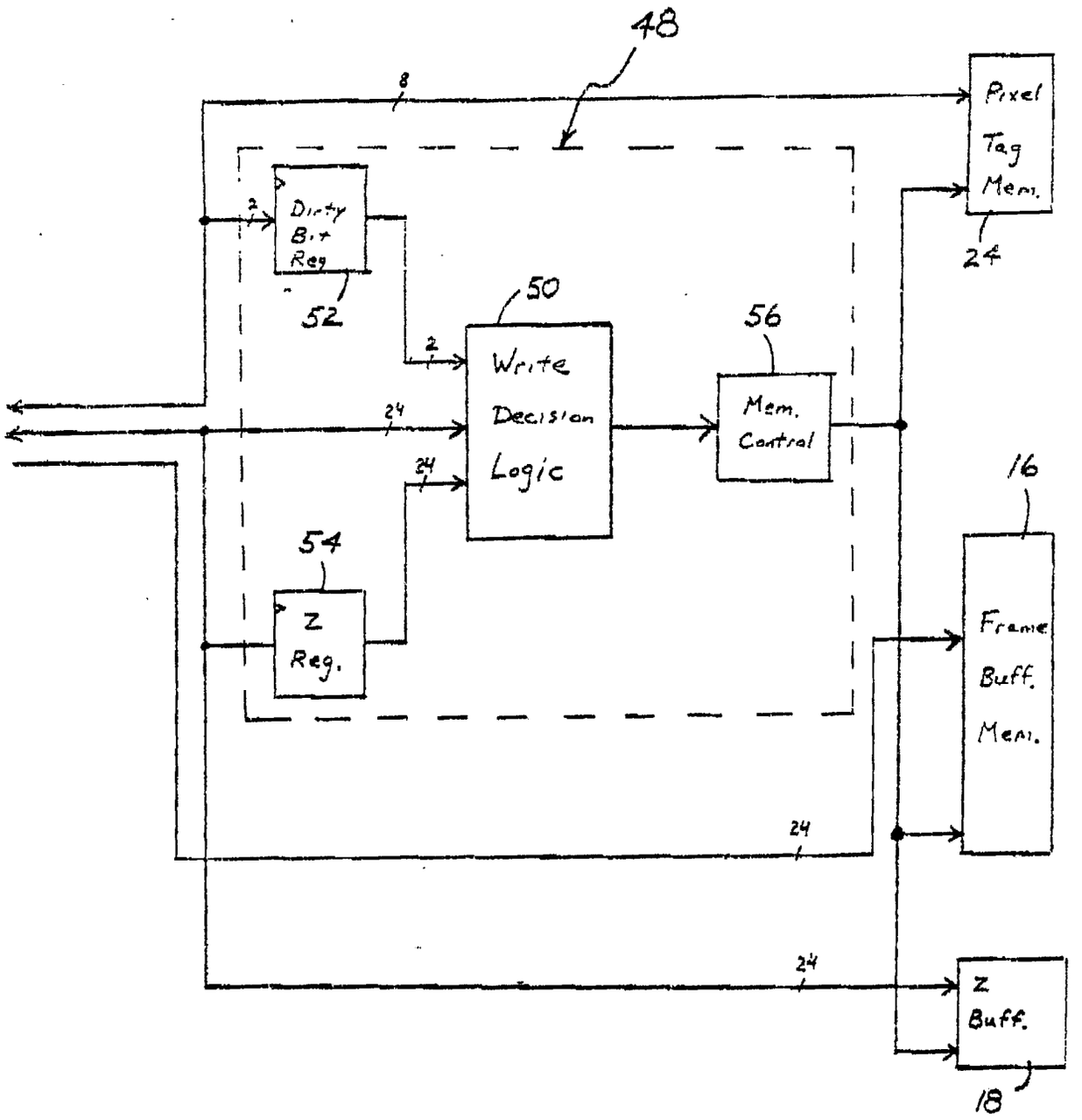

48
8
Pixel Tag Mem.
24
2
Dirty Bit Reg
52
50
Write Decision Logic
56
Mem. Control
24
24
54
Z Reg.
16
Frame Buff. Mem.
24
24
Z Buff.
18

FIGURE 6