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

(11) Publication number: **0 280 582 A2**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number: **88301742.8**

(22) Date of filing: **29.02.88**

(51) Int. Cl.^{4}: **G 09 G 1/00**

(30) Priority: **27.02.87 GB 8704653**
19.11.87 GB 8727119
18.01.88 GB 8801012

(43) Date of publication of application:
31.08.88 Bulletin 88/35

(84) Designated Contracting States:
AT BE CH DE ES FR GB GR IT LI LU NL SE

(71) Applicant: **CAPLIN CYBERNETICS CORPORATION LTD.**
Poplar Business Park Unit C26, 10 Prestons Road
London E14 9RL (GB)

(72) Inventor: **Phillips, David John**
284 Old Farm Avenue
Sidcup Kent DA15 8AR (GB)

(74) Representative: **Milhench, Howard Leslie et al**
R.G.C. Jenkins & Co. 26 Caxton Street
London SW1H 0RJ (GB)

(54) Improvements in computer graphics systems.

(57) A multi-tasking, multiple windowing computer graphics facility which does not place undue burdens upon the processing power of its host computer comprises a multi-planar memory to which data defining windows to be displayed in viewports on a display screen can be written, and a plurality of viewport controllers each individually associated with a respective one of the memory planes and capable of extracting data from each of a plurality of windows defined in its respective memory plane for effecting a corresponding display in any desired viewport location on the display screen. For the management of overlapping viewports, the window-viewport assignments which determine the display locations of viewports on the display screen include viewport priority indications which are taken into account in the reading out of data from the several planes of the multi-planar memory so that overlapping viewports are handled automatically and can appear transparent or opaque relative to one another as required. Windows can be created of any size and any number of planes deep (up to the available depth of the display memory), and opaque or transparent images can be moved freely around the display screen without requiring modification of the data in the memory by flexibly mapping windows in the display memory to viewports on the screen. A VRAM (video RAM) multi-planar memory is used and the viewport controllers are configured as modular VLSI components which can be cascaded for addressing any number of memory planes.

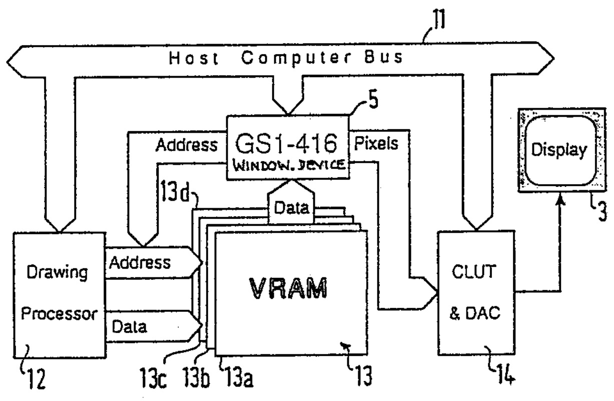

FIG. 3.

Bundesdruckerei Berlin

Description

IMPROVEMENTS IN COMPUTER GRAPHICS SYSTEMS

FIELD OF THE INVENTION

The present invention generally concerns improvements in or relating to computer graphics systems, and more particularly concerns window management systems and windowing devices for use with computer graphics workstations.

BACKGROUND OF THE INVENTION

In a conventional raster-scan computer graphics system, images to be displayed on a display screen are drawn into a reserved block of memory, sometimes called a frame buffer, within the system. Memory locations within the frame buffer have a one-to-one correspondence with display locations on the display screen and image data is read out sequentially from the frame buffer via video generation circuitry in accordance with the instantaneous address of the scanning beam on the display screen. This arrangement is adequate so long as there is no requirement to move complex multiple images rapidly around the display screen. However, as the requirements of graphics systems users have become more sophisticated there has been a corresponding increase in demand for complex images to be displayable and quickly relocatable on a screen.

The position and size of a displayed image may be controlled by the user, for example by way of a mouse which provides user determined positional and dimensional information. However, although a mouse can be moved swiftly across a table top by the user for repositioning a displayed image, in conventional graphics systems the image is commonly unable to keep up with the movement of the mouse. This is because of the one-to-one relationship which exists between the frame buffer and the display screen locations and because of the corresponding requirement that, in order to move an image from one part of the screen to another, it is necessary to copy the entire image data to new locations in memory and to replace the image data at the original memory locations by data corresponding to whatever was visually behind the image on the screen, whether it be a plain background or part of another image.

This apparently simple data handling operation does in fact give rise to major problems, since a multi-colour image of even moderate complexity may correspond to more than a million bits of data which have to be moved from one part of the frame buffer to other in order to move the image correspondingly. As a result, much of the available memory and bus bandwidth of conventional graphics systems has been occupied with shunting large blocks of data from one memory location to another and, in order to achieve this bulk data manipulation, a major software and processing overhead has been imposed on the host computer. In consequence, images have tended to move slowly or jerkily around the screen rather than moving smoothly around the screen in response to movement of a mouse or to other user originating commands. In addition, images cannot generally be made to appear and disappear instantaneously as and when the user so desires. Moreover, many desirable effects, such as transparency effects whereby one image can be seen behind another, have to be compromised since they present data manipulation problems which are virtually impossible to handle in real time except in the simplest of cases. Even though data processing devices have been developed which are capable of copying blocks of data from one area of memory to another at very high speed, the transfer rates required to overcome the above problems are still far beyond the reach of cost-effective contemporary technology.

The problem of moving multiple images about rapidly and/or smoothly on a display screen has been recognised by those skilled in the art of computer graphics, and it has been suggested that it should be possible to by-pass the limitations introduced by the data processing overheads by taking an alternative approach to image manipulation, this approach being generally referred to as hardware windowing.

Hardware windowing is a method of moving images around a display screen without the need to rearrange large blocks of image data within the memory. In principle the idea is simple. Image data in the frame buffer is transferred to the video generation circuitry in a controlled sequence such that any given area of memory, hereinafter referred to as a window, can be displayed at any chosen location on the display screen, such location hereinafter being referred to as a viewport. Hardware windowing is distinct from conventional serialised approaches wherein scanning through the display memory is effected sequentially as the display screen is scanned, since in hardware windowing a dedicated circuit is used to extract image data from the memory by jumping from place to place in memory as required. It then becomes easy to move any image instantly from one place to another, simply by specifying where it is to be displayed next.

However, although the concept of hardware windowing is well known, all attempts to build a workable graphics system based on this principle have hitherto suffered from a number of drawbacks resulting mainly from the inherent difficulty in performing the necessary logical operations on the image data at the extremely high video speeds required. Thus, designers of prior hardware windowing systems have had to limit display resolution, limit the number of viewports which can be simultaneously displayed on the screen, restrict viewport positions to particular locations on the screen, inhibit the ability to overlap viewports, or limit overlapping to non-transparent viewports. Furthermore, to avoid gross interference between window management and drawing activities, the hardware often has had to be designed to meet the particular requirements of a given drawing processor. Every hardware window management system previously available has suffered from some or all of these defects, and therefore the technique of hardware windowing has not hitherto been completely successfully implemented.

SUMMARY OF THE INVENTION

The present invention resides both in the concept of providing a multi-planar display memory wherein each plane of the memory has its own viewport controller for flexibly mapping windows from that memory plane to viewports of the display, and in the novel means hereinafter described for realising such a viewport controller which offers considerable advantages over the prior art. By virtue of such an arrangement, not only is the computational overhead required for managing multiple displays reduced by independent extraction of data from each plane of the memory, but also the difficulties hitherto experienced of managing overlapping viewports can very simply be overcome by the assignment of a viewport priority indication to each windows-viewport mapping and the provision of a means which takes account of the viewport priorities in the reading out of data from the several planes of the multi-planar memory.

As is described and explained hereinafter, the present invention overcomes all of the above-mentioned problems of prior art computer graphics systems, and uniquely provides a number of hitherto unachievable benefits to the art of window management.

The exemplary windowing device in accordance with the present invention which is described hereinafter is configured as a single VLSI component designed for use with planar display memory and to take advantage of the special capabilities of modern VRAM memory chips. It is modular and cascadable and can be used with almost any planar VRAM raster scan graphics system. It provides a means by which multiple windows can be displayed in viewports at high resolution with each viewport and window sized and located with perfect, single-pixel resolution. It handles overlapping viewports automatically and allows them to appear transparent or opaque relative to one another as required. It allows the user to create windows of any size, each any number of planes deep (up to the available depth of the display memory), and enables a user to move opaque or transparent images freely around the display screen without any modification of data in memory by flexibly mapping windows in the display memory to viewports on the screen.

A single windowing device as hereinafter described comprises a plurality of viewport controllers each capable of mapping a plurality of 1-bit windows from a respective plane of a multi-planar memory to viewports on the screen. In the exemplary embodiment that will be described hereinafter, each windowing device comprises four viewport controllers, each of which controls the manipulation of up to four windows on a single image plane. A plurality of such windowing devices can be cascaded to address any number of memory planes and enable any number of windows to be displayed in an arrangement which will hereinafter be referred to as a windowing device array. This highly modular architecture allows price and performance to be tailored precisely to the user's needs, in systems ranging from PCs to top-end graphics workstations.

The windowing device according to the invention treats the display memory as a collection of superimposed bit planes. A 1-bit window is any rectangular array of bits on one plane, and an n-bit window is a block of image data composed of n superimposed 1-bit windows of equal size and each drawn from a different plane. Each 1-bit window may be taken from any arbitrary location in its memory plane and displayed in any viewport location, and any combination of n planes is allowed. The conventional concept of a window corresponds in the system of the present invention to the restricted case of an n-bit window in which all the 1-bit components are at the same planar address (vertically aligned) and n is equal to the number of planes in the system. By removing this restriction, and arranging for n-bit window to be created simply by assigning n 1-bit windows located anywhere in memory to the same viewport with the same priority, the invention obtains great advantages in terms of performance and flexibility.

In the practice of the invention, to enable overlapping viewports to be handled each 1-bit window is assigned a priority by the user, the same priority being assigned to all n planar components of an n-bit window. When viewports overlap, a higher priority window is arranged to obscure a lower priority window. This obscuring is a visual effect only and does not involve the manipulation of the image data in memory, and the obscured area re-appears automatically and instantaneously when the overlapping viewport is moved or when the priorities are suitably altered. Equal priority windows from different planes are advantageously arranged to overlap transparently, and colour mixing effects within the overlap area can be determined at the choice of the user by appropriate setting up of a user programmable colour look-up table. When two windows of equal priority overlap, the 1-bit components which reside on different planes are combined to produce user-definable transparency effects. When equal-priority 1-bit windows share a common plane of origin, it is arranged that the window first displayed in a raster scan takes precedence and obscures any other overlapped window. Total transparency therefore occurs between any two multi-bit windows if they have equal priorities and occupy disjoint sets of planes.

A viewport of any shape can readily be created by using a 1-bit window as a mask. Double- (or multiple-) buffering of animated images is easily achieved by mapping windows to viewports sequentially; only the areas of the display memory which actually contain the buffered images are affected.

The locations in a display of viewports in which image windows are required to be displayed and the locations in memory of the windows themselves, are defined together in window-viewport assignments which are a set of absolute and relative co-ordinates on the display and in the memory. When using a windowing device in accordance with the present invention, it is necessary merely to specify in a single window-viewport assignment the location in memory of the planar components of the window and the screen Co-ordinates of the viewport in which it is to appear and the chosen image then appears instantly at the desired place on the screen where it either obscures or is obscured by any other windows which it may overlap, depending on whether its priority is higher or lower than theirs.

This simple method of defining windows and viewports makes programming of a graphics system in accordance with the present invention very straightforward, and abolishes the cumbersome software operations such as strip decomposition and Boolean computation of transparency effects which hinder other systems. The invention thus provides an extremely versatile, powerful system which enables complex visual effects to be produced effortlessly while making very efficient use of memory. All window boundary clipping and transparency effects are carried out automatically, in real time, with no user intervention. The windowing device is effectively invisible to the drawing processor (or whatever other device is being used to create bit maps in memory) and in general drawing proceeds at close to full speed under most circumstances even while the images being drawn are being moved all over the screen. The bus bandwidth required to create, relocate or remove a window is so small that the system overhead that this represents is in general negligible.

There is no theoretical limit to the number of windowing devices and therefore viewport controllers that a windowing device array can contain. It follows that there is no theoretical limit to the number of planes, windows and viewports which can be controlled by such an array. With an array implemented using VLSI devices, means can be provided for linking as many of the devices together in parallel as may be required in order to achieve the desired number of displayable viewports.

Other features of the invention are set forth with particularity in the appended claims and will be well understood from consideration of the following description of an exemplary embodiment of the invention which is illustrated in the accompanying drawings:

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1A shows an overview of a conventional computer raster graphics system which uses a serialiser to display memory contents precisely as they are stored, and Figure 1B is a similar showing of a raster graphics system in accordance with the present invention which enables selected areas of memory to be displayed instantly at any position on the display;

Figure 2 shows a number of examples of how images stored in a memory can be selectively displayed by use of a computer graphics system according to the present invention as shown in Figure 1B;

Figure 3 is a more detailed showing of a computer graphics system in accordance with the present invention and illustrates how the windowing device of the invention interacts with a multi-planar memory and a display under the control of a host computer;

Figure 4 shows a schematic functional block diagram of an exemplary windowing device according to the present invention as incorporated into the graphics system of Figure 3;

Figure 5 shows the detailed architecture of the windowing device of Figure 4;

Figure 6 shows how window and viewport positions are defined to the windowing device of the present invention by the host computer;

Figure 7A shows an exemplary display comprising three viewports originating from window data in a single memory plane and Figure 7B shows in detail the scanline A-A shown in the Figure 7A display;

Figures 8A, 8B and 8C show respectively the detailed structure of the merge logic component of the viewport controller as shown in Figure 5, and graphical representations illustrating an exemplary operation of the merge logic; and

Figure 9 shows how a larger plurality of memory planes may be controlled by a windowing device array consisting of a cascaded plurality of windowing devices in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Referring to Figure 1A of the drawings, a conventional raster graphics system might comprise a host computer 1 which provides overall management of the system, a display memory 2 for storing image data, a raster scan display screen 3 for displaying viewports, and a serialiser 4 which steps in strict progression through the display memory in raster fashion and reads data serially out from the memory as the display spot scans the display screen. The resulting display, as previously explained, is a one-to-one mapping of the image data onto the display screen so that images appear on the screen in the same relative positions as they are stored in memory. In order to move the images around the display, the corresponding data in the memory has to be rewritten to the desired new locations which, as has been previously explained, makes excessive use of the processing power of the host computer, particularly when overlapping effects are also required to be accommodated. In contrast to the prior art system of Figure 1A, the hardware windowing system of Figure 1B allows selected areas of memory to be displayed instantly at any desired display position. Under control of the host computer, the hardware circuit 5 performs the function of selectively extracting portions of image data from any location in the display memory 2 and transferring it for display at any desired location on the display screen 3 so that any given area of memory can be displayed at any chosen location on the screen. The host computer 1 can be arranged to draw image data directly into the display memory, or it can be used to control a separate drawing processor, not shown, which provides this function.

As is illustrated schematically in Figures 1A and 1B, the display memory can contain information relating to a number of different images, shown in the Figures exemplarily as a solid chevron 6, a bounded-W 7, and a chequered ball 8. In accordance with the teachings of the present invention, the data representative of these images can be stored in different areas of a single memory plane, or can be stored in different memory planes, or with some of the image data in one plane and other image data in one or more other planes. As will become apparent from the description hereinafter, the windowing device 5 of the present invention, under supervisory control of the host computer 1 which provides to the windowing device 5 window-viewport mapping assignments including information representative of the locations in memory where image data is stored in a window of the memory and the viewport locations whereat such image data is desired to be displayed and representative also of the priority to be accorded to the respective viewports for enabling overlapping viewports to be readily handled, extracts from the display memory 2 only image data appropriate to the required display and passes it to the display screen 3 for display as the information is required. The relative locations of the three images 6, 7 and 8 on the display screen 3 need bear no direct correspondence with the relative locations of the corresponding image data in the display memory 2 and can be varied as desired. Furthermore, image data of the three images has in the example of Figure 1B been read from the memory and displayed in a manner which causes portions of them to overlap on the display screen, the chevron 6 opaquely overlapping a portion of the chequered ball 8 and the bounded-W 7 and chequered ball 8 transparently sharing a common area of overlap. As will be described in greater detail hereinafter, the windowing device 18 of the present invention provides the logic and other functions required to ensure that only the required image data is read from the display memory and transferred to the display screen for display and that, in a case where images overlap transparently, overlapping data from the images is combined before being displayed on the display screen.

Figure 2 shows further examples of how the present invention enables image data stored in the display memory to be selectively extracted from memory and flexibly displayed on the display screen. In accordance with the teachings of the present invention, overlapping viewports are handled by storing image data in multiple memory planes and assigning priorities to window-viewport mapping instructions so that data is automatically read from memory in accordance with the priority assignments and when displayed automatically provides the desired overlapping display effect. The display 9 shown in Figure 2 is the same as that described with reference to Figure 1B and shows the features of opaque and transparent overlapping previously mentioned herein, the chevron 6 opaquely overlapping the chequered ball 8 which in turn transparently overlaps the bounded-W 7, such a display being obtainable in accordance with the teachings of the present invention by storing the bounded-W and the chevron in one display memory plane and the ball in another and setting the priority of the chevron at a high value and the priorities of the bounded-W and the ball at the same lower priority. As will be explained fully hereinafter, equal priority images read from different memory planes overlap transparently in the system of the invention. In the display 10 shown in Figure 2 the box 7a is overlapped and partially obscured by the chequered ball 8 and the W 7b overlies and partially obscures both the box 7a and the chequered ball 8, whilst the chevron 6 opaquely overlaps both the box 7a and the ball 8. The display 10 of Figure 2 might for example be obtained by storing image data representative of the chevron, the W, the ball and the box in different planes of a multi-planar memory and providing that the box has the lowest priority, the ball has a higher priority than the box but a lower priority than the W, and the chevron has a higher priority than either the box or the ball. In a colour display, display effects additional to those obtained by allocation of display images to selected planes of the display memory and by allocation of viewport priorities can be obtained by suitably programming a colour look-up table to which the video signals extracted from memory are applied before they are displayed.

In the exemplary embodiment hereinafter described, the windowing device 5 advantageously comprises four viewport controllers which share common areas of control and can each control up to four independent viewports and four associated image windows held in a single memory plane. A single windowing device 5 can thus control up to sixteen 1-bit windows held in four 1-bit deep memory planes to produce displays in up to sixteen viewports on a display screen. Multi-bit windows consisting of superimposed bits in the four bit planes can likewise be managed. The windowing device is adapted to be cascaded to form a windowing device array as aforementioned comprising a plurality of windowing devices arranged in parallel, and such an array can control the display of data from a large number of memory planes independently and can display up to four independent image windows from each plane of memory. The invention is not restricted to configuring the windowing device as comprising four viewport controllers, but by virtue of the use of VLSI technology a particularly advantageous chip configuration can be obtained with four viewport controllers on each chip without placing undue demands on the pin count and die size of the VLSI circuitry.

Referring now to Figure 3, there is shown therein a typical graphics system according to the present invention which incorporates a windowing device 5 comprising four independently operable viewport controllers. As shown, the windowing device 5 is adapted to be controlled by a host computer, not shown, via a host computer bus 11, and there is also provided a drawing processor 12 which draws or creates bit-maps in a planar video RAM 13 which is treated by the windowing device 5 as four superimposed bit planes each of which is individually associated with a specific one of the four viewport controllers within the windowing device. If the host computer has sufficient capacity, the drawing processor may optionally be excluded from the system in which case the host computer may be linked directly to the video RAM 13 so that it can draw directly into the memory. The planar video RAM, or VRAM, is shown as comprising four independent memory planes 13a-13d, and each memory plane can contain image data drawn into it by the drawing processor 12 and is arranged to be accessed independently by a specific one of the four viewport controllers within the windowing device. The viewport controllers operate to extract image data from the VRAM under control of the host computer and to transmit the extracted data to a user programmable colour look-up table and digital to analog converter (CLUT & DAC) 14 which control the display screen 3.

With the exception of the windowing device 5, the components shown in Figure 3 are obtainable as standard off-the-shelf components from a wide range of suppliers. For example the drawing processor 12 may be almost any processing device or system capable of creating image data in the form of bit-maps to be stored in the planar VRAM 13. However, a particularly effective planar drawing processor in this respect is the QPDM manufactured by Advanced Micro Devices (UK) Limited. Similarly, the choice of VRAM 13 and of CLUT & DAC 14 is not limited to specific makes or models. However, the embodiment of the invention herein described is optimised for use with the Brooktree range of RAM DAC's which are provided with a suitable programmable colour look-up table and in addition offer multiplexed pixel ports thereby avoiding the need for external multiplexing circuitry.

Figure 4 shows a functional block diagram of a windowing device 5 according to the invention, the device comprising four viewport controllers 15 only one of which is shown for the sake of clarity, together with functional blocks which are shared by all four viewport controllers within the windowing device, namely a counter 16 for synchronising the windowing device operation with the drawing processor and the raster display, a VRAM controller 17 responsive to the viewport controllers 15 for determining the extraction of image data from the VRAM 13, and global priority control logic 18 for controlling the overlapping display of viewports originating from memory planes controlled by different windowing devices of a windowing device array. The viewport controller 15 is shown in Figure 4 decomposed into its major subunits, namely a register file 19, a VRAM setup list (VSL) 20, merge logic and video data buffer 21, video control logic 22 and planar priority control logic 23, and the functions and the detailed construction of these subunits will be described in the following, it being remembered at all times that the windowing device of Figure 4 does in fact comprise four such viewport controllers 15.

In operation of each viewport controller, information identifying the locations in the VRAM display memory of the image windows and the viewport locations where the image windows are required to be displayed together with viewport priority information is downloaded into the register file 19 from the host computer via the host interface 24, it being recalled that each viewport controller 15 can control the display of up to four viewports. The information downloaded into the register file 19 is used to create a VRAM setup list which describes the actions to be performed by the VRAM controller 17 in extracting image data from the VRAM for display. The data output from the VRAM serial port is input a byte at a time to merge logic and video data buffer 21 which, under the control of information stored in the register file 19, shifts and merges the bytes of data which would each correspond to displays at a plurality of display pixels to achieve the required single pixel display accuracy before passing them to the video controller 22. The video controller 22 accepts the merged data and outputs it under control of the priority controller 23 for display. Video data is loaded from the VRAM into each of the four viewport controllers eight pixels at a time and is output to the display four pixels at a time, and the video controller takes care of this 8:4 multiplex operation; by virtue of this arrangement the maximum on-chip clock rate required is reduced to one quarter of the pixel rate, making the bandwidth required for high-resolution displays readily achievable using conventional CMOS fabrication.

The viewport priority information that is loaded into the register file 19 is utilized by the planar priority controller 23 to determine the display in overlapping viewports of image data extracted from that memory plane of the VRAM display memory which is associated with the respective viewport controller. The global priority controller 18 co-operates with the four planar priority controllers 23 within the windowing device in order to determine which viewport data originating from the four memory planes controlled by the windowing device should be passed to the display for a given display location. The global priority controller 18 also includes a priority bus interface which enables it to communicate over a priority bus 25 with other global priority controllers in similar windowing devices when several such windowing devices are connected in parallel to form a windowing device array as described hereinbefore. The global priority controller thus has the capability to determine the display of overlapping viewports corresponding to image data derived from different viewport controllers whether or not these are contained within the same windowing device.

In order to enable viewport controller 15 to share access to the VRAM address bus with the drawing processor 12, a handshake interface is provided between the VRAM controller 17 and the drawing processor 12 which requires the transmission between the VRAM controller and the drawing processor of video bus request ($\overline{\text{VBR}}$) and video bus grant ($\overline{\text{VBG}}$) control signals. This interface ensures that the viewport controller and the drawing processor do not both attempt to gain control of the VRAM at the same time.

The detailed architecture of the Figure 4 windowing device is shown in Figure 5. Each viewport controller contains three first in first out (FIFO) buffers, namely a VRAM setup list FIFO (VSL FIFO) 26 which together with test logic and VRAM setup list (VSL) compiler 27 and offset adder 28 the functions whereof will be explained hereinafter makes up the VRAM setup list 20 of Figure 4, a data FIFO 29 corresponding to the buffer between the merge logic 21 and the video controller 22, and a priority FIFO 30 which buffers priority information from the register file 19 to the planar priority controller 23. The three FIFO's taken together amount to less than 700 bits of buffering per viewport controller which is approximately one quarter of the storage that would be required with a serial line buffer approach. The four viewport controllers in the windowing device operate independently and only interact at their output stages where their outputs are synchronised so that identically defined viewports in separate controllers overlap precisely on the display.

For each viewport controller 15, each of the four available window-viewport assignments is defined by data downloaded from the host computer and held in registers within the register file 19. For each viewport controller the nth window-viewport assignment (n = 1 to 4) is fully defined by six co-ordinate parameters and a viewport priority parameter. As is shown graphically in Figure 6, the window-viewport co-ordinate definition comprises:

NAME	BITS	DESCRIPTION
$X_1(n)$	12	X co-ordinate of viewport left edge
$Y_1(n)$	12	Y co-ordinate of viewport top edge
$X_2(n)$	12	X co-ordinate of viewport right edge
$Y_2(n)$	12	Y co-ordinate of viewport bottom edge
$X_o(n)$	12	X-offset from Viewport address to window address
$Y_o(n)$	12	Y-offset from viewport address to window address
Pr(n)	3	Viewport priority

The viewport X_1, Y_1 and X_2, Y_2 co-ordinates define the screen locations of opposite corners of the viewport. The offset co-ordinates X_o, Y_o are added to the viewport X_1, Y_1 co-ordinates to provide the window start address in memory. The viewport priorities are defined by a 3-bit priority word which enables priority levels from 0 to 7 to be achieved, though it will be appreciated that this is exemplary and that greater or lesser numbers of priority levels can be provided for as desired. Figures 7A and 7B show respectively a possible arrangement of three 1-bit deep viewports V1, V2, V3 from a single memory plane where viewport V2 has been assigned a higher priority than viewport V1, and a scanline A-A of the display.

During each scan line, the viewport controllers operate independently and asynchronously except for their output stages where priority between planes is compared by the global priority controllers. The following description applies to the operation of one viewport controller only. The process of determining which pixel data is to be displayed in a given scan line begins in the preceding scan line time period, during which the VSL FIFO 26 is loaded with information from VSL compiler 27. During each scan line, the VSL compiler 27 thus loads the VRAM setup list into the VSL FIFO 26 in preparation for the following line. The VSL compiler 27 calculates the sequence in which viewport data is to be extracted from the VRAM. In order to achieve this, data in the appropriate form which is loaded into the register file 19 from the host computer is examined by the VSL compiler 27 to determine the viewport priority hierarchy for the memory plane in question. The VSL compiler makes no reference to the priorities of viewports on other planes.

Information loaded into the VSL FIFO 26 consists of an ordered sequence of entries, each entry being in the form (X_e, V#, S, P) where X_e is the X address at which the next visible viewport boundary occurs, V# is the number of the active viewport (1-4) at that boundary, S is a setup bit which signals whether or not a VRAM setup is required, that is to say whether the boundary in question is followed by window data (i.e. marks the start of a window) or by background (i.e. marks the end of a window), and P is a parity bit which signals whether VCOUNT is odd or even for the line in question. In the case of scan line A-A shown in Figure 7B the sequence of entries from the VSL compiler into the VSL FIFO would be as follows:

X_e	V#	S	P
$X_1(1)$	1	1	p
$X_1(2)$	2	1	p
$X_2(2)$	2	0	p
$X_1(3)$	3	1	p
$X_2(3)$	3	0	p

where each $X_i(n)$ is the relevant X-co-ordinate obtained from the register file 42.

As can be seen from Figure 7B the scan line has effectively been divided into a number of data fields. The first data field, from the leftmost side of the display to $X_1(1)$, is formed of background or default data since there is no viewport in this section of display. The second data field, between $X_1(1)$ and $X_1(2)$, is formed of viewport V1 pixel data. The third data field, from $X_1(2)$ to $X_2(2)$, can be broken down into two parts. The first part from $X_1(2)$ to $X_2(1)$ is common to both viewport V1 and viewport V2 but, since viewport V2 has a higher priority than viewport V1, the information associated with this part of the third field is exclusively viewport V2 pixel data. Had viewports V1 and V2 been of equal priority and from different image planes, the appropriate image data would have been extracted from both locations of memory and combined as required. However, since the images are both from the same memory plane they are not allowed to overlap transparently. The assignment of priorities to viewports is a matter for the user of the system, and in the event that two overlapping viewports from the same plane are assigned the same priority, the first viewport that is extracted from the memory plane will automatically take priority over and opaquely overlap the other. The second part of the third field contains only viewport V2 data since there is no overlap of images. The remaining fields of data are made up in a similar manner.

The hardware assembles this list of entries from the VSL compiler into the VSL FIFO for the scan line A-A of Figure 7B by working through the X edges defined in the register file 42 in order from left to right across the scan line in a series of logical passes. On the first pass, the logic tests each viewport to determine whether

$Y_1(n) \leqq Y_{A\text{-}A} \leqq Y2(n)$

indicating that viewport n is relevant to scanline A-A and identifies the relevant viewport which has the smallest value of $X_1(n)$, namely the viewport V1. Once the viewport V1 has been identified, a flag-bit is set noting the fact that the left (X_1) edge of this viewport has been passed. If the viewport has the highest priority of all viewports active at the boundary location tested, and it is not already the current display viewport, then the four fields of data X_e, V#, S and P above are loaded into the VSL FIFO 26.

Once the flag-bit has been set, the X_2 address of the viewport in question will be examined instead of the X_1 address during future passes so that the right edge, as seen on the screen, rather than the left edge will next be tested. Once both edges of the viewport have been detected, a second flag-bit is set which prevents that viewport from being subject to any testing in subsequent passes. Further similar passes are now made, examining X_1 or X_2 for each viewport or ignoring the viewport altogether as dictated by the set flag-bits. This continues until all viewports have been logically eliminated. In practice it is possible to accomplish the entire process of detecting viewports for a given scan line well within the time of a single scan line. The VSL FIFO buffer 26 is sufficiently large to buffer data for up to two consecutive scan lines, thereby enabling data for the next line to be loaded whilst the data for the current line is being read. The parity bit is provided in order to signal the transition to a new line since it toggles on each successive scan line.

On output, the contents of the VSL FIFO 26 is combined with offset data from the register file 19 and the result is used by the VRAM controller 17 to perform the correct sequence of VRAM setups for the scan line in question. For the scan line A-A, this process begins during the flyback period preceding scan line A-A. The VRAM controller 17 is a finite state machine and only one such VRAM controller is provided in the windowing device 5. In a windowing device array, all the VRAM controllers run synchronously. During the line flyback time before line A-A, any entry in the VRAM setup list in VSL FIFO 26 will cause the VRAM controller 17 to request control of the VRAM address bus from the drawing processor by asserting the not-VBR (Video Bus Request) signal. The VRAM controller is then loaded with a VRAM address (X_{VRAM}, Y_{VRAM}), where $X_{VRAM} = X_e + X_o$and $Y_{VRAM} = VCOUNT + Y_o$. When the drawing processor asserts not-VBG (Video Bus Grant) and releases control of the VRAM buses, the VRAM controller 17 sets up the VRAM serial ports on each memory plane to access data for the first viewport to appear from that memory plane. Once set up, each VRAM serial port is clocked until a number of words equal to the difference between successive values of X_e have been output. The next setup operation is then performed. The VRAM controller can simultaneously supply four setup addresses (one per memory plane). It determines which planes are set up on a particular cycle by controlling the not-RAS (Row Address Strobe) signal for each plane.

Image data is extracted from the VRAM in eight bit words and loaded into the merge logic 21. The VRAM is organised in 16-bit words and a 16:8 multiplexing operation is performed by alternately output-enabling the VRAM making up the high and low bytes. Data bytes output sequentially from the VRAM enter the merge logic 21 which, as is explained more fully below, shifts, masks and merges the data to single pixel resolution as specified by the values of the four bit quantities start-bit, stop-bit and offset-bit. These correspond to the least significant 4-bits of $X_1(n)$, $X_2(n)$ and $X_o(n)$ respectively, and define viewport boundaries and image offsets down to single pixel accuracy.

The operation of the merge logic 21 will now be described with reference to Figures 8A, 8B and 8C which show respectively a schematic block diagram of the merge logic circuit, a graphical depiction of the relationship between byte locations on the display and image bytes as fetched from memory, and a graphical representation of a corresponding sequence of merge logic operations.

In order to achieve single pixel resolution in the location of image data on the display and in the location of viewport boundaries, it is necessary to mask, shift and merge the image data bytes extracted from the VRAM memory before loading the data FIFO 29. An image in VRAM remains fixed, and hence the image data fetched by the viewport controller will not change. However, in order to be able to pan an image smoothly across the display, the data loaded into the data FIFO must change on a pixel by pixel basis as the viewport location on the display is changed. A simple example illustrating this situation is shown in Figure 8B where the upper row of the diagram represents byte locations on the display which must be filled exactly by the bytes loaded into the data FIFO 29, and the lower row represents image bytes as fetched from memory. In the example, a viewport starts at bit 2 of display byte D1, and the user has decreed that bit 6 of memory byte M1 should appear at this location. These bit locations are defined by the bottom three bits of X_e and X_o. These are 2 and 4 respectively. The merge logic must fetch image bytes and generate display bytes, masking off those bits which fall outside of the viewport boundary. Referring to Figure 8A, in this exemplary situation the merge logic operates as follows.

Memory byte M1 is loaded into input register 31 and is barrel shifted in barrel shifter 32 as defined by the bit offset, which is 4 in the case under consideration. Since the viewport starts in background, register 33 will be clear. The appropriate mask is applied to multiplexer 34 to select the low order two bits from register 33 and the high order six bits from the barrel shifter 32. The result is stored in register 33 by setting the control bit of multiplexer 35 appropriately, selecting the feedback path. As register 33 is loaded, register 31 receives memory byte M2 which is therefore also shifted. The mask applied to multiplexer 34 is now changed to select the most significant four bits from the barrel shifter 32 and the least significant four bits from the register 33. Multiplexer 35 control bit also changes to select the shifter output. This results in the first disply byte D1. The whole sequence is shown in Figure 8C for the exemplary situation of Figure 8B.

The merged data from the merge logic 21 is loaded into data FIFO 29 to achieve the necessary output synchronisation. The FIFO 29 is 8-bits wide by 32-bits deep and is equipped with a "FULL" output which halts the data coming in from the VRAM when necessary. The presence of the data FIFO 29 allows the viewport controller to start accessing the VRAM during the line flyback time, and buffers sufficient pixel data to enable a steady output of pixels to be passed to the display. It is important that this feature is included since the data stream from the VRAM into the merge logic is fragmented whenever the VRAM controller resets the VRAM serial port between windows.

On exit from the data FIFO, the data enters the video control logic 22 which multiplexes the 8-bit words into 4-bit subwords, controls the synchronisation of the data FIFO 29, and injects zero "background" data during intervals when no viewport is active so that the data FIFO 29 only has to buffer active pixel data.

At the same time as these video operations are taking place, viewport edge information X_e emerging from the VSL FIFO 26 is selectively loaded into the priority FIFO 30 together with appropriate priority data from the register file 19. This loading process is arranged so that priority values corresponding to the 4-bit video subword which contains a given viewport edge are merged into the row of the priority FIFO 30 corresponding to that edge. More particularly, as each VRAM set-up list entry is presented by the VSL FIFO 26, the value X_e is compared with the previous value of X_e at the top of the priority FIFO 30, and if it falls in a different subword the priority FIFO 30 is loaded. Associated with each X_e entry are four priority entries, one for each of the bits within the subword, which are loaded from the register file 19 at the same time as X_e. When two or more successive values of X_e fall within a common subword, the priority FIFO 30 is not shifted, but the "overlapped" priority values are updated. The result is a set of four priorities which will correspond precisely to the four display data bits at the video output. The output from the priority FIFO 30 is synchronised with the output from the video control logic 22 so that the correct four priority values are presented for each four-bit subword.

During each scan line, the value of HCOUNT is continuously compared with the X address at the top of the priority FIFO 30. When a match occurs, the priorities of the bits in a 4-bit subword are output to the planar priority control logic 23. This determines the highest priority at each of the four pixel locations so that bits from a viewport which does not have the highest priority at that instant can be masked as they are output from the video control logic 22. The priority FIFO 30 is shifted by one location at each match in order to present the next subword address for comparison. If, at a given 4-bit subword location, there is no match between HCOUNT and the X_e output of the priority FIFO 30, implying that there is no active boundary in that subword, the active priority will be that of the rightmost pixel of the last matching subword. In other words, the active priority during an active viewport transit remains what it was at the left edge of the viewport. Since the data FIFO 29 and the priority FIFO 30 are both loaded and read during a single scan line, only one of each is required.

The four priority values output from the priority FIFO 30 are passed via the planar priority controller 23 where they are compared with synchronised priority information from the other three viewport controllers within the windowing device. These are obtained from the global priority control logic 18 which can also incorporate priority information passed from other windowing devices in a multiple windowing device system via the priority bus. The priority control logic 23 gates each bit of the 4-bit subword emerging from the video control logic 22, enabling it only if its priority is the highest (or equal highest) offered at that pixel. The resultant four bits are now ready for output as the contribution of this plane to the four-pixel-wide video output bus.

In order to compare the priorities of competing windows at each pixel, the priority values are converted from their binary representation to thermometer code and a bitwise-OR is performed on the results. This operation is carried out in parallel for the four pixels which are output by each viewport controller in every cycle and yields a thermometer code value corresponding to the highest of the priorities being compared. The global priority controller performs a bitwise OR on the thermometer code values between planes and supplies each planar priority controller with the result.

As is shown in Figure 9 the planar organisation of the windowing device architecture makes it very straightforward to cascade several such devices as a windowing device array, each device controlling up to four memory planes in a multi-planar system. When more than one windowing device is used, it is necessary to determine the highest global priority overall at each pixel so that only the highest priority viewports are displayed. This is accomplished by way of a 28-bit priority bus compring seven bits for each of the four pixels. Connections to the priority bus are made via open-drain input/output pins, so that no external logic is required to perform the OR operation. In order for the bus to operate at the required speeds a pre-charge circuit is incorporated into each of the open drain outputs. The result is that each chip "sees" four priority words on the bus corresponding to the highest global priority at each of the four pixels. The priority bus runs at one quarter of the pixel rate, e.g. at 27.5MHz for 1280 × 1024 resolution at 60Hz refresh rate. It is the only linkage (apart from the standard timing signals) required in a multiple windowing device array.

From the foregoing description it will be seen that the present invention provides a hardware windowing system wherein each window is managed automatically in hardware and may be positioned, repositioned or removed instantaneously with no modification of data in memory. By use of one or more of the described windowing devices, a user can create, reposition and remove large numbers of viewports on a display screen, each viewport displaying any chosen window from the frame buffer. Each viewport may be any size and each window may be any number of bits deep depending upon the number of memory planes in the system. Viewports are completely independent and can overlap either transparently or opaquely. All boundary clipping and transparency effects are performed on the fly, and overlapping areas are automatically obscured, revealed or intermixed as required with no user intervention and no software or processing overhead.

Each of the windowing devices as hereinbefore described has the capability to control up to four planes of memory and up to four independent windows per memory plane. These may be stacked in any configuration ranging from sixteen 1-bit deep windows to four 4-bit deep windows or any combination in between. Multiple windowing devices may be cascaded to provide any number of windows, each any number of bits deep. For example, a sixteen plane system may be controlled by means of four of the described windowing devices to provide up to 64 1-bit text windows, or one 16-bit deep window with up to 48 1-bit text windows, or up to eight 8-bit deep windows, and so on. Window configuration is dynamically allocated and re-allocated as required.

The modularity and high performance of the described windowing device makes it a cost effective solution to image manipulation tasks in systems ranging from PC graphics cards to top-end workstations. It is compatible with most implementations of graphics standards such as GKS, CGI and DEGIS, and may be integrated with windowing protocols such as X windows and NeWS. Major applications of the invention include CAD, particularly for architecture, IC design and PCB layout, business graphics, drafting, medical imaging, cartography, and electronic publishing.

Whilst a specific and currently preferred embodiment of the invention has been described in the foregoing, it is to be appreciated that the described embodiment is exemplary in all respects and that many modifications, additions and variations could be made and will occur to those possessed of the relevant skills without departing from the spirit and scope of the present invention. For example, whilst the described windowing device comprises four viewport controllers each capable of processing four window-viewport assignments in a respective memory plane; the invention extends to a single such viewport controller and also extends to a viewport controller operating on the same principles but not adapted for the control of four viewports, the choice of four viewport controller per windowing device and four viewports controlled by each viewport controller being a judicious selection having regard to all of the operating requirements and constraints but otherwise being exemplary. The use of VRAM for the display memory is advantageous in view of the inherent suitability of VRAM for graphics applications, but the invention is not to be regarded as limited to the use of VRAM and an alternative embodiment could be constructed with DRAM (dynamic RAM) memory; such an alternative embodiment is in fact described in the Ph.D. thesis submitted by David J. Phillips, the inventor of the present invention, to the University of Kent at Canterbury, in England, in 1987 and forms the subject of British Patent application No. 8704653 from which the present application claims priority. Furthermore, whilst no mention has hitherto been made herein of the possibility of providing a zoom facility in the described windowing device, hardware zoom could readily be provided for each viewport by use of pixel replication techniques such as, for example, by variation of the pixel rate for replication in the X-direction and by displaying the same scan line more than once for replication in the Y-direction.

Claims

1. A hardware windowing system for use with a computer graphics facility for determining the transfer from a window in memory of image data to be displayed at any chosen viewport location of a display, the hardware windowing system comprising a plurality of viewport controllers each adapted to be individually associated with a single plane of a multi-planar memory for individually processing data transfer from one or more windows in its associated memory plane to viewports in the display.

2. A hardware windowing system as claimed in claim 1 wherein each of said plurality of viewport controllers is arranged to be capable of processing data transfer from a plurality of windows in its associated memory plane to viewports in the display, and wherein overlapping viewports are handled by arranging the viewport controllers to be responsive to viewport priority information included in window-viewport mapping assignments provided to the viewport controllers for selectively determining the transfer of image data out of their respective memory planes in accordance with corresponding viewport priority information.

3. A hardware windowing system as claimed in claim 2 wherein priority control means are provided responsive to the viewport priority information included in the window-viewport mapping assignments provided to the plurality of viewport controllers for determining the display in overlapping viewports of image data from plural memory planes.

4. A hardware windowing system as claimed in claim 3 wherein said plurality of viewport controllers are included in a modular cascadable windowing device whereof the priority control means is adapted for communication over a priority bus with the priority control means of one or more other such windowing devices interconnected with the first-mentioned windowing device in a windowing device array.

5. A hardware windowing system as claimed in claim 4 wherein the priority control means of said windowing device comprises a planar priority control means for handling overlapping viewports deriving from image data stored in windows of the memory planes associated with and controlled by the said windowing device, and a global priority control means coupled with said planar priority control means and adapted to communicate over said priority bus with one or more global control means of other such windowing devices cascaded with the first-mentioned windowing device, the said global priority control means being adapted for handling overlapping viewports deriving from image data stored in windows of memory planes associated with and controlled by different ones of the plurality of cascaded windowing devices.

6. A hardware windowing system as claimed in any of the preceding claims which is adapted for use with a raster scan computer graphics facility wherein window-viewport mapping assignments are available from a host processor for identifying to the system the required viewport locations of the display and the corresponding window locations in the multi-planar memory, said window-viewport mapping assignments including scan line co-ordinate information relating to viewport and window boundaries and each said viewport controller including means for receiving such viewport boundary scan line co-ordinate information from the host processor on a line-by-line basis, and means responsive to the received viewport boundary scan line co-ordinate information for defining a corresponding address list for use in extracting image data from memory for display.

7. A hardware windowing system as claimed in claim 6 wherein said window-viewport mapping assignments include for each viewport X_1 and Y_1 co-ordinates of the top left-hand corner of a rectangular viewport, X_2 and Y_2 co-ordinates of the bottom right-hand corner of the viewport, and Xo and Yo offsets indicative of the X and Y offsets of the corresponding corners of a corresponding window of a memory plane, and said means for defining said address list for use in extracting image data from memory for display is adapted to derive from such co-ordinate information window start addresses in memory and controls for starting and stopping the clocking out of image data from the appropriate memory locations for each scan line of the display.

8. A hardware windowing system as claimed in claim 7 wherein the window-viewport mapping assignments further include a viewport priority allocation, and said means for defining said address list is adapted to derive said window start addresses and said starting and stopping controls in dependence upon said priority allocations in the event of viewports overlapping on the display, the viewport priorities determining which viewport boundaries will predominate in the display.

9. A hardware windowing system as claimed in claim 7 or 8 wherein image data is arranged to be clocked out of memory in multi-bit words each corresponding to a plurality of pixels of the display, and the viewport controllers each include logic means for processing the data clocked out of memory so as to obtain single pixel viewport location accuracy in the display.

10. A hardware windowing system as claimed in claim 8 or 9 and wherein a memory controller is associated with said plurality of viewport controllers and is arranged to be responsive to said controls for organising the extraction of image data from said memory, the arrangement being such that once started the clocking out of image data proceeds until a stop control is received and in between times the memory is accessible for use by a drawing processor for entering fresh image data into memory.

11. A hardware windowing system as claimed in claim 10 wherein said memory controller includes a handshake interface for enabling co-operation with a drawing processor for time shared utilisation of the memory.

12. A hardware windowing system as claimed in any of the preceding claims wherein the memory comprises a multi-planar VRAM memory wherein image data is adapted to be stored in multi-bit words each corresponding to a plurality of display pixels, and the viewport controllers are adapted to operate at a speed which is a fraction of the pixel rate of the display but to output a plurality of pixels at a time, the arrangement thereby providing more time for the processing of image data by the viewport controllers without prejudice to the integrity of the display.

13. A hardware windowing system as claimed in any of the preceding claims adapted for use with a raster scan computer graphics facility and wherein said viewport controllers are adapted and arranged to process the image data required to produce each scan line of the display during the period of the immediately preceding scan line.

14. A hardware windowing system as claimed in any of the preceding claims wherein four said viewport controllers each capable of independently controlling the display of four windows in a single plane of a multi-planar VRAM memory are assembled together as a VLSI chip, and a single VRAM controller is provided on the chip and serves to provide a control interface between the four viewport controllers and the VRAM memory.

15. A viewport controller for use in a hardware windowing computer graphics facility for controlling the transfer of image data from a window in a planar memory to a viewport of a raster scan display, said viewport controller being adapted to be responsive to window-viewport mapping assignments provided thereto from a host computer for flexibly mapping windows in memory to viewports of the display and said window-viewport mapping assignments including co-ordinate data X_1, Y_1 and X_2, Y_2 representative of the desired locations in the display of opposed corners of a rectangular viewport and offset data Xo, Yo representative of the offsets in the X and Y directions of the actual locations in memory of the corresponding corners of the window that is required to be displayed in the said viewport from locations defined in memory by the co-ordinate data X_1, Y_1 and X_2, Y_2.

16. A viewport controller as claimed in claim 15 and comprising a register for receiving such window-viewport mapping assignments from the host computer, a compiler for processing the data received by the register and deriving therefrom boundary address and associated control signals for each scan line of the display, and a memory controller responsive to the output of the compiler for determining the appropriate extraction of image data from the memory for each scan line of the display.

17. A viewport controller as claimed in claim 16 wherein said window-viewport mapping assignments further include viewport priority designations for use in the display of overlapping viewports, and said compiler is adapted and arranged to take account of said viewport priority designations in the derivation of said viewport boundary address signals.

18. A viewport controller as claimed in claim 16 or 17 wherein said memory comprises a VRAM memory, said compiler comprises a VRAM setup list compiler, and said controller comprises a VRAM controller.

19. A viewport controller as claimed in claim 18 wherein said VRAM controller includes a handshaking interface for determining the sharing of the VRAM memory address and control busses with a drawing processor.

20. A viewport controller as claimed in claim 18 or 19 wherein the image data is arranged to be clocked out of the VRAM under control of the VRAM controller in the form of data words each corresponding to a plurality of display pixels, and the viewport controller includes means for logically processing the data words output from the VRAM so as to obtain single pixel display accuracy.

21. A viewport controller as claimed in any of claims 15 to 20 which is adapted and arranged to operate at a pixel rate which is a fraction of the pixel display rate, for providing more time for the internal operations of the viewport controller to be performed, and wherein pixel data is arranged to be output from the viewport controller for display a plurality of pixels at a time to compensate for the reduced pixel rate of the internal operations of the viewport controller.

22. A viewport controller as claimed in any of claims 15 to 21 which is formed as a VLSI circuit chip and wherein a plurality of such viewport controllers are formed on the chip.

25.02.88

0280582

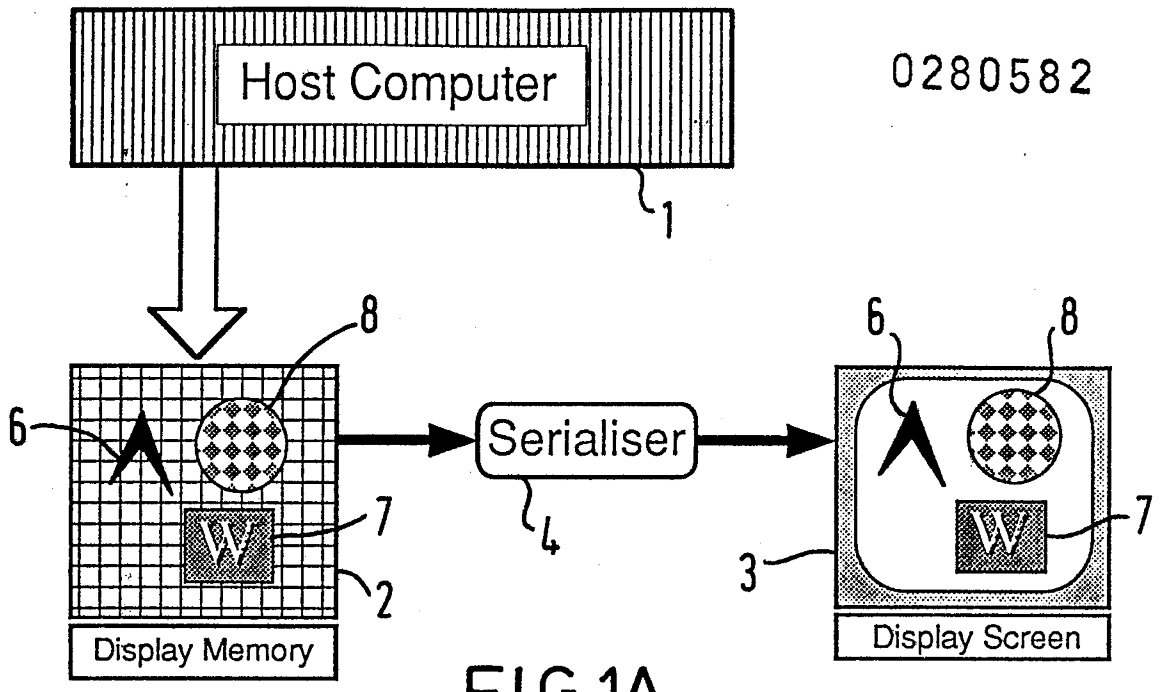

FIG.1A.

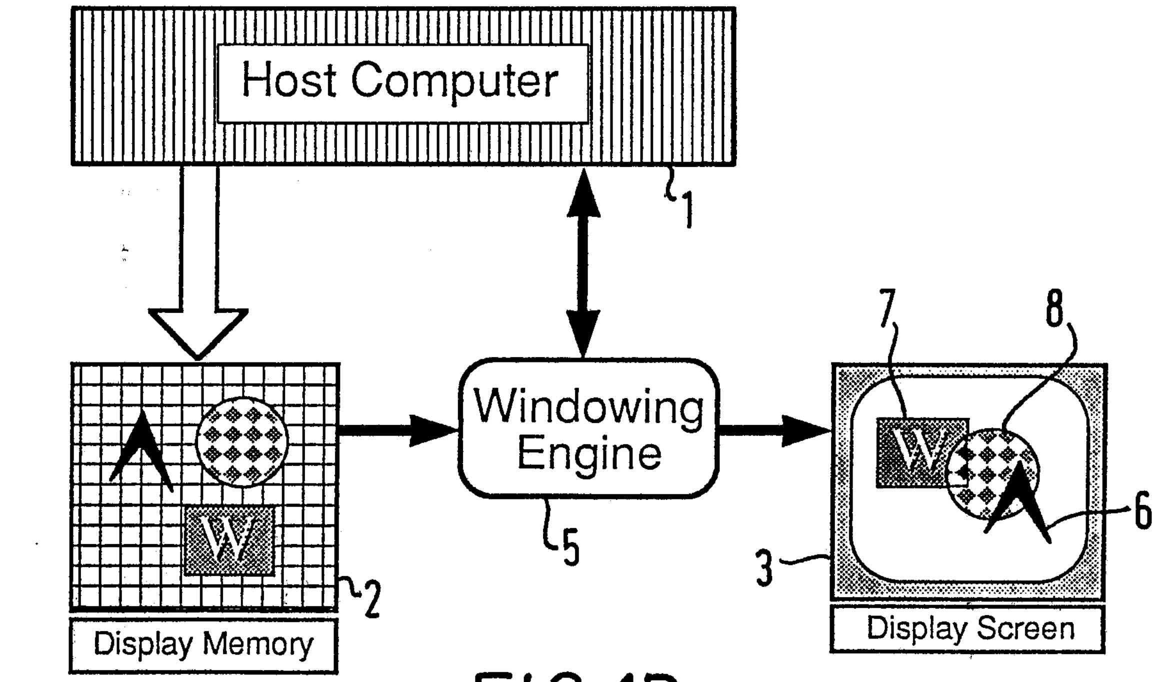

FIG.1B.

25·03·88

0280582

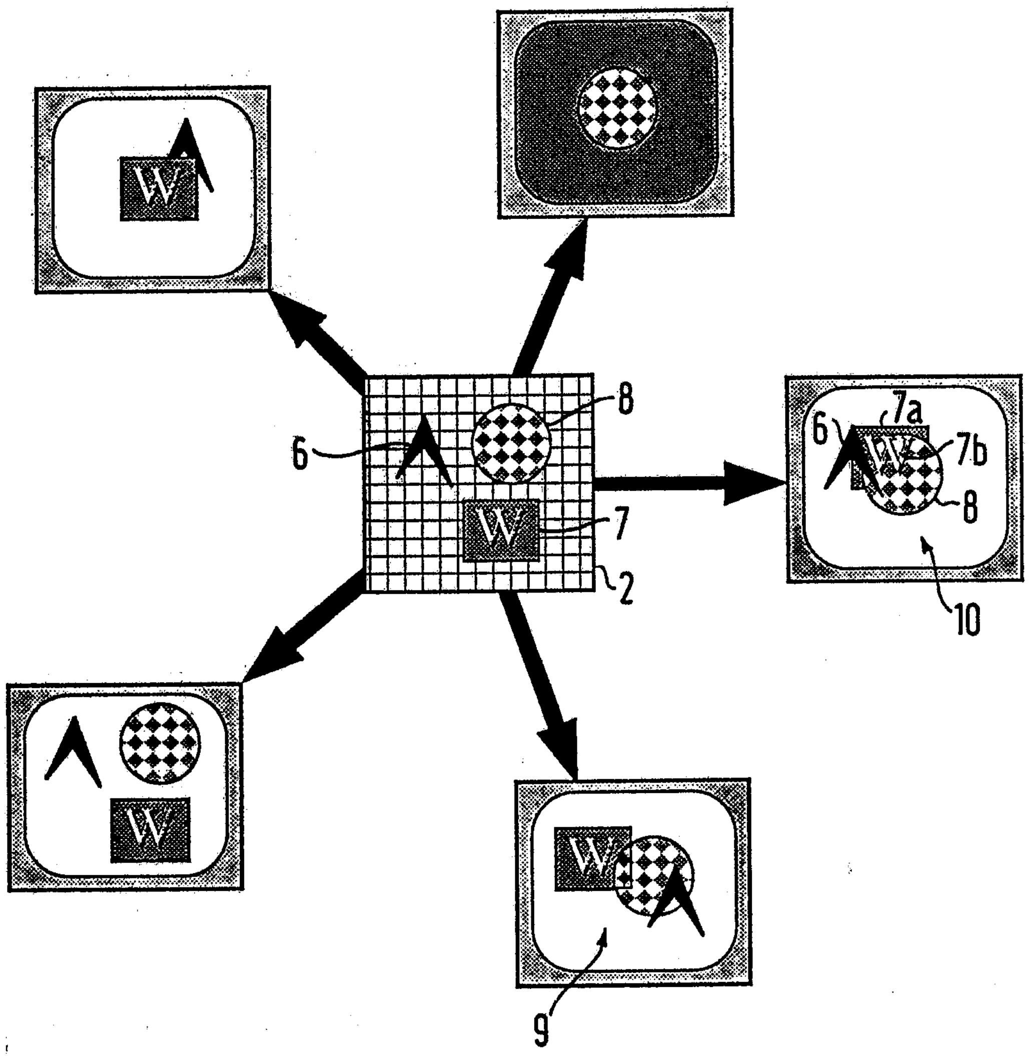

FIG. 2.

25·03·88

0280582

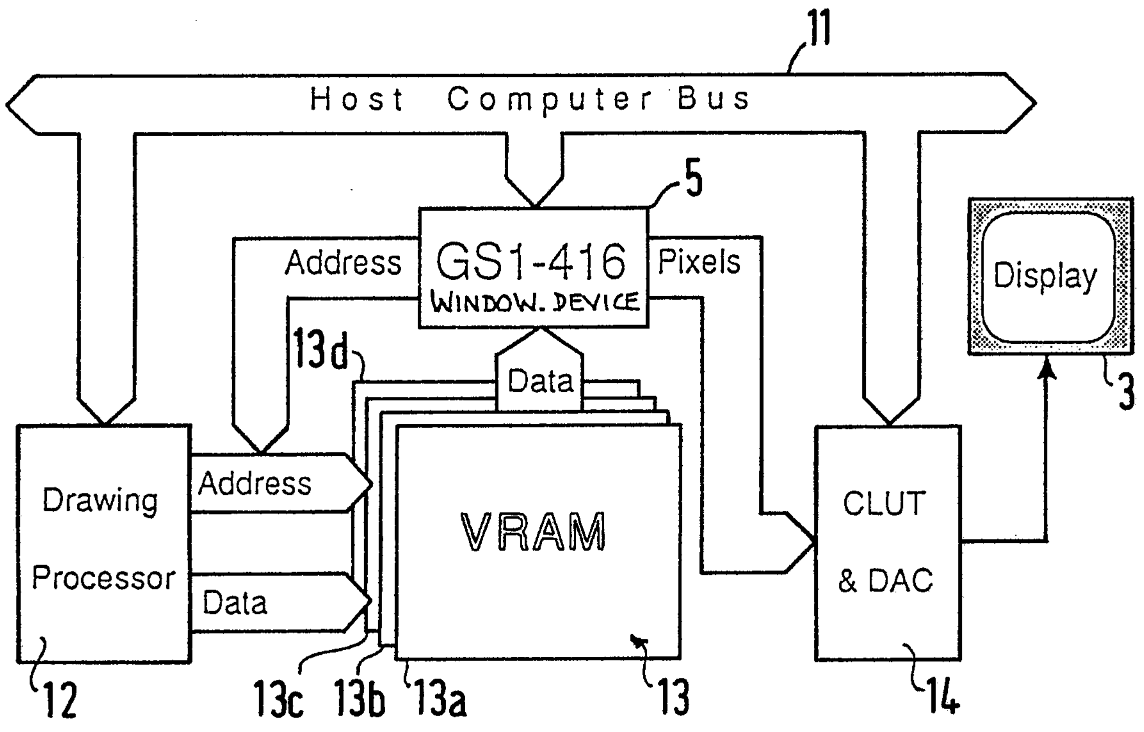

FIG. 3.

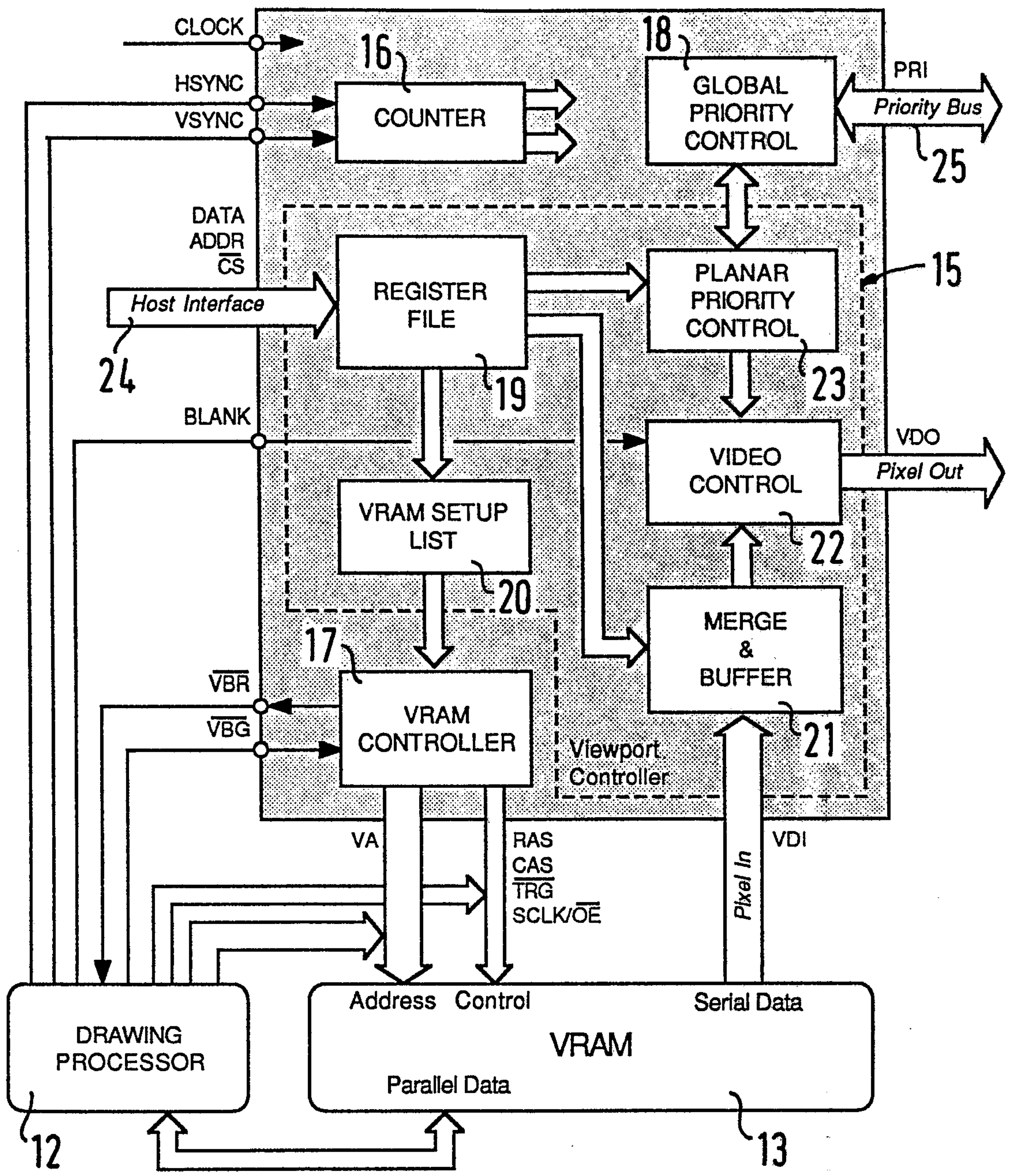
CLOCK
HSYNC
VSYNC
16
COUNTER
18
GLOBAL PRIORITY CONTROL
PRI
Priority Bus
25
DATA
ADDR
$\overline{CS}$
Host Interface
24
REGISTER FILE
19
PLANAR PRIORITY CONTROL
23
15
BLANK
VIDEO CONTROL
22
VDO
Pixel Out
VRAM SETUP LIST
20
17
MERGE & BUFFER
21
$\overline{VBR}$
$\overline{VBG}$
VRAM CONTROLLER
Viewport Controller
VA
RAS
CAS
$\overline{TRG}$
SCLK/$\overline{OE}$
VDI
Pixel In
DRAWING PROCESSOR
12
Address
Control
Serial Data
VRAM
Parallel Data
13

FIG. 4.

0280582

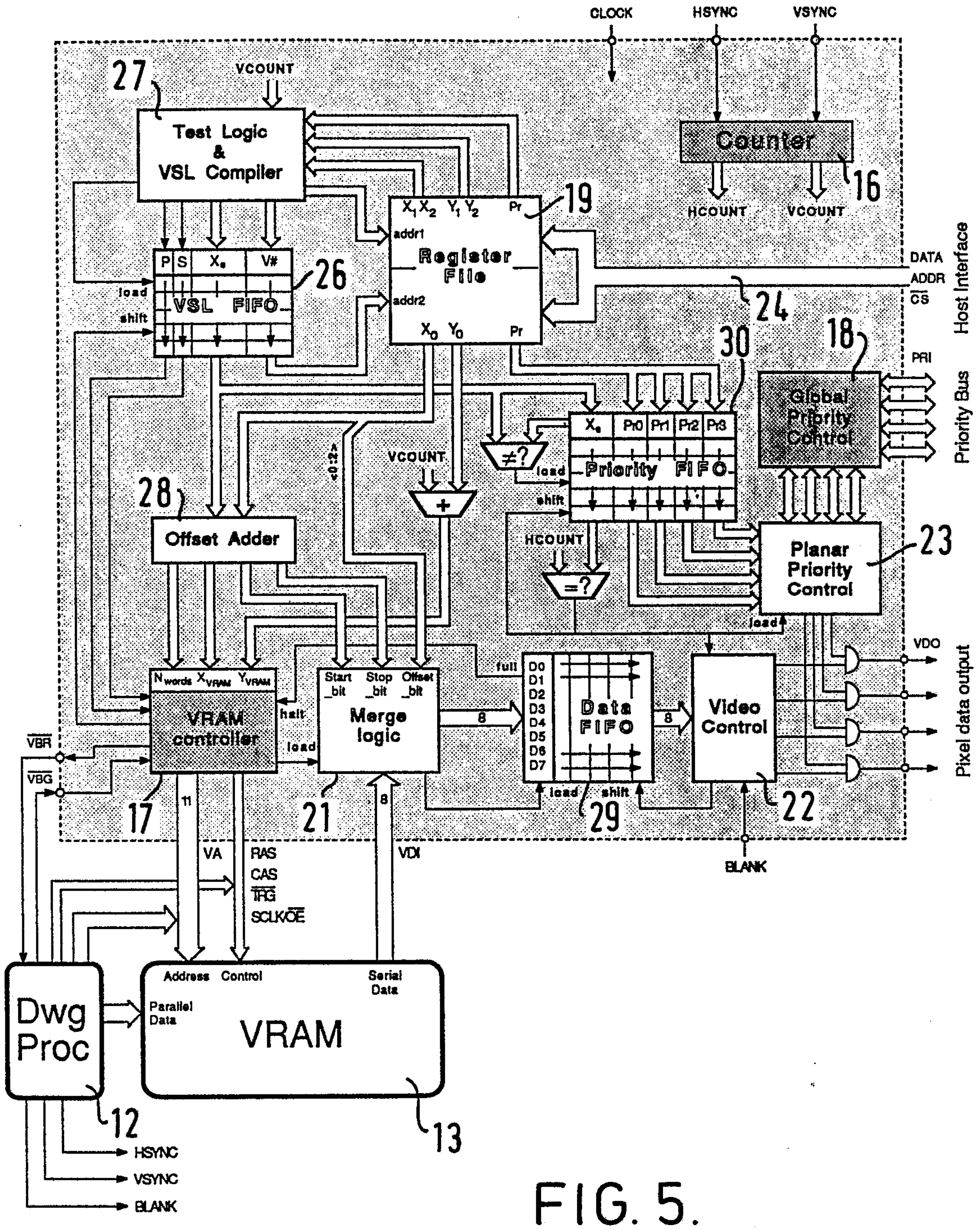

FIG. 5.

25.03.88

0280582

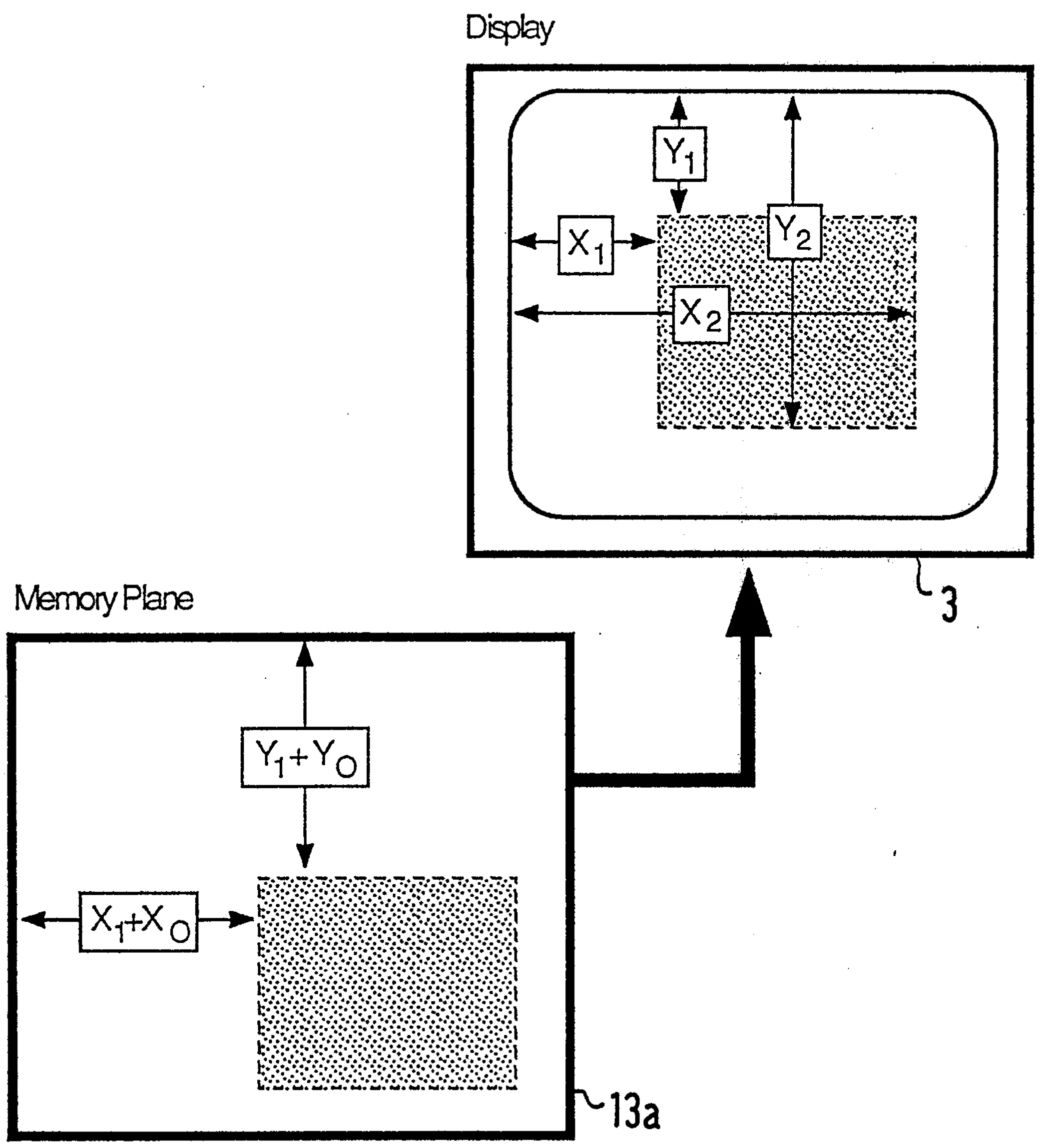

FIG. 6.

0280582

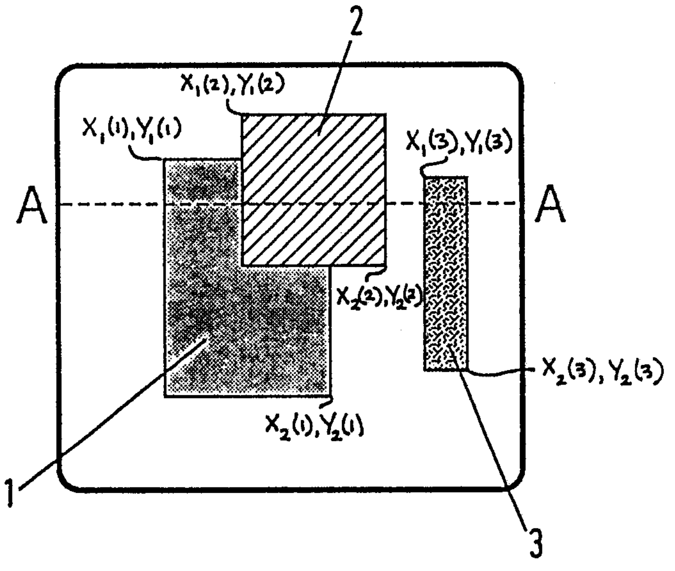

FIG.7A.

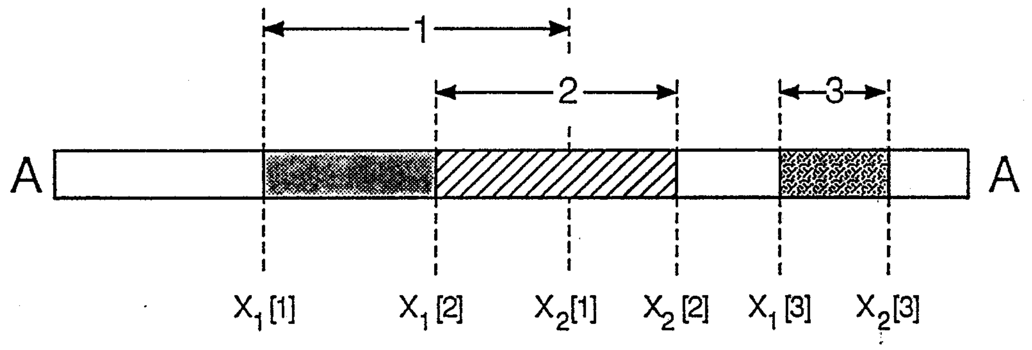

FIG.7B.

0280582

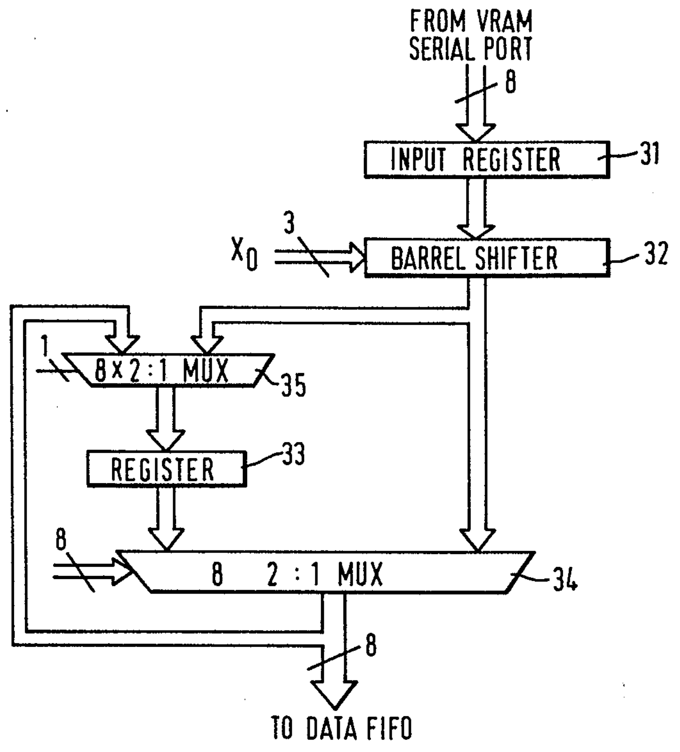

FIG. 8A.

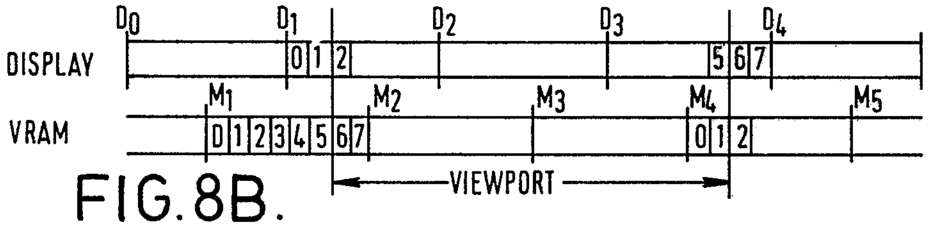

FIG. 8B.

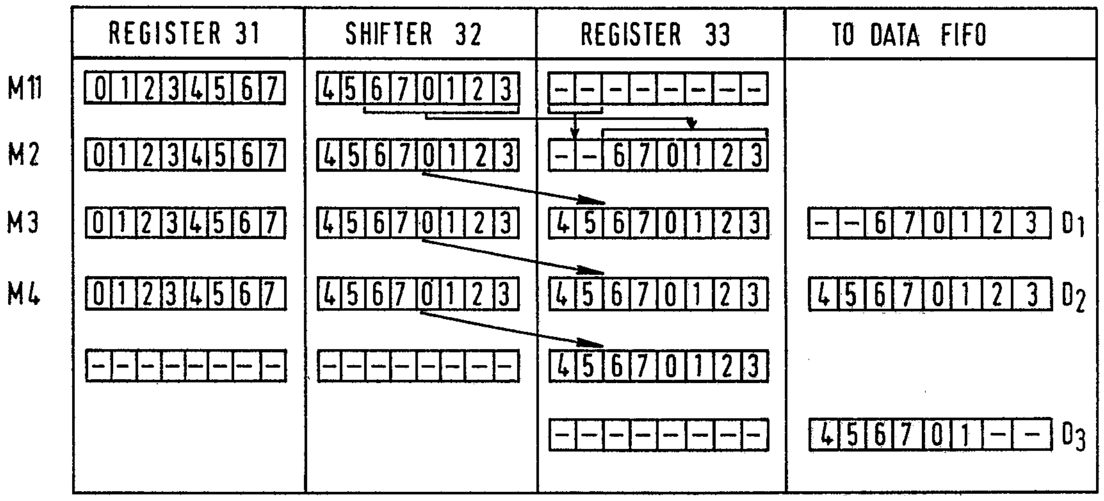

FIG. 8C.

0280582

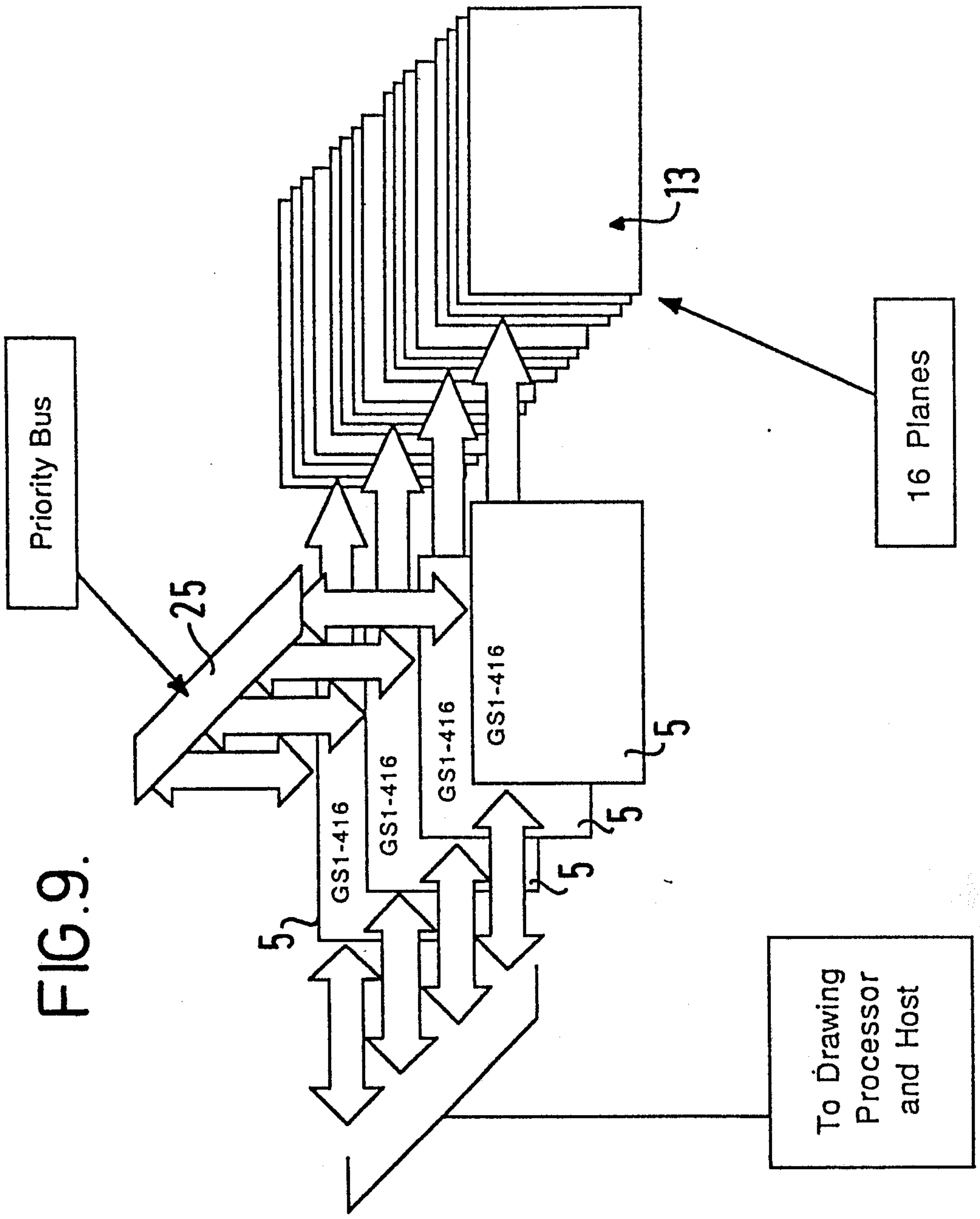

FIG.9.