US 20140285502A1

(19) **United States**

(12) **Patent Application Publication**

Hobbs

(10) **Pub. No.: US 2014/0285502 A1**

(43) **Pub. Date: Sep. 25, 2014**

(54) **GPU AND ENCODING APPARATUS FOR VIRTUAL MACHINE ENVIRONMENTS**

(71) Applicant: **Teradici Corporation**, Burnaby (CA)

(72) Inventor: **David V. Hobbs**, Surrey (CA)

(21) Appl. No.: **14/298,335**

(22) Filed: **Jun. 6, 2014**

Related U.S. Application Data

(63) Continuation of application No. 11/278,128, filed on Mar. 30, 2006, now Pat. No. 8,766,993.

(60) Provisional application No. 60/669,178, filed on Apr. 6, 2005.

Publication Classification

(51) **Int. Cl.**

G06F 3/14 (2006.01)

G06T 9/00 (2006.01)

(52) **U.S. Cl.**

CPC ***G06F 3/14*** (2013.01); ***G06T 9/00*** (2013.01)

USPC **345/520**

(57) **ABSTRACT**

A system encodes an image for a remote client. A graphics processor unit (GPU) renders an image in response to graphics commands received from a central processing unit (CPU) virtual machine. Image attributes are determined by display requirements from the remote client. A display encoder is associated with the GPU, encodes the image, and operates independent of the rendering of the image. A network interface may be associated with the CPU, GPU or display encoder and may transmit the encoded image across a network to the remote client.

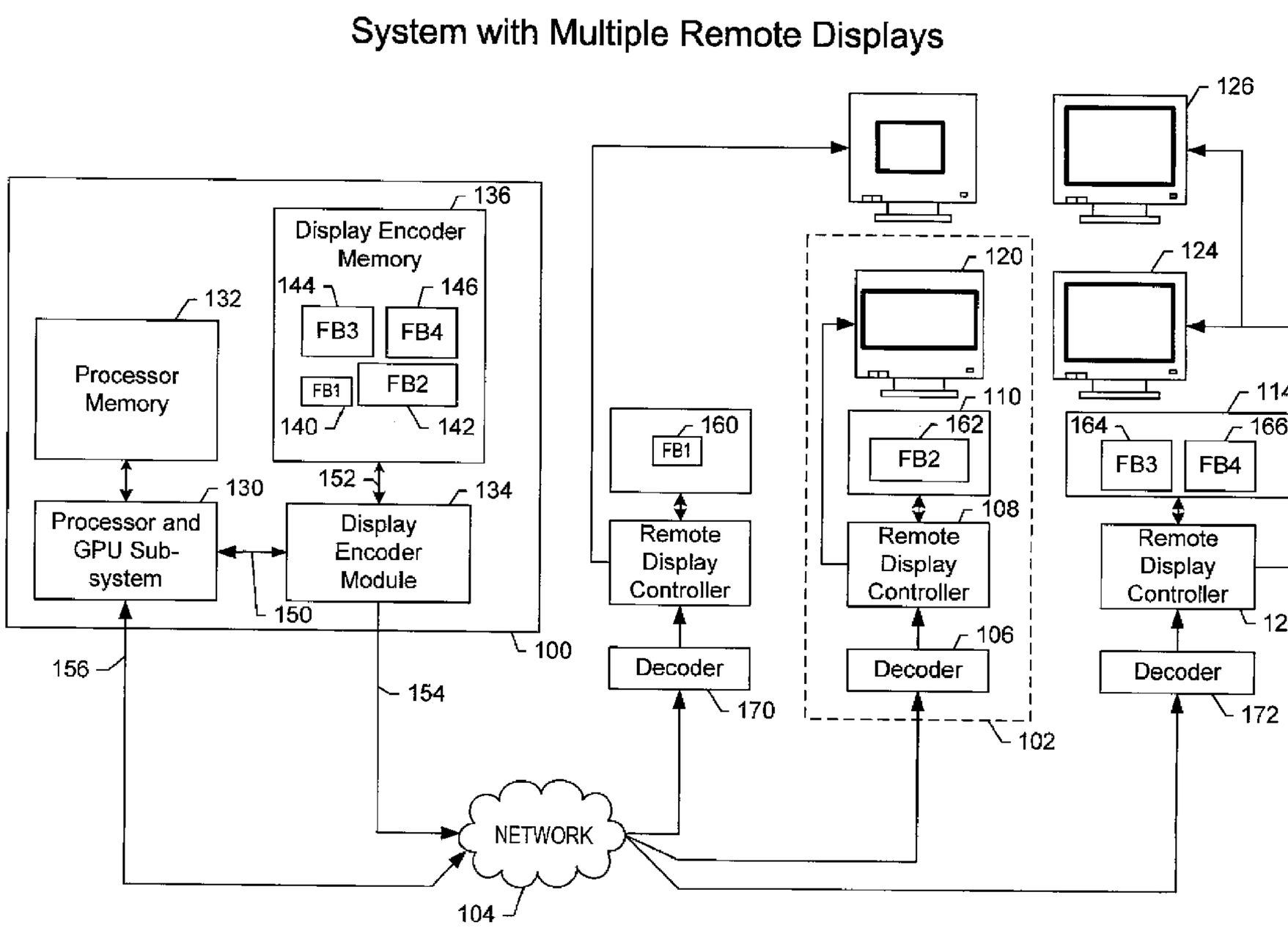


FIG. 1
System with Multiple Remote Displays

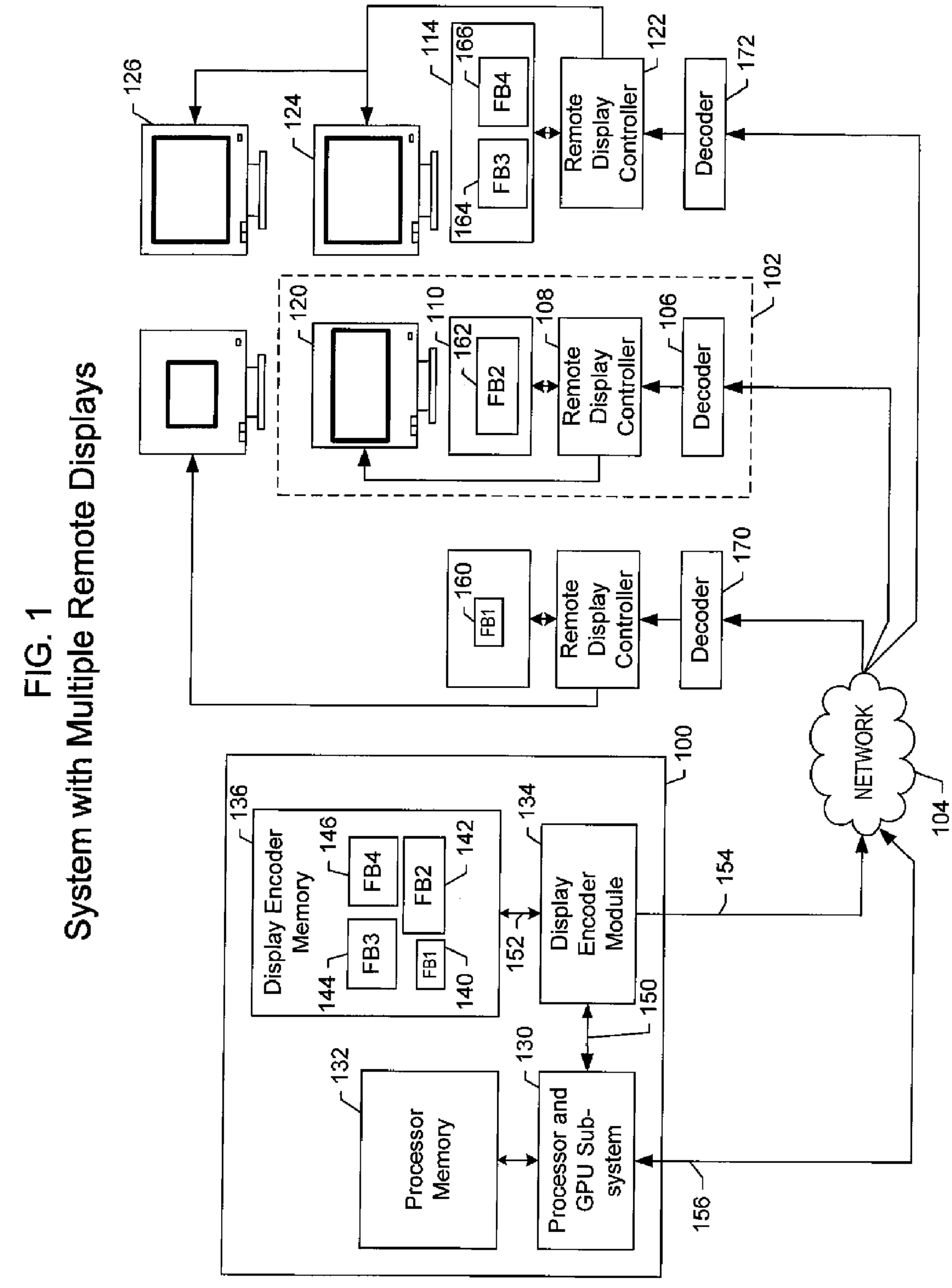


FIG. 2
Host Display Encoder

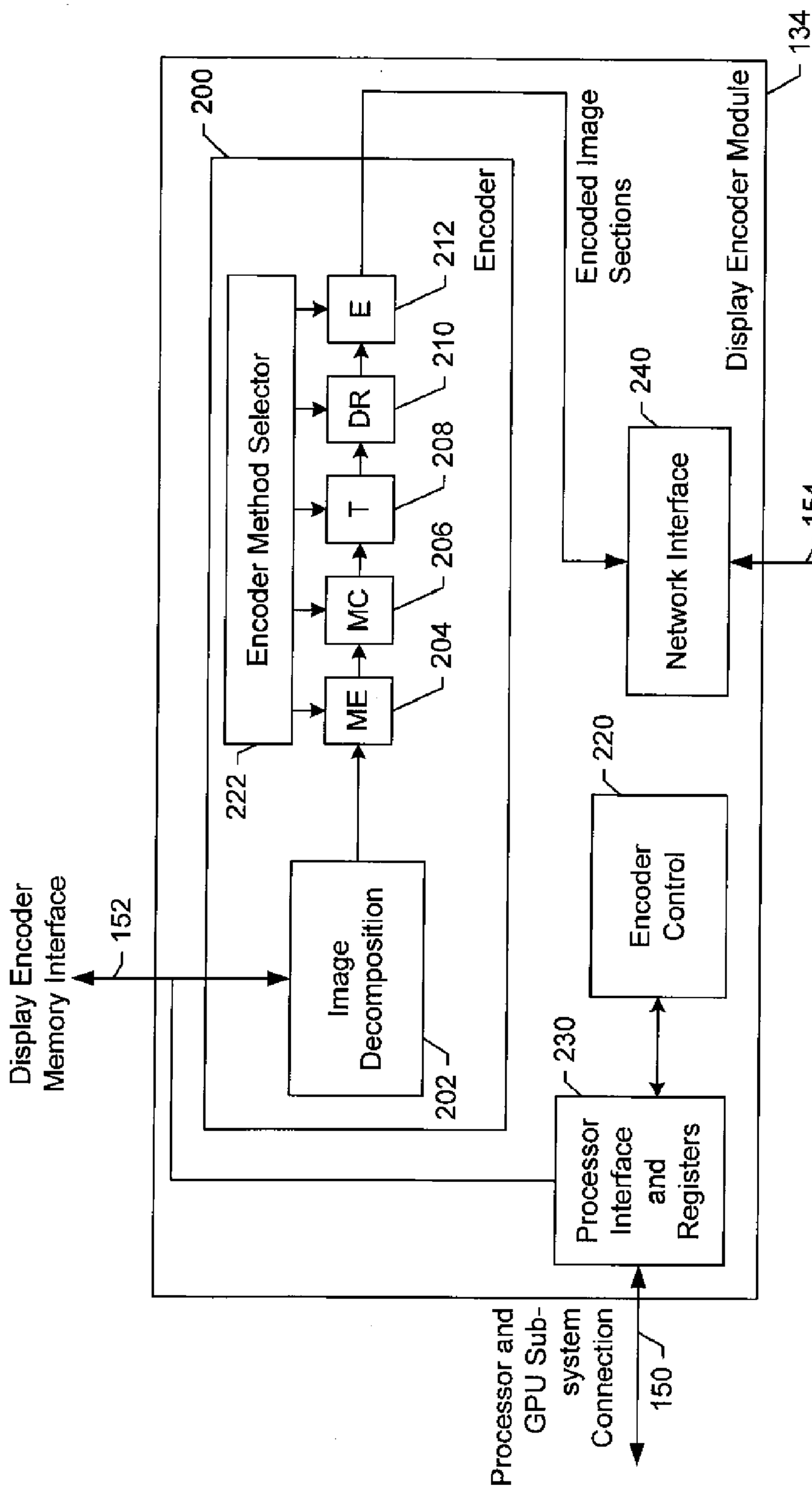


FIG. 3
Display Encoder on System Bus

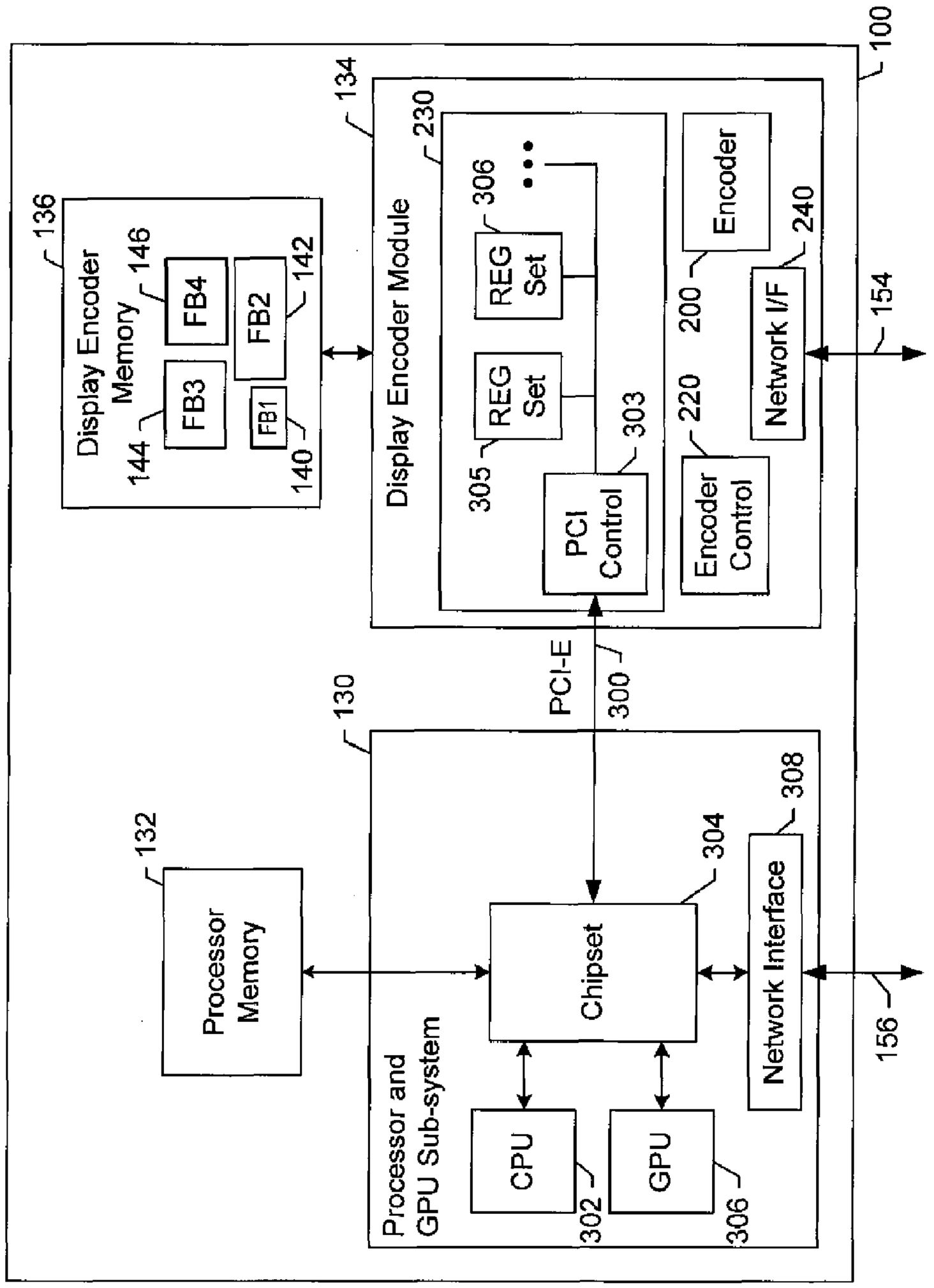


FIG. 4
Display Encoder on Image Data Bus

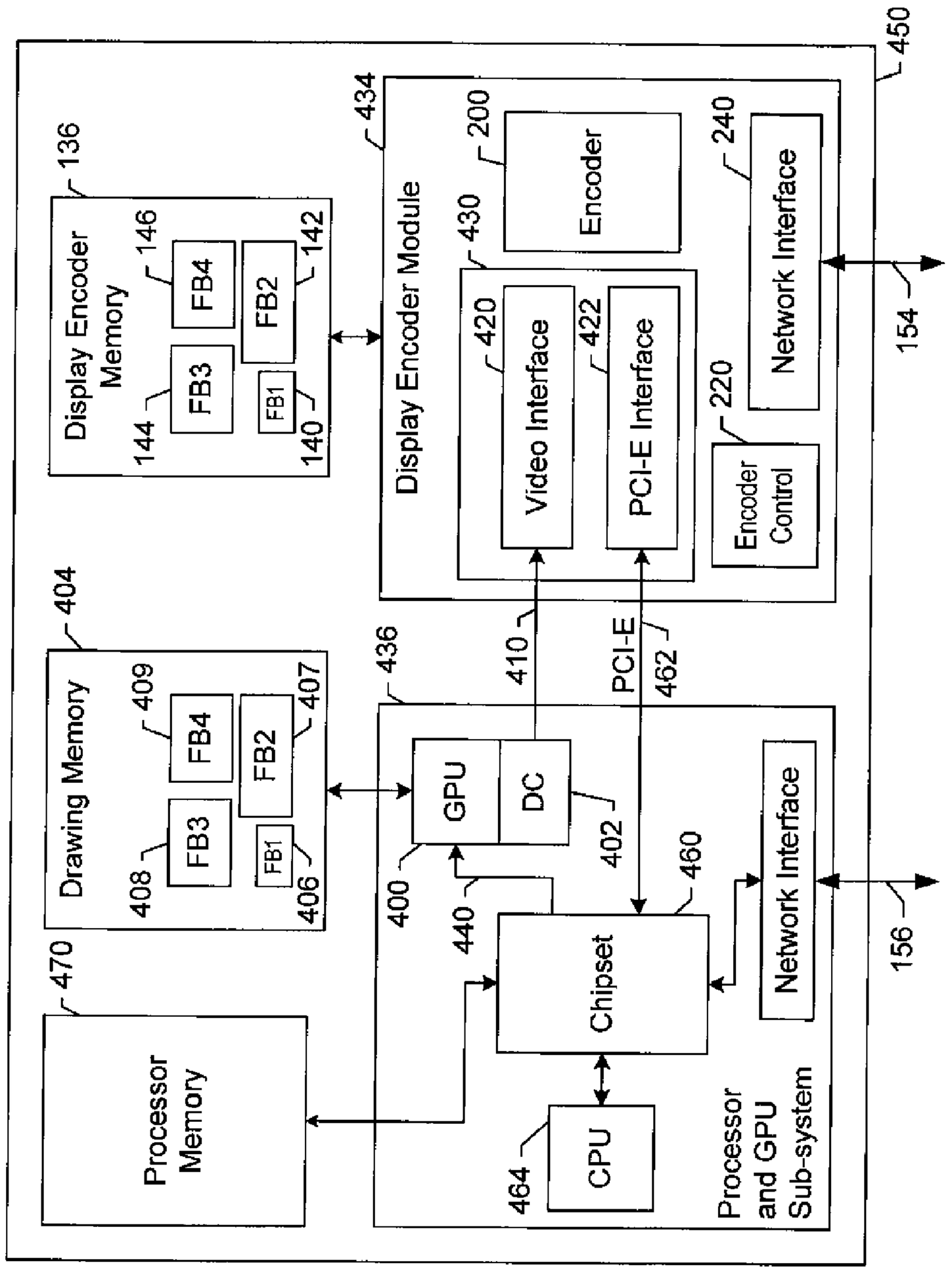


FIG. 5
Wide Display System Architecture

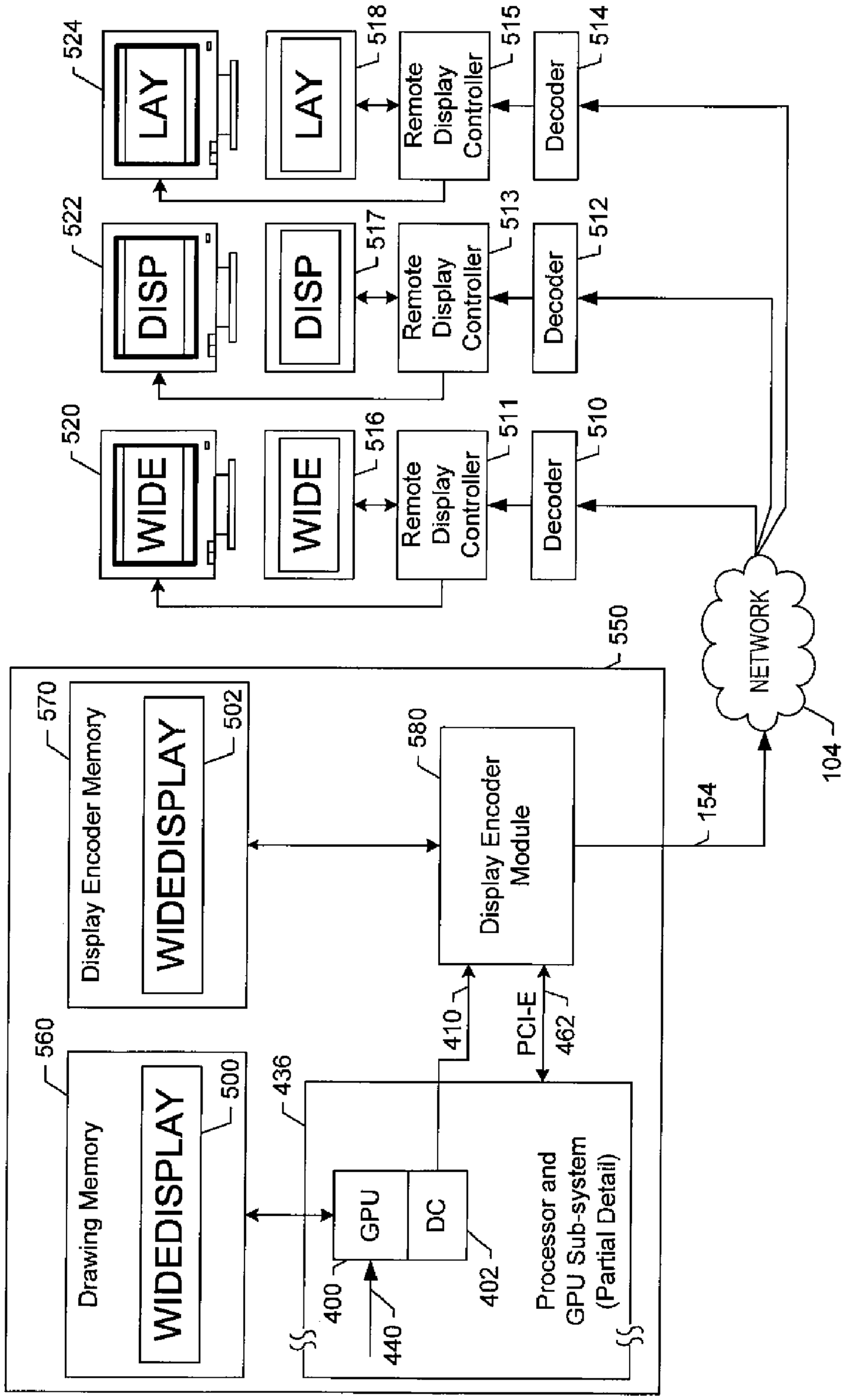


FIG. 6

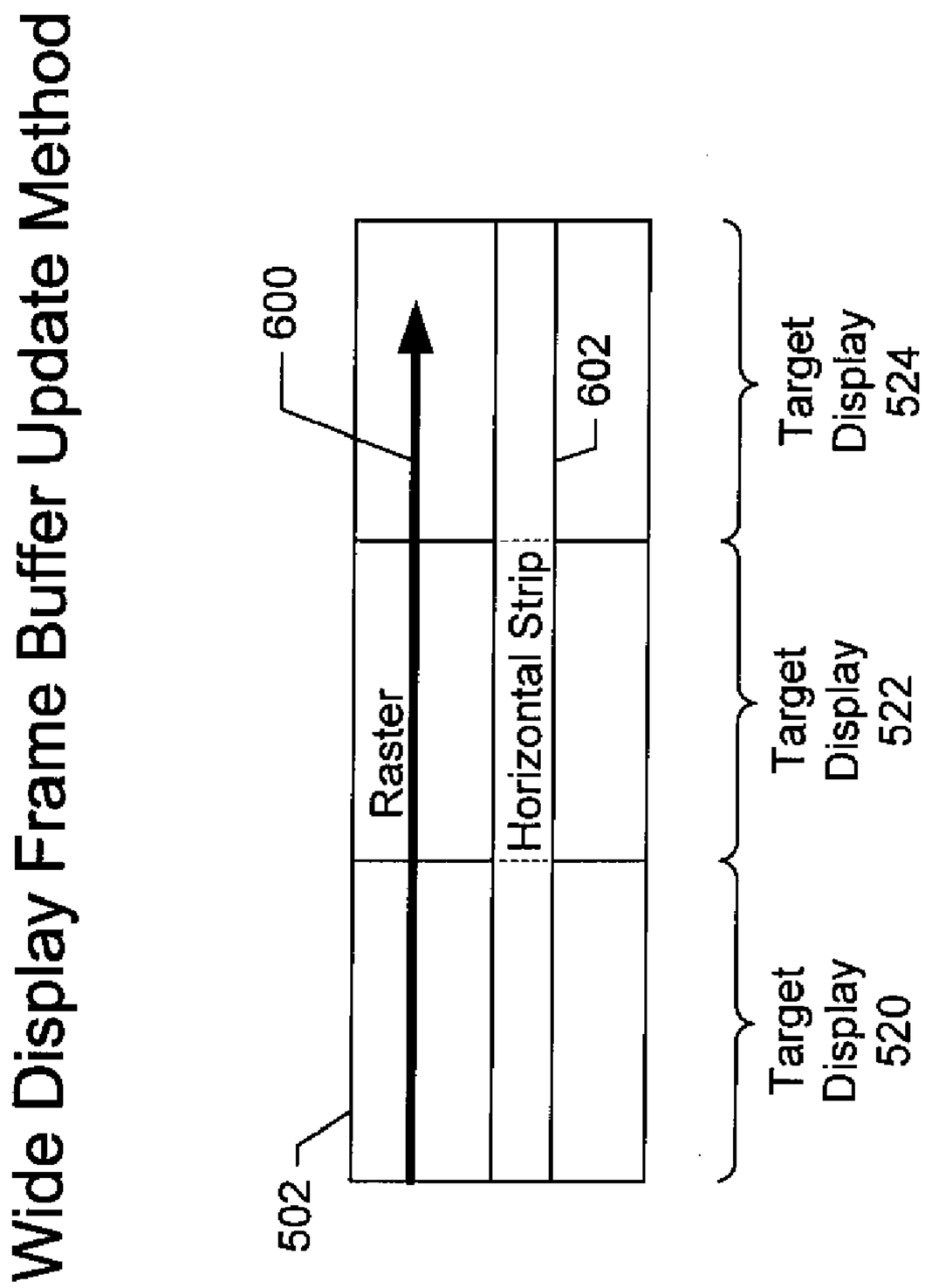

Wide Display Frame Buffer Update Method
502
600
Raster
Horizontal Strip
602
Target Display 520
Target Display 522
Target Display 524

FIG. 7
Multi-Display Remote Unit

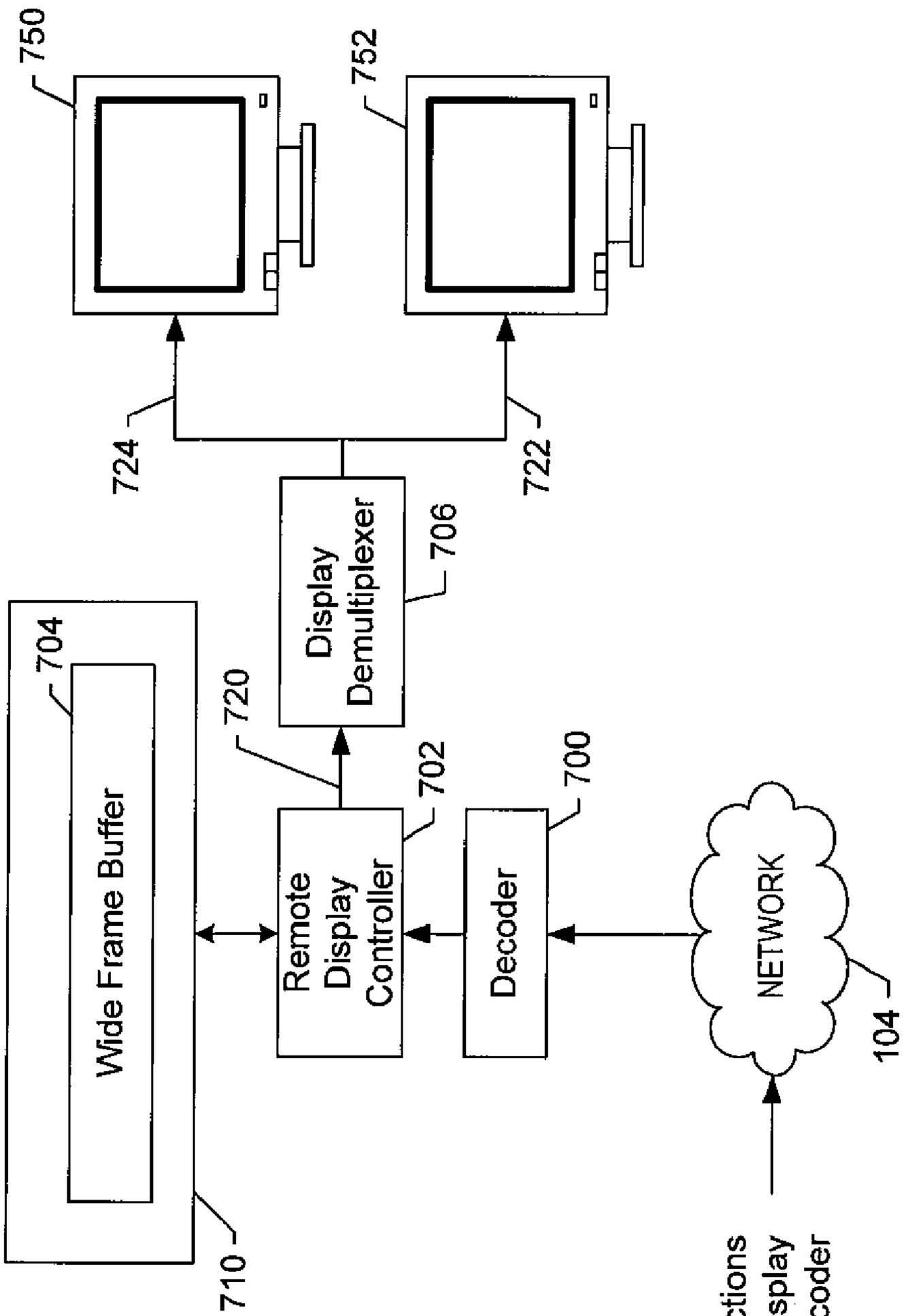
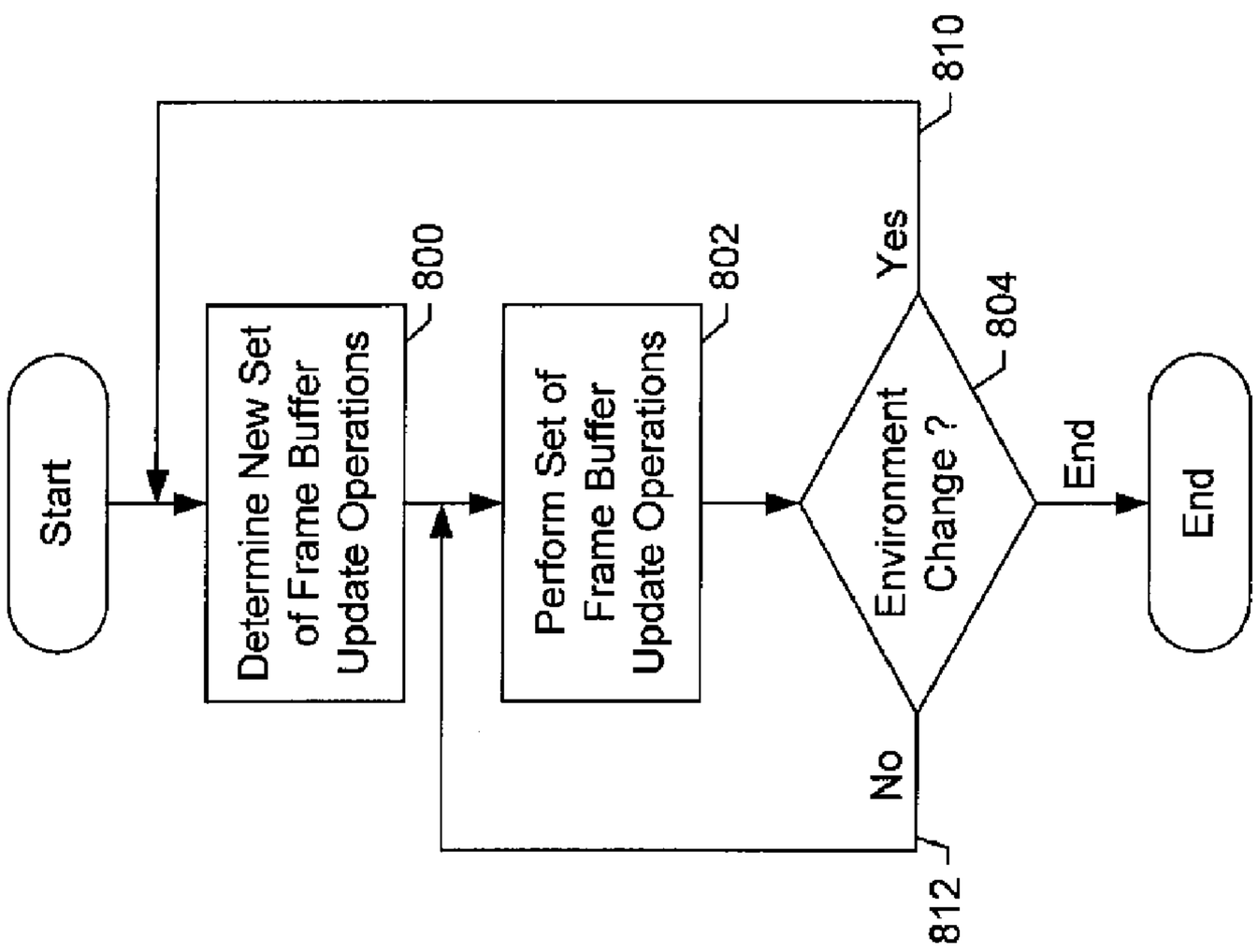


FIG. 8
Display Update Method



GPU AND ENCODING APPARATUS FOR VIRTUAL MACHINE ENVIRONMENTS

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] "This application is a continuation of co-pending U.S. patent application Ser. No. 11/278,128 entitled "Methods and Apparatus for Enabling Multiple Remote Displays" and filed Mar. 30, 2006 which claims priority to Provisional Patent Application Ser. No. 60/669,178, filed Apr. 6, 2005, each of which are incorporated herein by reference in their entirety.

BACKGROUND

[0002] 1. Field of the Invention

[0003] The present invention relates generally to encoding an image for a remote client. More specifically, the invention renders images by a graphics processor unit based on display requirements for a remote client.

[0004] 2. Discussion of Related Art

[0005] Advances in computer technology have made it economical for individual users to have their own computing system, which caused the proliferation of the Personal Computer (PC). Continued advances of this computer technology have made these personal computers very powerful but also complex and difficult to manage. For this and other reasons there is a desire in many workplace environments to separate the user interface devices, including the display and keyboard, from the application processing parts of the computing system. In this preferred configuration, the user interface devices are physically located at the desktop, while the processing and storage components of the computer are placed in a central location. The user interface devices are then connected to the processor and storage components with some method of communication.

[0006] One of the challenges with this approach relates to methods for providing users with multiple computer monitor configurations on their desktop with the identical user experience compared with multiple computer monitors connected to a local desktop computer system. These users include users in the financial services or CAD industries whose computers allow applications to spread across multiple displays from their local desktop computer.

[0007] The challenge with this configuration lies in the effective relay of images from multiple frame buffers at the data processing side of the system across the network to the desktop for display across multiple monitors at perceptually lossless image quality and within the latency limits of human perception.

[0008] Local computing platforms commonly use multiple frame buffers. As one example, Windows XP supports multiple local monitors to increase the active desktop area. As another example, gaming application use multiple frame buffers to enhance animated graphics. Less common methods for connecting multiple frame buffers to multiple remote monitors are described below.

[0009] One method for supporting multiple remote monitors connected to a data processor network by a standard network involves the copying of frame buffers from the data processor to the client equipment. One implementation of the frame buffer copy method is Virtual Network Computing (VNC). VNC uses a software driver on the data processor to read and compress the frame buffer on the data processor using delta buffer or other software-based compression methods. The compressed data is copied to the remote system using the Remote Frame Buffer protocol (RFB) on request by a client version of the VNC software which runs on the remote computer system. Some versions of VNC, such as the one provided by RealVNC support multiple displays by simultaneously connecting to different ports on the data processing system.

[0010] VNC has limited graphics support. While text and 2D graphics are drawn to a frame buffer in system memory, 3D graphics are usually handled by a GPU and not drawn to a frame buffer accessible to the VNC driver. This results in 3D windows left as blank areas on the client system. While one workaround is to use software to render 3D objects such as those drawn using 3D OpenGL, this approach is undesirable as it requires the data processor to perform drawing operations. VNC also effects data processing performance directly by sharing frame buffer access with the CPU and requiring CPU resources for data compression. Unlike VNC, which has limited 3D graphics support, VizServer™ from Silicon Graphics is a dedicated 3D rendering server which uses a multiprocessing server to render and transmit display images to remote users. Using general purpose CPUs for the compression of frame buffers is inefficient. For example, it is recommended that several dedicated processors are added to a VizServer systems to support the compression and copy of the frame buffer.

[0011] Another method for forwarding display information to the remote displays is the graphic command transfer method. X-Windows is one product that uses this method in a UNIX environment. X-Windows forwards drawing commands to the client system and requires that the client system supports the rendering hardware to draw the image at the remote system. To reduce the bandwidth needed by X-Windows, Low-Bandwidth X (LBX) is a proxy module that compresses the command stream before transmission. A major problem with the X-based solution is the requirement for remote operating system and graphics hardware to support the display application. This increases the desktop equipment cost and maintenance overheads which defeats the original objective of separating the data processor from the user interface components.

[0012] A brute force approach to enabling a system with multiple monitors is to integrate multiple parallel Keyboard-Video-Mouse (KVM) systems in the data processor, each with a dedicated connection to a remote monitor. KVM systems that use dedicated CATS or fiber cabling have distance limitations, and are costly to install and maintain due to the non-standard infrastructure requirements. KVM over IP systems use either frame buffer copy or graphic command transfer methods to copy the display information to the remote monitors and consequently suffer similar limitations to those systems described above.

[0013] A related market that uses multiple remote displays is the emerging Head Mount Display (HMD) industry. A typical head mount display is comprised of dual micro-displays, each displaying one channel of a stereo image. In most existing systems, separate analog video signals are run directly from the processing environment; however in-line architectures that capture and compress DVI signals for Ethernet and wireless transmission to a remote HMD have recently been proposed.

[0014] In summary, existing software methods for transferring display information to individual remote clients with

multiple monitors have limited capabilities, consume data processing resources and usually require a client software application to reformat the incoming display image to meet size and resolution requirements of the display area or even render the image based on transferred graphic commands. This requires complex and expensive client systems with relatively high maintenance requirements. Physical video cable extension techniques require additional cabling and have limited distance of operation.

SUMMARY

[0015] The invention provides methods and apparatus for enabling the separation of multi-user and multiple monitor digital display systems from a centralized computer using a standard network. This enables remote users to experience a high quality visual interface with little or no overhead imposed on the centralized computer system and using simple remote display components.

[0016] In one aspect, the invention provides methods for enabling a central computing system to render multiple independent images in frame buffers that match the pixel resolution and size of different target displays while an independent encoding module encodes and transmits the images to multiple remote display systems at the intended pixel resolution without impacting the performance of the central computer. This allows central computing systems to generate multiple independent images using anti-aliasing and other image enhancement methods without having the benefits lost during transmission and reformatting.

[0017] In another aspect, the invention provides a display encoder module with independent memory that stores copies of the computer frame buffers. This allows the encoding and transmission of images to multiple remote systems independent of computer processing. Unlike software-based screen scraping methods, a separate encoding module enables the selection of encoding methods and adjustment of image update rates to match remote display capabilities, timing and network conditions. The encoding module operates independent of the original image generation and without impacting the performance of the computing system.

[0018] In another aspect, the invention provides methods for multiple independent remote display systems to each display a partial image such that a wide image may be displayed by locating the monitors side by side.

[0019] In another aspect, the invention provides methods for a single remote decoder, display controller and de-multiplexer to display wide images across multiple displays.

[0020] In summary, the invention offers many benefits over existing methods of supporting multiple remote displays. Unlike software-based frame buffer copy methods, the invention does not reduce host computer performance by constantly polling frame buffers and is not limited to 2D graphics support. Neither does it require reformatting of frame buffer images to match remote display capabilities, thereby losing the advantage of features such as anti-aliased fonts. Unlike graphic command transfer methods, the method does not require complex or expensive remote components such as CPU, GPU or operating system. Unlike physical video cable extension techniques, the methods and apparatus described operate over existing standard corporate network infrastructure and are not subject to distance limitations. Many other features and advantages of the present invention will be realized upon reading the following detailed description, when considered in conjunction with the accompanying drawings, in which:

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] FIG. **1** illustrates an embodiment of a multiple remote display environment in which a host computer system is connected to multiple remote display systems by network;

[0022] FIG. **2** illustrates the internal architecture of a display encoder module;

[0023] FIG. **3** shows a host computer system with display encoder module connected to a processor and GPU subsystem by a PCI Express bus;

[0024] FIG. **4** shows an alternative embodiment of a host computer system that includes a GPU and display controller with dedicated drawing memory;

[0025] FIG. **5** shows an embodiment that supports a wide display format;

[0026] FIG. **6** illustrates the order in which a wide display frame buffer is processed in an example in which the frame buffer has the equivalent width of three remote displays;

[0027] FIG. **7** illustrates a remote system that supports multiple displays; and

[0028] FIG. **8** illustrates a method for updating remote displays.

DETAILED DESCRIPTION

[0029] FIG. **1** illustrates an embodiment of a multiple remote display environment. An example of such an environment is a centralized PC or server connected to multiple computer monitors across a network. In the environment described, the host computer system may be aware of the display capabilities of each of the remote display systems and may be capable of rendering different display images for each remote display such that the image attributes, such as pixel resolution, aspect ratio and size exactly match the capabilities of the target display.

[0030] Referring to FIG. **1**, host computer system **100** is connected to multiple remote display systems by network **104**. Host computer system **100** may be a sub-system within a personal computer (PC), it may be a stand-alone PC, or it may be a server. An example of network **104** is an Ethernet-based corporate IP network. In the embodiment shown, remote display system **102** is one of the remote display systems and is comprised of decoder **106** connected to remote display controller **108** with associated remote display memory structure **110** incorporating remote frame buffer **162**. In some remote memory embodiments, such as remote memory **114**, multiple frame buffers are incorporated (frame buffers **164** and **166** shown). Remote display controller **108** is connected to remote monitor **120**. Some remote display systems support multiple monitors, such as monitors **124** and **126** supported by remote display controller **122**.

[0031] Host computer system **100** includes processor and GPU sub-system **130** and its associated processor memory **132**, connected via data connection **150** to display encoder module **134**. Display encoder module **134** is connected to display encoder memory **136** by data connection **152**. Processor and GPU subsystem **130** includes one or more central processing units (CPUs) optionally coupled with one or more graphics processing units (OPUs) and may also be connected to network **104**, for example using connection **156** illustrated. In the embodiment, display encoder memory **136** incorpo-

rates multiple independent frame buffers, including frame buffer memory areas **140**, **142**, **144** and **146** shown. Frame buffer memory areas **140**, **142**, **144** and **146** each have a pixel resolution and size based on the attributes of one or more associated remote displays and therefore each frame buffer may have different pixel resolution and dimensions. Other configurations of display encoder memory with more or fewer frame buffers are also feasible.

[0032] In the embodiment shown, visual information derived by software applications on host computer system **100** targeting different display monitors is written to frame buffers in processor memory **132**. Each frame buffer may exactly match the pixel resolution and size of the corresponding target display, allowing images to be rendered using anti-aliased fonts or other display enhancement features useful for digital displays. In cases where images generated by host applications do not exactly match the pixel resolution and size of the target display, processor and GPU sub-system **130** have the ability to reformat or resize the image to match the display.

[0033] Display encoder module **134** reads frame buffers in processor memory **132** and stores the frame buffer information in independent frame buffers in display encoder memory **136**. The present invention employs methods to prevent display encoder module **134** from continuously polling or reading all frame buffers in processor memory **132** which adversely impacts system performance. These methods are discussed in the description of FIG. **2**.

[0034] In an alternative embodiment, visual information derived by software applications on host computer system **100** is written directly to display encoder module **134** where it is stored in independent frame buffers in display encoder memory **136**.

[0035] Display encoder module **134** then accesses frame buffer memory areas **140**, **142**, **144** and **146** in display encoder memory, independently encodes the display images associated with each remote display system and transmits encoded data from connection **154** across network **104** to their designated decoder(s). In the embodiment shown in FIG. **1**, the image in frame buffer **140** is transmitted to decoder **170**, the image in frame buffer **142** is transmitted to decoder **1 06** and the images in frame buffers **144** and **146** are transmitted to decoder **172** where they are decoded and stored in frame buffers **116** and **118** of remote memory **114**. Decoders **102**, **170** and **172** receive the display information, decode it and write it to the appropriate remote frame buffer where it is available for display on the appropriate remote monitor.

[0036] FIG. **2** illustrates the internal architecture of an embodiment of display encoder module **134** that decomposes and compresses the display sections of different frame buffers. In the embodiment, display encoder module **134** is comprised of several modules. Processor interface and registers **230** provides connection **150** to processor and GPU sub-system **130**. In one embodiment, connection **150** is a PCI-Express connection to processor and GPU sub-system **130** and processor interface and registers **230** provide a PCI-Express bus interface.

[0037] In an embodiment, connection **150** is a Digital Packet Video Link (DPVL) connection and processor interface and registers **230** captures the DPVL streams for each display and stores them in different frame buffers in display encoder memory **136**.

[0038] In another embodiment, connection **150** is a set of Digital Visual Interface (DVI) connections and processor interface and registers **230** captures the DVI stream for each display and stores them in different frame buffers in display encoder memory **136**.

[0039] In yet another embodiment, connection **150** is a VGA connection and processor interface and registers **230** provides VGA registers each display and stores the frame buffer for each display in display encoder memory **136**.

[0040] Encoder control **220** controls which frame buffer of display encoder memory to access and which region to access based on which regions have changed and require transmission. In an embodiment that copies frame buffers from processor memory **132** to display encoder memory **136**, encoder control **220** is responsible for copying frame buffer information. Various methods exist to prevent display encoder module **134** from repeatedly polling frame buffers in processor memory **132**, which adversely impacts system performance. One method uses a software table that tracks regions of processor memory that have been updated. This table may be maintained by the CPU, which monitors graphic commands and determines which areas of processor memory **132** are being updated. Changed areas are marked for reading in the software table. Display encoder module **134** then reads only areas marked in the software table and resets the table once the areas have been copied to display encoder memory **136**. An alternative method is to monitor dirty bits. Some CPU s include a memory management unit (MMU) with hardware sticky bits that are set when corresponding regions of memory have been written. In this embodiment, display encoder module **134** queries the MMU to establish which areas of processor memory **132** have been updated and instructs the CPU to reset the sticky bits once the memory has been read.

[0041] Multi-method encoder **200** includes an enhanced image encoding pipeline, controlled by encoder method selector **222** that selects encoding combinations to support different image content including lossy methods for natural images or lossless methods for computer generated text or graphic images. The encoder may also be tunable to different variations in image content such as color depth, etc. In the embodiment, the first stage in the encoder is image decomposition module **202**, which classifies the image type as a precursor to the encoding operation, which enables encoding based on image classification. Decomposition module **202** classifies the image into different image types such as background, text, picture or object layers based on spatial and temporal features such as contrast, color content, etc. Image type may be determined using image analysis methods or interpreting drawing commands. An example of an image analysis method is an image filter such as a text recognition filter. Decomposition module **202** separates the image into layers and different layers are subjected to different encoding methods in the pipeline. The other stages in the encoding pipeline include motion estimation (ME) **204**, motion compensation (MC) **206**, discrete cosine transform DCT and/or discrete wavelet transform DWT stage (T) **208**, data reordering stage (DR) **210**, and entropy (E) encoding stage **212**. Other stages can be included, depending on specific implementations of the present invention. The data reordering stage includes lossless data reordering operations e.g. color cache, LZW, run length coding, mask or data predictors, etc. The entropy encoding stage uses suitable encoders like arithmetic, Golumb or Huffman coders.

[0042] Network Interface **240** provides packetization, network controller and transport layer interface functions. For

example, packetized encoded data may be encapsulated using UDP/IP packetization and connection **154** may be an Ethernet LAN connection.

[0043] FIG. **3** shows an embodiment of host computer system **100** with display encoder module **134** connected to processor and GPU sub-system **130** by PCI Express bus **300**. In the embodiment shown, CPU **302** connects to processor memory **132** using chipset **304**. Chipset **304** also provide PCI-Express interconnect **300** to display encoder module **134**. In one alternative embodiment, chipset **304** supports GPU **306**, connected using PCI-Express or a graphics data bus such as AGP. In another alternative embodiment, chipset **304** provides network interface **308** for CPU **302** to gain access to the remote display systems and other networked devices connected to network **104**.

[0044] Processor interface and registers **230** of display encoder module **134** include PCI Express control module **303** connected to multiple display controller register sets, including but not limited to register sets **305** and **306** shown. Each register set presents itself as an independent display controller interface and each of these display controller interfaces has its own separate frame buffer. For example, in FIG. **3**, frame buffer memory **140** may be associated with register set **305** and frame buffer memory **142** may be associated with register set **306**.

[0045] Each display controller interface and associated register set is capable of being set to the same default display controller interface of processor and GPU sub-system **130**. One such example of a standard display controller interface and registers is the VGA interface.

[0046] In a virtual machine embodiment, such as the case when virtualization software such as VMWare or Xen runs on processor and GPU subsystem **130**, display encoder module **134** is instructed as to which display interface should be active at any given time and encoder control module **220** responds by controlling which frame buffer and register set is active. In the embodiment, the virtual machine software coordinates the virtual machine environment such that each virtual machine appears to have its own dedicated display interface. Display encoder module **134** then accesses the frame buffers in processor memory **132**, encodes the display information and transmits each encoded frame buffer across network **104** to a designated remote display system. In an alternative embodiment, the virtual machine environment writes image information to different address mapped regions of display encoder module **134**, each region mapping to a different frame buffer in display encoder memory **136**. Display encoder module **134** then accesses the frame buffers in display encoder memory **136**, encodes the display information and transmits each encoded frame buffer across network **104** to a designated remote display system.

[0047] FIG. **4** shows an alternative embodiment of host computer system **100** with alternative processor and GPU sub-system **436** that includes GPU **400** and display controller **402** with dedicated drawing memory **404** that hosts one or more frame buffers, including frame buffers **406**, **407**, **408** and **409** shown. In the embodiment described, GPU **400** connects to chipset **304** by a high-speed graphics bus **440** such as a PCI-Express or AGP bus. Images are rendered by GPU **400** in drawing memory **404** and then transferred to display encoder module **434** by GPU display controller (DC) **402** using various connection methods **410**. One method of transferring the frame buffer from the GPU to the display encoder module is to use a VGA connection. Another method uses a DVI connection. Yet another method uses the DPVL standard that supports the transfer of non-sequential elements using a DVI connection. Other methods include DisplayPort or Unified Display Interface (UDI) etc. Frame buffers may also be transferred from GPU **400** to display encoder module **434** across PCI-Express bus **300**. Processor interface and registers **430** of display encoder module **434** includes video interface **420** that connects to display controller **402** and PCI-Express interface **422**. Video interface **420** depends on the embodiment of display controller **402** and may be a VGA, DVI, DPVL, DisplayPort or other digital video interface. Note that while FIG. **4** refers to independent processor memory **132** and drawing memory **404**, it is equally applicable to architectures where the different functions of these memory systems is not clearly distinguished. Note also that while the embodiment refers to CPU **302**, GPU **400** and display controller **402** as distinct and independent components, it is equally applicable to architectural variations that blend these functions.

[0048] FIG. **5** shows an embodiment of the invention that supports a wide display format. In the embodiment, GPU **400** works with wide format frame buffer **500** and operates in the same manner as if connected to a single, wide display system. Display encoder module **434** receives video signal **410** in a format as previously described and stores the image in wide frame buffer **502** of display encoder memory **136**. Display encoder module **434** then processes the image and sends different sections of the image across network **104** to different decoders **510**, **512** and **514** shown. In FIG. **5**, image sections decoded by decoder **510** are stored in remote frame buffer **516** and displayed on remote display **520** by remote display controller **511**. Similarly, image sections stored in remote frame buffer **517** are displayed on remote display **522** and image sections stored in remote frame buffer **518** are displayed on remote display **524**. In this embodiment, display encoder module **434** controls the order and position of the display arrangement. In an embodiment, display controller **402** and display encoder module **434** also control the update rate of wide frame buffer **502** which does not affect the refresh rate of remote displays **520**, **522** or **524** shown. This allows display update rates to be lower than refresh rates in environments where the network has limited bandwidth available for display information.

[0049] FIG. **6** illustrates the order in which wide display frame buffer **502** is processed in an example in which frame buffer **502** has the equivalent width of three remote displays, such as the wide display system presented in FIG. **5**. As a first step, display encoder module **134** (in FIG. **5**) waits for raster signal **600** to load all of wide frame buffer **502**. As a second step, the image is processed in horizontal strips that span the entire wide frame buffer. Horizontal strip **602** shown is one such strip. As a third step, the image data is encoded using methods described previously and transmitted to designated remote display **520**, **522** or **524** in FIG. **5**.

[0050] In cases where the frame update rate is different to the display refresh rate, various techniques are available to decode the frames. In a first embodiment, a double buffer method is used to eliminate tearing problems associated with asynchronous systems. In another embodiment, a single buffer method is used. This method has a lower latency than the first method but does not eliminate the tearing of screens. An alternative method is to synchronize refresh rate of the remote display to a multiple of the refresh rate of GPU **400**. This method prevents any movement in the tear line and it

may be precisely positioned with proper phase control of the update and refresh timing. One example is to use an update rate exactly half the refresh rate and to position the tear line within the vertical retrace so that is not observable.

[0051] One method of increasing the update rate involves selective processing and transmission of frame buffer data. Using this method, display controller **402** knows which regions of drawing memory **404** should be updated and transmits only those regions. One method for achieving this is to use a nonsequential video communications protocol like DPVL.

[0052] In order to improve the response time of pointing devices, display encoder module **134** may obtain pointer location and type information from processor and GPU sub-system **130** rather than waiting for it to arrive from DVI signal **410**. This may be accomplished by either monitoring a pointer control register or receiving pointer commands directly from the operating system of processor and GPU sub-system **130**. In cases where this method of pointer display is implemented, the remote display decoder includes pointer hardware that turns the pointer on and off as it is moved between displays.

[0053] FIG. **7** illustrates a remote system that supports multiple displays. Remote display decoder **700** is connected to network **104**. Remote display controller **702** is connected to decoder **700**. Incoming images from host computer system **100** are decoded and stored across multiple frame buffers that combine to form virtual wide frame buffer **704**. Remote display controller **702** sequences display signal **720** to include the data for multiple displays. External display demultiplexer **706** decodes signal **720** and turns it into multiple streams **722** and **724** that are compatible with standard displays. Examples include separate VGA, DVI, or DPVL signals.

[0054] The use of multiplexed signal **720** minimizes the complexity of display de-multiplexer **706**. In one embodiment, de-multiplexing is alternated on each pixel between the displays. This method is applicable to displays of the same size and frequency. In another embodiment, multiplexed signal **720** is run at double the frequency. In yet another embodiment, frames containing multiple pixels are transmitted to each display in a sequence where each frame size is proportional to the relative pixel frequency of each display. It should be noted that this only works with a digital display capable of handling the jitter. Alternatively, analog displays may be supported by using a Phase Lock Loop (PLL) to generate multiple pixel clocks.

[0055] An enhancement on this method is to use a multiplexed signal that works on a standard signal set. For example, a double frequency Digital Video Out (DVO) signal may be implemented with additional control pins to support the de-multiplexing. This mechanism can then be used to directly drive a single DVO to DVI connector or a single DVO to VGA connector. Alternatively, two DVO to DVI connections or two DVO to VGA connections may be implemented using a de-multiplexing circuit.

[0056] One alternative embodiment of the remote display system shown in FIG. **7** involves a modification that controls the DVI clock. Referring to FIG. **7**, remote display controller **702** sends an image to remote display **750**, one frame at a time. In the embodiment, remote display **750** incorporates a frame buffer and is capable of storing at least one image frame. When remote frame buffer **710** has been updated by host computer system **100**, display de-multiplexer **706** is scheduled to transmit updated frame **710** to designated attached display **750** or **752**. The DVI clock is shut down during periods when no frame is being updated and is restarted once a new frame is ready; all the while maintaining the same pixel and blanking count to remain synchronized.

[0057] In an embodiment, a structure having an inverse function of demultiplexer **720** at host computer system **100** is included to multiplex two or more frame buffer sources into a single frame buffer input stream. In a multi-user system embodiment, display encoder module **134** maintains a copy of the display controller interface for each user.

[0058] In an alternative embodiment, a multi-user system such as that illustrated in FIG. **1** may be combined with a multi-display system such as the system shown in FIG. **5**. This embodiment supports multiple users, each user with multiple displays.

[0059] FIG. **8** shows a method used by display encoder modules **134** or **434** to update the associated remote displays based on a determined set of update operations. After initialization, a new set of frame buffer update operations is determined at step **800**. As one example of a set of update operations, an encoding sequence is established where all frame buffers are given equal priority and are encoded at the same update rate. As another example, different frame buffers or different areas of one frame buffer are encoded using different update rates. As another example, different frame buffers are updated at the same rate but using different encoding techniques which results in a change in proportional bandwidth used for the transfer a frame buffer to an associated remote display system. Another set of frame buffer update operations controls proportional bandwidth by combining different sequences and update rates with different encoding techniques. In an embodiment, the assignment of update rates and encoding techniques is based on display characteristics, frame buffer content or update history information, image content or other attributes. In an embodiment, a set of frame buffer update operations is determined based on frame buffer pixel resolution and size. In the embodiment, display encoder **200** provides proportional network bandwidth allocation based on relative frame buffer sizes. In another embodiment, a set of frame buffer update operations **800** is determined based on frame buffer content. In the embodiment, image decomposition module **202** classifies frame buffer contents as previously described.

[0060] The bandwidth of network link **154** is then allocated amongst competing remote systems based on frame buffer content. For example, a frame buffer containing a video sequence may receive proportionally higher update rate than a frame buffer containing text. In another embodiment, a more active display associated with a multi-display desktop may receive proportionally higher update rate. For example, wide display frame buffer **502** is associated with a multidisplay system incorporating monitors **520**, **522** and **524**. In the example, a pointing device sprite provides display encoder module **434** with an indication of which area of wide display frame buffer is active. Display encoder module **434** then uses a frame buffer update sequence that has a higher update rate for the active section. In another example, processor and GPU sub-system **130** or **436** may determine frame buffer update sequence **800**. In such an embodiment, processor and GPU sub-system **130** or **436** instructs display encoder module **134** or **434** as to which update rate and encoding method to use for each frame buffer, enabling different service qualities for different remote systems or different service qualities for different sections of one or more displays. In advanced sys-

tems, techniques such as forward error correction may be used to enable a higher service quality for selected remote displays.

[0061] As a next step **802**, a set of update operations is performed. In an embodiment, performing the set of update operations is a repetitive process in which frame buffer sections are sequentially either encoded or skipped over based on the update rates established in step **800** and then transmitted to target remote decoders over a shared network. The process is repeated for a fixed multiple of the lowest update rate to ensure all regions are encoded and transmitted. Other sequences, such as time-based sequences, sequencing as permitted by available processing resources, and sequencing determined by a subset of display regions at a time are also feasible.

[0062] As a next decision step **804**, a determination is made as to whether the environment has changed. If the environment has changed (reference numeral **81 0**), control returns to step **800** and a new set of frame buffer update operations is determined. If the environment is unchanged (reference numeral **812**), the established set of frame buffer update operations **802** is repeated. An environment change may be determined based on changes to frame buffer, remote display or other attributes. As one example, a change in pixel resolution of a remote display is detected. This may occur if a display is re-configured or physically replaced with another of different resolution. Such changes are detected by processor and GPU sub-system **130** or **436** (e.g. as provided by DDI information) and result in a corresponding change to the frame buffer size in display encoder memory **136**. As another example, the image content of a frame buffer, as determined by image encoder **200**, may be detected. Such a change is detected at the end of a video sequence or launch of a different application. As another example, a new frame buffer is added as may occur when an additional display is added to the system. As another example, a change in network bandwidth, congestion level or packet loss metric is detected and a new update sequence is determined.

[0063] While methods and apparatus for enabling multiple remote displays has been described and illustrated in detail, it is to be understood that many changes and modifications can be made to various embodiments of the present invention without departing from the spirit thereof.

What is claimed is:

1. A system for encoding an image for a remote client, the system comprising:

a Graphics Processor Unit (GPU) for rendering the image in response to graphics commands from a first virtual machine (VM) of a Central Processing Unit (CPU), wherein attributes of the image are determined by display requirements from the remote client;

a display encoder associated with the GPU and enabled to encode the image, the display encoder separated from the CPU by a peripheral bus and operating independent of the rendering of the image; and

a network interface associated with at least one of the CPU, the GPU or the display encoder for transmitting the encoded image across an Internet Protocol (IP) network to the remote client.

2. The system of claim **1**, wherein the display requirements comprise a pixel resolution requirement and wherein the graphics commands are determined by the pixel resolution requirement.

3. The system of claim **1**, wherein the display encoder comprises at least an image transform stage and an entropy encoder used to generate the encoded image, and wherein rendering the image is performed exclusive of the image transform stage or the entropy encoder.

4. The system of claim **3**, wherein the image transform stage is enabled to compute a DCT transform and wherein the entropy encoder comprises an arithmetic encoder.

5. The system of claim **1**, wherein the display encoder comprises an image decomposition stage enabled to provide classification of the image based on image type and wherein encoding of the image is in accordance with the classification.

6. The system of claim **5**, wherein the image type is determined by analysis of the graphics commands.

7. The system of claim **1**, wherein the display encoder comprises motion estimation and motion compensation stages.

8. The system of claim **1**, wherein the display encoder is enabled to encode a second image associated with one of i) an additional display of the remote client or ii) a second remote client.

9. The system of claim **8**, wherein the display encoder comprises an encoder control module for active frame buffer control between encoding the image and the second image.

10. The system of claim **9**, wherein the encoder control module is instructed by a virtual machine environment comprising a plurality of virtual machines with a plurality of associated images.

11. The system of claim **1**, wherein an update rate of the display encoder is based on activity of the image.

12. The system of claim **11**, wherein a region for the activity of the image is indicated by a pointing device location.

13. The system of claim **1**, wherein an update rate of the display encoder is determined by an available bandwidth of the IP network between the system and the remote client.

14. The system of claim **1**, wherein the encoded image transmitted by the network interface comprises selectively processed areas of the image, the selectively processed areas determined by the GPU.

15. The system of claim **14**, wherein the selectively processed areas comprise updated pixels of the image determined by the GPU, wherein the updated pixels are encoded by the encoder.

16. The system of claim **1**, wherein the GPU provides the display encoder with instructions related to an encoding method.

17. The system of claim **1**, wherein the attributes of the image comprise at least one of display resolution or aspect ratio.

18. The system of claim **1**, wherein the peripheral bus comprises a Peripheral Component Interconnect (PCI) Express bus.

19. The system of claim **1**, further comprising a VM environment maintained by the CPU, wherein each of a plurality of VMs of the VM environment appears to comprise a dedicated display interface for transmitting encoded data to a corresponding one in a plurality of remote display systems.

20. The system of claim **19**, wherein at least one in the plurality of remote display systems comprises a user input device and multiple displays.

* * * * *