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

(11) Publication number : **0 662 679 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94309871.5**

(51) Int. Cl.[6] : **G09G 5/02,** G09G 1/16

(22) Date of filing : **28.12.94**

(30) Priority : **03.01.94 US 177119**

(43) Date of publication of application :
12.07.95 Bulletin 95/28

(84) Designated Contracting States :
DE FR GB

(71) Applicant : **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor : **Rengan, Marco M.**
2054 S. Conference Drive
Boca Raton, Florida 33486 (US)
Inventor : **O'Hara, J.P. Michael**
557606 Arbor Club Way
Boca Raton, Florida 33433 (US)

(74) Representative : **Davies, Simon Robert**
I B M
UK Intellectual Property Department
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Apparatus for reformatting pixel data.

(57) Method and apparatus for effecting a hardware assisted pixel reformat during bit boundary block transfers (BITBLTs) is disclosed. In a preferred embodiment, hardware reformat logic is incorporated into a BITBLT engine of a computer system for automatically reformatting pixels from an m-bit source format to an n-bit destination format during screen refresh. The reformat logic comprises a lookup table, for mapping each bit of an m-bit pixel data word to a position in an n-bit destination pixel data word, and reformat engine connected to the lookup table for physically routing each bit of source pixel data to the appropriate destination position, as indicated by the lookup table. In this manner, bits may be mapped from one bit plane in a source bitmap to a second bit plane in a destination bitmap, thereby reformatting the pixel data, by simply programming the appropriate entries in the lookup table.

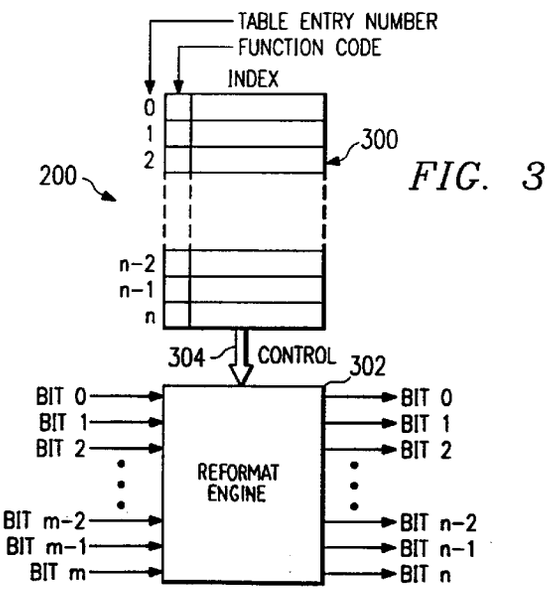


FIG. 3

EP 0 662 679 A1

The invention relates to apparatus for reformatting pixel data.

An image to be displayed on a monitor of a personal computer or workstation will typically be stored in system memory as a bitmap file, in which pixel colors are represented by a plurality of bits in a particular format. For example, in a 16-bit RGB format, the color of an individual pixel is represented by sixteen bits of pixel data, wherein the five least significant bits indicate the intensity of the blue color component, the five most significant bits indicate the intensity of the red color component and the remaining six bits indicate the intensity of the green color component. Alternatively, in a 24-bit BGR format, pixel color is represented by a 24-bit word, in which the least significant byte indicates the intensity of the red color component, the next byte indicates the intensity of the green color component and the most significant byte indicates the intensity of the blue color component.

Several different pixel color formats may be used within a computer or workstation to represent pixels, it being well known that the greater the number of bits used to represent a single pixel, the wider the range of colors that may be represented. For example, an image may be stored on an external storage device, such as a CD-ROM, in a 24 bits-per-pixel (bpp) bit map format, while the same image will be displayed on the monitor in a 32 bpp format. For that reason, it is necessary for computers to be able to efficiently convert pixel data from a source format, which in the above case would be 24 bpp, to a destination format, which in the above case would be 32 bpp.

Presently, the ability to convert pixel data from one color format to another using hardware support is extremely limited and is typically implemented on a pixel-by-pixel basis using software to convert each pixel from a source format to a destination format. This process necessarily expends several microseconds or tens of microseconds per pixel, such that a considerable amount of processing time is required to convert all of the pixels of a single image.

Alternatively, hardware may be used to reformat only a specific set of pixels, i.e., to convert from one color format to another. For example, the above conversion from a 24 bit-per-pixel format to a 32 bit-per-pixel format may be hardwired into the system. Unfortunately, this technique is also deficient in that it sacrifices great deal of flexibility in exchange for only a marginal increase in overall conversion speed.

Another technique currently in use is a color expansion technique used in connection with character data stored in a 1 bit per pixel format in which a "0" represents a background pixel and a "1" represents a foreground pixel. A foreground register and a background register store foreground and background colors, respectively, such that each "1" of source data causes the color corresponding to the value in the foreground register to be displayed at the corresponding pixel location and each "0" causes the color corresponding to the value in the background register to be displayed at the corresponding pixel location. This technique is also deficient, as its use is limited to a one bpp format and a choice between two colors.

The increasing use in the personal computer (PC) industry of operating systems having windowing capabilities, such as IBM's OS/2 and Microsoft's Windows, as well as the move toward graphics adapters having direct color modes, evidence the need for pixel reformat operations to be accomplished more rapidly and to allow for conversion from several different source formats to several different destination formats. Further evidence of this need is found in the fact that the industry is placing greater importance on the performance of PC graphics, as exemplified by the fact that a major selling point of graphics accelerator display adapters is their "WINMARK" number, which is a benchmark number assigned to computer components based on various performance criteria. It is highly likely that future WINMARK numbers will take into account the speed of memory-to-screen copy operations in which the source and destination formats differ.

While the available techniques for converting between pixel formats get the job done, they are exceedingly slow, especially in view of the new demands being placed on graphics subsystems. Although the conversion process may be accelerated somewhat by hardwiring certain conversions, such hardwired solutions do not provide the necessary flexibility required to enable conversion to and from a plurality of different pixel formats.

Therefore, what is needed is an arrangement for reformatting pixels encoded in a first, or source, format to a second, or destination, format in a rapid and efficient manner.

Accordingly, the present invention provides apparatus for reformatting pixel data comprising an m-bit word to an n-bit word, the apparatus comprising: a reformat engine for receiving said m-bit word and for outputting said n-bit word; and a lookup table electrically connected to said reformat engine and comprising at least one table entry associated with a bit position of said n-bit word for indicating a bit source; wherein said reformat engine creates an electrical connection between said indicated bit source and said associated n-bit word bit position, according to said look-up table, such that a bit supplied by said indicated bit source is output to said associated n-bit word bit position.

The foregoing problems are solved and a technical advance is achieved by method and apparatus for effecting a hardware assisted pixel reformat during bit boundary block transfers (BITBLTs) from a source memory location to a destination memory location. In a departure from the art, dedicated hardware, herein referred to

as reformat logic, is incorporated into a computer for automatically reformatting pixel data encoded in an m bit-per-pixel (bpp) source format to an n bpp destination format during screen refresh, for example.

In a preferred embodiment, the reformat logic is embodied in an otherwise conventional BITBLT engine of the system such that during each BITBLT from the source memory location to the destination memory location, each m-bit source pixel data word in the block pixel data being transferred is input to the reformat logic on m input lines. The reformat logic converts the pixel data from the m-bit source format to an n-bit destination format and outputs the reformatted pixel data to the destination memory location on n output lines. The reformat logic comprises a lookup table, for mapping each bit of pixel data from a first bit position in the m-bit source pixel data word to a second bit position in the n-bit destination pixel data word, and reformat engine connected to and controlled by the lookup table for physically routing the bits of pixel data from one of the m reformat logic inputs to one or more of the n reformat logic outputs, as indicated by the lookup table. In this manner, each bit of a pixel data word may be mapped from one bit position in an m bpp source bitmap to a second bit position in an n bpp destination bitmap, thereby reformatting the pixel data, by simply programming the appropriate entries in the lookup table.

In one embodiment, the lookup table comprises n table entries; one for each bit position of the destination pixel data word, or bit plane of the destination bitmap. For example, if the destination format is 32 bpp, the lookup table will comprises 32 table entries, numbered 0 to 31, corresponding respectively to bits 0 to 31 of the destination pixel data word. The two or more most significant bits of each table entry comprise a function code for causing a value at a source bit position indexed by the remaining bits of the table entry, which comprise a "source index," to be mapped into the destination bit position indexed by the table entry number, or for causing a binary 1 or 0 to be written to the indexed destination bit position, regardless of the value of the source index. The remaining bits of each table entry are referred to as the source bit position index because they comprise an index to the bit position of the source bit to be mapped to the corresponding destination bit position. For example, if the source bit position index of table entry 0 is 4h, then bit 4 of the source pixel data word is to be mapped to bit 0 of the destination pixel data word. In other words, bit plane 4 of the source bit map is to be mapped to bit plane 0 of the destination bit map. The above-described lookup table arrangement enables multiple planes of a source bitmap to be mapped to a single plane of a destination bitmap.

In an alternative embodiment, the lookup table comprises m table entries; one for each bit position in the source pixel data word, or bit plane in the source bitmap. Assuming that the format of the source bitmap is 24 bpp, the lookup table will comprise 24 individual entries, numbered 0 through 23, corresponding respectively to bits 0 through 23 of the source pixel data word. Again, the two most significant bits of each table entry comprise a function code, as described above, and the remaining bits comprise a destination, rather than a source, index. Therefore, if the index at table entry 0 is 4h, bit 0 of the source pixel data word will be mapped to bit 4 of the destination pixel data word. In other words, bit plane 0 of the source bit map will be mapped to bit plane 4 of the destination bit map.

The steering logic comprises appropriate logic gates and other hardware components for effecting the appropriate physical connections between the m inputs and one or more of the n outputs of the reformat logic for routing each bit of source pixel data to the appropriate destination bit position, or bit plane, as designated by the values in the lookup table.

A technical advantage achieved with the invention is that it can be used to reformat pixel data for representing images of any size.

A further technical advantage achieved with the invention is that it represents an improvement in performance over conventional software reformatting techniques.

Still a further technical advantage achieved with the invention is that it is flexible enough to convert any direct color format to any other direct color format. Additionally, it is flexible enough for a single embodiment to be used to perform color compression, expansion, conversion, and selection operations.

Embodiments of the present invention will now be described, by way of examply only, with reference to the accompanying drawings in which:

FIG. 1 is a schematic block diagram of a computer system.

FIG. 2 is a block diagram of a bit block transfer engine of the computer system of FIG. 1 embodying features of the present invention.

FIG. 3 is a block diagram of reformat logic for implementing the present invention.

FIGS. 4a-4c illustrate an exemplary reformat operation performed using the reformat logic of FIG.3.

FIG. 5 is a schematic block diagram of a possible implementation of a reformat engine of the reformat logic of FIG. 3.

FIG. 1 illustrates a computer 10 embodying features of the present invention. The computer 10 includes a host portion 10a and a video subsystem portion 10b to the left and right respectively of a dashed line 11. The host 10a comprises a CPU 12, system memory 14, and an external storage device 16, such as a hard

disk or CD-ROM, interconnected via a system bus 18. Images for display on a display 20 of the video subsystem 10b may be stored on the external storage device 16, as will be described.

In addition to the display 20, video subsystem 10b comprises a bus interface 22 for interfacing the video subsystem 10b to the host 10a. A graphics coprocessor 24 is connected to the bus interface 22 via a bus 25 and a video RAM (VRAM) 26, or "frame buffer," and a RAMDAC 28 are also connected to the bus interface 22 via the video bus 23. The display 20 is connected to the output of the RAMDAC 28.

A bit boundary block transfer (BITBLT) engine 30 is shown as being embodied within the coprocessor 24; however, it should be understood that the BITBLT engine 30 is independently addressable by the CPU 12 and may reside directly on the video bus 23 or on the system bus 16. The main function of the BITBLT engine 30 is to transfer rectangular blocks of data from a source memory location to a destination memory location. For example, the BITBLT engine 30 may be used to transfer blocks of data from system memory 14 to VRAM 26, from VRAM 26 to system memory 14, from one location in system memory 14 to another location in system memory 14, or from one location in VRAM 26 to another location in VRAM 26. In any case, it should be understood that in this context, the term "source" refers to the memory location from which a block of data is transferred, while "destination" refers to the memory location to which that block of data is transferred.

As previously indicated, digital pixel data for an image to be displayed on the display 20 may be stored on the external storage device 16. During a typical display operation, the pixel data is read from the external storage device 16 into system memory 14 and then transferred to VRAM 26 by the BITBLT engine 30. The digital pixel data is then output from the VRAM 26 to the RAMDAC 28. The RAMDAC 28 converts the digital data into analog red, green and blue signals for driving the display 20, which will typically comprise a CRT having red, green and blue electron guns, the intensities of which are controlled by the RAMDAC 28 signals.

The use of a BITBLT engine to move blocks of pixel data from a system memory to a VRAM and the use of a RAMDAC to convert digital pixel data output from a VRAM into analog red, green and blue signals for driving a CRT display are concepts that are well known in the art and will therefore not be further described. However, a problem arises when the format in which pixel data is stored in the source memory, which in this case is system memory 14, differs from that in which pixel data is stored in the destination memory, which in this case is the VRAM 26. For example, pixels comprising an image may be stored in system memory 14 in a 24 bpp format, but displayed on the display 20, and therefore stored in VRAM 26, in a 32 bpp format. Accordingly, at some point before the pixels can be displayed, the pixel data must be converted from the source pixel data format (i.e., 24 bpp) to the destination pixel data format (i.e., 32 bpp). As will be subsequently described in detail, the inventive technique solves the reformatting problem by providing reformat logic within the BITBLT engine 30 which logic is capable of effecting one or more types of reformatting operations during each BITBLT of pixel data from system memory 14 to VRAM 26.

FIG. 2 is a block diagram of the BITBLT engine 30. As illustrated, the BITBLT engine 30 comprises reformat logic 200 having m inputs and n outputs for converting pixel data from an m-bit source format to an n-bit destination format during a BITBLT, it being understood that "m" and "n" are used throughout the specification to denote the number of bits-per-pixel comprising the source and destination pixel data formats, respectively. Pixel data is input to the reformat logic 200 from the source memory location on the m input lines one m-bit pixel data word at a time output from the reformat logic 200 to the destination memory location one n-bit pixel data word at a time on the n output lines, as will be further described with reference to FIG. 4a.

FIG. 3 is a more detailed block diagram of the reformat logic 200 of FIG. 2. As illustrated, reformat logic 200 comprises a lookup table 300 connected to reformat engine 302 via control lines 304.

The function of the lookup table 300 is to map each bit of pixel data from a first bit position in the source pixel data word to a second bit position in the destination pixel data word. The lookup table 300 can also be described as mapping each bit of a first bit plane of a source bitmap to a second bit plane of a destination bitmap. The reformat engine 302 provides the physical connections between the m inputs and n outputs for appropriately routing each bit of pixel data from the source bit position (or bit plane) to the destination bit position (or bit plane) for effecting the reformatting operation(s) as specified by entries in the lookup table 300. One input bit may be connected to one or more output bits.

In one embodiment, the lookup table 300 comprises n table entries wherein each entry corresponds to a bit position in the destination pixel data word, or a bit plane of the destination bit map. Therefore, assuming the destination format comprises 32 bpp, there will be 32 entries in the lookup table 300 such that table entry 0 indicates the binary value to be mapped to destination bit position 0, table entry 1 indicates the binary value to be mapped to destination bit position 1, and so on through table entry n-1. The two or more most significant bits of each table entry comprise a function code for causing a value at a source bit position indexed by the remaining bits, which comprise a "source index", to be mapped into the destination bit position indexed by the table entry number or causing a binary 1 or 0 to be written to the indexed destination bit position, regardless of the value of the source index. A list of specific 3-bit function codes and their corresponding operations are

set forth below in Table I:

Code	Operation
000	Write binary 0 to indexed destination bit
001	Write binary 1 to indexed destination bit
010	Map indexed source bit to indexed destination bit
011	Invert source bit
100-111	Reserved

Table I

The following examples are provided to illustrate the use of the above function codes. If the function code of table entry 0 is 000, a binary 0 will be written to bit 0 of the destination pixel data word, regardless of the value of the bit indexed by the source index. If the function code of table entry 0 is 001, a binary 1 will be written to bit 0 of the destination pixel data word, regardless of the value of the bit indexed by the source index. If the function code of table entry 0 is 010, the value of the bit indexed by the source index will be mapped to bit 0 of the destination pixel data word unchanged. If the function code of table entry 0 is 011, the inverted value of the bit indexed by the source index will be written to bit 0 of the destination data word.

As indicated, so long as the function code of a particular table entry is 10, the corresponding source index will be used to index a bit in the source pixel data word to be mapped to the bit in the destination pixel data word indexed by the table entry number. For example, assuming the value of the source index at table entry 0 is 4h, and further assuming that the corresponding function code is 10, bit 4 of each source pixel data word will be mapped to bit 0 of each destination pixel data word. In other words, bit plane 4 of the source bit map will be mapped to bit plane 0 of the destination bit map. It should be understood that indexing the lookup table 300 in the manner described above enables more than one source bit plane to be mapped to a single destination bit plane, which may be desirable in certain color expansion operations known in the art. It should also be understood that the number of bits comprising the source index in the lookup table will be x, where $m = 2^x$. For example, for a 32 bpp source format, the number of bits comprising the source index will be 5, leaving 3 bits of a 1-byte table entry to be used as a function code.

In an alternative embodiment, the lookup table 300 comprises m entries corresponding to each of the m-bits of the source pixel data word. Accordingly, each table entry will comprise a function code and a destination, rather than a source, index. Similar to the function of the source index, the destination index indexes the bit in the destination pixel data word to which the bit of the source pixel data word indexed by the table entry number is to be mapped. In this alternative embodiment, assuming that the destination index at table entry 0 is 4h, and further assuming that the corresponding function code is 10, bit 0 of each source pixel data word will be mapped to bit 4 of each destination pixel data word, thereby effectively remapping plane 0 of the m bpp source bitmap to plane 4 of the n bpp destination bitmap.

The table entries in the lookup table 300 are initially set by software, which software may be stored in system memory 14 and executed by CPU 12, according to the particular reformat operation(s) to be performed. Once the lookup table 300 has been initialized, it is used to set the hardware reformat engine 302 so that each of the m bits of each source pixel data word are correctly routed to one or more of the n bit positions of the destination pixel data word, as further illustrated and explained with reference to FIGS. 4a-4c.

FIG. 4a-4c illustrate the operation of the reformat logic 200 of the present invention. It should be understood that the reformat operation shown in FIGS. 4a-4c is intended only as an example to illustrate the flexibility of the present invention and should not be interpreted as showing a particularly desirable type of format operation.

Referring to FIG. 4a, reference numeral 400 designates reformat logic for reformatting a source pixel data word 400a encoded in a 4 bpp source format to a destination pixel data word 400b encoded in an 8 bpp destination format. An initialized lookup table 402 includes eight table entries 0 through 7 and is connected to reformat engine 404 in which input-output connections have been set according to the table entries of the lookup table 402, as will be described.

Referring to the lookup table 402, table entry 0 comprises a function code of 00, corresponding to a "write binary 0" operation. As a result, the source index is disregarded, the reformat engine 404 creates an electrical connection between output 0 and logic 0, as indicated by a line 406, and a binary 0 is output to bit 0 of destination pixel data word 400b. Table entry 1 comprises a function code of 01, corresponding to a "write binary

1" operation. As a result, the source index is disregarded, the reformat engine 404 creates an electrical connection between output 1 and logic 1, as indicated by a line 408, and a binary 1 is output to bit 1 of the destination pixel data word 400b. Table entry 2 comprises a function code of 10, corresponding to a "map indexed source bit" operation, and a source index of 01. As a result, the reformat engine 404 creates an electrical connection between output 2 and input 1, as indicated by a line 410, and the binary value in bit 1 of the source pixel data word 400a, which in this case is a 1, is mapped to bit 3 of the destination pixel data word 400b. Table entry 3 comprises a function code of 10 and a source index of 00. As a result, the reformat engine 404 creates an electrical connection between output 3 and input 0, as indicated by a line 412, and the binary value in bit 0 of the source pixel data word 400a, which in this case is a 1, is mapped to bit 3 of the destination pixel data word 400b. In a similar fashion, the reformat engine 404 creates electrical connections between outputs 4, 5, 6 and 7 and inputs 3, 2, 1 and 0, respectively, as indicated by lines 414, 416, 418 and 420, respectively, to map binary values in bits 3, 2, 1 and 0 of the source pixel data word 400a (which binary values are 0, 1, 0 and 1, respectively) to bits 4, 5, 6 and 7 of the destination pixel data word 400b.

FIGS. 4b and 4c illustrate the mapping of the four planes of a 4 bpp bitmap 410 to the eight bit planes of an 8 bpp destination bitmap 412 effected by the above described operation. For clarity, function codes are not shown in the lookup table 402 of FIG. 4b and each source index is expressed as a decimal value. Planes 0 and 1 of the destination bitmap 412 will comprise all zeros and all ones, respectively, due to the effect of the function codes at table entries 0 and 1 (FIG. 4a), respectively. Furthermore, bit plane 1 of the source bitmap 410 is mapped to bit planes 2 and 6 of the destination bitmap 412, bit plane 0 of the source bitmap 410 is mapped to bit planes 3 and 7 of the destination bitmap 412, bit plane 3 of the source bitmap 410 is mapped to bit plane 4 of the destination bitmap 412 and bit plane 2 of the source bitmap 410 is mapped to bit plane 5 of the destination bitmap 412. The above mapping is more clearly illustrated in FIG. 4c.

FIG. 5 is a partial schematic block diagram of a preferred implementation of the reformat engine 302, for performing the reformat operation illustrated in FIGS. 4a-4c. As shown in FIG. 5, source bits of the source pixel data word 400a (FIG. 4a) are applied to inputs 0-3 of several multiplexors (MUXes) 504a-504d. The two bit source indices at table entry numbers 0-3 of the lookup table 402 are applied to select inputs S0, S1 of the MUXes 504a-504d, respectively, to select the source bit to be output from the respective MUX 504a-504d. The outputs of MUXes 504a-504d are applied to inputs 2 and 3 of MUXes 508a-508d, respectively. Inputs 0 and 1 of each of the MUXes 504a-508d are tied to logic 0 and logic 1, respectively. The two bit function codes at table entry numbers 0-3 of lookup table 402 are applied to select inputs S1, S0 of MUXes 508a-508d, respectively, for selecting an input of the respective MUX to be output therefrom to bit position 0-3, respectively, of the destination pixel data word 400b (FIG. 4a).

For example, the function code 00b of table entry 0 is applied to the select inputs of MUX 508a, causing the bit applied to input 0 thereof, which is a 0, to be output to destination bit position 0. Similarly, the function code 01 of table entry 1 is applied to the select inputs of MUX 508b, causing the bit applied to input 1 thereof, which is a 1, to be output to destination bit position 1. Referring to the MUXes 504c, 508c, the source index 01b of table entry 2 is applied to the select inputs of MUX 504c, causing the bit applied to input 2 thereof, which is a 1, to be output from MUX 504c to inputs 2 and 3 of MUX 508c. The function code 10b of table entry 2 is applied to the select inputs of MUX 508c, causing the bit applied to input 2 thereof, which is a 1, to be output from MUX 508c to destination bit position 2. Finally, referring to MUXes 504d, 508d, the source index 00b of table entry 3 is applied to the select inputs of MUX 504d, causing the bit applied to input 0 thereof, which is a 1, to be output from MUX 504d to inputs 2 and 3 of MUX 508d. The function code 10b of table entry 3 is applied to the select inputs of MUX 508d, causing the bit applied to input 2 thereof, which is a 1, to be output from MUX 508d to destination bit position 3.

Although not shown, it should be understood that additional MUXes connected in a similar fashion will be provided within the reformat engine with respect to destination bits 4-7 of the destination pixel data word 400b (FIG. 4a). It should also be understood that the circuitry illustrated in FIG. 5 is for the purposes of example only and that any number of known logic techniques and components may be used to implement the reformat engine 302.

In a particular method of using of the above-described invention, each source color component is copied to the corresponding destination color component; in other words, the source red component is copied to the destination red component, the source green component is copied to the destination green component and the source blue component is copied to the destination blue component. If the number of bits comprising the destination color component is less than the number of bits comprising the corresponding source color component (i.e., $m > n$), a truncation operation will be performed, with the least significant bits of the source color component being discarded. The truncation operation should preferably take into account the value of the most significant discarded source color component bit (via a rounding operation) so that the remaining value may be rounded up or down, as appropriate.

Alternatively, if the number of bits in the destination color component is greater than the number of bits in the corresponding source color component (i.e., m < n), the high order bits of the source color component will be repeated in the low order bits of the destination color component to render the best approximation.

The following four examples illustrate the above-described method:

1. Color Expansion--16 bit RGB source format (5 bits red, 6 bits green, 5 bits blue) to 32-bit XRGB destination format (8 bits each red, green and blue).
2. Color Conversion--24-bit RGB source format (8 bits each red, green and blue) to 24-bit BGR destination format (8 bits each blue, green and red).
3. Color Compression--24-bit RGB source format (8 bits each red, green and blue) to 16-bit RGB destination format (5 bits red, 6 bits green, 5 bits blue).
4. Color Selection--Extract planes 0 and 4 from 8-bit source bitmap to create a 2-bit destination bitmap.

Although example 4 is perhaps a bit esoteric, it illustrates the flexibility of the present invention. In the above manner, the following types of reformatting operations can be easily effected using the apparatus and method of the present invention by simply programming the lookup table 300 of FIG. 3 with appropriate function code and index values for causing the appropriate input/output connections to be created by the reformat engine 302, as previously described:

1. color expansion--converting a pixel of m bits into a pixel of n bits where m < n;
2. color conversion--converting a pixel of m bits in one format to a pixel of n bits in another format where m = n;
3. color compression--converting a pixel of m bits into a pixel of n bits where m > n; and
4. color selection--converting a pixel by copying selected planes.

In operation, the entries in the lookup table 402 are initially set by software. In this manner, the reformat logic 200 can be programmed to perform any one of the above types of reformatting operations. Pixel data representing an image to be displayed on the display 20 is copied from the external storage device 16 into system memory 14, where the data is stored as a bitmap of m bpp. Rectangular blocks of pixel data retrieved from system memory 14 by the BITBLT engine 30 are input to the reformat engine 302 one m-bit pixel data word at a time on m input lines and are reformatted into an n-bit destination pixel data word by the reformat engine's connecting each of the m input lines to one or more of the n output lines, as specified in the lookup table 300.

It is understood that the present invention can take many forms and embodiments. The embodiments shown herein are intended to illustrate rather than to limit the invention, it being appreciated that variations may be made without departing from the spirit or the scope of the invention. For example, rather than being embodied within the BITBLT engine 30, reformat logic 200 may reside directly on the system bus 18 or video bus 23, such that any time data is written from one location to another, regardless of whether this operation is performed as a BITBLT by the BITBLT engine 30, reformatting will be performed, for example, when a CPU MOV instruction is executed. Furthermore, reformat engine 302 may comprise any number of combinations of known logic elements and circuits. Additionally, the lookup table 300 may comprise any one of a number of various memory devices and the display 20 may comprise a type of display other than a CRT, such as a liquid crystal display (LCD), for example.

An embodiment provides apparatus for reformatting pixel data from a first bitmap format to a second bitmap format: a lookup table comprising a plurality of table entries each indicating a bit source, wherein each of said table entries is associated with a bit plane of said second bitmap and comprises an index for indexing a bit plane of said first bitmap; and reformat engine connected to said lookup table, wherein for each of said table entries, said reformat engine creates an electrical connection between said indicated bit source and said associated second bitmap bit plane for mapping bits from said indicated bit source to said associated second bitmap bit plane.

Claims

1. Apparatus for reformatting pixel data comprising an m-bit word to an n-bit word, the apparatus comprising:

   a reformat engine for receiving said m-bit word and for outputting said n-bit word; and

   a lookup table electrically connected to said reformat engine and comprising at least one table entry associated with a bit position of said n-bit word for indicating a bit source;

   wherein said reformat engine creates an electrical connection between said indicated bit source and said associated n-bit word bit position, according to said look-up table, such that a bit supplied by said indicated bit source is output to said associated n-bit word bit position.

2. The apparatus of claim 1 wherein said at least one table entry comprises a source index for indexing a bit position of said m-bit word, said indicated bit source comprises said indexed m-bit word bit position and said bit supplied by said indicated bit source comprises a binary value stored at said indexed m-bit word bit position.

3. The apparatus of either of Claims 1 or 2 wherein said at least one table entry comprises a "write binary one" function code, said indicated bit source comprises a positive five volt source and said bit supplied by said indicated bit source comprises a binary one.

4. The apparatus of either of Claims 1 or 2 wherein said at least one table entry comprises a "write binary zero" function code, said indicated bit source comprises electrical ground and said bit supplied by said indicated bit source comprises a binary zero.

5. The apparatus of any preceding claims wherein said reformat engine comprises at least one logic element the state of which is controlled by said at least one table entry.

6. The apparatus of any preceding claims wherein said conversion occurs during a bit block boundary transfer (BITBLT) of pixel data from a first memory location to a second memory location.

7. Apparatus of any precedingclaim, wherein said m-bit word is derived from a first bitmap and said n-bit word forms part of a second bitmap.

8. A bit boundary block transfer (BITBLT) engine for converting pixel data from a source format to a destination format during a BITBLT of said pixel data from a source memory location at which pixel data is stored in said source format to a destination memory location at which said pixel data is to be stored in said destination format, the BITBLT engine comprising:

reformat engine connected to said source memory location and having a plurality of inputs electrically connected to said source memory location for receiving a source format pixel data word therefrom and at a plurality of outputs electrically connected to said destination memory location for outputting a destination format pixel data word thereto; and

a lookup table connected to said reformat engine and comprising table entries each associated with a bit position of said destination format pixel data word, wherein each of said table entries comprises a source index for indexing a bit position of said source format pixel data word and a function code for indicating a bit to be mapped to said associated destination format pixel data word bit position;

wherein when said function code comprises a first value, said bit to be mapped to said associated destination format position comprises a bit at said indexed bit position of said source format pixel data word.

9. The apparatus of Claim 8 wherein when said function code comprises a second value, said bit to be mapped to said associated destination format pixel data word bit position comprises a binary one.

10. The apparatus of Claim 8 wherein responsive to said function code comprising a third value, said bit to be mapped to said associated destination format pixel data word bit position comprises a binary zero.

11. The apparatus of Claim 8 wherein said source format is m bits-per-pixel and said destination format is n bits-per-pixel, wherein m and n positive integers, and said lookup table comprises n table entries.

12. A method of converting pixel data from an m bit-per-pixel format to an n bit-per-pixel format during a bit block boundary transfer (BITBLT) of said pixel data from a source bitmap to a destination bitmap, the method comprising;

initializing a lookup table comprising n table entries, wherein each of said n table entries is associated with a destination bit plane of said destination bitmap, such that each of said n table entries indicates a source of bits to be mapped to said associated destination bit plane;

for each table entry, using reformat engine to create an electrical connection between said associated destination bit plane and said indicated bit source;

using a BITBLT engine to retrieve a block of said pixel data from said source bitmap;

inputting said retrieved pixel data block to said reformat engine one m-bit word at a time; and

outputting said block of data from said reformat engine to said destination bitmap one n-bit word at a time.

13. The method of Claim 12 wherein each of said n table entries comprises a source index and wherein, for each of said n table entries, said initializing further comprises initializing said source index to index a source bit plane to be mapped to said associated destination bit plane and said creating step further comprises creating an electrical connection between said indexed source bit plane and said associated destination bit plane.

14. The method of Claims 12 or 13 wherein at least one of said n table entries comprises a function code, and wherein for each of said at least one of said n table entries, said initializing step further comprises setting said function code to a first value and said creating step further comprises creating an electrical connection between a positive five volt source and said associated destination bit plane.

15. The method of either of Claims 12 or 13 wherein at least one of said n table entries comprises a function code, and wherein for each of said at least one of said n table entries, said initializing step further comprises setting said function code to a second value and said creating step further comprises creating an electrical connection between a electrical ground and said associated destination bit plane.

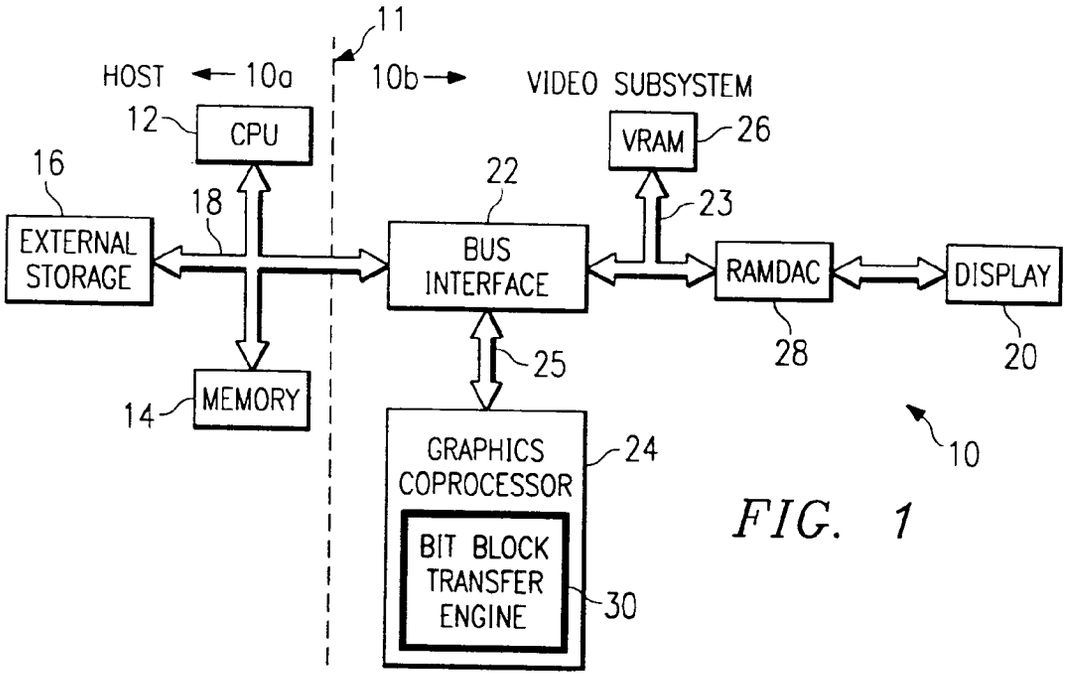


FIG. 1

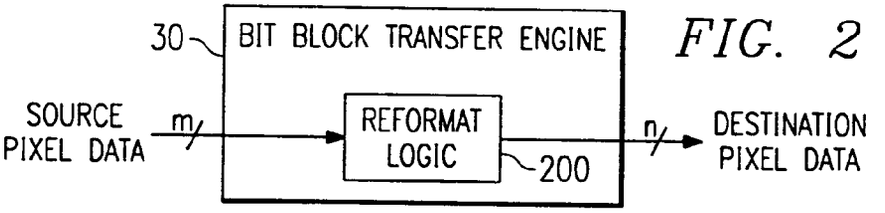


FIG. 2

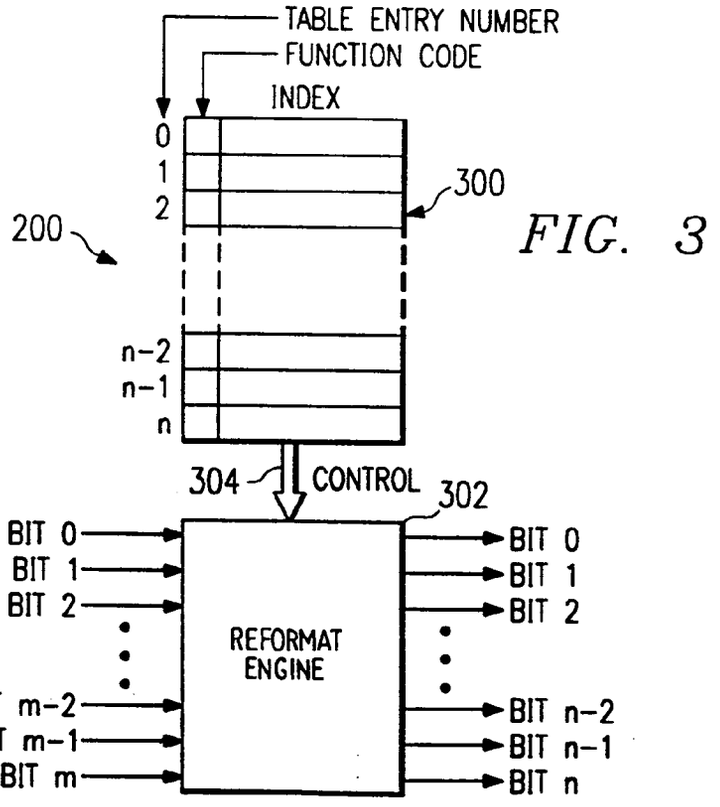
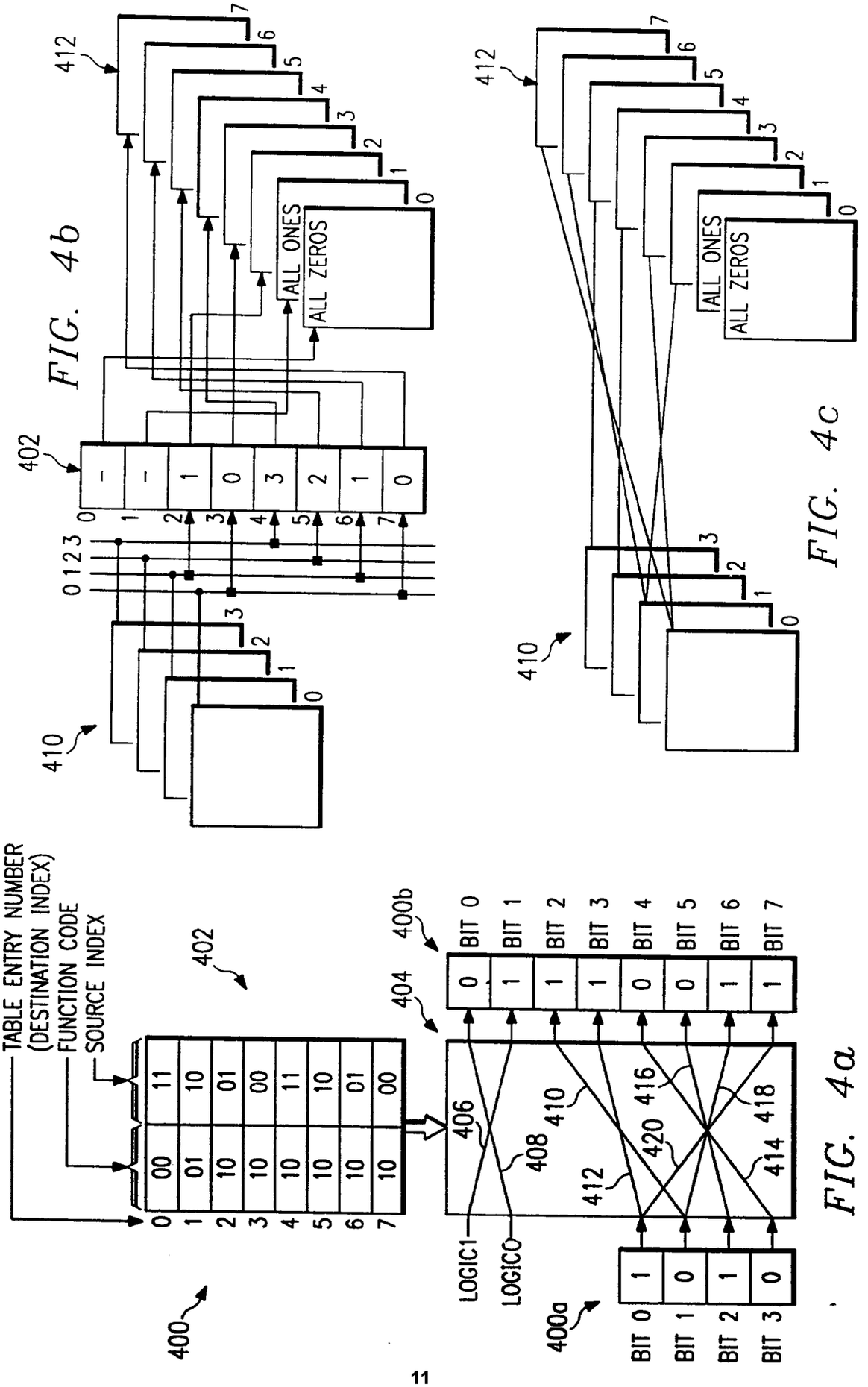


FIG. 3


400
TABLE ENTRY NUMBER (DESTINATION INDEX)
FUNCTION CODE
SOURCE INDEX
402
00 01 10 10 10 10 10 10
11 10 01 00 11 10 01 00
0 1 2 3 4 5 6 7
400b
404
BIT 0 BIT 1 BIT 2 BIT 3 BIT 4 BIT 5 BIT 6 BIT 7
0 1 1 1 0 0 1 1
406 408 410 412 414 416 418 420
LOGIC1 LOGIC0
400a
BIT 0 1
BIT 1 0
BIT 2 1
BIT 3 0
FIG. 4a
412
ALL ONES
ALL ZEROS
410
FIG. 4c
412
ALL ONES
ALL ZEROS
402
– – 1 0 3 2 1 0
0 1 2 3 4 5 6 7
0 1 2 3
410
FIG. 4b

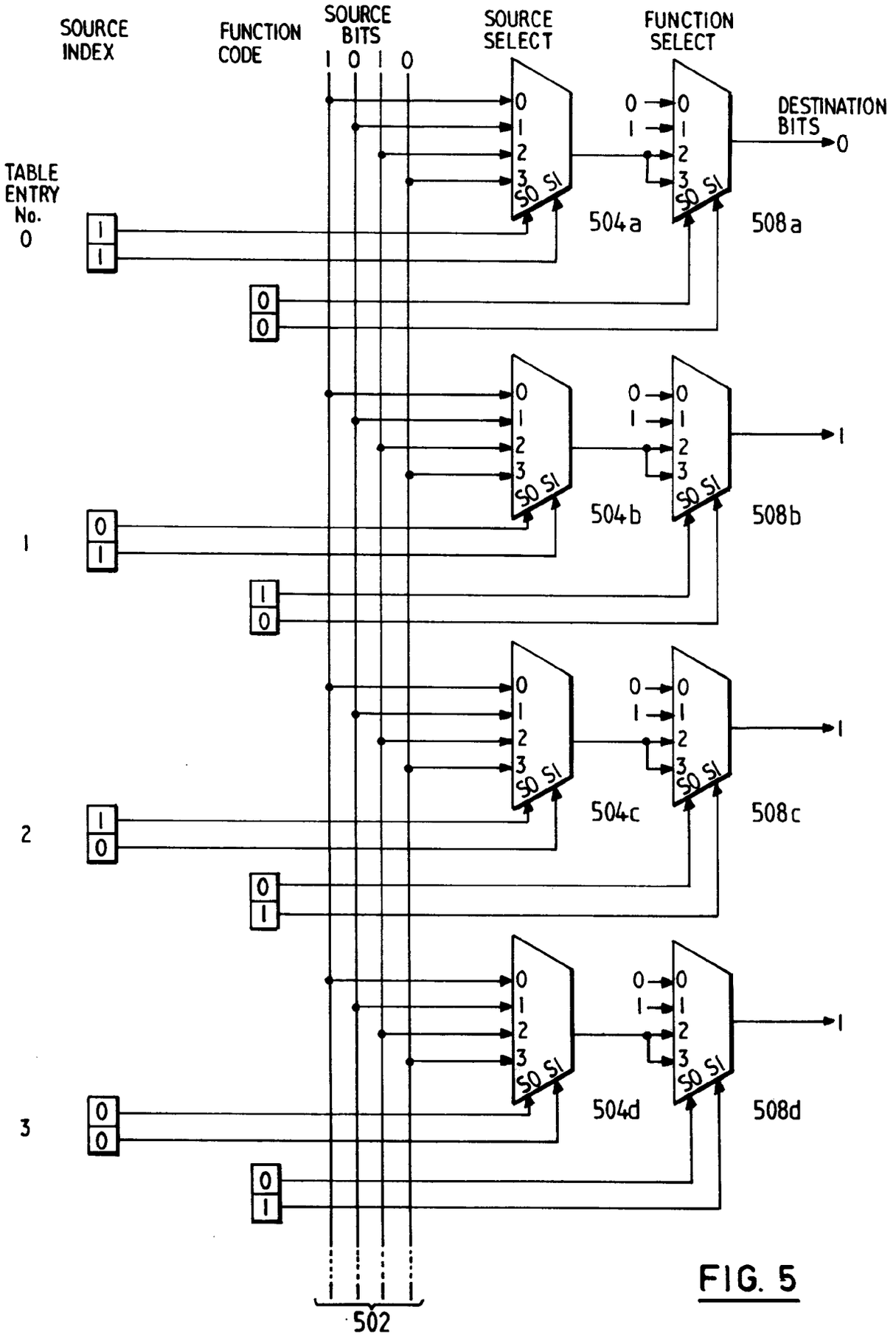

SOURCE INDEX
FUNCTION CODE
SOURCE BITS
1 0 1 0
SOURCE SELECT
FUNCTION SELECT
DESTINATION BITS
TABLE ENTRY No.
0
1
2
3
504a
504b
504c
504d
508a
508b
508c
508d
SO SI
502

FIG. 5

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 30 9871

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	EP-A-0 410 777 (TEXAS INSTRUMENTS INC.) * Abstract * * column 10, line 41 - line 48; figures 5,8 *	1,8,12	G09G5/02 G09G1/16
A	GB-A-2 234 096 (APPLE CO. INC.) * Abstract * * page 10, line 18 - page 11, line 12; figures 1,3,4 *	1,8,12	
A	IBM TECHNICAL DISCLOSURE BULLETIN., vol.33, no.3A, August 1990, NEW YORK US pages 145 - 152 'Data Width and Format Conversion Subsystem for a Graphics Coprocessor' * whole article *	6,8,12	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G09G

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	8 May 1995	Corsi, F

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)