TECHNIQUES FOR FRAME BUFFER ANIMATION

Kellogg S. Booth
Stephen A. MacKay

Computer Graphics Laboratory
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
(519) 886-1351

ABSTRACT

Hardware features widely available on current frame buffer systems offer a surprising variety of techniques for limited animation. Properly used, they provide the illusion of real-time animation for a large class of practical applications. As hardware prices continue to drop we may expect these same features to be available on low-cost raster display terminals. Colour lookup tables, zoom-pan-scroll registers, selective memory write, and crossbar switching on data paths all invite imaginative solutions to problems of limited computation and transmission. Used singly or in combination they have important implications for videotex systems such as Telidon. We examine these techniques and make some suggestions for future hardware.

KEYWORDS: animation, colour lookup table, crossbar switch, frame buffer, Telidon, zoom-pan-scroll.

EXTENDED ABSTRACT

Background

A number of manufacturers offer digital frame buffers consisting of a large chunk of memory (512x512 or 1024x1024 pixels, each with 8 to 32 bits), a video generation system to display the entire memory 30 or 60 times a second, some type of interface to a host computer and frequently a dedicated microprocessor (conventional or custom bitslice) directly coupled to the frame buffer. These systems always have some type of colour lookup table, usually a set of mapping registers to control zoom, pan and scrolling of images, and (less often) some additional hardware features which enhance the basic memory planes by providing cursor and overlay planes. Our interest here is to pinpoint those aspects of the architecture which can most easily be exploited to provide the appearance of full animation. We will illustrate our discussion with a specific hardware configuration, an Ikonas 2000 Raster Display System, but the same principles apply to many other commercially available frame buffer systems.

Ron Baecker has already championed the proposition that dynamic graphics is no longer the sole domain of calligraphic systems, but he observed that "current frame buffers are unsuited to dynamic imagery" because of the requirement, inherent in frame buffer architecture, that changes to the image require explicit modification of pixels in memory [1, page 54]. The amount of computation required for major changes, coupled with the need to synchronize those changes with video generation have precluded full real-time animation.

Texts by Foley and van Dam [4] and Newman and Sproull [7] explain basic frame buffer architecture. For our purposes it is sufficient to regard frame buffers simply as large memories having two-dimensional (x,y) addresses. We will be looking entirely at static images, in the sense that the contents of memory pixels will never change. Dynamics are added by modifying the pattern in which pixels are read from memory and the way in which their contents ("bits") are interpreted to provide colour information for the analog circuitry. Most frame buffers perform the latter task using a colour lookup table.

A lookup table consists of a set of high-speed registers (many frame buffers actually have two sets operating in parallel to achieve video rates) which perform a mapping from pixel values stored in image memory to the digital colour levels sent to the D/A converters. Originally, lookup tables were added to frame buffers for gamma correction (compensation for non-linearities in hardware and in human perception) and to provide pseudo and false colour for image processing. Sloan and Brown [9] provide a very good survey of the innovative uses to which colour tables have subsequently been put.

This work was supported by the Natural Science and Engineering Research Council of Canada under NSERC grant No. A4037. The paper was typeset by Waterloo Computer Typography on an APS-μ5 using software developed by Richard J. Beach.

As each pixel is accessed during the display of a single frame, the various bit planes are read to form three 8-bit colour numbers, one each for red, green and blue (RGB). These numbers then act as indices into the red, green and blue colour maps from which the actual digital intensities are read. This is indicated schematically in Figure 1. Random access to the lookup table registers requires that they be extremely fast. In a 1,000-line display the colour tables are accessed once every 30 nanoseconds. This is substantially higher bandwidth than for pixel memory, but there are far fewer registers in the look-up table (typically 256 to 1024 entries) so the incremental cost is relatively low.

Dick Shoup is generally credited with the insight for using a colour lookup table to provide a limited form of frame buffer animation. His tutorial paper is an excellent overview which we will only summarize here. Shoup makes the following observation in the introductory section of that paper which provides our point of departure.

> Fortunately, even motion which is graphically very simple can produce a vastly more effective visual communication than a still image. In short, *a little animation goes a long way*. If we are willing to forego full Disney-style animation and strongly limit the content and dynamics of our images, then much simpler, highly interactive solutions can be found [8, p. 9].

Colour table animation is only one technique for achieving dynamics with a static pixel memory. We will examine other techniques based on the zoom, pan and scroll features of a frame buffer and on the use of crossbar switches at various points in the data paths between pixel memory and the D/A converters.

Colour Table Animation

Shoup's taxonomy of colour table animation includes colour cycling, alternate-colour animation, and bit plane extraction. (Shoup does not use the term bit plane extraction, but includes it as an extension of alternate-colour animation; we prefer to consider it separately because it is an important idea.) Each technique treats values stored within pixels differently, the interpretation depending upon the particular intensities stored within the lookup table.

Colour cycling is achieved by rotating entries in the look-up table. This gives the illusion of motion as colours appear to "flow" across the screen. Variations that cycle only some of the colours, or that independently cycle various subsets of the entries often at different rates, are also possible.

Alternate colour animation paints more than one image into the frame buffer memory using different pixel values for each image. Setting the lookup table entries to background for all but one of the images to background allows a single im-

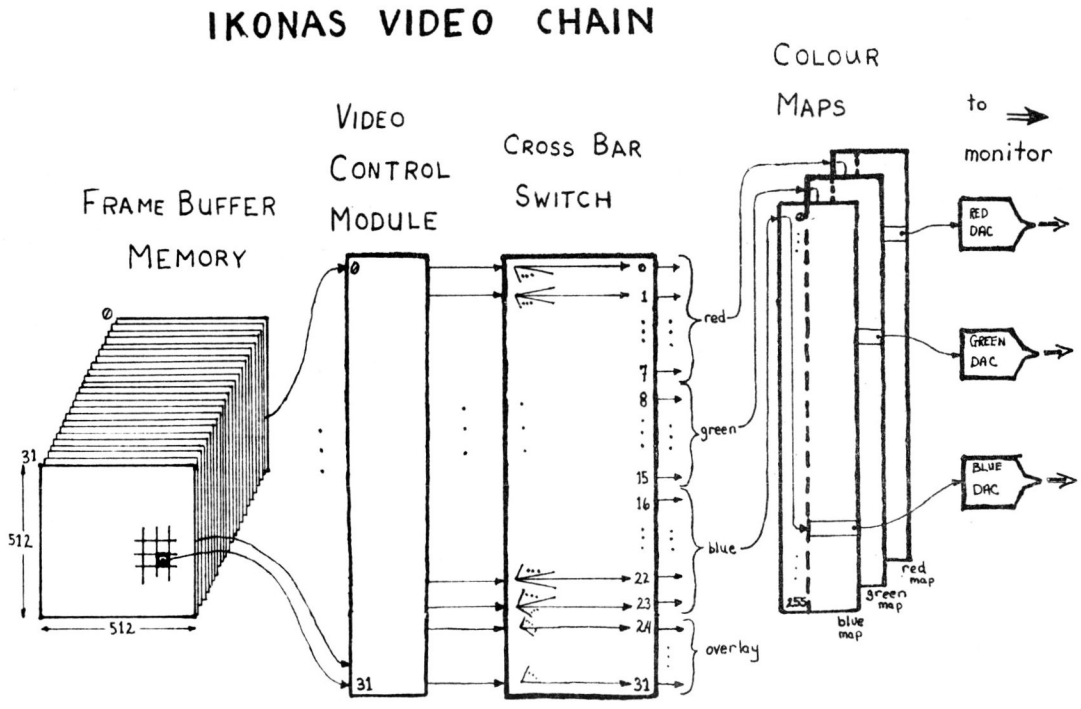

Figure 1

age to be viewed. Changing the subset of "visible" colours alternately displays one and then another of the images.

Bit-plane extraction assigns to each overlap region a unique colour. A simple way to achieve this is to partition the frame buffer into bit planes (we are now thinking of a pixel in memory as a string of bits, one in each plane) and assign each image to a separate set of bit planes. Loading the lookup tables to "ignore" all but the bit planes corresponding to a particular image allows images to be displayed separately. It is possible to select more than one image for simultaneous display. This can be used to mix images, overlay one with another using a priority scheme, or even to add "transparency" by allowing one image to show "through" another.

The advantage of these techniques over a more traditional approach in which images are recreated within the pixel memory as they are displayed is two-fold. No computation time is required to modify pixel memory, only the lookup table entries are changed. Equally important, there is little or no data transmission from the host because the various settings for the lookup table can be stored within unused locations of the pixel memory or within memory associated with the microprocessor. The end result is that the illusion of full animation can be created, sufficient for a large number of practical applications. These same advantages can be achieved using other hardware features common in frame buffer architectures.

The Video Chain

Many frame buffers have a video chain similar to that in Figure 1. Pixel values are read under control of a video module that determines the (x,y) addresses in the pixel memory. These depend upon the viewport and windowing parameters that select a window within the frame buffer and a corresponding viewport on the monitor. Additional pixel replication or zoom factors for both the horizontal and vertical directions are used to scale the window inside the viewport. Pixel values are then routed through a crossbar switch which may shift, permute or simply ignore various bits within the numbers to produce the three 8-bit values actually sent to the colour lookup tables for red, green and blue. These then index into the true digital intensities which are subsequently convert-

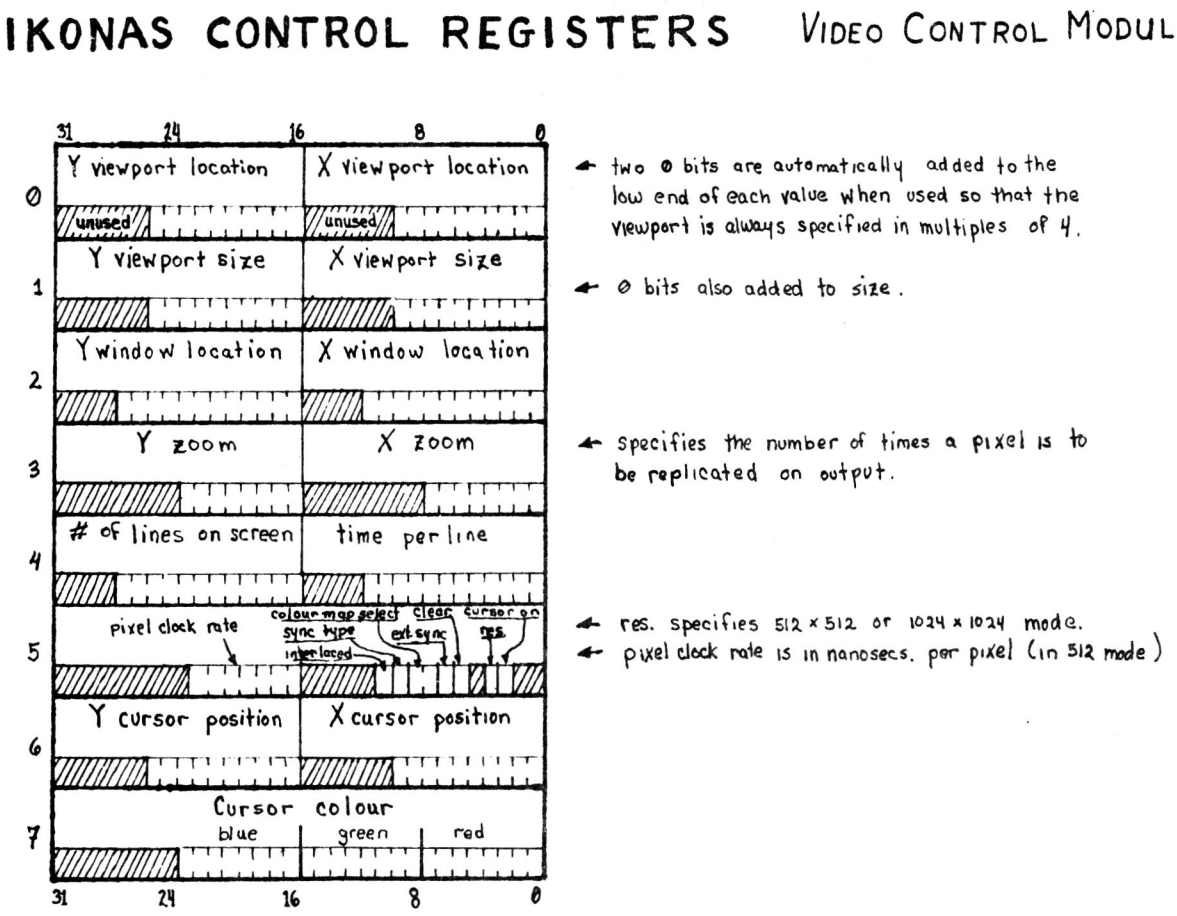

Figure 2

ed to analog signals (possibly with a final stage of gamma correction) that drive the three electron guns in a colour monitor.

A schematic representation of the video control module is shown in Figure 2. Register 0 specifies the (x,y) coordinates of a position on the 512x512 surface of the display monitor which will correspond to the upper lefthand corner of the image. Register 1 specifies the size (number of pixels) to be displayed both vertically and horizontally. Oversize viewports wrap around. A window within the pixel memory is specified by Register 3 which is the (x,y) address of the first pixel value to be displayed in the upper lefthand corner of the viewport. Finally, Register 4 provides a replication count which specifies the number of times that each memory pixel is to displayed, effectively providing independent zoom factors for both the vertical and horizontal directions. The remaining registers control the timing associated with each pixel display, the scan rate of the electron guns, and various other features including the (x,y) position and colour of a crosshair tracker.

Each step in the video chain can be used to effect animation. Lookup tables partition pixel values into displayed and non-displayed colours. Zoom-pan-scroll animation partitions the pixels themselves. Crossbar animation partitions bits within pixels to achieve yet a third effect. In each case we capitalize on the fact that only a small number of special registers need be modified, rather than large numbers of pixels in memory.

Zoom-Pan-Scroll Animation

The simplest zoom-pan-scroll animation exists when the pixel memory is logically divided into different regions, each containing a separate, lower-resolution picture. Figure 3 illustrates the case in which four 256x256 images are stored within a 512x512 frame buffer. By setting vertical and horizontal zoom factors of two with a 512x512 viewport, the display can cycle through the four images by successively setting the window location to the upper lefthand pixel of each subimage.

One advantage of this approach over lookup table animation is that each pixel maintains a full 24-bit colour value instead of sacrificing palette size for enhanced dynamics. This is clearly a tradeoff between spatial resolution and intensity resolution. For the particular example of 256x256 subimages it is interesting to note that for a large class of pictures the

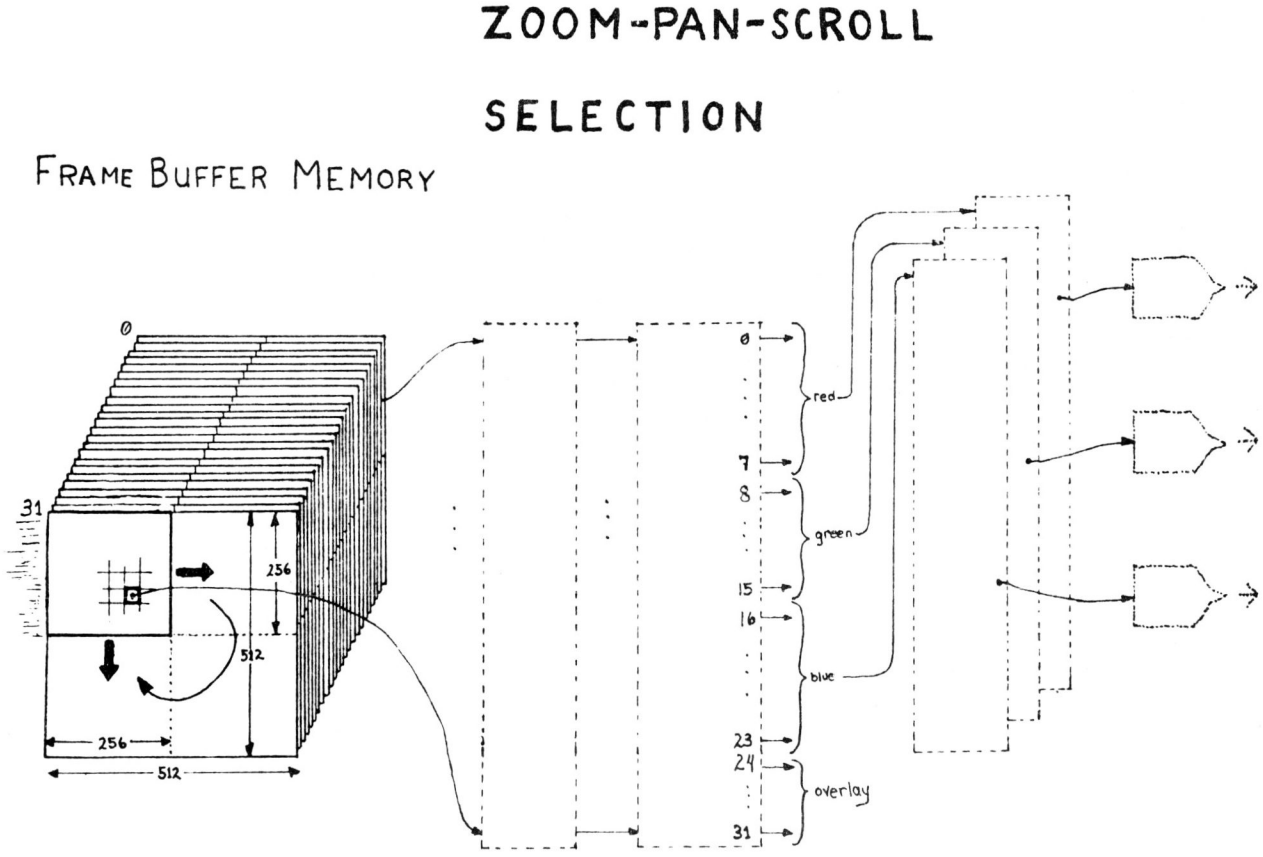

Figure 3

apparent resolution of a 256x256 image in which multiple intensity levels are used for antialiasing will be superior to that of a 512x512 rendering of the same scene with no antialiasing.

The zoom-pan-scroll technique can be extended to even more subimages, at the expense of further reduction in spatial resolution, as illustrated in Figure 4 in which sixteen different views of an Imperial Walker are shown. Animating the images produces a fairly realistic effect, somewhat diminished by the low-quality of the 128x128 subimages. In this case a combination of zoom-pan-scroll with 256x256 images, 4 bits of antialiasing, and eight levels of bit plane extraction provide equivalent animation but with much improved image quality. Additional hardware is needed to achieve this.

Crossbar Animation

The Ikonas system has a full crossbar switch which will route any of the 32 bits from pixel memory to any of the 24 input lines of the lookup tables. The original idea for the crossbar switch is due to Henry Fuchs [5]. In addition to rerouting the inputs, the crossbar switch is also capable of ignoring input and passing a one bit through to any output. The crossbar switch is set by loading each of 24 registers with a number between 0 and 32, the latter indicating an ignored input.

With the crossbar switch in the video chain all of the 32 bit planes can be routed to any of the lookup table inputs. One common configuration is to use the crossbar switch to successively choose 8 bit planes, allowing a double-buffered display. Figure 5 illustrates this double buffering for a z-buffer algorithm in which the high-order 16 bit planes are dedicated to storing depth information and the low-order bit planes constitute two intensity buffers.

With a 32-bit frame buffer we can store four 8-bit images, eight 4-bit images, sixteen 2-bit images or even thirty-two 1-bit images. Crossbar animation coupled with zoom-pan-scroll and colour table animation is a particularly effective combination.

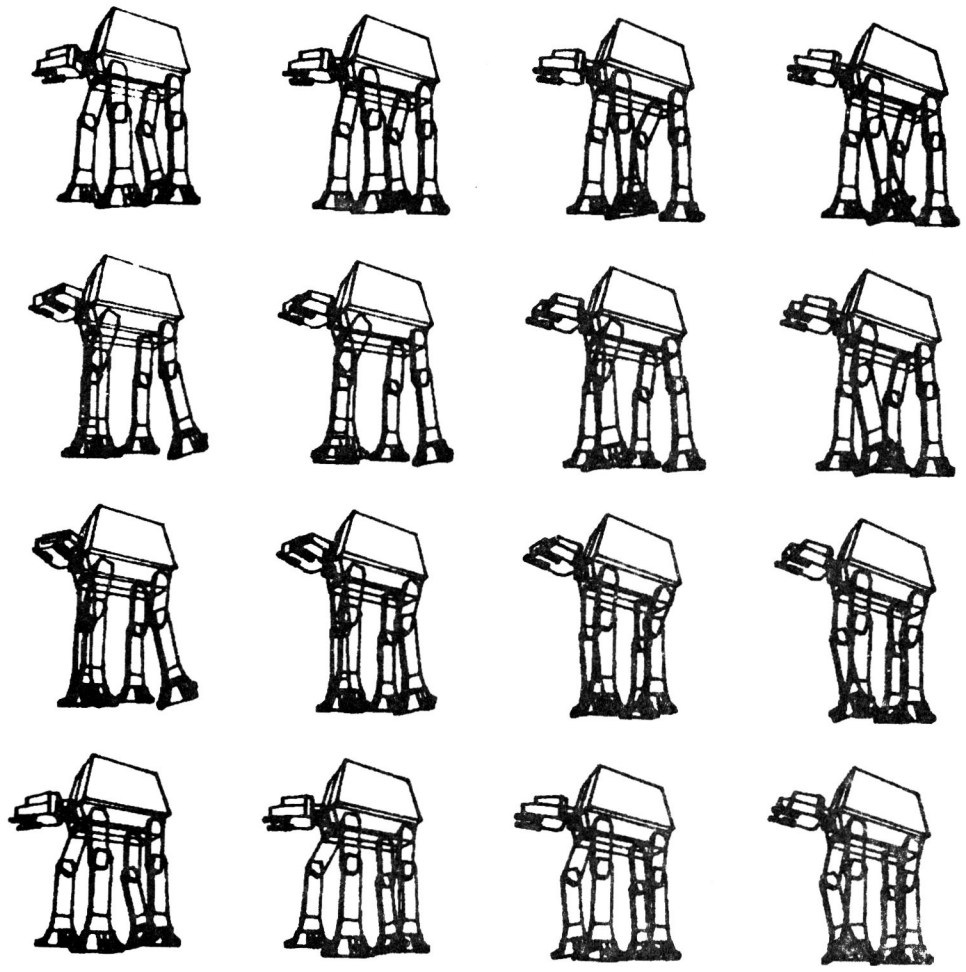

Figure 4

Figure 6 illustrates the crossbar switch settings for displaying a 4-bit image using a sixteen entry colour table.

Other Hardware Considerations

So far we have only been concerned with the display of static images within the pixel memory. A significant consideration remains; we must still scan convert the images into pixel memory. Fortunately the complexity of this task is greatly simplified by some further features found in many frame buffers.

A *write mask* (z mask) is a register that controls which bits within a pixel change during a write to the frame buffer. This substantially cuts down on the overhead of producing pictures for colour table or crossbar animation. Normally either the host or the frame buffer's microprocessor would have to perform a read-modify-write operation to insert only those bit planes involved with a particular subimage. In many cases this is quite expensive when performed on a host, and even on special purpose bitslice microprocessors it can still be significantly slower than simple writes. Setting the write mask to select only those bit planes being written allows the host or microprocessor to proceed as if the other bit planes were not present.

This arrangement is deficient in two respects. The first is that the write mask is implemented as part of the pixel memory, each bit plane inhibiting itself when not selected. The problem with this approach is that all accesses to the frame buffer use the same write mask. The mask thus becomes a shared resource between the host and the microprocessor whose use must be synchronized. The overhead associated with this coordination can easily be more than that of performing a read-modify-write cycle in software. Instead there should be a separate write mask associated with each processor connected to the frame buffer so that every bus cycle can potentially make a different selection of bit planes.

Our second suggestion for improving the write mask is to add two additional mask registers, one which selects those bits to receive a "standard" setting and a second with the values for the standard settings. A write to pixel memory consists of modifying only those bit planes selected by one of the mask registers, those in the first mask receiving the data from the host or microprocessor, those in the second mask receiving the standard settings.

A second crossbar switch should sit between the processor and the pixel memory (actually, one crossbar switch for *each* processor if there is more than one, avoiding the resource allo-

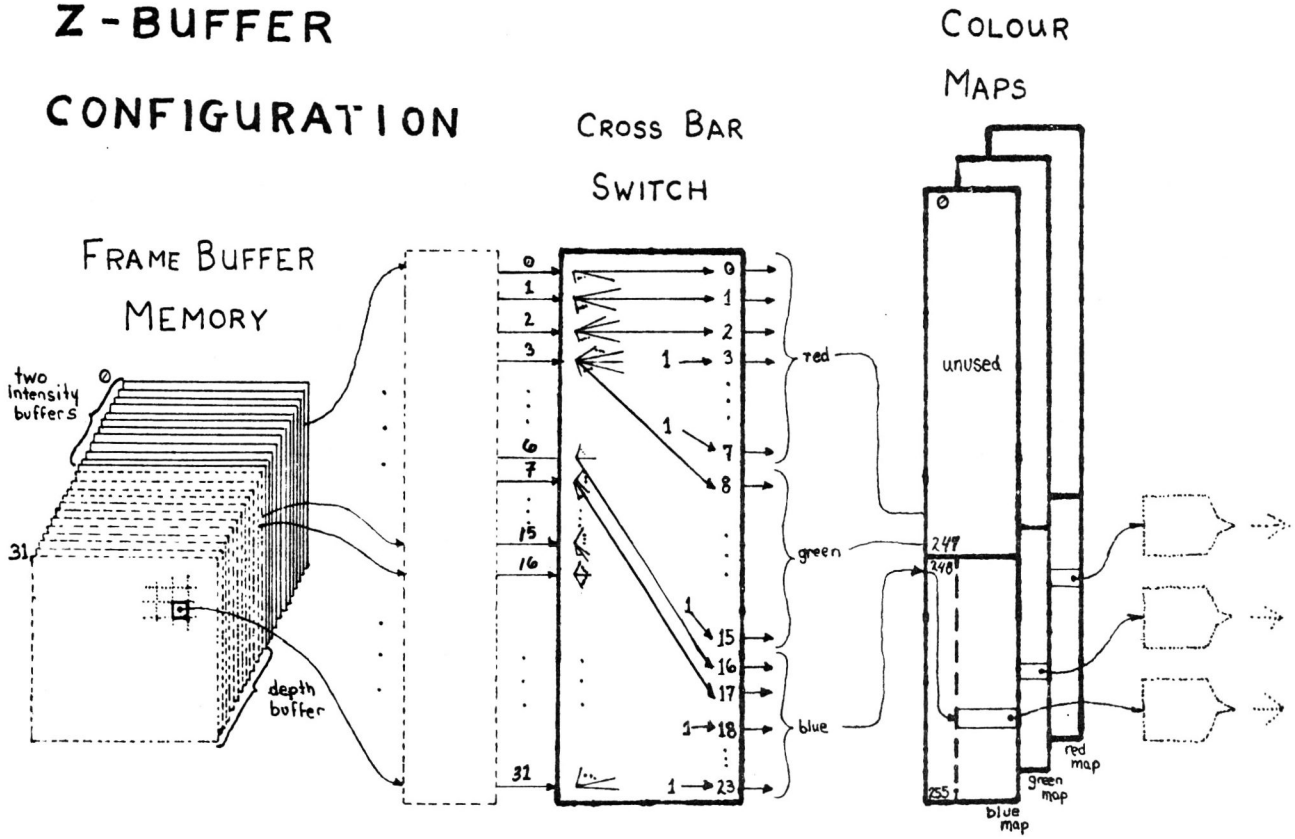

Figure 5

cation problem previously mentioned for the write mask) so that writes as well as reads can be rerouted. This again is to simplify the software. In the case of a z-buffer algorithm, double buffering becomes transparent to the scan conversion routines because the current buffer always appears to be in the same bit positions. A similar advantage obtains for the bit plane extraction technique.

In addition, all of the crossbar switches should include "constant" outputs of both 0's and 1's and maybe even the ability to select input bits from the current (x,y) address (at least from the low-order bits, which are convenient for generating textured patterns).

Some of these features are special cases of a more general bus structure which allows a number of operations to be performed as pixels are written in memory. Frank Crow describes a particular implemention . [3].

Two additional points are worth making. They concern the zoom-pan-scroll registers. The first is a simple observation regarding limitations on smooth or "Hollywood" scrolling. Both the pan and scroll registers operate only in multiples of four pixels. This is due to the way that pixel memory is organized. Unfortunately this produces discrete rather than continuous pan and scroll which is perceived as being "jumpy". Were this problem rectified, the same effect would still be present when the window is zoomed because each pixel is always replicated the specified number of times. This can be overcome by adding two new zoom registers which contain vertical and horizontal replication counts for only the *first* pixel on each scanline. If the replication count is limited to at most 256 (represented as 255 within the register) all four counts could be stored within the current 32-bit Register 3.

Our final observation on zoom-pan-scroll is that it would be particularly convenient to have independent control over each bit plane. To our knowledge the only frame buffer implementing this idea is the Norpak VDP-1. It has separate zoom, pan and scroll registers for each bit plane. This allows portions of the picture to be moved relative to each other providing an animation capability not available with any of the previously mentioned techniques. Unfortunately the later VDP-2 controls four bit planes with a single register (another concession to memory organization). This is still useful, but would be more useful if a crossbar switch were also present.

Implications for Telidon

It is our contention that future videotex systems such as Telidon must incorporate at least some of these features if they

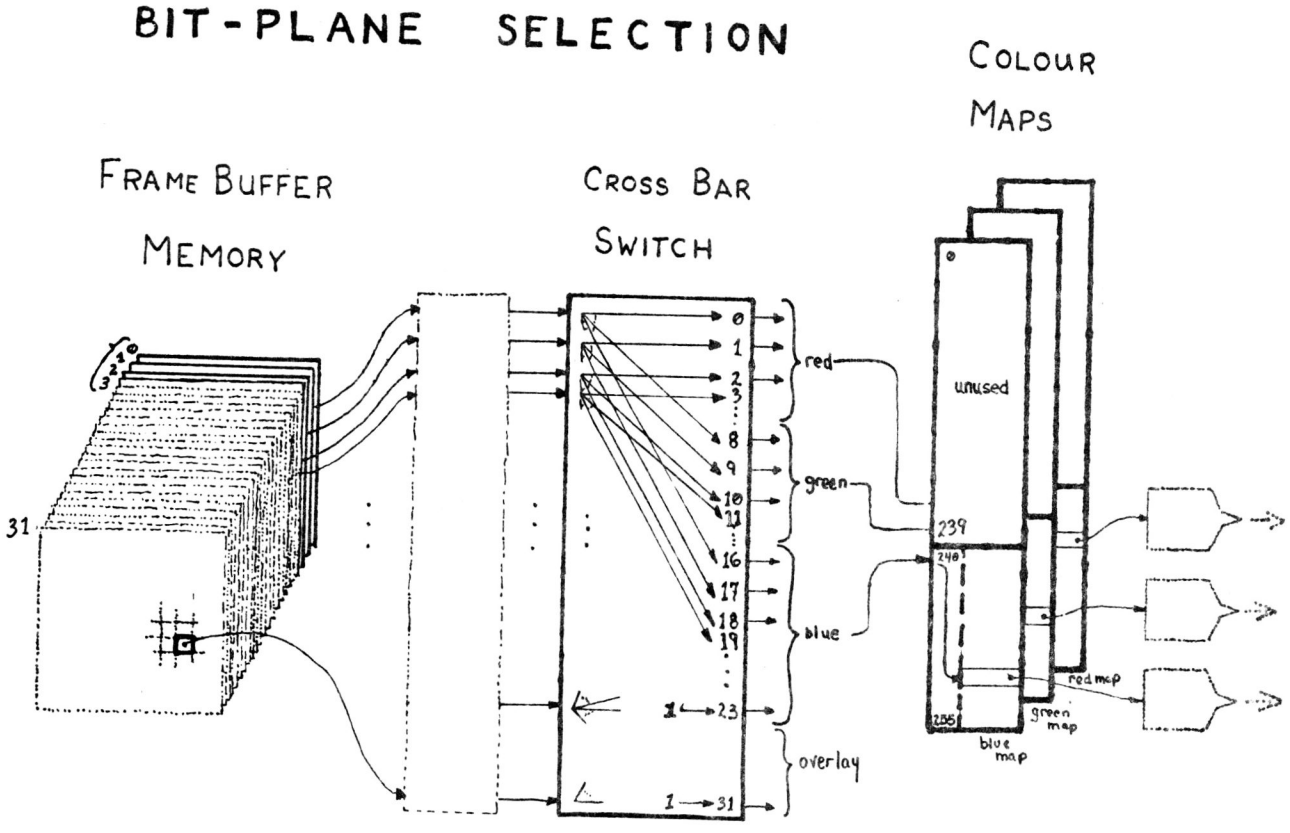

Figure 6

are to become competitors in the information providing market. The steady drop in hardware costs will soon make many of the features cost-effective, especially in the projected mass markets for videotex terminals. We expect that full 24-bit frame buffers will remain beyond the price range of the average consumer for some time to come, but that modest extensions to include colour lookup tables and zoom-pan-scroll registers are easily within today's technology.

Current Telidon terminals already suffer from needlessly limited palettes because they lack colour lookup tables. The poor choice of colours now available (there are *two* different ways to specify both black and white!) precludes any attempt at antialiasing images or at displaying full colour images. Work at MIT's Architecture Machine Group has already demonstrated that with a proper choice of palette a relatively small number of colours (no more than 256) can achieve a very realistic imitation of a full 24-bit system [6].

Frame buffer animation techniques are useful not only for their limited animation capability, but also for providing the interactive feedback necessary if Telidon is to become a truly two-way system. Instantaneous feedback for menu-driven systems can easily be achieved using colour table or crossbar animation techniques. A variety of projects within the Computer Graphics Laboratory at Waterloo are investigating other uses for these techniques within the Telidon environment.

References

[1] R. M. Baecker, Digital video display systems and dynamic graphics, (Siggraph '79 Conference) *Computer Graphics Quarterly 13:2* (August 1979) pp. 48-56.

[2] H. G. Bown, C. D. O'Brien, W. Sawchuk and J. R. Storey, *A General Description of Telidon: A Canadian Proposal for Videotex Systems*. CRC Technical Note No. 699-E, Communications Research Centre, Canadian Department of Communications, Ottawa (December 1978).

[3] F. C. Crow and M. W. Howard, A frame buffer system with enhanced functionality, (Siggraph '81 Conference) *Computer Graphics Quarterly 15:3* (August 1981) pp. 63-69.

[4] J. D. Foley and A. van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley (1982).

[5] H. Fuchs, personal communication.

[6] A. Lippman, Computational video: Television as a human interface, *Proceedings of the 7th Canadian Man-Computer Communications Conference* (June 1981) pp. 133-137.

[7] W. M. Newman and R. F. Sproull, *Fundamentals of Interactive Graphics*, McGraw-Hill (1979).

[8] R. G. Shoup, Colour table animation, (Siggraph '79 Conference) *Computer Graphics 13:2* (August 1979) pp. 8-13.

[9] K. R. Sloan, Jr. and C. M. Brown, Colour map techniques, *Computer Graphics and Image Processing 10:4* (August 1979) pp. 297-317.