A mapping and memory chip hardware which provides symmetric reading/writing of horizontal and vertical lines

by D. L. Ostapko

This paper describes a mapping and memory chip hardware for enhancing the performance of an APA display. The approach describes a modification to the primary port of a quasi-two-ported memory. This modification allows several contiguous horizontal or vertical bits to be read or written in one cycle. The number of bits that can be stored is given by the number of memory chips. The hardware modifications can be on or off chip, and if on chip, the chip can still be used as a conventional memory chip. Simple modifications to the hardware will support different screen sizes.

Introduction

Following the adage that a picture is worth a thousand words, considerable effort has been put forth in developing pleasing electronic displays. One approach that shows considerable promise in providing arbitrarily complex, flicker-free images is the All-Points-Addressable (APA) display. This type of display uses a random access memory to store the image information point by point. The image data contained in the memory can then be used to refresh the display screen at a frequency that is adequate to prevent flicker. Matick et al. [1] give a good summary of APA displays as well as several proposed memory chip modifications. The memory chip proposed in that paper provides the basis for the chip described in this paper. The usual limitation of an APA display is the bandwidth of the memory subsystem. The chip proposed in [1] contains a quasi-second port as well as modifications to the primary port. This paper describes a mapping and additional hardware that further improve the memory bandwidth by allowing a single memory access to modify a series of pixels in either the horizontal or vertical direction [2].

System description

In order to simplify the explanation of the mapping and the memory chip hardware, the particular configuration shown in **Figure 1** is assumed. Several modifications to this configuration are described later.

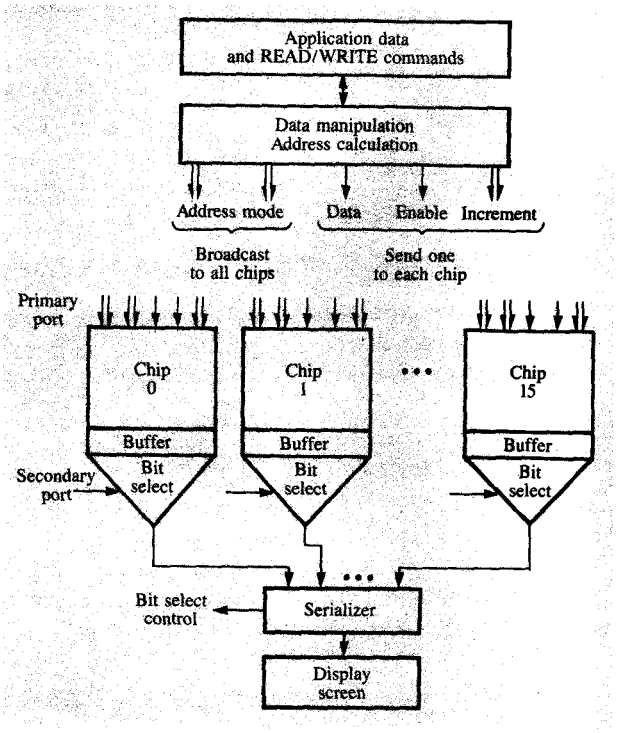

Figure 1

System configuration.

The generation of data and read or write commands is assumed to be provided by the application. Since the data width to the memory is 16 bits, the data must be presented as a series of read or write commands for horizontal or vertical strings having a maximum of 16 bits. Each command will also include the X, Y coordinates where the string is to begin on the display screen and a mask of length 16 indicating which of the 16 data items are to be stored or retrieved, so that after any required shifting the mask bits can be used directly as chip enable signals. Horizontal strings are stored to the right and vertical lines downward from the starting address X, Y, where X and Y are in the interval 0 to 1023. It is assumed that none of the addresses corresponding to bits in the strings extends beyond the maximum screen coordinates.

Data manipulation and address calculation is the process of determining the actual signals that must be sent to each memory chip. The data manipulation consists of a circular shift or rotation of the data and mask by an amount determined by the X, Y coordinates of the starting location on the screen. In addition to the data and mask bit, each memory chip must also receive a bit increment and a word increment signal. The bit address, word address, and enable signals are broadcasted to each chip. The transformations necessary to determine these signals are given in the next section. The importance of this mapping between memory locations and display screen locations is that any contiguous 16 horizontal or vertical bits on the screen are mapped to 16 different memory chips and can, therefore, be stored or retrieved in one memory cycle.

The 16 memory chips shown in Fig. 1 are assumed to be 64K-bit chips organized as 256 words by 256 bits. This provides enough storage for a 1024 by 1024-pixel display screen. These chips are assumed to have a quasi-second port with a 256-bit buffer and bit and word increment logic on the primary port similar to that described in [1]. In the following text, it is assumed that bit and word addresses are provided simultaneously, which simplifies the description of the required on-chip logic. If they are multiplexed, a single modified incrementer can be used. Each chip will also receive a 4-bit code defining the chip number. This code may be either on or off chip.

The purpose of the serializer is to take the appropriate bits from the chip buffers and generate the data stream that is used to control the intensity of the beam being scanned across the screen. In this application, the first bit of all buffers is sent to the screen followed by the second bit, etc. The proposed chip requires a modification to the serializer logic, which is described later.

Screen to memory mapping

Figures 2(a) and **2(b)** show the mapping of the screen image onto the memory chips. It is seen that each bit of a horizontal string of length 16 or less is placed into a different chip. Each is placed in the corresponding location of that chip except for the bit increment for those bits that are separated from the starting point by an even byte boundary, where an even byte boundary is a coordinate that is a multiple of 16. The mapping of 16 vertical bits is somewhat more complicated. For vertical lines, each bit is placed in the corresponding location of different quadrants of different words. The rule is that the quadrants increase by 1 modulo 4 and the word address increases by 1 each time the quadrant becomes 0. This sequencing through quadrants and words is best described by associating a 4-bit quantity with a position in the vertical line, where the two low order bits define the quadrant and the two high order bits define the two low order bits of the word address. Sequencing through the locations then corresponds to sequencing through the consecutive numbers 0 to 15. Because of this sequencing, the word address must be increased by four for those bits in the string that are separated from the starting point by an even byte boundary. The important property of the mapping is that 16 horizontal or vertical bits can be written to memory in one cycle because no two bits are mapped to the same chip.

Although 16 horizontal or 16 vertical bits can be mapped from/to the memory in one read/write cycle, it is also important to determine the computational and hardware complexity of the mapping. The complexity is manifested in the data manipulation and address calculation process and in the hardware and signal lines for the memory chip and the serializer.

Address calculation and data manipulation

The bit and word addresses that are to be broadcasted to the memory chips must be generated from the X, Y coordinates of the starting point of the line. Since the horizontal and vertical modes differ, each must be considered separately.

- *Horizontal mode*

From Fig. 2, it is seen that the 6 low order bits of the bit address are given by the integer part of X divided by 16 and the 2 high order bits are given by Y modulo 4. Thus,

Bit address (7, 6, 5, 4, 3, 2, 1, 0) = $Y(1, 0), X(9, 8, 7, 6, 5, 4)$.

The word address is given by the integer part of Y divided by 4. Thus,

Word address (7, 6, 5, 4, 3, 2, 1, 0) = $Y(9, 8, 7, 6, 5, 4, 3, 2)$.

Figure 2 shows that incrementing the coordinates of a point by one unit in either the horizontal or vertical direction moves the point to the next chip. Therefore, the chip number that contains a location X, Y is given by $(X + Y)$ modulo 16. This quantity is used to define the amount of right circular shift that must be applied to the data. Thus, the properly aligned data and enable signals are given by

Data = $((X + Y)$ modulo 16) shift of input data

and

Enable = $((X + Y)$ modulo 16) shift of input mask.

Since every even byte is placed in a different bit position, the number of chips that receive a chip increment is given by $(X + 16)$ modulo 16 or X modulo 16. This vector of $(16 - X$ modulo 16) number of 0s followed by $(X$ modulo 16) number of 1s must also be shifted by $(X + Y)$ modulo 16 as were the data and enable signals. Thus,

Bit increment = $((X + Y)$ modulo 16) shift of $(16 - X$ modulo 16) number of 0s concatenated with $(X$ modulo 16) number of 1s.

Since all horizontal lines are in the same word,

Word increment = 16 number of 0s.

In order to inform the memory chips that the reading or writing mode is horizontal, the mode signal is used and

Mode = 0.

Any other signals such as those necessary to distinguish between read and write, and between primary and secondary ports, and any multiplexing signals that may be used are not described.

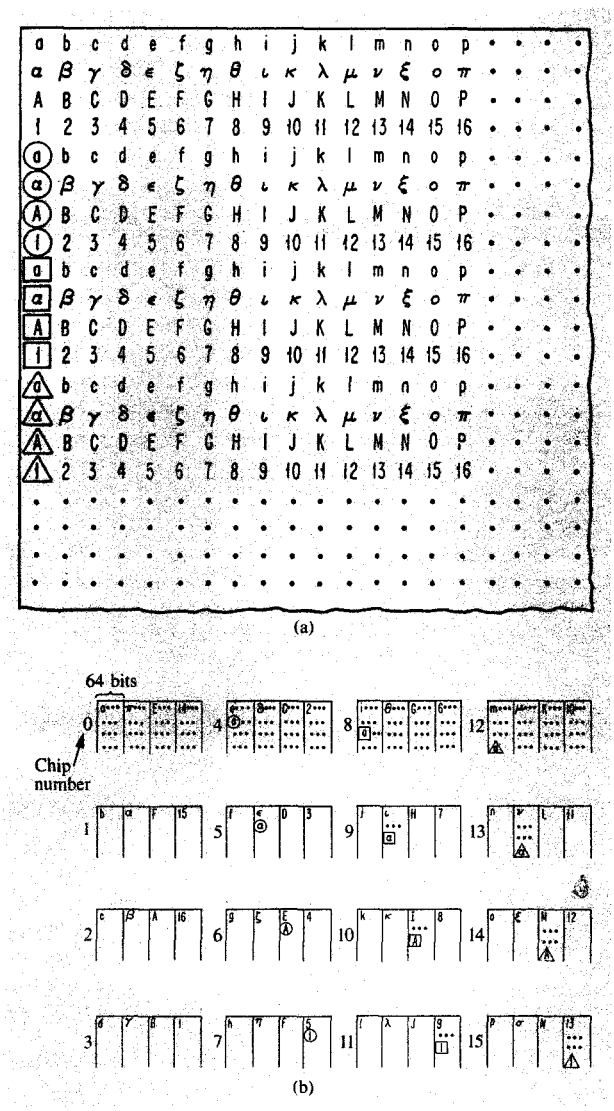

Figure 2

(a) Display screen image, (b) memory representation.

- *Vertical model*

The address of the first bit in the vertical line is the same as that calculated for the horizontal mode. However, this bit is to be placed in chip number $(X + Y)$ modulo 16. In the vertical mode, the on-chip hardware adds the chip number to the 2 low order bits of the word address concatenated to the 2 high order bits of the bit address. Before the address

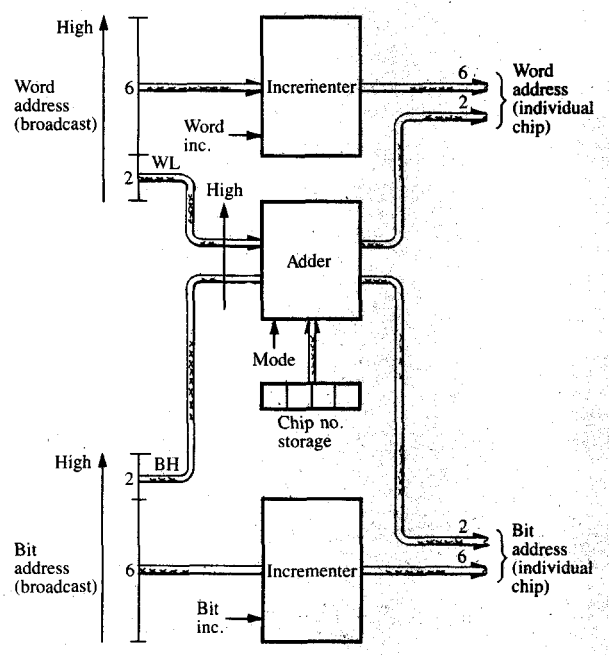

Figure 3

Additional logic for memory chip.

can be broadcasted to all chips, $(X + Y)$ modulo 16 must be subtracted from the appropriate portion of the address. The 2 high order bits of the bit address must be decreased by $((X + Y)$ modulo 16) modulo 4 or $(X + Y)$ modulo 4. Thus,

$$\text{Bit address } (7, 6) = Y \text{ modulo } 4 - (X + Y) \text{ modulo } 4 = -X \text{ modulo } 4.$$

The 6 low order bits of the bit address remain unchanged; thus,

Bit address = $(-X$ modulo 4), $X(9, 8, 7, 6, 5, 4)$.

The 2 low order bits of the word address are modified by subtracting the integer part of $(X + Y)$ modulo 16 divided by 4. Thus,

$$\begin{aligned}\text{Word address } (1, 0) &= (\text{integer of } Y/4) \text{ modulo } 4 \\ &\quad - \text{integer of } ((X + Y) \text{ modulo } 16)/4 \\ &= (\text{integer of } Y/4) \text{ modulo } 4 \\ &\quad - (\text{integer of } (X + Y)/4) \text{ modulo } 4 \\ &= (\text{integer of } -X/4) \text{ modulo } 4.\end{aligned}$$

The 6 high order bits of the word address remain unchanged; thus,

Word address = $Y(9, 8, 7, 6, 5, 4)$, (integer of $-X/4$) modulo 4.

The expressions for the data and enable signals also remain unchanged. Thus,

Data = $((X + Y)$ modulo 16) shift of input data

and

Enable = $((X + Y)$ modulo 16) shift of input mask.

All vertical lines have the same bit address before modification by the chip number. Thus, the bit increment that is broadcasted is given by

Bit increment = 16 number of 0s.

Vectors of bits in the vertical direction are placed in successive words as a result of the modification by the chip number. Thus, the number of chips that receive a word increment of 4 is given by $(Y + 16)$ modulo 16 or Y modulo 16. The vector of $(16 - Y$ modulo 16) number of 0s followed by (Y modulo 16) number of 1s must also be shifted by $(X + Y)$ modulo 16. Thus,

Word increment = $((X + Y)$ modulo 16) shift of $((16 - Y)$ modulo 16) number of 0s concatenated with (Y modulo 16) number of 1s.

The mode signal that indicated that reading or writing is to be in the horizontal mode is

Mode = 1.

From the above equations, it is seen that the address calculations and data manipulations are relatively simple. It should be possible to implement them in either hardware or software.

On-chip address translation

The additional on-chip hardware must accept or store a 4-bit chip number and selectively perform a 4-bit addition depending on the mode signal. **Figure 3** shows the structure of this added logic.

The translation affects only the 2 low order bits of the word address and the 2 high order bits of the bit address. Let BH be the 2 high order bits of the bit address and WL be the 2 low order bits of the word address. The function performed is an addition of the 4-bit chip number to the four bits formed by concatenating WL and BH. The 4-bit result of the addition, ignoring carry out, defines the new WL and BH in the same order as in the input. The four bits that define the chip position or number can be either on-chip storage or signal pins that are connected to the appropriate signals.

It should be noted that the portion of the word and bit addresses that may be altered by their respective incrementers is disjoint from the portion that is modified by the 4-bit adder. Therefore, the incrementers can be removed or implemented off-chip without affecting the logic necessary to allow both horizontal and vertical reading/writing.

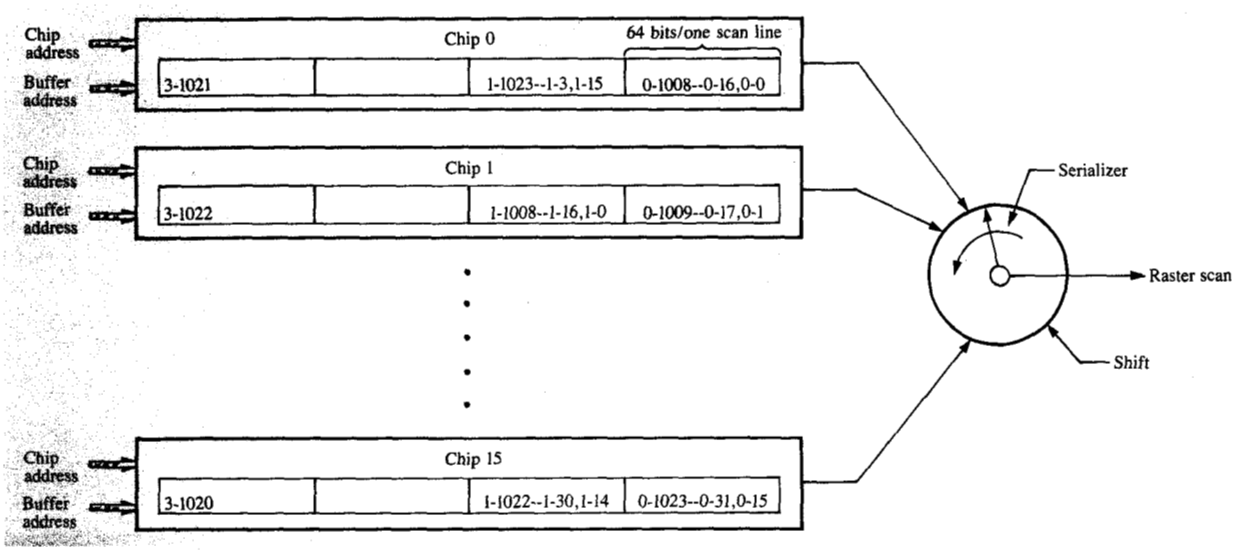

Figure 4

Memory to raster conversion.

Removing the function provided by either incrementer eliminates the bit addressability in that direction. Without bit addressability, each read/write must begin on an even byte boundary. Thus, the choice of bit addressability in both, one, or neither direction is possible as well as the choice of an implementation that is on-chip, off-chip, or multiplexed. It is also possible to implement the addition function off-chip. The added function does not prevent the usual operation of the memory chip in that the adder is completely disabled in mode 0.

Memory to raster scan conversion

In addition to algorithmically simple logic for address generation and data manipulation, and a small amount of additional on-chip hardware, it is also important that screen refresh be easily performed.

Figure 4 shows the scheme necessary for serializing the output bit stream. The scheme requires only one modification to the typical raster generation scheme. In a typical application, each of the 16 memory chips would load its 256-bit secondary port buffer using the same word address. From Fig. 2, it is seen that 4 scan lines are contained in the data stored in the 16 buffers.

The scan data are formed by taking 16 bits, 1 from each of the buffers in order, incrementing the bit location in all buffers, then taking another 16 bits until 4 scan lines have been generated. In the typical organization, the order in which the 16 bits are taken from the chips is constant and begins with chip number 0. The modification that is introduced is that the starting location within the chips is incremented by 1 modulo 16 after each scan line is completed. Since there are 4 scan lines stored per memory word, the scan origin will be 0 for every memory word address that is a multiple of 4.

Conclusions

This paper has presented a mapping and hardware modifications that allow symmetric reading/writing of horizontal and vertical lines from/to the refresh memory of an APA display. In addition to eliminating the asymmetric relation between reading/writing horizontal and vertical lines, the bandwidth to memory has effectively been increased. The extra hardware is modest and can be implemented in a variety of ways. If the extra logic is placed on-chip, the chip can still operate as a conventional memory chip.

In addition to being suitable for a 1024 by 1024-pixel display, small variations of the on-chip logic allow the screen size to be doubled in either direction. Doubling the screen size in either direction doubles the number of chips to 32 and the data path width to 32; doubling in both directions increases the number of chips and the data width to 64. The number of bits necessary to describe the chip number increases to 5 or 6. In addition, the on-chip adder increases to 5 or 6 bits. Doubling the X dimension of the screen increases to 3 the number of low order bits of the word

address that are passed through the adder. Doubling the Y dimension of the screen increases to 3 the number of high order bits of the bit address that are passed through the adder. Doubling in both directions increases both WL and BH to 3 bits. It should be possible to design efficient on-chip hardware that allows the flexibility necessary for these alternate configurations. Further modifications of the same hardware allow either horizontal lines or rectangular blocks to be written in one memory cycle [3].

Acknowledgments

The author would like to acknowledge the contributions and encouragement of George Almasi, Stuart Burroughs, Daniel Ling, and Andrew Stankosky in the development of these algorithms and hardware.

References

1. Richard Matick, Daniel T. Ling, Satish Gupta, and Frederick Dill, "All Points Addressable Raster Display Memory," *IBM J. Res. Develop.* **28**, 379–393 (1984, this issue).
2. D. L. Ostapko, "A Mapping and Memory Hardware for Writing Horizontal and Vertical Lines," Invention Disclosure (YO881-0529, serial 509697), August 1981.
3. S. H. Burroughs and D. L. Ostapko, "A Mapping and Memory Hardware for Writing Rectangles and Horizontal Lines," Invention Disclosure (YO883-0085), February 1983.

Received January 26, 1984

Daniel L. Ostapko *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Ostapko joined IBM in 1968 in East Fishkill, New York, where he worked on models for wirability analysis and prediction. In 1971, he transferred to Poughkeepsie, New York, where he worked on reliability, serviceability, availability, test pattern generation, and effective use of hardware redundancy. He was involved in the development of design automation algorithms for PLAs, including minimization, test pattern generation, mapping algorithms for folding PLAs, and a hardware design language. In 1978, he transferred to the Thomas J. Watson Research Center, where he developed algorithms for subdividing PLAs and investigated memory chip hardware for enhancing the performance of an all-points-addressable display. He is currently involved in the definition and programming of a highly parallel processor. Dr. Ostapko received a B.S. and a B.S.E.E. from Trinity College in Hartford, Connecticut, in 1963 and 1964 and an M.S. and a Ph.D. in electrical engineering from Northwestern University in Evanston, Illinois, in 1966 and 1968.