A Rectangular Area Filling Display System Architecture

Daniel S. Whelan

Computer Science Department
California Institute of Technology

4716:TM:82

A Rectangular Area Filling Display System Architecture

Daniel S. Whelan
California Institute of Technology
Pasadena, California 91125

Technical Memo #4716

A Rectangular Area Filling Display System Architecture

Daniel S. Whelan
Computer Science Department
California Institute of Technology

Abstract

A display system architecture which has *rectangular area filling* as its primitive operation is presented. It is shown that lines can be drawn significantly faster while rendition of filled boxes shows an $O(n^2)$ speed improvement. Furthermore filled polygons can be rendered with an $O(n)$ speed improvement.

Implementation of this rectangular area filling architecture is discussed and refined. A custom VLSI integrated circuit is currently being designed to implement this rectangular area filling architecure and at the same time reduce the display memory system video refresh bandwidth requirements.

Introduction

Presently, most computer graphics systems are of two architectures. Calligraphic displays render lines as their primitive operation whereas raster scan displays render pixels as their primitives. This paper discusses a new type of display that renders filled rectangular areas as its primitive operation. This type of display can be considered a sub-class of raster scan displays and as such it retains all of the attributes of such display systems.

This paper examines algorithms for rendering lines, filled boxes and filled polygons and compares time complexities for these algorithms on conventional raster scan displays with those on rectangular area filling displays. This analysis will show that the design and implementation of a rectangular area filling display system is desirable.

Very Large Scale Integration (VLSI) is beginning to show itself in graphics hardware. VLSI has made computer graphics possible by providing large inexpensive random access memories (RAMs) and

several researchers have demonstrated its use for realizing high-performance hardware in a low-cost medium [Clark 1980, Gupta 1981]. Furthermore major manufacturers will be providing such integrated circuits in the near future (NEC and Intel). These uses of VLSI technology will make high performance graphics systems low cost propositions in the near future.

Other researchers have chosen to use the flexibility of VLSI to design display systems that have to be considered radically different architectures [Fuchs 1981, Locanthi 1979]. This paper uses VLSI technology to implement a "processor per pixel" display system. This device is different from previous attempts in that it attempts to ride the coat tails of the RAM manufacturers to provide a large number (4096) of processors per device.

Applications exist for which a rectangular area filling display system would provide substantial system improvements. One such application area is Computer Aided Design (CAD), which may gain the most from such a display since most CAD applications involve the display of non-shaded filled polygons most of which are rectangles. Another application area is real time animation systems since the display system can provide the nice property of robustness. That is, as the system becomes overloaded, its performance degrades smoothly.

Algorithms

This section will analyze algorithms for the rendition of lines, filled boxes and filled polygons on traditional *pixel* oriented displays and on *rectangular area filling* display systems.

The primitive operation that pixel oriented display systems perform is the setting of a 1×1 rectangular area to some intensity i in some constant time τ_p. The primitive operation that a rectangular area filling display system performs is the setting of an axis-aligned $n \times m$ rectangular area to some intensity i in some constant time τ_a. Thus, rectangular area filling display systems have all the generality of pixel oriented display systems since we can set $n = m = 1$. The relative performance of the two systems is determined by τ_p/τ_a. In some cases, τ_a may be smaller than τ_p since some inexpensive pixel based systems require software to select the pixel from within a word of pixels. The following discussions of algorithms will use τ_a and τ_p but the reader should consider $\tau_a \approx \tau_p$.

For the purpose of discussing line drawing algorithms, a line can be described independently of its endpoints by the differences in x and y between the endpoints. These differences will be referred to as Δ_x and Δ_y. Efficient line drawing algorithms for pixel display devices [Bresenham 1965] require $\max(|\Delta_x|, |\Delta_y|)+1$ pixel write operations. Such algorithms can be directly implemented on rectangular area filling display systems, thus the worst case line drawing performance of the two systems is similar. Of course, horizontal and vertical lines are rectangles with a height or width of a single pixel and can be drawn in time τ_a. In fact, on a rectangular area filling display system, any line can be composed of $\min(|\Delta_x|, |\Delta_y|) + 1$ boxes, so it is possible to render a line in time $\tau_a \cdot (\min(|\Delta_x|, |\Delta_y|) + 1)$.

Worst case behavior for this algorithm occurs when $|\Delta_x| = |\Delta_y|$ and the algorithm requires $|\Delta_x|+1$ display write operations which is the same behavior as algorithms on pixel displays. Average behavior of these two line drawing algorithms can be calculated by considering an uniform angular distribution of lines of unit length. If the line length is much larger than one pixel, then the constant term can be dropped and the mean relative performance, $\overline{p}$, of these two algorithms is:

$$\overline{p} = \frac{\int_0^{2\pi} \tau_p \cdot \max(|\Delta_x|, |\Delta_y|)\, d\theta}{\int_0^{2\pi} \tau_a \cdot \min(|\Delta_x|, |\Delta_y|)\, d\theta}$$

Symmetry allows us to only consider the range from 0 to $\frac{\pi}{4}$. In this range, $\max(|\Delta_x|, |\Delta_y|)$ reduces to $\cos\theta$ and $\min(|\Delta_x|, |\Delta_y|)$ reduces to $\sin\theta$ yielding:

$$\overline{p} = \frac{\int_0^{\frac{\pi}{4}} \tau_p \cdot \cos\theta\, d\theta}{\int_0^{\frac{\pi}{4}} \tau_a \cdot \sin\theta\, d\theta} = \frac{\tau_p \cdot \sin\theta|_0^{\frac{\pi}{4}}}{-\tau_a \cdot \cos\theta|_0^{\frac{\pi}{4}}} = \frac{0.707\tau_p}{0.293\tau_a} = 2.413\left(\frac{\tau_p}{\tau_a}\right)$$

If $\tau_a = \tau_p$ average line drawing time is 241% faster using a rectagular are filling display system than with a conventional pixel based system. Furthermore, if we consider that in many applications, the angular distribution of lines is not uniform but tends to emphasize horizontal and vertical lines, the rectangular area filling display performs even better.

Consider filling a $l \times w$ axis-aligned rectangular area such that $lw = n^2$. This requires n^2 write operations on a pixel based display system thus requiring time $\tau_p \cdot n^2$. On a rectangular area filling display system, this operation is a primitive and as such requires one display write operation and requires time τ_a which can be considered as an $O(n^2)$ improvement.

Filling times for convex polygons are more difficult to compare since polygons can differ in both area and shape. In general though, convex polygons can be thought of as having a minimum bounding box of $\Delta_x \times \Delta_y$ pixels. On a pixel based display system the worst case polygon fill will require $\Delta_x \cdot \Delta_y$ write operations whereas on a rectangular area filling display system, the worst case behavior is $\min(\Delta_x, \Delta_y)$ write operations. Such performance can be achieved by using ordinary scan-line algorithms.

Consider a distribution in shape of polygons of area n^2 such that the shorter dimension of the minimum bounding boxes of these polygons is uniformly distributed between 0 and n. Pixel based display system always require n^2 display write operations while a rectangular area filling display system requires at the worst only n display write operations, an order n improvement. The mean relative performance, $\overline{P}$, is:

$$\overline{P} = \frac{\tau_p \cdot n^2}{\tau_a \frac{1}{n} \int_0^n \min(l, n^2/l)\, dl} = \frac{\tau_p \cdot n^2}{\tau_a \frac{1}{n} \int_0^n l\, dl} = \frac{\tau_p \cdot n^2}{\tau_a \frac{1}{n} \frac{l^2}{2}\Big|_0^n} = \frac{\tau_p \cdot n^2}{\tau_a \frac{n}{2}} = 2n\left(\frac{\tau_p}{\tau_a}\right)$$

Non-convex polygons can be decomposed into convex polygons. In practice convex and non-convex polygons are often fractured into axis-aligned trapezoids. The time required to fill polygons that have be fractured into trapezoids still shows an order n performance improvement over pixel based systems since an order n performance improvement is made for each trapezoid.

In summary, algorithms can be found that make rectangular area filling display systems perform better than pixel based display systems at line drawing, box filling and general polygon filling operations.

Display Memory System Architecture

Pixel based raster scan display systems are common place because of the availability of high density random access memories. Unfortunately, RAM manufacturers are striving to build larger and larger $n \times 1$ RAMs without considering that other RAM organizations might yield viable products. It is because of this $n \times 1$ memory organization that we have pixel based display systems. In this paper, another RAM organization is presented that enables rectangular area filling display systems to be

built. Furthermore since this RAM design is very similar to current RAMs being produced today, it should be able to be produced as a high density part at low cost.

Conventional pixel based display systems are built up of $n \times 1$ memory circuits that are organized into a $m \times z$ linear array where m is usually some integral multiple of n and z is the the number of bits per pixel. Figure 1 illustrates how such a system is built up from $n \times 1$ RAMs. A typical display system might have $m = 1,000,000$ and $z = 24$. The important thing to realize is that this organization imposes a restriction that only one pixel may be modified in any memory cycle. That is, out of $24,000,000$ bits of RAM only 24 may be altered at any one time. This is the bottleneck that conventional RAM devices impose upon the graphics system designer.

There are ways to make this bottleneck appear less severe but usually this means that the designer has built a system that allows more than one pixel to be accessed at a time. For example, Sutherland [Gupta 1981] has built a display system that allows 64 neighboring pixels to be modified at once and although it is true that the system performance may be 64 times better, such a system is only altering $24 \cdot 64$ bits out of $24,000,000$ which is still a small fraction.

Figure 2 illustrates why the conventional RAM organization causes this bottleneck. A static RAM contains a rectangular array of memory cells each similar to the cell shown in Figure 3. A single cell is selected by first selecting a row of the memory array and then selecting a bit within that row. Both selections are made by the binary decoders labeled row select and column select. The RAM cell works by writing its contents on the bit lines when selected. During a read cycle, sense amplifiers detect the state of the RAM cell by detecting voltage changes on the bit lines. During a write cycle, one of the bit lines is pulled to ground and the RAM cell is set to a 1 or a 0.

Since $n \times 1$ RAMs use binary address decoders and thus hide all but one memory element from the outside world, the only practical way to make a large improvement in display performance is to design a RAM chip expressly for use as a graphics display memory. The previous section illustrated that rectangular area filling is a desirable primitive operation. Because we want to deal with rectangles, it seems that the display memory system ought to be able to address rectangles in a natural way. Another desirable feature would be if the memory organization could decrease the video refresh overhead that is usually associated with high resolution displays. It is often the case that the computer or graphics

processor can only get every other memory cycle or may not even be able to access the video memory during active scan periods. This means that video refreshing often requires between fifty and eighty percent of the display bandwidth.

Addressing rectangles within a rectangular memory array can be easily accomplished by providing the x and y coordinates of the lower left corner and upper right corner points. The problem is to design a decoding scheme that is cascadable, that is, one that will allow an arbitrary size rectangular array to be built. As was pointed out earlier, both conventional RAMs and display memory systems use binary address decoders and the address decoding process can be though of as a hierarchy of binary decoders. Such a simple cascading scheme is desirable and for this reason, the decoders within our rectangular area filling RAM ought to be the same type of decoder that is used to but together an array of these RAM chips.

In order to provide two dimensional addressability, the memory array within the RAM will have to be modified slightly. First, notice that the static RAM of Figure 2 has only row select signal lines running across the array. The rectangular area filling RAM will need column select signal lines running across the memory array also. Of course, the basic RAM cell of Figure 3 will have to be modified slightly to make use of these column select lines. This involves the addition of two pass transistors as is illustrated in Figure 4. The operation of an array of such memory cells is as follows. A memory cell is selected if its column and row select lines are both 1. When selected, the memory cell behaves as it did before, forcing its state onto the bit lines. Notice however, that if a band of column select lines and a band of row select lines are set to 1, a rectangle of memory cells are simultaneously selected. These cells may be written in the same way that one memory cell is written in a conventional RAM. Reading poses a problem in that cells in different selected rows may have had different values prior to being selected and for this reason, this architecture only allows single rows to be read. This memory array architecture provides us with a way of implementing a rectangular area filling RAM only requiring us to design a *banded* address decoder. In addition, the extra wiring channels and pass transistors have not resulted in an extremely larger memory cell, enabling the implementation of large memory arrays on a single die.

The design of a *banded* address decoder is straightforward. A *banded* decoder requires two address, a lower address and a upper address, and selects the band of outputs between and inclusive of the two

addresses. Assume that one can build a n bit decoder that takes an address i where $i < 2^n$ and selects all outputs in the range $[i \ldots 2^n]$. Call that the lower address decoder. Similarly construct another decoder that selects all outputs in the range $[0 \ldots i]$ and call that the upper address decoder. The logical AND of these two decoders selects the band $[la \ldots ua]$ where la is the lower address and ua is the upper address. Either type of decoder can be constructed by building a binary decoder and adding a propagation chain to it as is illustrated in Figure 5. A propagation tree can be used to reduce the propagation time from $O(n)$ to $O(\log n)$. A complete *banded* decoder is illustrated in Figure 6. Notice that it provides select and carry inputs that are used in the cascading of the decoders.

Decreasing video refresh bandwidth requirements is also rather straightforward. If we construct a display memory out of these RAM chips such that rows in the memory array map onto scan lines on the display then a whole scan line of video refresh data can be read in one memory cycle. Since we don't have the pins to bring all of the bits out of the chip in parallel, read data is shifted out serially, as is appropriate for video refreshing and only requires the addition of a shift register to each RAM chip. Adding a latch between the RAM array and the shift register makes the device easier to design into a system since the one video refresh cycle per scan-line time doesn't have to come at any time tightly synchronized with the video refresh operation but has to occur in a window of one scan-line time.

Figure 7 illustrates the complete architecure of a rectangular area filling RAM. The memory array is surrounded on two sides by the row and column *banded* decoders. The *decoder strobe* input is used to gate the outputs of the decoders. The bottom of the memory array is connected to sense amplifiers which detect the small voltage variations on the bit lines during read operations and convert these signals to a logic level representing the value of the stored bit. The outputs of the sense amplifiers are inputs to a 64-bit wide latch. When data is read, it is loaded into the latch and, sometime later the outputs of the latch are loaded into a 64-bit shift register and shifted off the chip to provide video data. The final component is labeled *write drive circuitry* and consists of the logic necessary to pull the proper bit line to ground during write cycles.

Figure 8 illustrates how these RAM chips may be composed to form a larger display memory system. This figure illustrates a single plane and thus for a full display system, the RAM chips must be arranged in a three-dimensional array instead of a two-dimensional array as in the case of ordinary

systems. Decoding for the array, is done with *banded* decoders. A decoder output is used for the select input while neighboring decoder outputs are used for the two carry inputs. A multiplexor is needed to select the proper serial video data stream.

Implementation Refinements

A 4096 device rectangular area filling RAM is currently being implemented in n-MOS. Several refinements have been made to the architecture to reduce the circuitry, power dissipation and the package pin count.

One of the most notable changes is that the RAM is being implemented as a pseudo-static RAM instead of as a static RAM. This change changes the memory cell from an eight transistor circuit to the six transistor circuit illustrated in Figure 9. Because of the reduction in the number of transistors and the elimination of a power line, the pseudo-static RAM cell is much smaller than the static cell. Furthermore since ordinary static RAMs use six transistor cells, comparable size devices should be achievable. Currently 16K static RAMs are available and many 64K devices have been reported in the literature [Ohzone 1980, Wada 1981]. Pseudo-static implies that the device is static in one mode of operation and dynamic in another mode. This means that the device must be periodically refreshed but in a display memory, refreshing occurs naturally through the video refresh accesses and thus imposes no additional overhead. Pseudo-static devices also draw less current than static devices so the device power dissipation is expected to be much less.

In an effort to reduce the number of pins required by the RAM, 12 address pins are used and are time multiplexed to read in all 24 bits. This time multiplexing has the side affect of allowing one of the two *banded* address decoders in Figure 7 to be discarded since a single *banded* decoder can be time multiplexed and the outputs latched at the appropriate times by the select driver circuitry. These changes are illustrated in Figure 9.

Conclusions

The performance improvements that rectangular area filling display systems can provide justifies their implementation. Architectures such as this one, which can be described as "processor per pixel"

architectures, necessitate implementations in silicon and since these architectures can provide notable improvements in system performance, this is an area where manufacturers should concentrate their efforts.

The system described here does have some drawbacks. While it can fill polygons fairly fast, it is not significantly faster at filling shaded polygons since shading usually implies that neighboring pixels have slightly different values. For the same reason, the architecture does not aid in the rendition of anti-aliased lines. However, both shading and anti-aliasing can be performed by this system since the rectangle size can be set to 1×1 causing the system to behave like an ordinary pixel system.

Further work might concentrate on a scan-line filling RAM. The previous analysis indicates that scan-line filling can offer significant performance improvements but will not be as good as rectangular area filling since scan-lines are always horizontal. Shading algorithms might be easyily implemented on a scan-line filling RAM. Both scan-line and rectangular area filling RAMs allow scan-lines to be filled and therefore can be used in implementing real-time visible surface systems using a Watkins [Watkins 1970] algorithm.

Acknowledgments

The author would like thank Jim Kajiya and Carver Mead for their many helpful discussions. In addition, he would also like to acknowledge conversations with Henry Fuchs. This research was sponsored by the Defense Advanced Research Project Agency, ARPA Order number 3771, and monitored by the Office of Naval Research under contract number N00014-79-C-0597.

References

[Bresenham 1965] J. E. Bresenham
"Algorithm for computer control of a digital plotter"
IBM Systems Journal, Vol. 4, No. 1, 1965, pp. 25-30.

[Clark 1980] James H. Clark
"Structuring a VLSI System Architecture"
LAMBDA, Vol. 1, No. 2, 1980, pp. 25-30.

[Fuchs 1981] Henry Fuchs and John Poulton
"PIXEL-PLANES: A VLSI-Oriented Design for a Raster Graphics Engine"
VLSI Design, Vol. 2, No. 3, 1981, pp. 20-28.

[Gupta 1981] Satish Gupta, Robert F. Sproull and Ivan E. Sutherland
"A VLSI Architecture for Updating Raster-Scan Displays"
Computer Graphics, Vol. 15, No. 3, 1981, pp. 71-78.

[Locanthi 1979] Bart Locanthi
"Object Oriented Raster Displays"
Proceedings of Caltech Conference on Very Large Scale Integration,
1979, pp. 215-225.

[Ohzone 1980] Takashi Ohzone, Juro Yasui, Takeshi Ishihara and Shiro Horiuchi
"An 8K×8 Bit Static MOS RAM Fabricated by n-MOS/n-Well CMOS Technology"
IEEE Journal of Solid-State Circuits,
Vol. SC-15, No. 5, October 1980, pp. 854-861.

[Wada 1981] Toshio Wada, Hiroshi Yamanaka, Mitsuru Sakamoto, Hirohiko Yamamoto and Shigeki Matsue
"A 16 DIP, 64 kbit, Static MOS-RAM"
IEEE Journal of Solid-State Circuits,
Vol. SC-16, No. 5, October 1981, pp. 488-491.

[Watkins 1970] Gary S. Watkins
"A real-time visible surface algorithm"
University of Utah Computer Science Department
UTECH-CSc-70-101, June 1970.

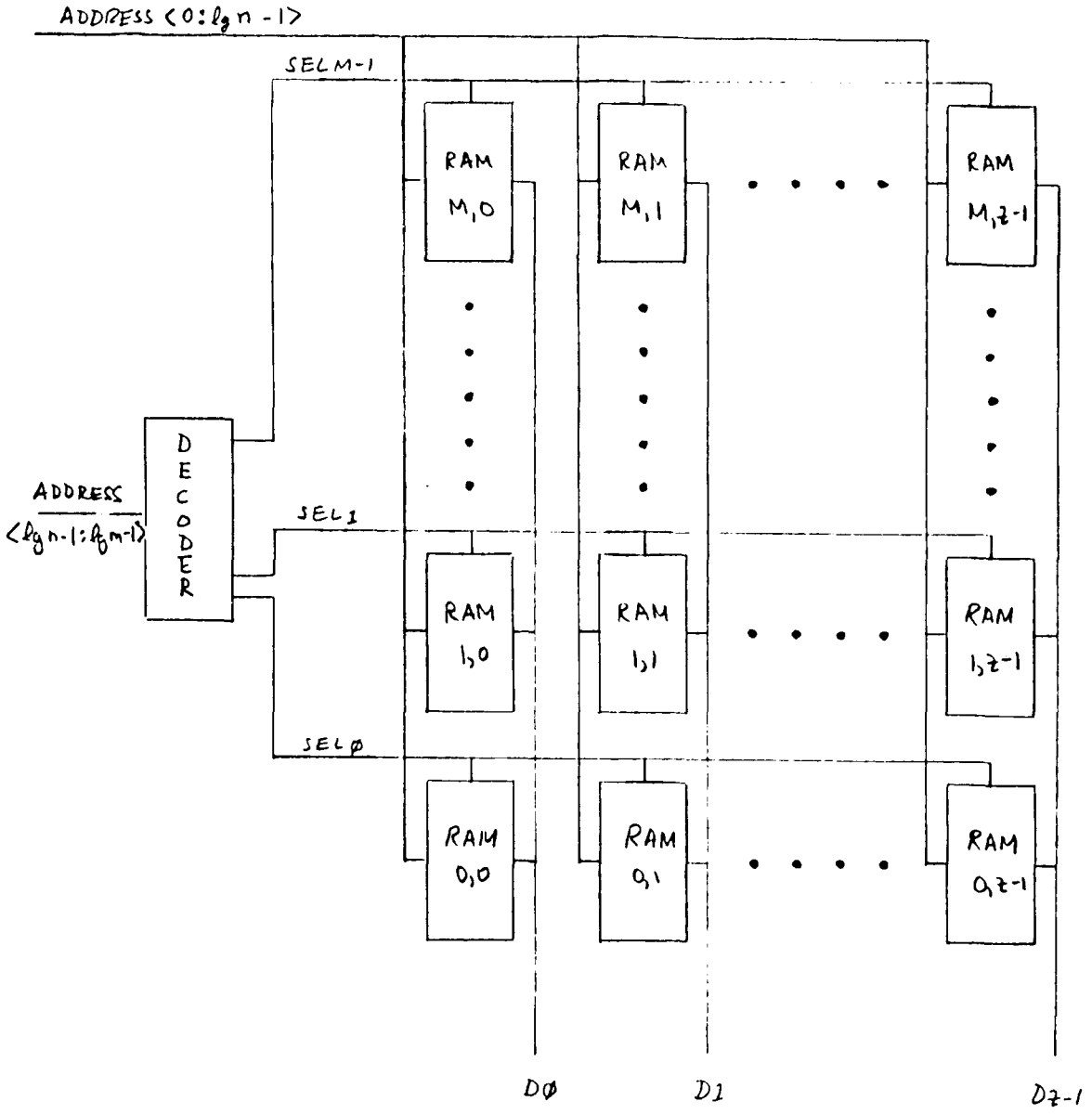

Figure 1: Composition of an m by z memory from n by 1 memory chips.

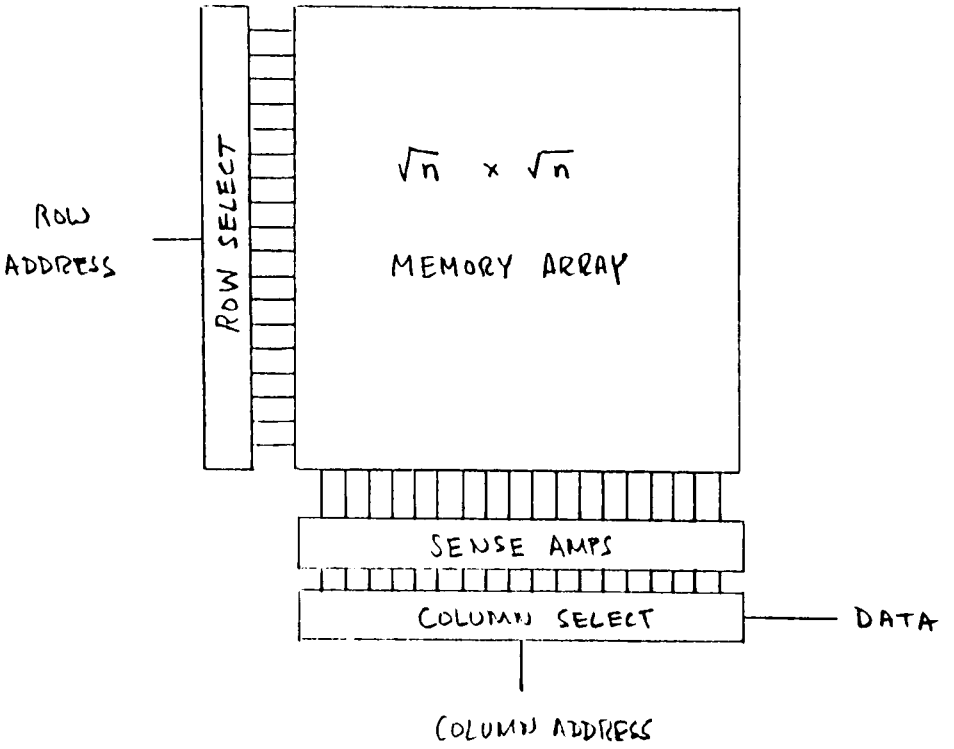

Figure 2: Floorplan of a typical n by 1 static RAM.

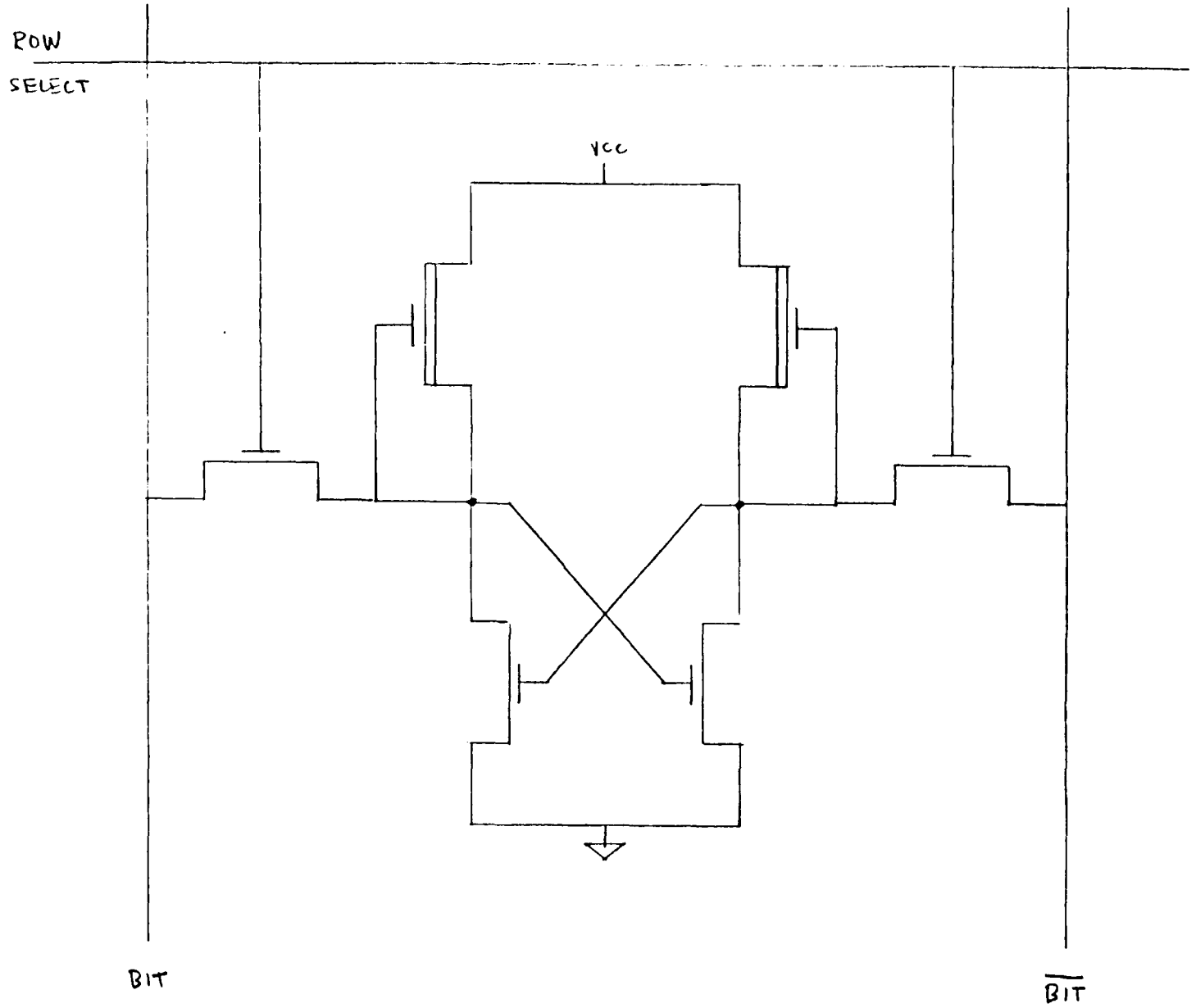

Figure 3: A six transistor static memory cell.

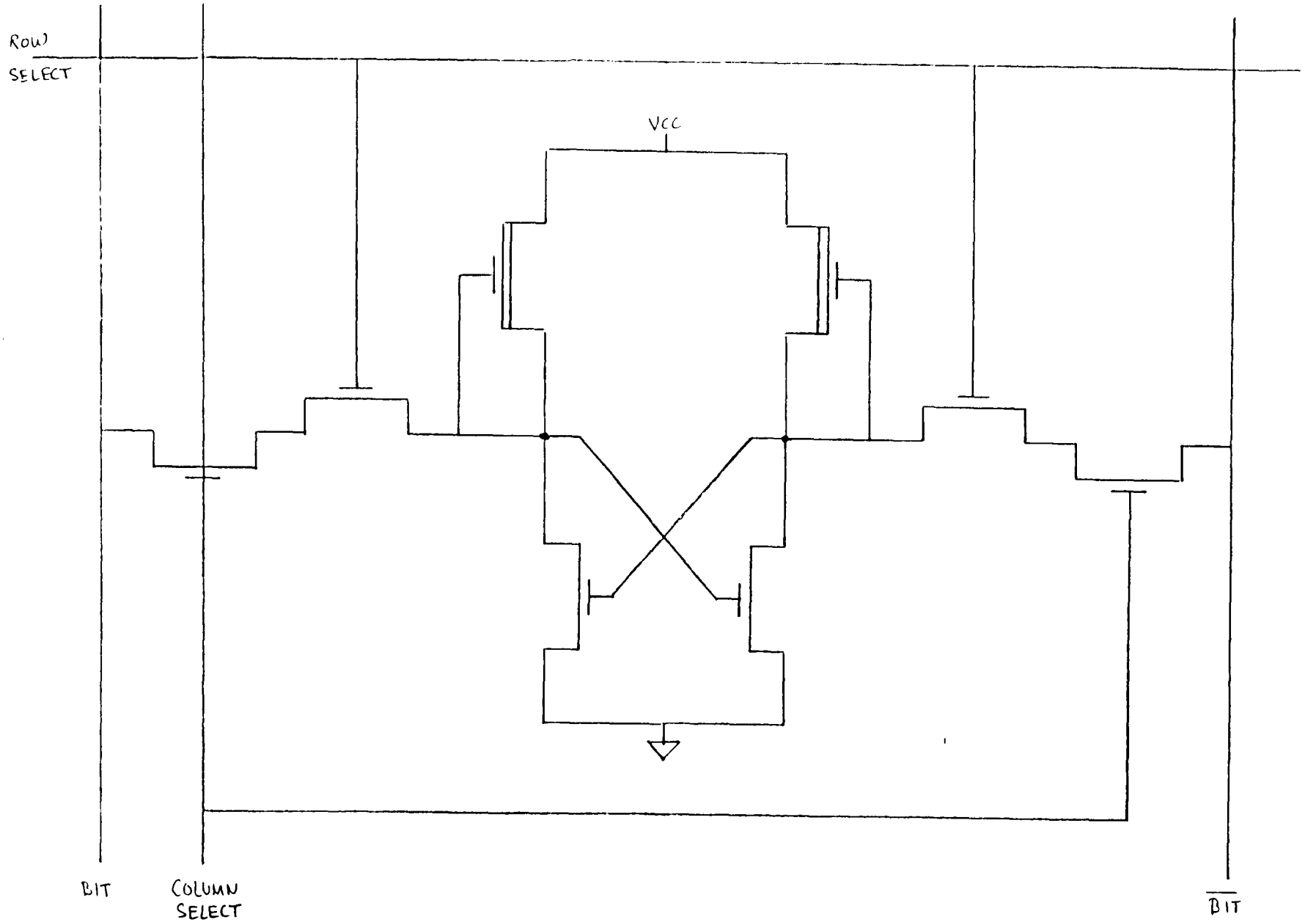

Figure 4: Static memory cell with row and column select capability.

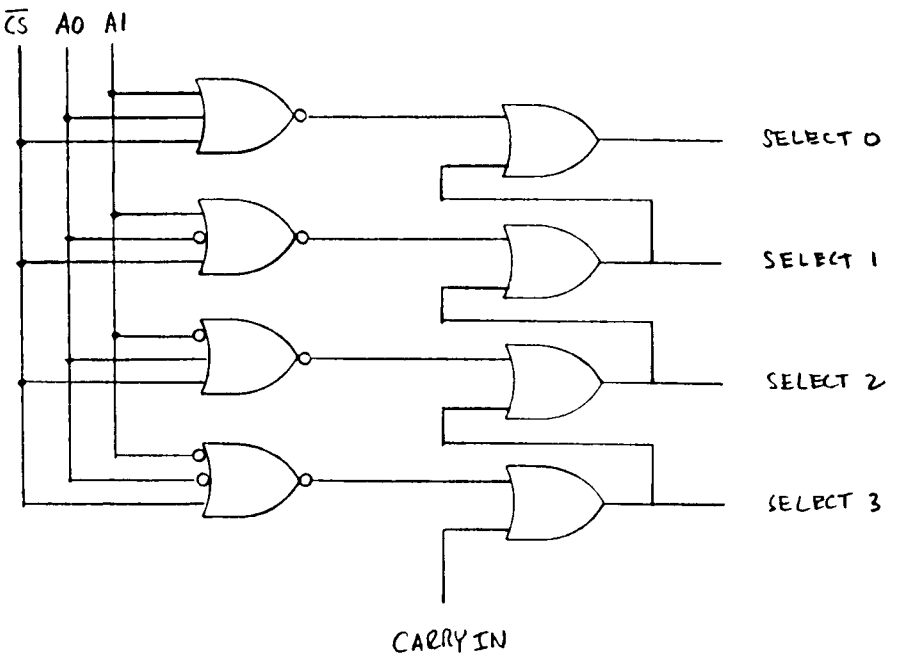

Figure 5: Binary decoder with a propagation chain.

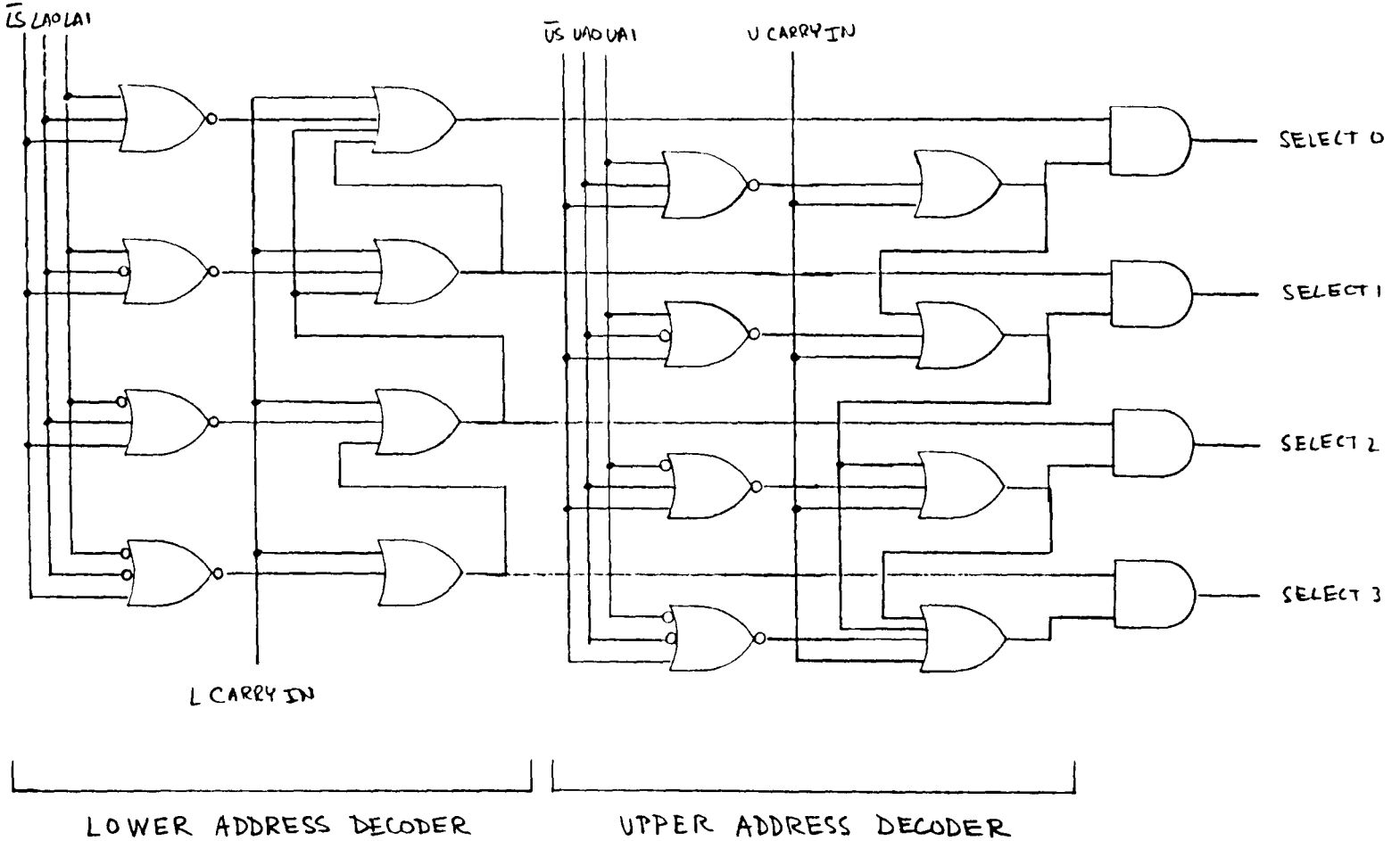

Figure 6: A complete 2 bit banded decoder implemented with a propagation tree.

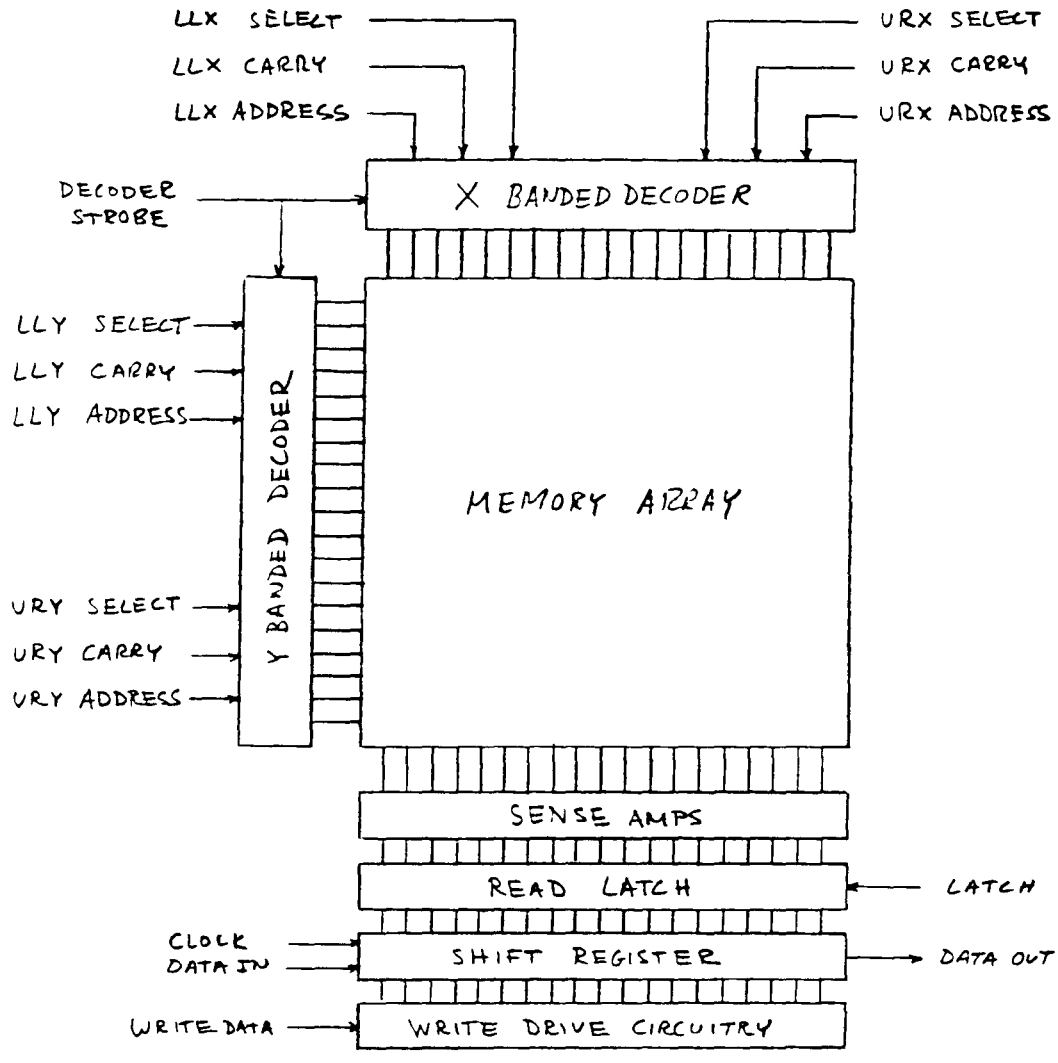

Figure 7: Rectangular Area Filling RAM architecture.

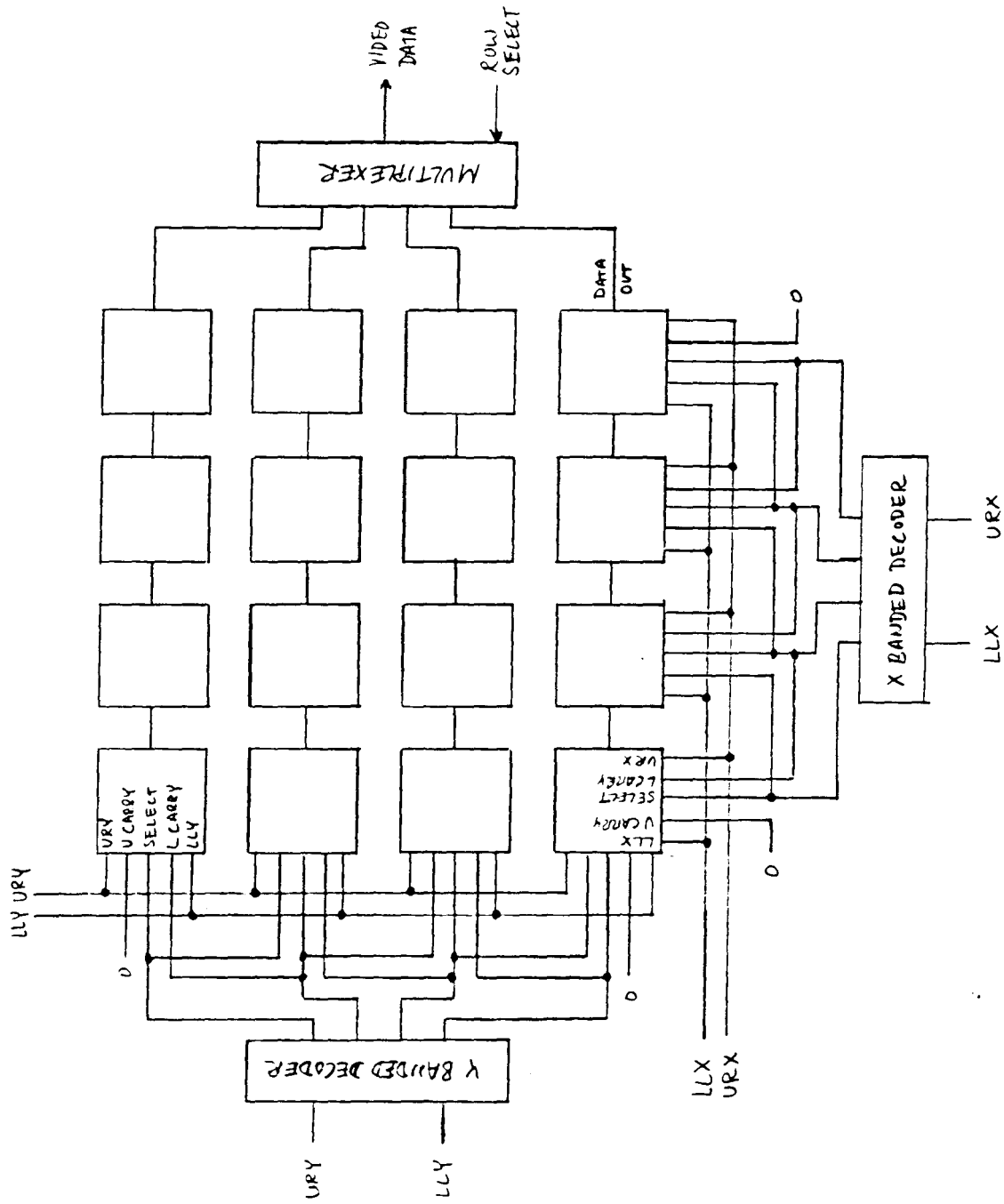

Figure 8: Composition of a display memory system out of Rectangular Area Filling RAMs.

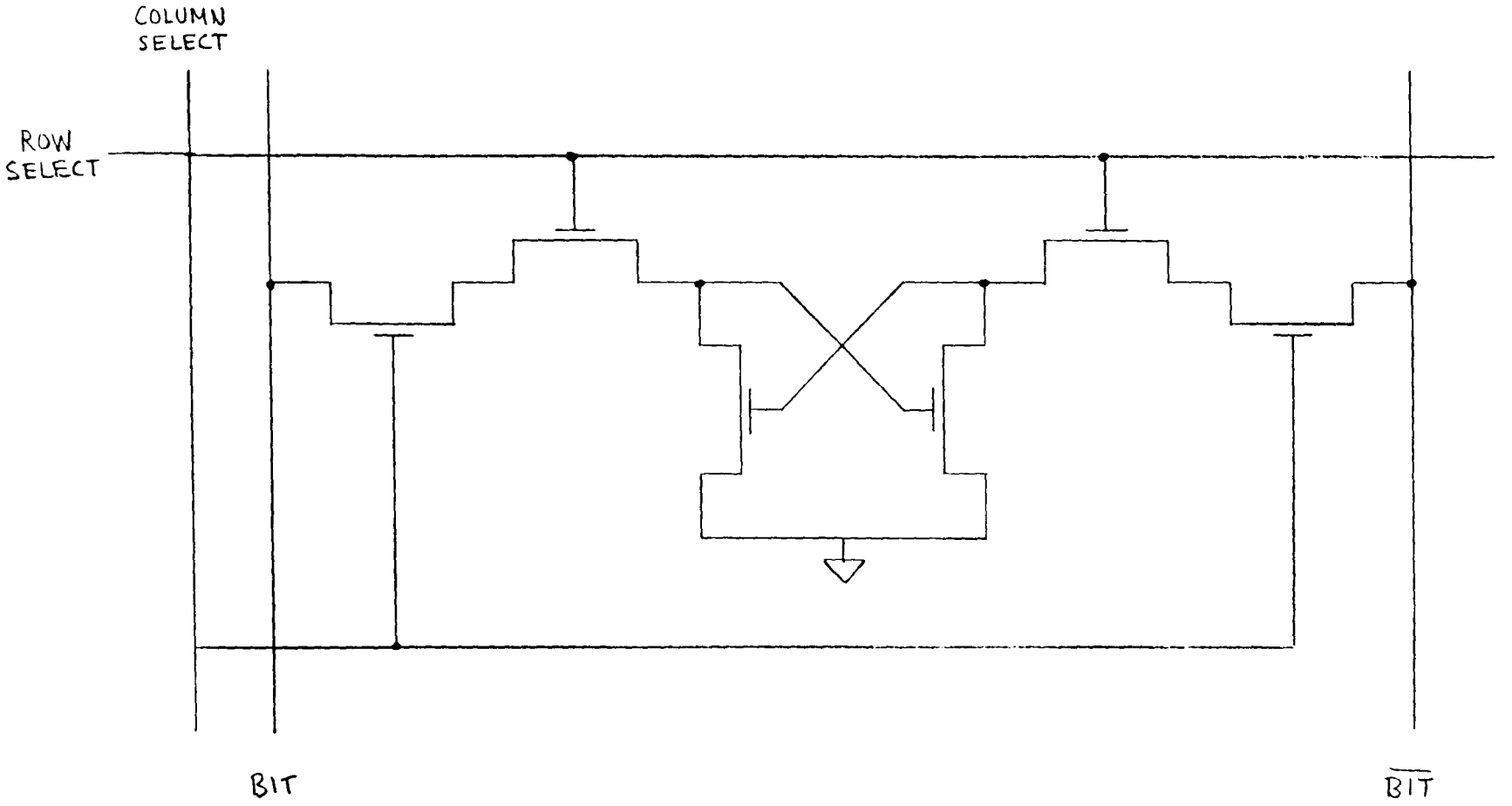

Figure 9: A psuedo-static RAM cell with row and column selects.

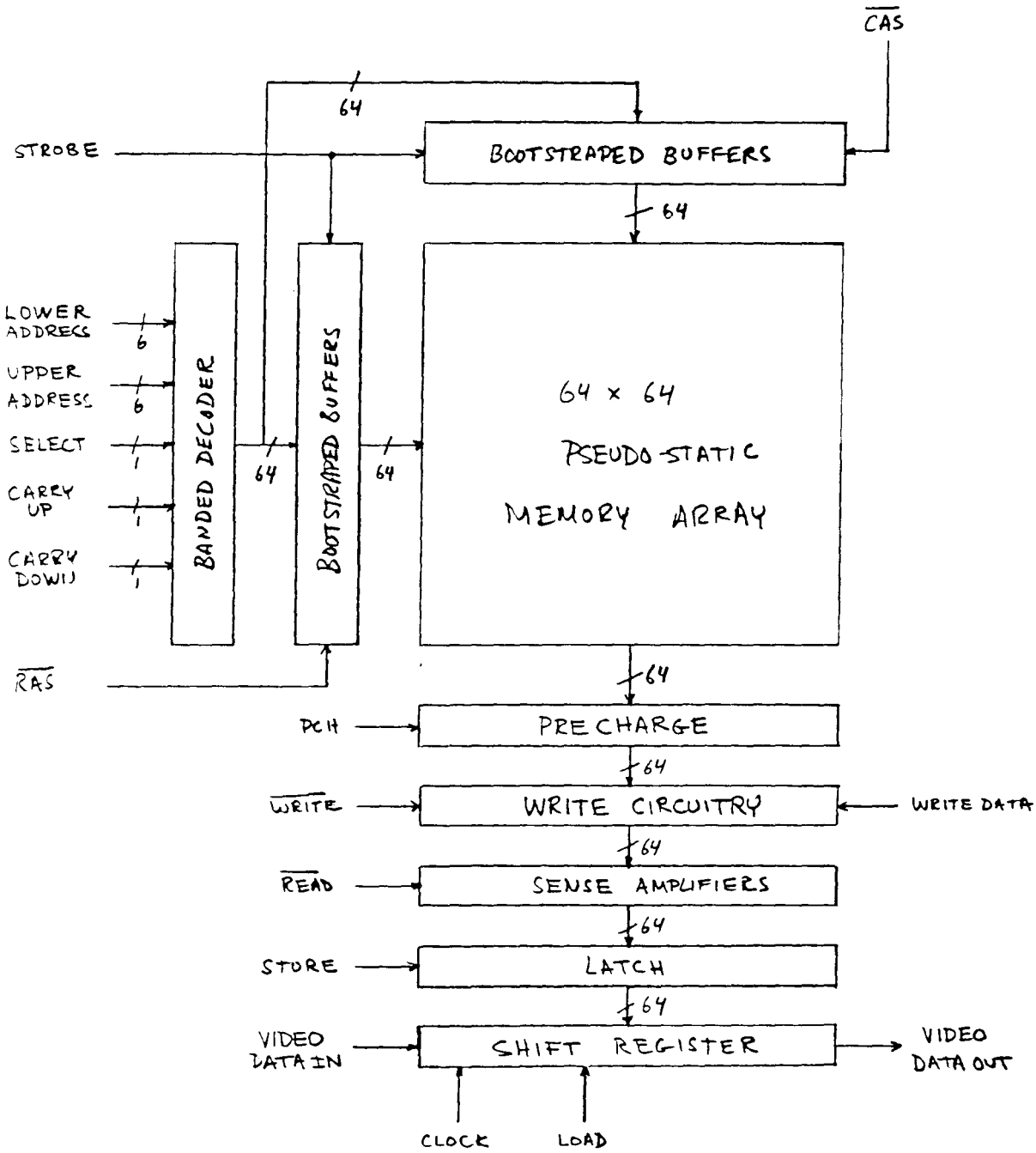

Figure 10: Floorplan of the 4096 bit Rectangular Area Filling RAM currently being implemented.