Object Oriented Raster Displays

Bart Locanthi
California Institute of Technology
Pasadena, California 91125

This paper describes a special-purpose MOS/LSI memory device and its design. This device was designed as an attempt to speed up the display and animation of multicolor raster display images, and is highly specialized to that task. Generalizations of the approach taken would be suitable, and possibly economical for geometrical computations encountered in the course of manipulating integrated circuit layouts, such as design rule checking. These generalizations will be discussed, as will the design and implementation of this particular chip.

The principal motivation behind this design is the observation that the horizons of speed simply have not kept pace with the density improvements we have seen in semiconductor devices. These density improvements have brought about the personal computer, and along with it the re-emergence of interactive computer graphics. Indeed, the resolution and color range of bitmap displays is still increasing rapidly. However, the raw speed to manipulate these high resolution displays is not easy to come by. Witness the increasing development of 32 and 64 bit machines which have no reason to be that wide except to be able to blast display bits around faster. The objective of this design is to reduce the computer-to-display bandwidth required to make display changes at a given rate.

The research described in this paper was sponsored by the Defense Advanced Research Projects Agency under contract number N00123-78-C-0806.

Figure 1 shows a typical bitmap display hierarchy with user interactions at the top and display interactions at the bottom. The user works at a high level of abstraction in a language (possibly graphical) tailored to his application. Users are typically low bandwidth devices. The computer interacts with the display in terms of bits in its memory, a very general form of interaction. The bandwidth available to change bits in memory is very high, although generally much less than is desired in terms of display speed.

Object Oriented Raster Displays

Although the computer eventually has to deal with bits, the data structure from which the bits must be generated is abstracted in varying degrees of specialization. With this specialization comes a more limited form of description and therefore more conciseness. Applications such as IC design often do not need the full generality of bitmap displays and can get by with this more limited form of description. With specialized LSI devices we can raise the primitive level of displayable objects and thus allow the computer to interact with the display in more abstract terms. The price is clear: generality. The payoff is conciseness and an effective increase in display bandwidth available.

The level of abstraction I have chosen to implement is that of rectangles. In a typical IC layout, the vast majority of displayed objects are rectangles or collections of rectangles. This is becoming increasingly true as DA systems attempt to do more and more for the user. Polygons and circular arcs are certainly more general, but the relative verbosity of expression weighed against the frequency of their use (not to mention complexity of implementation) favors the more simple representation of rectangles.

The processing for a bitmap display is amortized over a centralized display memory. Raster conversion is handled by the main processor, and display bits are read out either by hardware or special

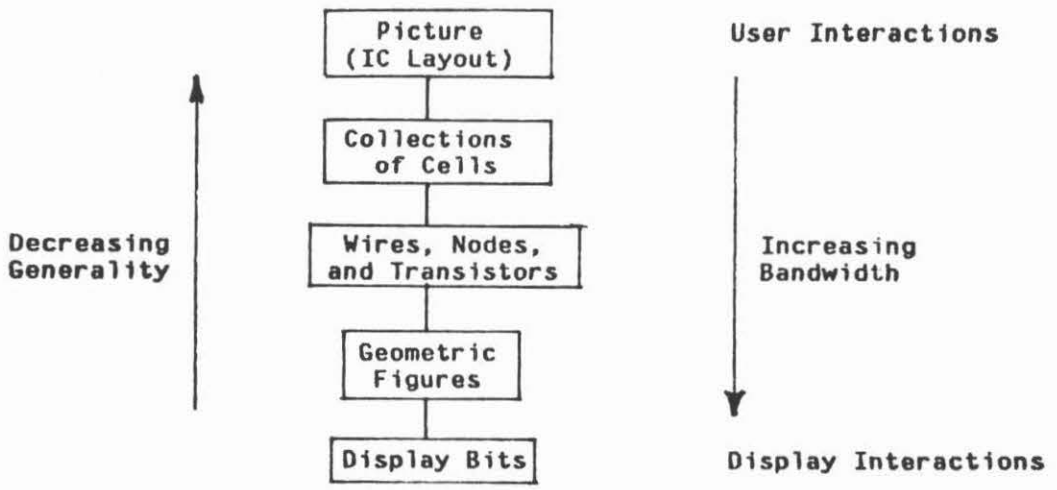

Figure 1. Typical Bitmap Display Hierarchy

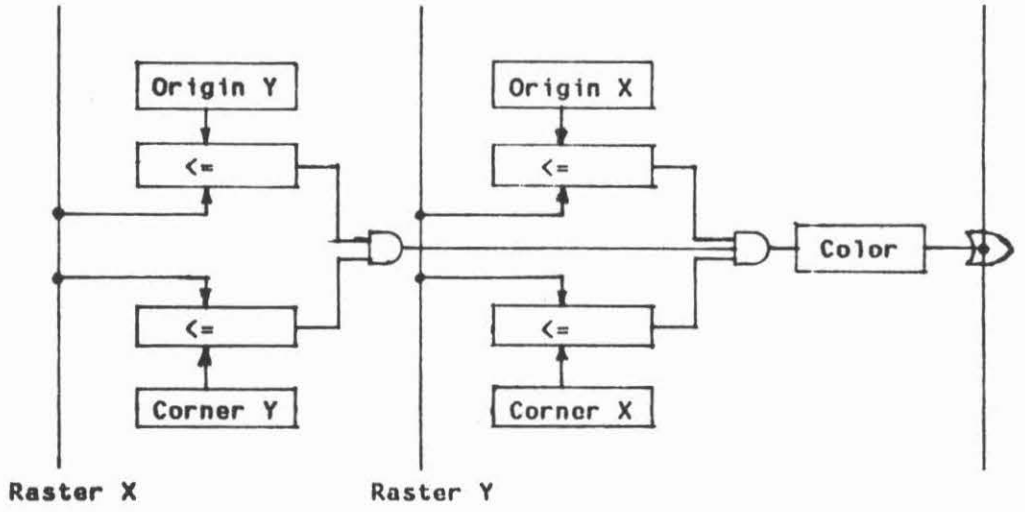

Figure 2. Rectangle Cell

microcode. No on-the-fly processing is performed. It is of course possible to speed up the raster conversion process while retaining the bitmap, resulting in an object oriented display of sorts. This approach was taken in the Xerox Alto, which has special microcode to manipulate rectangular areas in the bitmap. However, once the decision is made to store the display image in bitmap form, animation of the display becomes very tricky, since the images making up the display are ORed together. I think an adequate solution to this requires the display image to be regenerated for every frame. While it may be possible to contrive a combination of hardware and microcode to do this, I question how well this approach would scale for more complicated pictures or higher resolution displays.

A Rectangle Display Chip

A simpler solution is to distribute raster conversion hardware throughout the display memory. In addition to a high level description of its image, each object also knows how to convert itself to a raster image. For rectangles this raster conversion hardware is very simple indeed. Each rectangle only needs to be able to tell if it contains the raster point, and give its contribution to the color if this is so. The raster scans through the entire screen and the color at each point is determined by combining the contributions from each rectangle.

Figure 2 shows the organization of a rectangle cell. For convenience, the boundaries of the rectangle are given in absolute coordinates and are determined by two diagonally opposing vertices, call them the origin and corner. Each rectangle has a color, which is conveniently represented as an address in a color map. Colors are combined by ORing color map addresses. The test for containment is simple (i.e. contains(p) = origin <= p AND corner >= p), and minimally obtained by ANDing the carries out of four subtractions. The process of raster conversion is then broadcasting the raster coordinates to all rectangle objects and ORing the color map address to produce the appropriate color for each point.

Figure 3 is a schematic for one bit of a register/ comparator pair. The storage element is a standard 6 transistor static RAM cell. The comparator is a simple alternating polarity ripple through design using 7 transistors. As figure 4 demonstrates, the space occupied by logic almost exactly equals the space occupied by storage. This is a good balance of memory and logic for a static organization. However, the use of dynamic storage devices would upset this balance considerably, since the area required for logic would not decrease appreciably. Even with extreme specialization and application of cleverness, the fact remains that logic is expensive. (Similar arguments make the construction of associative memories hard to justify.) Imagine if you will the gross imbalance that would result from attempting a more general primitive element than rectangles.

Cavalier application of conservative design rules (6 micron single level polysilicon, no buried contacts) resulted in 8 rectangles on a chip which measures about 5000 microns square. Using 8 bit precision in x and y coordinates and color, this amounts to 320 bits of storage. In a chip where half of the interior area is occupied by logic, this gives some indication of the state of integrated circuit technology available to a university. A technology capable of producing a 16K static RAM could produce an 8K bit rectangle chip. This translates to 200 rectangles of 8 bit precision, or 125 rectangles of 12 bit precision.

Scaling

Object oriented displays scale in a particularly simple way. The amount of storage required for application A is proportional to the number of objects displayed in application A. Capability can be added simply by adding more storage. Higher precision displays are generated by using higher resolution parts. Bitmap displays, however, require memory capacity AND bandwidth proportional to the square of the precision desired.

No claims are made for the coding density of object oriented displays. Naturally, simple pictures require very few parts.

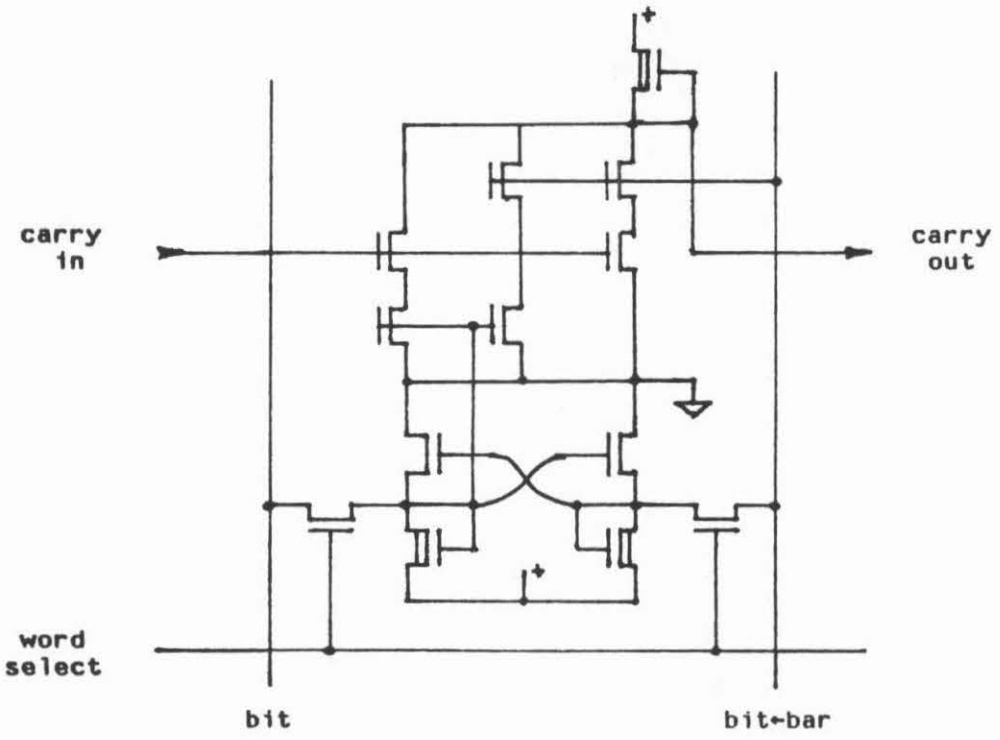

Figure 3. Storage/Comparator Schematic

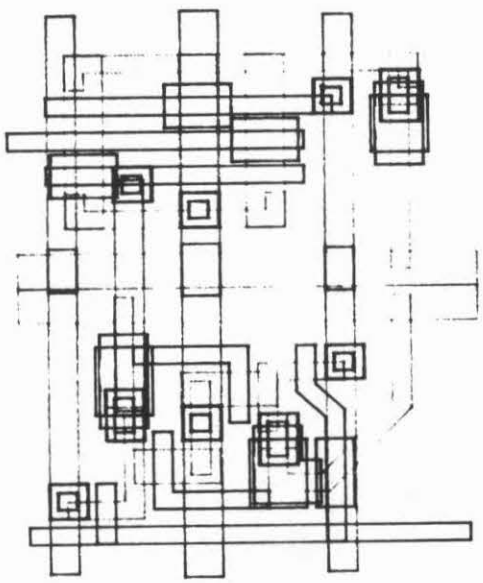

Figure 4. Storage/Comparator Layout

Conversely, complicated pictures may require more bits of object oriented storage than bitmap storage. The object oriented display is a run length, or derivative encoding of a picture. No particular relation exists for the amount of storage needed to express the spatial derivative of a picture as compared to that needed for the picture itself, although previous work with run length encoding of pictures indicates that these amounts are usually about the same.

Certain phases of integrated circuit design involve the display of fairly simple pictures with which a high degree of interaction is desired. During other phases it is desirable to see large fractions of the circuit at the same time. This can mean the display of literally hundreds of thousands of rectangles for a typical VLSI chip, each rectangle only occupying a few pixels. For these cases it would be desirable to have a bitmap storage backup for an object oriented display, using the object storage as a fast raster conversion hardware front end.

Speed

Each element of an object oriented display must perform a computation for every pixel in every frame. Some high resolution non-interlaced displays have bit times on the order of 15 nanoseconds. Even taking advantage of the simplicity of rectangles, there is still some question as to whether the processors will be able to keep up with the raster or not. Let me amend that. For a 15 nanosecond per pixel display, there is no question at all.

The ripple carry subtraction circuit used in the rectangle display chip simulated to about 10 nanoseconds per bit. An 8 bit compare would thus require 80 nanoseconds in the worst case, the worst case being at the boundary of a rectangle. This is certainly sufficient for the 256 by 256 resolution possible with an 8 bit chip. For higher resolutions, the edge detection decisions will always arrive late, and accuracy will suffer from the variation in speed of the rectangle processors.

This variation can be equalized by clocking the individual carries

and skewing the bits of the raster coordinates as they are presented to the rectangle processors. Skewing and pipelining carries is also a convenient way of building high speed counters, which are needed to calculate the raster position and should be placed on-chip anyway. It seems therefore that a very high speed object oriented storage chip could be constructed without much difficulty.

Another approach to the speed problem is to keep a completely static design and let the logic be as fast as it turns out to be. Regardless of how slow the comparison logic turns out to be, the worst that can happen is that it can't keep up with changes in the low order raster bits. We are guaranteed monotonicity, only linearity suffers.

Regardless of whether we believe improvements in IC processes will eventually be able to cope with the speed or not, it makes sense to adopt the simplest possible design. A simple design will be compact and will thus enjoy some advantage in speed over a more complicated part. It may also work, an important consideration in a world where turnaround is measured in months.

System Integration

Where does an object oriented storage device fit into a system? Figure 5 illustrates a suitable embellishment, Class Rectangle, transcribed into Simula from a successor object oriented language, Smalltalk. By the use of this mechanism, the user deals with the protocol of the general rectangle object. The fact that certain operations such as display and movement are implemented in hardware is invisible. All the user cares is that the rectangles he creates do his bidding; Class Rectangle handles the details of interacting with the display memory.

Certain operations of Class Rectangle are not handled directly by the rectangle chip, but are maddeningly close to the basic function. The test for containment of a point is a basic primitive of each rectangle processor, yet it is not conveniently available

```
CLASS rectangle(origin,corner,color);
REF(point) origin,corner; INTEGER color;
! Excerpts from SSP's Class rectangle;
BEGIN

    REF(point) PROCEDURE center;
    center :- origin.plus(corner).times(0.5);

    PROCEDURE show;
    ! This is what it would be for a bitmap display;
    BEGIN
        INTEGER i,j;
        FOR i := origin.x TO corner.x DO
        FOR j := origin.y TO corner.y DO
        displaycolor(i,j) := displaycolor(i,j) OR color;
    END;

    PROCEDURE moveby(p); REF(point) p;
    ! includes an implicit show;
    BEGIN
        origin :- origin.plus(p);
        corner :- corner.plus(p);
    END;

    BOOLEAN PROCEDURE contains(p); REF(point) p;
    contains := origin <= p AND corner >= p;

    REF(rectangle) PROCEDURE clip(r); REF(rectangle) r;
    clip :- IF origin > r.corner OR corner < r.origin THEN NONE
    ELSE NEW rectangle(origin.max(r.origin),corner.min(r.corner));

    REF(rectangle) PROCEDURE including(p); REF(point)p;
    ! returns the rectangle including the point p;
    including:-NEW rectangle(origin.min(p),corner.max(p));

END;
```

Figure 5. Class Rectangle

from outside, except by using the color field of each rectangle as an address and priority encoding the colors out from the assorted chips. Containment when applied to a set of rectangles gives solves the pointing problem in constant time.

Even more useful would be the test for overlap applied to a set of rectangles. The basic test for overlap involves two point comparisons, each of which is similar to the comparisons performed by the rectangle chip. If this operation were available from the outside, the mutual overlap of a set of rectangles could be found in linear time. Whether or not this is useful to you depends on how much automatic design rule checking you like to do. Anticipating the return of some rectangle display chips for testing, I may soon find out just how much automatic design rule checking I should have done.

Summary

A highly specialized MOS/LSI integrated circuit has been described. The sole reason for the existence of the circuit is to speed the display and animation of multicolor raster display images composed of rectangles. By such specialization it is hoped that the circuit will perform it's intended task faster and more economically than bitmap displays.

Using standard silicon gate technology and relaxed design rules, the circuit provides storage and raster conversion hardware for 8 rectangles, using 8 bit precision, on a chip measuring about 5000 microns square. While generalizations are possible and in some cases useful, I felt the added cost in design time and circuit complexity could not be justified for the first implementation.

There are of course abstract pleasures from designing systems cleanly in an object oriented way. However, the ultimate justification for me will come when I can drag parts of an IC layout across a display that comes alive with the animation made possible by this kind of "smart" memory device.

References

1. **Birtwistle, G., Dahl, O. J. Dahl et al, Simula Begin, Petrocelli/ Charter 1973.**

2. **Ingalls, D., The Smalltalk-76 Programming System: Design and Implementation, Proceedings of the ACM Conference on Principles of Programming Languages, February 1978.**