Palo Alto Research Center

The Optical Mouse,

and an Architectural Methodology for Smart Digital Sensors

by Richard F. Lyon

XEROX

The Optical Mouse,

and an Architectural Methodology for Smart Digital Sensors

Richard F. Lyon

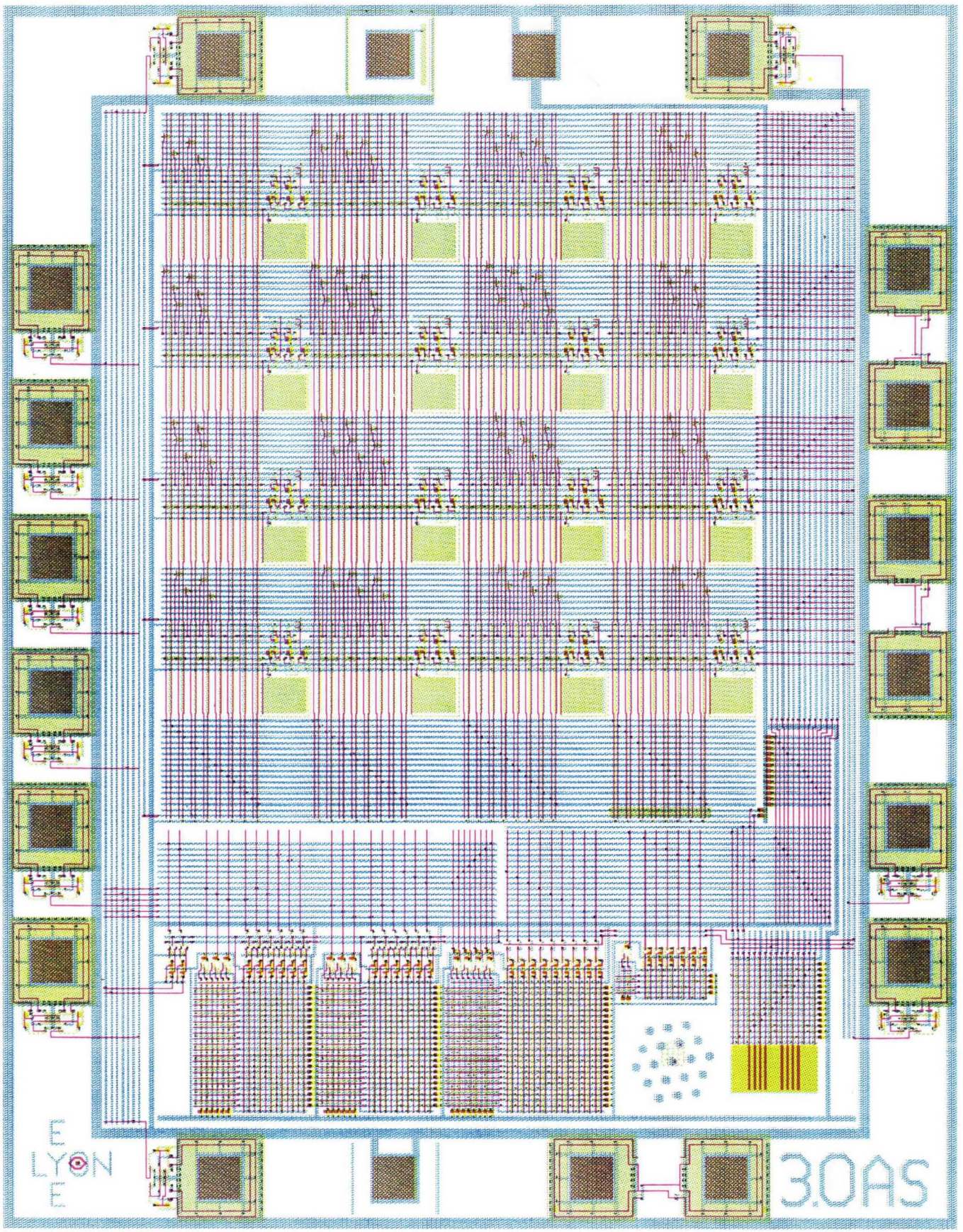
VLSI-81-1 AUGUST 1981

Abstract: The optical mouse is a pointing device for controlling the cursor on a personal workstation display; the mouse is held in one hand and is moved around on a pad to move the cursor on the display. Unlike earlier electro-mechanical mice, motion is detected optically by watching the pad, so there are no internal moving parts. The key to the optical mouse is a sensor chip that reports motion of visible spots relative to the chip coordinate system, using a combination of new techniques. The first technique is a simple "mostly digital" circuit that produces digital image snapshots (bitmaps) of features in a contrasting field, using self-timed circuit techniques and mutually inhibiting sensors. The second technique is the tracking of features in bitmap images, using an easy-to-track contrasting pattern, with a detector array and inhibition network matched to that pattern. The current implementation of the optical mouse chip uses a four-by-four NMOS photo-diode array, and tracks light spots hexagonally arrayed in a dark background.

The optical mouse project illustrates the power of making custom LSI design and implementation capabilities, and simplified design methodologies, available to system designers. After a development effort of less than two man-months, and a total elapsed time of six months from conception, the first implementation of the 3mm by 4mm chip proved to be fully functional and compatible with existing electro-mechanical mice used on personal computers.

A version of this report will be presented as an invited paper at the *CMU Conference on VLSI Systems and Computations*, October 1981.

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304


E
LYON
E
3.0AS

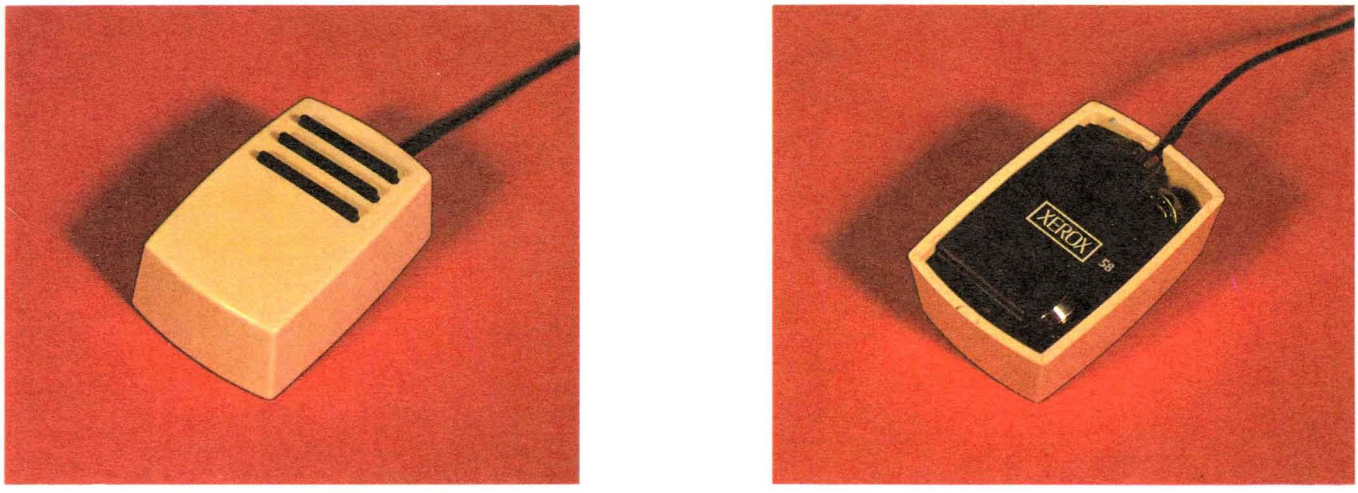


Figure 1a. The Xerox Wheel Mouse

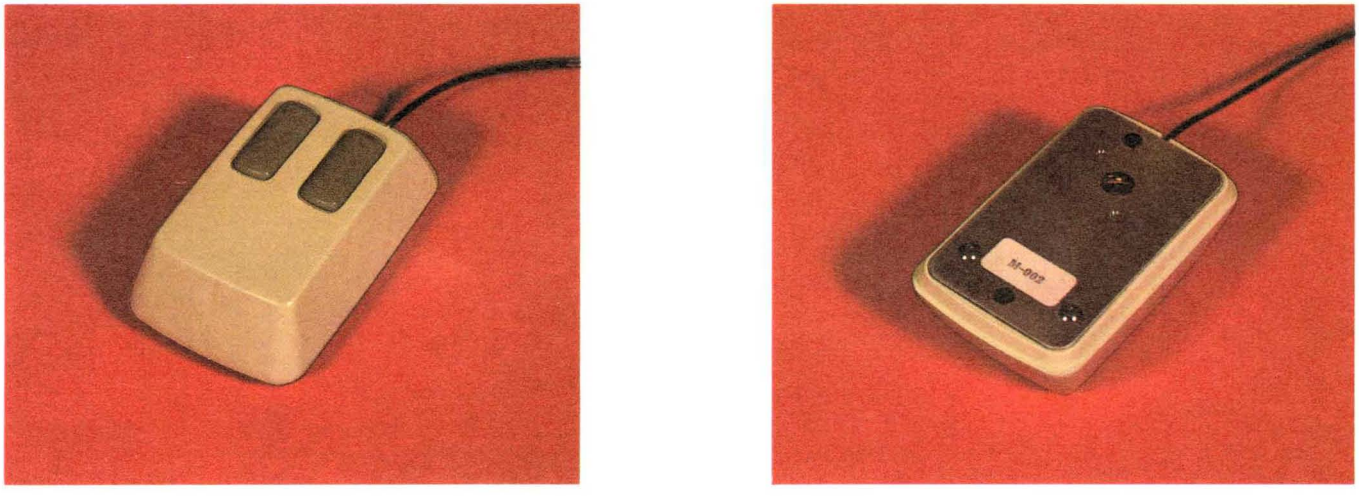


Figure 1b. The Xerox Ball Mouse

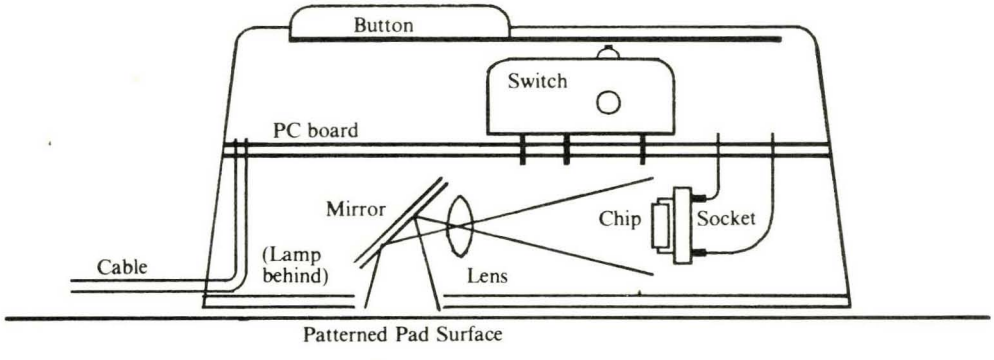


Figure 1c. The Optical Mouse

1. Introduction

A *mouse* is a pointing device used with interactive display-oriented computer systems, which tracks the movement of a user's hand as the user pushes the mouse about on a pad (usually on the work surface next to the user's keyboard). Mice have recently become available in the office products market as a part of the Xerox "Star," the 8010 Professional Workstation [Business 1981, Seybold 1981-1, and Seybold 1981-2].

The work reported here is motivated by the desire for a high-reliability mouse with no moving parts (excluding button switches if any). In Xerox research, the mouse has been in popular use for over eight years, and has been found to be preferable to other pointing devices [Card *et al.* 1977]. However, it has not been outstandingly reliable; the balls or wheels can get dirty and slip on the pad, rather than rolling, or the commutators can get dirty and skip. This is likely to be a significant problem in maintaining workstations that use the mouse in an uncontrolled environment. Another disadvantage of the electro-mechanical mouse is that it's expensive; the one-chip optical mouse is cheap. And the special patterned pad that it needs to make it work is cheap, too, as it can be printed for about a penny on an ordinary ink press.

The goal of a mouse with no moving parts has been achieved through the use of a combination of innovations in electro-optics, circuits, geometric combinatorics, and algorithms, all implemented in a single custom NMOS integrated circuit (patent pending).

2. Background on mouse implementations

Electro-mechanical mice were first developed in the 1960's at Stanford Research Institute, and are described in [Newman & Sproull 1973, Englebart 1970, and Englebart & English 1968, Englebart *et al.* 1967]. The original mouse used a pair of wheels turning potentiometer shafts to encode X and Y motion into analog signals. Each wheel turns as the mouse is moved along its respective dimension, and slips sideways as the mouse is moved in the orthogonal dimension; both wheels turn and slip simultaneously as the mouse is moved diagonally. More recent mice are shown in figure 1, and described below.

The mouse was redesigned at Xerox to use ball-bearings as wheels, and optical shaft encoders to generate a two-bit quadrature signalling code (see figure 2). That is, the motion of a wheel caused the two output bits for that dimension to form square waves in quadrature, with phase and frequency determined by the direction and speed of travel; each bit transition represented motion of one resolvable step, which was used to move the cursor one pixel on the screen [Hawley *et al.* 1975]. The mouse was again redesigned to use a ball instead of two wheels, eliminating the drag of side-slipping wheels [Rider 1974, and Opocensky 1976]; it was built like a trackball [Koster 1967], with shafts turning against the ball, and using commutators as shaft encoders (figure 1b).

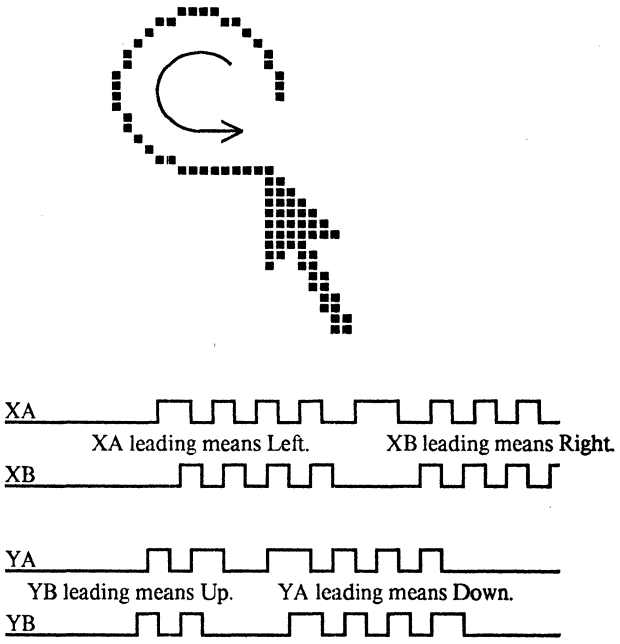


Figure 2. Quadrature Encoding of Pointer Motion on a Bitmap Display System.

The concept of an optical mouse, which "watches" the pad and tracks an image of it, is not entirely new; however, until now the problem of extending the familiar one-dimension quadrature encoding techniques to two dimensions has not been satisfactorily solved. A popular attempt has been to use a "grid-tracking" concept to try to directly emulate the quadrature commutator scheme, using a pair of optical detectors for each dimension. Unfortunately, there is no known easy way to separate the optical images of the lines for the two dimensions, and to make the mouse work even when rotated.

In the electro-mechanical mouse and our new spot-tracking optical mouse, motion is detected relative to the mouse body axes, independent of mouse body rotation; they use "pads" with no inherent coordinate systems. The grid-tracking idea would detect motion relative to the pad axes, and degrade with mouse body rotation. It is not obvious which tracking style is preferable, but the way the electro-mechanical mouse and optical mouse work now is certainly acceptable.

3. Overview of the imager and motion tracker—a smart digital sensor

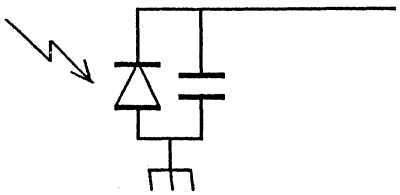
The mechanism described in this paper combines two novel concepts to make a one-chip imaging and tracking system for an optical mouse. The imaging technique may have other applications, as well (but does not compete with dense CCD analog imagers). The optical tracking imager for the mouse application has been implemented in the form of an NMOS chip, which is compatible with the Xerox mouse; it has been packaged in a standard mouse housing, and is in routine use.

- The first concept is a simple "mostly digital" circuit that produces digital image (bitmap) snapshots of bright features in a dark field, using self-timed circuit techniques and mutually inhibiting light sensors (a variation on this technique, which detects dark features in a light field, is also discussed).
- The second concept is a tracking algorithm, involving an easy-to-track contrasting pattern, a detector array and inhibition network matched to the pattern, and the design of the digital machine that takes images of that pattern as input and tracks relative image motion.

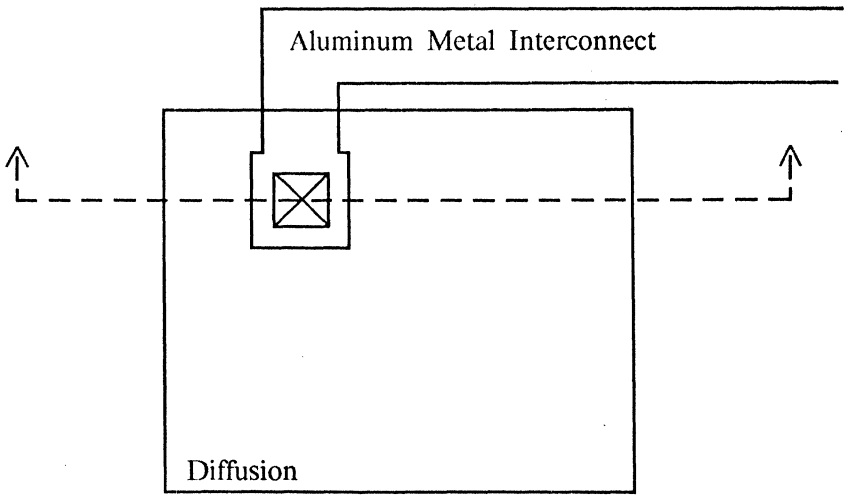
Both concepts apply equally well to either linear or two-dimensional sensor arrays.

There are other novel aspects of the mouse chip, such as the integration of sensors, memory, and logic in a single array, using a standard MOS logic technology. The chip also illustrates several interesting layout, circuit, and timing styles that are widely applicable.

Symbol:



Layout:



Cross Section:

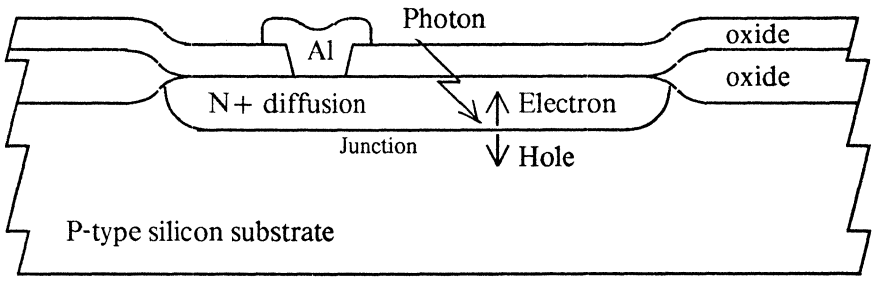


Figure 3.
An NMOS Photo-Diode.

4. Architectural methodology

The optical mouse chip was designed as an experimental application, in a new domain, of the logic, timing, circuit, and layout design methodologies taught by [Mead & Conway 1980]. It was designed with the goal of fab-line and process-parameter independence, so it utilizes only very simple and conservative device models, design rules, circuits, and timing techniques. Those methodologies have been informally extended into an *architectural methodology* for sensors, which have to deal with real-world analog effects and convert them to stable and reliable digital form in the face of wide parameter variations. An architectural methodology is a set of guidelines and constraints that help a designer pick a system architecture, by showing how certain architectural concepts can be implemented and made to work, with high certainty. An architectural methodology for a different domain is discussed in [Lyon 1981].

The layers of design methodologies used to map a concept into a layout must be supported by a compatible implementation system that will map that layout into working silicon. Such a system, described in [Hon & Sequin 1980 and Conway *et al.* 1980], was used to carry out the implementation of the Optical Mouse design as part of a multiproject chip set, on an outside vendor fab line.

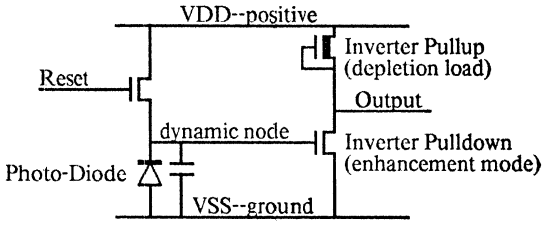
The benefits of this approach are clear in the resulting chip: design time was very short, standard switch-level simulation could be used to verify the correctness of the circuits, the first implementation worked, several orders of magnitude of light-level variation are tolerated, and the techniques developed are very robust against process parameter variation, temperature variation, etc.

The idea of using *lateral inhibition* to make a digital imager was conceived in June 1980; the rest of the techniques discussed here were developed while writing up the inhibition idea, in June and July 1980. A chip design was done quickly in the latter part of July, and was debugged by hand cross-checking of the layout against design sketches (thanks to C. P. Thacker, some bugs were found and corrected). After the chip was into implementation, our tools for design rule checking, circuit extraction, and simulation became more available, and the design was verified as correct except for some non-fatal design rule violations.

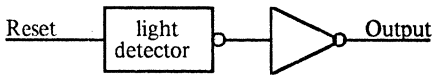
Finished chips were delivered by the implementation system in December, and were quickly tested on a crude test lash-up connected to the mouse port on a personal workstation. Later, with the help of several interested colleagues, a completely packaged mouse prototype based on this chip was completed.

The optical mouse chip should be regarded as only the first representative of a new architectural methodology for smart digital sensors. It seems clear that there will be many more applications of bits and pieces of this methodology to sensors of all sorts. For example, even in something so simple as an analog-to-digital converter, great performance enhancements can be made by using self-timed successive approximation logic to optimize speed while avoiding metastable conditions.

Circuit Diagram (NMOS)



Logic Diagram



Output changes from low to high
at a rate proportional to light level.

Figure 4. Simple "Analog" Imager Cell

5. Digital imager description

Because it is easily available to us at Xerox, the NMOS integrated circuit technology was chosen to implement the optical mouse chip; other technologies, such as PMOS, CMOS, or bipolar, could be used as well. In NMOS, when light strikes the circuit side of a chip, the photons get converted to hole-electron pairs with some reasonable quantum efficiency (see figure 3); the holes are generally attracted to the negative-biased p-type silicon substrate, while the electrons are attracted into n-type diffused source/drain regions and channel regions [Sequin & Tompsett 1975]. Thus, light is detected by collecting negative charge (electrons). If a node is isolated by a turned-off transistor, it is said to be a "dynamic node". A dynamic node which has been charged to a positive voltage will "leak" to a lower voltage as light is received. An imager is simply an array of subcircuits, with a dynamic node in each, which can watch the declining voltages and make a sensible bitmap image from them.

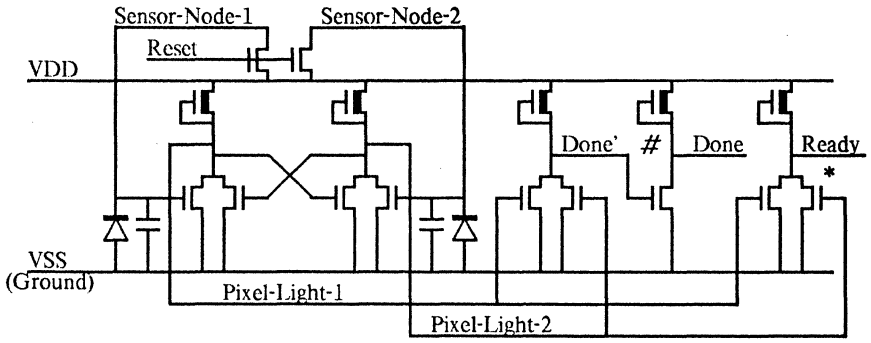
The guts of each imager pixel (subcircuit or cell) is therefore a dynamic node, a transistor to "reset" it high and then isolate it, and an "inverter" circuit to sense the voltage of the node and communicate it out to other circuits. The output voltage from the inverters will start low when the array is reset, then go toward high as the corresponding dynamic nodes go low due to light. Figure 4 shows a schematic diagram of this simple "analog" imager cell.

An array of analog imagers of this sort has a digital all-low output initially, then has an interesting analog image for a while, but eventually ends up in the digital all-high state until it is reset. Both of its digital states are uninteresting. What we would like is a way to get an interesting digital bitmap image reliably. A way to do this is to implement a form of "inhibition" between cells, so that after some cell outputs have gone high, all others are held low and the picture is stable from then on. This is somewhat analogous to the lateral inhibition in the retina of most biological vision systems [von Bekesy 1967]. It has the desirable effect of producing sensible images, almost independent of light level. Such digital sensor arrays can be built in a self-timed loop of logic that recognizes stable images, latches them, resets, and starts over, at a rate roughly proportional to the light intensity.

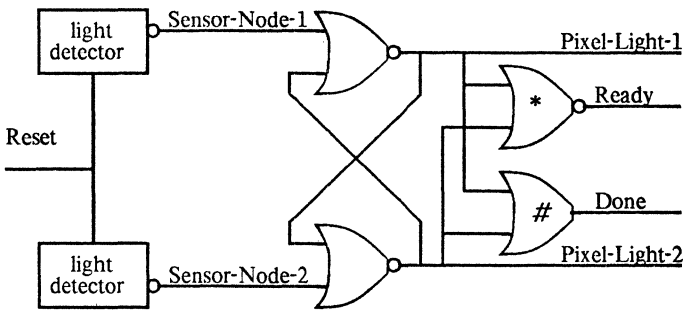
The simplest imager with mutual inhibition is the two-pixel system shown in figure 5. Each pixel circuit is essentially a NOR-gate, with one input of each being the light-sensitive dynamic node, and the second input being the output of the other cell. The initial reset state is 00, with outputs being pulled low by the NOR inputs that are connected to the initially high dynamic nodes. The final state can be either 01 or 10, since 00 will decay with time and 11 is not possible as the output of cross-coupled NOR gates.

The existence of a final state can be sensed by an OR gate whose logic threshold is higher than the thresholds of the pixel NOR gates. Intermediate and metastable states will have both output voltages near the NOR gate thresholds, but distinctly below the OR gate threshold. So this two-pixel digital imager compares the light level at two points, and indicates when it has made a decision (but there is no bound on how long it might take, even in bright light).

Circuit Diagram



Logic Diagram



* Low-Threshold-NOR gate
# High-Threshold-OR gate

Figure 5. Two-Pixel Digital Imager

More complicated logic can be used to detect stable images (Done) in larger sensor arrays with more complicated inhibition NOR networks.

The concept illustrated by the two-element imager is the use of additional transistors to convert the image sensing inverters to cross-coupled NOR gates, as in a flip-flop. Any pairs of elements in an imaging array may be chosen to be connected by these two-transistor mutual inhibition subcircuits. For example, each pixel may be connected with its eight neighbors in a square grid, resulting in nine-input NOR gates.

For any pattern of inhibition and any shape and size image array, the set of possible stable images can be easily enumerated. For example, in a three-by-three array with *neighbor inhibition*, the following eight images can be derived by inspection (notice that all 0 bits are inhibited from changing to 1, by virtue of having a neighbor equal to 1):

```
0 0 0    1 0 1    0 1 0    0 0 0    1 0 1    1 0 0    0 1 0    0 0 1
0 1 0    0 0 0    0 0 0    1 0 1    0 0 0    0 0 1    0 0 0    1 0 0
0 0 0    1 0 1    0 1 0    0 0 0    0 1 0    1 0 0    1 0 1    0 0 1
```

Of course, in larger arrays the images are more interesting, and often more numerous.

In section 9, we will show that by using a four-by-four sensor array, with inhibition of cells up to 2.9 or more pixels away, it is easy to formulate a simple and reliable tracking algorithm that follows spots in a hexagonal array and represents their motion with a quadrature code.

6. Digital imager logic definition

In the mutually-inhibiting detector array, the cells race to see which can be the first within a neighborhood to get enough light and inhibit the others. To formally define the logic of these cells, including general done-detect capability, we use four logic variables in each cell: Sensor-Node, Pixel-Light, Spot-Detected, and Cell-Done. Start by resetting to the state Sensor-Node = 1, Pixel-Light = 0, Spot-Detected = 0, and Cell-Done = 0. Then, with the following logic, wait until Cell-Done = 1 in all the cells:

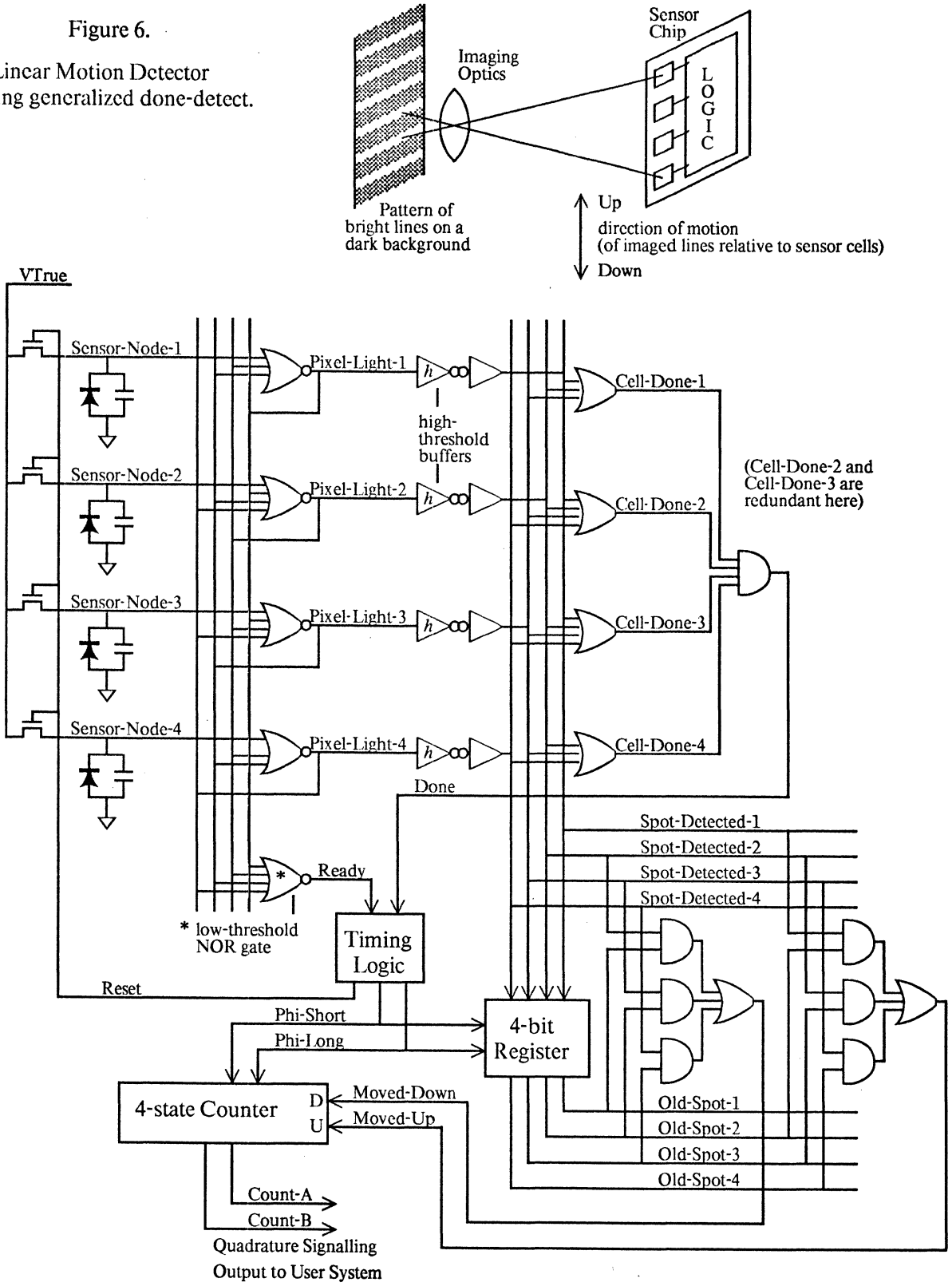
```
Sensor-Node discharges slowly from 1 to 0 as light hits.
Pixel-Light = NOR ( Sensor-Node, Pixel-Light's from other cells in neighborhood )
Spot-Detected = High-Threshold-Buffer ( Pixel-Light )
Cell-Done = OR ( Spot-Detected, Spot-Detected's from other cells in neighborhood )
```

The inhibition network is defined by choosing an inhibition neighborhood for each cell. Generally, we choose neighborhoods symmetrically, such that if A inhibits B, then B inhibits A; we say A "is coupled with" B, reflecting the cross-coupled NOR structure. In many cases, the inhibition neighborhood of some cells will be all other cells in the array; Cell-Done signals from such cells will be redundant, but may be implemented just for the convenience of layout regularity.

Typical Configuration:

Figure 6.

Linear Motion Detector
including generalized done-detect.



Note that we do not use the inhibition NOR gate output itself for done-detection, but a buffered version of it after a high threshold buffer (inverter pair); this is the easiest way to prevent false done-detection during a metastable condition [Seitz 1980]. The buffered signal is not used for inhibition, since that would make it participate in the metastable condition, and because the extra delay would cause oscillatory metastable states.

7. One-dimensional tracking imagers

The simplest application that illustrates the digital imager/tracker is a linear motion sensor, which is built from a row of imager cells looking at white stripes (approximately orthogonal to the row of imager cells) on a dark background. It is possible to apply our digital imager idea directly to the familiar quadrature detection scheme, by using four sensors in two interleaved coupled pairs; *i.e.*, sensors A B C D would have A coupled with C and B coupled with D. The possible stable images that can result are these four:

```
0 0 1 1   0 1 1 0   1 1 0 0   1 0 0 1   (two-pair inhibition)
```

If the white and dark line widths are both equal to about twice the sensor spacing, these images correspond in an obvious way to positions of the stripes relative to the sensors. Any two adjacent sensor outputs, say A and B, can be used directly as quadrature output signals that sequence through the states 00, 01, 11, 10, forward or backward, depending on the direction of motion. The advantage over previous optical quadrature detectors is that no fixed threshold or specific light level is needed. The sensors will cycle at a rate depending on the light level, and latched outputs will be made available to the host system.

Another linear tracking scheme that is closer in spirit to our two-dimensional tracker uses narrow white lines (about one-third white) and a different inhibition pattern. If four imager cells are used, and we arrange to have each cell inhibit cells up to two steps away (say cells at distance less than 2.5), then we get a set of three stable images, shown here:

```
1 0 0 1   0 1 0 0   0 0 1 0   (radius 2.5 inhibition)
```

If the white line spacing (imaged onto the chip) is about three cell widths, then these images correspond in an obvious way to positions of the bright lines relative to the cells (1=bright); see figure 6. The figure illustrates a simple digital machine (on the same chip) that would compare the current image with the previous image (*i.e.*, the machine has only three states) and output a signal that says *moved up* or *moved down.* Thus we have a relative motion sensor for one dimension of travel. A 2-bit counter is used to convert to the familiar quadrature signal representation which is convenient for asynchronous sampling by the host system.

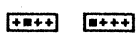
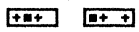
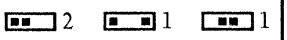
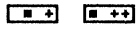
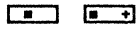
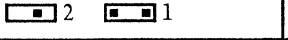
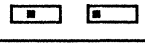
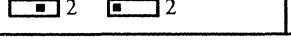
Inhibition Neighborhoods	Inhibition Radius	Stable Images and how many of each	Total Images
	<1	 1	1
	—	 2 1 1	4
	1.5	 2 1	3
	2.5	 2 1	3
	3.5	 2 2	4

Figure 7a. Inhibition neighborhoods and stable images for a four-by-one sensor array.

Black cells are "responsive" (a dot is detected there).
White cells are "inhibited" and not responsive.
Cells with a + are not inhibited by the indicated responsive cell.

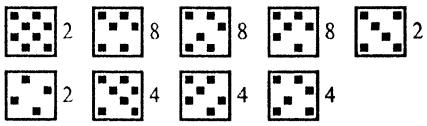
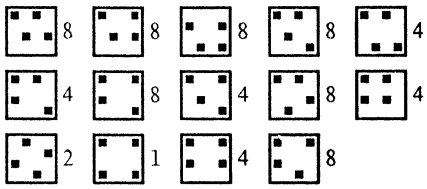
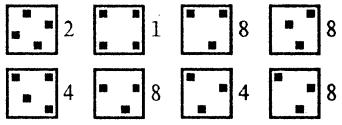
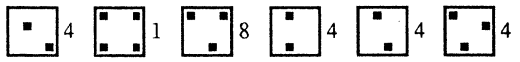
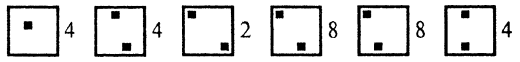
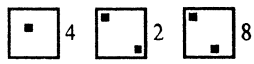
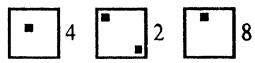
Inhibition Neighborhoods	Inhibition Radius	Stable Images and how many of each	Total Images
	<1	 1	1
	1.1	 2 8 8 8 2 2 4 4 4	42
	1.5	 8 8 8 8 4 4 8 4 8 4 2 1 4 8	79
	2.1	 2 1 8 8 4 8 4 8	43
	2.3	 4 1 8 4 4 4	25
	2.9	 4 4 1 8 4	21
	* 3.0s	 4 4 2 8 8 4	30
	3.1	 4 4 2 8 8	26
	3.2	 4 2 8	14
	3.7	 4 2 8	14
	4.3	 4 4 8	16

Figure 7b. Inhibition neighborhoods and stable images for a four-by-four sensor array.

* "3.0 special" was chosen for the optical mouse; its images are simply characterized as either a central dot or a pair of dots on opposite edges but not sharing an edge.

Other spacings, inhibition patterns, numbers of cells, etc., can be applied easily to the linear motion detector problem. The real challenge is to make it work in two dimensions, and to make it tolerant of rotation (of the imager with respect to the pattern). After discussion of inhibition patterns, we show how to extend the 4-element one-dimensional line-tracker to a 4-by-4-element two-dimensional dot-tracker.

8. More about inhibition

First we need to understand patterns of inhibition. We can do this with pictures like those above, but showing only a single 1 (in each possible position) and the set of elements that are inhibited (forced to 0) by being coupled with it. Other elements remain unknown and are designated + (at least one +, if any exists, must change to a 1 to make a stable image).

For the first one-dimensional tracker, the inhibition patterns are these:

```
1 + 0 +    + 1 + 0    0 + 1 +    + 0 + 1    (two-pair inhibition)
```

And for the second they are these:

```
1 0 0 +    0 1 0 0    0 0 1 0    + 0 0 1    (radius 2.5 inhibition)
```

In many cases, we can specify inhibition neighborhoods as all cells within a certain radius, by Euclidean distance in the plane, assuming unity cell spacing. We choose a radius such that no cells fall at exactly that radius, to avoid ambiguity; hence radius 1.5 means cells at distance 1.414 in the plane are inhibited, but cells at distance 2.0 are not. Some inhibition neighborhoods, however, cannot be specified simply by a radius; two-pair inhibition is an example.

Figure 7 graphically tabulates a succession of inhibition neighborhoods and the resulting stable images, for four-element linear sensor arrays and four-by-four two-dimensional sensor arrays. Square symmetry is assumed to reduce the complexity of the figure.

Notice that radius 2.9 is the smallest inhibition neighborhood such that when comparing images, no dot can appear to have moved to two different adjacent pixels. That is, this sequence cannot occur:

```
   old          new
0 0 0 0      1 0 0 0    (moved up-left or down-right?)
0 1 0 0      0 0 0 0    (can't happen for radius > 2.83)
0 0 0 0      0 0 1 0
0 0 0 0      0 0 0 0
```

What appears to be most useful is the "3.0 special" pattern of inhibition, a cross between the radius 2.9 and radius 3.1 patterns (radius 3.0, where points separated by exactly three pixels are coupled only if they are corners). The stable images that can result from this inhibition pattern fall into two classes: Either there is a single 1 in the central quad of pixels, or there are two 1's on

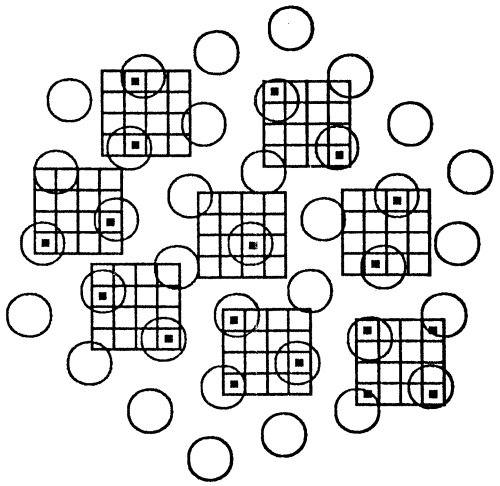


Figure 8.
Various positions of 4x4 imagers with respect to a hexagonal dot array,
showing ways to see all the possible stable images for radius 2.9 or more inhibition.

opposite edges (but not on adjacent corners). If the 1's represent white dots being imaged onto the chip, any motion of the dots forming the image will either leave one or two dots within the field of view, or one dot will leave the field of view and the other will stay. So there is always a dot to track.

9. Image tracking using bitmaps

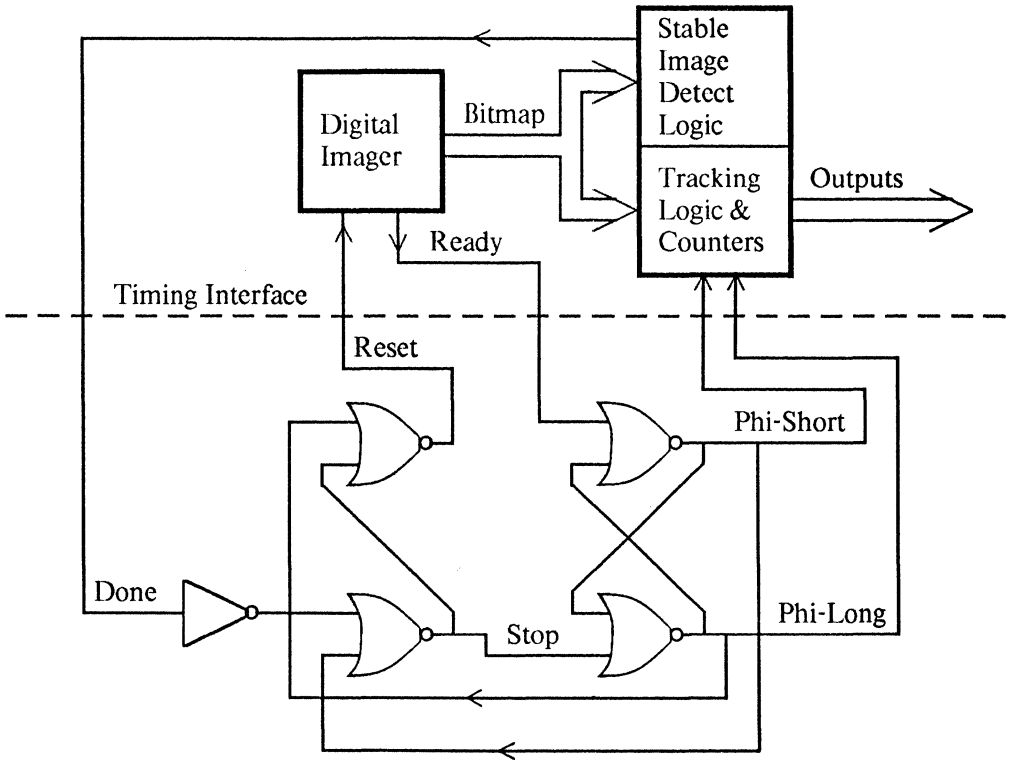
The general tracking concept is to use a hexagonal array of white dots (which just looks like dots of constant spacing but no particular orientation when seen through a small window at an arbitrary angle), and to pick a dot spacing such that bitmap images can be associated with the dot array easily and movement can easily be detected by comparing successive snapshots. The white dot spacing should be slightly more than the inhibition distance, as a general rule of thumb. For example, using radius 2.9, "3.0 special", or 3.1 inhibition with a four-by-four sensor array, we recommend a dot spacing of about 3.4 pixels, because that is about the average distance between dots in the stable images with two dots. Then the dots in the stable images correspond in an obvious way (see figure 8) to positions of one or more dots of the hexagonal dot array.

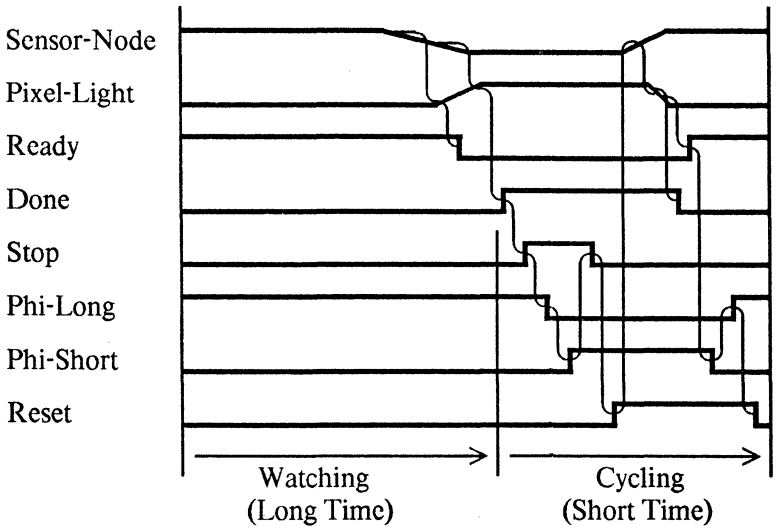
If we use radius 2.9 inhibition instead of "3.0 special" or 3.1, the "four-corners" image would give us an interesting problem. Although the images of two and three dots are easy to integrate into a set of images of dots in a hexagonal array, the image of four dots is not. Worse than that, it is possible for a positioning of two dots near opposite corners to force the four-dot image to occur; then it is impossible to tell in which pair of opposite corners the dots were really seen. This is why the "3.0 special" pattern was developed--it eliminates the four-corner image and the images of three dots, while still allowing all the images of two dots, some of which would have been eliminated by going to radius 3.1. The images of three dots are not really missed, since seeing only two of the three dots still guarantees that with movement at least one of the dots will remain in the field of view, so the image can be tracked by looking at local dot motion.

Counting all rotations and mirrorings, there are 30 distinct stable images for the "3.0 special" inhibition. Of the 900 combinations of two successive stable images, most have an obvious interpretation in terms of movement of the white dots with respect to the imager; those that do not have an obvious interpretation must be handled by the tracking algorithm, but will probably not occur often.

A possible non-specific implementation of the tracking algorithm is simply a finite-state machine which takes one stable image as input (possibly encoded in just a few bits), looks also at its current state (state equals previous input, most likely), and outputs a signal indicating direction of movement based on the state and input, and also outputs a new state. If the machine is built of a simple PLA (programmed logic array) with no special encoding, the PLA can have as many as 32 inputs and 900 product terms, which would occupy most of a reasonable size NMOS chip. The size could be reduced by first encoding the 30 images into 5 bits (PLA with 10 inputs instead of 32),

Figure 9. Imager and Logic tied together by Self-timed Clock Circuit, with timing waveform diagram.





and by not decoding image pairs which are meaningless or which correspond to no motion (maybe about 600 terms instead of 900); so it may fit in a quarter of a chip. We are still free to design the tracking algorithm and specify PLA outputs required, and program the PLA accordingly (*i.e.* the tracking problem may be regarded as a simple matter of programming). A more specific tracking algorithm and a novel compact implementation of it will be described in section 11.

10. The self-timed action of the imaging/tracking system

Before going into tracking algorithms, we should indicate how the imager and the synchronous finite-state machine get tied together by timing logic to make a self-timed machine, and how they control the output logic that generates the two pairs of quadrature signals that the host computer wants to see as an indication of mouse movement.

What is needed is a circuit which will generate two-phase nonoverlapping clock signals to run the digital logic, such that each cycle is synchronized to the reset-done cycle of the imager. This same clock runs an up-down counter controlled by the PLA for each of X and Y, to generate quadrature signals which can be communicated off chip. So we have three things to design, the particulars of which are not interesting in isolation: done and ready detectors, clock and reset signal generation circuit, and up-down counter with quadrature outputs.

These parts are blocked out, along with logic-level and timing details of the clocking circuit, in figure 9. Clocks are generated through a delay-independent (self-timed) handshake with the imager array, and it is assumed that the digital logic is fast enough to keep up with the imager (this assumption becomes a constraint for the designer to satisfy). The generated clocks are called Phi-long and Phi-short, to indicate which one is of unbounded length; Phi-long should be used as a quasi-static feedback enable to keep the logic alive and insensitive to light while waiting for the imager. The steps of operation of the clock generator are in quick succession as follows:

Start in the initial sensing state, just after Reset ← 0;
  Ready = 1 (meaning all Pixel-Lights are 0),
  Done = 0 (meaning not a stable image),
  Phi-long = 1 (this is during the long, or waiting, clock phase),
  Phi-short = 0 (because the other phase is 1),
  Stop = 0 (because not Done yet).
After a little light is received, some Pixel-Light output starts toward 1; then:
  Ready ← 0 (at some irrelevant time before the picture is stable).
When enough light is received, one or more Spot-Detected's goes to 1, the picture becomes one of the stable images, and this happens:
  Cell-Done ← 1 in all cells,
  Done ← 1,
  Stop ← 1,
  Phi-long ← 0,
  Phi-short ← 1,
  Stop ← 0,
  Reset ← 1,

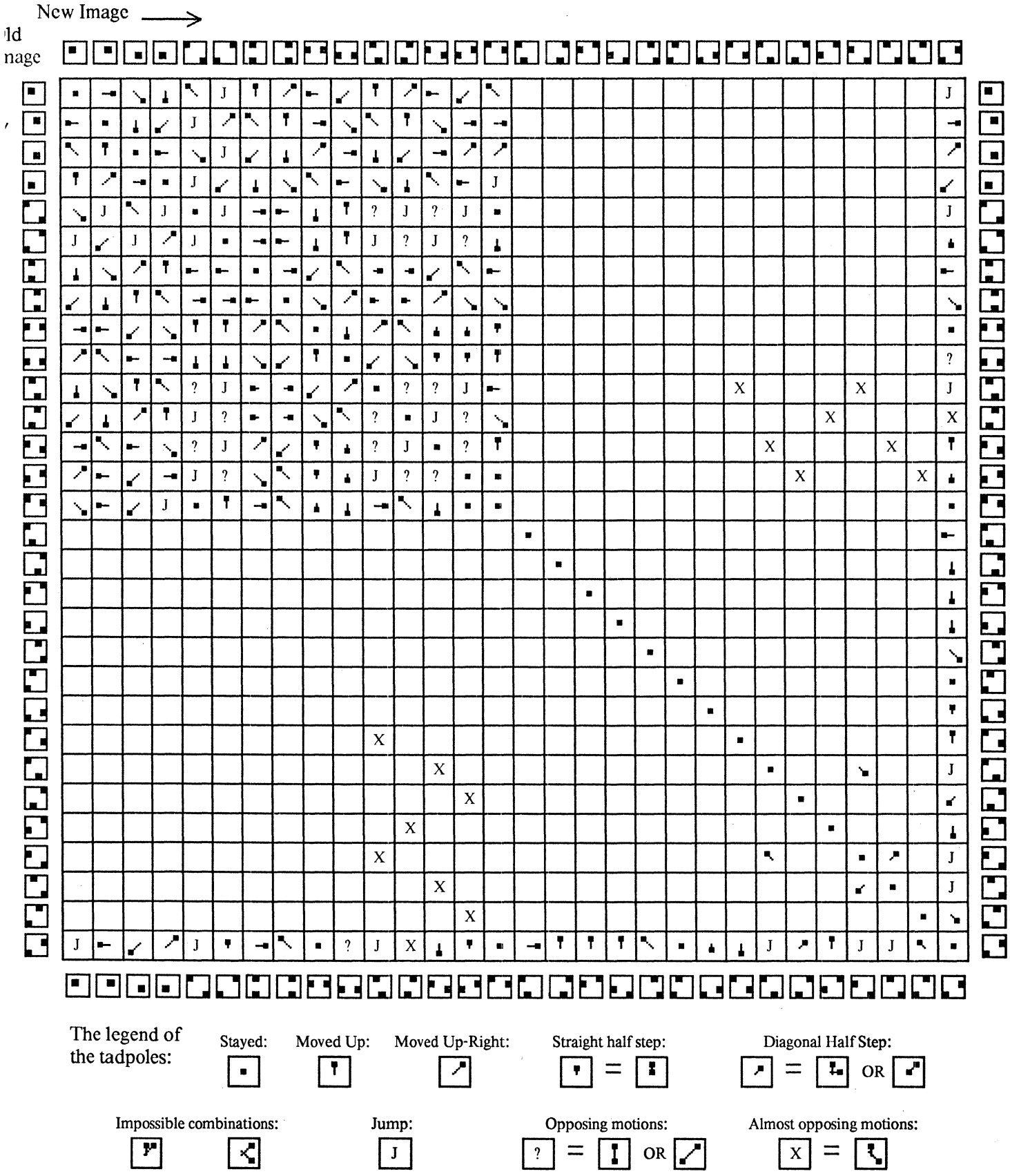


Figure 10.
The "900 little squares" approach to tracking bitmap images in the optical mouse chip.

Sensor-Node ← 1 in all cells,
Pixel-Light ← 0 in all cells,
Spot-Detected ← 0 in all cells,
Done ← 0,
Ready ← 1,
Phi-short ← 0,
Phi-long ← 1,
Reset ← 0,

And it is all back where it started, having gone through a cycle.

The good thing about this technique is that it doesn't care how slow the imager is; everything is willing to wait till there is a solid digital answer. Hopefully, the imager will receive enough light to cycle faster than once every few hundred microseconds on the average, so it will be able to get image samples often enough to track mouse motion of several thousand steps per second.

The counters needed for X and Y simply count through four states, in either direction (up or down), changing only one bit at a time (*i.e.*, 00, 01, 11, 10). This is a simple case of either a Gray-code counter or a Johnson counter (Moebius counter). The PLA (tracker machine) outputs needed to control the counters are just Right-X, Left-X, Up-Y, and Down-Y.

In the scheme actually implemented, the counters run through eight states, so that the tracking algorithm can report a finer gradation of motion (Up-Half-Y, etc.). Only four states, representing full steps, would be seen by the host system; the states mentioned above are simply augmented by an alternating "least significant bit", so the eight-state sequence is 000, 001, 010, 011, 110, 111, 100, 101.

11. Designing and implementing a tracking algorithm

That brings us back to tracking algorithms. The simplest "algorithm-design" technique is to get a big piece of paper, draw the 30 stable images across the top and again down the left side, and make 900 little squares to fill in. For each combination of an old image from the left edge and a new image from the top edge, write in the square which way it looks like the dots moved, and by how much (half step or full step).

Figure 10 is a partially filled-in table of moves, to illustrate the concept. One quickly develops simple algorithms to describe the reasoning about filling in the squares. But how do we write some simple rules to do this in a digital machine, without resorting to precomputing all the cases? To fit the capabilities of VLSI, we have come up with a distributed local algorithm which can be implemented right in the imager array. Each pixel saves its old value in a register, and on each cycle compares it with its new value and that of all its neighbors. Each pixel reports one of eleven results (my dot moved <to one of 8 neighbors>, my dot stayed, my dot disappeared, or I didn't have a dot to track) to some decision logic. The decision logic then just has to see what gets reported, and filter out contradictions (a move and a stay can be converted to a half-step move).

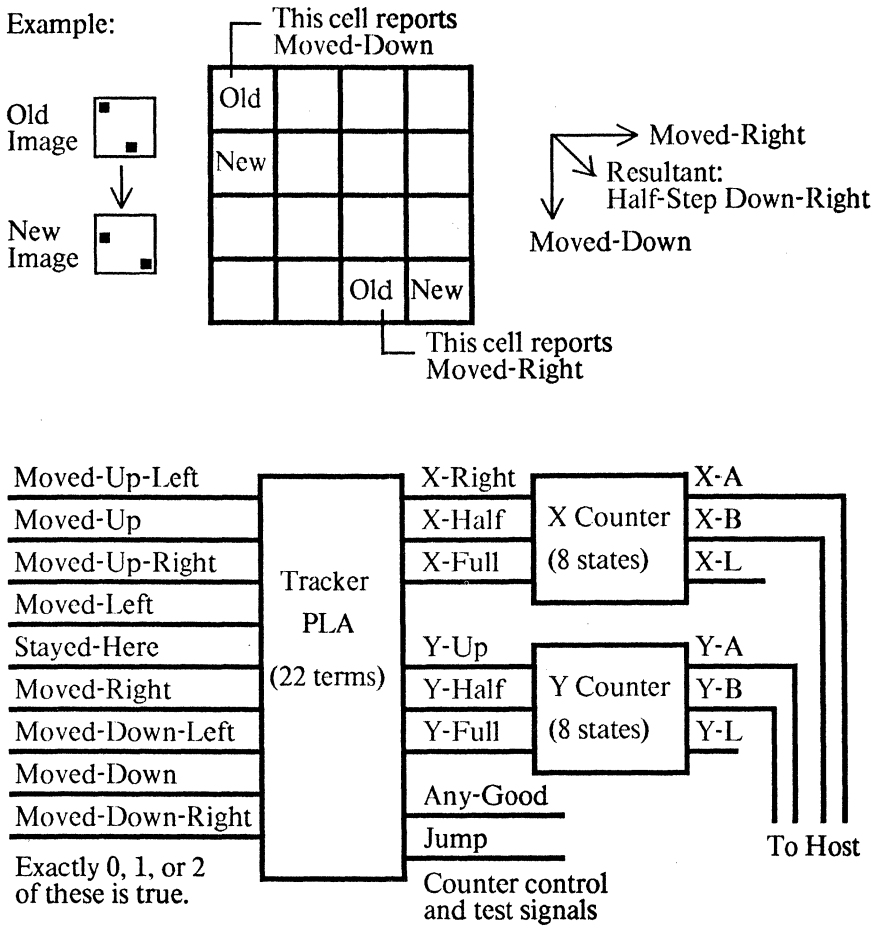


Figure 11. Tracking Spots by Comparing Images

The decision logic can also be partially distributed as a set of nine AND-OR-INVERT gates running through the array (one for each of the eight move directions and one for the no-move case—disappearing dot and no dot to track are not reported). These gates report a low logic state if a pixel had a dot in the old picture, AND the appropriate neighbor has a dot in the new picture, OR any other pixel met a similar condition. A single 9-input conflict resolution PLA is needed outside the array to decode combinations of zero, one, or two reported move directions and to produce the counter control signals (see figure 11). Actually, of the 36 conceivable patterns of more than one indicated movement, only twelve are both possible and clearly meaningful (as half-steps); so the logic can be very simple (PLA with only 20 terms, for the eight possible full-steps and the eight possible half-steps, four of which occur two ways). Any other sequence, whether sensible or not, will produce no count commands. The eight-state up-down counters are also most easily designed as PLA's.

The algorithm just described gives results which agree with the table of examples above (with appropriate conflict resolution logic programmed in the PLA to generate the half-steps). Question marks are interpreted as no move.

12. The mouse chip layout

A mouse chip has been designed in NMOS, as a direct one-chip substitution for the existing electro-mechanical mouse works (to go with a light and three button switches). For complete compatibility, the chip includes debounce electronics for the button switches. A floorplan of the chip is shown in figure 12. It is about 3.5mm by 4.5mm in a typical NMOS process (with lambda=2.5 microns, or 5-micron lines).

There is a single layout for a programmable sensor and logic cell, which can be customized for each position in the array to implement any inhibition pattern and the described tracking algorithm. The logic to detect a stable image is also partly programmable and distributed. The cell layout with programming for the top left position is shown in figure 13. A logic diagram with more details of the overall chip function is shown in figure 14.

The layout style used in this first version of the chip treats a sensor cell with its logic and memory as a low-level cell, and constructs the array by selective programming of the cells in different positions. This approach costs large amounts of wiring area, since every cell has to have access to every other cell's `Pixel-Light` line. This area penalty was not regarded as a problem, until it was realized that it causes a related light sensitivity problem—about 90% of the photons get lost in the wires, far from the sensor nodes where they could do some good. To improve light sensitivity, and also to improve the magnification ratio needed in the optical path, we have switched to a new layout style, using a densely packed array of N+ diffused areas as sensor nodes, with all logic in compact regular structures outside of it. A new chip based on this approach has been designed by M. P. Haeberli, and will be the subject of a future report.

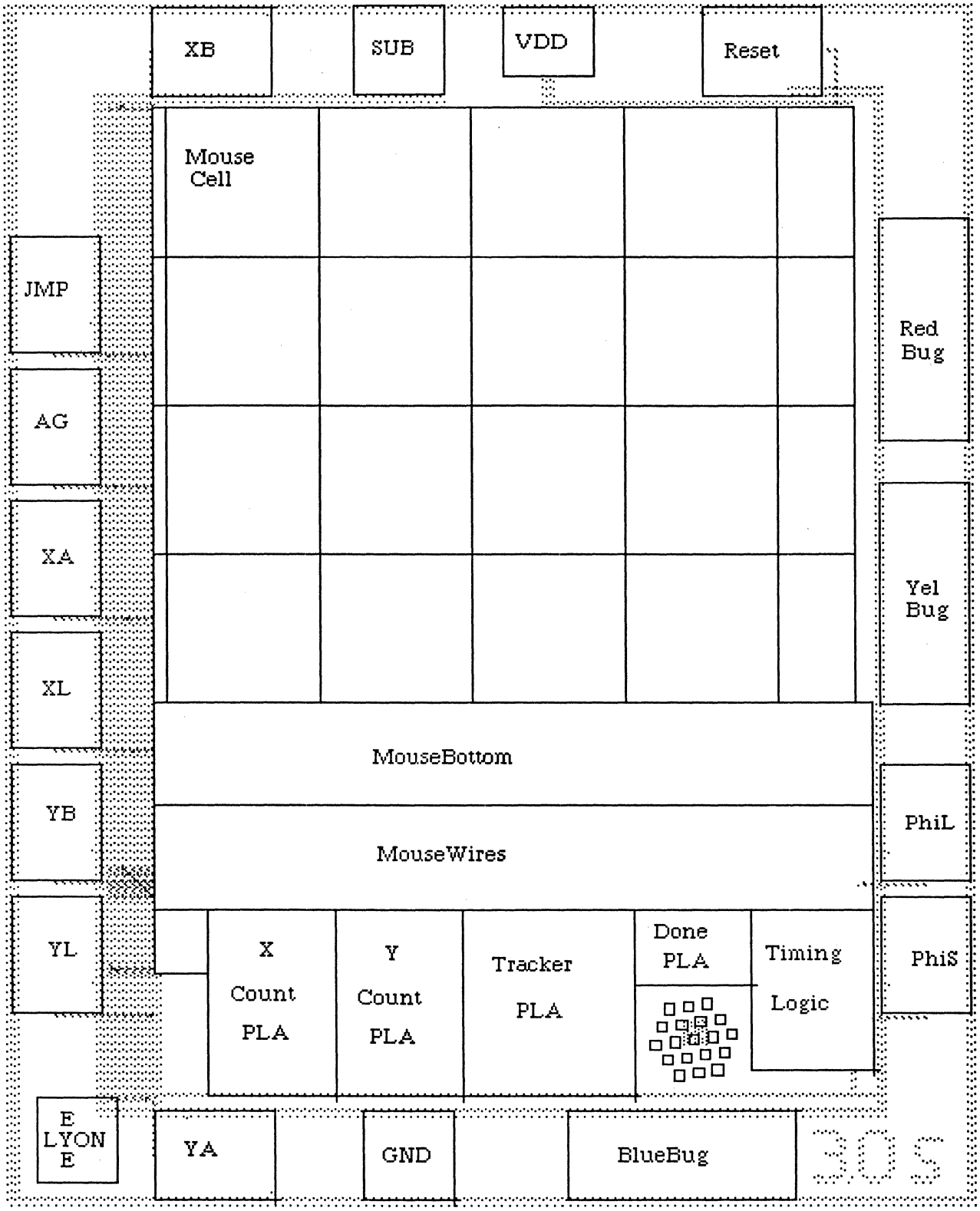
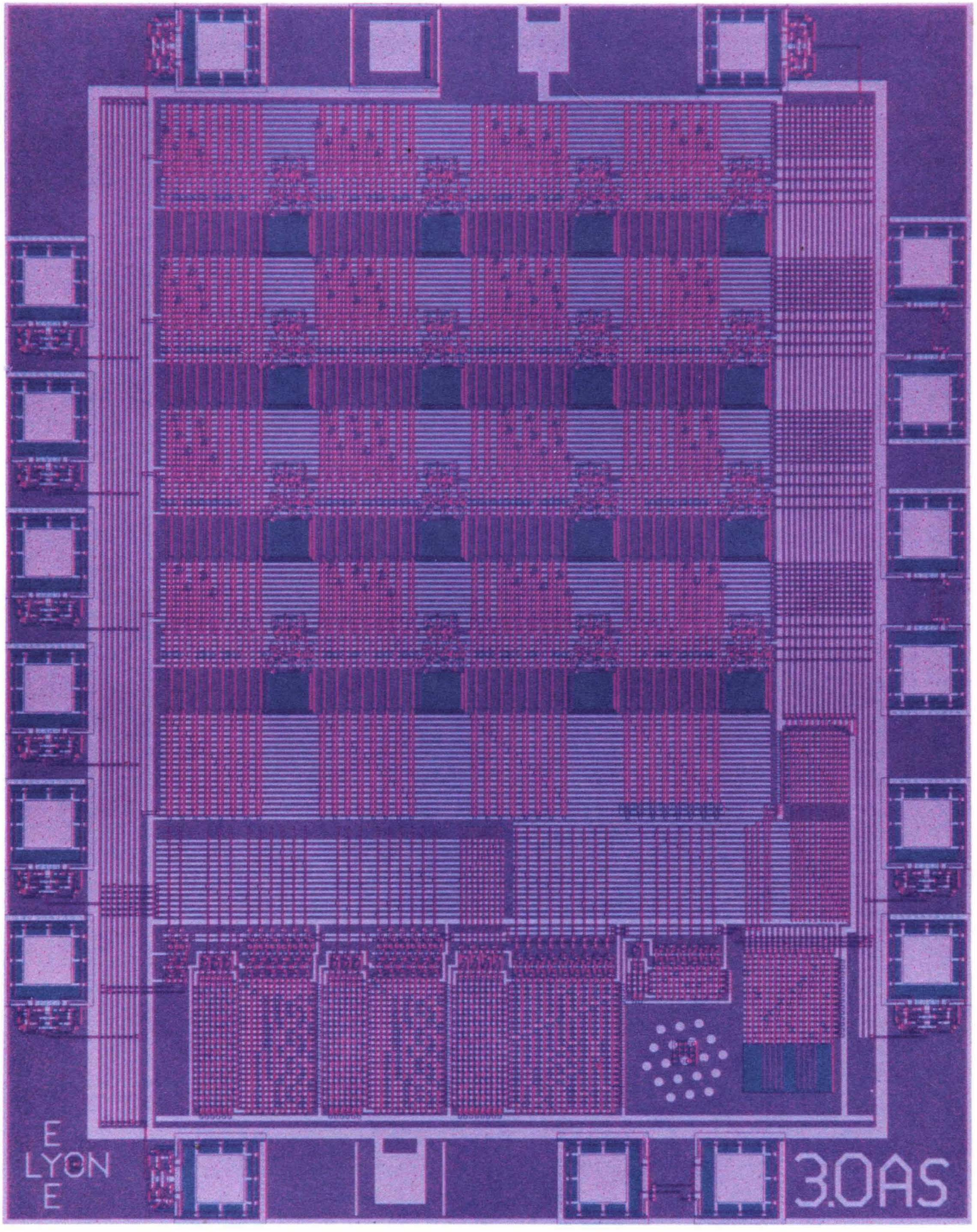
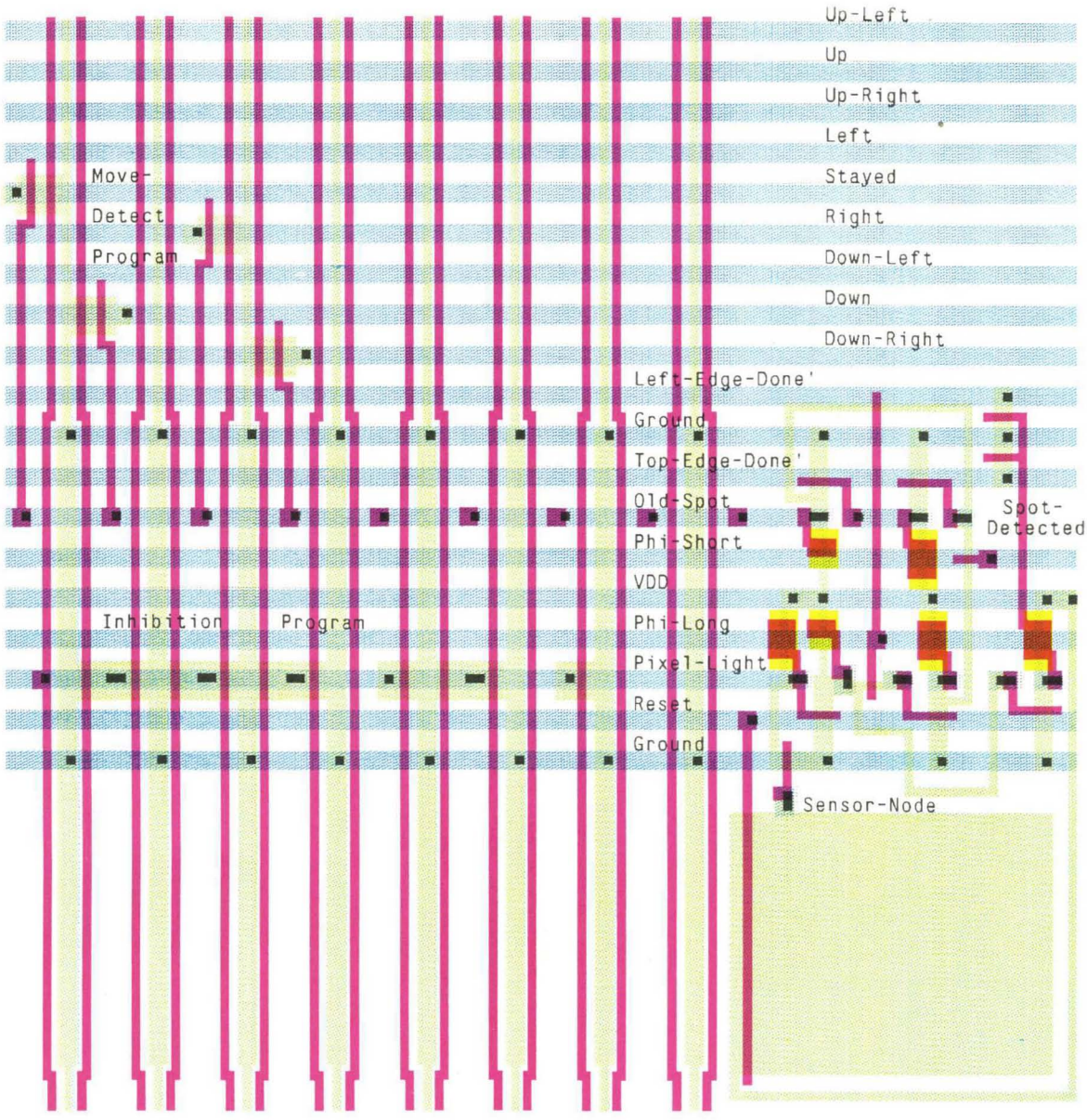


Figure 12. Floorplan of the Optical Mouse Chip


E
LYON
E
3.0AS



Pixel-Light distribution wires (red)

Figure 13
The Layout of the Upper Left Optical Mouse Cell

One other layout feature of note is the regular structure used for the "random" timing logic. It is essentially just like one plane of a PLA, except that it can also be programmed with contacts between the lines running orthogonally through it. With a bit of optimization, this becomes topologically identical to I^2L-style gate-matrix layout. Look for more on this in a future report.

13. Tracking dark spots, instead of light spots

If we redefine the logic of the sensor cells, we can make an array that looks for a set of images of dark spots in a light field; this approach has some different and interesting properties. In the design previously described (light-spot detector), the cells race to see which can be the first within a neighborhood to get enough light and inhibit the others; in this new technique, the cells want to see which can be the last to get enough light—which requires quite a different logical approach. To define the logic of this new cell, we use five logic variables in each cell: Sensor-Node, Pixel-Light, Pixel-Dark, Spot-Detected, and Cell-Done. Start by resetting to the state Sensor-Node = 1, Pixel-Light = 0, Pixel-Dark = 1, Spot-Detected = 0, and Cell-Done = 0. Then, with the following logic, wait until Cell-Done = 1 in all the cells (Sensor-Node goes slowly from 1 to 0 as light hits):

```
Pixel-Light   =  NOR ( Sensor-Node, Spot-Detected )
Pixel-Dark    =  Invert ( Pixel-Light )
Spot-Detected =  NOR ( Pixel-Light, Pixel-Dark's from other cells in neighborhood )
Cell-Done     =  High-Threshold-OR ( Pixel-Light, Spot-Detected )
```

A simple three-pixel example, diagrammed in figure 15, will serve to clarify the properties of this kind of detector array. Note that when all cells have received light, it is possible for the array to arrive at a stable state in which no dots were detected (Spot-Detected = 0, Pixel-Light = 1 in all cells). Any set of dots which is a *subset* of an image that would have been detected by the equivalent inhibition pattern in a light spot detector array is a possible stable image.

Therefore, for the three-pixel neighbor-inhibiting dark spot sensors, we get these stable images:

```
1 0 1 (1)    0 1 0 (1)    ("complete" images)

1 0 0 (2)    0 0 0 (1)    ("subset" images)
```

For four-by-four arrays, the additional stable "subset images" are illustrated in figure 16. One result is that with the radius 2.9 inhibition pattern, seeing spots on opposite corners does not force the four-corners image, but is actually most likely to give the correct two-corners image. A more general result is that the spot pattern to be tracked does not need to be so closely matched to the inhibition pattern, since the circuit is willing to wait for spots to really be there before it claims to see them; a pseudo-random distribution of dots would probably work quite well.

With this technique, it would be possible to make a linear motion tracker with only three cells, each inhibiting all the others, with a dark line spacing of three cells or greater; similarly, a 2-D

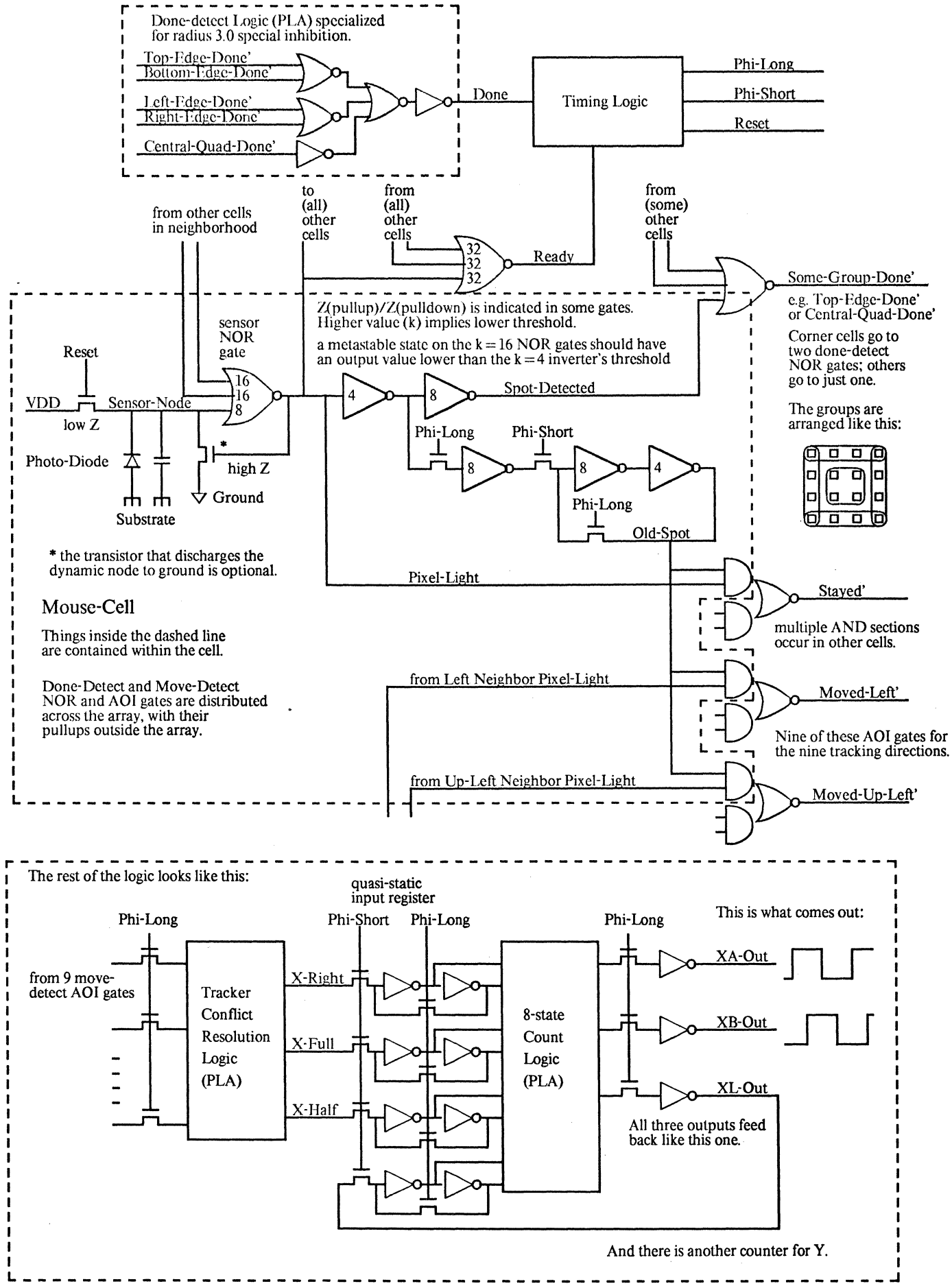
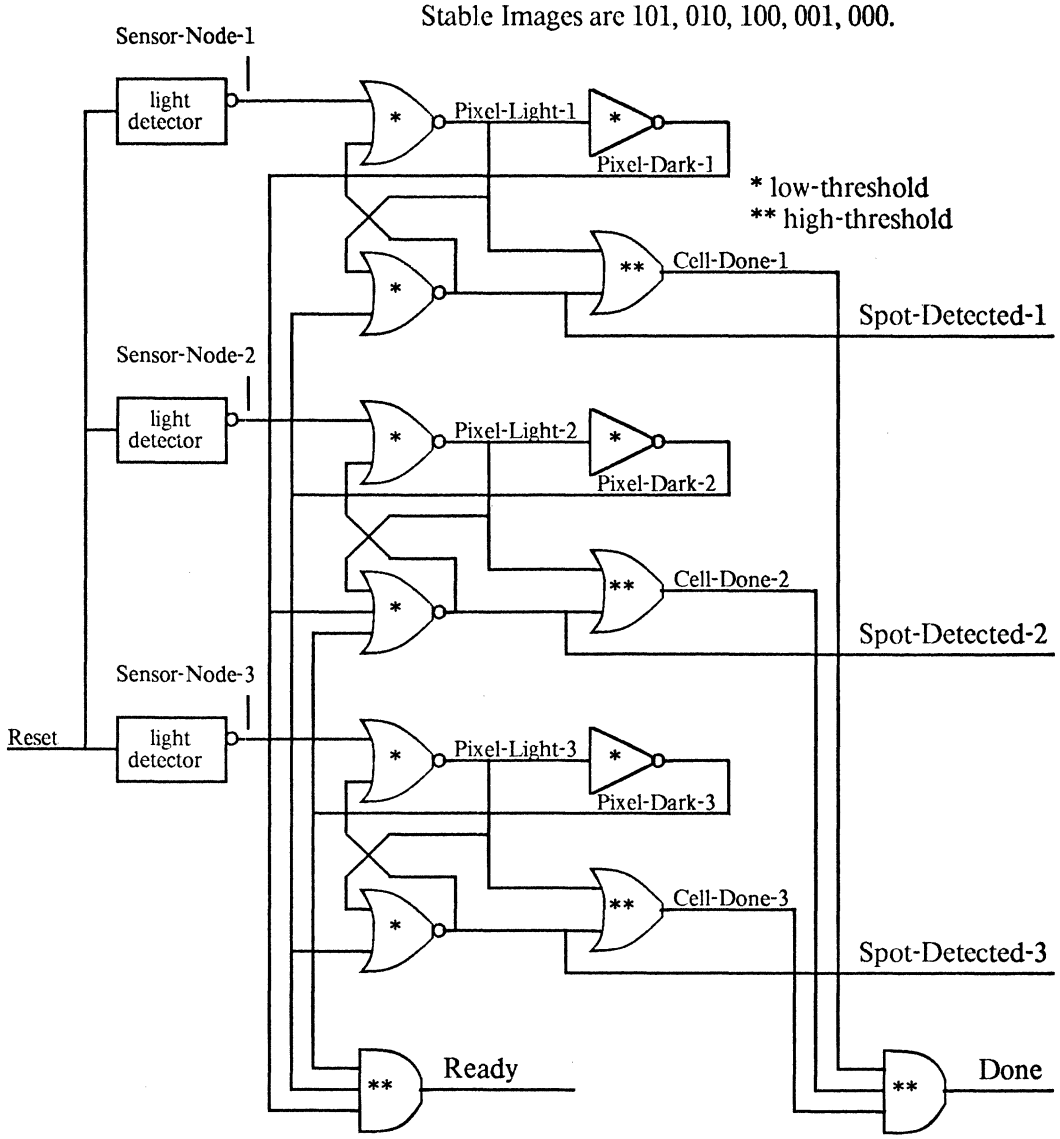


Figure 14.
Details of the Logic of the Prototype Optical Mouse Chip

Figure 15. Three-pixel Dark-spot Sensor Example,
with nearest-neighbor inhibition.
Stable Images are 101, 010, 100, 001, 000.



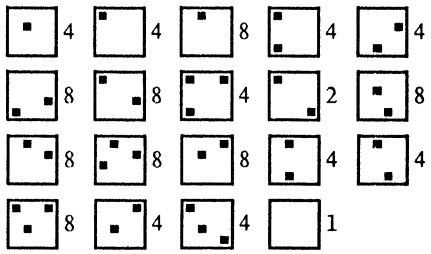
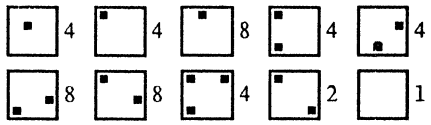
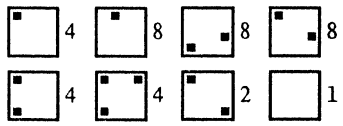
Inhibition Radius	Additional "Subset" Images seen by dark-spot sensor array	Grand Total
2.1		146
2.3		72
* 2.9		60
3.0s		43
3.1		39

Figure 16.

Additional patterns for the four-by-four dark-spot detector array.

* Radius 2.9 may be best for the dark-spot detector scheme.

tracker might be built with just a three-by-three array of cells. For the linear tracker, the image sequence for uniform motion could be either the 100, 010, 001, 000 cycle or the 100, 010, 001 cycle. These trackers would have to assume that a dot disappearing from one edge and/or appearing on the other represents a step of motion (or a half step, depending on what assumptions are made about the line spacing).

14. Of mice and pens

The optical mouse's compact internals will allow it to be repackaged into various other forms. For example, a pen-like device with a big base that keeps it from falling over might be desirable. A "ball-point" tracking device that watches a golfball-like pattern of dots on a rolling ball in the tip of a pen may also be useful.

15. Summary

The optical mouse embodies several ideas that are not obvious extensions of standard digital or analog design practices, but which contribute to the design of robust sensors of the analog-to-digital sort. Using the concept of lateral inhibition, sensor cells that are trivial and useless alone become powerful in their synergism. A sensor array that forces itself into a useful and informative stable digital state is very easy to deal with, through standard digital techniques. It is especially useful if it can decide when it has reached such a stable state, and when it has been reset enough to be ready to start over, for then it can be regarded as self-timed, and clocks can be generated that cycle it quickly yet reliably.

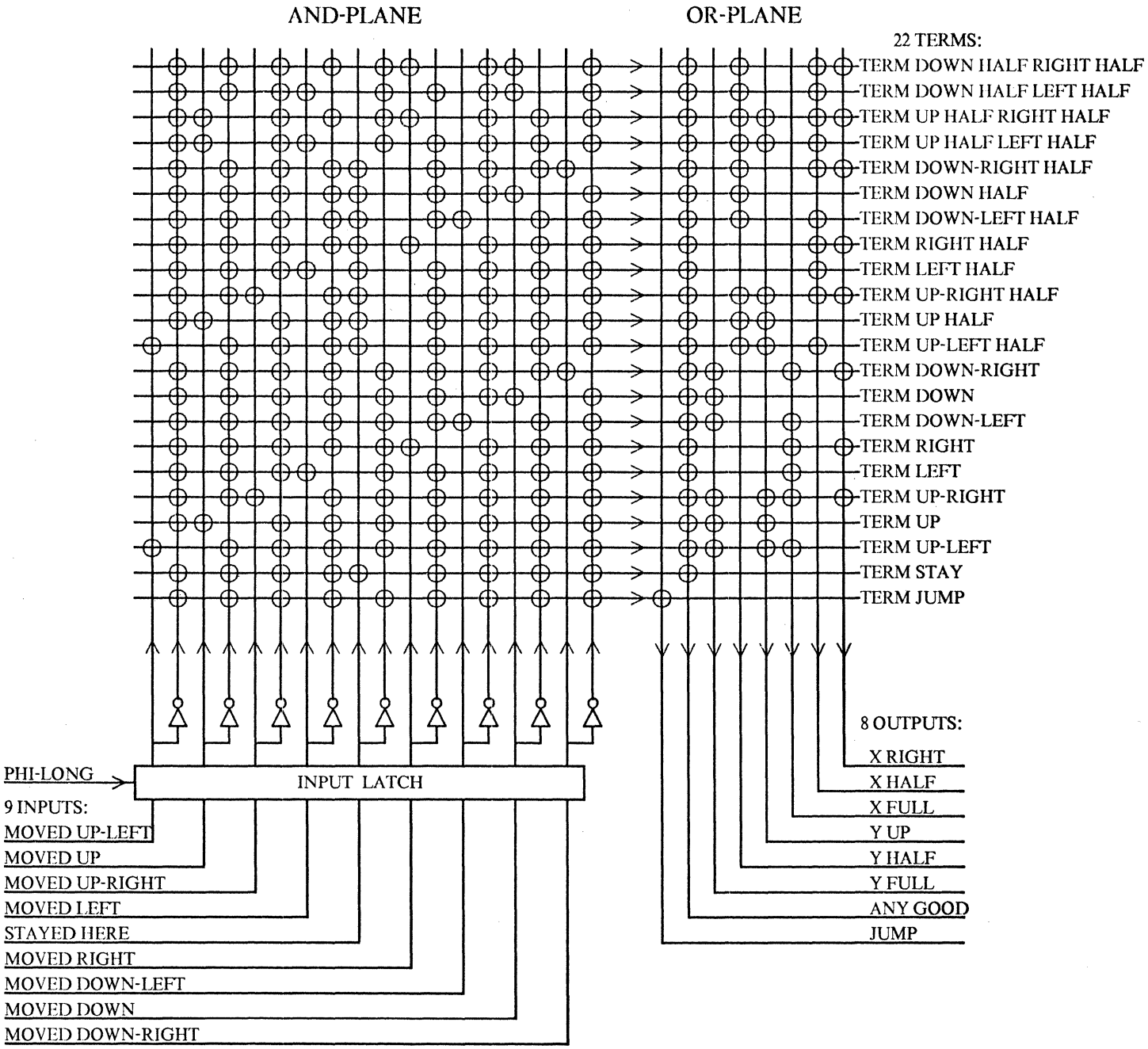
The optical mouse is just one simple example of an application of smart digital sensors, which happens to involve a few stages of logic to arrive at the answer in the desired format. Fortunately for this project, the NMOS technology that we know and love for logic is also well suited for sensing photons; so once the ideas and algorithms were firm, the chip design was relatively routine, and quick-turnaround implementation was available through the standard well-greased path.

The interrelated inhibition neighborhoods, contrasting patterns, sets of stable images, and tracking strategies for the optical mouse application have been thoroughly discussed in the text, and do not seem amenable to summarization here.

16. Concluding remarks

We have examined a family of smart digital sensors, specifically including motion-sensing imagers, which may find applications in places other than the mouse. Other applications of mutually-inhibiting and/or self-timed light detectors can be imagined, such as in character recognizers, edge detectors, light-controlled oscillators, etc. Other kinds of sensors can benefit from some of the same techniques.

A complete optical mouse has been in use for many months, with only one minor problem: when one is forced to use a workstation with an electro-mechanical mouse after becoming accustomed to the optical mouse, the erratic performance is an annoying contrast.



CIRCLES INDICATE GATE INPUTS TO AND-GATES (TERMS) AND OR-GATES (OUTPUTS).
LOGIC EQUATIONS CAN BE READ OFF BY INSPECTION.

EXAMPLE AND-PLANE EQUATION:

TERM DOWN HALF RIGHT HALF = (NOT MOVED UP-LEFT)
AND (NOT MOVED UP)
AND (NOT MOVED UP-RIGHT)
AND (NOT MOVED LEFT)
AND (NOT STAYED HERE)
AND (MOVED RIGHT)
AND (NOT MOVED DOWN-LEFT)
AND (MOVED DOWN)
AND (NOT MOVED DOWN-RIGHT).

EXAMPLE OR-PLANE EQUATION:

X RIGHT = (TERM DOWN HALF RIGHT HALF)
OR (TERM UP HALF RIGHT HALF)
OR (TERM DOWN-RIGHT HALF)
OR (TERM RIGHT HALF)
OR (TERM UP-RIGHT HALF)
OR (TERM DOWN-RIGHT)
OR (TERM RIGHT)
OR (TERM UP-RIGHT).

Appendix A.
Symbolic Representation of the Tracker Conflict Resolution PLA.

References

[Business 1981]
Business Week, "Will the boss go electronic, too?" pp 106-108, May 11, 1981.

[Card *et al.* 1977]
S. K. Card, W. K. English, and B. Burr "Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT", Xerox Palo Alto Research Center SSL-77-1, April, 1977.

[Conway *et al.* 1980]
L. A. Conway, A. G. Bell, and M. E. Newell, "MPC79: The Large-Scale Demonstration of a New Way to Create Systems in Silicon," *Lambda—The Magazine of VLSI Design.* pp. 10-19, Second Quarter, 1980.

[Englebart 1970]
D. C. Englebart, "X-Y position indicator for a display system", U. S. Patent 3,541,541. Nov. 17, 1970.

[Englebart & English 1968]
D. C. Englebart and W. K. English, "A Research Center for Augmenting Human Intellect", FJCC 1968, Thompson Books, Washington Books, Washington, D. C., p. 395.

[Englebart *et al.* 1967]
D. C. Englebart, W. K. English, and M. L. Berman, "Display-selection techniques for text manipulation", *IEEE Transactions on Human Factors*, HFE-8, 1, 5, 1967.

[Hawley *et al.* 1975]
J. S. Hawley, R. D. Bates, and C. P. Thacker, "Transducer for a display-oriented pointing device", U. S. Patent 3,892,963. July 1, 1975.

[Hon & Sequin 1980]
R. W. Hon and C. H. Sequin, *A Guide to LSI Implementation*, Xerox PARC Technical Report SSL-79-7, Palo Alto, California, 1980.

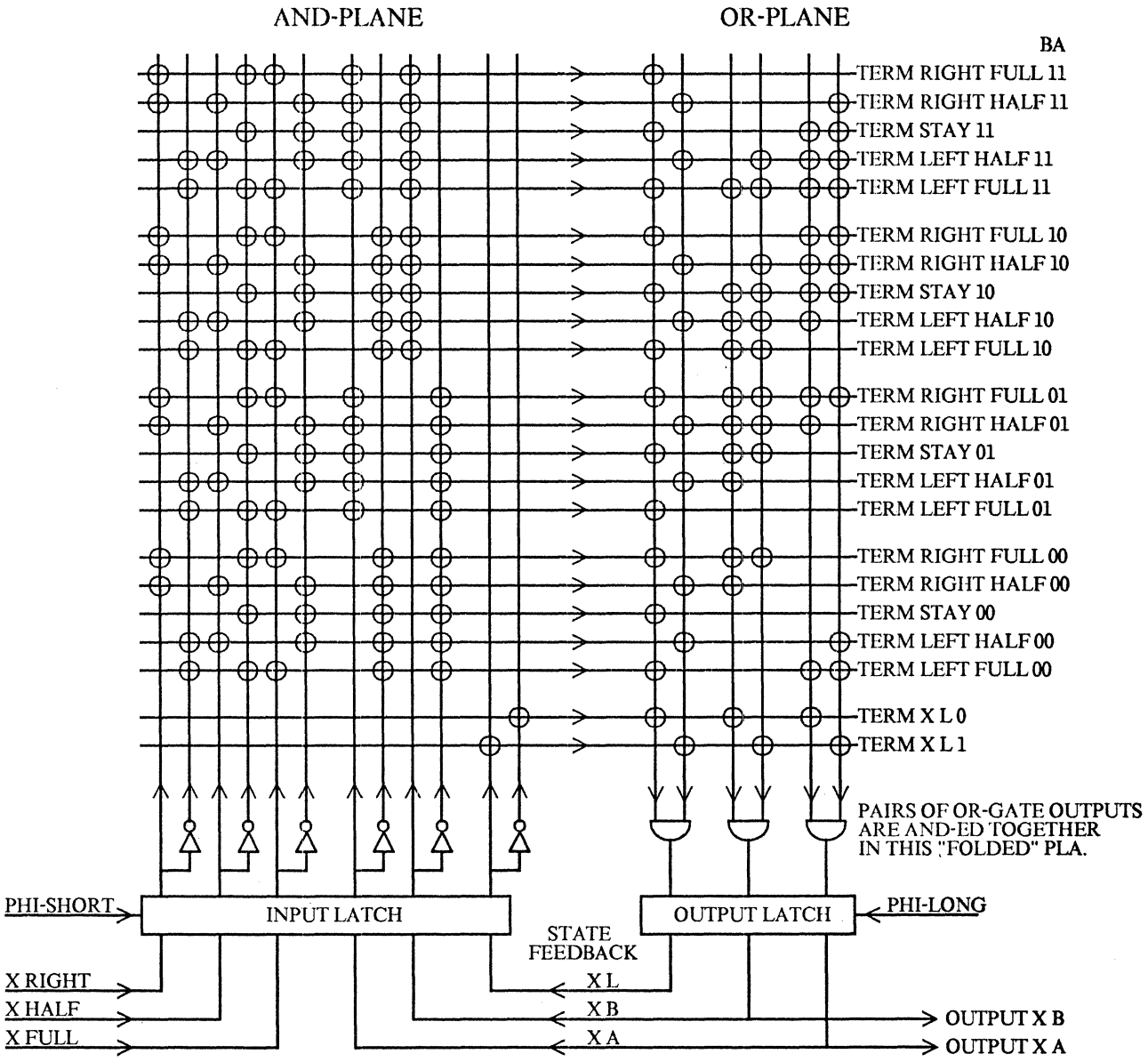
[Koster 1967]
R. A. Koster, "Position control system employing pulse producing means indicative of magnitude and direction of movement", U. S. Patent 3,304,434. Feb. 14, 1967.

[Lyon 1981]
R. F. Lyon, "A Bit-Serial VLSI Architectural Methodology for Signal Processing", *VLSI 81 Very Large Scale Integration*, (Conference Proceedings, Edinburgh, Scotland. John P. Gray, editor), Academic Press, August, 1981.

[Mead & Conway, 1980]
C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, Mass., 1980.

[Newman & Sproull 1973]
W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw Hill, 1973.

[Opocensky 1976]
W. J. Opocensky, "Cursor position device", U. S. Patent 3,987,685. Oct. 26, 1976.



CIRCLES INDICATE GATE INPUTS TO AND-GATES (TERMS) AND OR-GATES (OUTPUTS).
LOGIC EQUATIONS CAN BE READ OFF BY INSPECTION.

EXAMPLE AND-PLANE EQUATION:

TERM RIGHT FULL 11 = (X RIGHT)
AND (NOT X HALF)
AND (X FULL)
AND (X A)
AND (X B).

EXAMPLE OR-PLANE EQUATION:

XL = ((TERM X L 1)
OR (TERM RIGHT HALF XX)
OR (TERM LEFT HALF XX))
AND ((TERM X L 0)
OR (TERM RIGHT FULL XX)
OR (TERM STAY XX)
OR (TERM LEFT FULL XX)).
(OR-ING TERMS FOR ALL VALUES OF XX)

Appendix B.
Symbolic Representation of the Eight-State X Counter PLA.

[Rider 1974]
R. E. Rider, "Position indicator for a display system", U. S. Patent 3,835,464. Sept. 10, 1974.

[Seitz 1980]
C. L. Seitz, "Ideas about arbiters," *Lambda—The Magazine of VLSI Design.* pp. 10-14, First Quarter, 1980.

[Sequin & Tompsett 1975]
C. H. Sequin and M. F. Tompsett, *Charge Transfer Devices*, Academic Press Inc., New York, 1975.

[Seybold 1981-1]
The Seybold Report, "Xerox's 'Star'." Vol. 10, No. 16. April 27, 1981.

[Seybold 1981-2]
The Seybold Report on Word Processing, "The Xerox Star—A 'Professional' Workstation." Vol. 4, No. 5. May, 1981.

[von Bekesy 1967]
G. von Bekesy, *Sensory Inhibition*, Princeton University Press, 1967.

XEROX

XEROX

The Optical Mouse, and an Architectural Methodology for Smart Digital Sensors by Richard F. Lyon

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

VLSI-81-1