Optical Mouse
Functional Specification

Robert Garner

24 July 82

I. Introduction

The optical mouse (OM) system involves three components: the 16-pin nMOS chip, the special patterned surface (the mouse pad), and the optical system which resolves the dot array onto the chip's sensor array. This document contains the schematic diagrams for the chip, describes and specifies the external behavior, and develops the equations for the dimensions of the pad dots, sensors, and the optical system magnification. Other aspects of the optics (such as the required depth of field and light intensity) are not developed here. Figure 1 is a block diagram of the OM chip and figure 2 is its floorplan.

2. Theory of Operation

2.1 Photo Diode Operation

The optical mouse input transducer consists of a 4x4 array of reversed-biased p-n diodes. Each diode is a 170 micron square of n-type diffusion over the p-type substrate. The substrate is connected to ground while the diffusion is momentarily charged positive. After the dynamic node (i.e., sensor) is charged positive, it is then isolated by a turned off transistor. Figure 6, the overall schematic, and figure 7 show the sensors as parallel capacitors and diodes.

If a photon strikes the silicon semiconductor with an energy greater than its band gap energy (i.e., with a wavelength less than 1.1 micrometers--the infrared), the photon is absorbed and causes the generation of an electron-hole pair. If the electron-hole pair is generated within the junction's depletion region, the electron drifts toward the positively charged n-type dopants and the hole drifts toward the negatively charged p-type dopants of the substrate. Thus, the positive charge on the diffusion is reduced over time at a rate proportional to the photon flux. In other words, the output of the dynamic sensor node diminishes as light is received.

Quantities which effect a sensor's response are the depth below the surface of the depletion (transition) region, the width of the depletion region (i.e., relative impurity concentration) and the spectral contect of the light. The sensors are most sensitive if the light frequency is near the band gap energy (infrared and visible) and the depletion region is wide and near the surface. Note that the optical mouse is based on an architectural timing scheme that is insensitive to changes in these parameters.

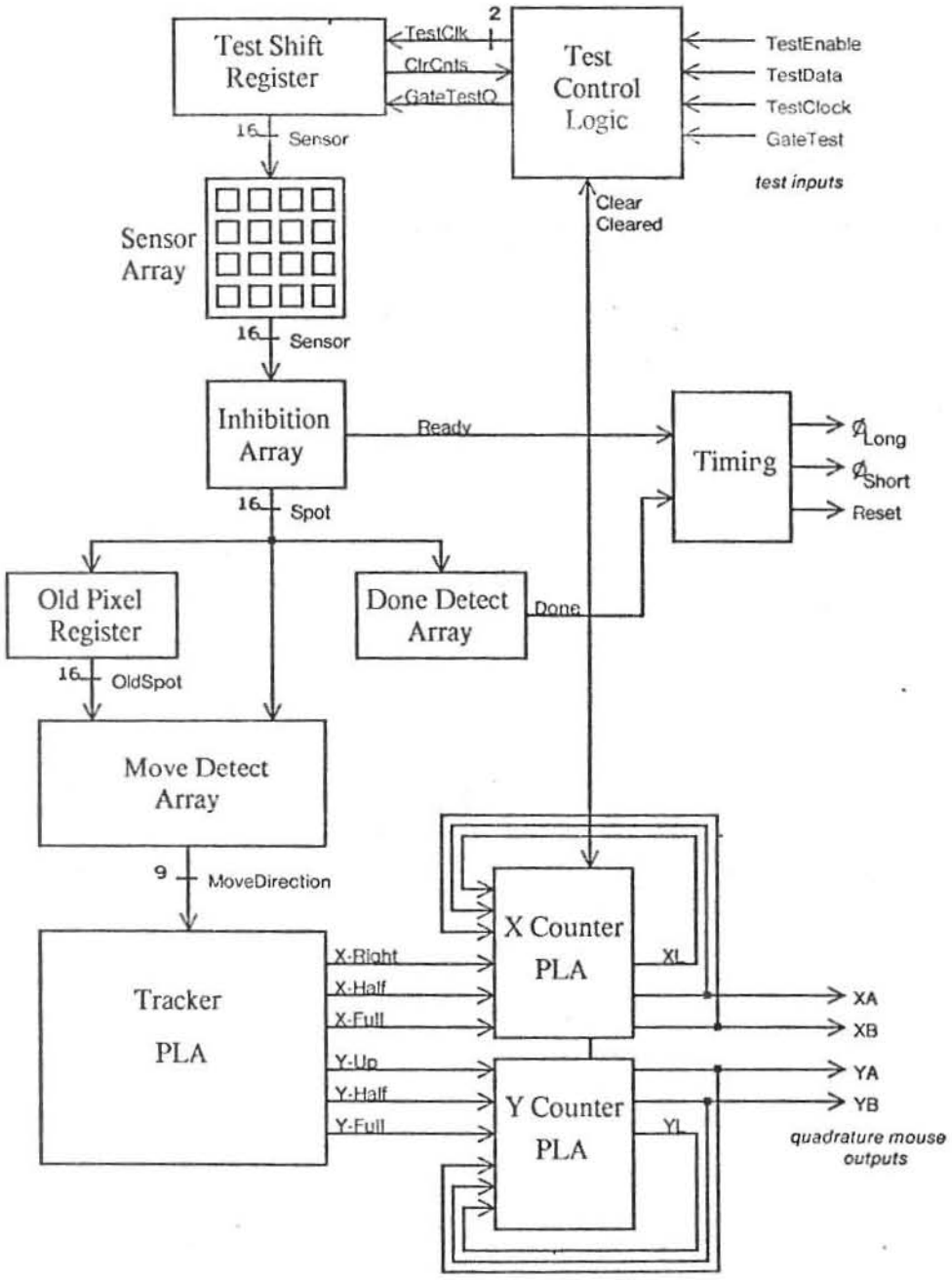


Figure 1. Optical Mouse Block Diagram

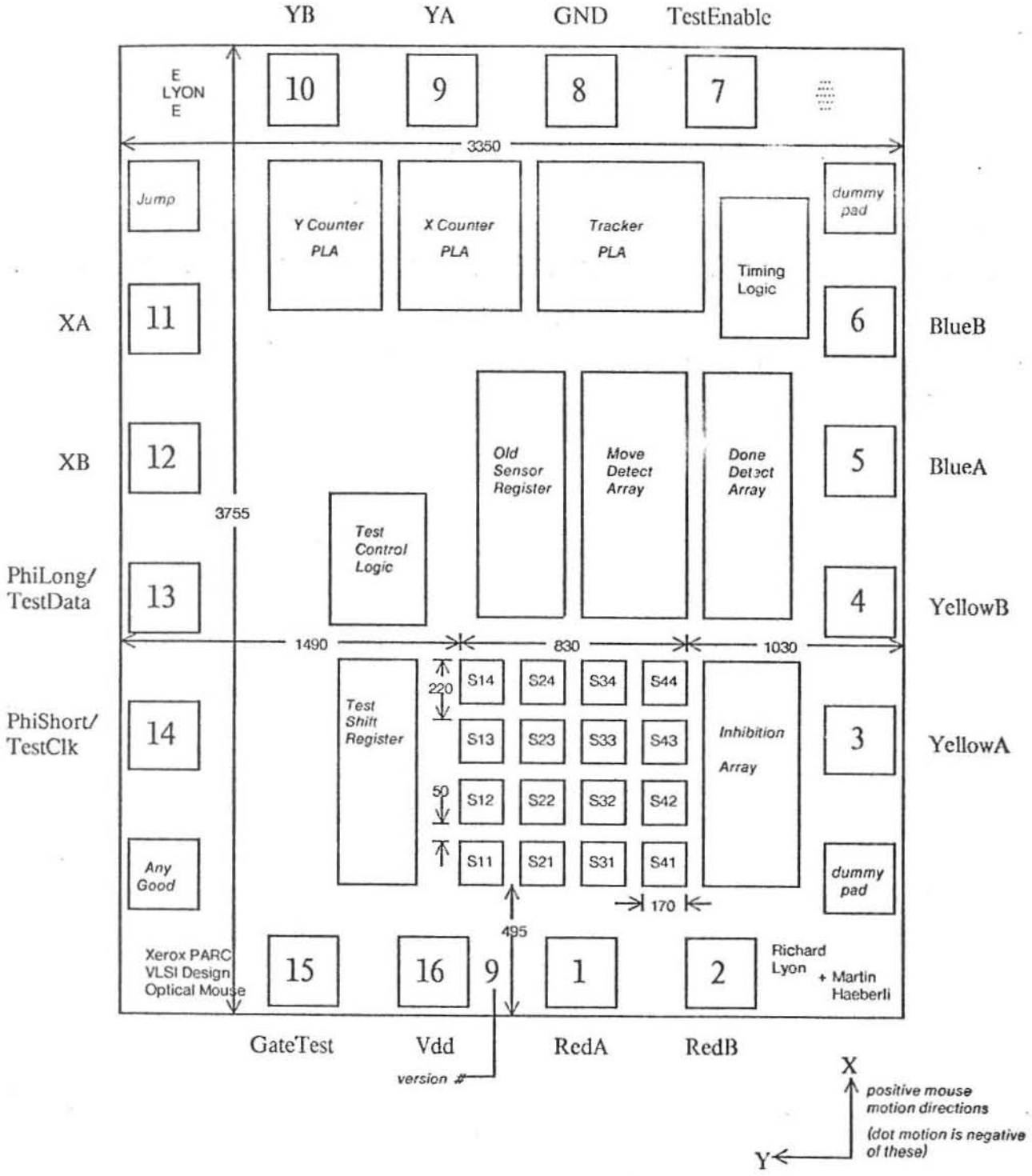


Figure 2. FloorPlan, Pin Numbers, Directions, & Dimensions

Notes: (1) All dimensions in microns. The external size does not include distance to scribe marks, typically x microns away.

(2) Substrate should be conected to ground (pin 8).

2.2 Inhibition and Time Periods

Time is partitioned into into two epochs: a light-gathering, or "watching" phase and a recording phase. During the gathering phase the sensors are pre-charged to Vdd; thereafter, they discharge at rates dependent upon the intensity at each sensor node. Since the all-charged and all-discharged states are not interesting, a snap-shot of the image is obtained by cross-coupling the sensors in a "lateral-inhibition" manor. When the voltage on a particular sensor drops below a gate threshold, all sensors in its inhibition neighborhood are ignored. Because the inhibition pattern is large, only a few sensor nodes need to discharge before the snapshot is complete. The wired-in inhibition patern about each sensor (see figure 3) implies 30 stable 4x4 images which can exist. These are shown in figure 4.

The output of a sensor node, as inhibited by its neighbors and inverted, is called PixelLight (see the schematic). Thus, if a sensor node discharges before any of the sensors which inhibit it discharge, its PixelLight equals 1. An inhibited or still-charged sensor has PixelLight = 0. Figure 7 is a transistor-level schematic of the Pixel and Inhibition arrays

During the light-gathering phase, eventually either PixelLight = 1 in one of the middle four sensors of the the 4x4 array, or PixelLight = 1 for two sensors around the edge of the array (i.e., a stable patterns appears) and the signal Done becomes true and the chip enters the recording phase.

On the chip, the Done signal is actually computed with a great deal of redundancy: Done is set to true only when, for each sensor, PixelLight = 1, or one of its inhibitor's has PixelLight = 1. Note that a buffered version of PixelLight, called Spot, is used for detecting Done.

2.3 Determining Motion

During the recording phase the stable image output of the watching phase is compared with the resultant image of the previous watching phase, present in the "old spot register" (see schematic). If there is implied motion, the X-direction and/or Y-direction counters are updated. The output of these counters directly drives the four mouse quadrature outputs connected to the host machine.

To determine if there is motion, for each image position of the 4x4 array, the "motion detect array" decides whether Spot = 1 and OldSpot = 1 for any of the eight, conterminous sensors of the previous image. Thus, computed for each sensor, is the direction an active spot may have moved: up and left, or right, not-at-all, etc.. The results for all sensors are or'd together, and consequently, of the move detect array's nine outputs (see schematic), either one, or two, or none (in the case of jumps) of them can be true. Figure 10 shows the schematic of the move detect array and figure 6 shows the implied motion resulting from pairs of "old" and "new" images.

The nine outputs of the move detect array then feed the "tracker PLA" which sorts them into signals which directly control the X and Y counter PLAs. For each counter, there are three outputs: right (or up), half, or full. full = 1 for motions of at least one sensor center-to-center distance and half = 1 for diagonal distances less than this (see figure 5). right = 1 says the counter should increment, right = 0 implies is should be decremented. The low-order bit of the X and Y counters (the half-step bits, XL' and YL') have no external connection (although they should). The two upper bits of the PLAs follow the Gray code, while the low bit is toggled: 000, 001, 010, 011, 110, 111, 100, and 101.

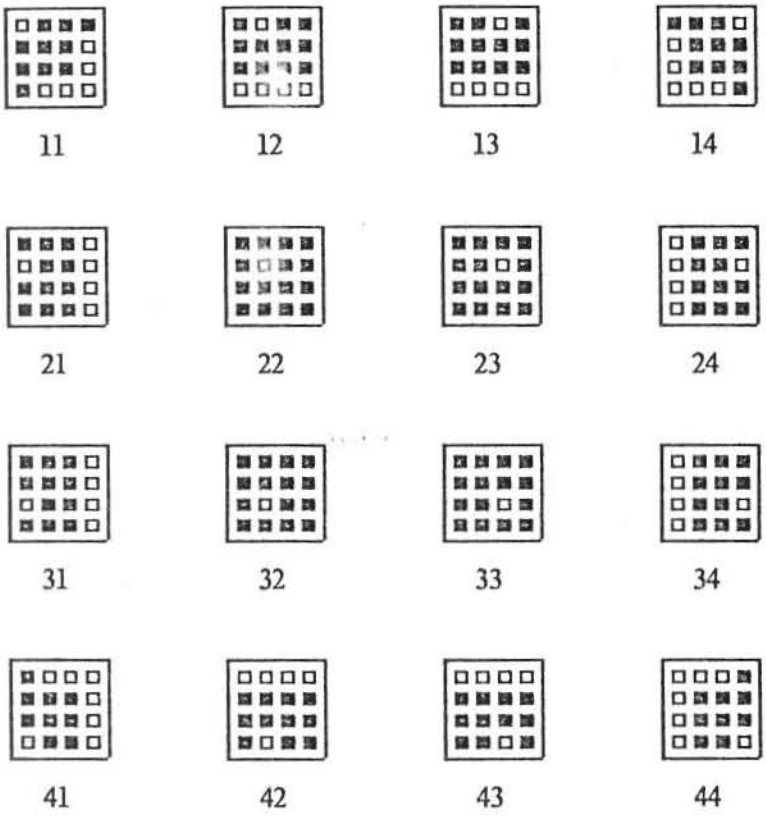


Figure 3. Inhibition Patterns for Each Sensor

Open squares represent nodes which are not inhibited.

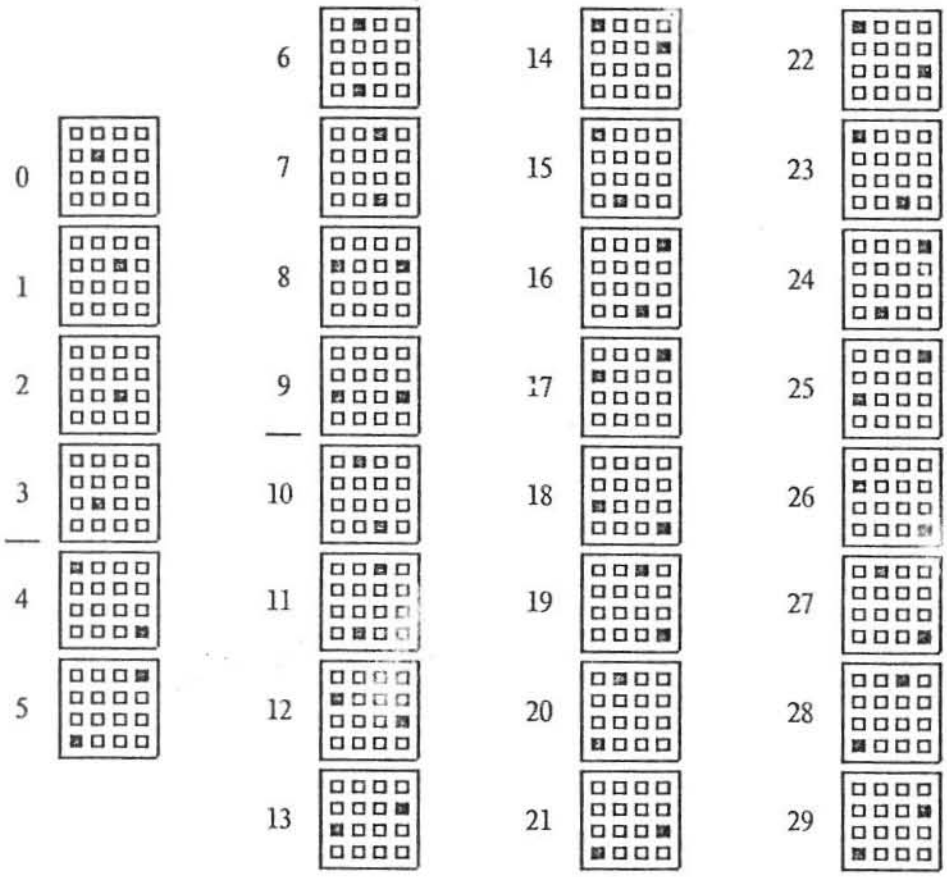


Figure 4. All Possible Stable Image Patterns

Old Image	NewImage	True Move Detect Array Outputs	Interpretation	Effect on XY Counters
0	1	Right	Straight Full-Step	X ← X + 1
2	12	Right	Straight Full-Step	X ← X + 1
10	7	Right, Stayed	Straight Half-Step	X ← X + 1/2
3	1	Up-Right	Diagonal Full-Step	X ← X + 1 Y ← Y + 1
2	29	Up-Right	Diagonal Full-Step	X ← X + 1 Y ← Y + 1
8	10	Up-Right	Diagonal Full-Step	X ← X + 1 Y ← Y + 1
29	24	Right, Up	Diagonal Half Step	X ← X + 1/2 Y ← Y + 1/2
10	12	Down-Left Up-Right	Opposing	X ← X Y ← Y
10	22	Left Up-Right	Nearly Opposing	X ← X Y ← Y
0	0	Stayed	No Motion	X ← X Y ← Y
29	29	Stayed	No Motion	X ← X Y ← Y
0	29	*none*	Jumped	X ← X Y ← Y
10	5	*none*	Jumped	X ← X Y ← Y

Figure 5. Typical Image Pairs and Their Effect on the XY Counters

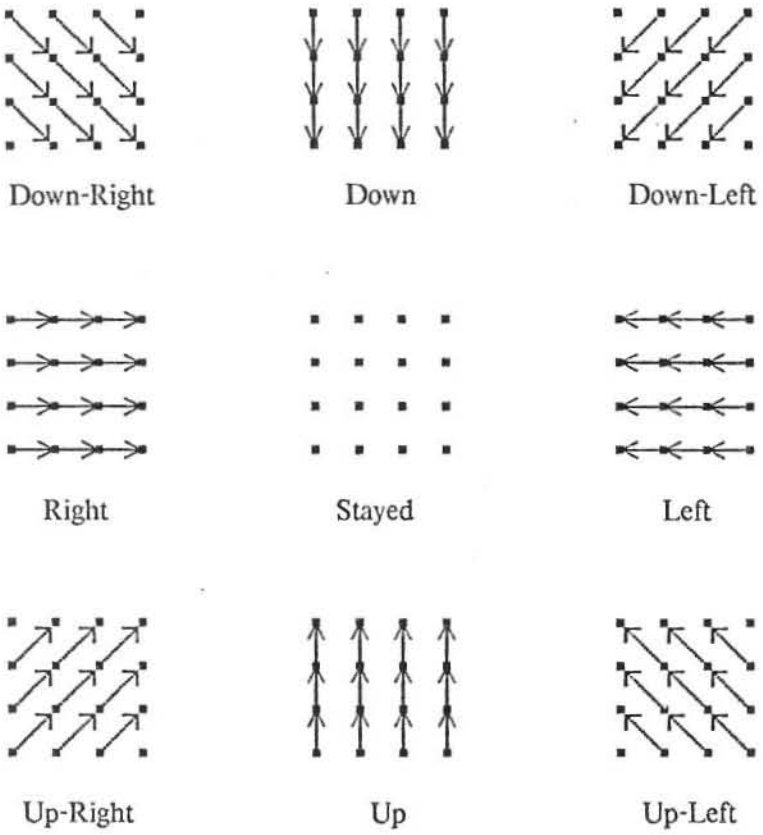


Figure 6. Dot Equations for Move Detect Array

OldSensor is tail of vectors and current Sensor value is head of vectors

2.4 Quadrature Outputs

The upper two bits of the X and Y counters are the quadrature outputs of the mouse: XA, XB, YA, and YB. Mouse motion is with respect to the chip itself: if the mouse is moving right (i.e., the imaged pad dots are moving left across the sensors), then positive transitions on the XB output will occur before positive transitions on the XA output. Likewise, if the mouse is moving down (i.e., the imaged pad dots are moving up across the sensors), then positive transitions on the YA output will occur before positive transitions on the YB output. Up (positive Y) and right (positive X) directions, with respect to the sensor array, are shown in figure 3. (Note that, regardless of optical image reversals or chip orientations, the directions implied by the quadrature outputs can be made correct by interchanging outputs XA with XB or YA with YB.) The distance between positive (or negative) transitions on XA and XB corresponds, at minimum, to 1/200 of a inch.

2.5 Self-Timed Operation and Clocks

The optical mouse chip is self timed: the watching and recording phases cycle at a rate dependent upon the light flux. There are two internal clocks: phiLong and phiShort. phiLong is generally high during the watching phase, while phiShort is high during the recording phase. phiLong is used to recirculate data in the pseudo-static registers; implemented this way (see schematic) because the whole chip is being bombared by light which would otherwise cause ALL the dynamic nodes to discharge. phiShort gates data onto to the storage nodes (the old spot register and the X and Y counter PLAs) which record the results of the last watching phase.

The cycling of the self-timed, light-controlled logic starts with the sensor array being charged by the signal Reset during the recording phase (phiShort high). See figure 9. After all of the sensors are high, Ready goes high, causing phiShort to drop, which then causes phiLong to rise, thereby removing the Reset pulse. After some light is received, Ready drops and then Done rises (i.e., a stable image appears). This triggers Stop (a timing logic state bit), consequently causing phiLong to drop, and phiShort to rise, thereby turning off Stop. The zeroing of Stop causes Reset to become true, and, fortunately we are back to the beginning of the cycle.

As the mouse is self-timed, the light must be intense enough to enable the outputs to faithfully follow a fast mouse. (Suppose the mouse can be moved at the lightning-fast rate of 70 inches/second. At the nominal count rate of 200 counts/inch, this implies that the clock frequency must be at least 14 kHz (70 microseconds). A reasonable goal is a clock rate of 5 kHz, which would ideally correspond to a tracking rate of 25 inches/sec. The degradation of LEDs and incandescent lights over the life time of the mouse should be considered.

Note that the duration of the low part of phiLong can not be greater than the time it takes light to discharge the dynamic nodes controlled by either phiLong or phiShort. The phiLong-low, "danger" period equals about 9 gate delays (see figure 9) plus the time required to charge the sensor array to Vdd. Since the duration of phiLong = 1 approximately represents the time required to discharge a sensor node, which is a dynamic node, the light should not be so intense as to cause the duty cycle of phiLong to approach 50%. (phiLong's low period was measured once at 1.1 microseconds.)

2.6 Mouse Pad

This brings us to the subject of the mouse pad. The pattern chosen is a hexagonal pattern of white dots on a black background. Ideally, the white dots are circles, but other shapes, such as hexagons, are fine. This pattern was chosen because, for all possible mouse orientations, the disparity between the minimum and maximum distance between dots is less for a hexagonal array than for a square array (.866 versus .707, or 18%). Furthermore, a hexagonal array has three axes of symmetry instead of the square's two. The required pad dot size and dot separation are derived in section 5.

The pad should be printed with an ink (such as a carbon-based ink) which will absorb light near the infra-red end of the spectrum. On the other hand, the (white) paper should be a good infra-red reflector. The contrast ratio should be as high as possible in order to reduce the effects of dirt and smudging.

3. Pin Descriptions

Pin	Name	Function
1	Red Debouncer A.	See figure 7 for schematic of debouncer pad pairs
2	Red Debouncer B.	"
3	Yellow Debouncer A.	"
4	Yellow Debouncer B.	"
5	Blue Debouncer A.	"
6	Blue Debouncer B.	"
7	TestEnable.	When LOW, disables internal test circuitry and causes pins 13 & 14 to output PhiLong & PhiShort. When HIGH, pins 13 and 14 become test inputs (see below).
8	Ground	
9	YA	Quadrature output for Y direction.
10	YB	"
11	XA	Quadrature output for X direction.
12	XB	"
13	PhiLong/TestData	As an output (when pin7 = HIGH), equals internal PhiLong clock (HIGH while gathering light, see fig. 5). As an input (when pin7 = LOW), supplies data to Test Shift Register (see figure 6). A HIGH value in the Shift Register causes sensor node to be discharged. The input value should remain stable for at least 1μS after the HIGH-to-LOW edge of TestClock.
14	PhiShort/TestClock	As an output (when pin7 = HIGH), equals internal PhiShort clock (HIGH when cycling data registers, see fig. 5). As an input (when pin7 = LOW), supplies clock to Test Shift Register (see figure 6). While the clock is HIGH, the register is shifted and the TestData pin value is entered into the first position (see fig. 6).
15	GateTest	Ignored when TestEnable = LOW. When TestEnable = HIGH, a HIGH level on GateTest applies the pattern in the Test Shift Register to the sensor array. (see fig. 6).
16	VDD	+5 VDC

4. Logic Equations

4.1 Inhibition & Done Detect Arrays

Figure 3 is an encoding of both the inhibition and done detect equations. In the case of the inhibition logic, each array of figure 3 represents a NOR gate with PixelLight*xy* as its output. A solid black box indicates an input term to the NOR gate, equal to the row-column coordinates of the box. An isolated white box indicates an input to the NOR gate which is the Sensor node itself. For example, the equation for PixelLight21 follows. Ready is the NOR of all PixelLight's.

PixelLight21 = *not*(Sensor21 *or* PixelLight11 *or* PixelLight31 *or* PixelLight41 *or* PixelLight12 *or* PixelLight22 *or* PixelLight32 *or* PixelLight42 *or* PixelLight13 *or* PixelLight23 *or* PixelLight33 *or* PixelLight43).

In the case of the done detect logic, each array of figure 3 represents a NOR gate with Group*xy*Done' as its output. Group*xy*Done' indicates that one or more of the PixelLights which define the inhibition neighborhood for PixelLight*xy* is true. A solid box and the single isolated white box indicate the input terms to the NOR gate, equal to Spot*xy*. (Recall that Spot*xy* is a buffered version of PixelLight*xy*.) For example, the equation for Group21Done follows. Done is just the NOR of all GroupDone's.

Group21Done = *not*(Spot11 *or* Spot21 *or* Spot31 *or* Spot41 *or* Spot12 *or* Spot22 *or* Spot32 *or* Spot42 *or* Spot13 *or* Spot23 *or* Spot33 *or* Spot43).

4.2 Move Detect Array

For each sensor, the move detect array computes, given the conterminous sensor values of the previous cycle, which direction the projected spot moved. Figure 10 is a schematic of the move detect array.

Figure 6 is an encoding of the move detect logic equations: Each direction, such as DownRight', is the output of a NOR gate with nine AND gates as inputs. The two inputs of each AND gate are given by head and tail of each line of figure 6 For example, the equation for DownRight' is:

DownRight' = *not*((Spot22 *and* OldSpot11) *or* (Spot23 *and* OldSpot12) *or* (Spot24 *and* OldSpot13) *or* (Spot32 *and* OldSpot21) *or* (Spot33 *and* OldSpot22) *or* (Spot34 *and* OldSpot23) *or* (Spot42 *and* OldSpot31) *or* (Spot43 *and* OldSpot32) *or* (Spot44 *and* OldSpot33).

4.3 Timing Logic

The schematic for the timing logic is shown in figure 9.

4.4 Tracker PLA

The schematic for the Tracker PLA is shown symbolically in figure 13. Circles represent inputs of minterms and minterms of outputs. For example, the equation for YUp is

YUp = MovedUpLeft *and not*(MovedUp) *and not*(MovedUpRight) *and not*(MovedLeft) *and not*(Stayed) *and not*(MovedRight) *and not*(MovedDownLeft) *and not*(MovedDown) *and not*(MovedDownRight))
or (*not*(MovedUpLeft) *and not*(MovedUp) *and not*(MovedUpRight) *and* MovedLeft) *and not*(Stayed) *and not*(MovedRight) *and not*(MovedDownLeft) *and not*(MovedDown) *and not*(MovedDownRight))
or (*not*(MovedUpLeft) *and not*(MovedUp) *and not*(MovedUpRight) *and not*(MovedLeft) *and not*(Stayed) *and not*(MovedRight) *and* MovedDownLeft *and not*(MovedDown) *and not*(MovedDownRight))
or (MovedUpLeft *and not*(MovedUp) *and not*(MovedUpRight) *and not*(MovedLeft) *and* Stayed *and not*(MovedRight) *and not*(MovedDownLeft) *and not*(MovedDown) *and not*(MovedDownRight))
or (*not*(MovedUpLeft) *and not*(MovedUp) *and not*(MovedUpRight) *and* MovedLeft) *and* Stayed) *and not*(MovedRight) *and not*(MovedDownLeft) *and not*(MovedDown) *and not*(MovedDownRight))
or (*not*(MovedUpLeft) *and not*(MovedUp) *and not*(MovedUpRight) *and not*(MovedLeft) *and* Stayed) *and not*(MovedRight) *and* MovedDownLeft) *and not*(MovedDown) *and not*(MovedDownRight))
or (*not*(MovedUpLeft) *and* MovedUp *and not*(MovedUpRight) *and* MovedLeft) *and not*(Stayed) *and not*(MovedRight) *and not*(MovedDownLeft) *and not*(MovedDown) *and not*(MovedDownRight))
or (*not*(MovedUpLeft) *and not*(MovedUp) *and not*(MovedUpRight) *and* MovedLeft *and not*(Stayed) *and not*(MovedRight) *and not*(MovedDownLeft) *and* MovedDown *and not*(MovedDownRight))

4.5 X and Y Counter PLAs

The schematics for the X and Y Counter PLAs are shown symbolically in figure 14. Circles represent inputs of minterms and minterms of outputs. For example, the equation for XA is

XA = ((-XL *and* -Clear) *or* (-Up *and* -Half *and* Full *and* -XA *and* -XB *and* -Clear) *or*
(Up *and* Half *and* -Full *and* -XA *and* XB *and* -Clear) *or* (Up *and* -Half *and* Full *and* -XA *and* XB *and* -Clear) *or*
(-Up *and* Half *and* -Full *and* XA *and* XB *and* -Clear) *or* (-Half *and* -Full *and* XA *and* XB *and* -Clear) *or*
(Up *and* Half *and* -Full *and* XA *and* XB *and* -Clear) *or* (Up *and* -Half *and* Full *and* XA *and* XB *and* -Clear) *or*
(-Up *and* -Half *and* Full *and* XA *and* -XB *and* -Clear) *or* (-Up *and* Half *and* -Full *and* XA *and* -XB *and* -Clear)
or (-Half *and* -Full *and* XA *and* -XB *and* -Clear))
and ((XL *and* -Clear) *or* (-Up *and* -Half *and* Full *and* -XA *and* -XB *and* -Clear) *or*
(-Up *and* Half *and* -Full *and* -XA *and* -XB *and* -Clear) *or* (Up *and* -Half *and* Full *and* -XA *and* XB *and* -Clear) *or*
(-Half *and* -Full *and* XA *and* XB *and* -Clear) *or* (Up *and* Half *and* -Full *and* XA *and* XB *and* -Clear) *or*
(Up *and* -Half *and* Full *and* XA *and* XB *and* -Clear) *or* (-Up *and* -Half *and* Full *and* XA *and* -XB *and* -Clear) *or*
(-Up *and* Half *and* -Full *and* XA *and* -XB *and* -Clear) *or* (-Half *and* -Full *and* XA *and* -XB *and* -Clear) *or*
(Up *and* Half *and* -Full *and* XA *and* -XB *and* -Clear))

5. Pad and Sensor Dimensions

As shown in figure 16, the mouse pad is a white paper tessellated with black dots whose centers are at the vertices of adjoining equilateral triangles. There are two quantities which specify this hexagonal pattern pad: the distance between dots D and the dot diameter dia. Currently, D = .0178" (445 μm) and dia = .008" $\pm$ .001" (203 $\pm$ 27 μm). The dots can be isomorphic to shapes in the range from hexagons to circles.

The sensor array, on the other hand, is a square array of square sensors, specified by the sensor center-to-center distance S. Currently, S = 220 μm (.00865").

5.1 Min, Avg, and Max Dot Separation

The raw dot count rate varies depending on one's direction of travel across the the equilateral triangles. As shown in figure 16, the dot rate is maximum if one moves parallel to the dot center-to-center line. The rate is minimum if one moves 30° to this axis.

q_{max} is just the side of a unit equilateral triangle:

$$q_{max} = 1.00 .$$

q_{min} is the height of a unit equilateral triangle:

$$q_{min} = \cos(\pi/6) = \text{sqrt}(3)/2 = .866 .$$

q_{avg} is the average between the triangle's side and height:

$$\begin{aligned} q_{avg} &= q_{min}/(\pi/6 - 0) * (\text{integral from 0 to } \pi/6 \text{ of sec } \alpha \, d\alpha) \\ &= (3*\text{sqrt}(3)*d/\pi) * (\text{integral from 0 to } \pi/6 \text{ of sec } \alpha \, d\alpha) \\ &= (3*\text{sqrt}(3)*d/\pi) * \ln(\sec \pi/6 + \tan \pi/6) \\ &= 3*\text{sqrt}(3)\ln(3)/2\pi \\ &= .909 . \end{aligned}$$

5.2 Min, Avg, and Max Stable Pattern Sensor Separation

Only a fixed number of all possible sensor pairs participate in the dot imaging processes: of the 30 stable patterns (fig 4), 26 of them involve a pair of sensors. The seperation between these sensors, in units of sensor size (see figure 15) is called a.

The minimum seperation occurs for patterns like 8 and 9:

$$a_{min} = 3.00 .$$

Since, in practice, the mouse can not exactly maintain an orientation of 0°, a practical minimum seperation is an average between patterns 6, 11, and 15:

$$a_{pmin} = a_6/3 + a_{11}/3 + a_{15}/3$$
$$= 3.00/3 + 2*3.16/3 = 3.10 .$$

The maximum seperation occurs for patterns 4 and 5:

$$a_{max} = \text{sqrt}(3^2 + 3^2) = 4.24 .$$

Since, in practice, the mouse can not exactly maintain an orientation of 45°, a practical maximum seperation is an average between patterns 4, 22, and 23:

$$a_{pmax} = a_4/3 + a_{22}/3 + a_{23}/3$$
$$= 4.24/3 + 2*3.61/3 = 3.93 .$$

Finally, the average seperation is

$$a_{avg} = 1/26 (2a_{4,5} + 4a_{6\text{-}9} + 4a_{10\text{-}13} + 8a_{14\text{-}21} + 8a_{22\text{-}29})$$
$$= 1/26 (2*\text{sqrt}(18) + 4*3 + 4*\text{sqrt}(10) + 8*\text{sqrt}(10) + 8*\text{sqrt}(13))$$
$$= 3.36 .$$

5.3 Determination of Magnification, Sensor Separation, & Dot Separation

The lens magnifies dots separated by the distance qD on the object plane and (ideally) projects them onto the sensor array of the image plane with a stable pattern seperation of aS. Thus,

$$m = i/o = (aS)/(qD) \ . \qquad (1)$$

Likewise, when a dot on the pad (object plane) moves a distance of 1/c, where c is the count rate, its image moves m/c. During this time, the mouse will increment the X and/or Y counter by one.

Furthermore, when an imaged dot moves the stable-pattern distance aS, the X and/or Y counter must increment by three. (Look at the stable patterns of figure 4: the dot pairs are separated by 3 in the X and/or Y directions.) Thus, the following proportion is true:

$$\frac{m/c}{1 \text{ count}} = \frac{aS}{3 \text{ counts}}$$

or,

$$m = (caS)/3 \ , \qquad (2)$$

or,

$$c = (3m)/(aS) \ , \qquad (3)$$

or,

$$S = (3m)/(ac) \ . \qquad (4)$$

Equating equations (1) and (2) results in

$$D = 3/qc \ . \qquad (5)$$

Equation (5) implies that the pad dot seperation is only a function of the desired count rate and a choice of the count rate scaling factor (which is a function of orientation). The bottom line is that 3/qD dots must pass before the sensor array in order to achieve a count rate of c. Choosing a value for q ties the count rate to either the min, max or avg dot rate directions. However, as demonstrated in the next section, a choice of q is not the controlling factor of mouse count rate variation.

According to eq. (2), m and S are directly proportional: consequently, a choice of the sensor size fixes the lens magnification, and vice versa. For example, suppose m is decreased, but S is not, then, according to eq. (3), the count rate will go down. However, according to eq (5), the dot spacing must be increased or else the mouse will track even more slowly.

If the desired count rate is 200 counts/inch and we assume the average value of q (the mouse visits all angles between 0 and 30° with equal probability) and the lens magnification matches the sensor seperation, then the pad dot seperation should be:

$$D_{c200} = 3/(.909*200) = .0165" \ (420\ \mu m).$$

5.4. Determination of S and m.

Once the constants a and q are chosen, the seperation between array sensors then depends on the magnification, or, antithetically, the magnification depends on the sensor seperation.

If we choose the average value of a (i.e., the stable patterns will occur with equal probability), then equations (2) and (4) can be used to design a mouse which will, on the average, count at 200 steps/inch:

$$S_{c200} = (3m)/(200a_{avg})$$
$$= (3m)/(200*3.36)$$
$$= .00446m \text{ (inches)}$$

or

$$m_{c200} = 224S \ .$$

Given a magnification of 1.8, the required sensor seperation is

$$S_{c200} = .00446(1.8) = .00803" = 200\,\mu m \ .$$

Likewise, given a magnification of 1.938, the required sensor seperation is

$$S_{c200} = .00446(1.938) = .00864" = 220\,\mu m \ .$$

5.5 Variation in Counting Rates

The counting rate varies as a function of orientation. Two factors control this variation: the number of dots per inch as a function of angle (q) and the average value of the stable pattern dot seperation *actually* occurring for a particular orientation (a). It is the second factor which has a larger impact on the count rate than the pad itself.

The total possible count rate variation due to q is 13.4% between the min and max extremes and 4.7% between min and avg.

For the count rate variation due to a, there are two cases. When $a > a_{avg}$, the count rate increases since more than one stable pattern is possible before the mouse is moved the distance of 1/qc. In the worst case, an extra half step of motion can be added for about every three counts (for example, between pat(4) and pat(27)). In practically, it is probably more like an extra half step for every six counts, for a total count rate increase of 8.3%.

When $a < a_{avg}$, the count rate decreases as the mouse must be moved slightly further to evoke count steps. This factor equals the difference between the actual a and a_{avg} (the value the mouse was designed to). Using the practical minimum, we get a factor of $a_{pmin}-a_{avg}/a_{avg}$, or about 7.7%.

The lowest counting rate occurs when the short side (0°) of the sensor array travels parallel to the dot-to-dot center line (0°). This rate differs from the average by 7.7% (as explained in previous paragraph).

Interestingly enough, the highest rate does not occur when the short side of the sensor array travels at 30° to the dot-to-dot center line. Even though there are more dots/inch in this direction (lower q), the count rate is reduced because $a < a_{avg}$. Thus, what would otherwise be a 9.1% larger than average rate, is actually reduced by 7.7% to be approximately the average count rate (only 1.4% greater than average).

The highest count rate occurs when the sensor array is moved at a 45° angle with respect to the dot-to-dot center line (0°). In this case $a > a_{avg}$, and the count rate is 8.3% larger than average, as explained above.

In summary, the lowest rate is 7.7% below average, and the highest is 8.3% above, for a total variation of 16% between min and max. If one wanted to design the mouse so that the 200 count/inch track rate equaled the lowest count rate, the average count rate would be 200+.077*200 = 215 counts/inch and the dot seperation should be

$$D_{c215} = 3/(.909*215) = .0153" \quad (383\ \mu m).$$

m or S can then be set to any value desired. If S is left at 220 μm, then the new required value of m is

$$m_{c215} = (215*3.36*.00864)/3$$
$$= 2.08 \ .$$

5.6 Historical Notes

I. The current mouse was originally designed assuming that the desired rate of 200 counts/inch represented the average counting rate. MH suggested that m be changed from 1.938 to 1.8 instead of to 2.08.

Since, in versions 7 and 9, S was not correspondingly changed to 200 μm, a magnification of 1.8 would give a theoretical average counting rate of (eq. 2):

$$c_{avg} = (3m_{cavg})/(a_{avg}S_{cavg})$$
$$= (3*1.8)/(3.36*.00865)$$
$$= 185 \text{ counts/inch} \ ,$$

and a minimum rate of 171 (7.7% less).

If S is not changed, the dot spacing should be increased from .0165" to .0178":

$$D_{c200} = 3/(q_{avg}c_{avg})$$
$$= 3/(.909*185)$$
$$= .0178" \quad (445\ \mu m)$$

II. The original PARC-prototype mouse (versions 1 and 3) was designed with a sensor seperation of 320 μm (.0126"). Thus, the intended magnification was (eq. (4)):

$$m_{c200} = 224S = 224*.0126 = 2.82.$$

However, the magnification was later determined to be about 1.9, so the pad spacing had to be adjusted to insure proper operation. From eq(1), the seperation was changed to:

$$D_{c135} = (a_{avg}S)/(q_{avg}m_{cavg})$$
$$= (3.36*.0126)/(.909*1.9)$$
$$= .0245"$$

Thus, the theoretical average count rate became (from eq. 5):

$$c_{avg} = 3/(q_{avg}D) = 135 \text{ counts/inch} .$$

6.7 Determination of Dot Diameter

The image of the pattern pad dot should be large enough such that its smallest dimension covers the diagonal of the sensor. Conservatively assuming that the sensor width equals the sensor center-to-center distance, the minimum dot diameter is

$$dia_{min} = s*sqrt(2)/m = .0122/m .$$

The maximum diameter should be no more than about twice this value.

Assuming a magnification of 1.938,

$$dia_{min} = .0122/1.94 = .0063" ,$$

and a magnification of 1.8 yields

$$dia_{min} = .0122/1.8 = .0068" .$$

Rounding up, the spec becomes .008" ± .001" (203 ± 27 μm).

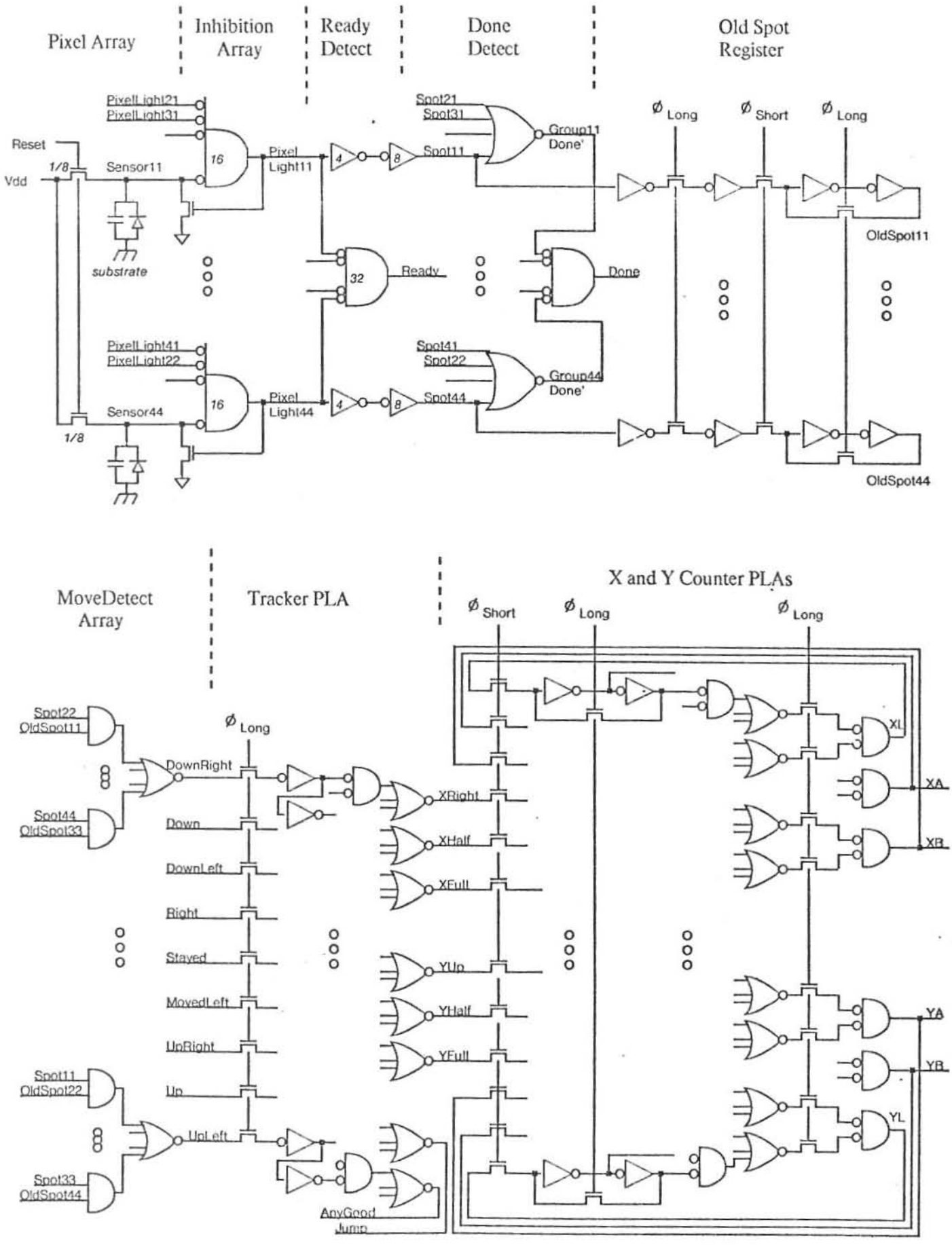


Figure 7. Overall Schematic

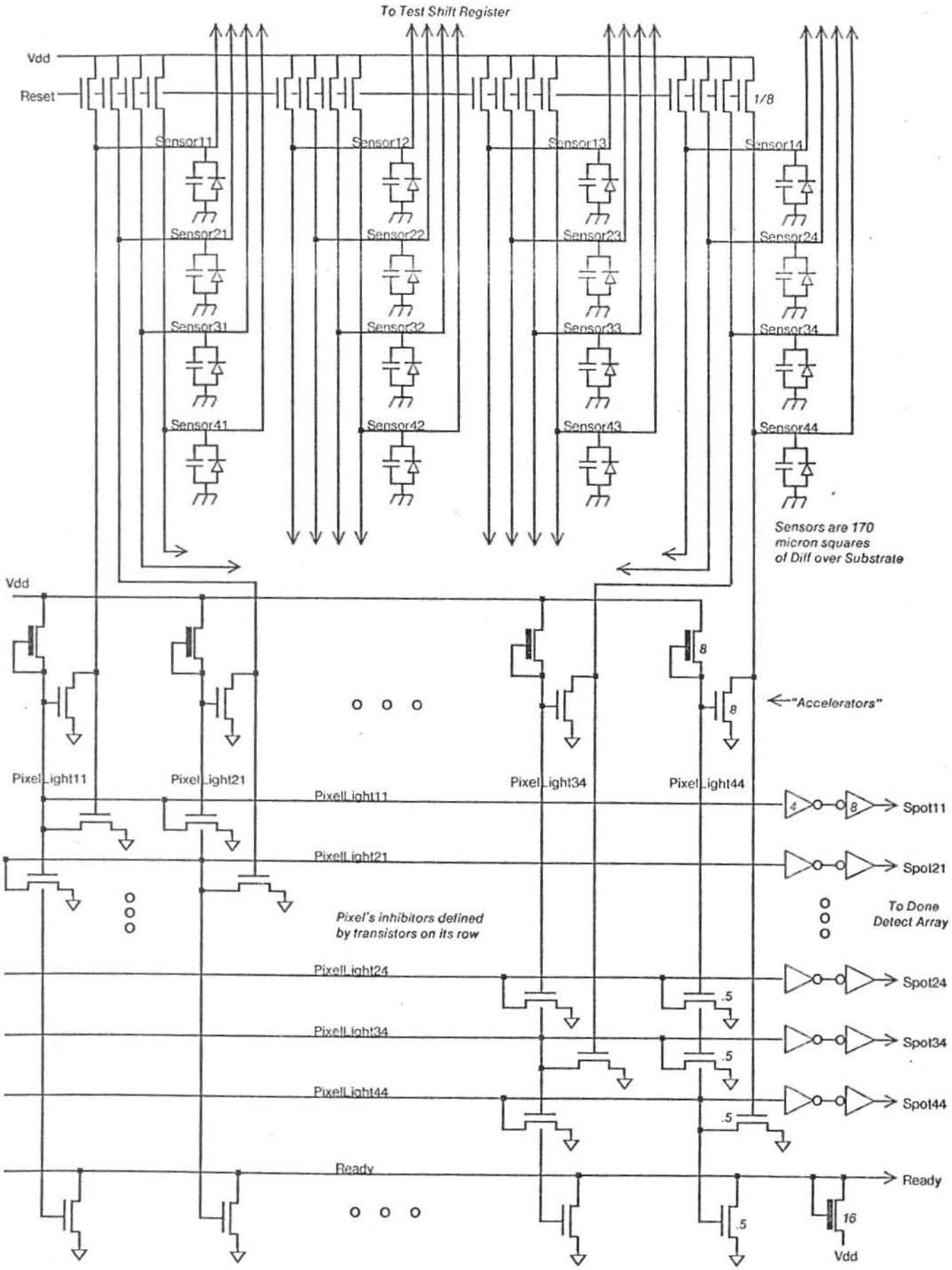
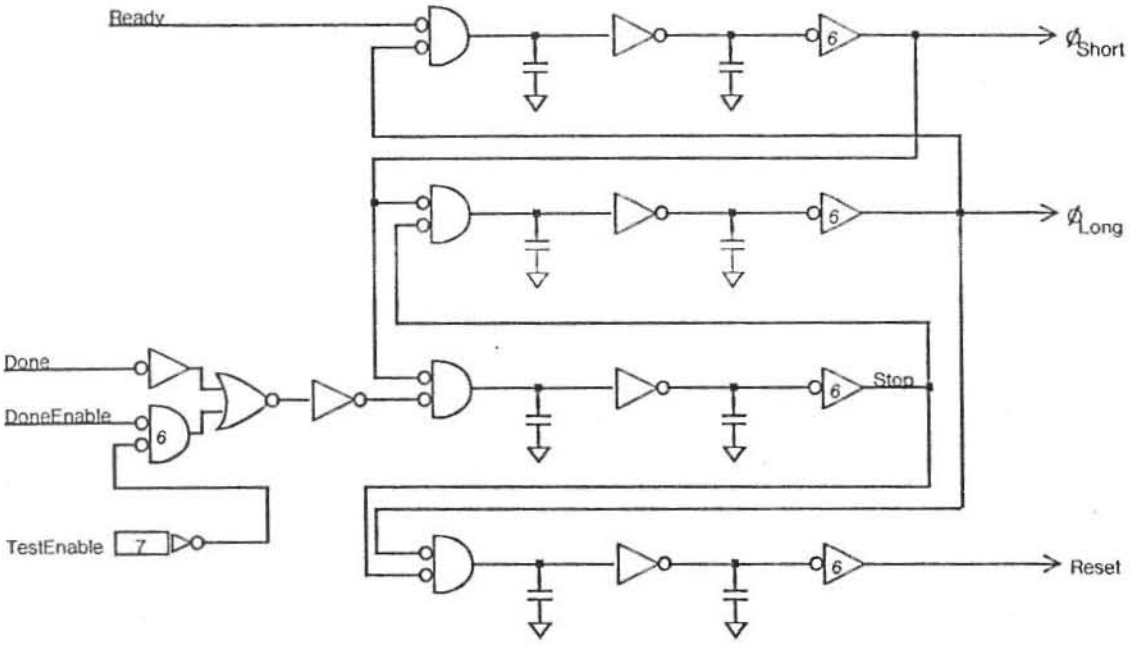


Figure 8. Pixel Array, Inhibition Array, & Ready Detect

Italic numbers are L/W for transistors and Zpu/Zpd for inverters.



Capacitors are 160x10 micron depletion-mode gates, or about .8 pF.
All gates have Zpu/Zpd = 6.

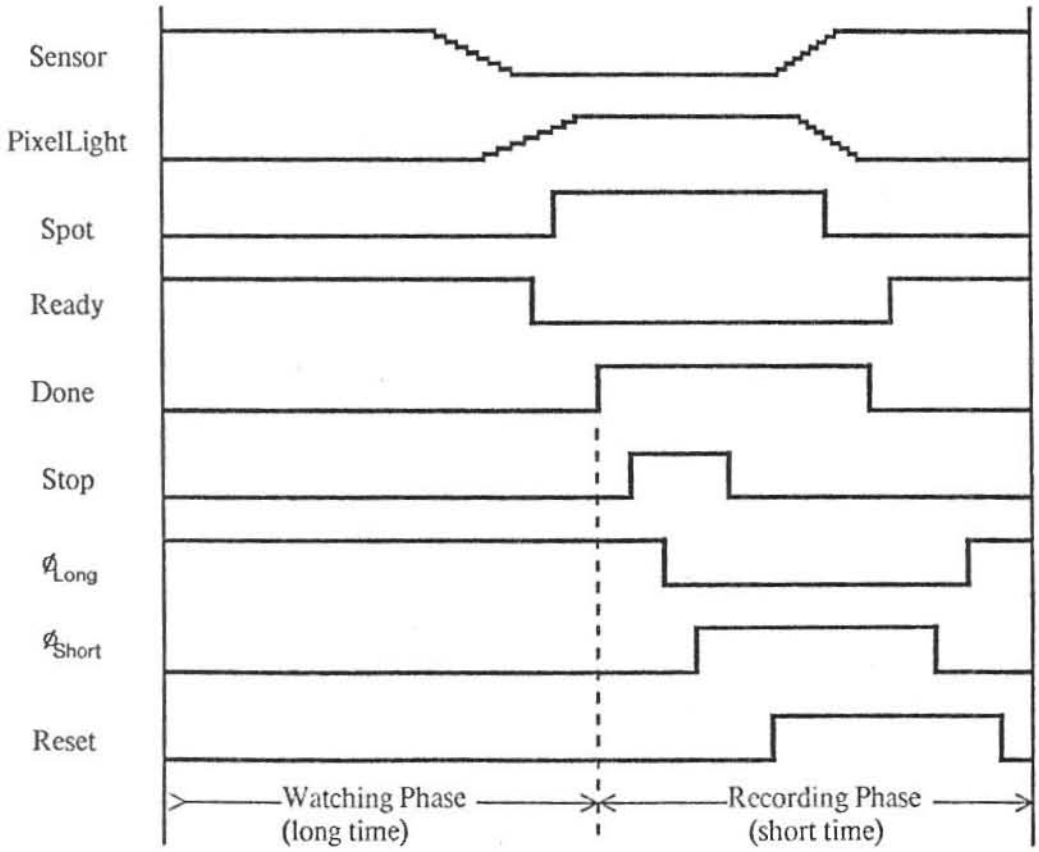


Figure 9. Timing Logic

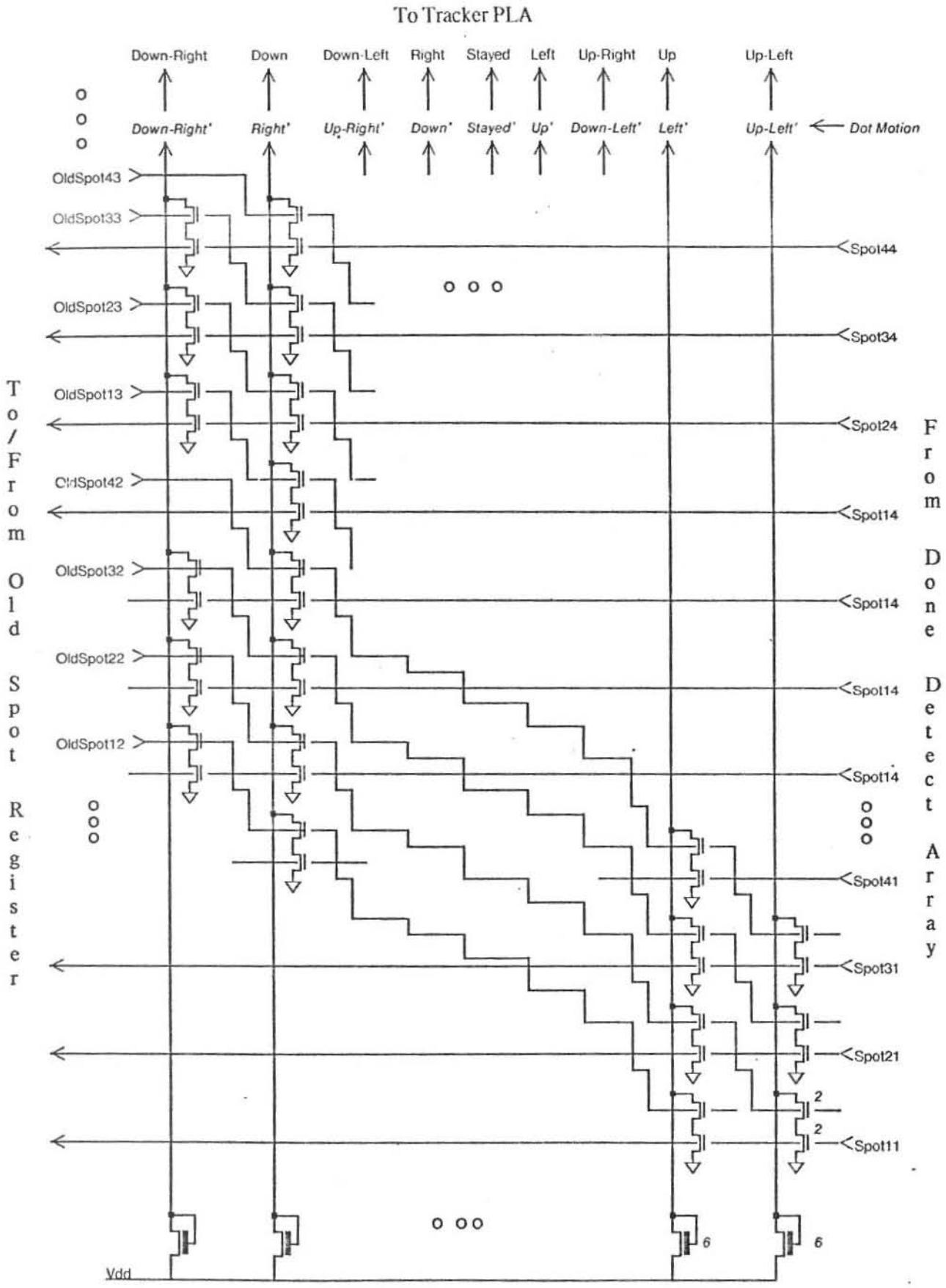


Figure 10. Move Detect Array

Italic numbers are L/W for transistors

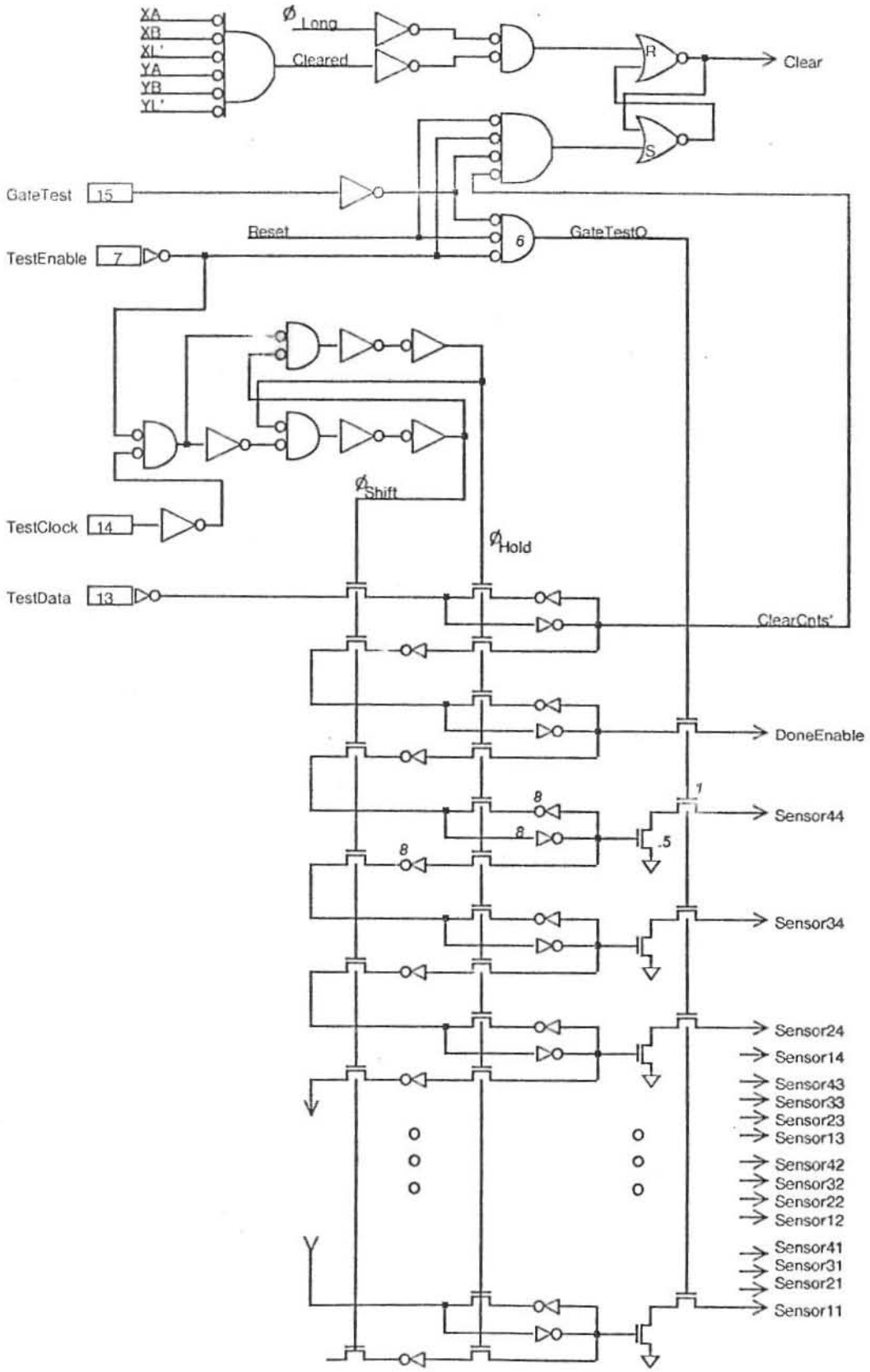
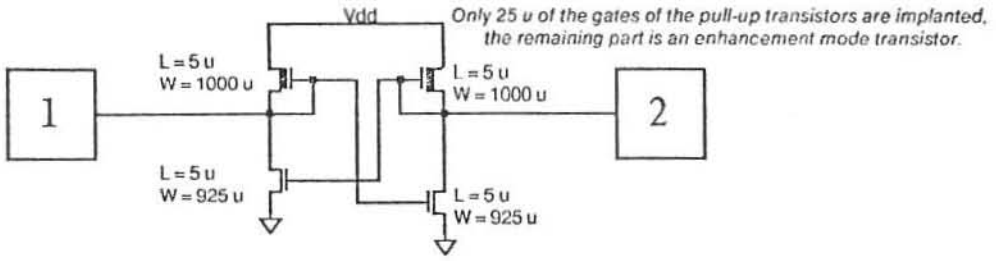
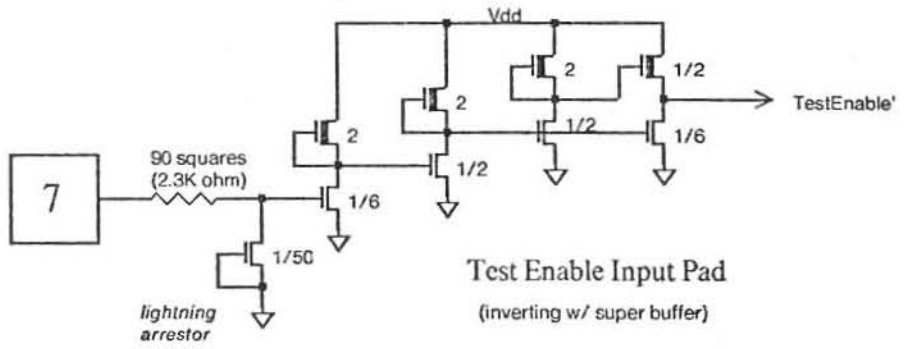


Figure 11Test Shift Register & Control

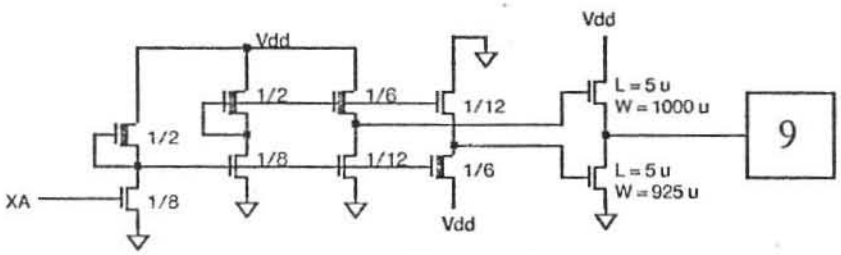
Italic numbers are L/W for transistors and Zpu/Zpd for gates



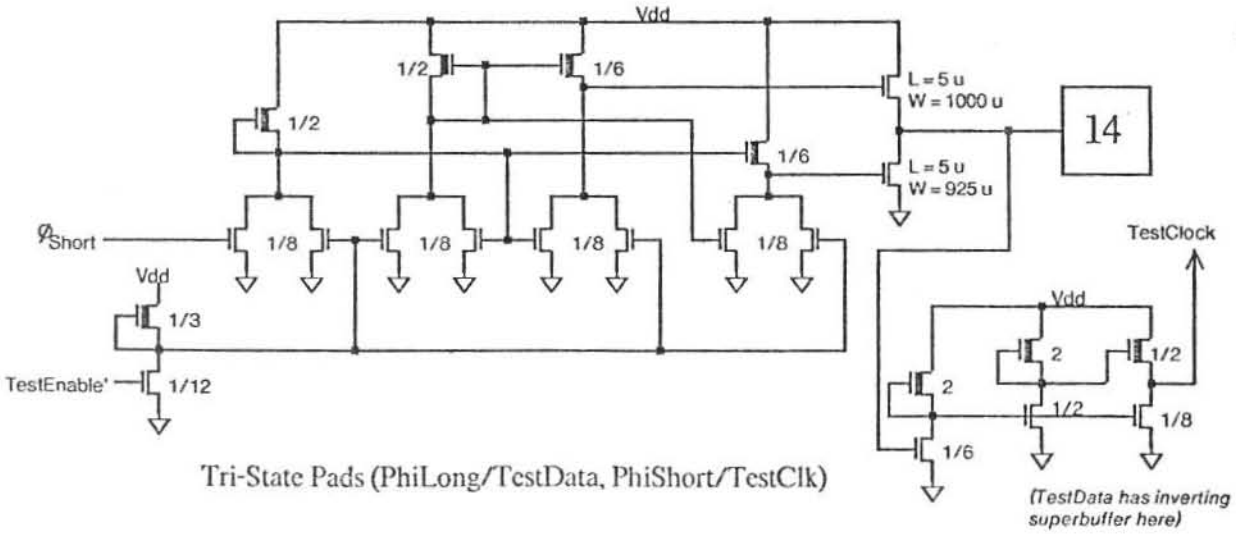
Debouncer Pads (RedA, RedB, YellowA, YellowB, BlueA, BlueB)



Test Enable Input Pad
(inverting w/ super buffer)



Output Pads (YA, YB, XA, XB)
(non-inverting)



Tri-State Pads (PhiLong/TestData, PhiShort/TestClk)

(TestData has inverting superbuffer here)

Figure 12.Optical Mouse Pads

Numbers are length/width for transistors

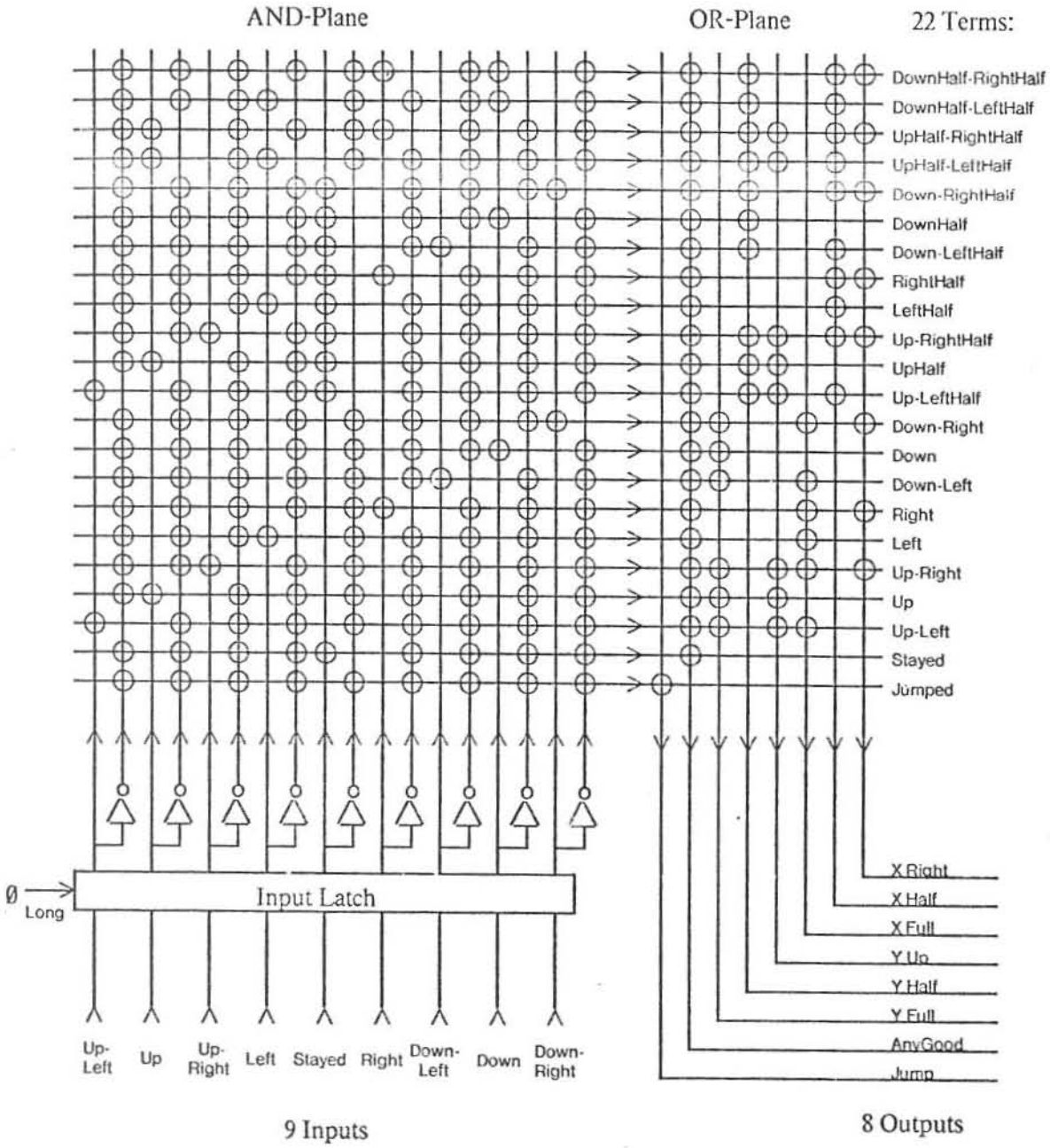


Figure 13. Tracker PLA

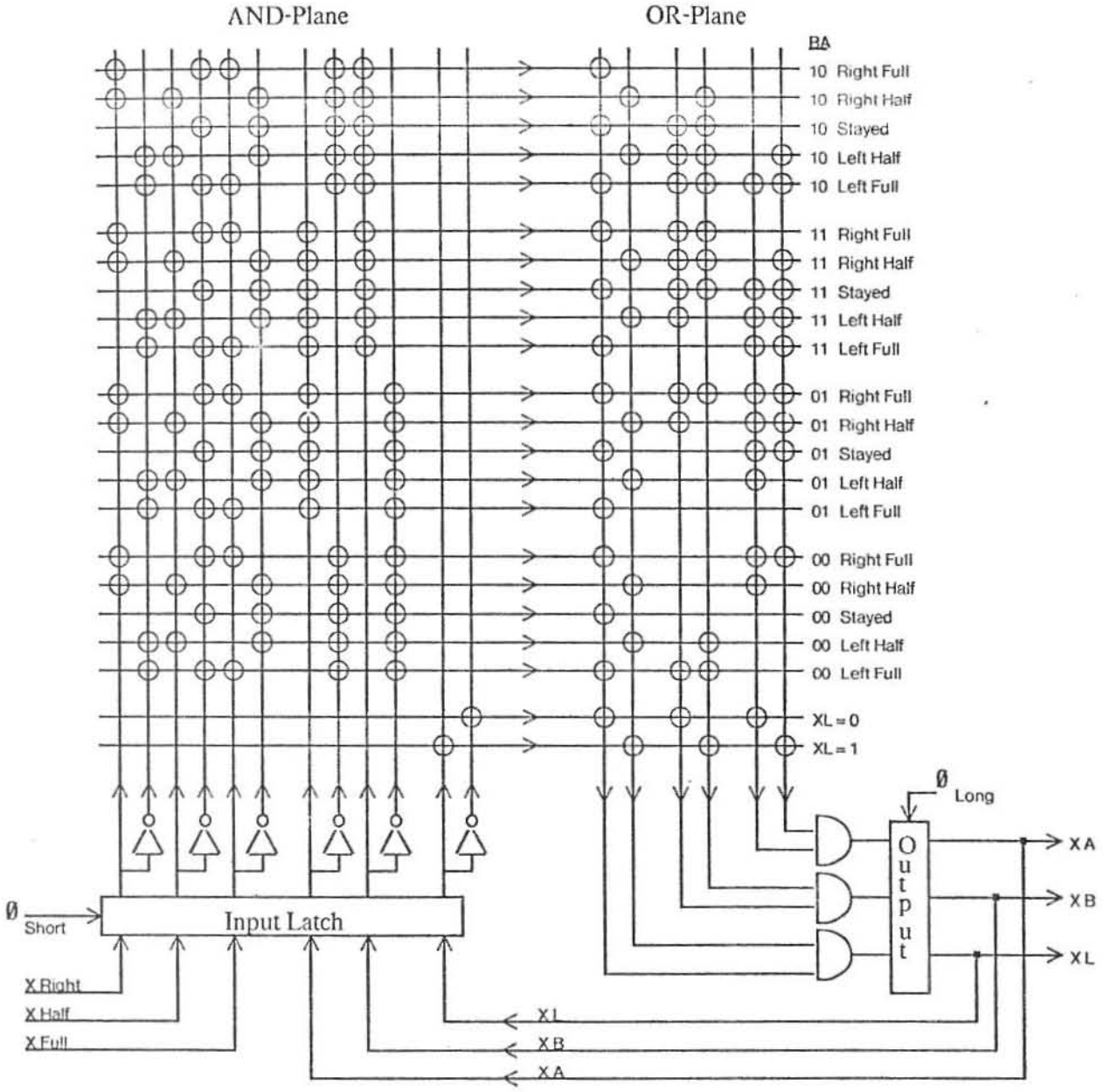


Figure 14. X Counter PLA

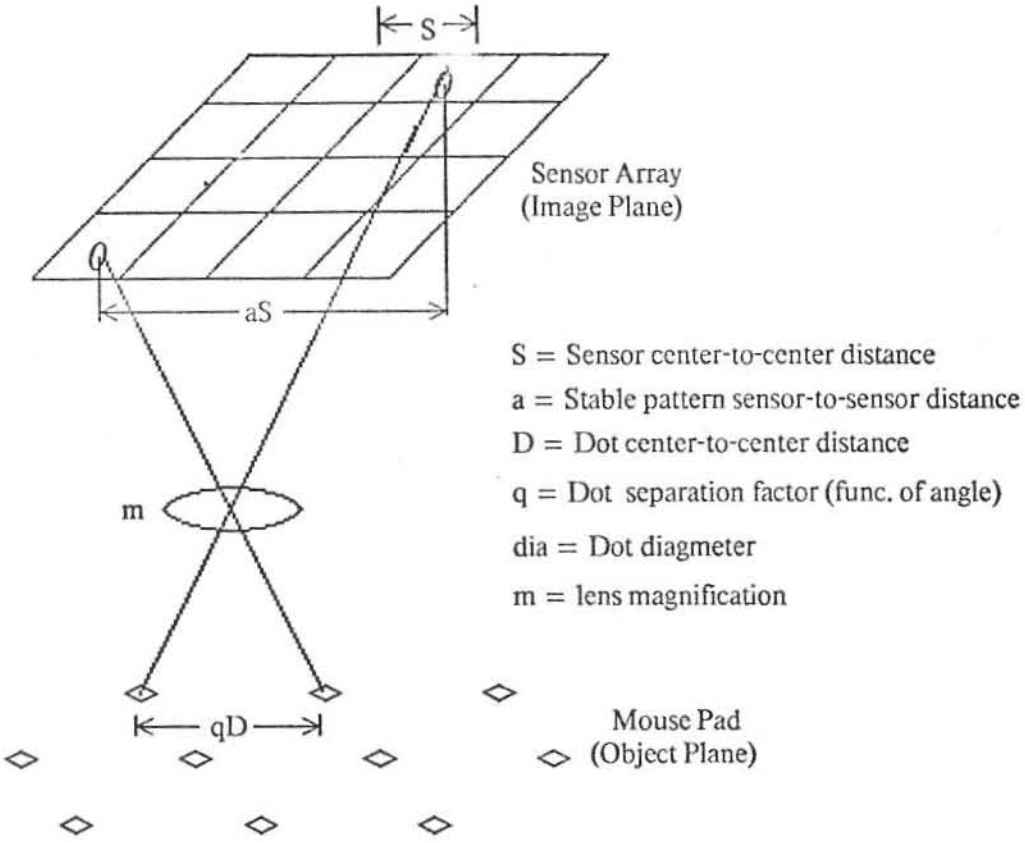


Figure 15. Projection of Pad onto Sensor Array

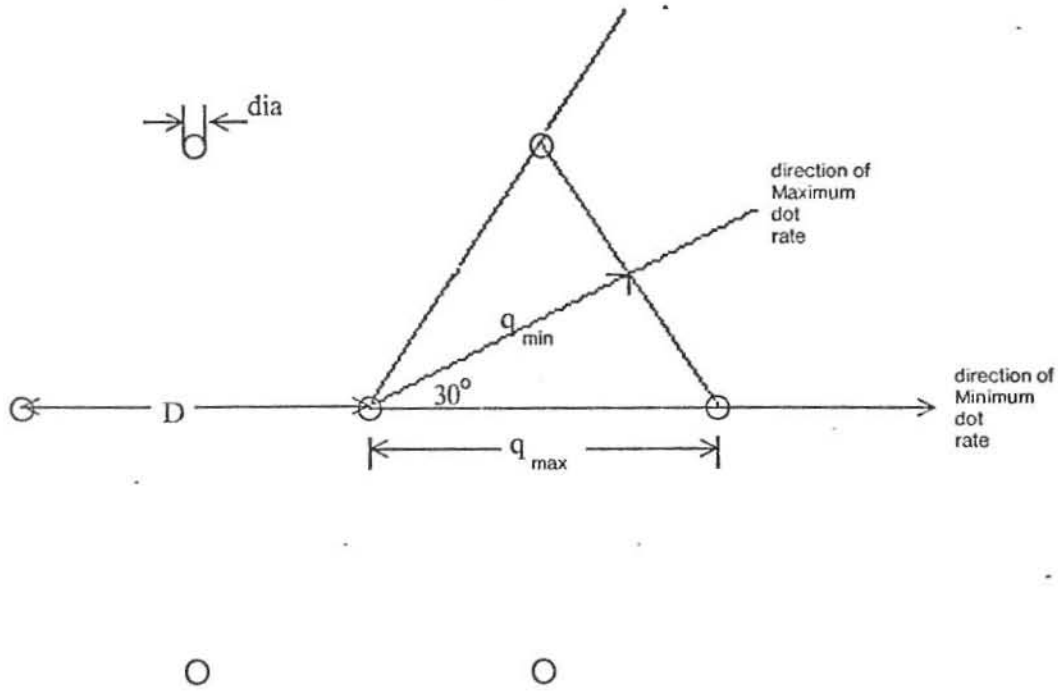


Figure 16. Hexagonal Pad Pattern Showing min & max Count Directions