Final Report

HARDWARE FOR A THREE-DIMENSIONAL DISPLAY

APPENDICES

Contract XG-2972

Between the United States Government

and

The President and Fellows of

Harvard University

Submitted to

Norman Prince

Contract Officer

Associate Professor Ivan E. Sutherland

Principal Investigator

August 1968

Final Report

HARDWARE FOR A THREE-DIMENSIONAL DISPLAY

APPENDICES

Associate Professor Ivan E. Sutherland

Principal Investigator

August 1968

APPENDIX 1

A Clipping Divider

(Thesis)

A CLIPPING DIVIDER

A thesis presented by

Robert Fletcher Sproull

to the

Committee on Applied Mathematics

in partial fulfillment of honors requirements

for the degree of

Bachelor of Arts

Harvard College

Cambridge, Massachusetts

13 May 1968

I. INTRODUCTION

This paper outlines a hardware technique designed to streamline some of the more complicated and time-consuming computations required for computer displays. The hardware is capable of performing operations on each piece of input data before it is displayed. Using this equipment, changes in the view of the data presented on the display can be made in real-time by altering parameters of the hardware operations rather than by changing the input data. Thus considerable computer time is saved by using the hardware features. These processes are capable of producing perspective views of three-dimensional data and of displaying a specific portion of the drawing defined by the input data.

For many computer graphics applications, the size and resolution of conventional displays is inadequate. We can, however, avoid screen-size limitations by storing information in page coordinates with a range several times the actual screen size. A portion of this page drawing, defined by a window on the page, is selected to display on the scope. The items within the window are mapped onto a portion of the display screen, called a viewport on the screen. This allows the scope screen to be partitioned into several viewport regions, each with different views presented.

This thesis describes a special piece of hardware, the clipping divider, which decides what part of a line, if any, is visible in the window. It then maps the visible portion of the line onto the viewport region. Typical

software implementations of this process for the PDP-1 computer require 5-7 ms. of processor time per line. Such a long computation time is acceptable if the display file is to be composed only once, but if the window and viewport dimensions are changing frequently, the computation time is excessive. The hardware implementation presented here will average 8 μs. a line and will not infringe upon central processor time.

The flexibility of the windowing operation permits savings in data storage as well as computation. The same data set used for internal processing can also be used directly for display. This eliminates the necessity of a display file, and thus represents a saving in storage and in computer time required to pack the display file. Since screen size is a parameter of the windowing operation, we are not limited to use of standard 10-bit displays.

The design of the clipping divider described here makes use of a new functional unit, the proportional divider. The clipper hardware contains eight of these units which are appropriately connected and controlled to accomplish the window clipping and viewport mapping operations. We have found that the use of proportional dividers is not limited to these two graphics applications. Other possible applications include a device to perform matrix multiplication, matrix division, and other linear vector computations.

The major work of this thesis was the specification and design of the logic to control the eight proportional dividers

in the clipper. The control unit includes about 25 flip-flops and 530 logic gates, wired in 3 card cages. The wiring is specified by 35 size C drawings. The control has been constructed and has passed static tests. It has not yet been tested at design speed with the proportional dividers.

The clipping divider described here is part of a larger project designed to produce three-dimensional perspective views through a special headset. The wearer will have the illusion of viewing a three-dimensional image. The user's head position is accurately measured and used to compute the changing view as he moves his head. Data appropriate to the view are presented to the clipper, which "clips" the lines to the boundaries of the user's viewing region, and does the division required to produce a true perspective view. The lines are then displayed to the user through a head-mounted CRT and appropriate optics.

The configuration of the complete system is shown in Figure 1. The operations of the various units are as follows:

1. The channel is the communicator between the central processor and the three hardware units. Instructions are fetched from the central memory, and appropriate data and control signals are passed to the three computation units. The exact operations of this processor have yet to be specified.

2. The matrix multiplier, not a part of this thesis, performs a matrix transformation on three-dimensional page information provided by the channel. It is capable of multiplying a four-vector by a 4x4 matrix which is stored

in hardware registers. The resulting four-vector is passed to the clipper or back through the channel to the computer. The multiplier may thus function as a driver for the clipper, or as a fast means of 4x4 matrix multiplication.

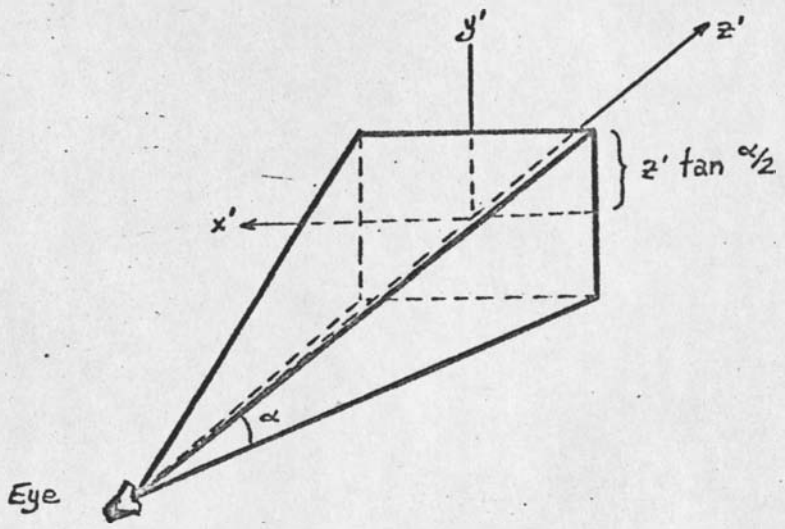
3. The clipper performs the three-dimensional windowing and viewport mapping operations, and prepares values of line endpoints in scope coordinates for the display generator.

4. The display generator is merely a line-drawer. For standard two-dimensional use, it will drive a conventional 10" CRT. For three-dimensional use, the head-mounted CRT will be attached. If the central processor feeds the correct matrix to the matrix multiplier, the view presented to the user should correctly reflect his head motions.

II. THE CLIPPING PROCESS

The purpose of the clipper is to decide which parts of a line in space coordinates are within the user's viewing region. It then computes the perspective view of any visible line segments.

The clipper requires that data about line endpoints be specified in the eye coordinate system. This coordinate system determines the user's viewing region:



Any lines which pass through the volume of the rectangular prism are visible to the observer.

The conversion from page coordinates stored in memory to eye coordinates is performed by the matrix multiplier. The matrix transformation consists of three operations: (1) establish the view position, (2) establish the view direction, and (3) establish the scale of the eye coordinates x', y', and z'.

$$[x\ y\ z\ 1] \times [\text{Translation}] \times [\text{Rotation}] \times [\text{Scale}] = [x'\ y'\ z'\ 1]$$

Input data in homogeneous coordinates.

Matrix stored in matrix multiplier.

Data passed to clipper.

The scaling function of the transformation matrix has special significance. The head-mounted CRT optics assume a viewing angle α of 40°. Thus a point is within the viewing region if

$$\begin{aligned} |x'| &\leq z' \tan \alpha/2 \\ |y'| &\leq z' \tan \alpha/2 \\ z' &\geq 0 \end{aligned} \qquad (1)$$

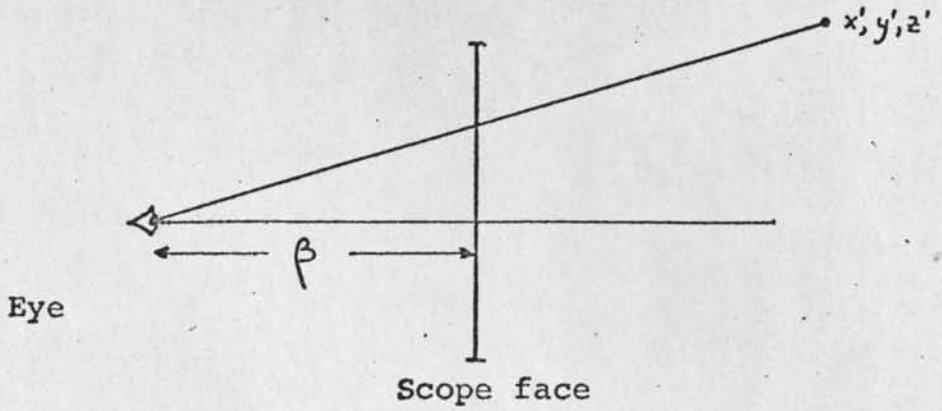
For computational convenience, z' is scaled by the factor $\cot \alpha/2$, so that we may rewrite the conditions as

$$\begin{aligned} |x'| &\leq z' \\ |y'| &\leq z' \\ z' &\geq 0 \end{aligned} \qquad (2)$$

If a point is within the viewing region, geometry shows that its perspective projection onto the scope can be computed by:

$$x_{scope} = \beta \frac{x'}{z'} \qquad y_{scope} = \beta \frac{y'}{z'} \qquad (3)$$

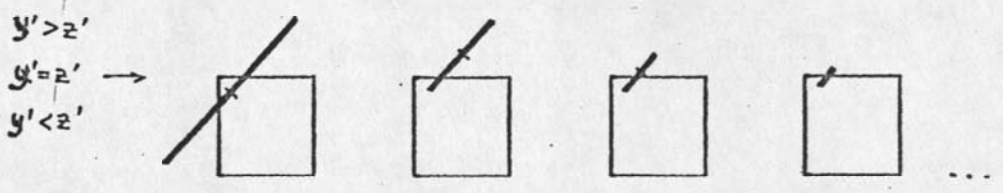
where the constant β is absorbed into the matrix transformation.



In practice, the clipper does not make decisions about actual points in or near the viewing region, but about lines as described by their endpoints. Some lines will not pass through any part of the view region, and some will lie entirely within that region. In the cases where only part of the line passes through the viewing region, the clipper must compute the information necessary to display the visible line segment.

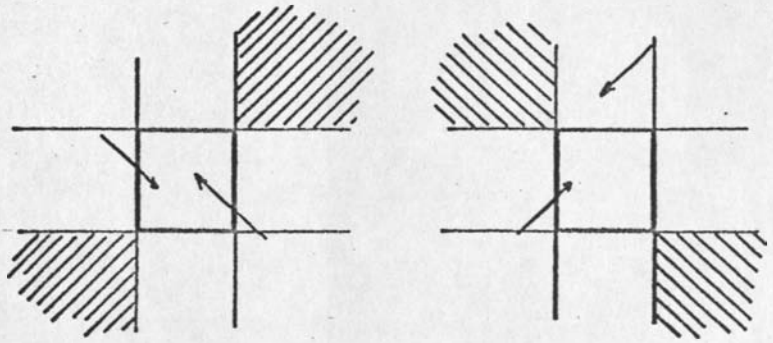
The various distinct cases presented to the clipper are shown in Figure 3. If both endpoints are already within the window boundaries, as is shown in (a), the clipping process need not be performed, and we may proceed directly to the perspective division. If both endpoints are off the screen, but on the same side of the window, the clipper is capable of announcing immediate rejection of the line, as in (d). If, as in case (d), the x' coordinate values of both endpoints are to the right of the right-hand edge of the window, no points along the line can possibly fall within the window. Cases (b), (c), and (e) permit no immediate decisions, and require further computation.

The computation used to determine the intersections of the line and the edges of the window involves calculating the coordinates of the midpoint of the line in x', y', and z' coordinates, and examining the midpoint coordinate values. We note that half of the line still intersects the window boundary, and half does not. The second half of the line is "thrown out" by placing the midpoint coordinates in the registers holding the "thrown out" endpoint coordinates. The step is repeated, taking the midpoint of the "new" line, and so forth until the "endpoints" coincide at the intersection point. This process must be performed twice to discover both intersection points. In the clipper design, we perform these two operations in parallel in order to achieve high speed.



This algorithm solves the situations of Figure 3 (b) and (c). If one endpoint is originally in the window, the clipper box searches for only one intersection of the line and the window boundary. When the clipping process is complete, the points of intersection, N_f and P_f, are guaranteed to be in the window. If an endpoint was originally in the window, then the resulting answer is merely that endpoint.

Lines which fail to intersect the viewing, such as Figure 3 (e), are rejected by the clipping process. This is accomplished by defining regions of absolute failure:



If any computed midpoint falls within the shaded regions, the line is rejected. Note that the regions defined for lines with positive slope are different from those for lines with negative slope.

We now wish to perform the perspective division suggested by (3), and map these points onto the CRT screen using viewport dimensions supplied by the central processor (Figure 2d). We have the points N_f and P_f (still with z' information):

$$N_f = (x'_{Nf}, y'_{Nf}, z'_{Nf}) \qquad P_f = (x'_{Pf}, y'_{Pf}, z'_{Pf})$$

Then, using the viewport dimensions V_{xs}, V_{xc}, V_{ys}, V_{yc}, where the subscripts refer to size and center (see Figure 2), we compute the scope coordinates:

$$x_{Ns} = \left(\frac{x'_{Nf}}{z'_{Nf}}\right) V_{xs} + V_{xc}$$

$$y_{Ns} = \left(\frac{y'_{Nf}}{z'_{Nf}}\right) V_{ys} + V_{yc}$$

$$x_{Ps} = \left(\frac{x'_{Pf}}{z'_{Pf}}\right) V_{xs} + V_{xc}$$

$$y_{Ps} = \left(\frac{y'_{Pf}}{z'_{Pf}}\right) V_{ys} + V_{yc} \qquad (4)$$

In each case, the division is guaranteed not to overflow, since the points are in the window. The resulting scope coordinates are passed to the line generator for display.

The case of two-dimensional clipping is almost identical with that of three-dimensional, except that values for the window size and position (which take the place of z' information) are fetched from hardware registers. The mapping scheme is outlined in Figure 8a. Information in page coordinates is examined to see if it intersects the window in a manner similar to the three-dimensional case. Then the coordinates of any clipped lines are mapped onto the scope screen by the same division procedure described above.

III. PROPORTIONAL DIVIDERS

The hardware implementation of the clipping and division procedures is accomplished with units called proportional dividers. Figure 6 shows schematically the basic arrangement, which consists of an accumulator and a shift register attached to a one's complement adder. The adder forms the difference $D = A - \Delta$. Various tests can then be performed on D, A, and Δ, and D is either strobed into register A, or not. Register Δ can be shifted right in one-bit steps.

As a single unit, this configuration accomplishes very little -- it accepts a string of "accept $D \rightarrow A$" or "reject D" signals. If the test used to generate this string insures strobing A when the signs of A and D match, the string is actually the result of dividing the number in A by that in Δ. Consider using this sequence of decisions to drive another such unit (Figure 7). If this sequence represents the quotient $Q = \frac{A_1}{\Delta_1}$, and if the bit sequence of A drives unit 2, we have finally in A_2

$$A_{2f} = - Q\Delta_{20} + A_{20} \tag{5}$$

This is precisely the necessary computation for the scope coordinate information,

$$x_{Ns} = \left(\frac{x'_{Ns}}{z'_{Ns}} \right) V_{xs} + V_{xc} \tag{4}$$

The clipping process is accomplished with four proportional dividers ganged together. The A registers contain

the current endpoint coordinates. The Δ registers initially contain the vector which, when subtracted from the A vector, will yield the A vector for the other endpoint. Tests are first made to discover which endpoints, if any, are initially within the window region. Then the delta registers are shifted right one step. When the adder is enabled, the resultant D vector will be the line's midpoint. Then tests are made on the D vector to discover which half of the line should be "thrown out." This decision determines whether the D vector is strobed into the A registers. Then the delta registers are shifted right and the process is repeated until all the delta registers are zero. The A registers then contain the coordinates of the intersection of the line and the window boundary.

These two processes do not exhaust the possible applications of proportional dividers. With slightly altered control circuitry, the divider units can greatly facilitate the problem of two-dimensional display subroutining. The aim is to be able to position instances of the subroutine picture at will on the page. It is particularly convenient to specify the absolute size of the instance, and thus be able to use one subroutine data set for instances of all sizes.

We will assume in the discussion that there are several master pictures (e.g. display data for a transistor, a resistor, or a capacitor), a page picture with subroutine calls to these masters, and a scope display resulting. In three-dimensional subroutining, the mapping may be visualized as:

$$\left[\text{Master data}\right] \times \left[\begin{array}{c} \\ A \\ \\ \end{array}\right] \times \left[\begin{array}{c} \\ B \\ \\ \end{array}\right] = \left[\text{Eye data}\right]$$

Master-to-page transfrm. Page-to-eye transfrm.

Thus the matrix AxB should replace the current matrix B in the matrix multiplier. The old matrix B may be placed in a push-down stack save in order to reëstablish the old page-to-eye transformation when exiting from the subroutine. Subroutines may then be nested to any depth.

The more complicated two-dimensional process is illustrated in Figure 8b. When the subroutine call is contemplated, the window on the page is defined by W_L and W_R, and the viewport on the scope by V_L and V_R. The subroutine call specifies that a portion of the master defined by M_L and M_R is to be mapped onto the page in an instance specified by the coordinates I_L and I_R. The aim is to provide to the clipper the information necessary to map the master portion directly to a new viewport portion. As shown, the mapping

$$\begin{aligned} W'_L &\rightarrow W_L \\ W'_R &\rightarrow W_R \\ V'_L &\rightarrow V_L \\ V'_R &\rightarrow V_R \end{aligned} \qquad (6)$$

will accomplish this task, where

$$V'_L = V_c + V_s \left[\frac{I_L - W_c}{W_s}\right]^*$$

$$V'_R = V_c + V_s \left[\frac{I_R - W_c}{W_s}\right]^*$$

$$W'_L = M_c + M_s \left[\frac{W_L - I_c}{I_s}\right]^*$$

$$W'_R = M_c + M_s \left[\frac{W_R - I_c}{I_s}\right]^* \qquad (7)$$

where the asterisk means that the maximum absolute value of the expression may not exceed 1. In Figure 8, the expressions exceeded 1, and were replaced by ± 1, depending the sign of the expression. Thus $W'_L = M_L$ and $W'_R = M_R$ for Figure 8. Each of the quantities in (7) must be computed separately for x and y coordinates. The subscripts s and c refer to size and center; any variable with these subscripts enjoys the relation:

$$A_c = (A_L + A_R) / 2$$
$$A_s = (A_R - A_L) / 2 \qquad (8)$$

Figure 9 shows a more complicated mapping in which the instance is not entirely contained within the window area. We can simplify the problem by noting that $W_L < I_R < W_R$ (x coordinates), which means that the right instance edge is in the window, and thus

$$W'_R = M_R$$

$$V'_L = V_L \quad . \tag{9}$$

We need only compute the other two values, W'_L and V'_R. This is accomplished with the proportional divides given in (7), where we know that the divisions do not overflow.

In practice, the subroutine call places the old window and viewport information on a push-down stack and causes the new values to be computed. Within the subroutine, data from the master picture are presented to the clipper for display, using the new mapping parameters. Exiting from the subroutine involves popping the old mapping information off the stack. Subroutines can thus be nested to a depth limited only by stack storage.

IV. CLIPPER ORGANIZATION

The clipper hardware consists of eight proportional dividers, miscellaneous data paths for transferring data to appropriate registers, and extra storage registers for holding window, viewport, and "old endpoint" coordinates. The data paths are all 20 bits wide to provide 18-bit resolution with round-off and overflow information. The control for the clipper is designed as a separate unit.

The complexity and quantity of data paths do not facilitate construction, but are required for parallel operation. The complexity of the data paths affords considerable flexibility: data can be inputed in absolute coordinate format, relative coordinate format (vector increments), or center/size format (facilitating instance positioning in two-dimensional subroutining). Input can be 72 bits wide from the matrix multiplier or 18 bits wide from the channel. The extra registers have enough storage to be able to perform the new window and viewport computations without stopping to input more data.

Figure 10 shows the actual arrangement of registers. In use, the data pertaining to one endpoint are processed in the top four dividers, and those pertaining to the other endpoint in the bottom four. The dividers are ganged four across for clipping and in two 2-divider units for the perspective division and the two-dimensional subroutining operations. The couple-by-two and couple-by-four conditions are switched by the clipper control.

The tests which determine whether to strobe the divider accumulators are implemented by the control. During division, testing logic is attached to the "master" unit, and the decision is issued to both master and slave. For clipping, the appropriate tests are implemented using signs of all four delta's and all four computed midpoint coordinates (D's). The delta signs must be tested since the regions of failure vary according to the direction of the line segment under test.

The hardware implementation of the clipping algorithm does not follow exactly the description of Section II. The question "Is the midpoint on the same side of the window boundary as the A vector?" is difficult to answer without elaborate and slow register comparison logic. The process is altered slightly, as described in Appendix I, to simplify hardware design. In addition, there are several coordinate transformations contained within the clipping procedure to change the question "Is A greater than Δ?" to "Is A-Δ greater than zero?", since the latter involves only a test of sign bits.

V. CLIPPER CONTROL

The main work involved in this thesis was the design of the clipper control logic. The control drives some 60 enable and strobe signals required by the data logic, and responds to about 100 sign bit tests which are generated in the data section. The requirements of the control are:

1. It must handle communications between the clipper and the other processors of Figure 1.

2. It must provide the proper enable signals to the adder buses based on the configurations desired. These include the ability to input data, fetch data from storage registers, enable the proportional dividers, etc.

3. It must implement all the tests which drive the proportional dividers, based on sign bit tests from the data section. Appropriate strobe pulses are then prepared.

All these operations are performed in response to a "directive" word which is received with each service request. The directive may specify the following operations:

1. Fetch data from clipper registers.
   Microcoding designates the registers requested, and the output format desired.

2. Input data to clipper registers.
   Microcoding specifies the registers involved and the input format (e.g. absolute, relative, center/size).

3. Input data and "box."
   This inputs new master and instance information, and then performs the two-dimensional subroutining operation, leaving the new window and viewport values in the appropriate clipper storage registers.

4. Input data and "clip."
   This is the standard clipping process. Many variants may be specified in the microcoded portion of the directive:

   1. Two-dimensional (use window size stored in clipper), or three-dimensional (use z' information provided).

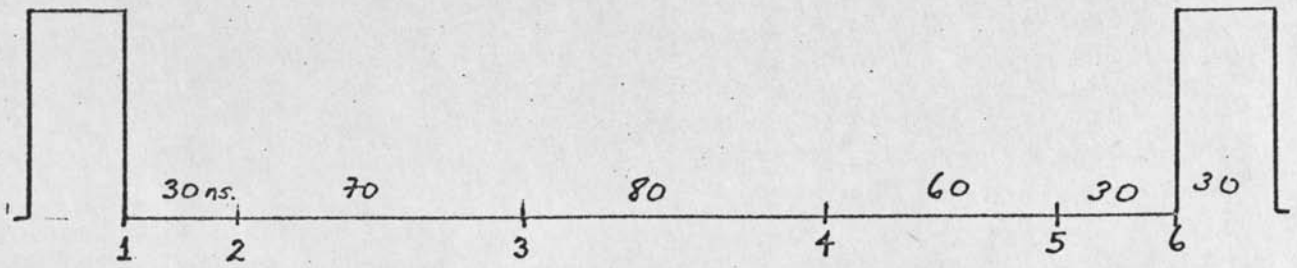
   2. Minimum effort mode. This allows the clipping process

to proceed only as far as necessary to establish whether the line presented for clipping intersects any part of the window. This has applications in a particular hidden-line removal scheme and in coincidence detection with stylus input devices such as the Rand Tablet.

3. Curve mode. Lines with negative z' are accepted. This is equivalent to dropping the rule $z' \geq 0$ from the conditions (2).

In order to control these processes, twenty "states" of the clipper control are defined, as shown in Figure 11. The progression of steps required for a standard clipping operation such as Figure 3 (b) is shown in heavy lines. Figure 12 shows a block diagram indicating the sources of the control signals. The bus enabling signals are driven by logic connected to the state flip-flops. Strobe pulses are made up from test results and state information. The next state is determined from tests and the present state information.

The time spent in each step is determined by the complexity of the functions performed during that state. States last either 150 ns. or 300 ns., depending upon the time required for data buses and adders to settle. Thus the wait and output states may be short, but the clip state is long since the adders must settle. A typical long cycle might be partitioned as follows:



1. New state transition.
2. Enable signals to adder buses.
3. Buses settled.
4. Adders done (80 ns. add).
5. Tests done.
6. New state determined, as well as strobes.

This feature is implemented by inhibiting one clock pulse for a long cycle.

The control is also wired to handle the "curve mode" operation. This special feature allows three-dimensional information with negative z' values to be processed. In physical terms, the line in question is behind the observer, but parametric generation of three-dimensional space curves produces the three-vector (x', y', z') with the understanding that the point's perspective position is to be

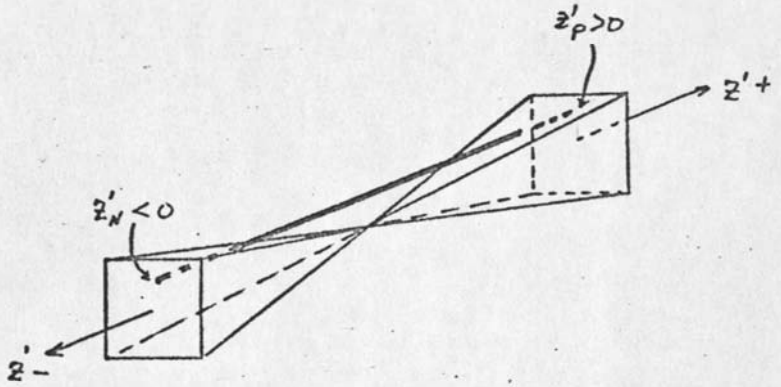
$$\left(\frac{x'}{z'}, \frac{y'}{z'}\right)$$

which can still be on the screen, even if z' is negative, as long as

$$|x'| \leq |z'|$$
$$|y'| \leq |z'|$$

In curve mode, if z' coordinates of both endpoints are negative, both are complemented, and the clipping process is begun.

Upon completion, the control returns to the idle, or wait state. If either endpoint has negative z', but not both, the clipping process is attempted twice, once using complemented values, and once using true values. The double clipping is required for lines such as:



In this case, the clipper outputs two lines to the line drawer.

APPENDIX I

A more precies definition of the clipping process is as follows:

We define two vectors $\underline{\Phi}_N$ and $\underline{\Phi}_P$ by the Boolean:

$$\underline{\Phi}_\alpha = \{(x'_\alpha - z'_\alpha > 0), (x'_\alpha + z'_\alpha < 0), (y'_\alpha - z'_\alpha > 0), (y'_\alpha + z'_\alpha < 0)\}$$

Then if the intersection $\underline{\Phi}_N \cap \underline{\Phi}_P \neq 0$,we announce that the line is outside the viewing region, as in Figure 3 (d) where $x'_P - z'_P > 0$ and $x'_N - z'_N > 0$. If both vectors $\underline{\Phi}_\alpha = (0,0,0,0)$ then we announce that both endpoints are inside the viewing region, and proceed to the perspective division.

In many cases, such as Figure 3 (b), (c), and (e), the decision is not as simple. Consider the directed line segment $\overrightarrow{PN}$ in Figure 4. Neither N or P is inside the window. We form the differences

$$\Delta_{Px} = x'_N - x'_P \qquad \Delta_{Py} = y'_N - y'_P \qquad \Delta_{Pz} = z'_N - z'_P$$

If any fixed proportion of these delta values is added to $P = (x'_P, y'_P, z'_P)$ we obtain a point along the line $\overrightarrow{PN}$. If we add the full delta values, we find the other endpoint, N, which is not in the window. In this case, the delta registers are shifted right one step and the process is repeated. This time, as the delta values are added to the coordinates of the point P, we find the midpoint of $\overrightarrow{PN}$, P_{tl}. In fact, P_{tl} is in the window, and the $P = (x'_P, y'_P, z'_P)$ is replaced

by $P_{t1} = \{ x_P' + \Delta_{Px}, y_P' + \Delta_{Py}, z_P' + \Delta_{Pz} \}$, where the delta values are the current (shifted) values. The deltas are then shifted again and the process repeated. When the delta registers all finally reach zero, we are at the point P_f, the desired answer.

Note that the third midpoint P_{t3} was in the "prime test failure region" (defined here as $x_P' - z_P' > 0$ or $y_P' - z_P' > 0$) and the coordinates of P_{t2} were not replaced by those of P_{t3}. The next trial, P_{t4}, was successful. The same process is done (simultaneously, but independently) for the directed segment $\overrightarrow{NP}$, yielding the point N_f as the final result.

The case of Figure 3 (e) is somewhat more difficult. Rejection cannot be announced on the basis of Φ vectors, but is discovered only after several clipping steps. Figure 5 defines another region for the directed segment $\overrightarrow{PN}$, the "secondary test failure region." Point P_{t1} fails the prime test, and also fails the secondary test ($x_P' - z_P' < 0$ or $y_P' - z_P' < 0$). In this case, any further excursions along the line $\overrightarrow{PN}$ will yield no points on the window, and the line is rejected.

The condition of Figure 3 (b) requires special treatment. The first attempt P_{t0} is indeed the other endpoint N, which is in the window, and satisfies all acceptance criteria. The information is strobed into the P registers, but the P delta registers are zeroed. Thus, subsequent computed "midpoints" are just the point N, and are accepted each time.

SUMMARY:

1. First test
   a. $\underline{\Phi}_N = \underline{\Phi}_P = \{0,0,0,0\}$ ⟶ clip done
   b. $\underline{\Phi}_N \cap \underline{\Phi}_P \neq 0$ ⟶ reject
   c. otherwise, go to 2.
2. First clip step
   a. Prime test OK ⟶ replace P with P_{t0} and zero deltas.
   b. Prime and secondary tests fail ⟶ reject
   c. Otherwise, go to 3.
3. General clip step
   a. Prime test OK ⟶ replace P with P_{ti}
   b. Prime and secondary tests fail ⟶ reject
   c. All deltas zero ⟶ clip done
   d. Otherwise, go to 3.

FIGURES

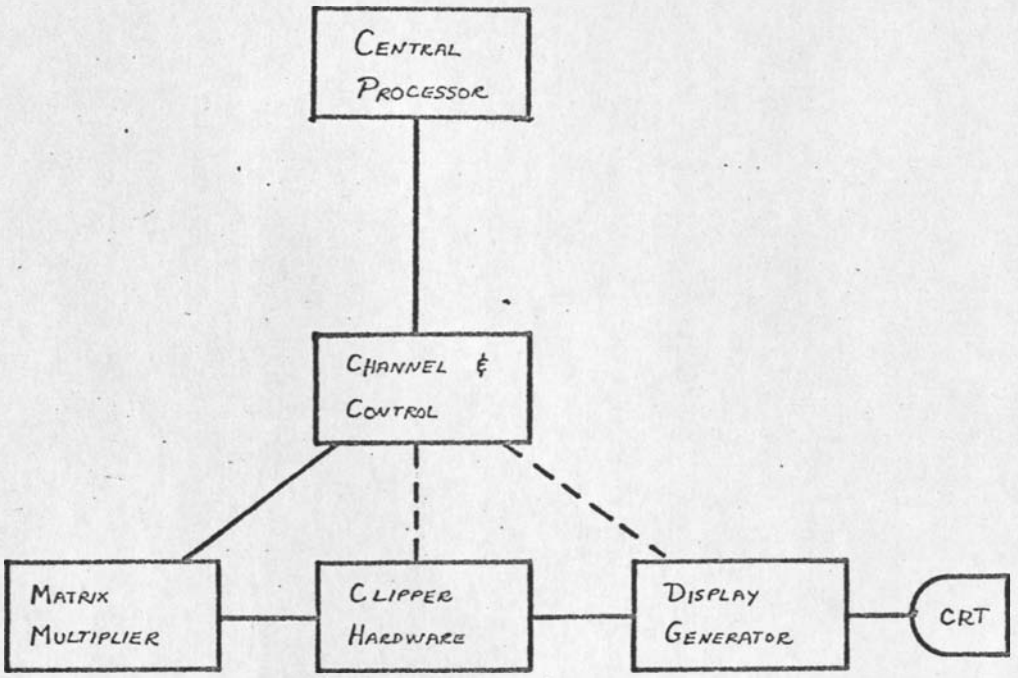
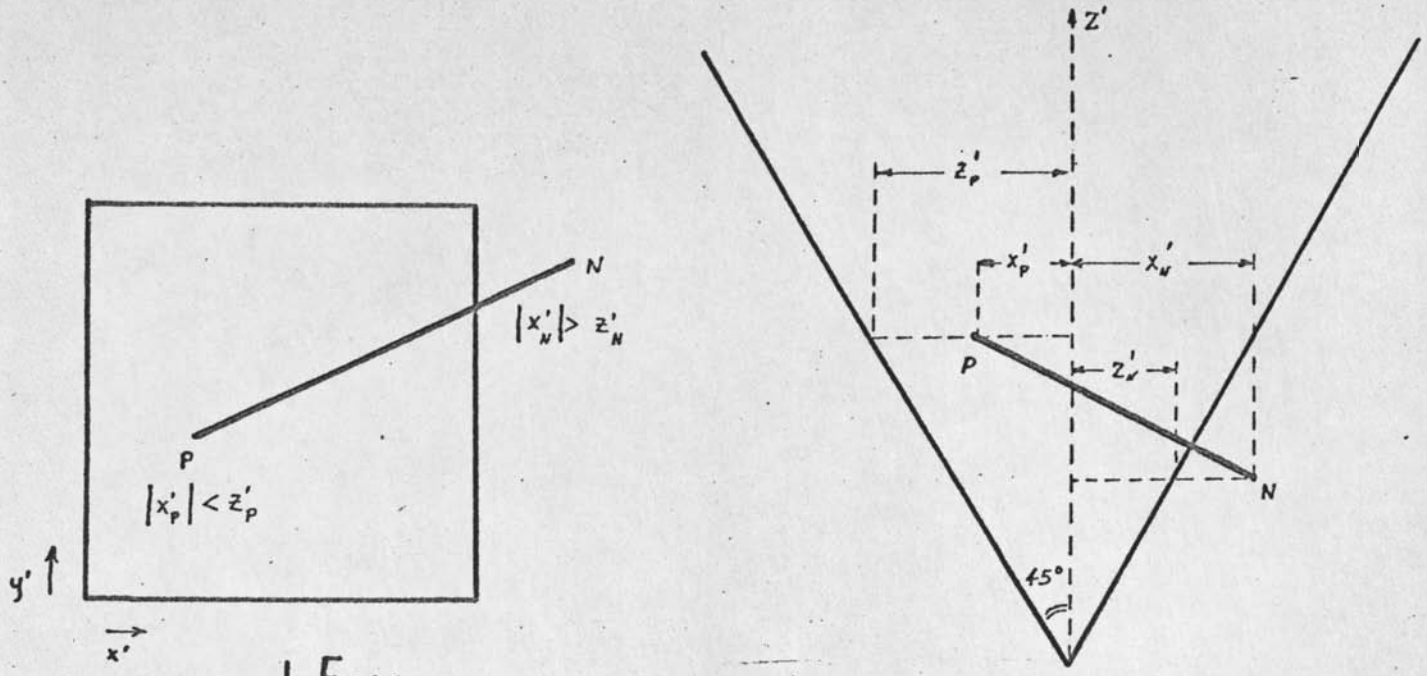
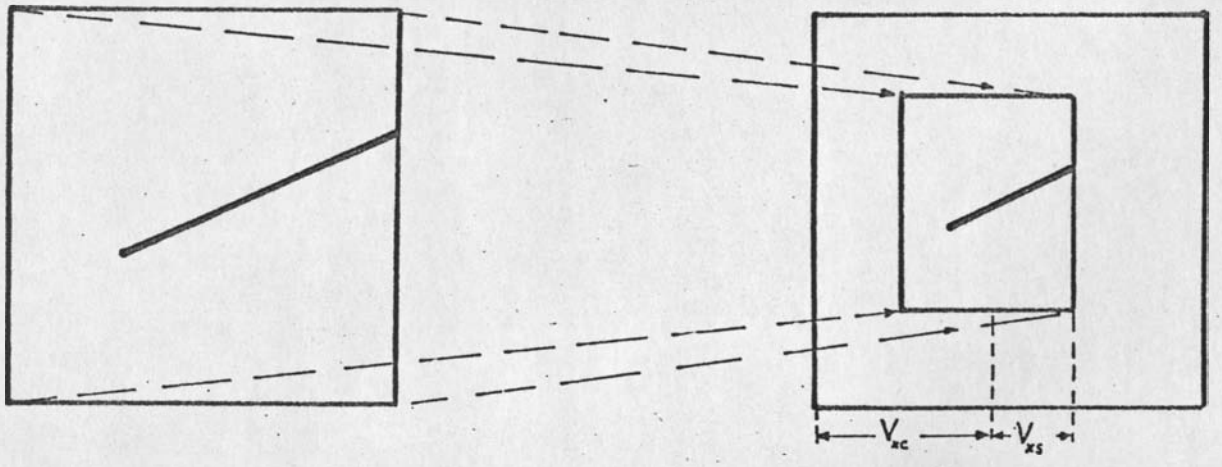

CENTRAL PROCESSOR
CHANNEL & CONTROL
MATRIX MULTIPLIER
CLIPPER HARDWARE
DISPLAY GENERATOR
CRT

FIG. 1



a. LINE IN 3 DIMENSIONAL WINDOW

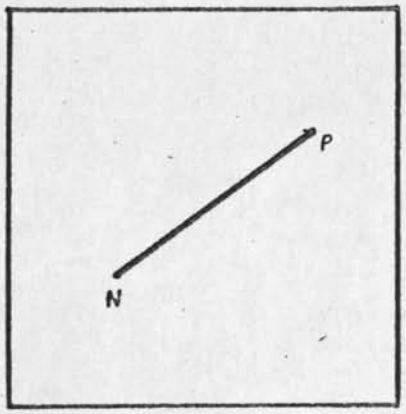
b. EDGE VIEW



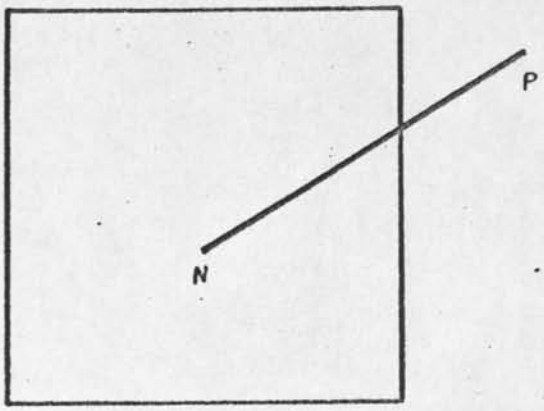
c. AFTER CLIPPING

d. SCOPE VIEW

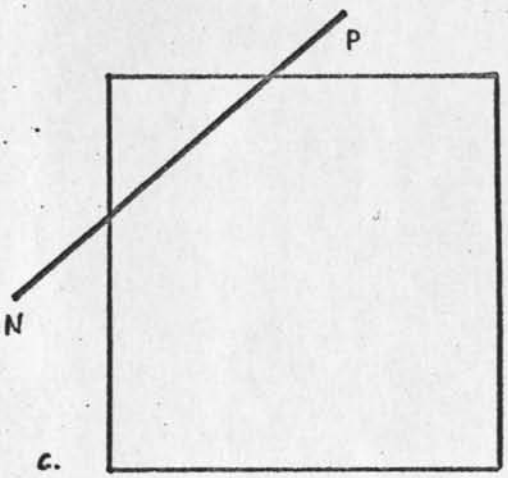
FIG. 2



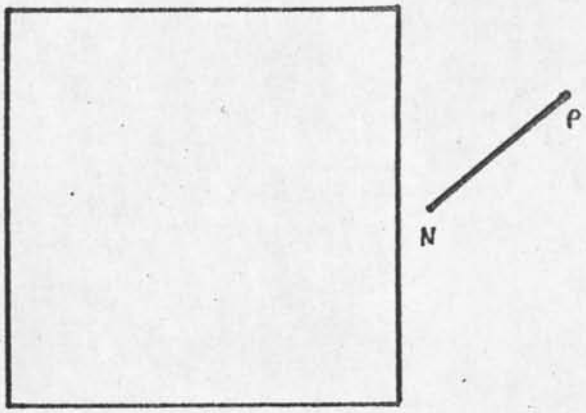
a.



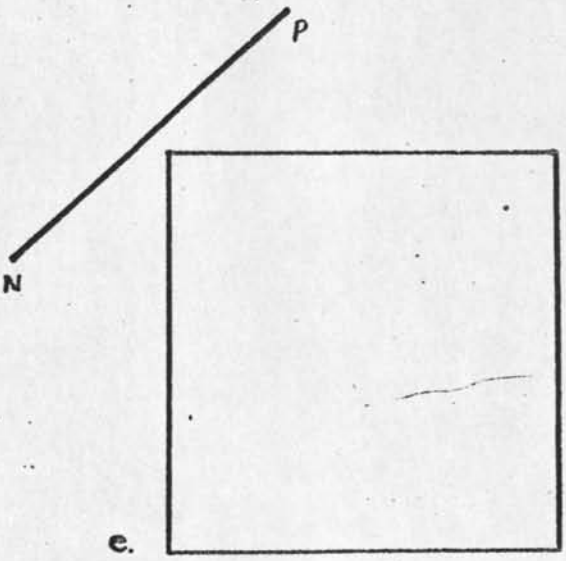
b.



c.



d.



e.

FIG. 3

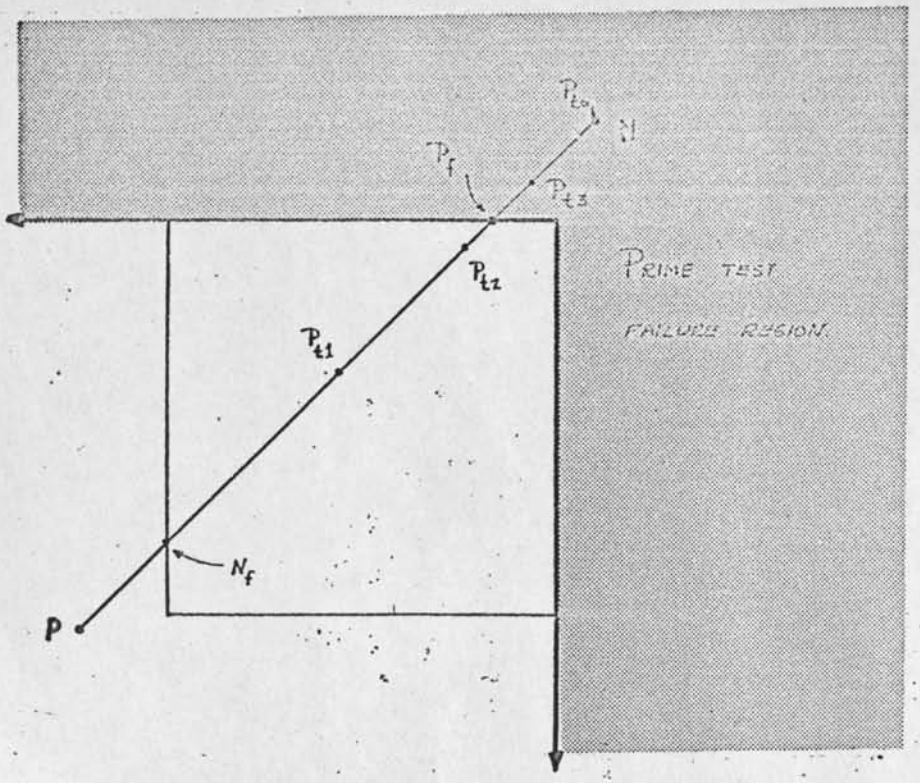

P_{t_0}
N
P_f
P_{t_3}
P_{t_2}
P_{t_1}
PRIME TEST
FAILURE REGION
N_f
P

FIG. 4

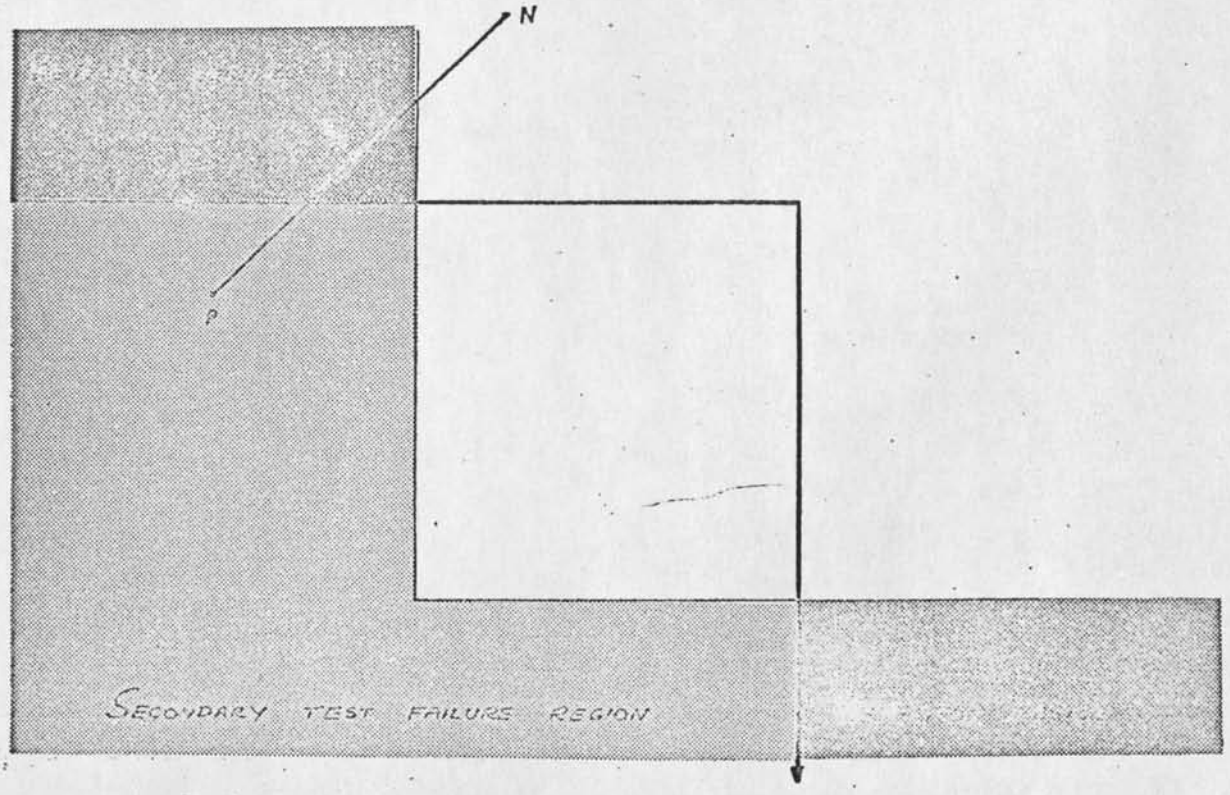

N
P
SECONDARY TEST FAILURE REGION

FIG. 5

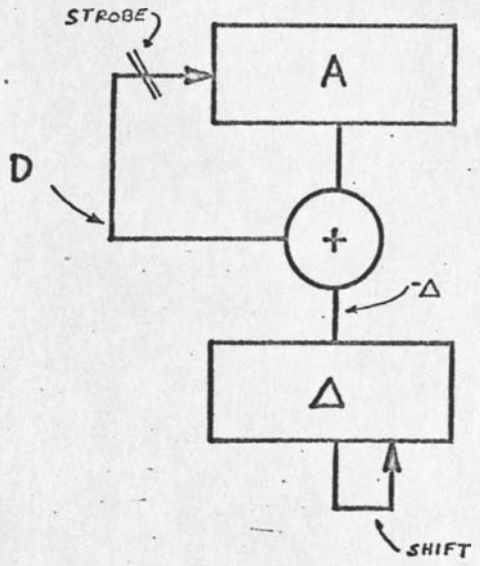


FIG. 6.

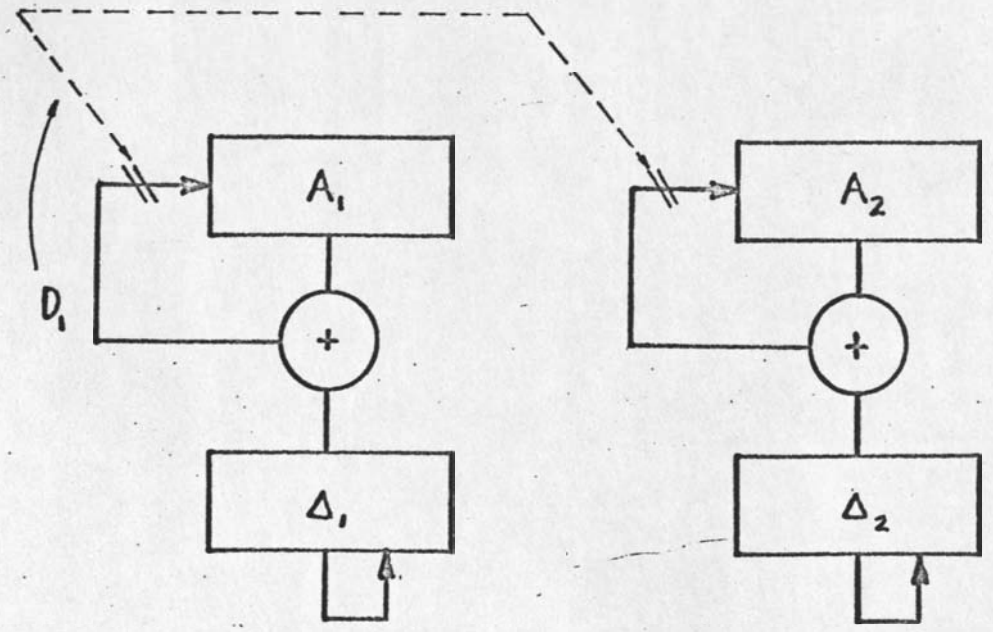


FIG. 7.

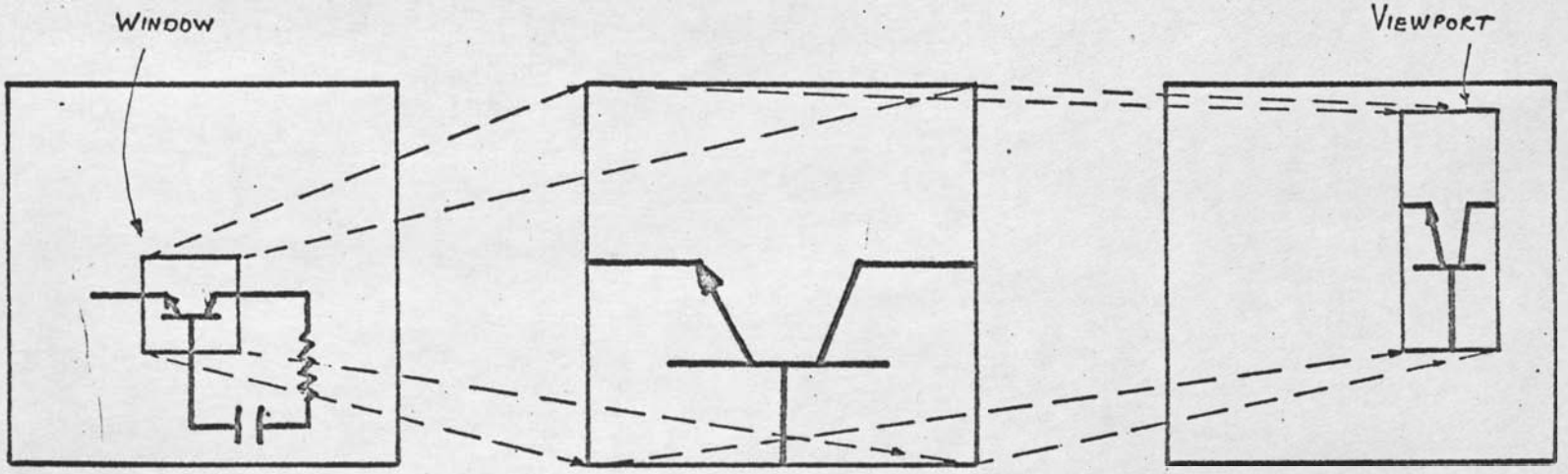


FIG. 8a.

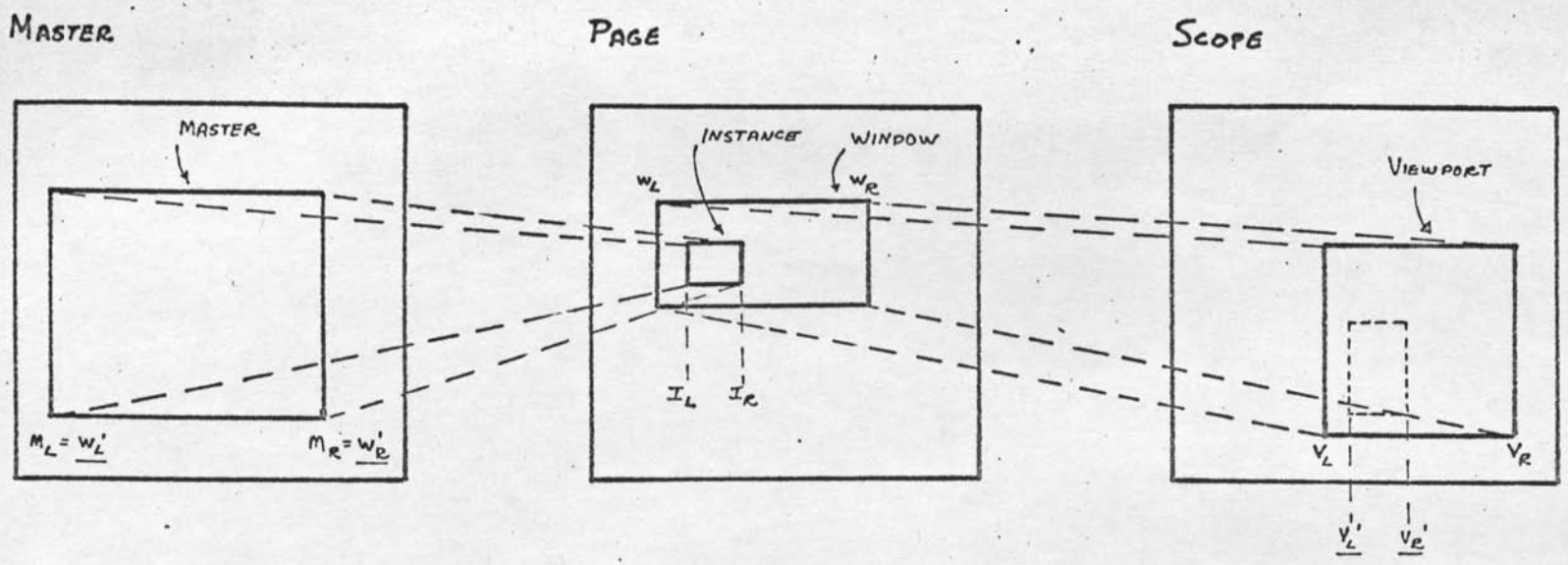

MASTER
PAGE
SCOPE
MASTER
INSTANCE
WINDOW
VIEWPORT
W_L
W_R
I_L
I_R
$M_L = W_L'$
$M_R = W_R'$
V_L
V_R
V_L'
V_R'

FIG. 8b.

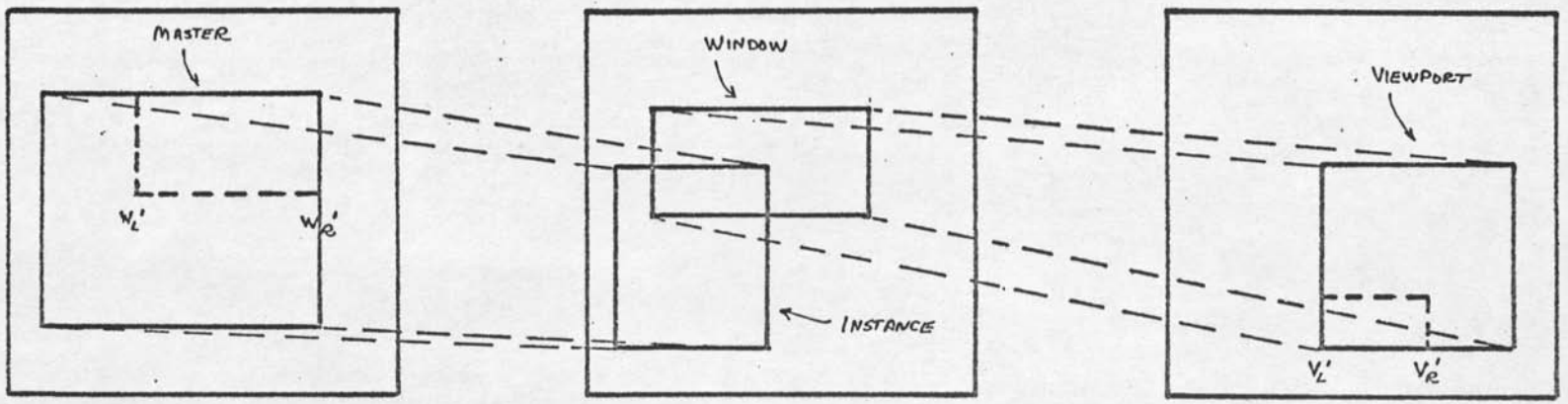

MASTER
WINDOW
VIEWPORT
W_L'
W_R'
INSTANCE
V_L'
V_R'

FIG. 9.

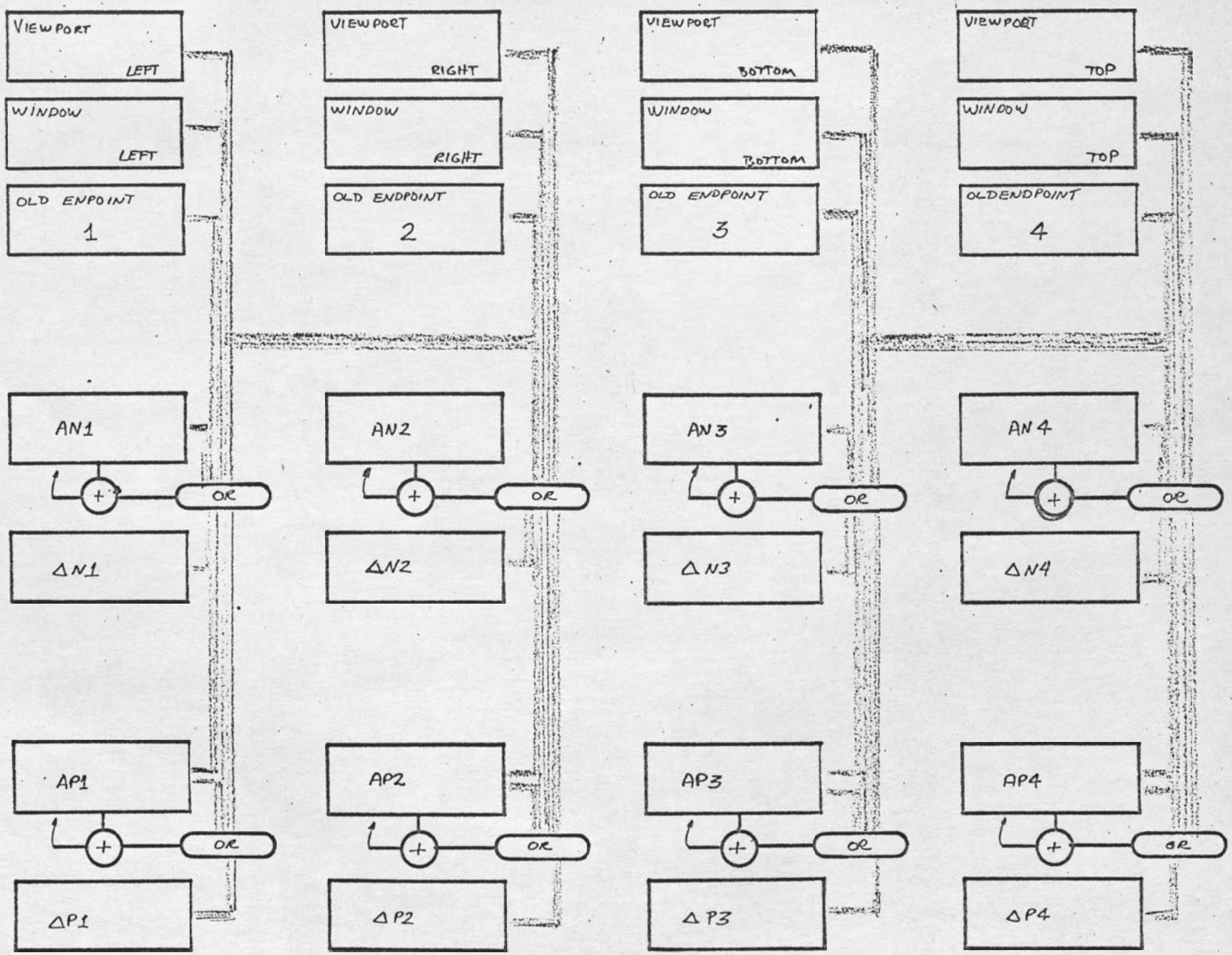

VIEWPORT
LEFT
WINDOW
LEFT
OLD ENPOINT
1
VIEWPORT
RIGHT
WINDOW
RIGHT
OLD ENDPOINT
2
VIEWPORT
BOTTOM
WINDOW
BOTTOM
OLD ENDPOINT
3
VIEWPORT
TOP
WINDOW
TOP
OLDENDPOINT
4
AN1
AN2
AN3
AN4
OR
ΔN1
ΔN2
ΔN3
ΔN4
AP1
AP2
AP3
AP4
ΔP1
ΔP2
ΔP3
ΔP4

FIG. 10.

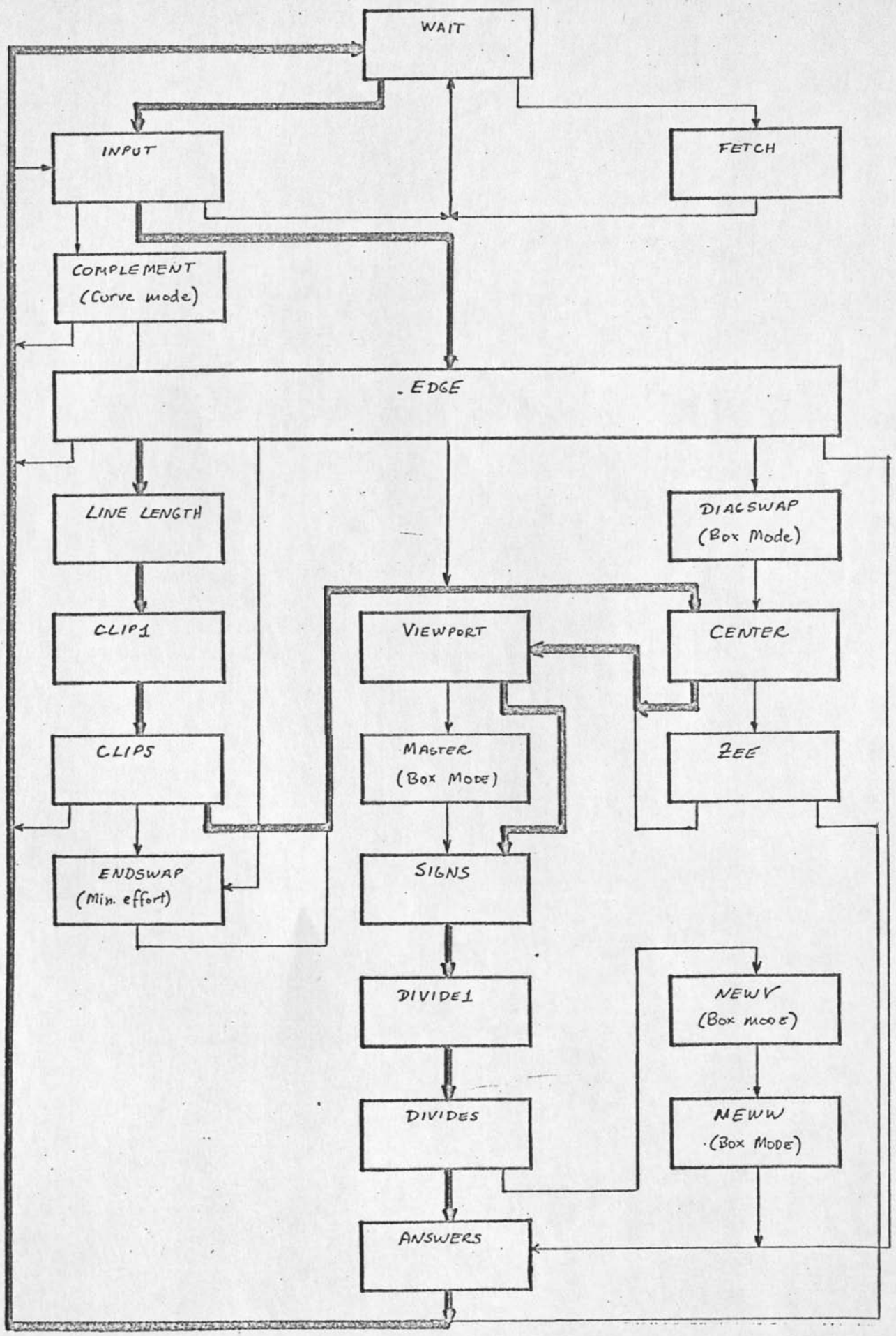

WAIT
INPUT
FETCH
COMPLEMENT
(Curve mode)
EDGE
LINE LENGTH
DIAGSWAP
(Box Mode)
CLIP1
VIEWPORT
CENTER
CLIP2
MASTER
(Box Mode)
ZEE
ENDSWAP
(Min. effort)
SIGNS
DIVIDE1
NEWV
(Box mode)
DIVIDE2
NEWW
(Box Mode)
ANSWERS

FIG. 11.

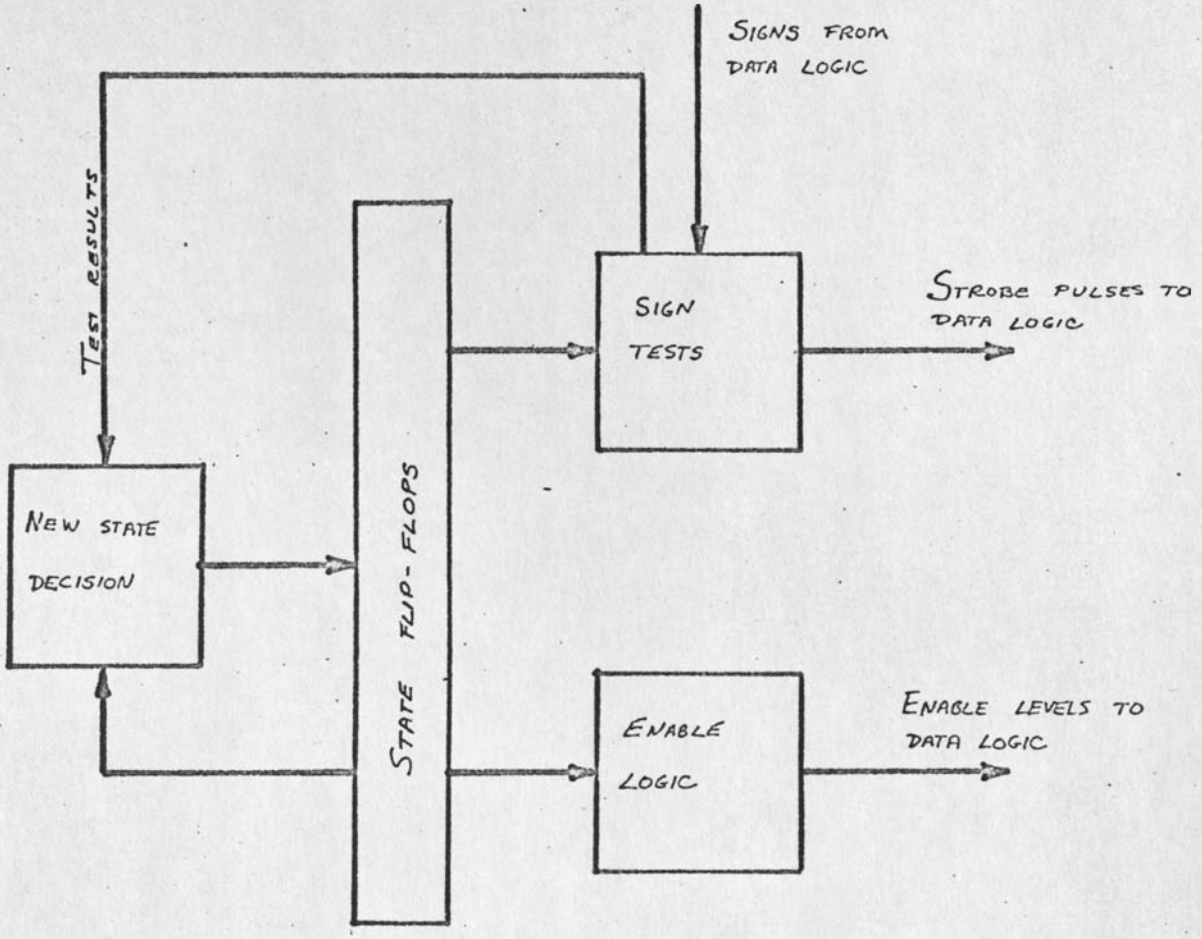

SIGNS FROM DATA LOGIC
TEST RESULTS
SIGN TESTS
STROBE PULSES TO DATA LOGIC
NEW STATE DECISION
STATE FLIP-FLOPS
ENABLE LOGIC
ENABLE LEVELS TO DATA LOGIC

FIG. 12.

APPENDIX 2

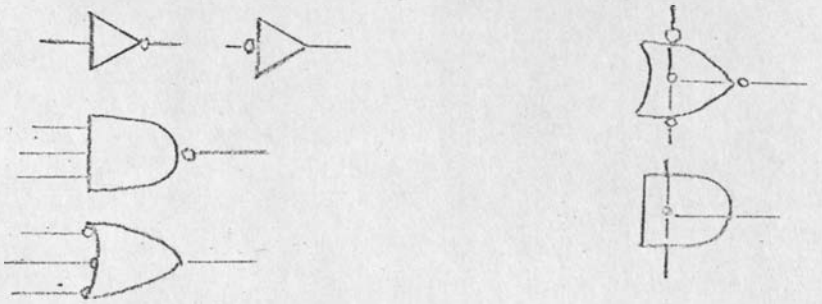
Prints for the Matrix Multiplier

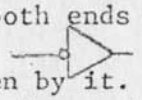
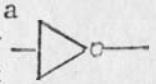
DRAWING CONVENTIONS - 3-D DISPLAY

The six block schematic drawings describing the matrix multiplier and the clipper box are drawn with the following conventions.

1. Labeled wires. If the condition indicated by the label is true, the wire will be in the "high" (+5) state. This is true regardless of the circles, etc. that may be included on the wire. Occasionally in the matrix multiplier the subscript L is used to designate a true low level; e.g., FOO(L) means FOO is true when the wire is low.

2. There are only 6 legal logic symbols:



3. Circles may be sprinkled at the ends of wires to suit the artist. In general, one should try to circle both ends of a wire if you circle either. Thus the choice of  or  depends on the logic driving it and being driven by it. It is not always possible to put circles at both ends of wires.

4. Wires which go to other drawings will end in large circles and be given a label. Such wires may go to more that one other drawing. Not all such wires follow this convention, but the context of those which do not should make the meaning clear.

5. Wire which go elsewhere on this drawing will normally end in an arrow, generally pointing to where the other end of the wire is.

6. Nomenclature. The bits of registers are numbered with the PDP-1 convention, that is with the most significant bit as subscript 0, the least significant bit as subscript 17, generally. Because some registeres are extended further to the left, additional left bits are labeled subscript e1, e2, etc. Additional bits to the right of the register are labeled subscript 18, 19, etc.

Matrix Multiplier

Introduction

The matrix multiplier is a digital multiplier that will multiply a four-element vector by the sixteen terms of a four by four matrix. Each 18 binary bit element of the input vector is presented in turn, and four resulting products are accumulated to provide the four-element result vector. This result vector is then read out term by term. Although an inferfacing control unit could be constructed to work with any memory unit, the one currently used at Harvard interfaces the PDP-1 computer. It uses the normal I/O buss but a faster interface and control unit is contemplated. There follows first a discussion of the general hardware capability, and then the programming necessary to accomplish this.

Number Representation in the Matrix Multiplier

The number representation implemented is exactly that of the currently used PDP-1. The Identity Matrix will be represented by setting diagonal elements to 377777. This causes the loss of one bit of precision, exactly the same loss as would be incurred by using 200000 and then scaling up by one.

The matrix product register is capable of accumulating 20 bit sums, which represents numbers in the range $-4<x<4$. The control is such that registers report only the sign and the fractional part of the results back to the PDP-1. "Overflow" is defined as the condition in which any matrix product register contains a number with a non zero integer part.

When the matrix box is used for simple matrix multiplication overflow is reported to the PDP-1. Overflow reports only the final size of the accumulated matrix product. Thus, the product of

$$[.8 \quad .8 \quad .8 \quad .8] \begin{bmatrix} .8 \\ .8 \\ -.8 \\ -.8 \end{bmatrix}$$

will not produce an overflow indication even though the intermediate sums may be greater than 1. If overflow occurs, the programmer should scale down one of the matrices he is multiplying and try again.

When the matrix box is used for display, the control passes all 20 bits of the matrix product register to the clipper.

Normalization in the Matrix Multiplier

The registers which store the matrix are capable of shifting to the left. Using this capability, it is possible to normalize the entire 4x4 matrix simultaneously. A normalize instruction will cause the magnitude parts of the sixteen registers to be shifted left until the largest of them exceeds a certain number.

Normalization serves the purpose of making more significance available to the output of the multiplication. The components of the vector being multiplied may be of any value. In particular, they may be as large as 1. Thus the maximum value in the matrix should be less than 1/4 if the output sum is to be less than 1.

Two extra bits are provided in the matrix product register. Thus overflow is not a problem when the output is not being returned to the computer. The matrix elements should be normalized to the largest value less than 1.

There are, therfore, two conflicting requirements for normalization. First, and most important, is the ability to shift left until the three most significant bits of some register are 001 (or 110). This makes the largest value lie in the range $1/8 \leq x < 1/4$. Second, another instruction shift left until the most significant bit of some register is 1. This places the values in the range $1/2 \leq x < 1$.

Programming the Matrix Multiplier

The matrix multiplier (m. m.) and clipper are very simply interfaced to the PDP-1. The PDP-1 IOT instruction provides the directive, and the data is provided on the PDP-1's IO buss. Normally the m.m. AC output is hooked to the clipper input. The PDP-1 can monitor this link, either reading m.m. output, or setting the clipper input.

Both the clipper and the matrix multiplier require a 6 bit directive (control instruction) and an input data word (18 bits for the matrix multiplier and 36 bits for the clipper). The interface contains three registers to hold the data and directive (one 6 bit and two 18 bit). Both pieces of equipment contain an inflag and an outflag. The inflag announces to the equipment that there is data and a directive on the line for processing. The inflag is cleared by the equipment upon taking the next data and directive. The outflag announces that the output buss or AC's of the equipment contains data to be unloaded. Neither piece of equipment will undertake further operations until its outflag

has been cleared. These five status signals correspond to the utility bits listed below:

Utility Bit	Status
0	m.m. outflag
1	m.m. inflag
2	clipper outflag
3	clipper inflag
4	m.m. ready flag (Used to determine ready state when m.m. outflag not requested.)

The matrix multiplier (m.m.) six bit directive is coded as follows:

Bit	0	1	2	3	4	5
Load	1	0	(row)		(col)	
Unload	0	0	(row)		(col)	
Unload AC	0	1	0	0	(col)	
Normalize	0	1	0	1	x	mode
Self Multiply	0	1	1	0	of1	of2
Multiply	1	1	(row)		of	clac

Reference to Row and Column refer to the internal storage of the m.m. as below:

	Col 00	Col 01	Col 10	Col 11
Row 00	A	B	C	D
Row 01	E	F	G	H
Row 10	I	J	K	L
Row 11	M	N	O	P

Load will put the input data into the m.m. at the position specified by bits 2-5. Unload retrieves the number specified and places it on the output buss. Normalize mode = 0 shifts each element of the matrix left until any element meets the requirement that bit 0 and bit 1 differ.

Normalize mode = 1, shifts each element left until bits 0, 1, 2 differ from bit 3. If bit 3 is the high order significant bit, then no overflow can occur on a multiply. Multiply takes the input data number and multiplies that number by each element of the row specified and adds the product of each multiplication to the associated column AC. If bit 5 is set, the AC is cleared before multiplication. If bit 4 is set, the outflag is raised at the end of the multiply.

For example, if the the directive 64_8 were given with the number on the input lines and the matrix loaded into A-P as diagrammed above, the result of the operation would be:

Register:	AC00	AC01	AC10	AC11
Contents:	(AC00)+Z*E	(AC01)+Z*F	(AC10)+Z*G	(AC11)+Z*H

The symbol * represents scalar multiplication, + normal scalar addition, and (ACij) denotes the contents of the particular AC before the operation.

Self multiply needs no data to operate. It is used to automatically generate points in three-space on curves represented parametrically in homogeneous coordinates as $[t^3\ t^2\ t\ 1]$. The self multiply consists of a set of operations like the following: Take row 00 of each column, dropping the low order bit, adding the contents of row 01 shifted right five places, and returning the results back to row 00. The rest of the set consists of repeating this process for rows 01 and 10. If oF1 is set, the m.m. will raise the outflag when the AC contains row 00 with the low order bit dropped. If oF2 is set, the m.m. will raise the outflag just before replacing row 00.

A typical sequence of commands to multiply a 1x4 vector by a 4x4 matrix that had already been loaded into the m.m. would be:

Commands	Commands
Put X onto input lines	
Load directive with 61_8	Clears AC, multiplies input lines by row 00, and accumulates in AC.
Put Y/ onto input lines	
Load directive with 64_8	Multiplies input lines by row 01, and accumulates in AC.
Put Z onto input lines	
Load directive with 70_8	Multiplies input lines by row 10, and accumulates in AC.

Commands	Commands
Put W onto input lines	
Load directive with 76_8	Multiply input lines by row 11, accumulate results in AC, set output flag.

At this point the four AC's can be unloaded to retrieve results.

Matrix Multiplier

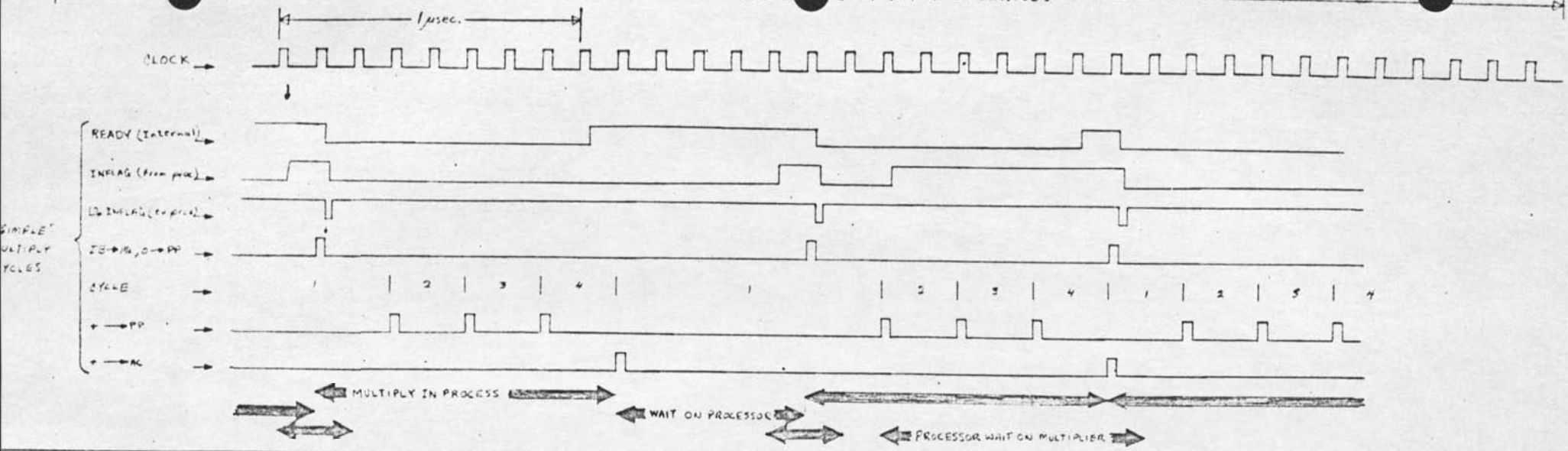
Print No.	Date	Title
	7/14/68	multiplier control
3D-1	7/14/68	structural diagram
3D-2	7/14/68	matrix multiplier - dwg2
3D-3	7/14/68	matrix multiplier - dwg3
3D-4	7/14/68	matrix multiplier - dwg4
3D-5	7/14/68	matrix multiplier - dwg5
3D-6	7/14/68	matrix multiplier - dwg6
3D-7	7/14/68	matrix multiplier - dwg7
3D-8	7/14/68	matrix multiplier - dwg8
3D-9	7/14/68	matrix multiplier - dwg9
3D-10	7/14/68	matrix multiplier - dwg 10
3D-11	7/14/68	matrix multiplier - dwg11
3D-12	7/14/68	matrix multiplier - dwg12
3D-13	7/14/68	matrix multiplier - dwg13
3D-14	7/14/68	matrix multiplier - dwg14
3D-15	7/14/68	matrix multiplier - dwg15
3D-16	7/14/68	matrix multiplier - dwg16
3D-17	7/14/68	matrix multiplier - dwg17
3D-18	7/14/68	matrix multiplier - dwg18
3D-19	7/14/68	matrix multiplier - dwg19
3D-20	7/14/68	matrix multiplier - dwg20
3D-21	7/14/68	matrix multiplier - dwg21

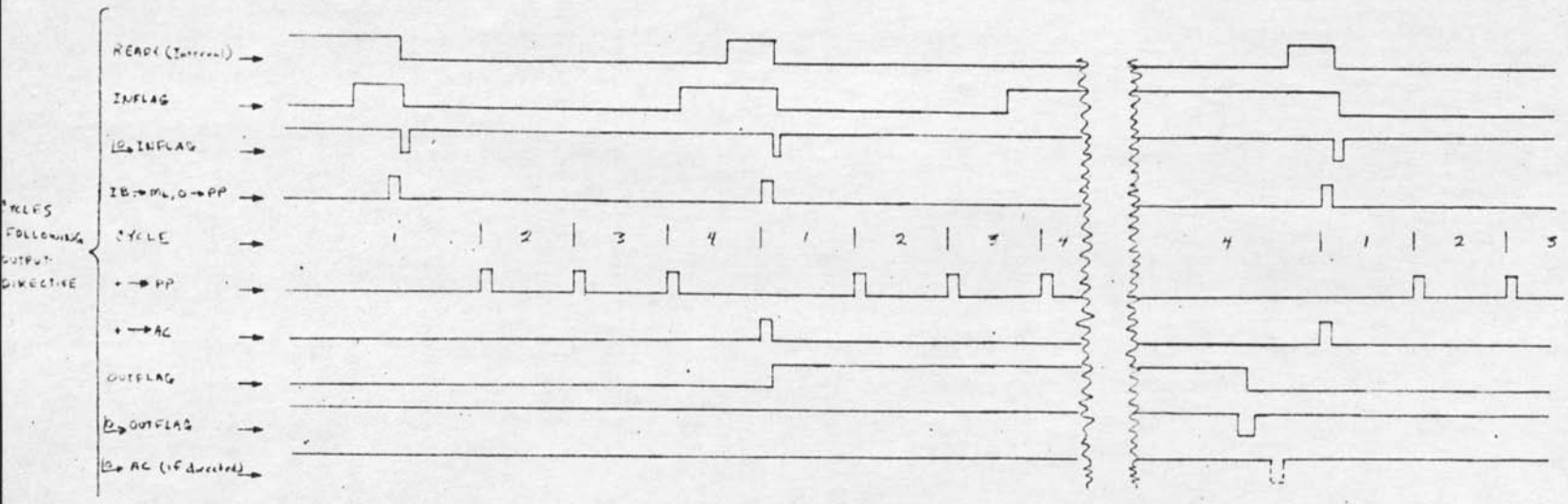
Matrix Multiplier

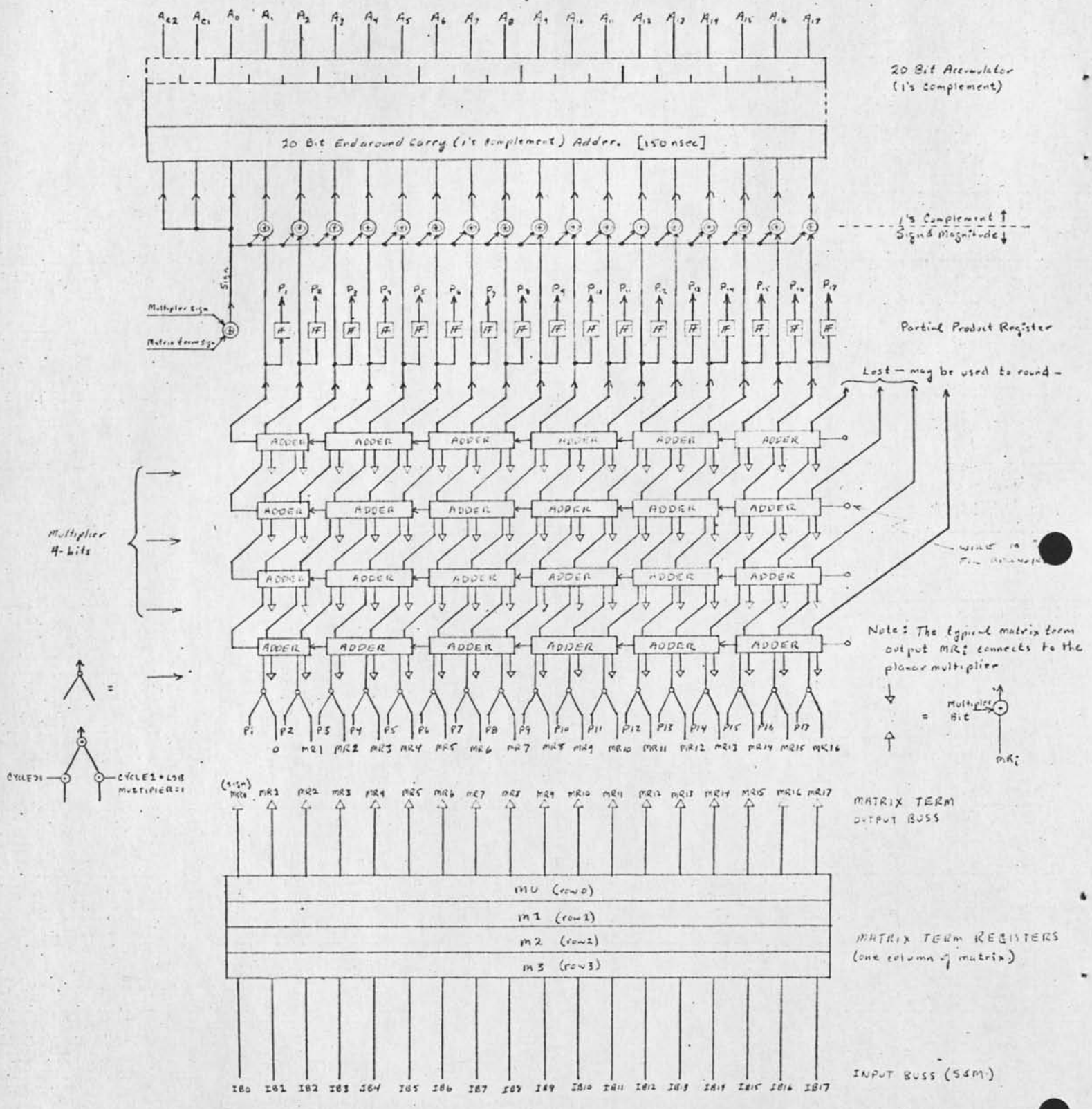
3D-22	7/14/68	matrix multiplier - dwg22
3D-23	7/14/68	master clock
3D-24	7/14/68	multipler control - dwg No.1
3D-24	7/14/68	fast 3-bit adder
3D-25	7/14/68	multiplier control - dwg No.2
3D-26	7/14/68	multiplier control - dwg No. 3
3D-27	7/14/68	multiplier control - dwg No.4
3D-28	7/14/68	multiplier control - dwg No.5
3D-29	7/14/68	multiplier control - dwg No.6
3D-30	7/14/68	multiplier control - dwg No.7
3D-31	7/14/68	multiplier control - dwg No. 8
3D-32	7/14/68	multiplier control - dwg 9
3D-33	7/14/68	multiplier control dwg 10
3D-34	7/14/68	multiplier control - dwg 11
3D-35	7/14/68	multiplier control - dwg 12
3D-36	7/14/68	multiplier control - dwg 13
3D-37	7/14/68	multiplier control - dwg 14
3D-38	7/14/68	multiplier control - dwg 15
3D-39	7/14/68	multiplier control - dwg 16
3D-40	7/14/68	multiplier control - dwg
3D-41	7/14/68	multiplier control - dwg
3D-42	7/14/68	output buss and lights
3D-43	7/14/68	light drivers
3D-44	7/14/68	adder zero detection
3D-45		timing

Slot	E	C	D	P (POWER SUPPLY CAGE)
1	Q50	CT801	GT804	GT809
2	CT502	CT501	GD801	GT809
3	CT802	CT505	GD801	GT809
4	CT805	GT801	GL801	CL804
5		GT801	GD806	
6		GT805	GD806	
7		GT801	GL810	
8		GT804	CL806	
9		CT809	GD801	
10		GT810	CL801	
11	GD804	GT810	GD801	
12	GT809	FT801	GT804	
13		GT813	GD501	
14		GT810	GT805	
15		GT801	GT805	
16		GT804	GT802	
17		GT809	GT804	
18		KT801	CL801	
19	L54	GT805	GD801	
20	L54	GD810	GD801	
21	L54	FT801	GL806	
22	GT804	KT801	GD806	
23	CT902	KT801	CL810	
24	CL806.	KT801	CL806	
25	GD505	CT813	GD801	
26	GD801	GT809	GD801	
27	CL801	FT801	GD801	
28	CL801	CT804	GT804	

HARVARD UNIVERSITY
3-D DISPLAY
MULTIPLIER CONTROL
REVISED 14 JULY 1968

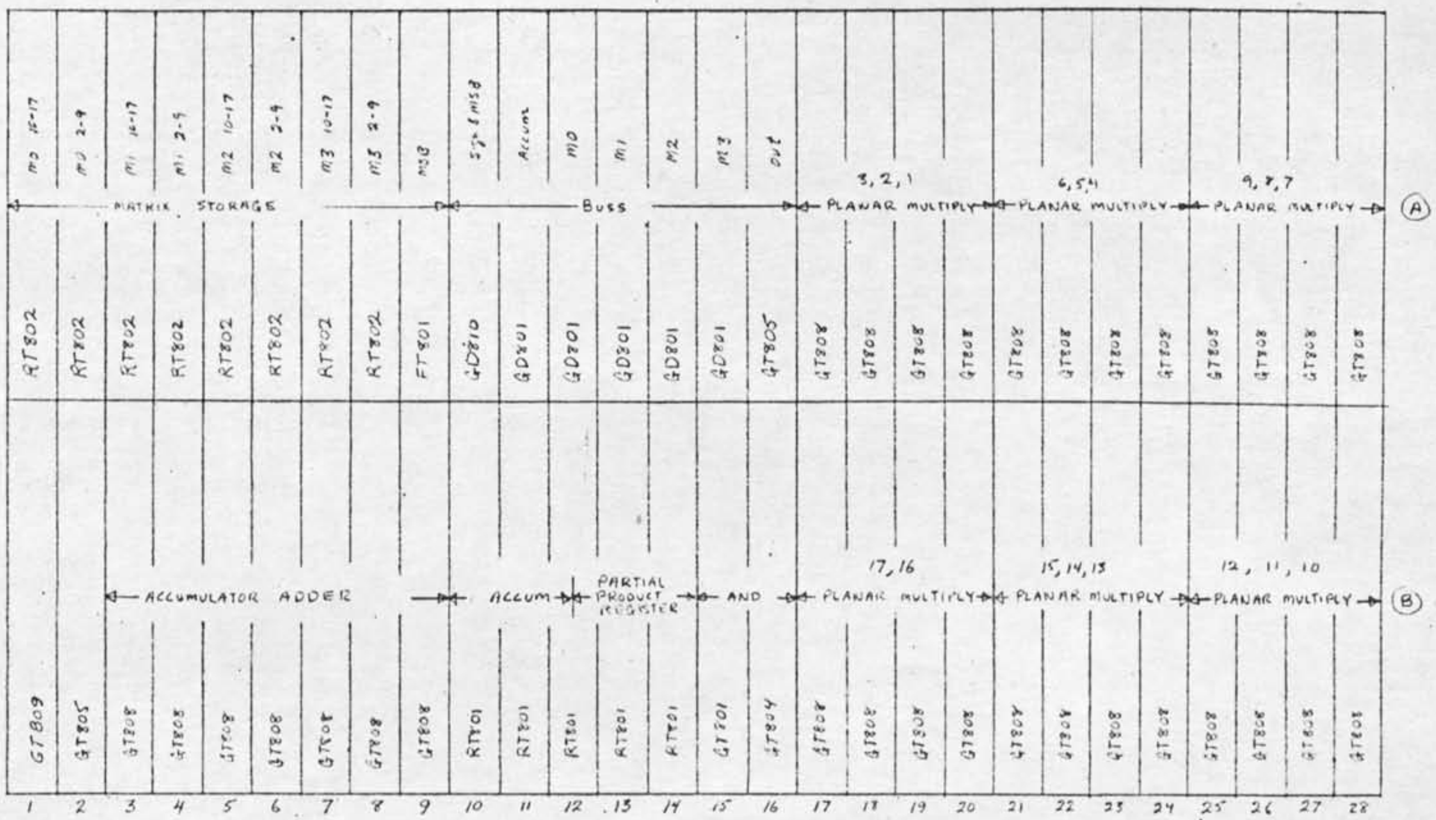

1 μsec.
CLOCK
READY (Internal)
INFLAG (From PDP)
INFLAG (to PDP)
IB→MB, 0→PP
CYCLE
1 2 3 4 1 2 3 4 1 2 3 4
+ →PP
+ →AC
MULTIPLY IN PROCESS
WAIT ON PROCESSOR
PROCESSOR WAIT ON MULTIPLIER


READY (Internal)
INFLAG
INFLAG
IB→MB, 0→PP
CYCLE
1 2 3 4 1 2 3 4 4 1 2 3
+ →PP
+ →AC
OUTFLAG
OUTFLAG
AC (if desired)



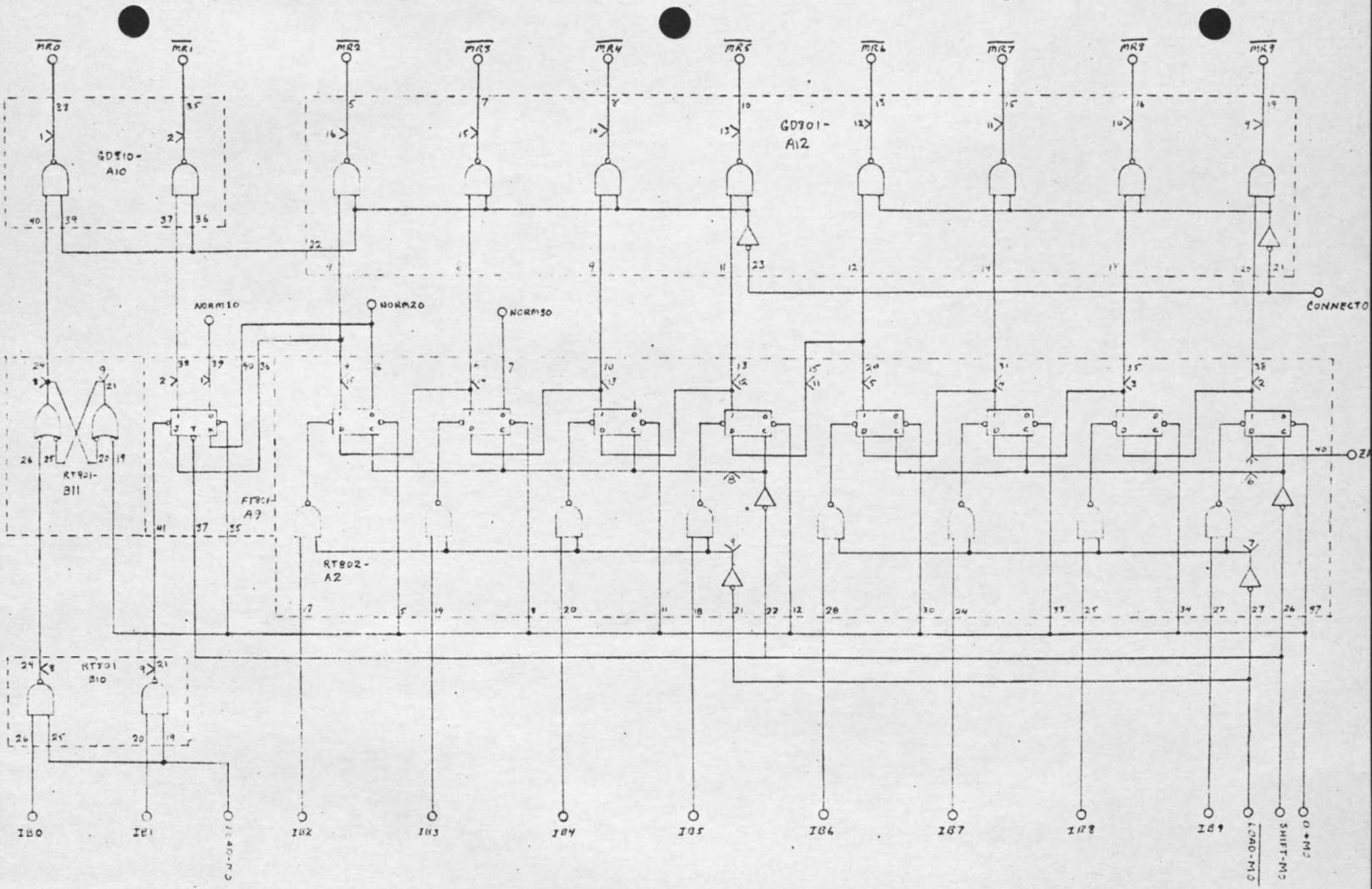
REVISED 14 JULY 1968

MATRIX MULTIPLIER
PHYSICAL LAYOUT
(CARD SIDE)



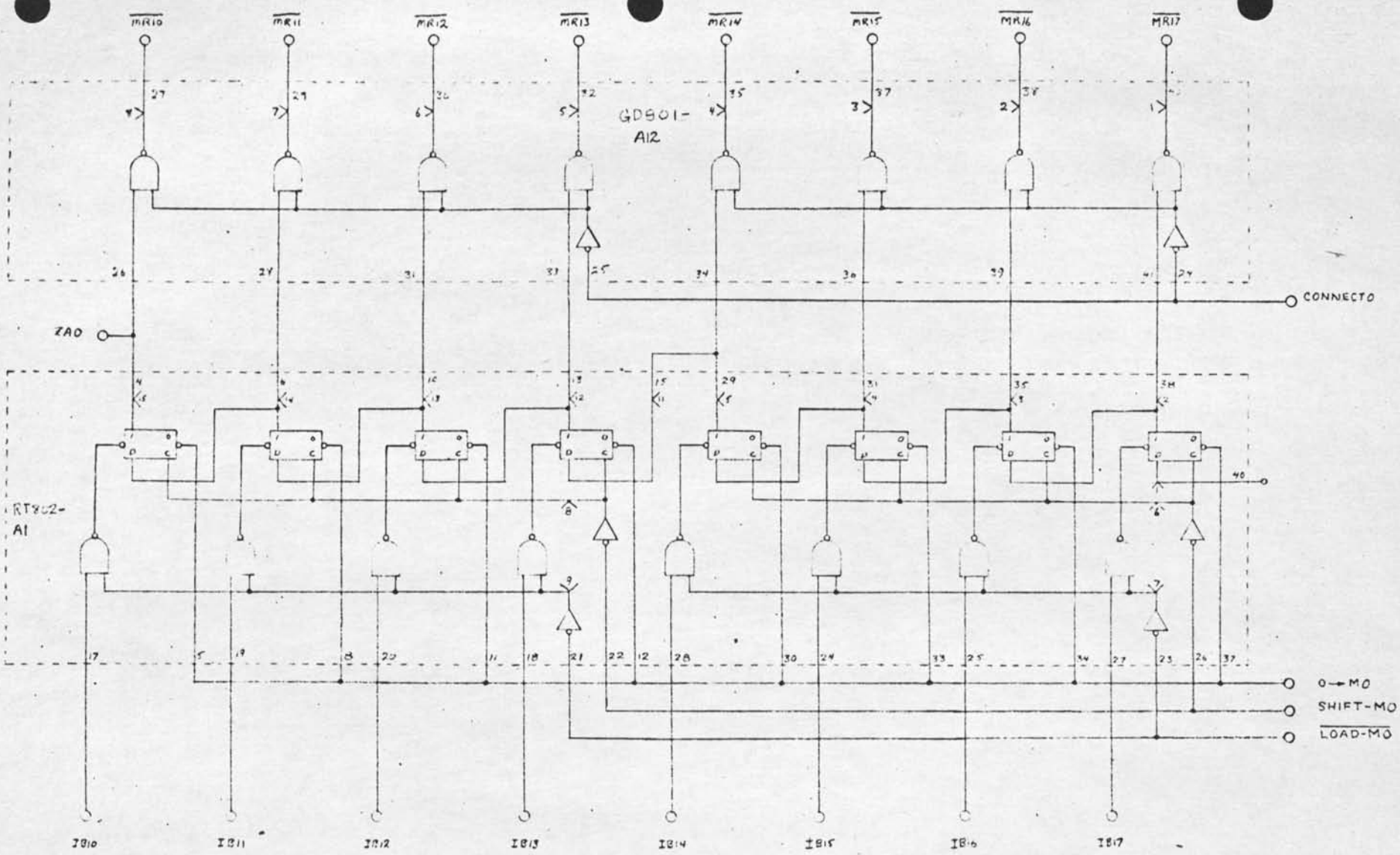
HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 2

30-2

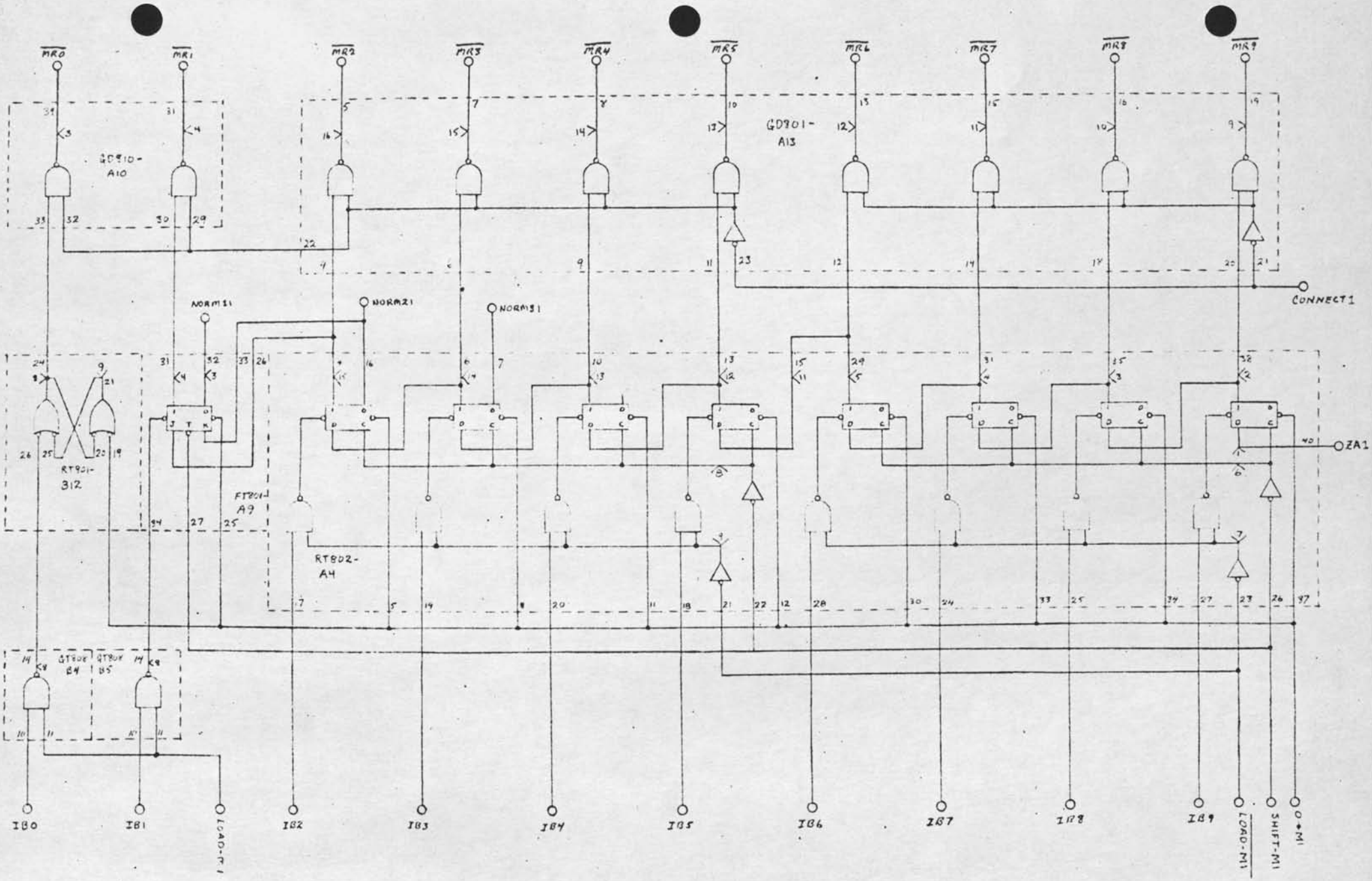


HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 3

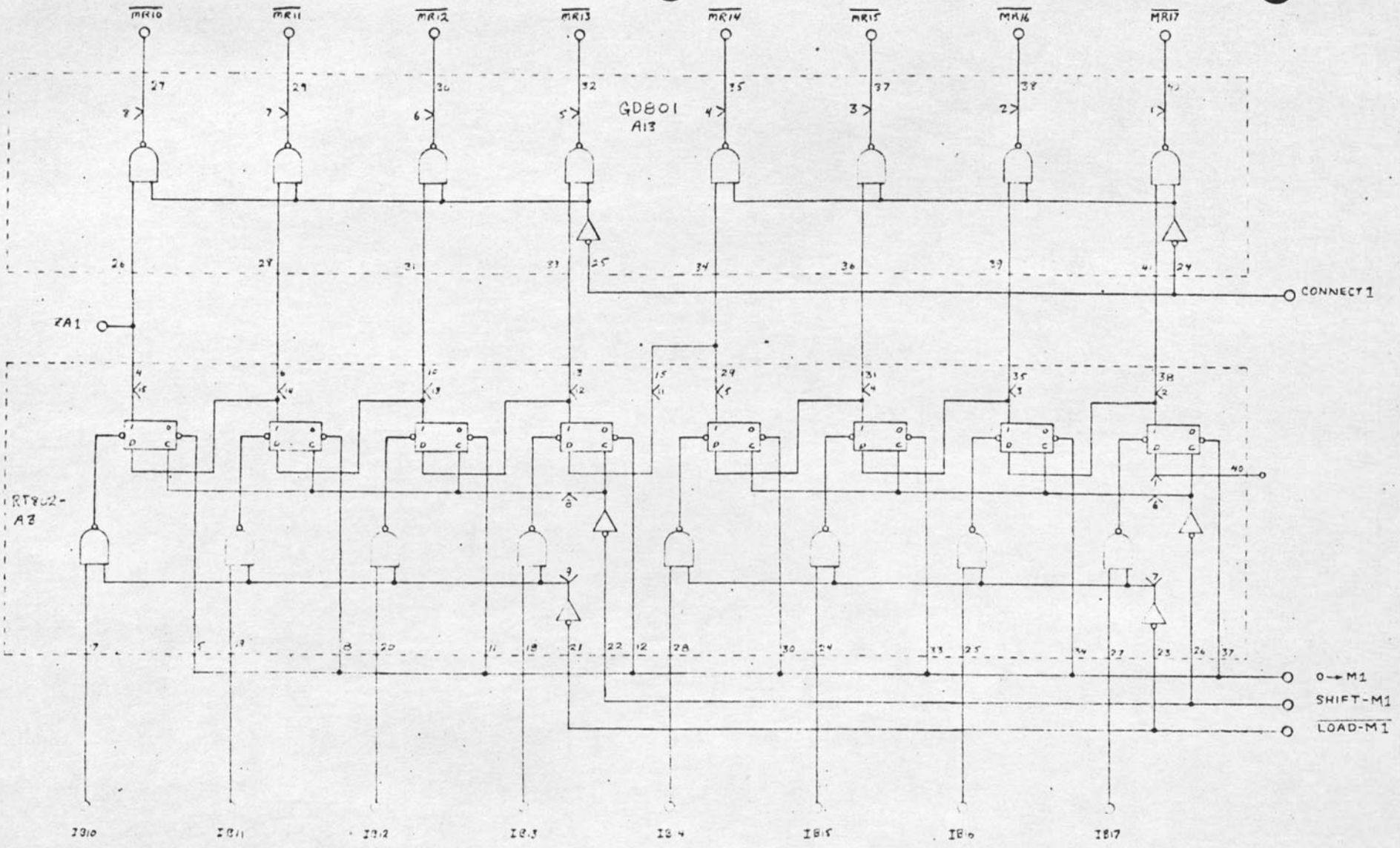
3D-3



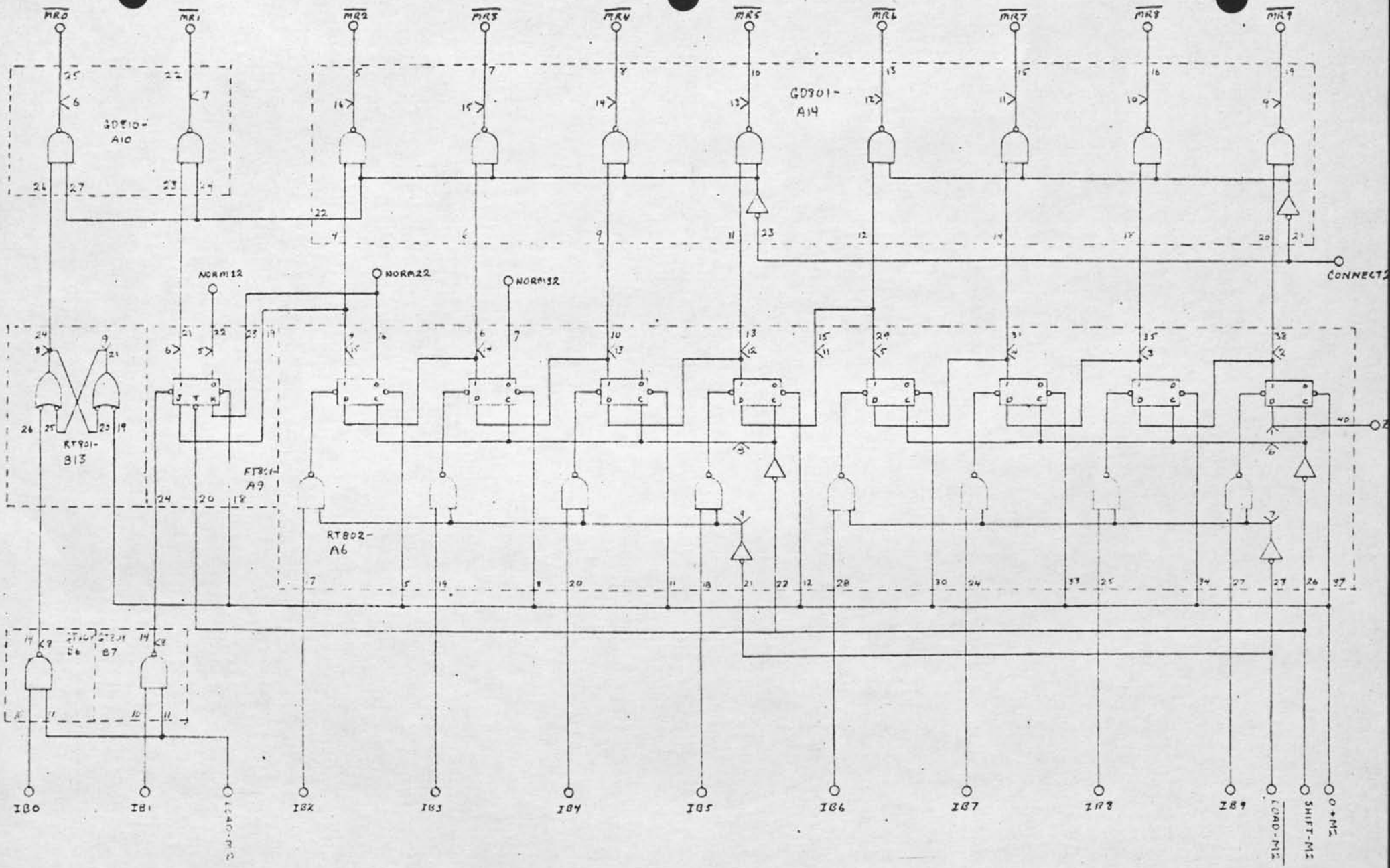
HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 4



HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG #
3D-5

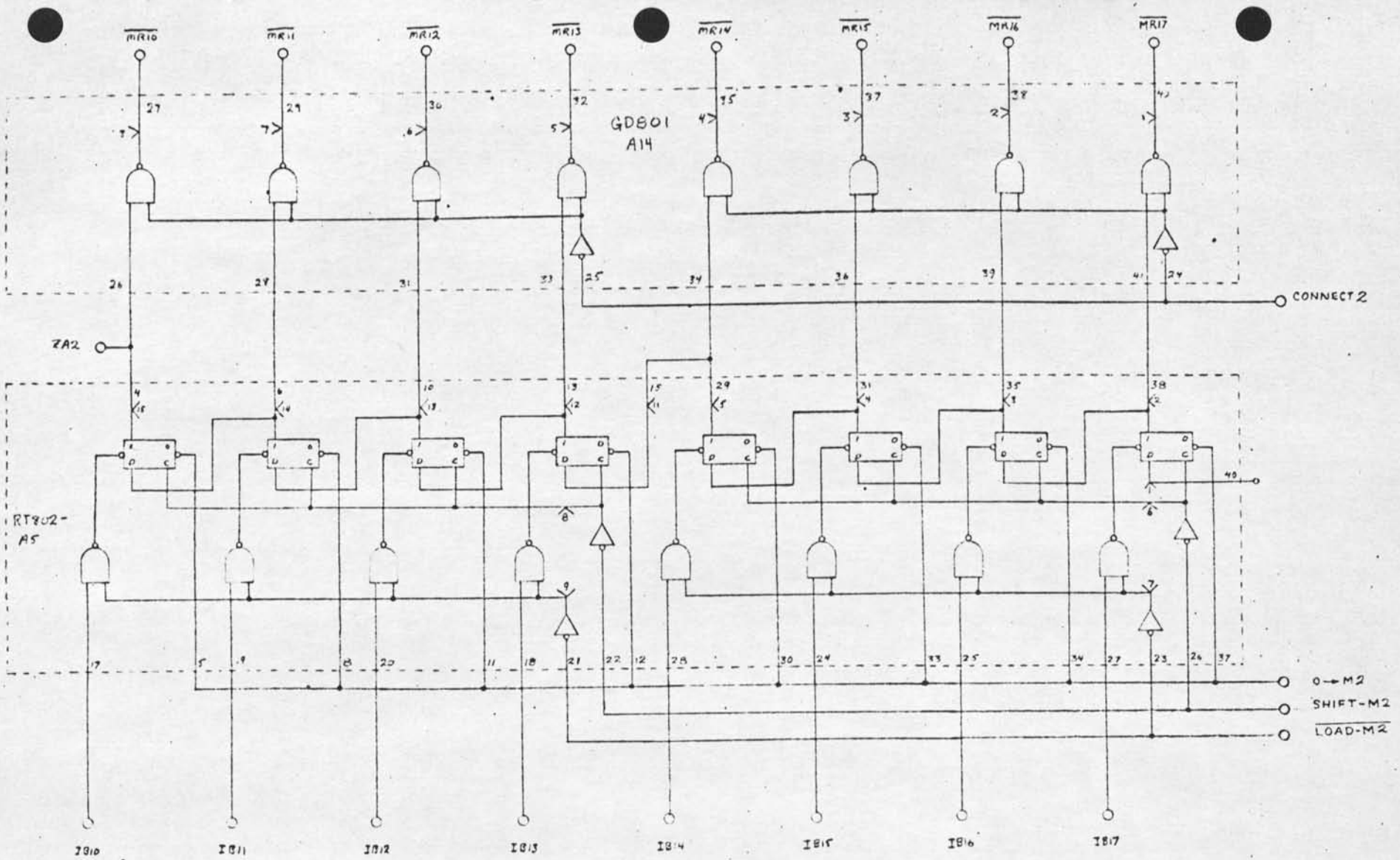


HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 6

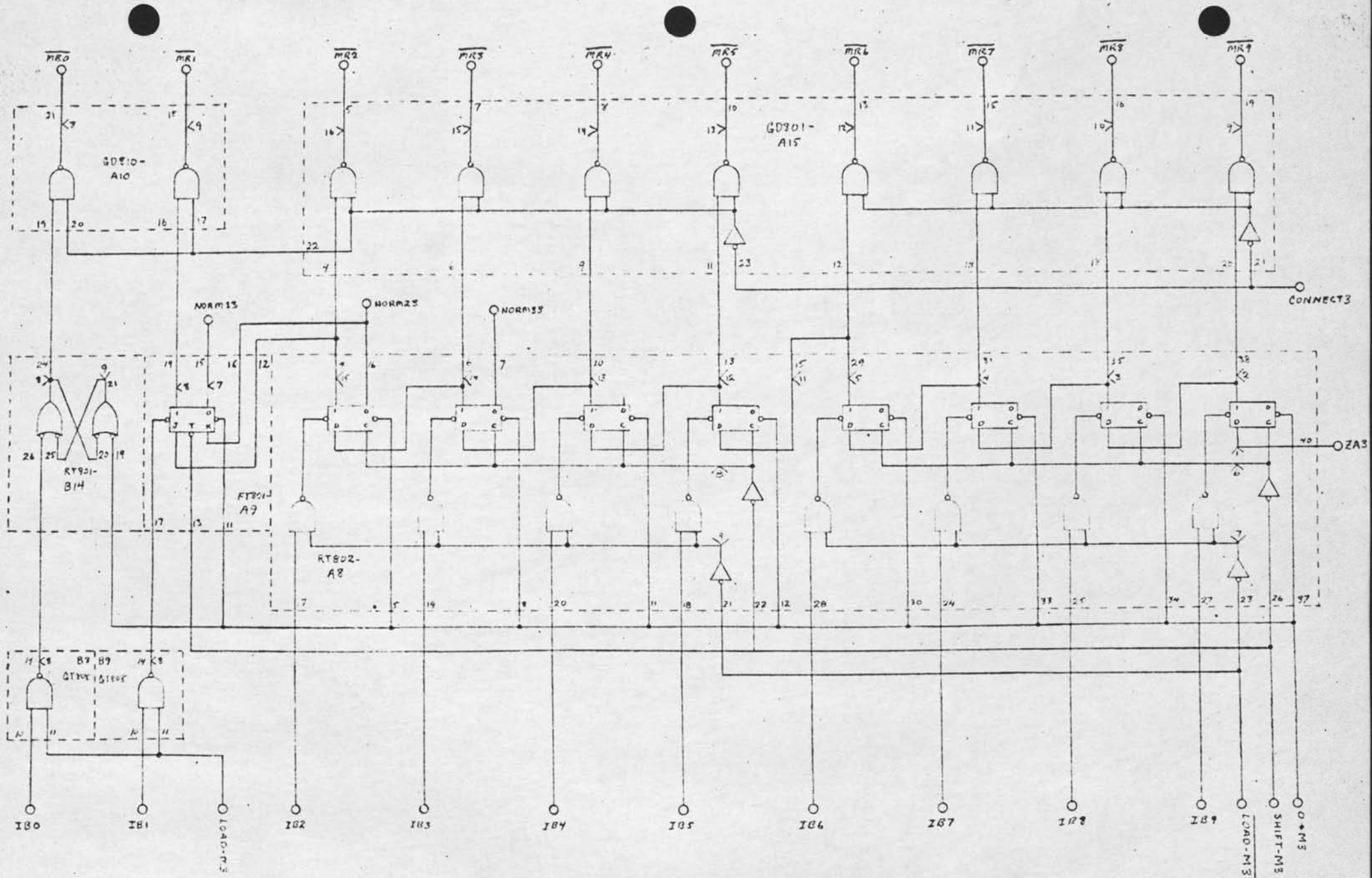


HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG7

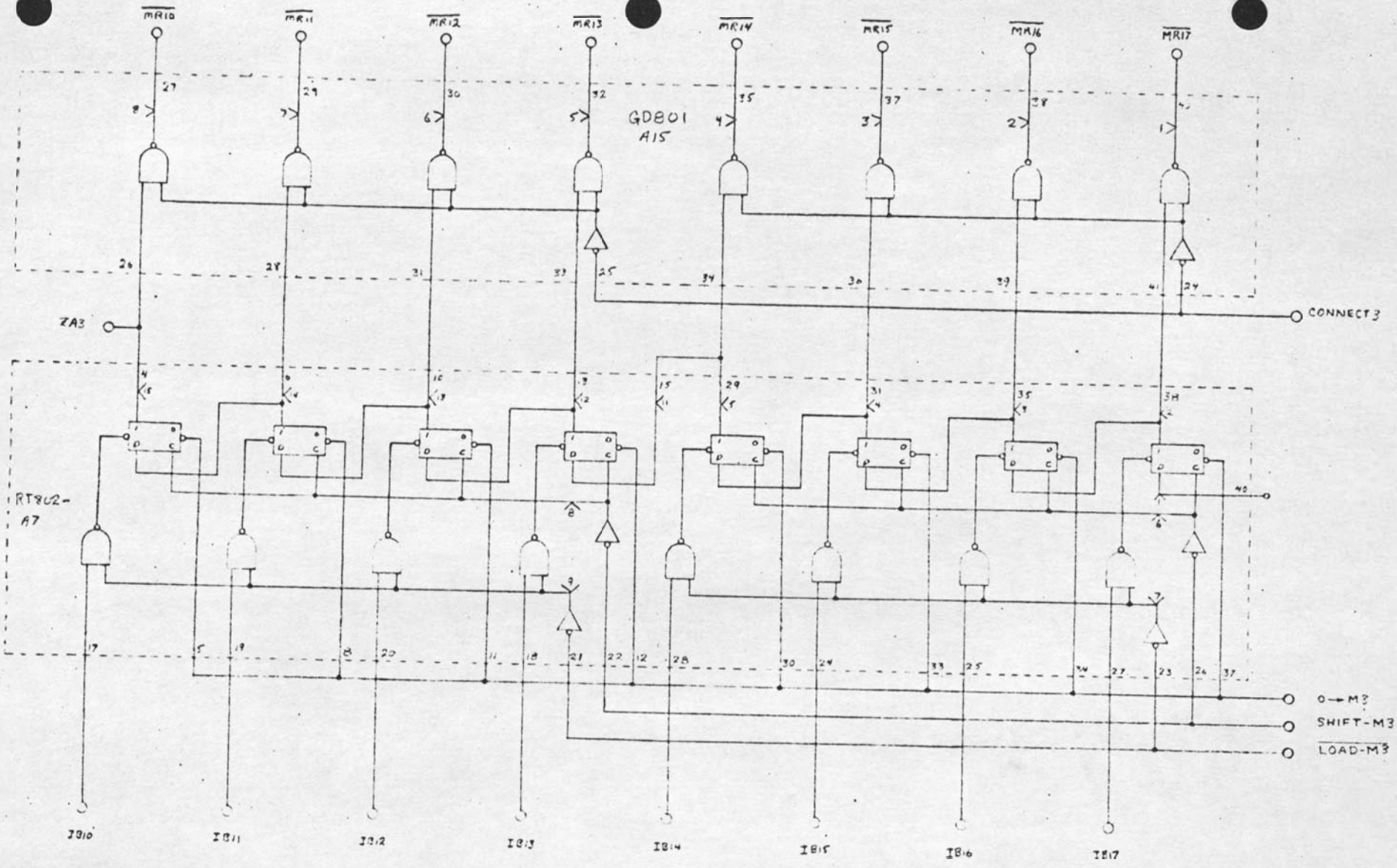
3D-7



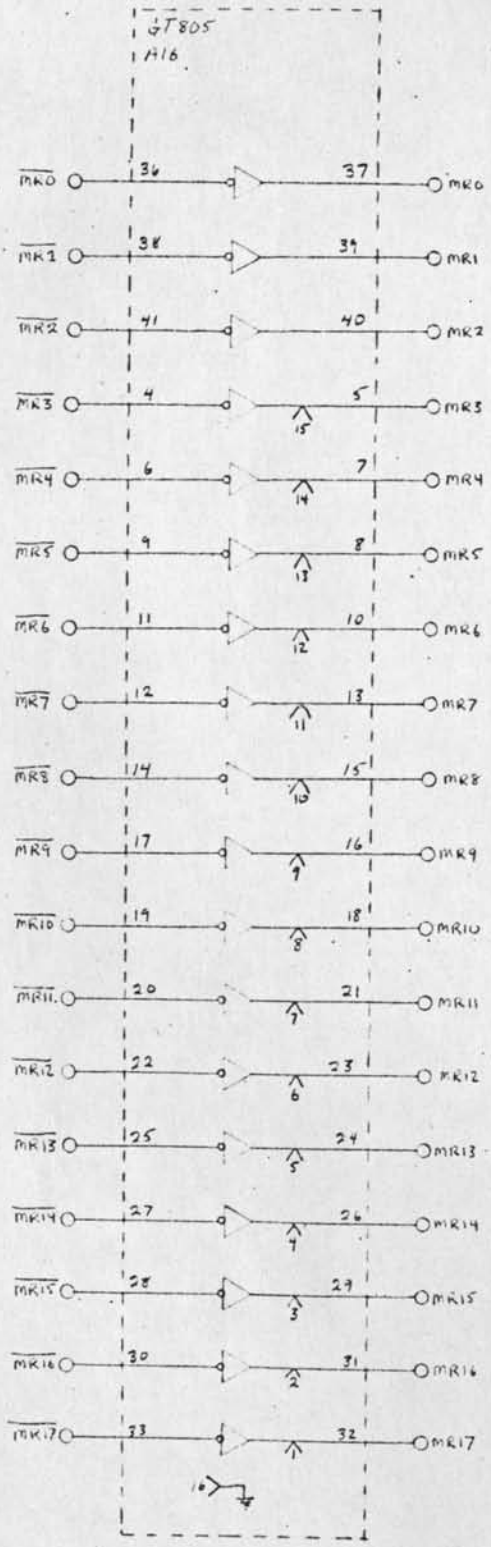
HARVARD UNIVERSITY
[illegible]
MATRIX MULTIPLIER
DWG 8



HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG
3D-

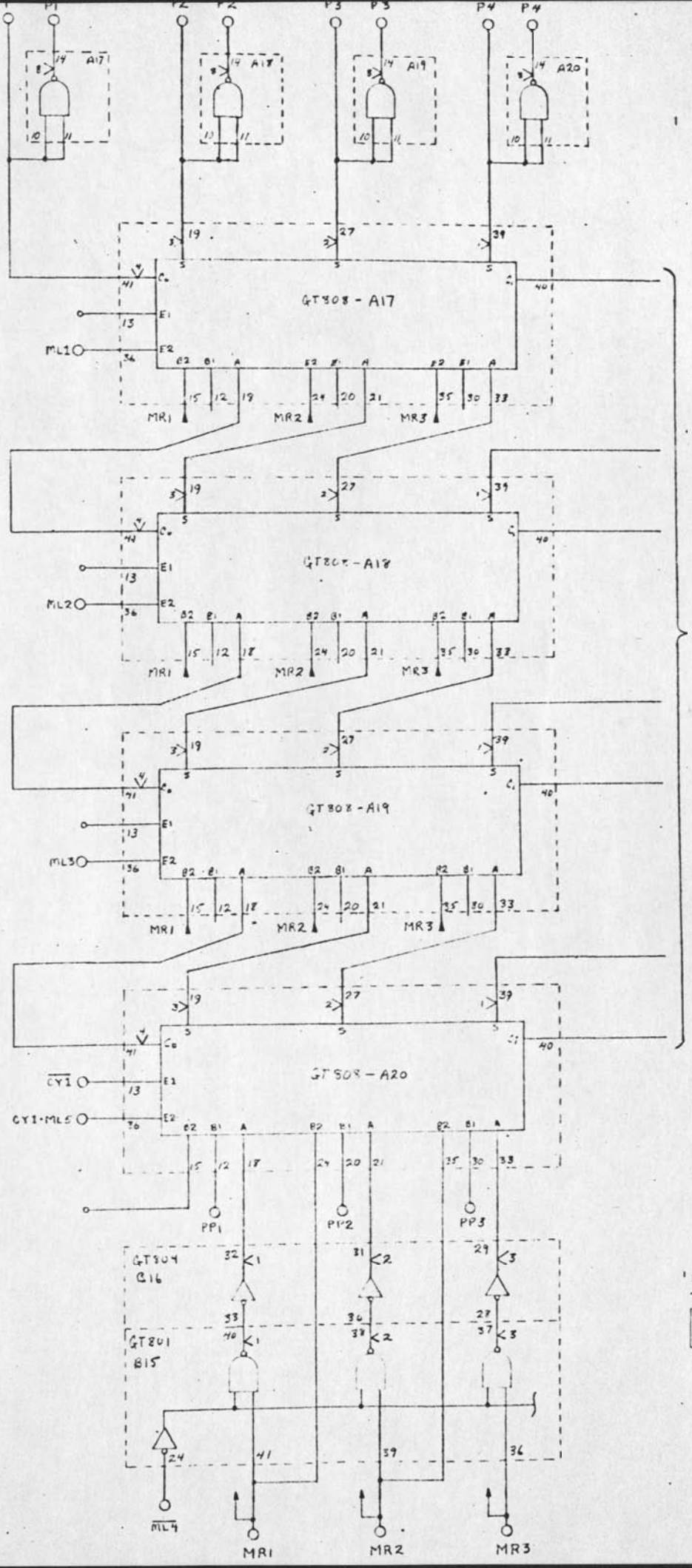


HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 10



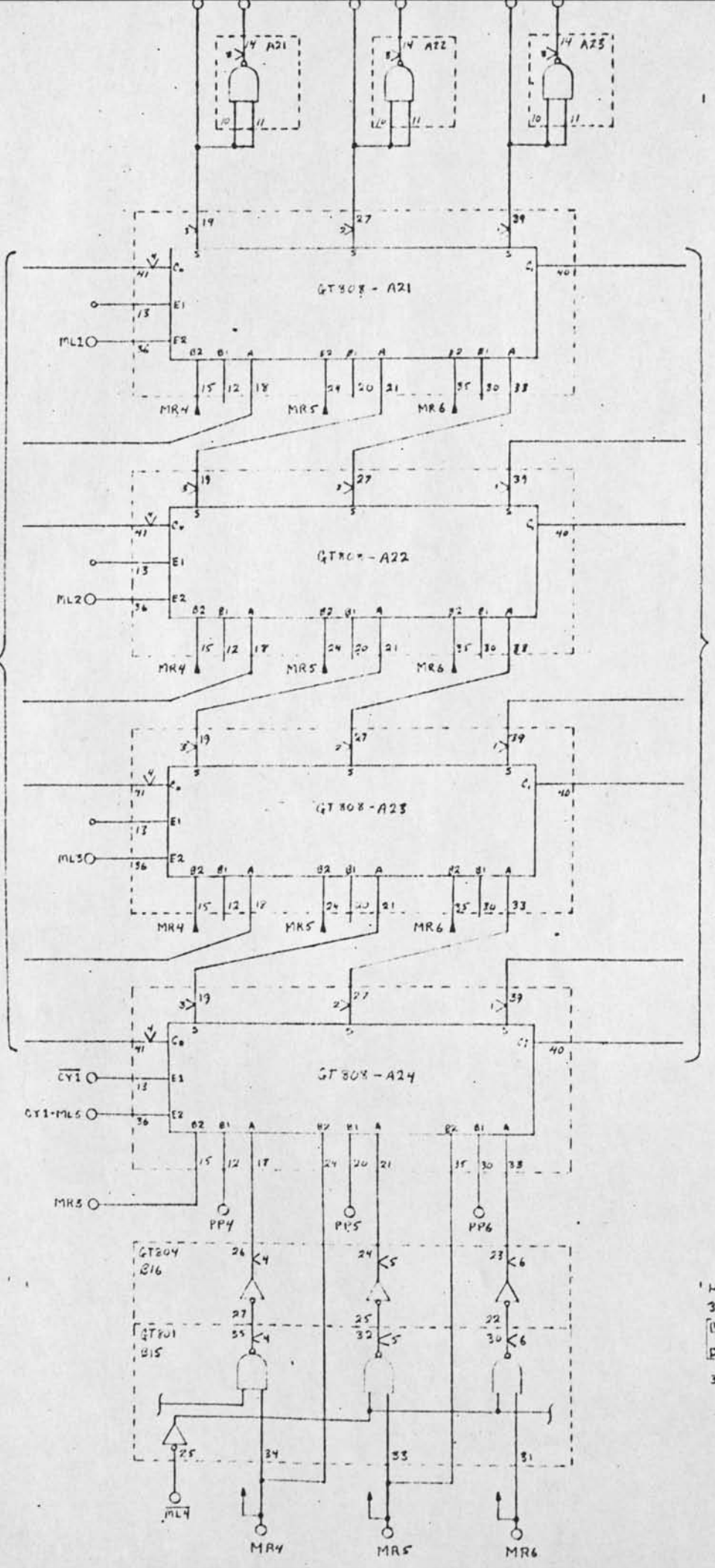
HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
Dwg # 11

8D-11

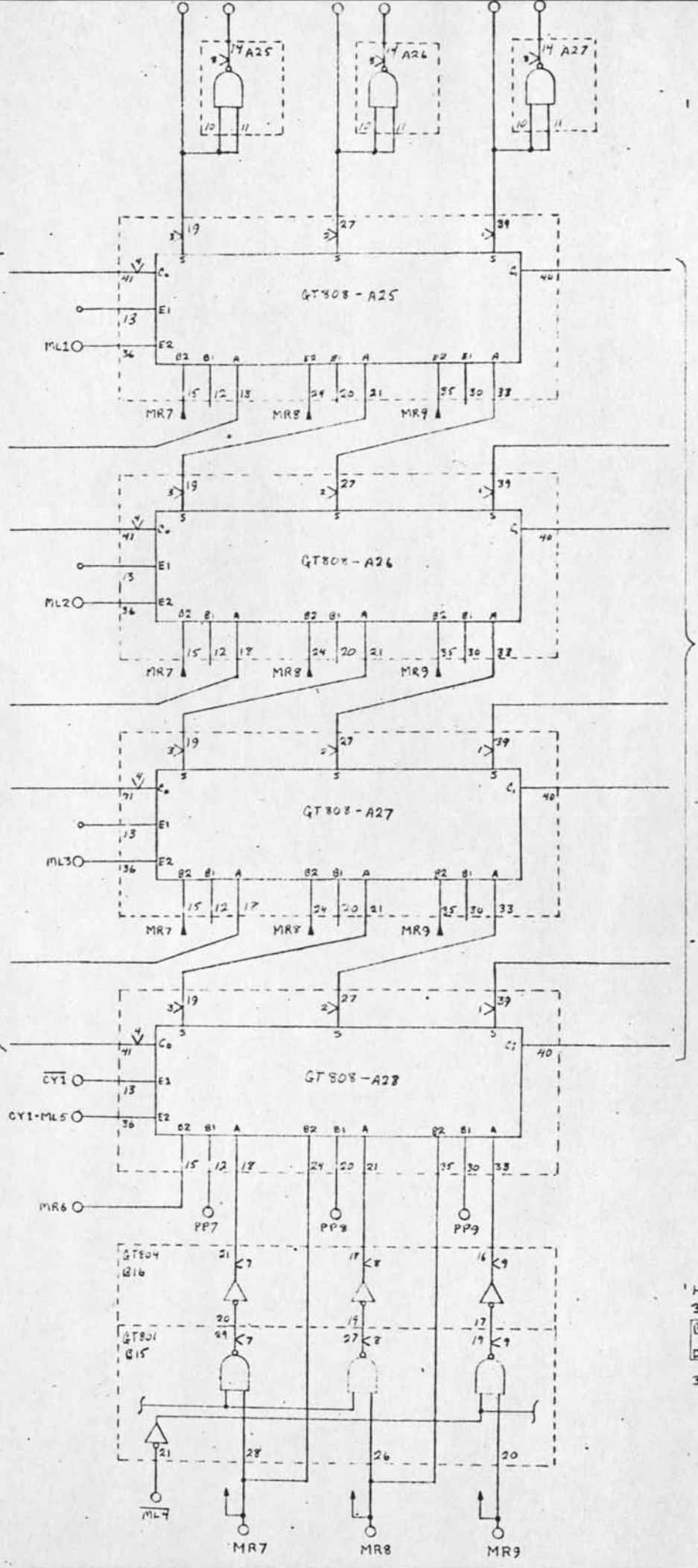


HARVARD UNIVERSITY
30 DISPLAY
MATRIX MULTIPLIER
DWG 12

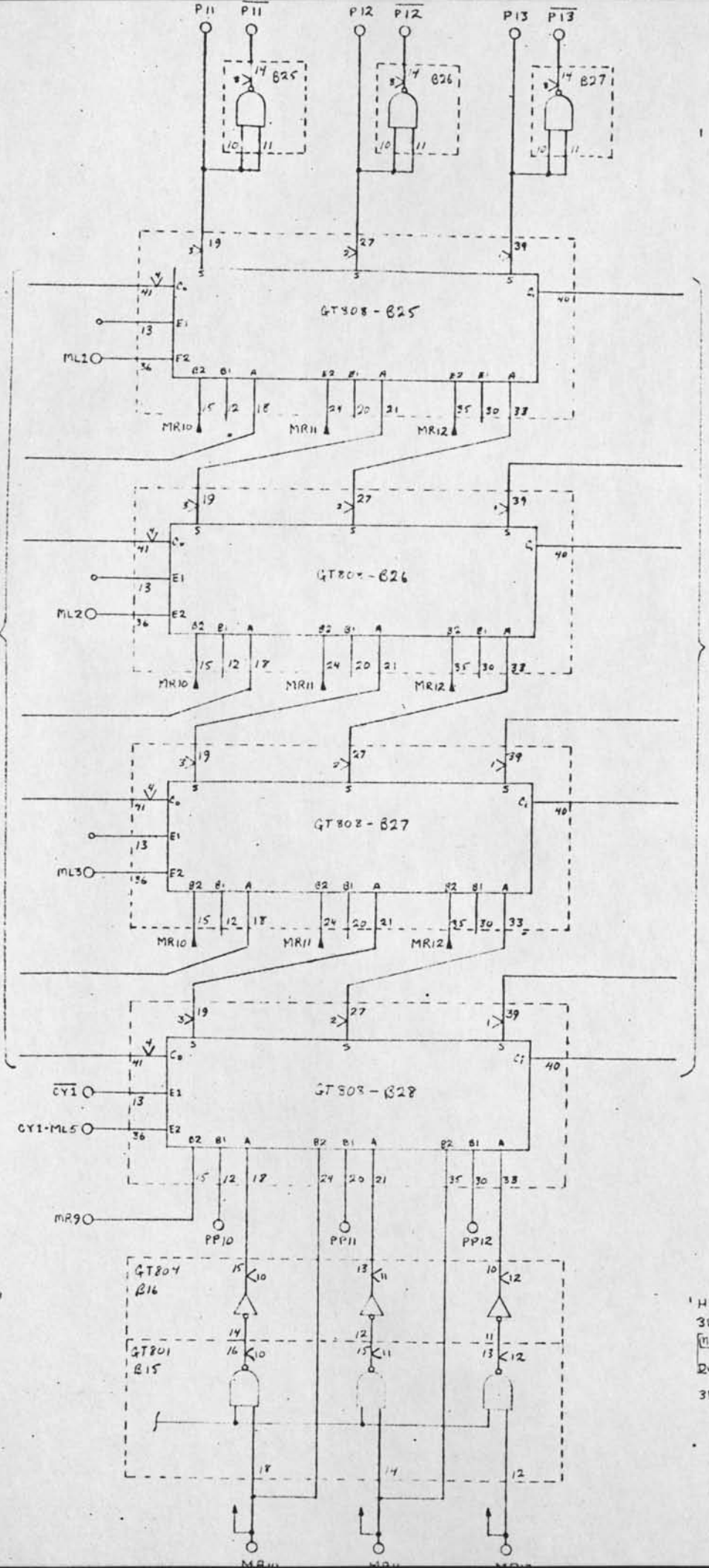
3D-12



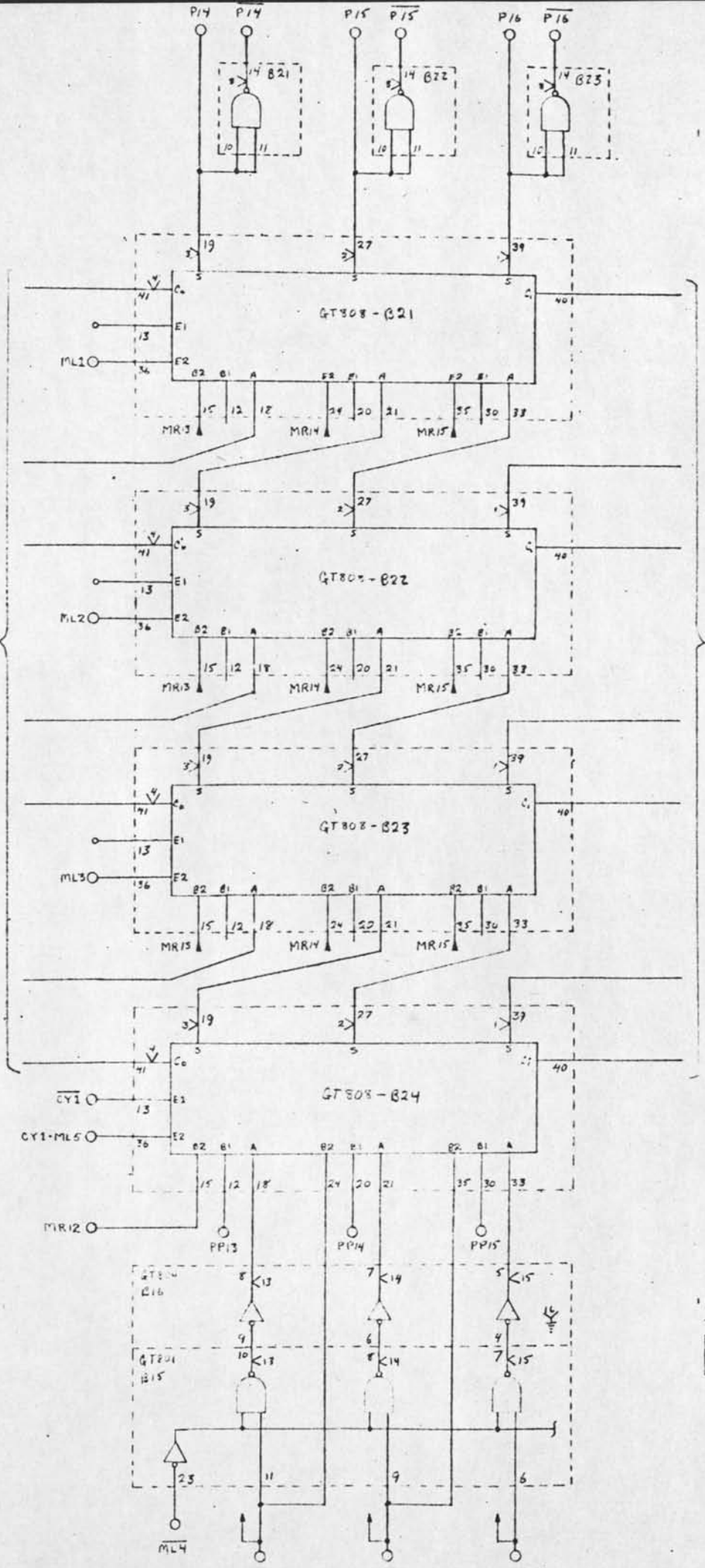
HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 13
3D-13



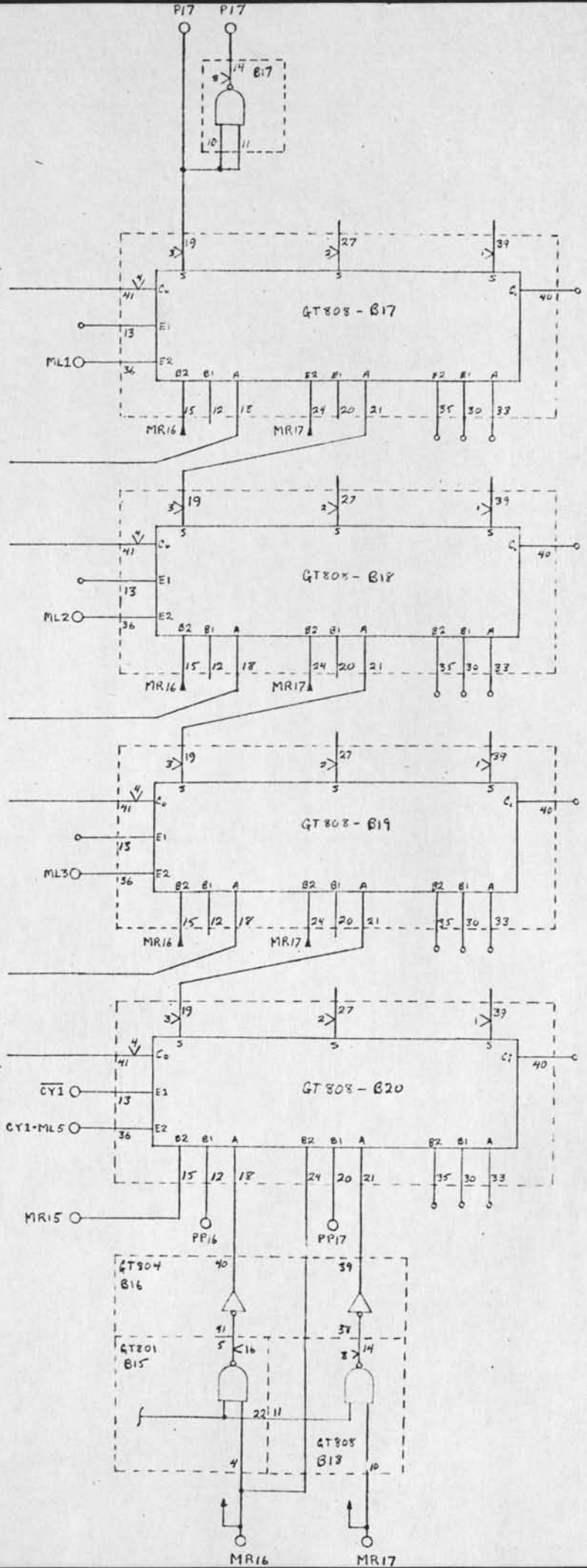
HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG. 14
3D-14



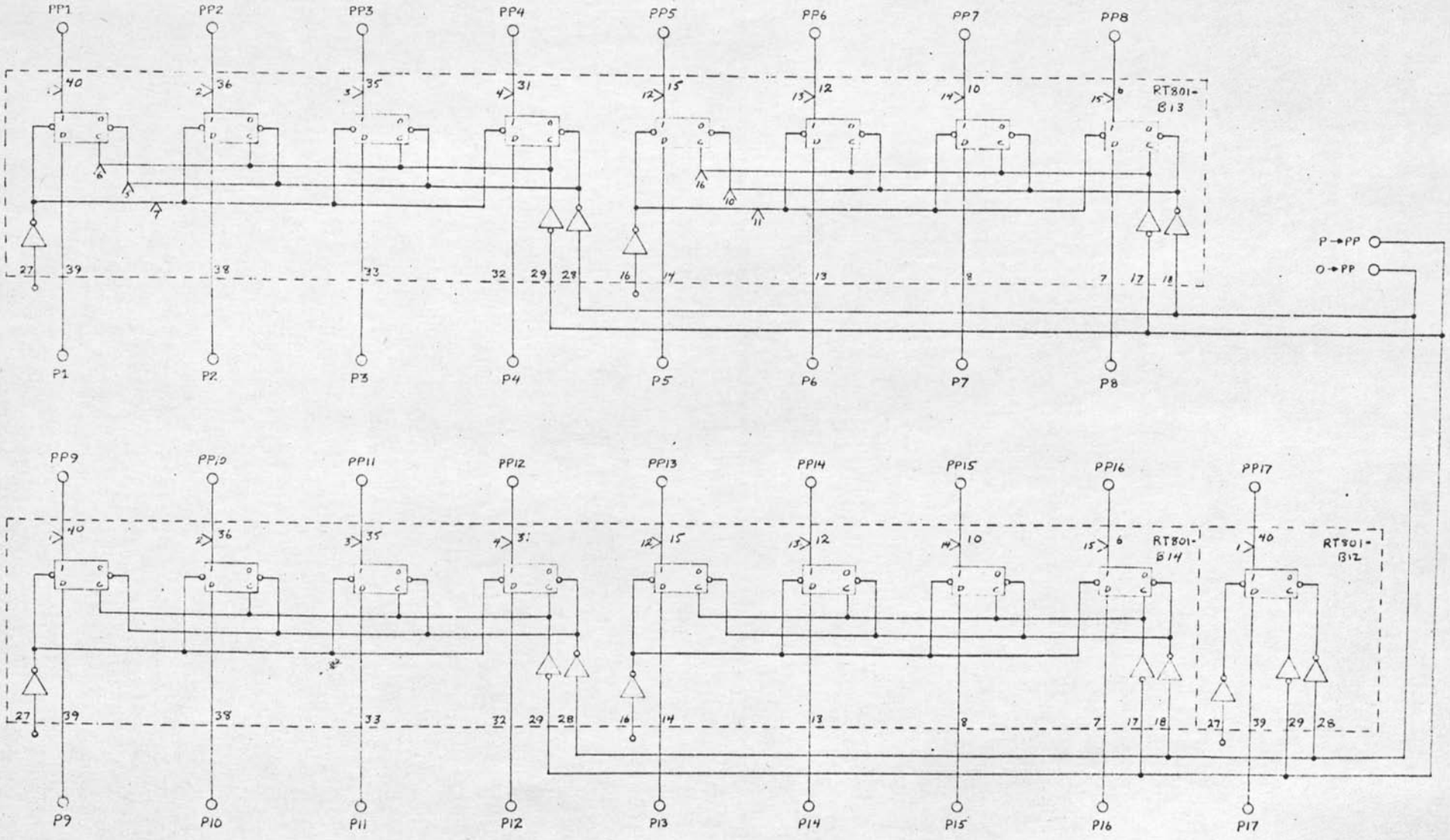
HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 15
3D-15



HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 16
3D-16

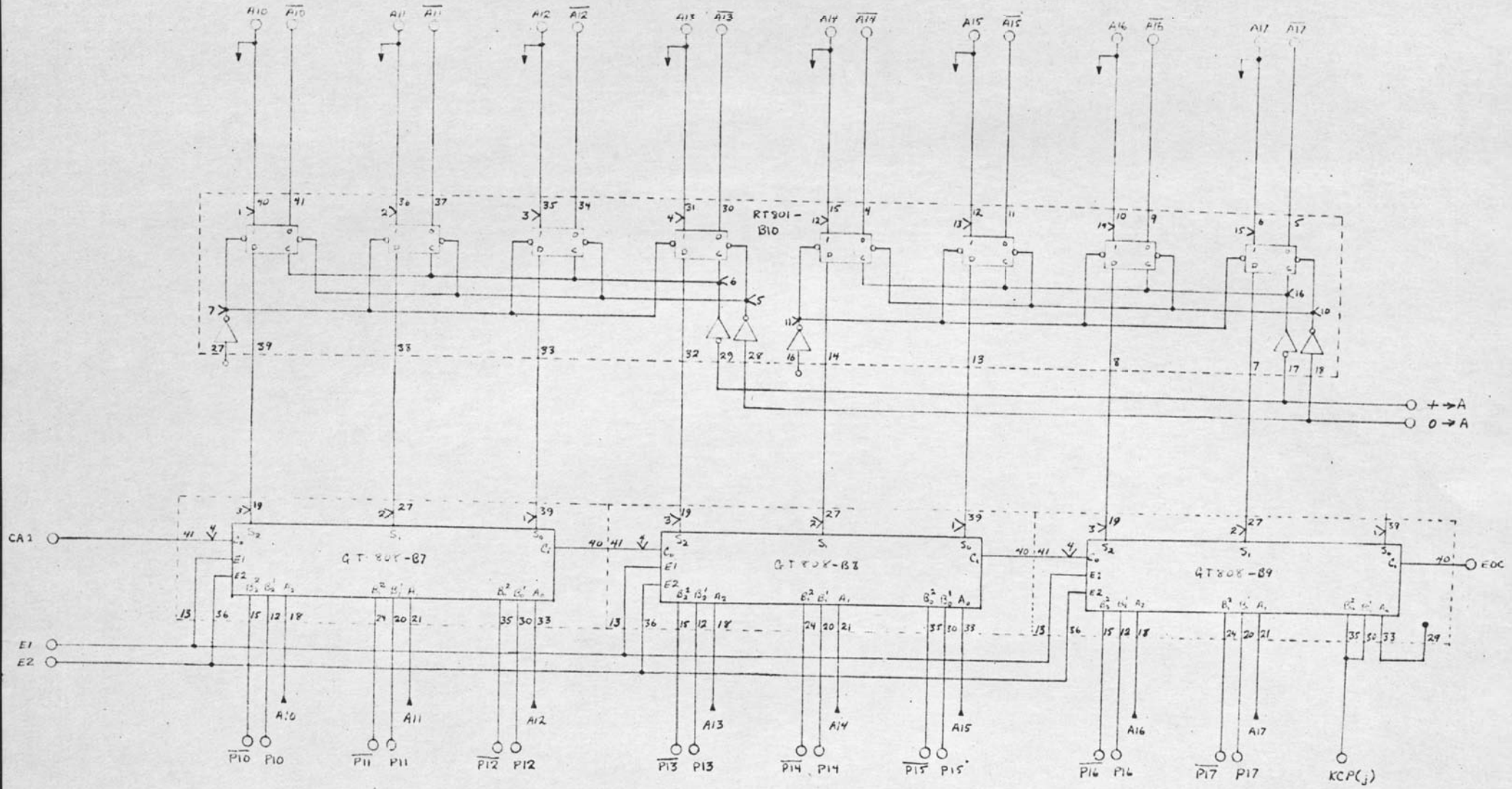


HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
Dwg 17
3D-17

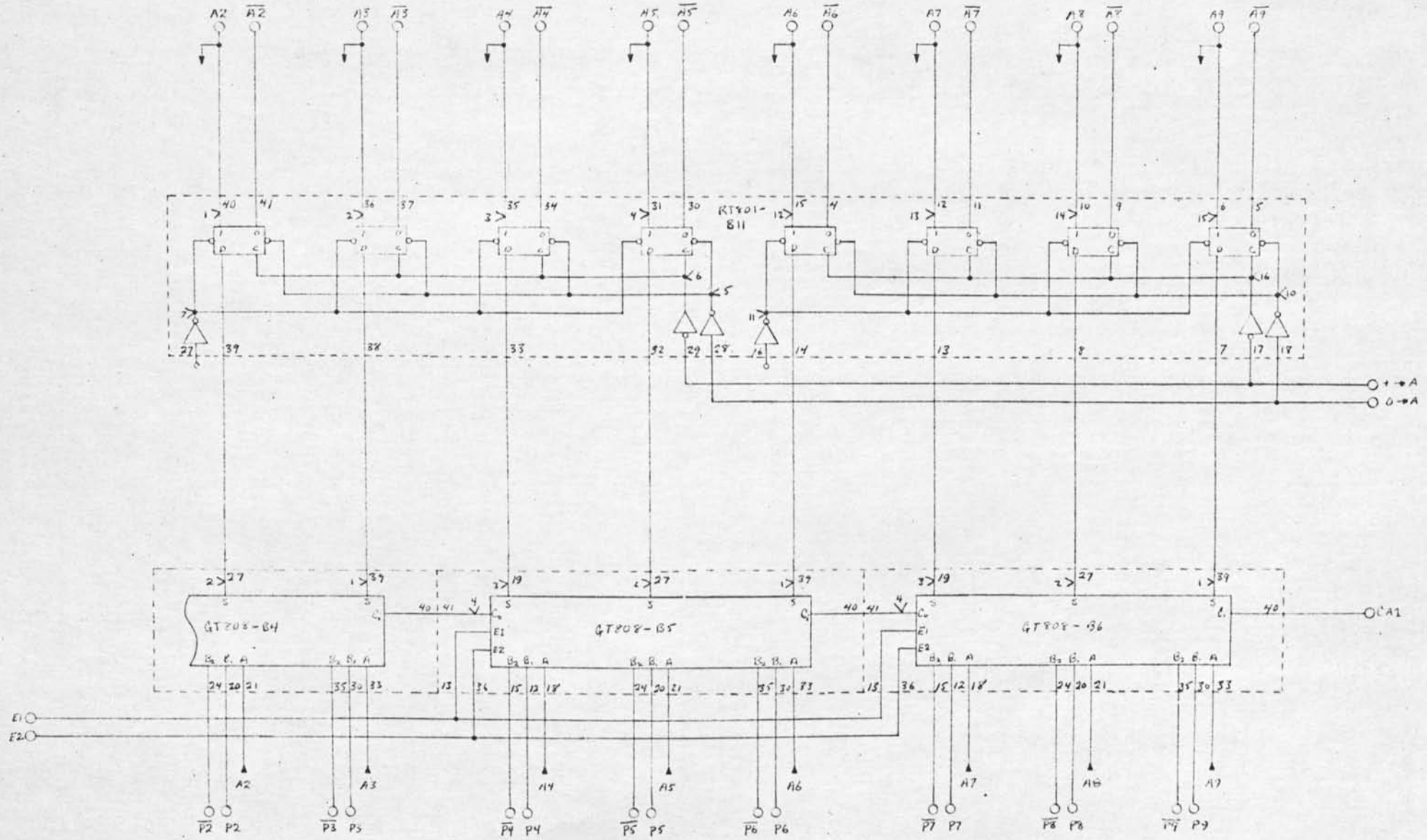


HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 18 P[illegible] PROD REG

30-18

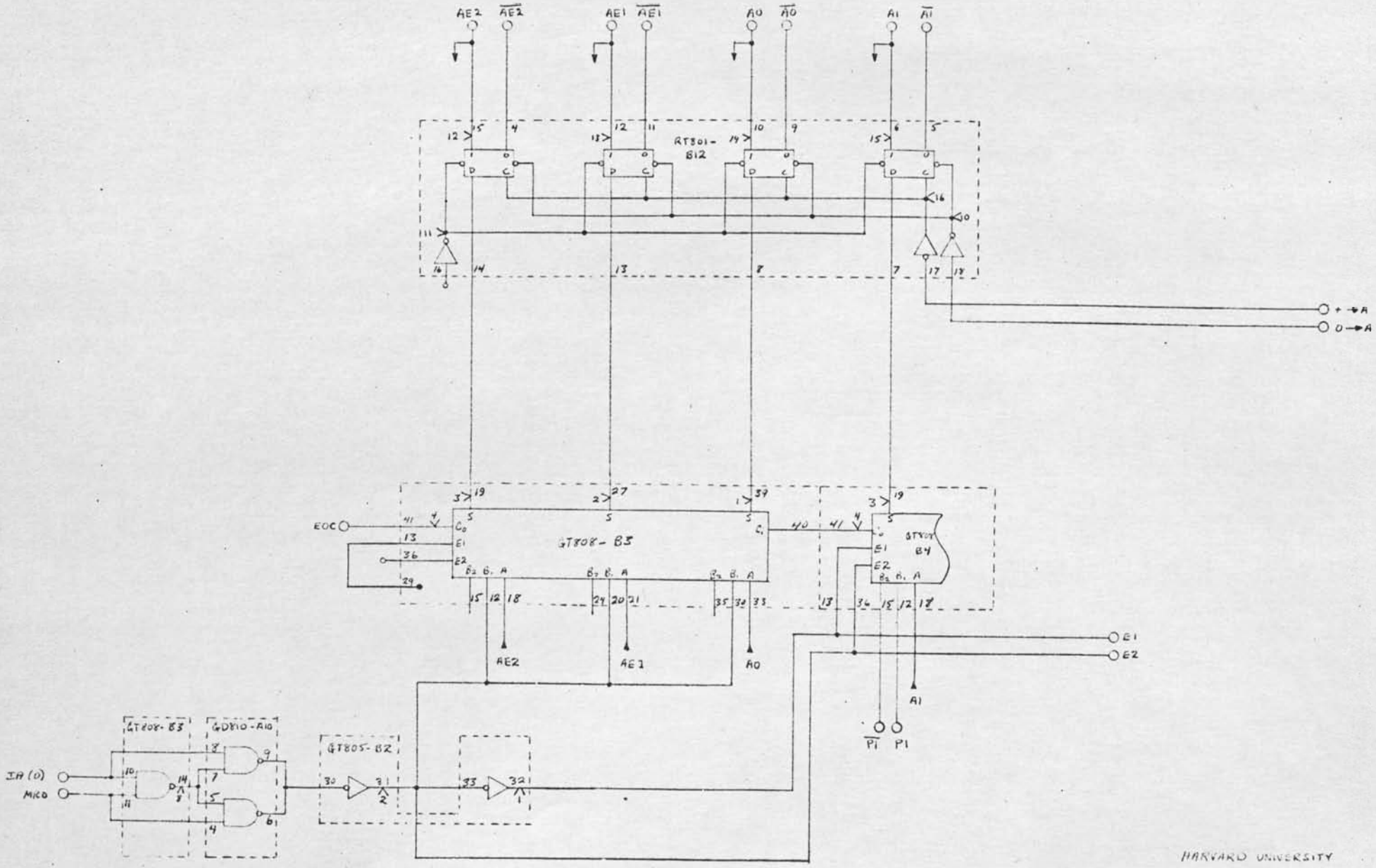


HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG.



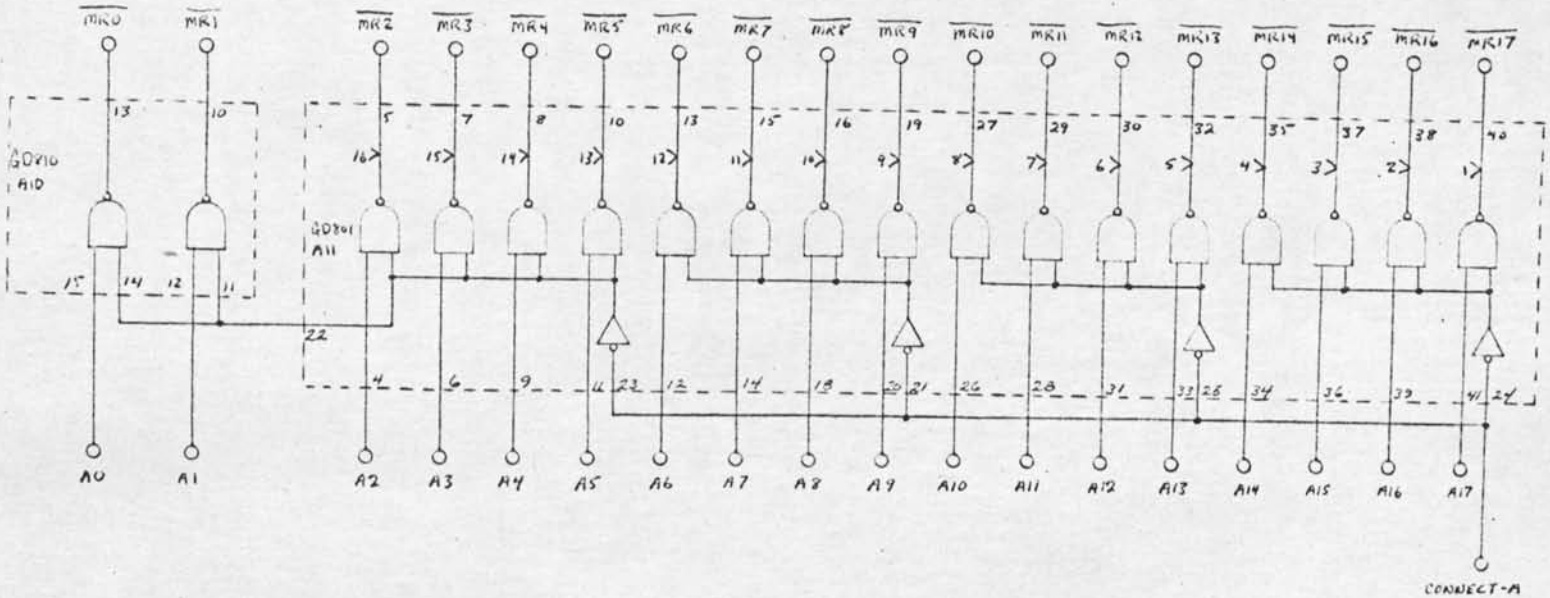
HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLEXER
DWG 20

30-20



HARVARD UNIVERSITY
3D DISPLAY
MATRIX MULTIPLIER
DWG 21

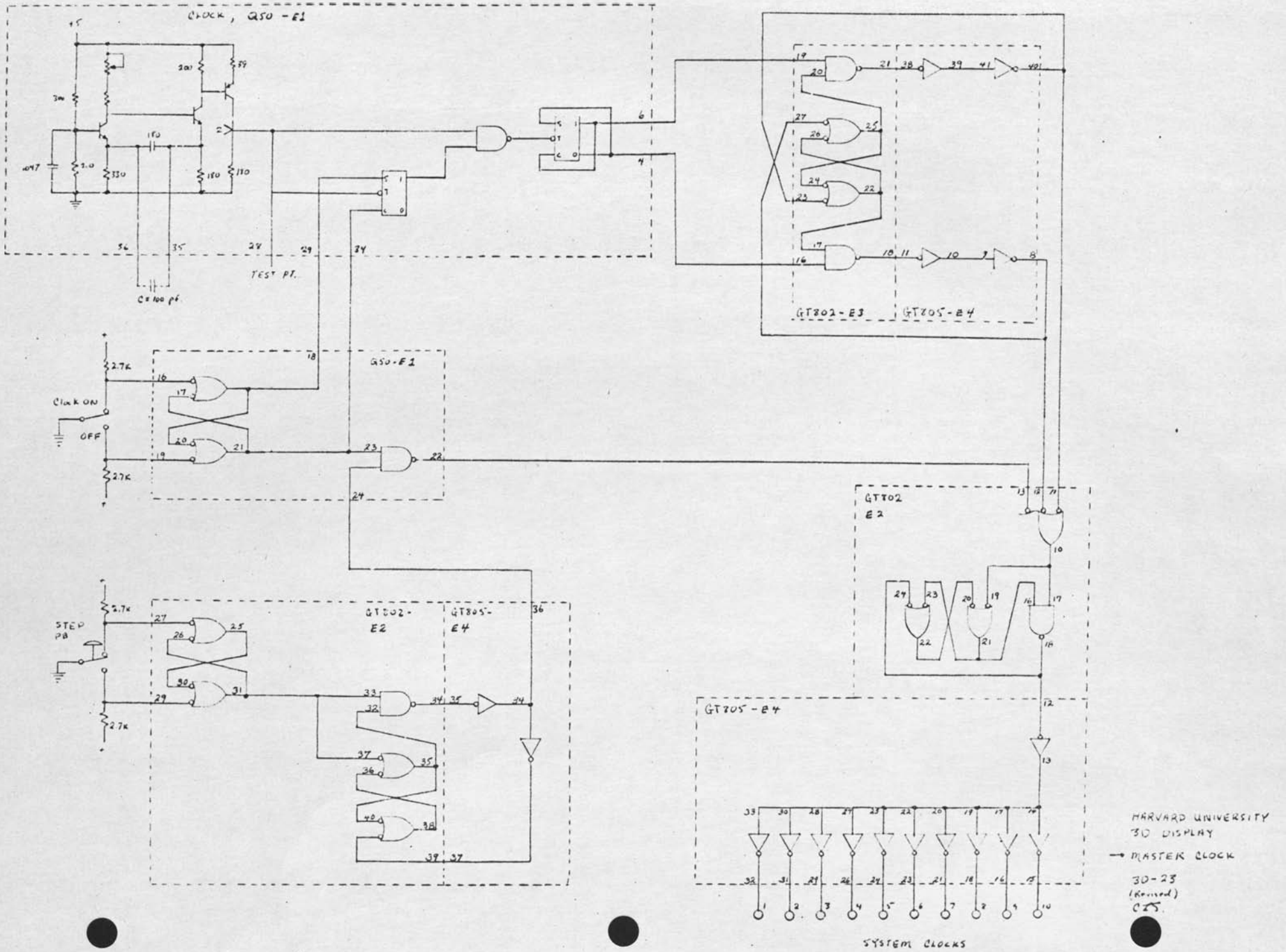
3D-21

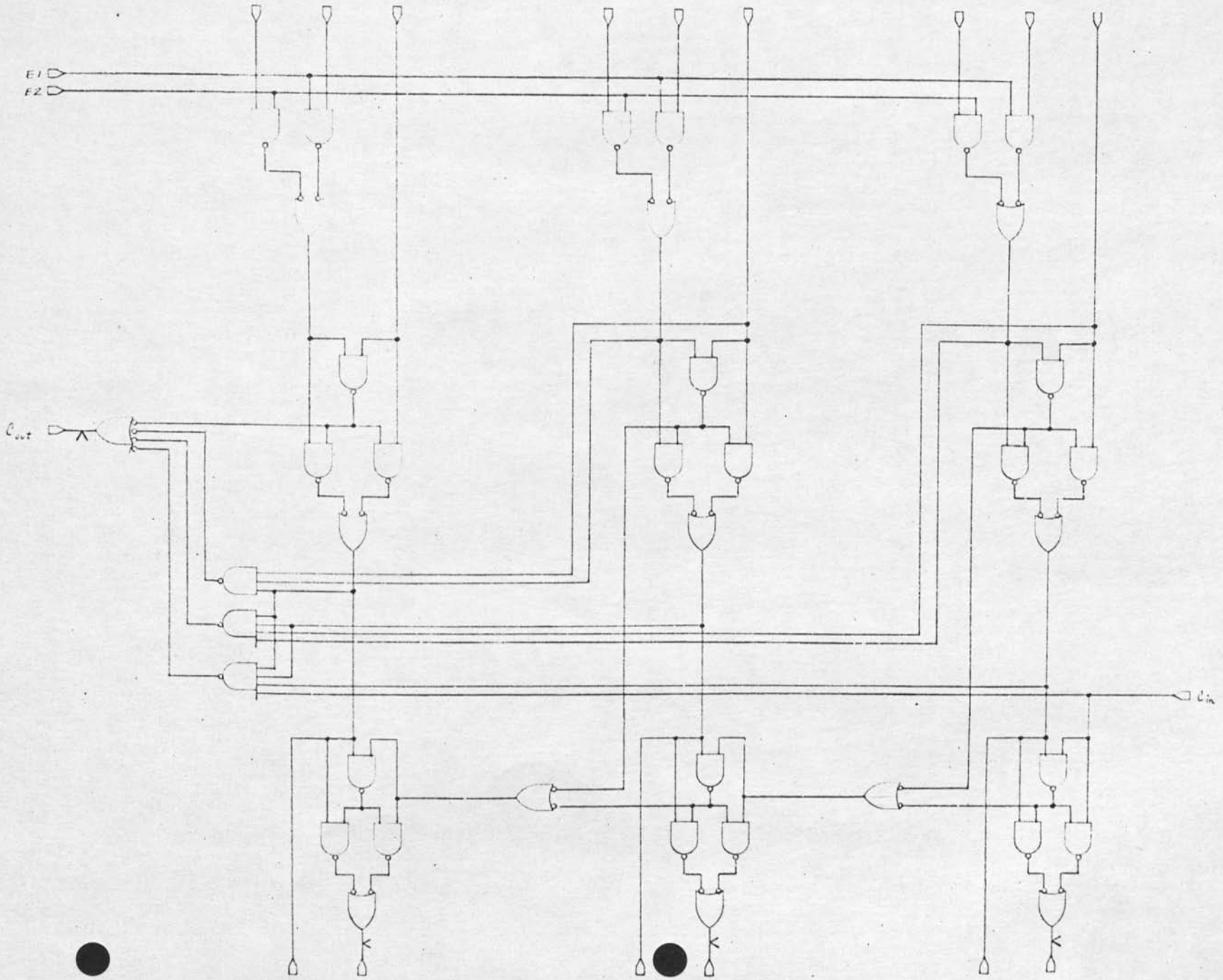


HARVARD UNIVERSITY
3D DISPLAY

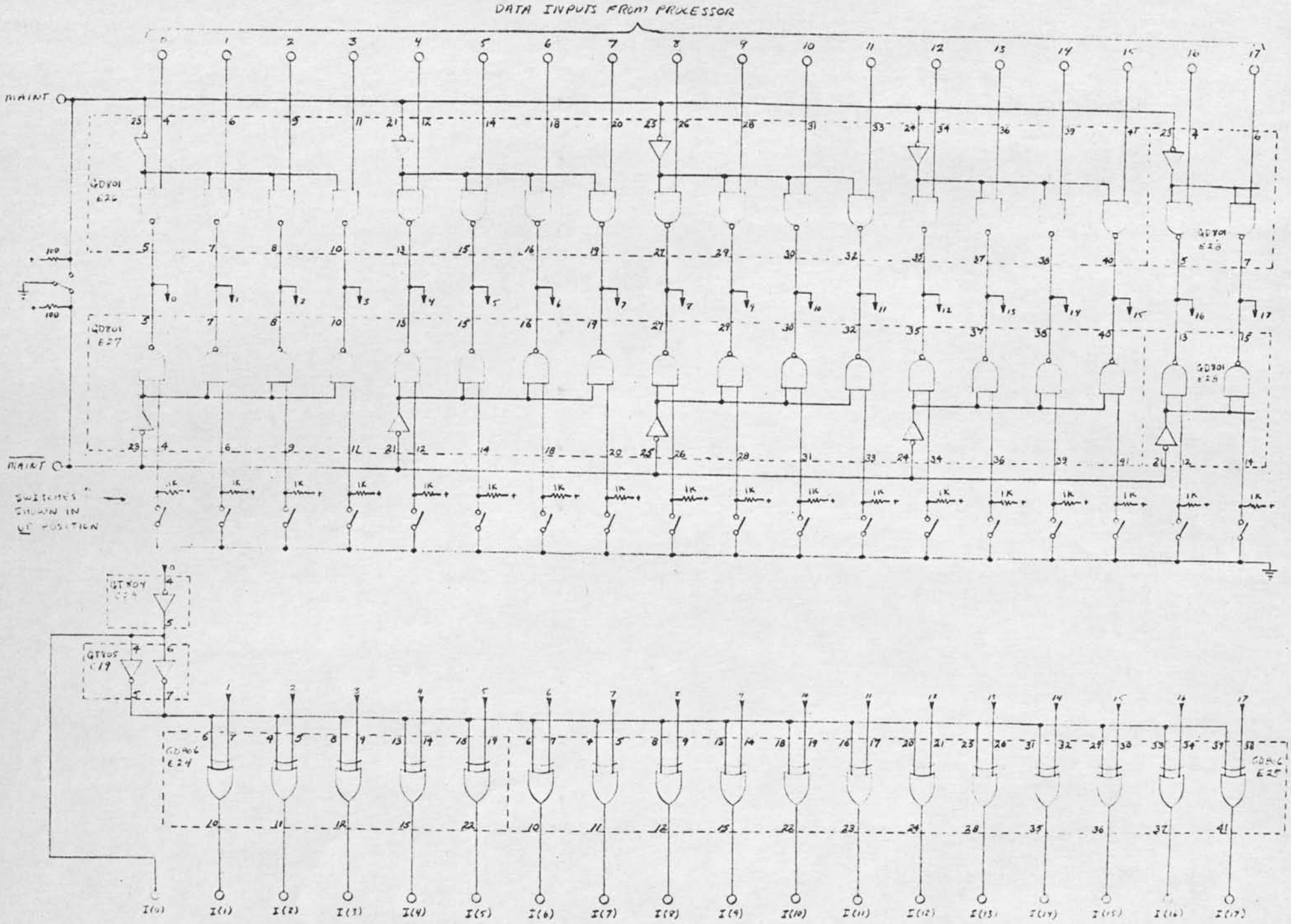
MATRIX MULTIPLIER
DWG 22

80-22


CLOCK, Q50 - E1
+5
300
39
3m
150
.047
1.0
330
180
110
C = 100 pf
TEST PT.
GT802 - E3
GT805 - E4
Q50 - F1
Clock ON
OFF
2.7K
STEP PB
GT802 - E2
GT805 - E4
GT802 E2
GT805 - E4
SYSTEM CLOCKS
HARVARD UNIVERSITY
3D DISPLAY
MASTER CLOCK
30-23
(Revised)
C25.



HARVARD UNIVERSITY
3-D DISPLAY
FAST 3-BIT ADDER
[MONITOR SYSTEMS GT-502]
{GATES ARE HLTTL [TI-74H
T = 6 nsec.

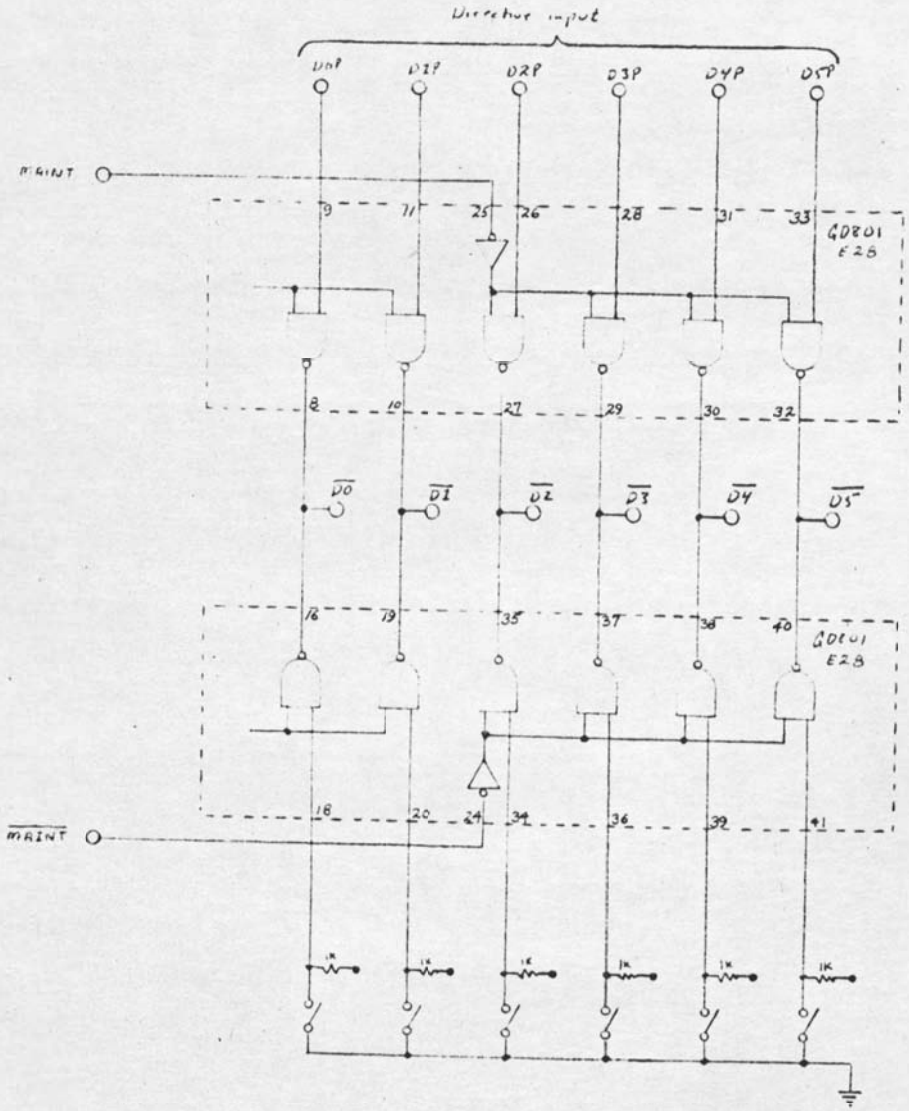


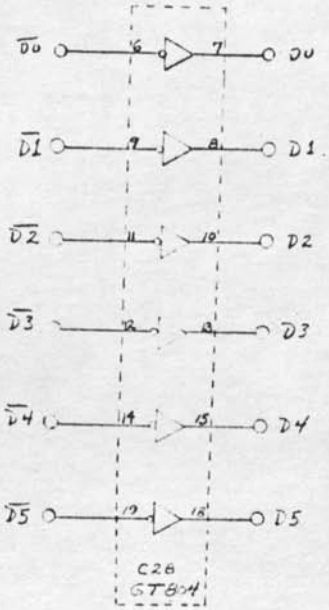
MULTIPLIER CONTROL
DWG # 2

Maintenance & Processor Data
Inputs & I's Comp outputs

HARVARD UNIVERSITY
DISPLAY

BD-24



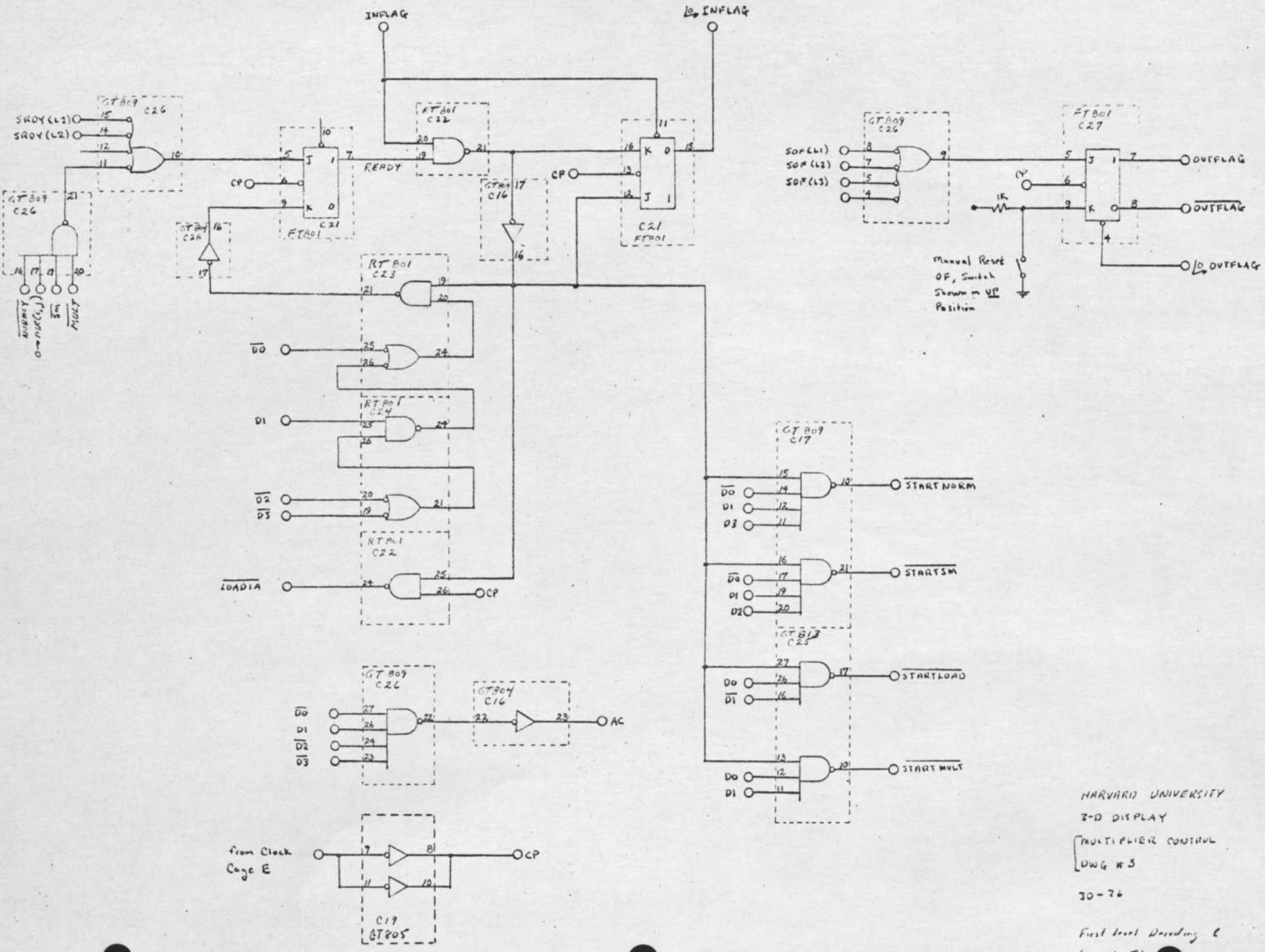


HARVARD UNIVERSITY
3-D DISPLAY

MULTIPLIER CONTROL
DWG #2

Maintenance & Processor Directive Inputs.

3D-25



HARVARD UNIVERSITY

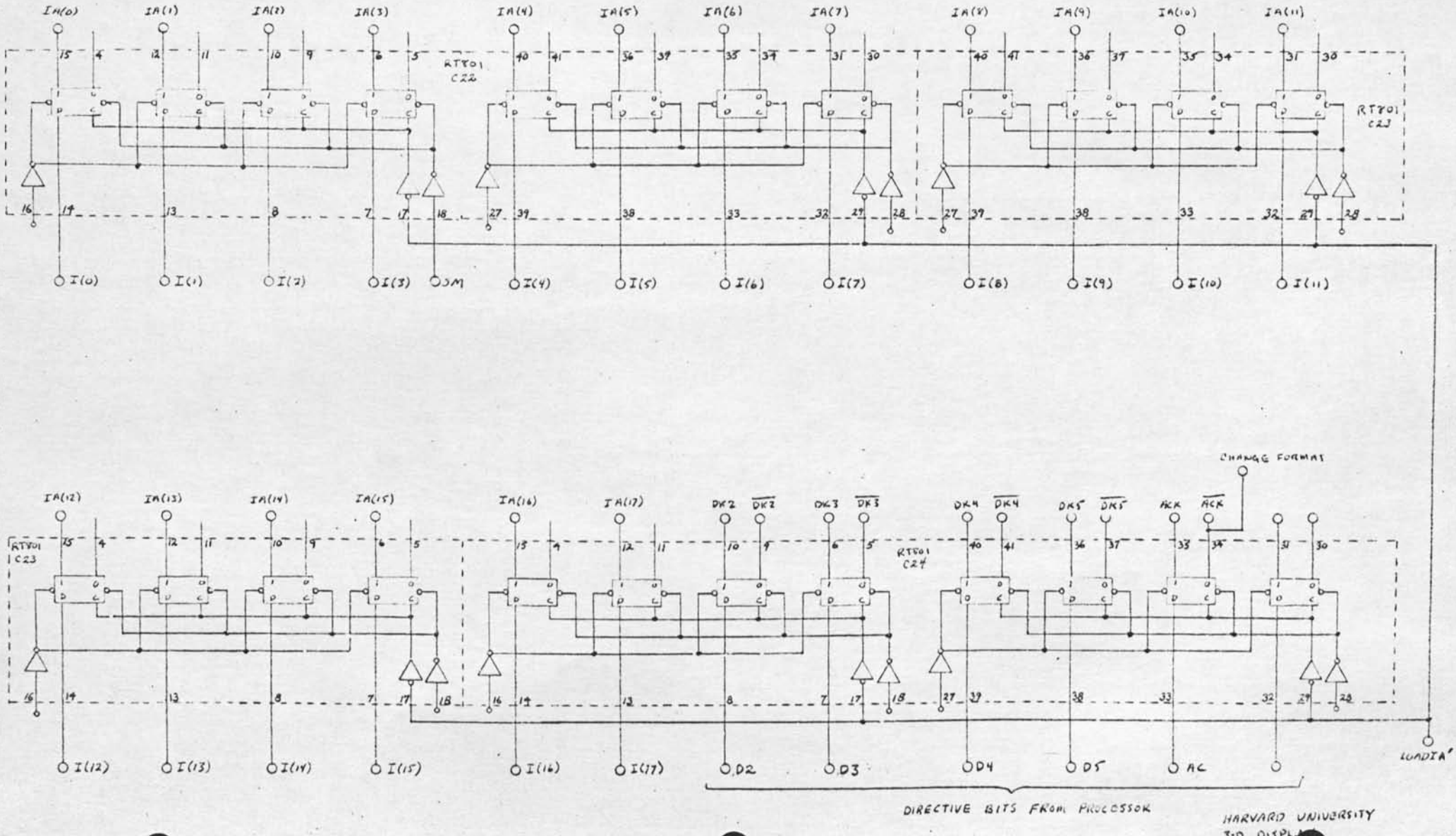
3-D DISPLAY

MULTIPLIER CONTROL

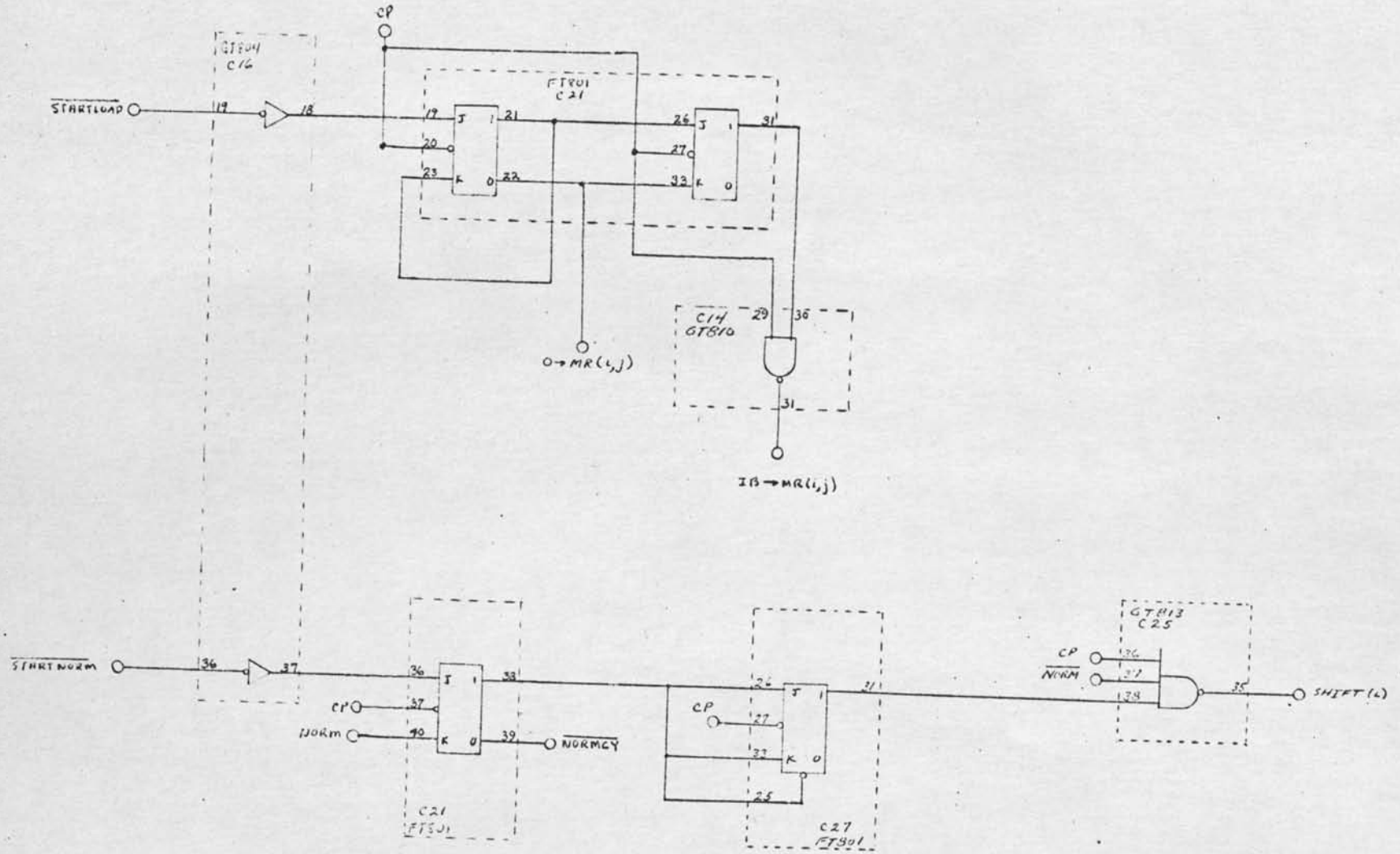
DWG # 3

30-76

First level Decoding & Control Tim



HARVARD UNIVERSITY
3-D DISPL
MULTIPLIE TRUL
DWG # 4



HARVARD UNIVERSITY

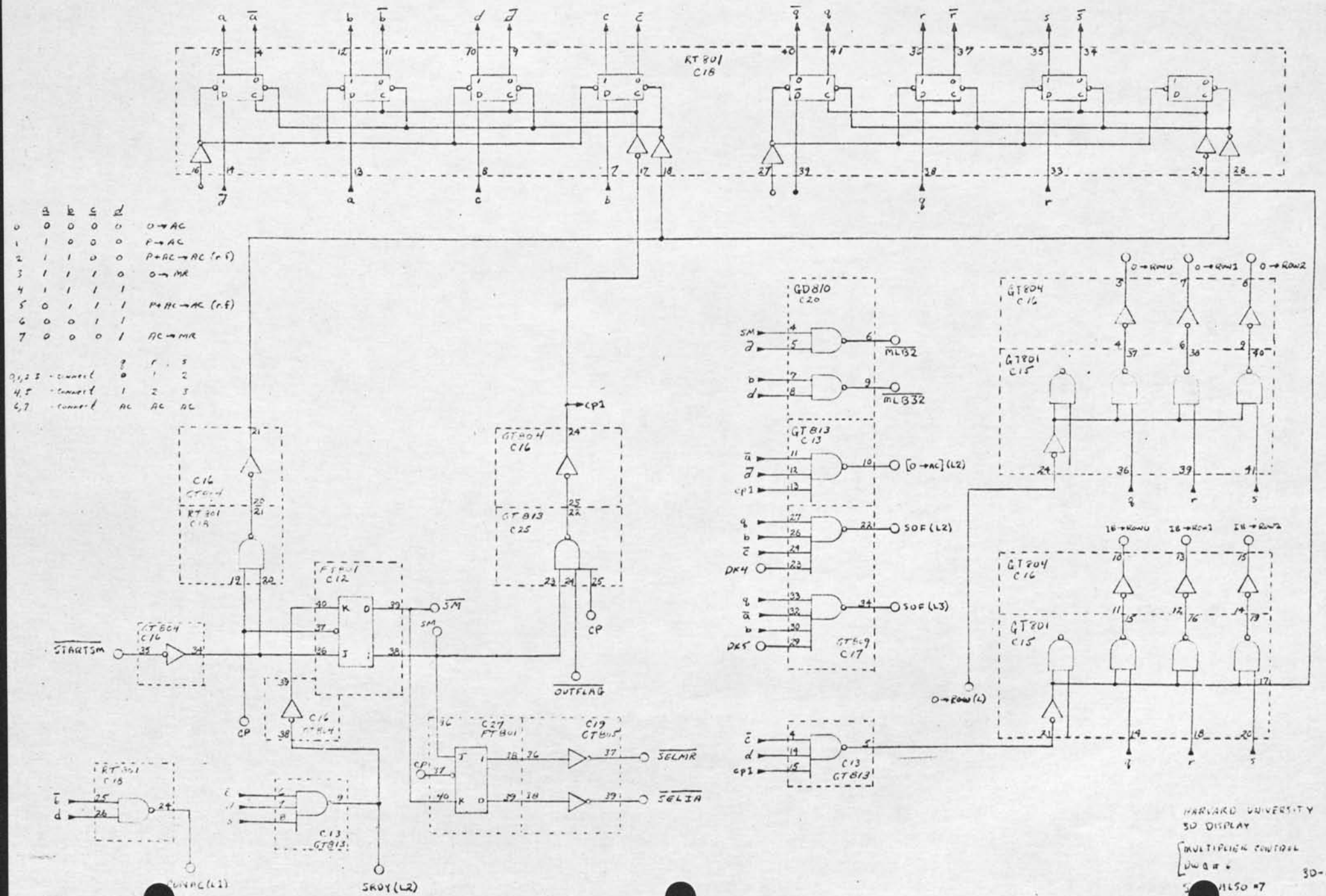
3-D DISPLAY

MULTIPLIER CONTROL
DWG # 5

Typical Gates

Level, IV Size

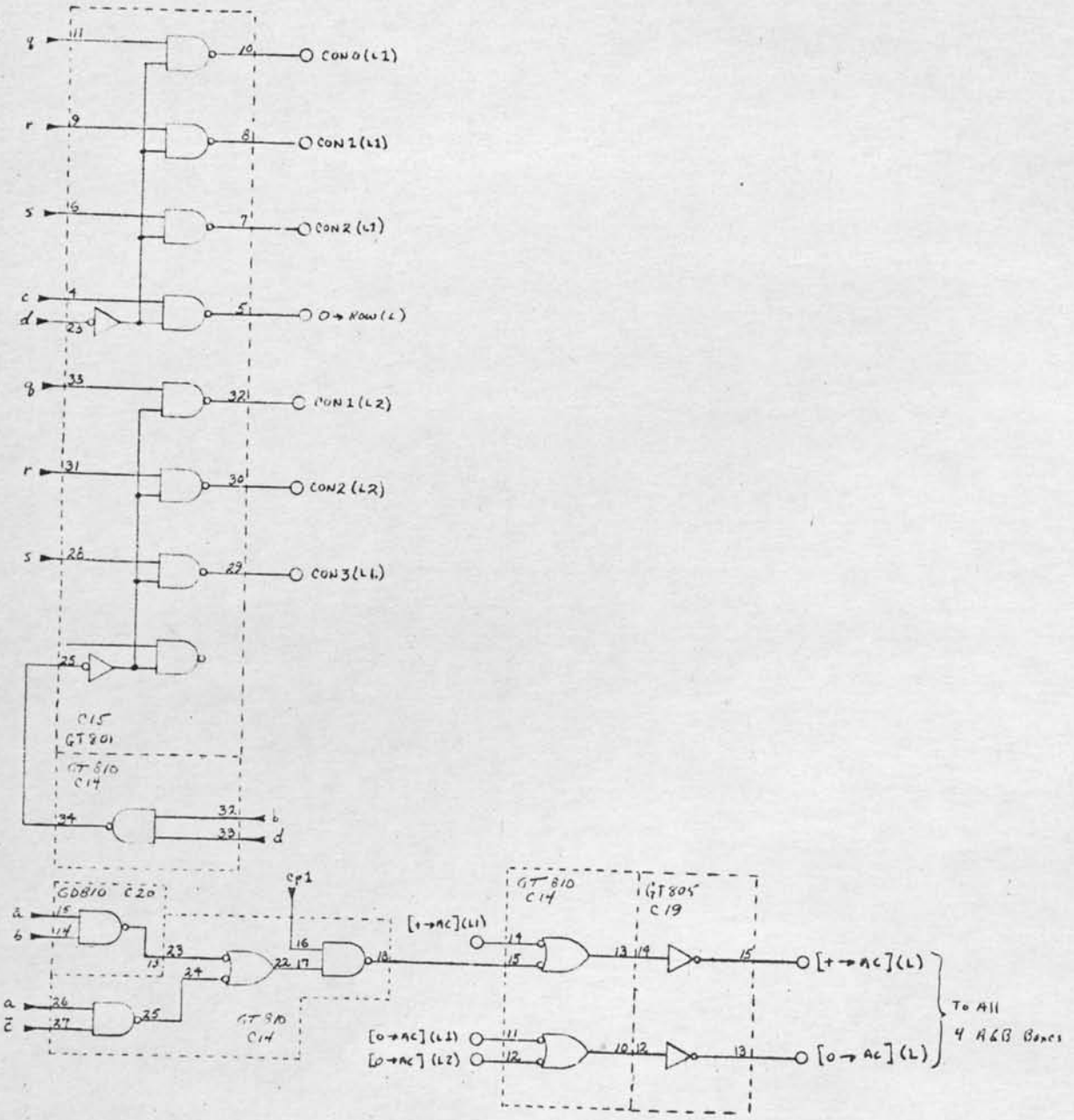
30-28



	a	b	c	d	
0	0	0	0	0	0→AC
1	1	0	0	0	P→AC
2	1	1	0	0	P+AC→AC (r.f)
3	1	1	1	0	0→MR
4	1	1	1	1	
5	0	1	1	1	P+AC→AC (r.f)
6	0	0	1	1	
7	0	0	0	1	AC→MR

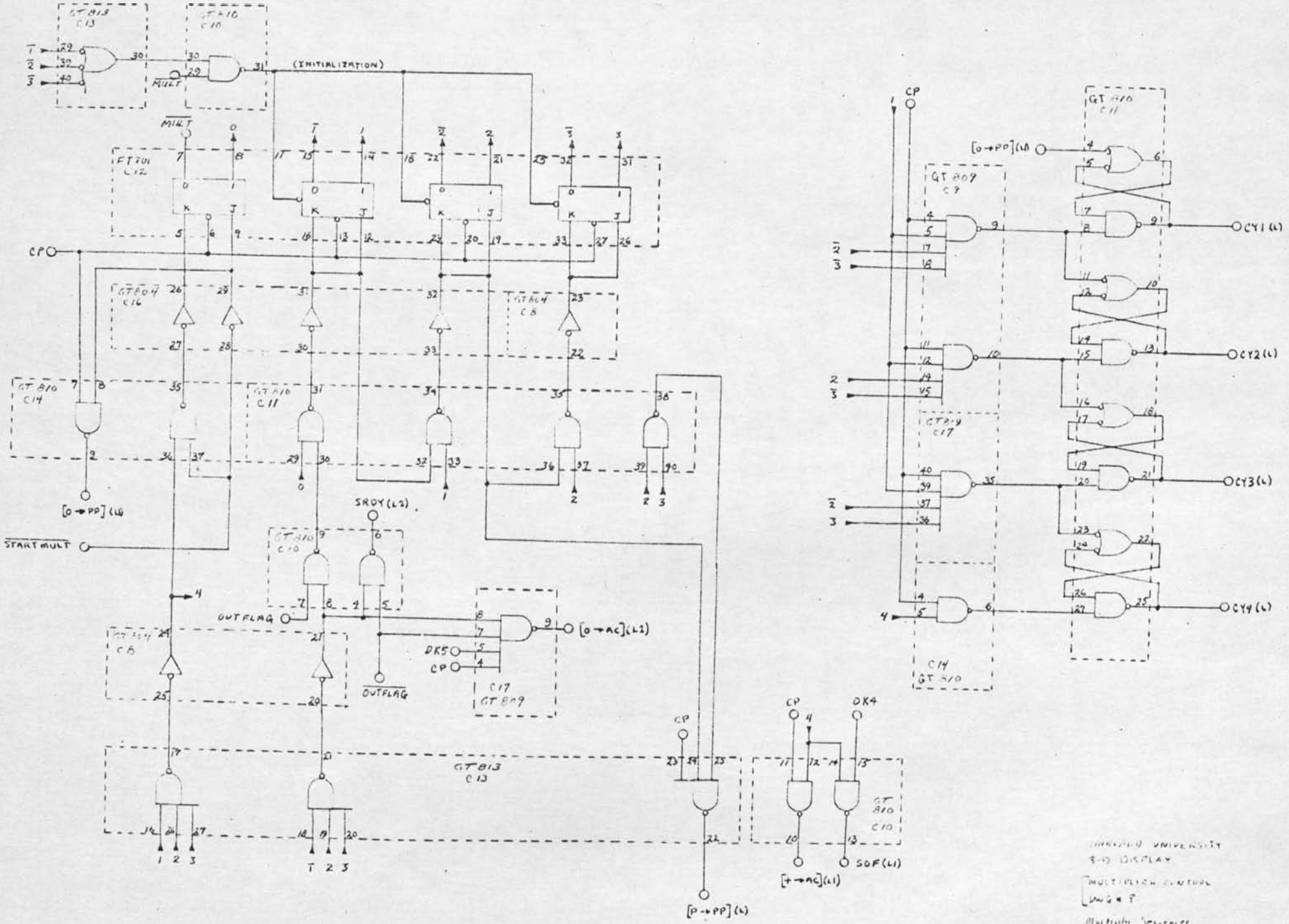
	q	r	s
0,1,2,3 - connect	0	1	2
4,5 - connect	1	2	3
6,7 - connect	AC	AC	AC

HARVARD UNIVERSITY
3D DISPLAY
MULTIPLIER CONTROL
DWG # =
SHEET 7
Shift Multiply Sequencer

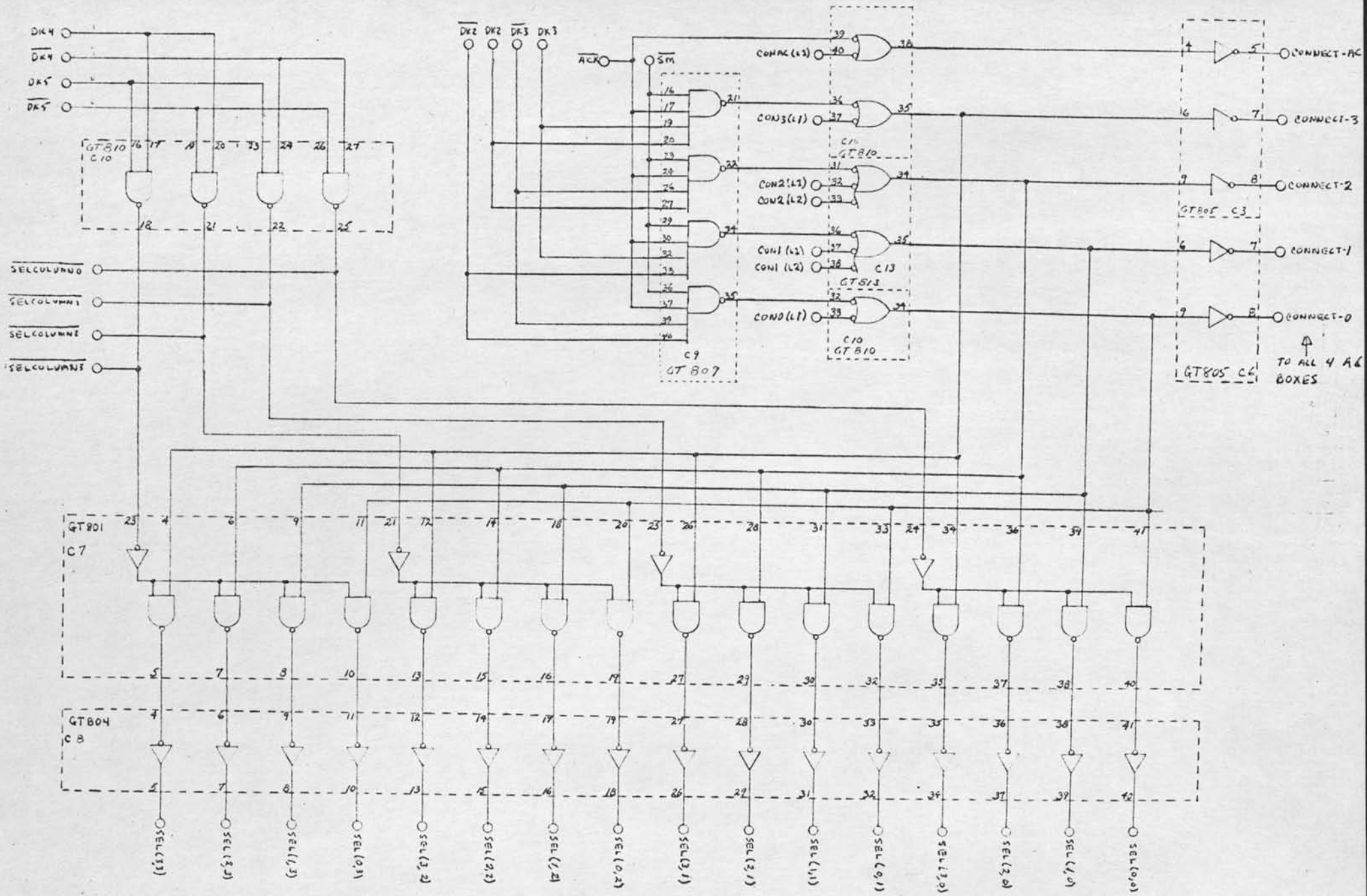


Note: small letters → refer to dwg #6

HARVARD UNIVERSITY
3-D DISPLAY
MULTIPLIER CONTROL
DWG #7
30-30



STANFORD UNIVERSITY
B-10 DISPLAY
MULTIPLEX CONTROL
Dwg # ?
Multiply Sequencer
B10-31



HARVARD UNIVERSITY

3-D Display

MULTIPLIER CONTROL
DWG. 9

COLUMN DECODE

32

HARVARD UNIVERSITY

3-D Display

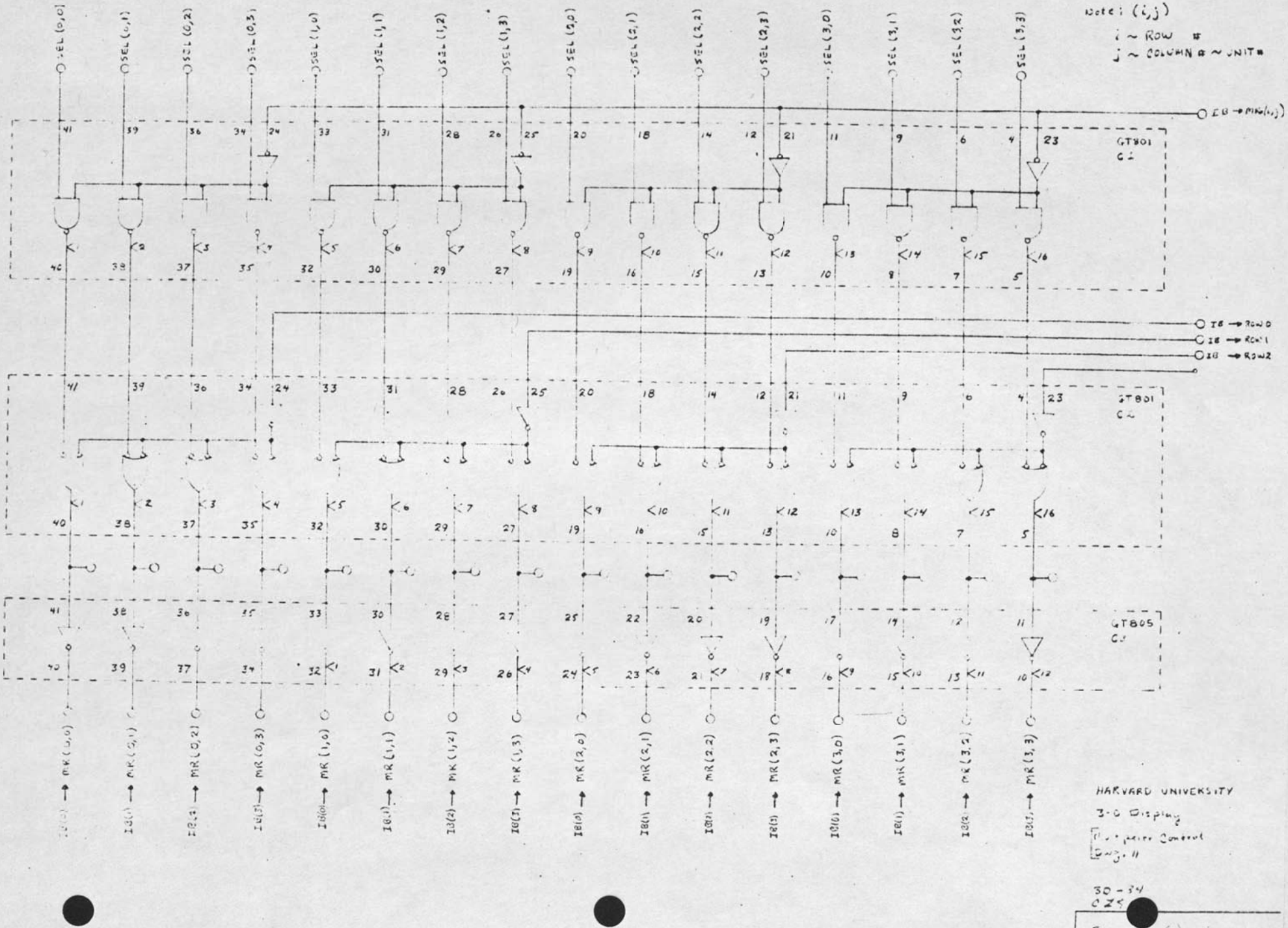
Multiplier Control

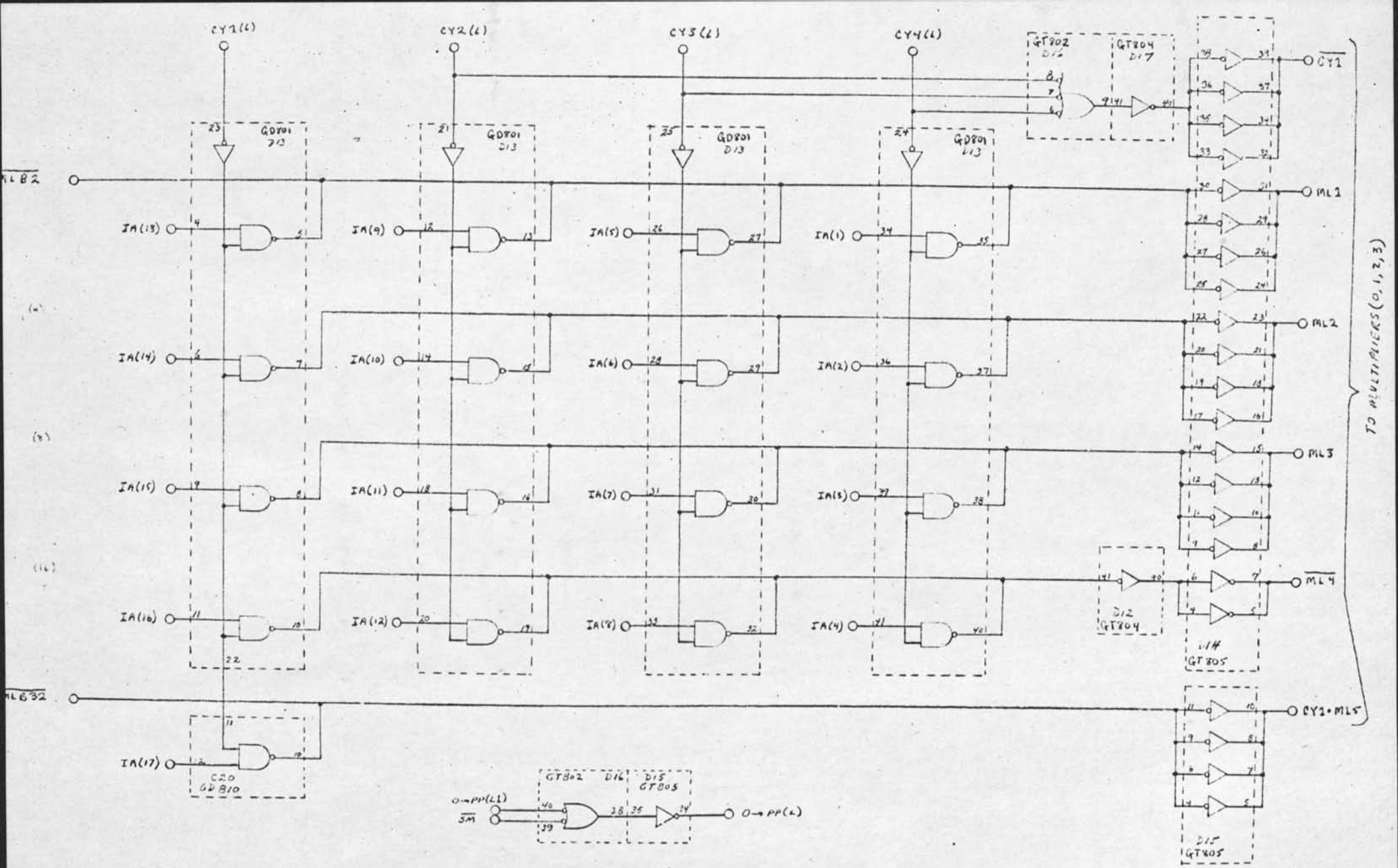
Dwg. 10

30-33

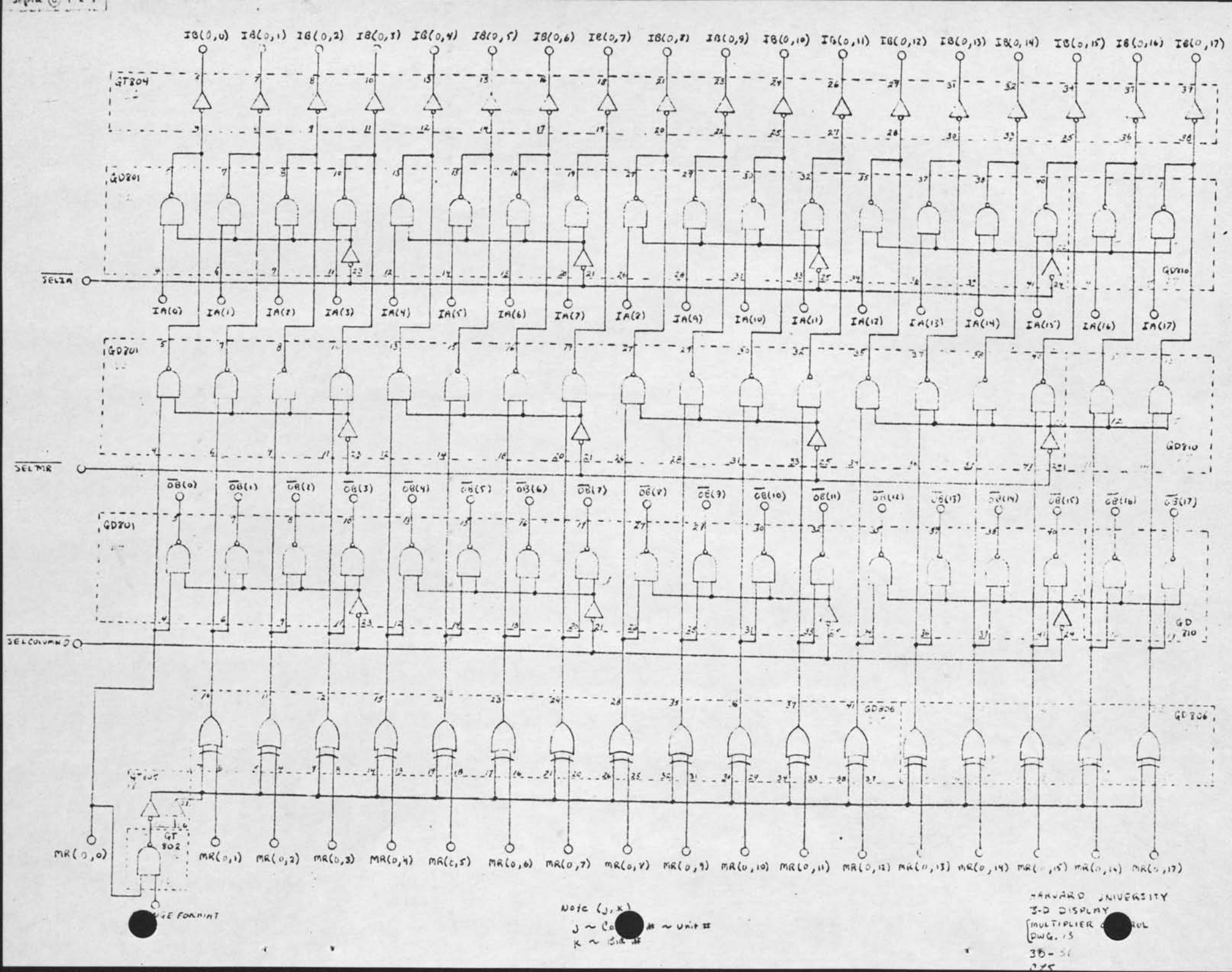
CXS

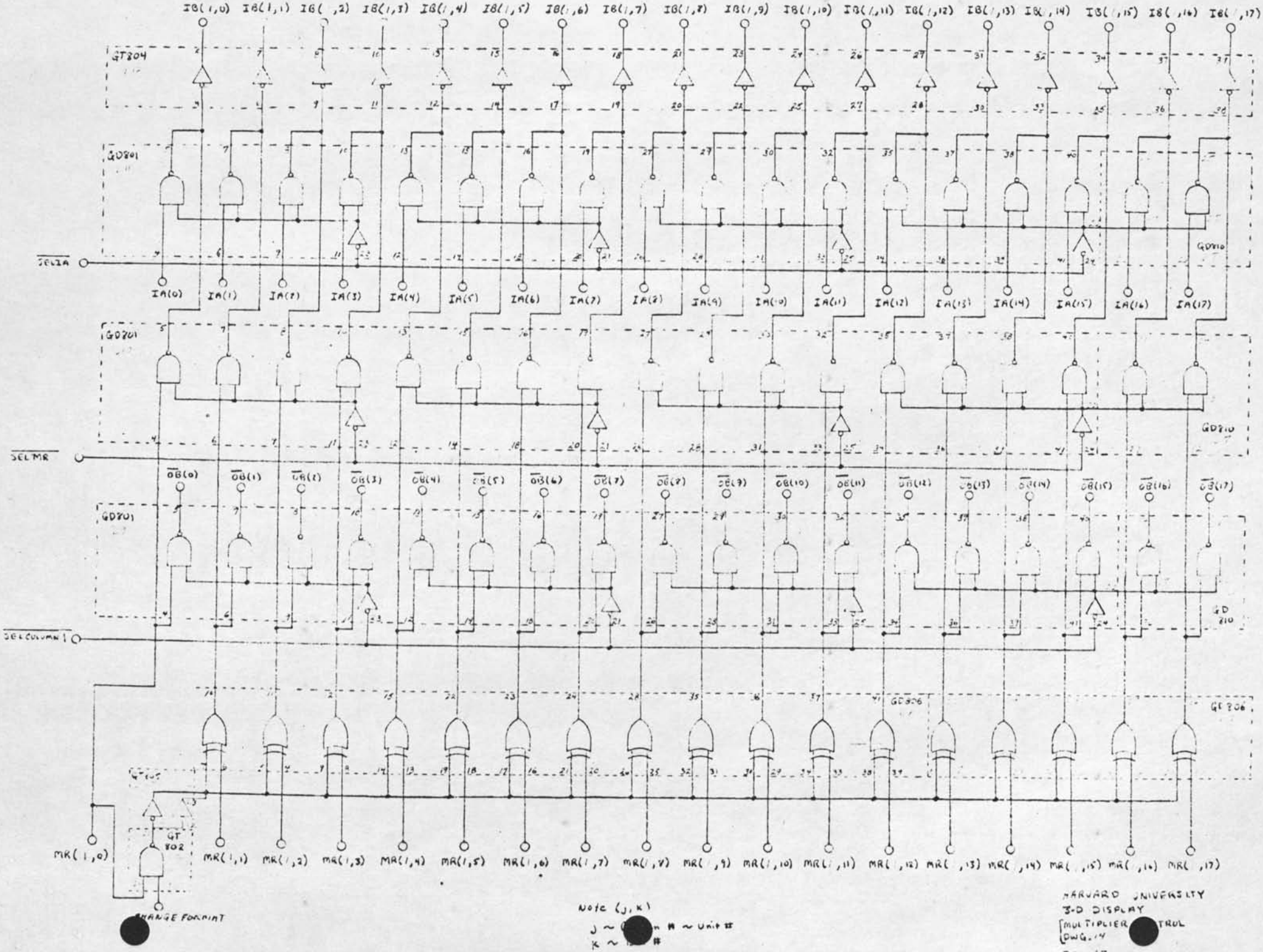
Sept. 1962

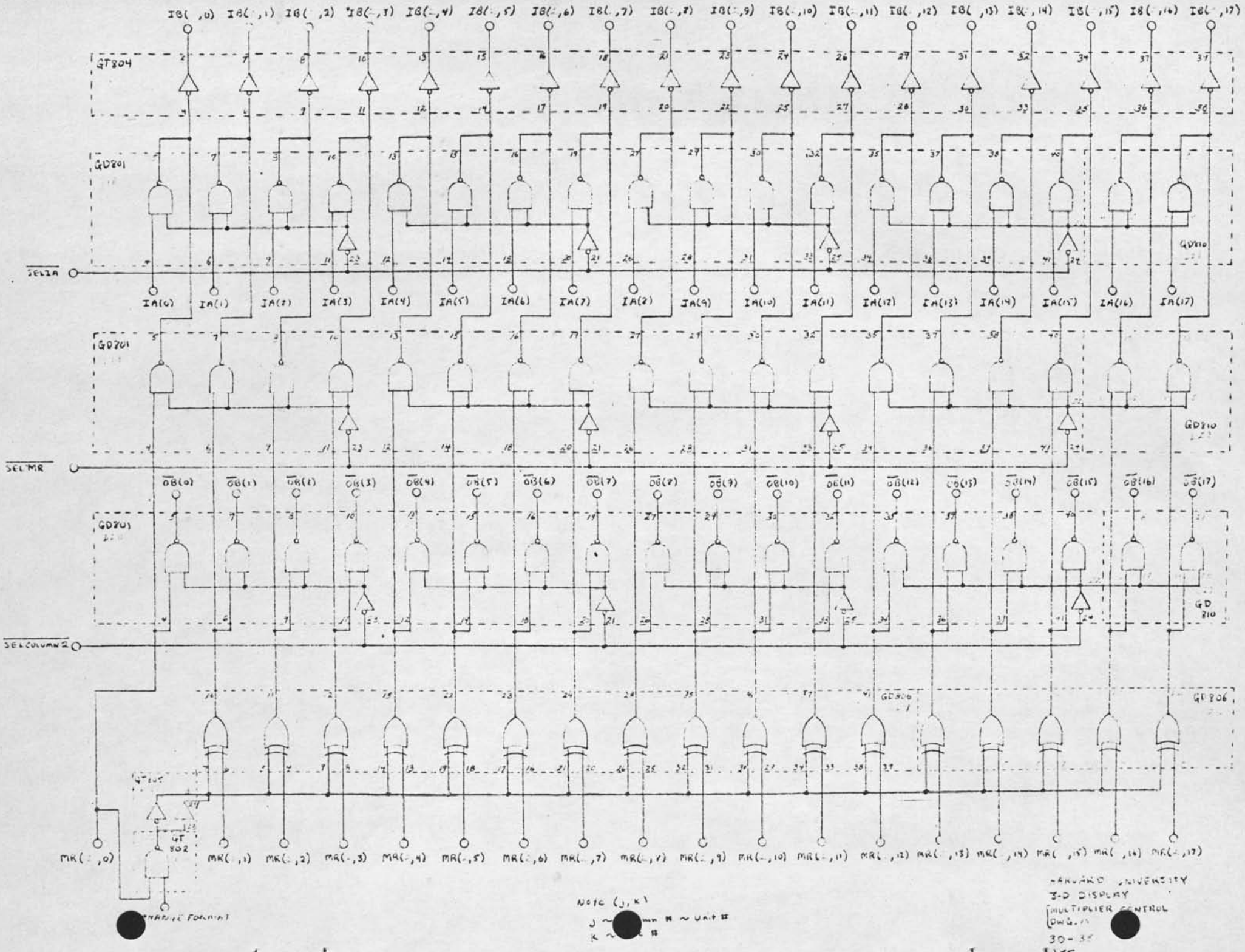

Note: (i,j)
i ~ ROW #
j ~ COLUMN # ~ UNIT #
SEL (0,0)
SEL (0,1)
SEL (0,2)
SEL (0,3)
SEL (1,0)
SEL (1,1)
SEL (1,2)
SEL (1,3)
SEL (2,0)
SEL (2,1)
SEL (2,2)
SEL (2,3)
SEL (3,0)
SEL (3,1)
SEL (3,2)
SEL (3,3)
EB → MR(i,j)
GTB01
IB → ROW 0
IB → ROW 1
IB → ROW 2
GTB01
GTB05
MR (0,0)
MR (0,1)
MR (0,2)
MR (0,3)
MR (1,0)
MR (1,1)
MR (1,2)
MR (1,3)
MR (2,0)
MR (2,1)
MR (2,2)
MR (2,3)
MR (3,0)
MR (3,1)
MR (3,2)
MR (3,3)
HARVARD UNIVERSITY
3-D Display
Register Control
30 - 34
CRT
Sepia 1 (2)

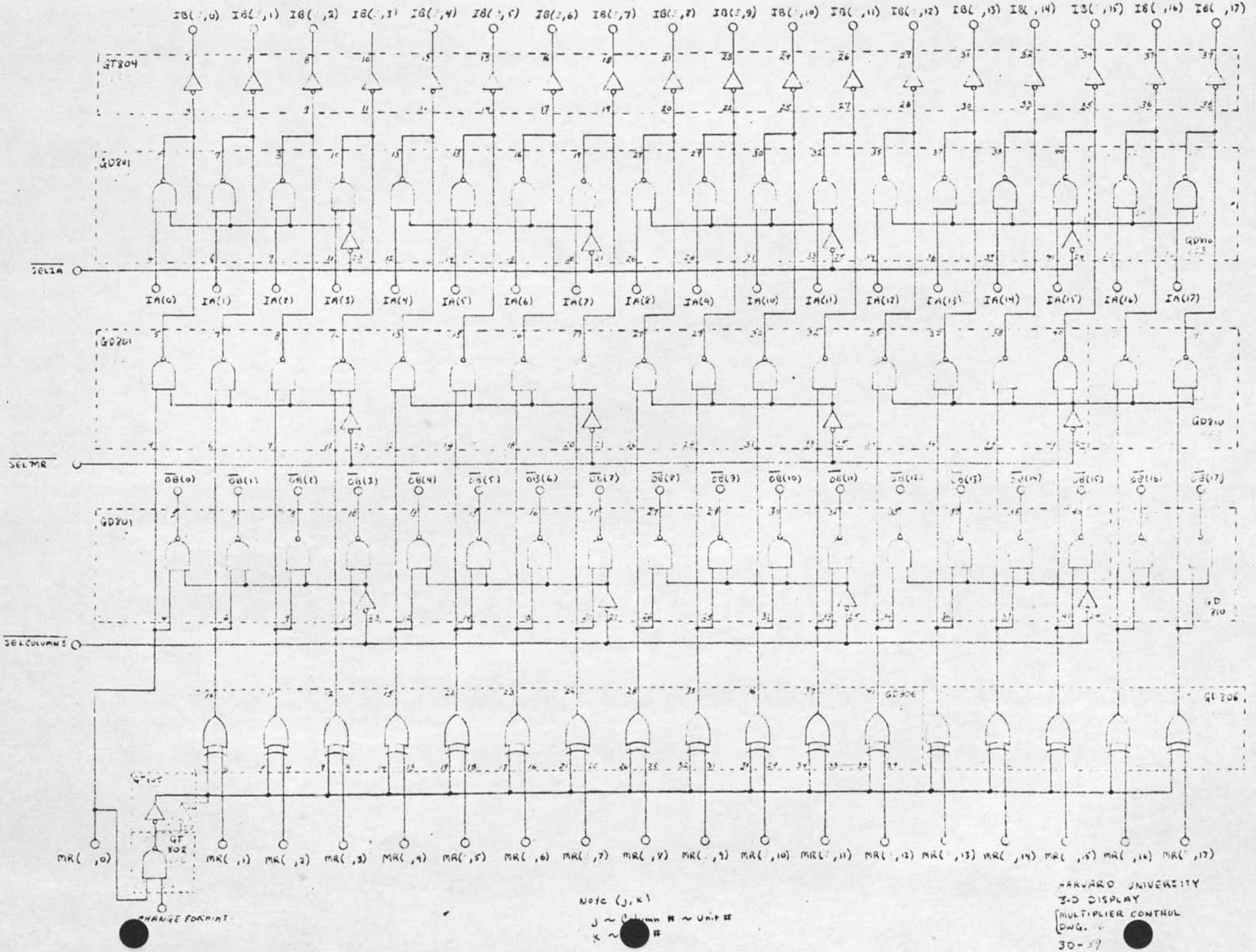


HARVARD UNIVERSITY
3D DISPLAY
[MULTIPLIER CONTROL
DWG. 12
30-35
C&S


IB(0,0) IB(0,1) IB(0,2) IB(0,3) IB(0,4) IB(0,5) IB(0,6) IB(0,7) IB(0,8) IB(0,9) IB(0,10) IB(0,11) IB(0,12) IB(0,13) IB(0,14) IB(0,15) IB(0,16) IB(0,17)
GT804
GD801
GD810
SELIA
IA(0) IA(1) IA(2) IA(3) IA(4) IA(5) IA(6) IA(7) IA(8) IA(9) IA(10) IA(11) IA(12) IA(13) IA(14) IA(15) IA(16) IA(17)
GD701
SELMR
OB(0) OB(1) OB(2) OB(3) OB(4) OB(5) OB(6) OB(7) OB(8) OB(9) OB(10) OB(11) OB(12) OB(13) OB(14) OB(15) OB(16) OB(17)
GD801
GD 810
SELCOMMAND
GD806
GT 802
MR(0,0) MR(0,1) MR(0,2) MR(0,3) MR(0,4) MR(0,5) MR(0,6) MR(0,7) MR(0,8) MR(0,9) MR(0,10) MR(0,11) MR(0,12) MR(0,13) MR(0,14) MR(0,15) MR(0,16) MR(0,17)
Note (J,K)
J ~ Co
K ~ Bit #
HARVARD UNIVERSITY
3-D DISPLAY
DWG. 13
3D-

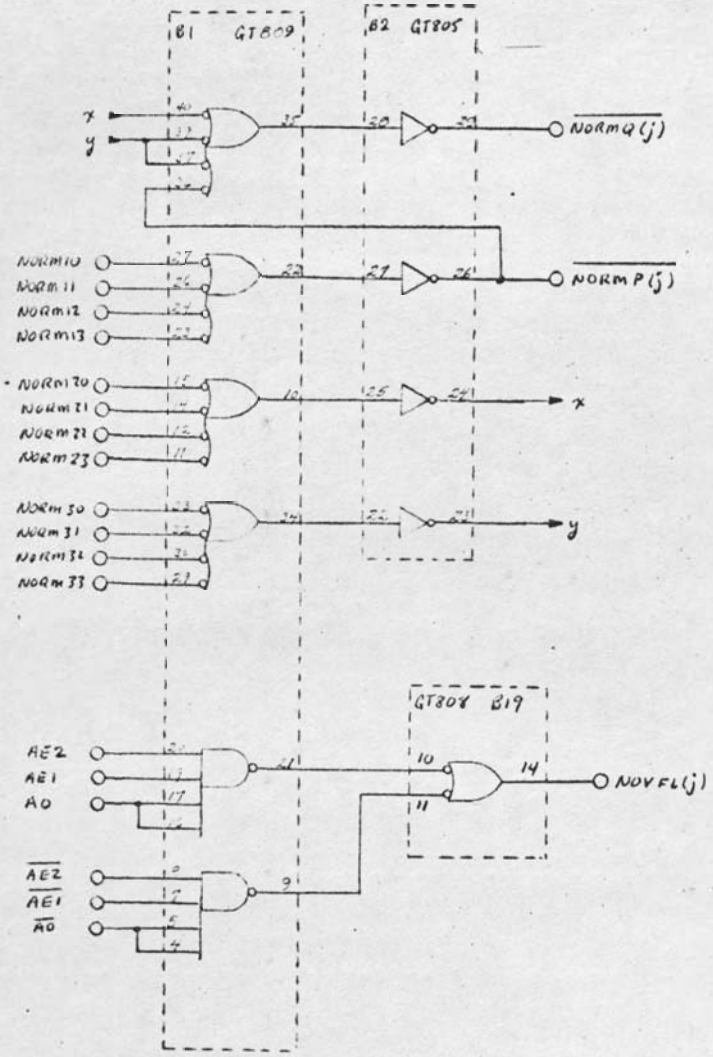

IB(1,0) IB(1,1) IB(,2) IB(1,3) IB(1,4) IB(1,5) IB(1,6) IB(1,7) IB(1,8) IB(1,9) IB(1,10) IB(1,11) IB(1,12) IB(1,13) IB(,14) IB(,15) IB(,16) IB(,17)
GT804
GD801
GD810
SELIA
IA(0) IA(1) IA(2) IA(3) IA(4) IA(5) IA(6) IA(7) IA(8) IA(9) IA(10) IA(11) IA(12) IA(13) IA(14) IA(15) IA(16) IA(17)
GD801
GD810
SELMR
OB(0) OB(1) OB(2) OB(3) OB(4) OB(5) OB(6) OB(7) OB(8) OB(9) OB(10) OB(11) OB(12) OB(13) OB(14) OB(15) OB(16) OB(17)
GD801
GD 810
SELCOLUMN 1
GD806
GT 806
GT 802
MR(1,0) MR(1,1) MR(1,2) MR(1,3) MR(1,4) MR(1,5) MR(1,6) MR(1,7) MR(1,8) MR(1,9) MR(1,10) MR(1,11) MR(1,12) MR(1,13) MR(1,14) MR(1,15) MR(1,16) MR(1,17)
CHANGE FORMAT
Note (J,K)
J ~ Unit #
K ~ #
HARVARD UNIVERSITY
3-D DISPLAY
MULTIPLIER CONTROL
DWG. 14
30-37
285

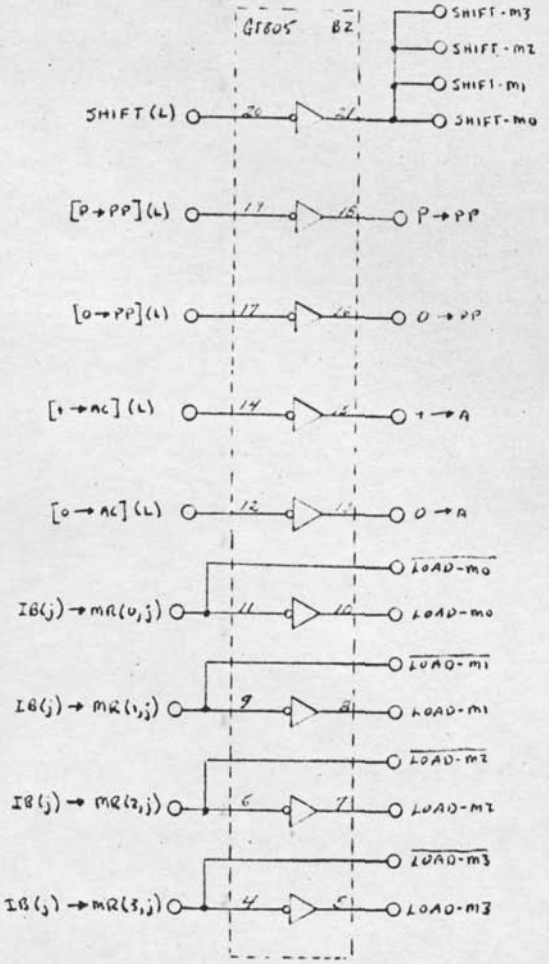

GT804
GD801
GD810
SELIA
IA(0) IA(1) IA(2) IA(3) IA(4) IA(5) IA(6) IA(7) IA(8) IA(9) IA(10) IA(11) IA(12) IA(13) IA(14) IA(15) IA(16) IA(17)
GD801
GD810
SELMR
OB(0) OB(1) OB(2) OB(3) OB(4) OB(5) OB(6) OB(7) OB(8) OB(9) OB(10) OB(11) OB(12) OB(13) OB(14) OB(15) OB(16) OB(17)
GD801
GD 810
SELCOLUMN2
GD806
GT806
GT802
MR(-,0) MR(-,1) MR(-,2) MR(-,3) MR(-,4) MR(-,5) MR(-,6) MR(-,7) MR(-,8) MR(-,9) MR(-,10) MR(-,11) MR(-,12) MR(-,13) MR(-,14) MR(-,15) MR(-,16) MR(-,17)
HARVARD UNIVERSITY
3-D DISPLAY
MULTIPLIER CONTROL


IB(,0) IB(,1) IB(,2) IB(,3) IB(,4) IB(,5) IB(,6) IB(,7) IB(,8) IB(,9) IB(,10) IB(,11) IB(,12) IB(,13) IB(,14) IB(,15) IB(,16) IB(,17)
GT804
GD801
GD810
SELIA
IA(0) IA(1) IA(2) IA(3) IA(4) IA(5) IA(6) IA(7) IA(8) IA(9) IA(10) IA(11) IA(12) IA(13) IA(14) IA(15) IA(16) IA(17)
GD801
GD810
SELMR
OB(0) OB(1) OB(2) OB(3) OB(4) OB(5) OB(6) OB(7) OB(8) OB(9) OB(10) OB(11) OB(12) OB(13) OB(14) OB(15) OB(16) OB(17)
GD801
GD810
SELCOLUMNS
GD806
GT806
GT802
MR(,0) MR(,1) MR(,2) MR(,3) MR(,4) MR(,5) MR(,6) MR(,7) MR(,8) MR(,9) MR(,10) MR(,11) MR(,12) MR(,13) MR(,14) MR(,15) MR(,16) MR(,17)
CHANGE FORMAT
Note (J, K)
J ~ Column # ~ Unit #
K ~ #
HARVARD UNIVERSITY
3-D DISPLAY
MULTIPLIER CONTROL
DWG.
30-
225

Repeat 4, for j = 0, 1, 2, 3

logic goes in Boxes A_j & B_j





$0 \rightarrow MR(0,j)$, $0 \rightarrow MR(1,j)$, $0 \rightarrow MR(2,j)$, $0 \rightarrow MR(3,j)$ become $0 \rightarrow M0$, $0 \rightarrow M1$, $0 \rightarrow M2$, $0 \rightarrow M3$

$IB(j,n)$ becomes IBn, for $n = 0, \ldots, 17$.

MRn becomes MR(j,m) $(n = 0, \ldots, 17)$

↑ from A_j & B_j

↑ to control

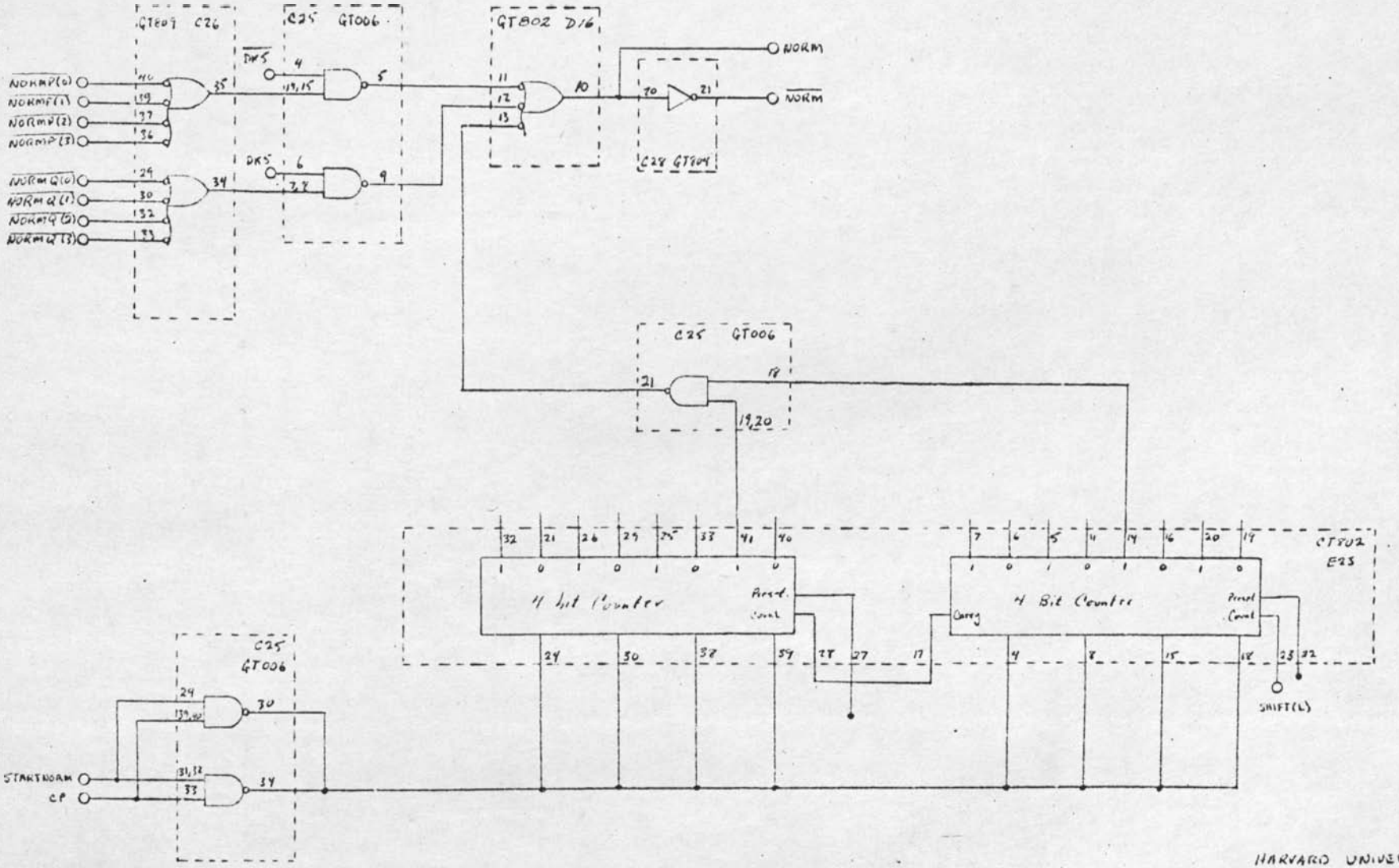
↑ from Control

↑ to A_j & B_j

HARVARD UNIVERSITY
3-D DISPLAY
[MULTIPLIER CONTROL
DWG

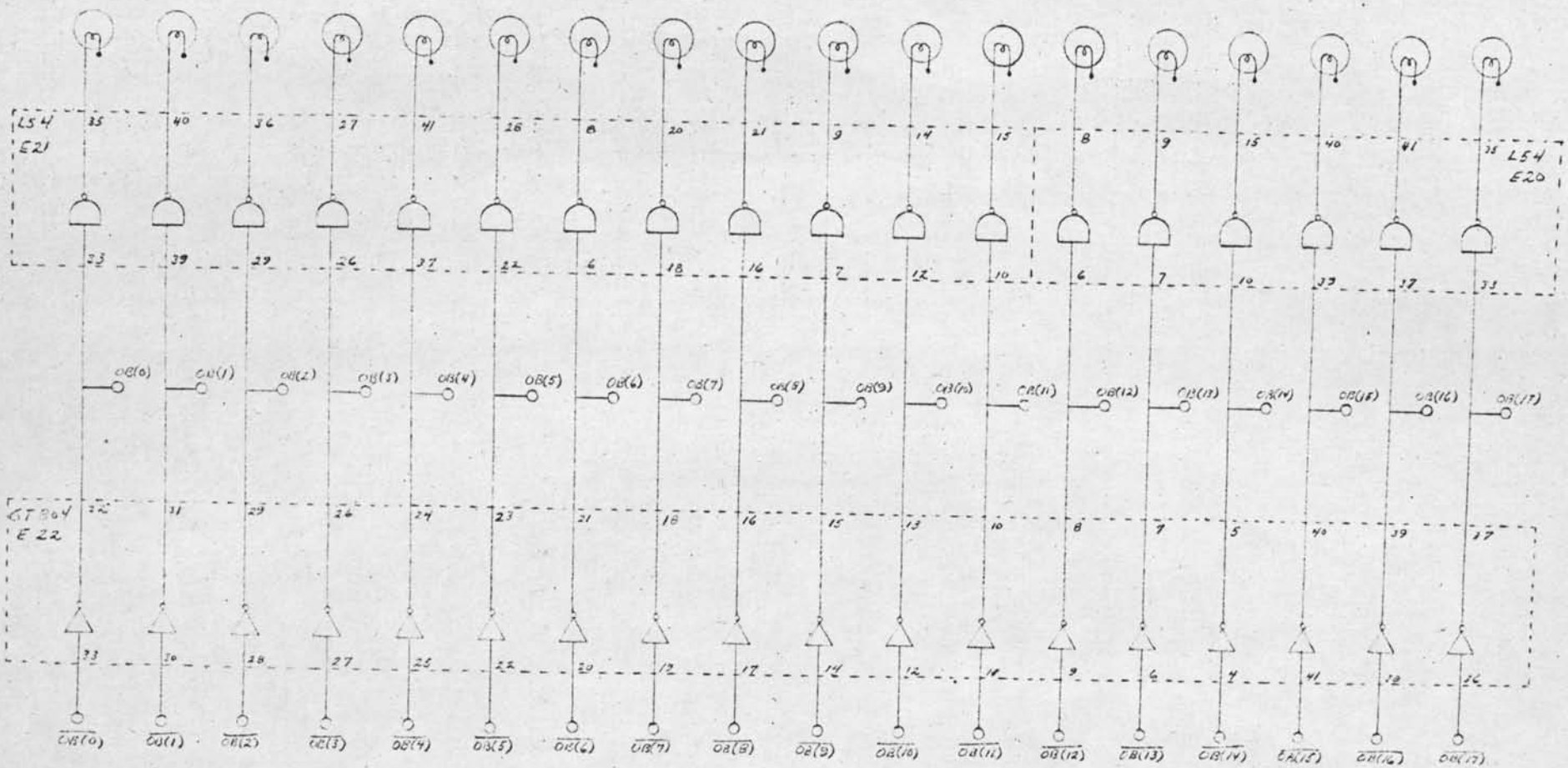
3D-40

Normalize & overflow first level nets, [illegible] forms of A_j & B_j to control.



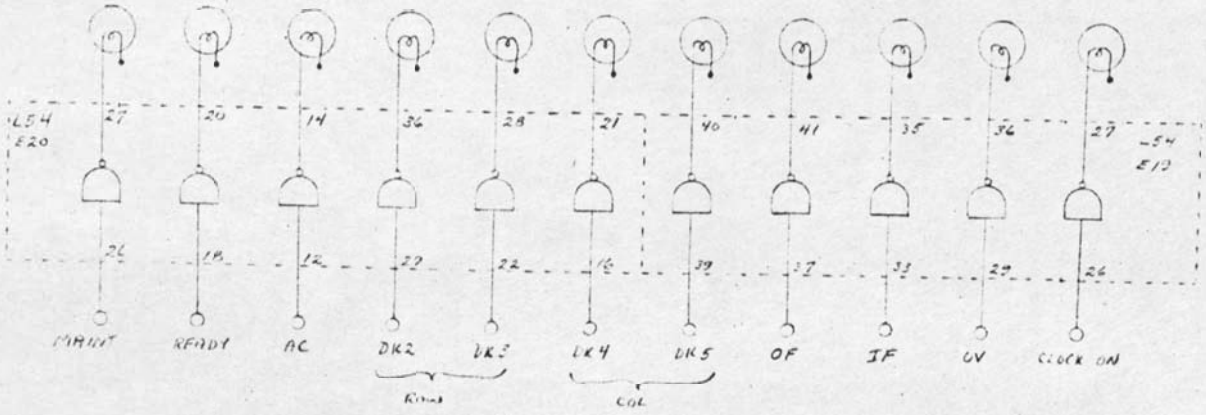
HARVARD UNIVERSITY
3-D DISPLAY
MULTIPLEX CONTROL
DWG #

3D-41



HARVARD UNIVERSITY
3D DISPLAY

3D-42
OUTPUT BUSS & LIGHTS

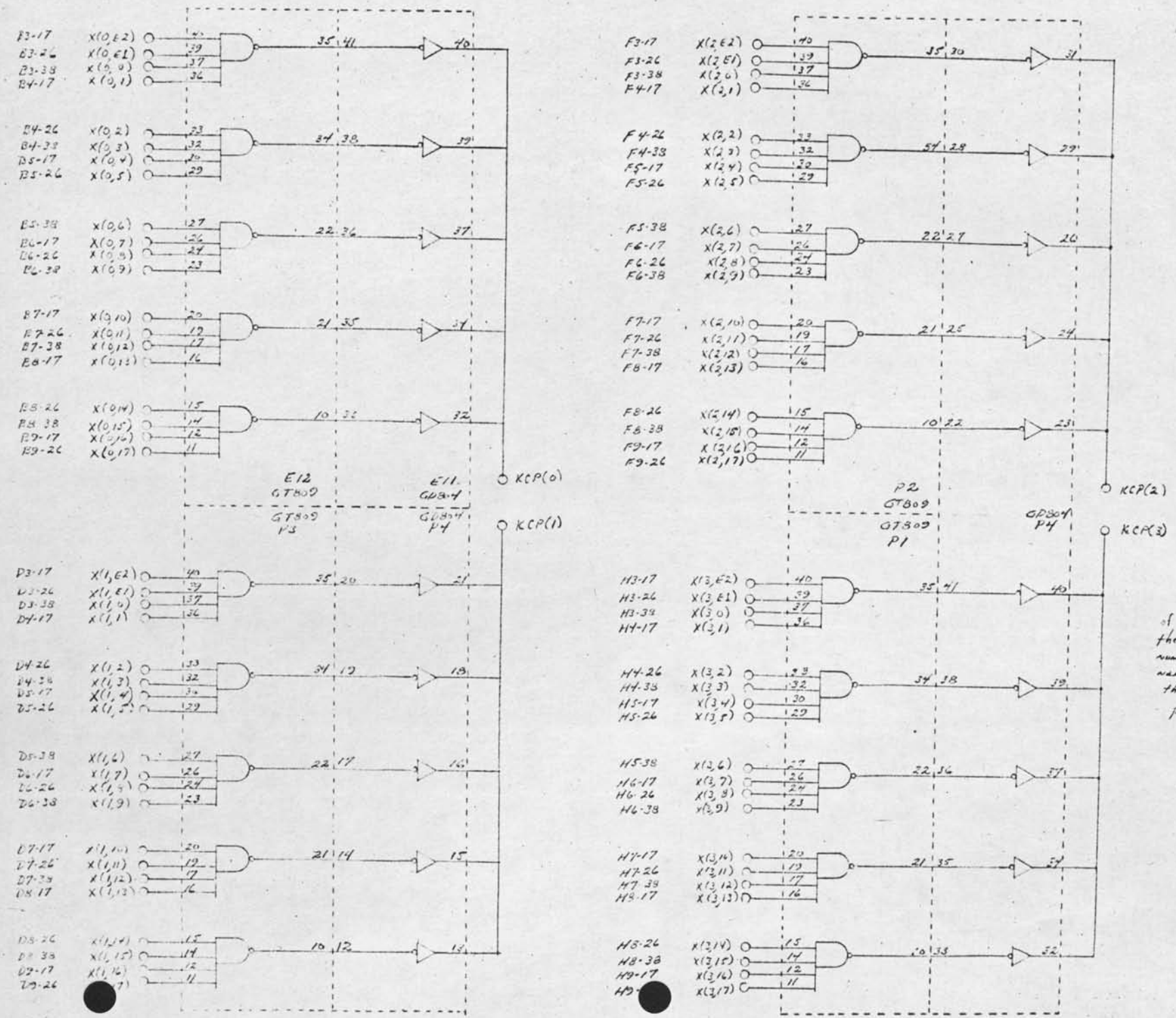


HARVARD UNIVERSITY

3D DISPLAY

30-43

LIGHT DRIVERS


E12
GT809
P3
E11
GP804
P4
KCP(0)
KCP(1)
P2
GT809
P1
GP804
P4
KCP(2)
KCP(3)
NOTE: X(i,j) is the
of the xor of the jth bit
the GT809 as columns i. The
number to the left of the
name is the pin location of
the xor result.
P is the power supply cage
MARYLAND UNIVERSITY
3-D DISPLAY
ADDER ZERO DETECTION
DWG 3D-44

APPENDIX 3

Prints for the Clipping Divider

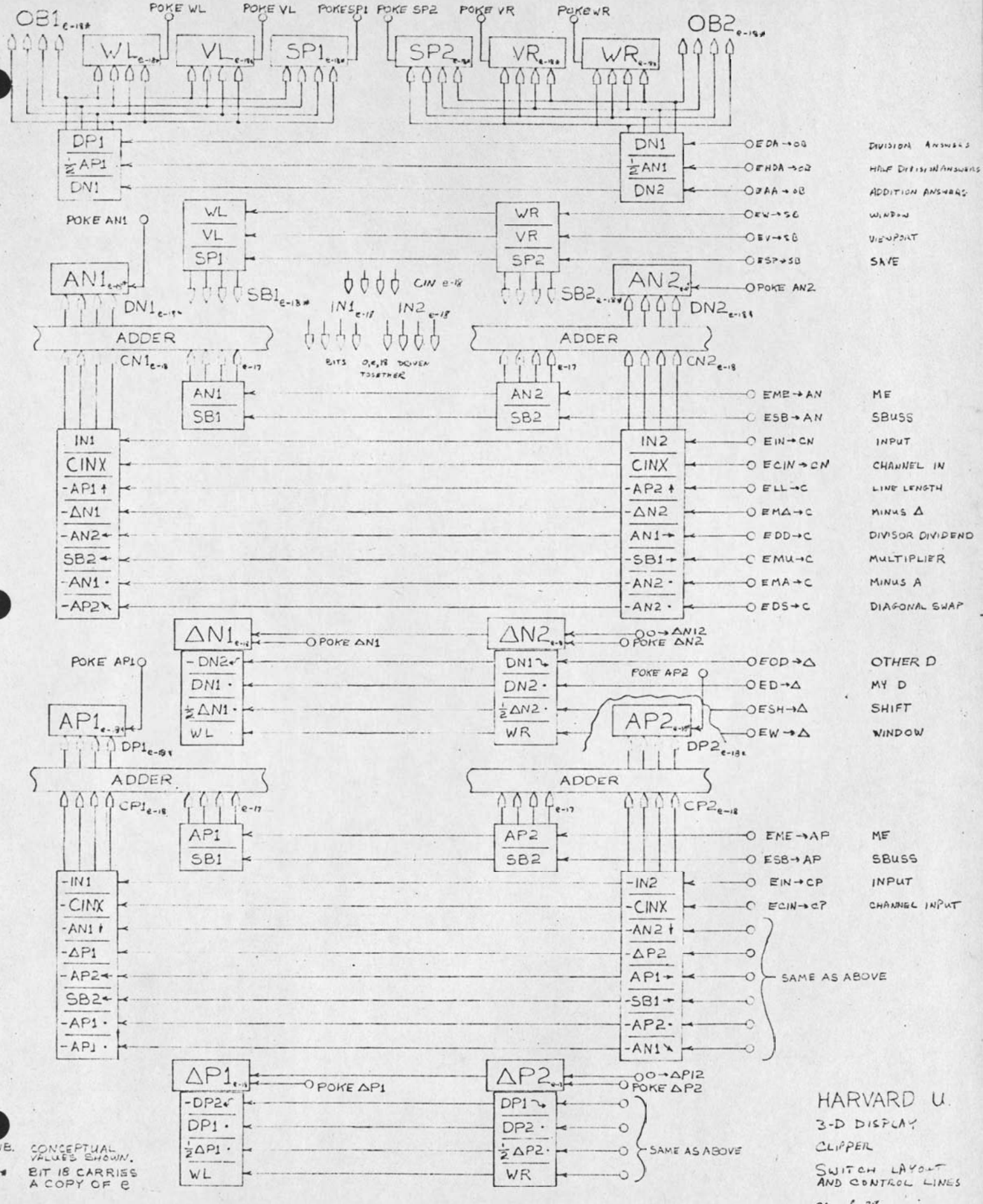
Part 1 Clipper Data Paths

Clipper Data

Print No.	Date	Title
C1	7/30/68	Switch layout and control lines
C2	11/11/67	3-Display clipper card placement
C3		Storage register gating
C4	8/20/68	storage register
C5	8/20/68	window - storage register
C6	8/20/68	viewport - storage register
C7	8/20/68	dn1 register and gates
C8	8/20/68	A and d registers zero detection
C9	8/20/68	C-buss switch
C10	8/20/68	dp1 register and gates
C11	8/20/68	A and d registers zero detection
C12	8/20/68	c-buss switch
C13	8/20/68	dn2 register and gates
C14	8/20/68	A and d registers zero detection
C15	8/20/68	c-buss switch n2
C16	8/20/68	dp2 register and gates
C17	8/20/68	A and d registers zero detection
C18	8/20/68	c-buss switch p2
C19	8/20/68	clipper adder n1
C20	11/3/67	clipper adder p1
C21	11/3/67	clipper adder n2
C22	11/3/67	clipper adder p2
C23	8/20/68	xor detection

Clipper Data

C24	8/20/68	xor detection
C25	8/20/68	xor detection
C26	8/20/68	xor detection
C27	8/20/68	clipper - zero nets
C28	8/20/68	cable connections



NB. CONCEPTUAL VALUES SHOWN.

* BIT 18 CARRIES A COPY OF e

HARVARD U.

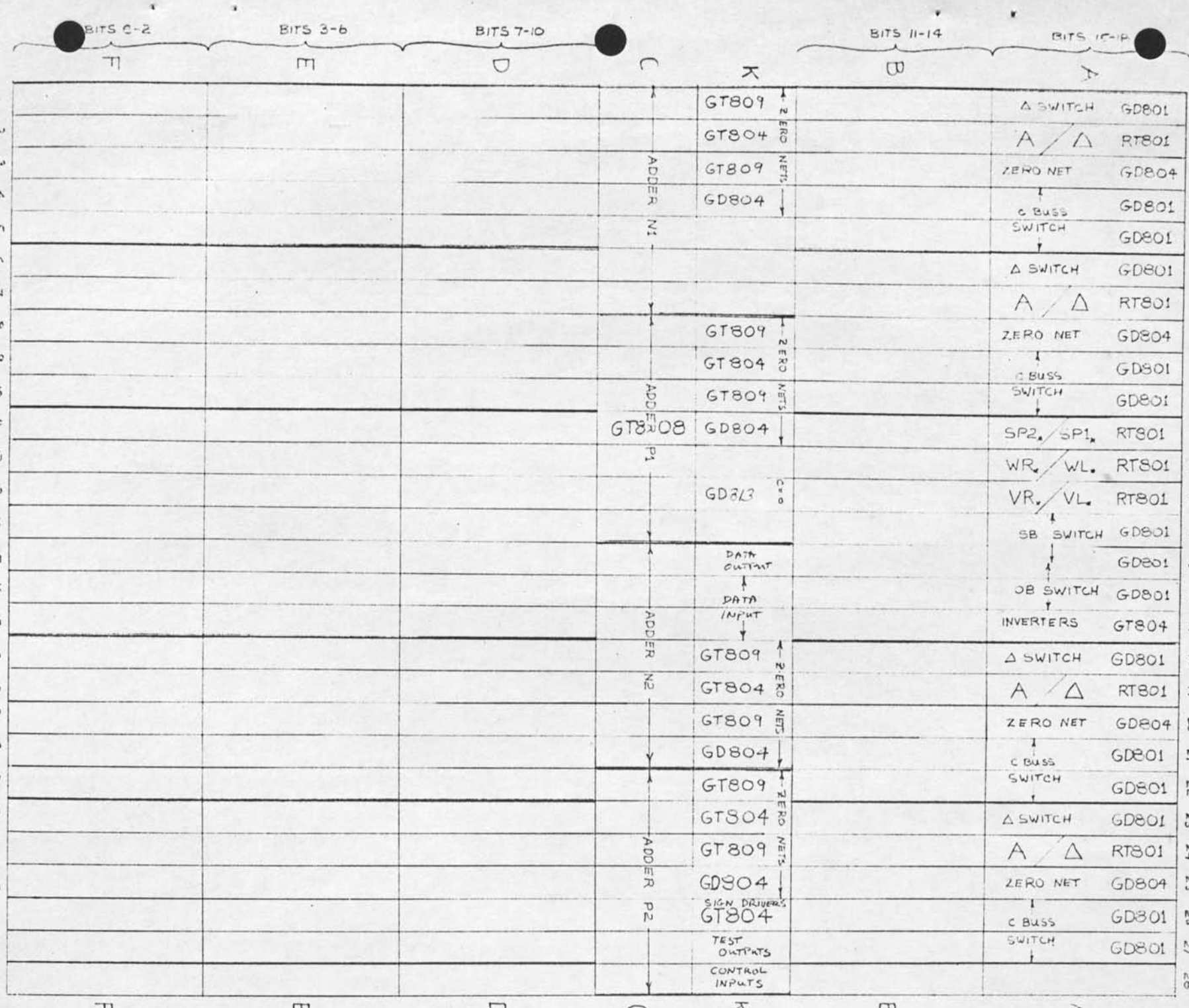
3-D DISPLAY

CLIPPER

SWITCH LAYOUT AND CONTROL LINES

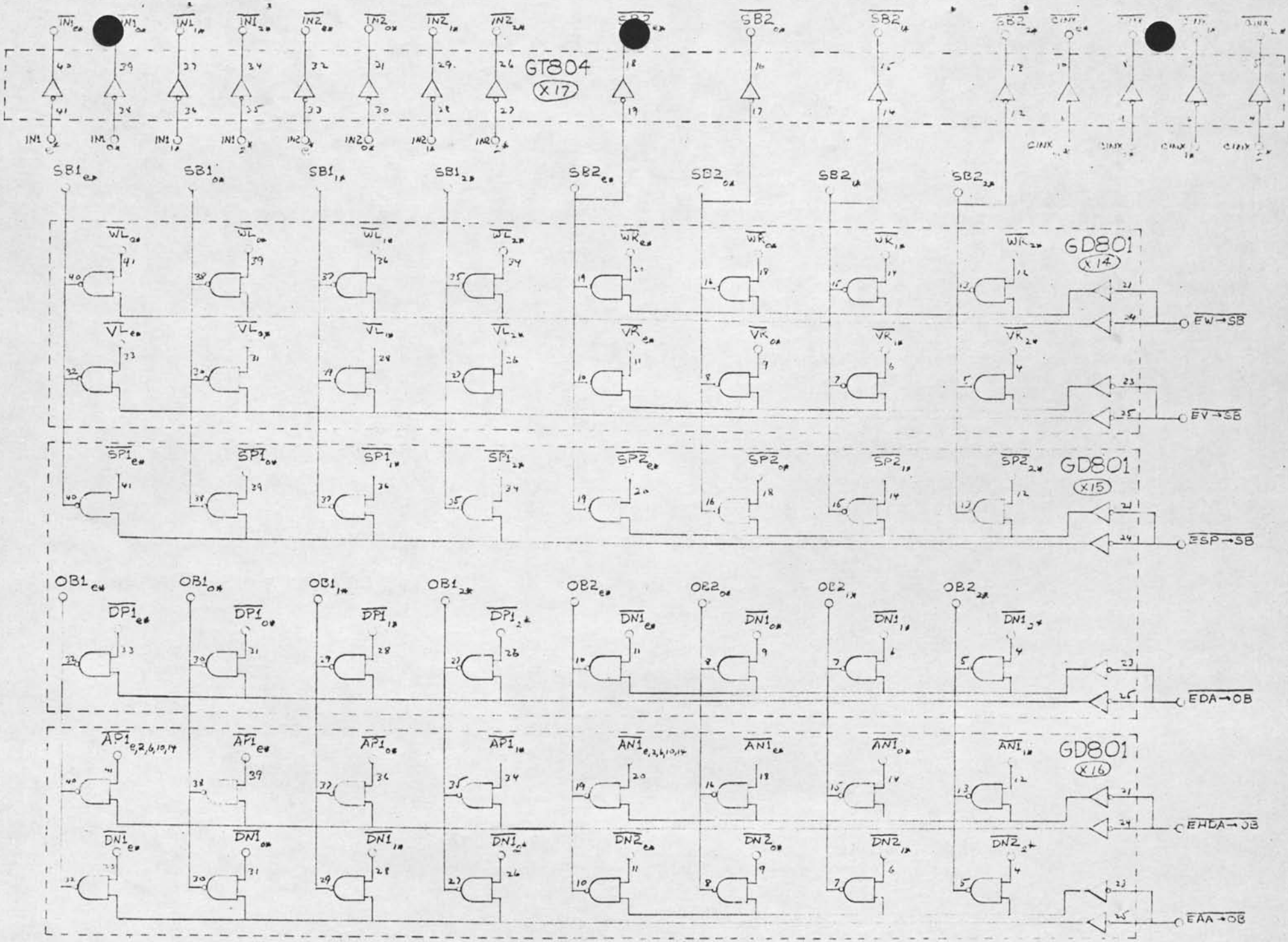
C1 of 28

SET REVISED 30 JULY 1968



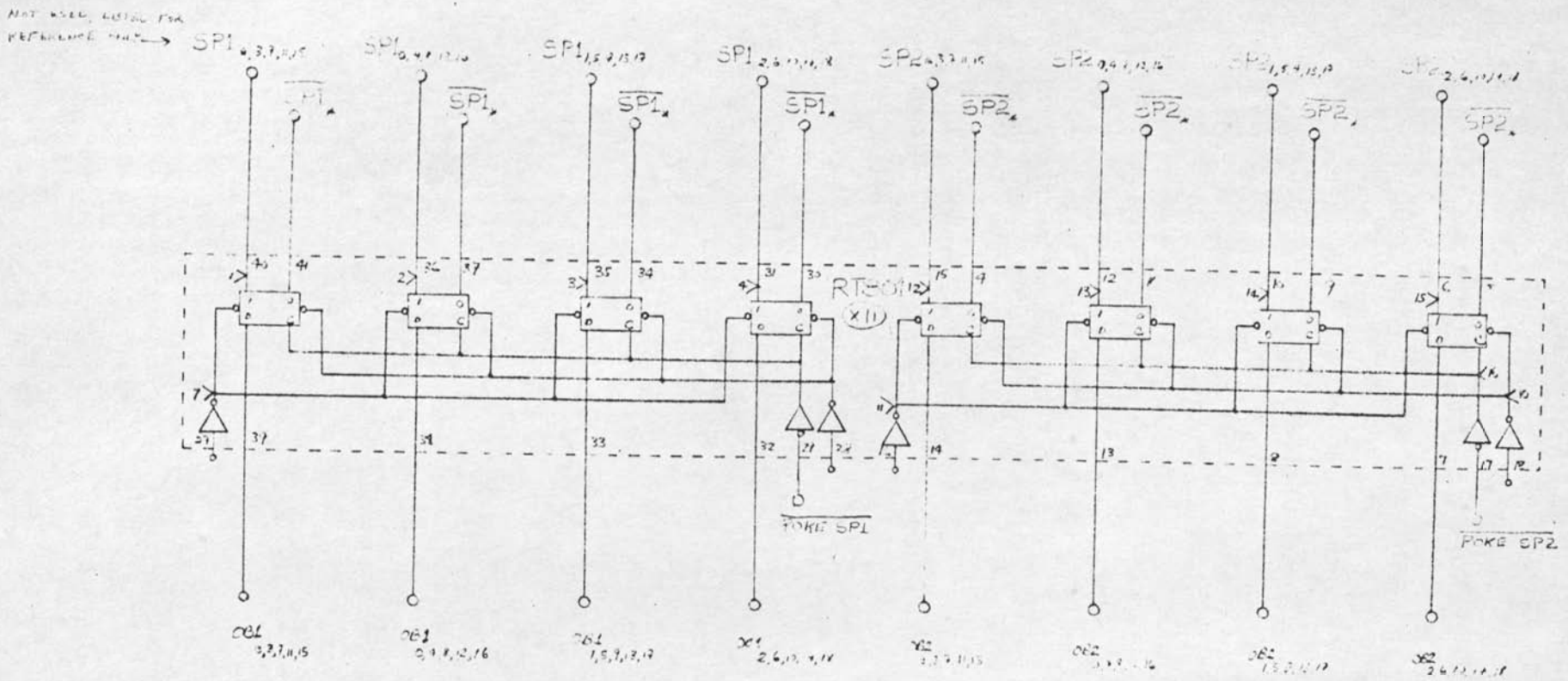
HARVARD UNIVERSITY
3-D DISPLAY CLIPPER
CARD PLACEMENT

NOV 11, 1967



HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
STORAGE REGISTER CARDS

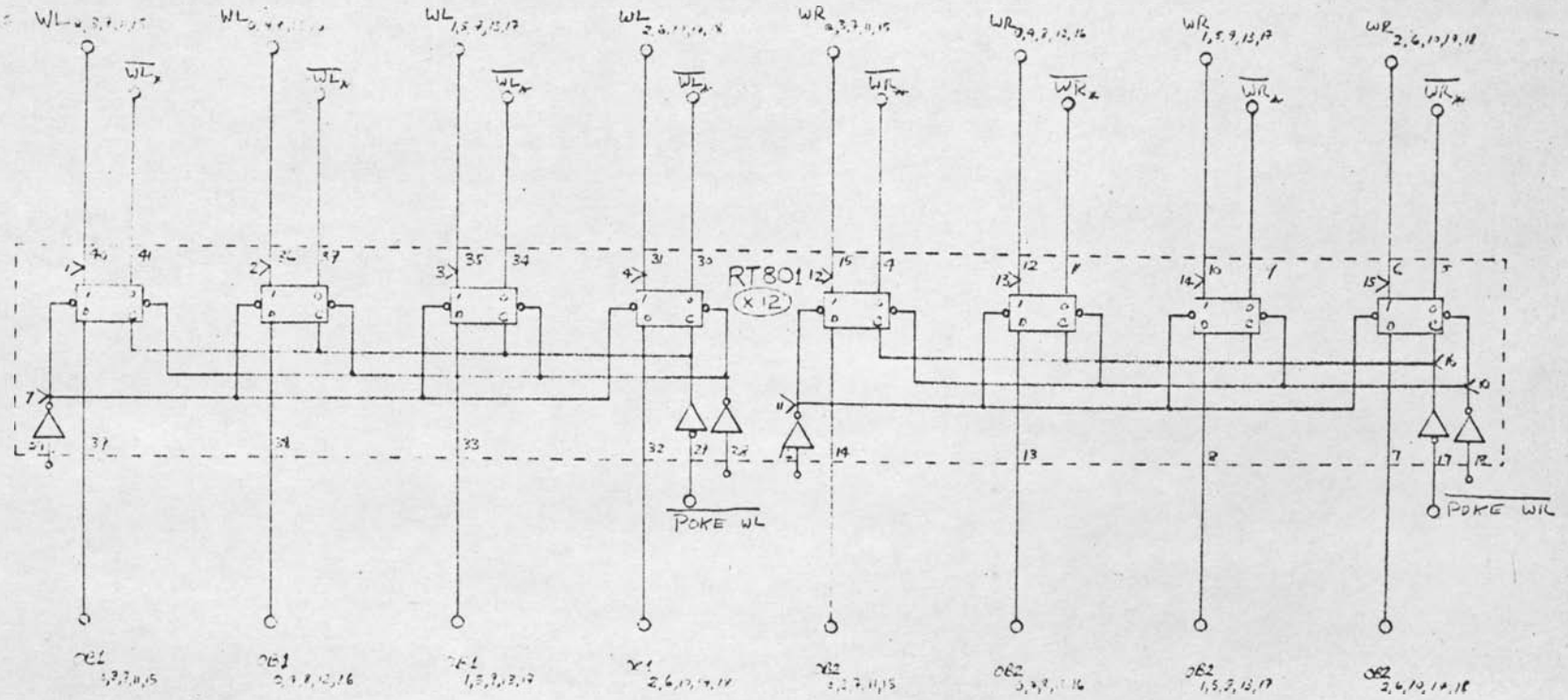
C3



* SAME BIT DESIGNATIONS AS ABOVE

HARVARD UNIVERSITY
3-D DISPLAY

CLIPPER
SP
STORABLE REGISTER
04

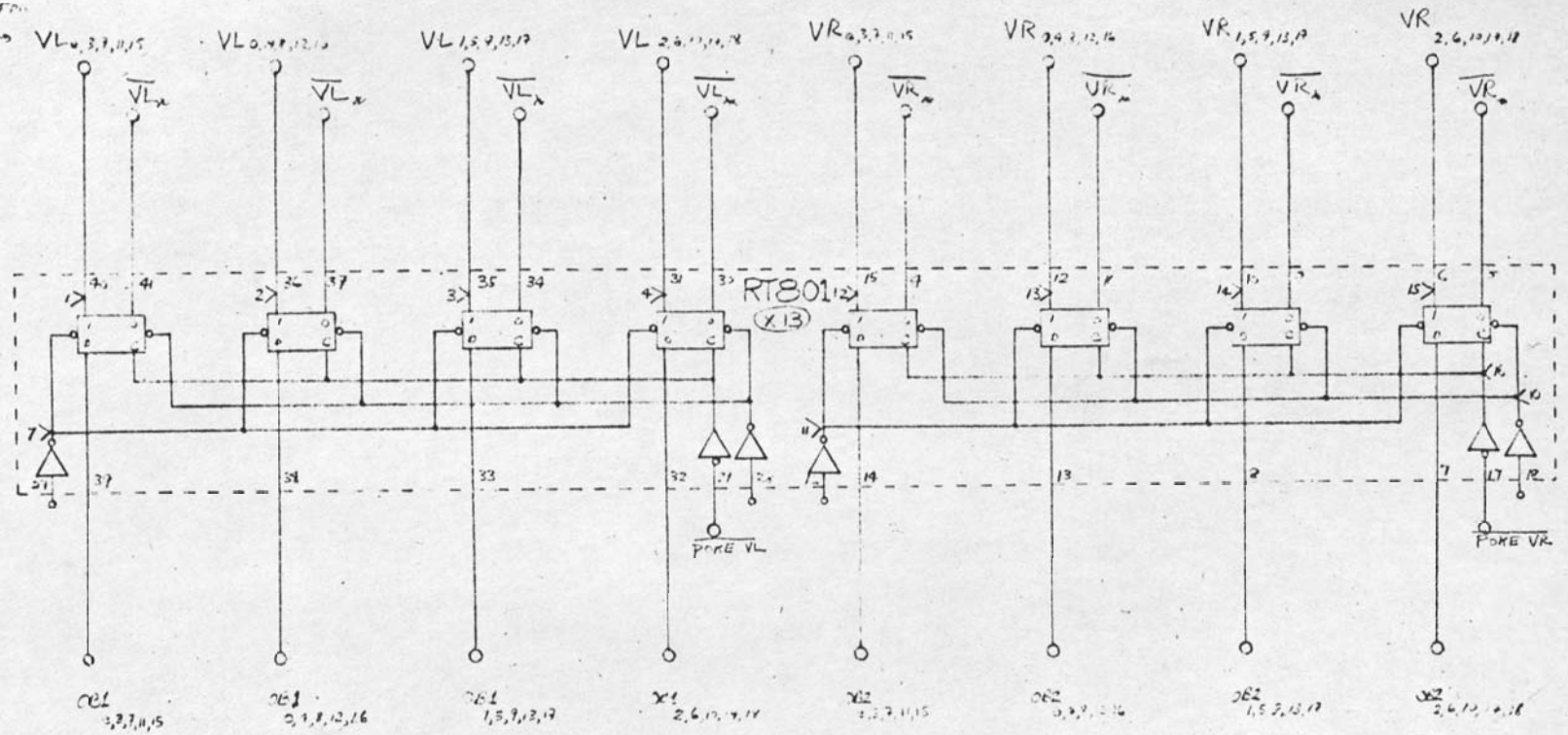


* SAME BIT DESIGNATIONS AS ABOVE

HARVARD UNIVERSITY
3-D DISPLAY

CLIPPER
WINDOW
STORAGE REGISTER

.05

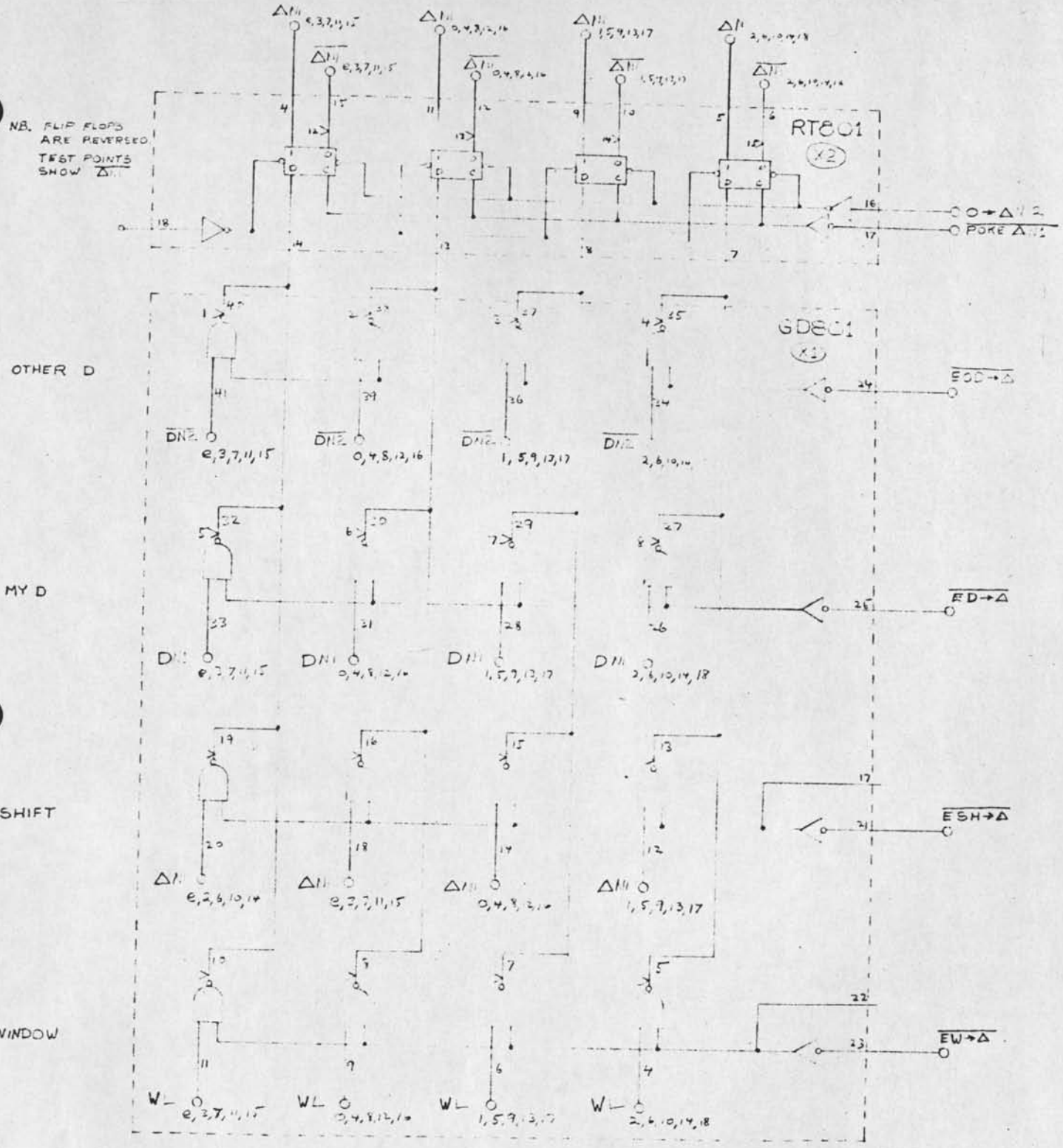


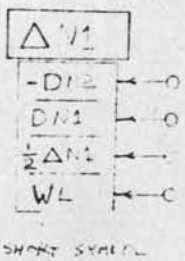
* SAME LIT DESIGNATIONS AS ABOVE

HARVARD UNIVERSITY
3-D DISPLAY

CLIPPER
VIEWPOINT
STORAGE REGISTER

C6



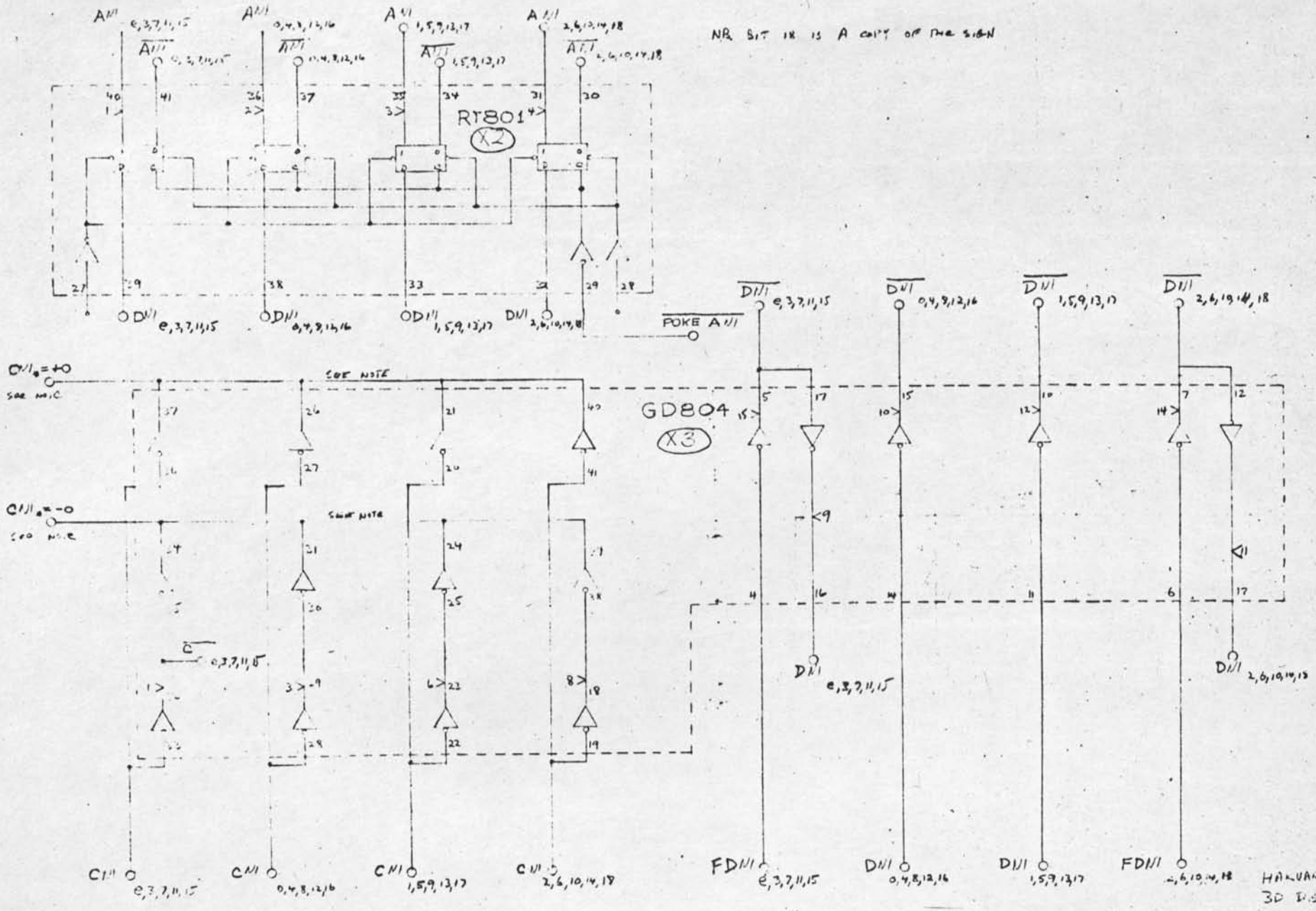


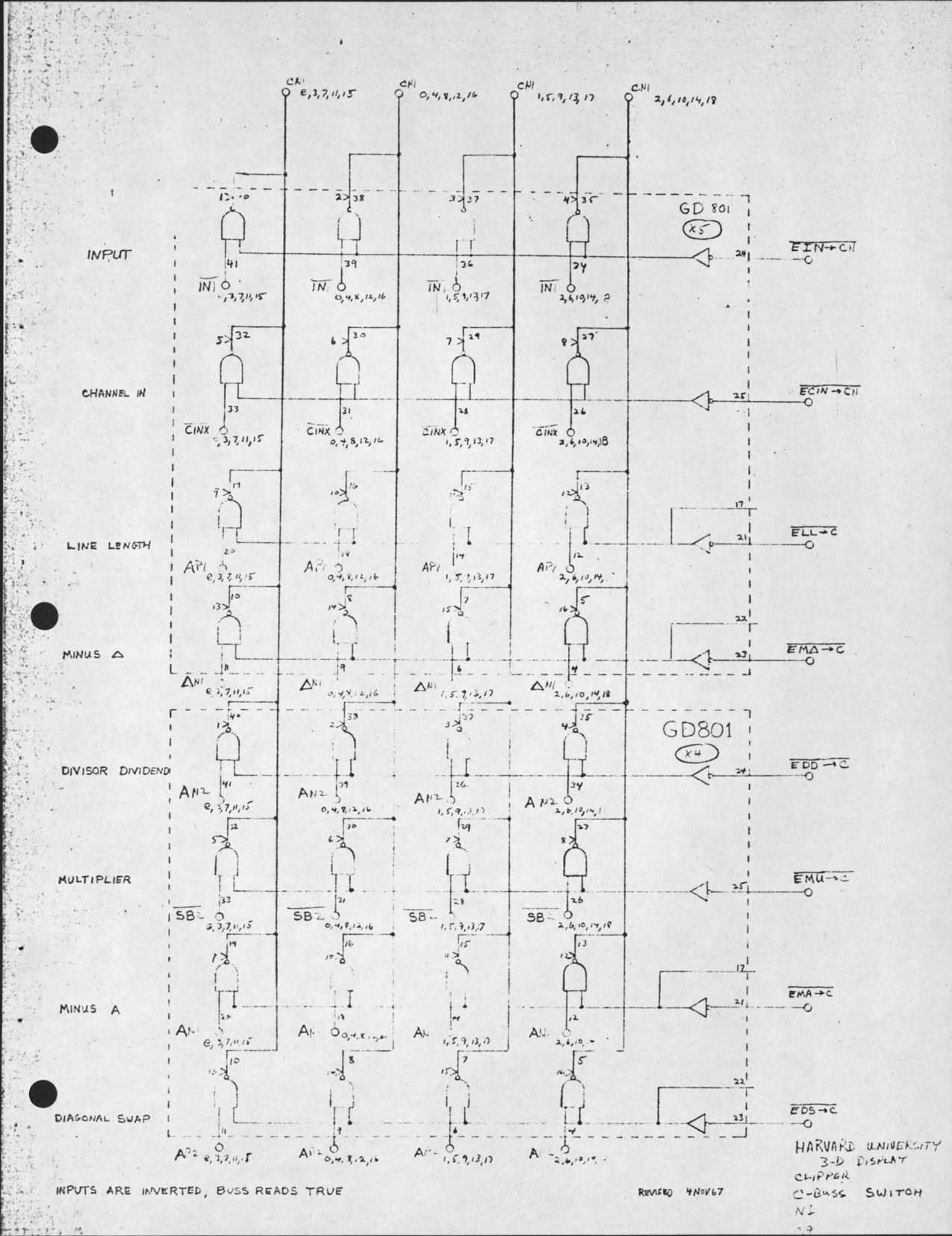
SHORT SYMBOL

NB. BECAUSE Δ FLIP FLOPS ARE REVERSED, WHAT GOES INTO THE SWITCH IS THE SAME AS WHAT COMES OUT OF Δ.

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER

Δ V1 REGISTER AND GATES

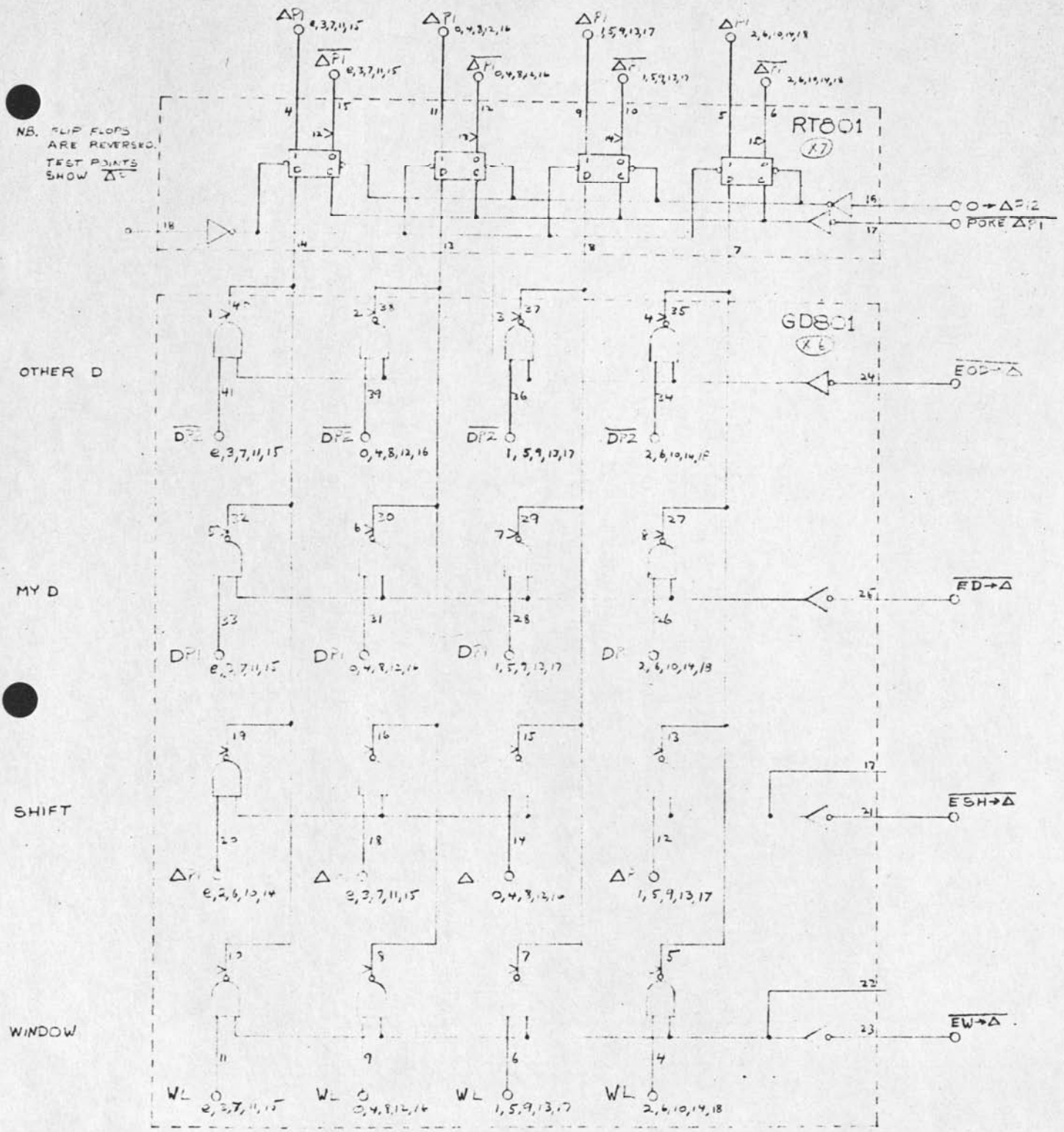

NB BIT 18 IS A COPY OF THE SIGN
RT801
X2
GD804
X3
POKE A
SEE NOTE
SEE NOTE
SEE NOTE
SEE NOTE
C
NB. C =0 ARE THREE LINES WITH SUBSCRIPTS e-5, 6-11, 12-17. E.G. CN1e-5=0 THE FOUR WIRES ON THIS DRAWING MARKED "SEE NOTE" ARE SCHEMATIC. (STRAIGHT BITS OF THE WIRES TOGETHER AS FIRST WIRES 6-11, THEN 12-17.
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
A AND Δ REGISTERS
ZERO DETECTION NOTE
C8 C11 C14 C17

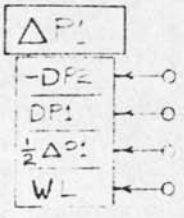


INPUTS ARE INVERTED, BUSS READS TRUE

REVISED 4NOV67

HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER
C-BUSS SWITCH
N2



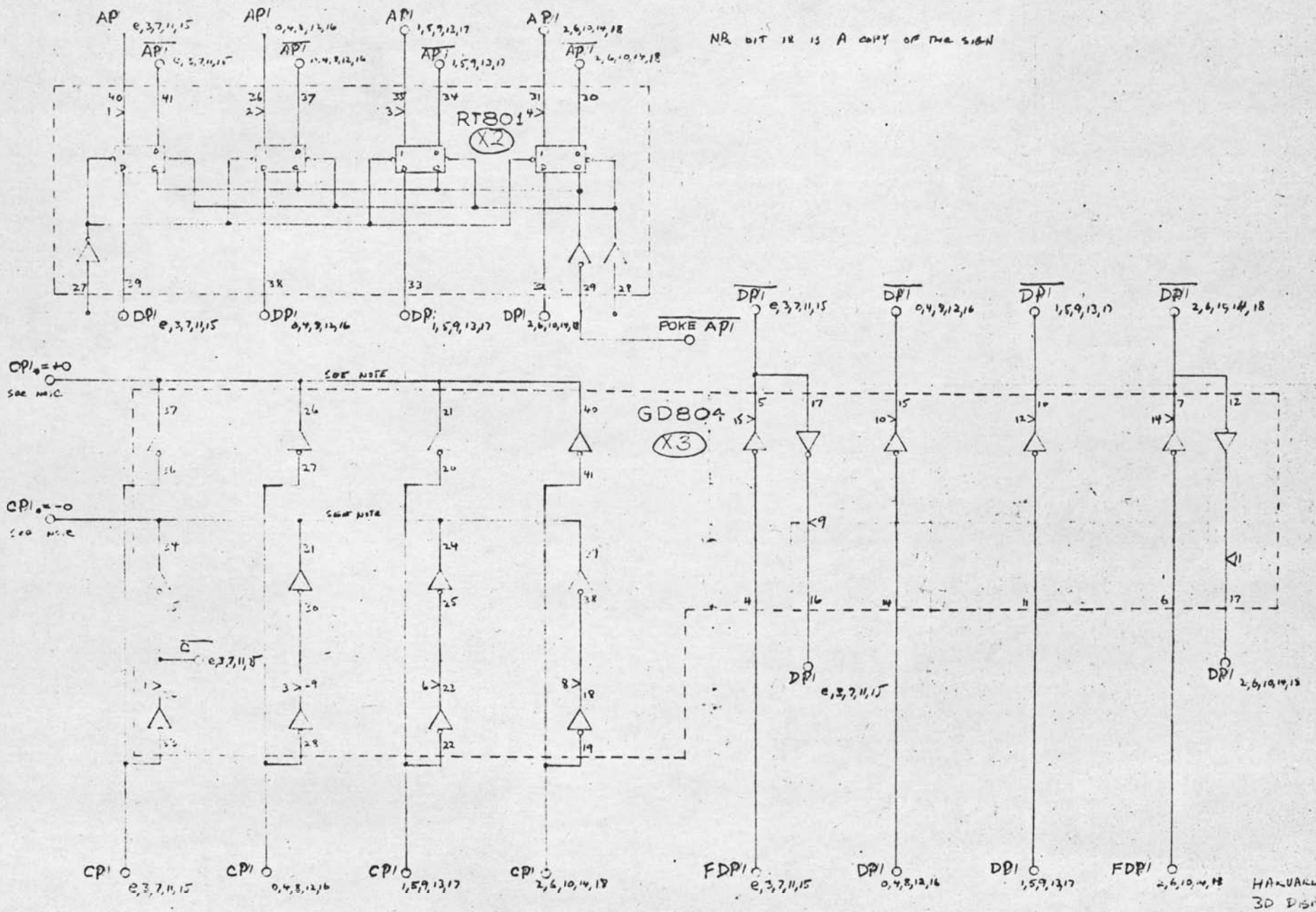


SHORT SYMBOL

NB. BECAUSE Δ FLIP FLOPS ARE REVERSED, WHAT GOES INTO THE SWITCH IS THE SAME AS WHAT COMES OUT OF Δ.

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER

ΔF REGISTER AND GATES



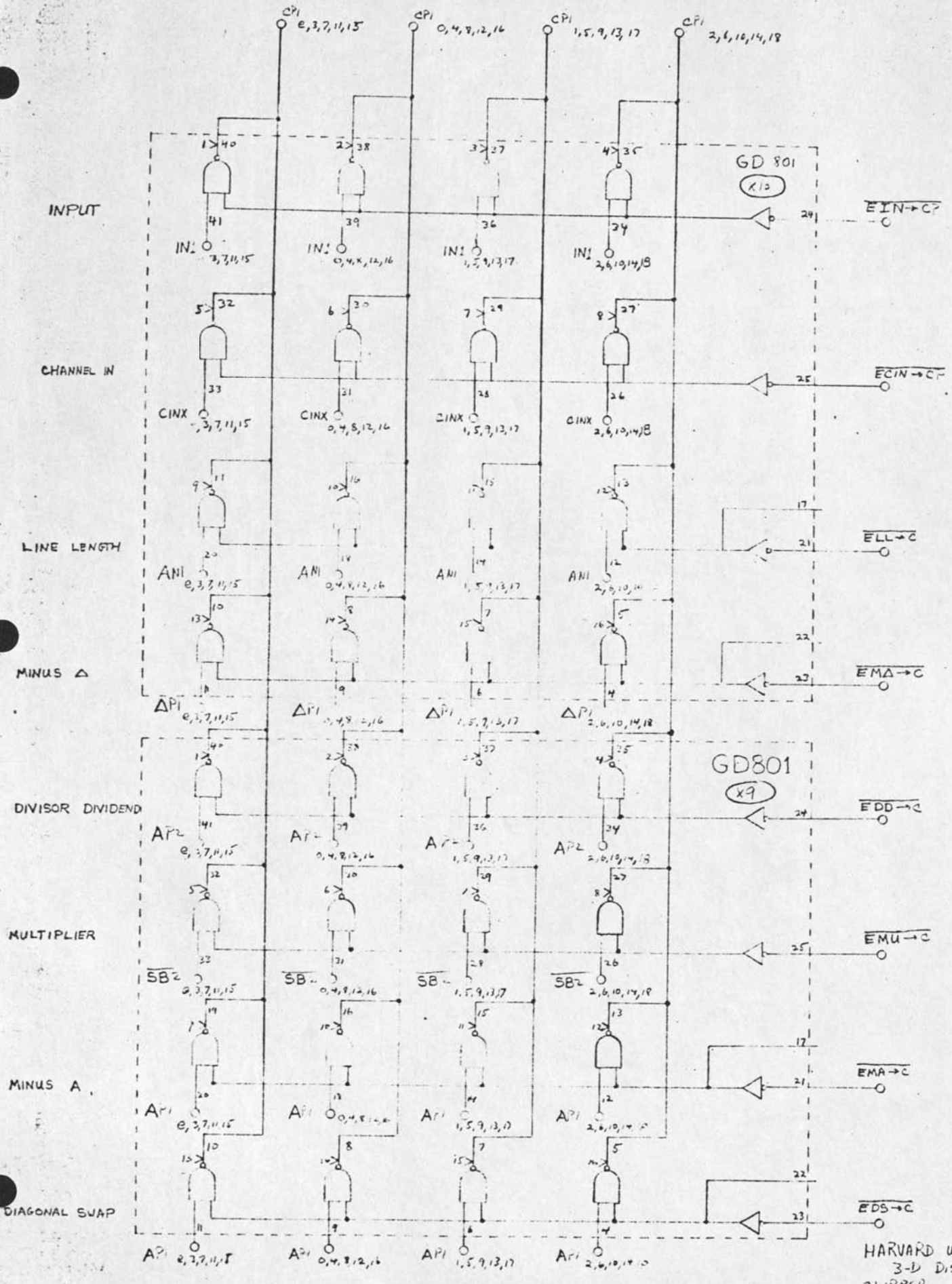
NB. MSB IS A COPY OF THE SIGN

NB. C#=0 ARE THREE LINES WITH SUBSCRIPTS e-5, 6-11, 12-17. D.G. CN1e-5=0 THE FOUR WIRES ON THIS DRAWING MARKED "SEE NOTE" ARE SCHEMATIC. PHYSICALLY BITS e-5 ARE WIRED TOGETHER AS ARE BITS 6-11, AND 12-17.

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER

A AND Δ REGISTERS
ZERO DETECTION NETS

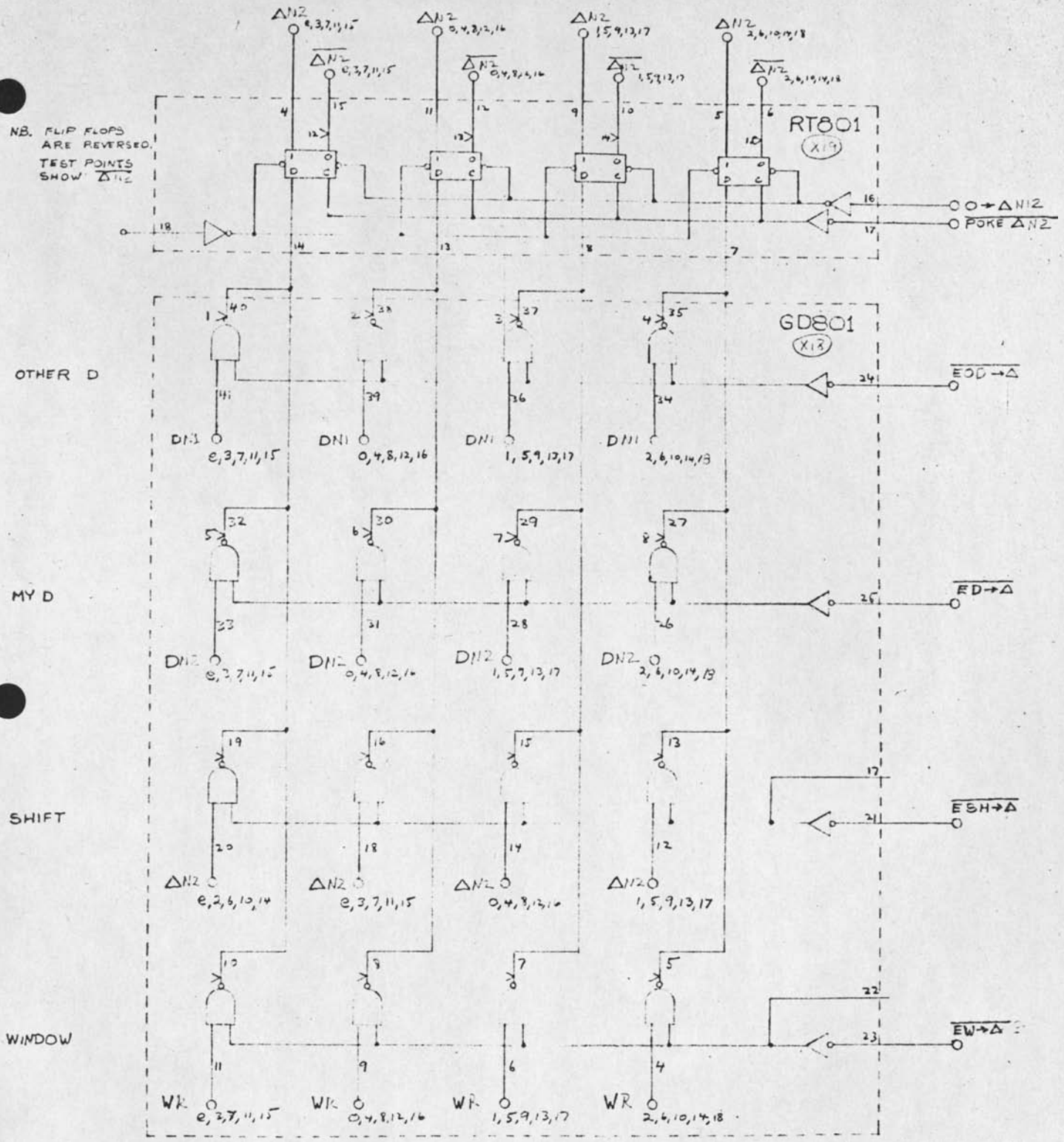
C11 C14 C17

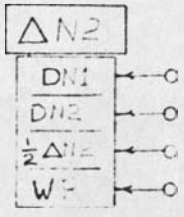


INPUTS ARE INVERTED, BUSS READS TRUE

REVISED 4NOV67

HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER
C-BUSS SWITCH
P1



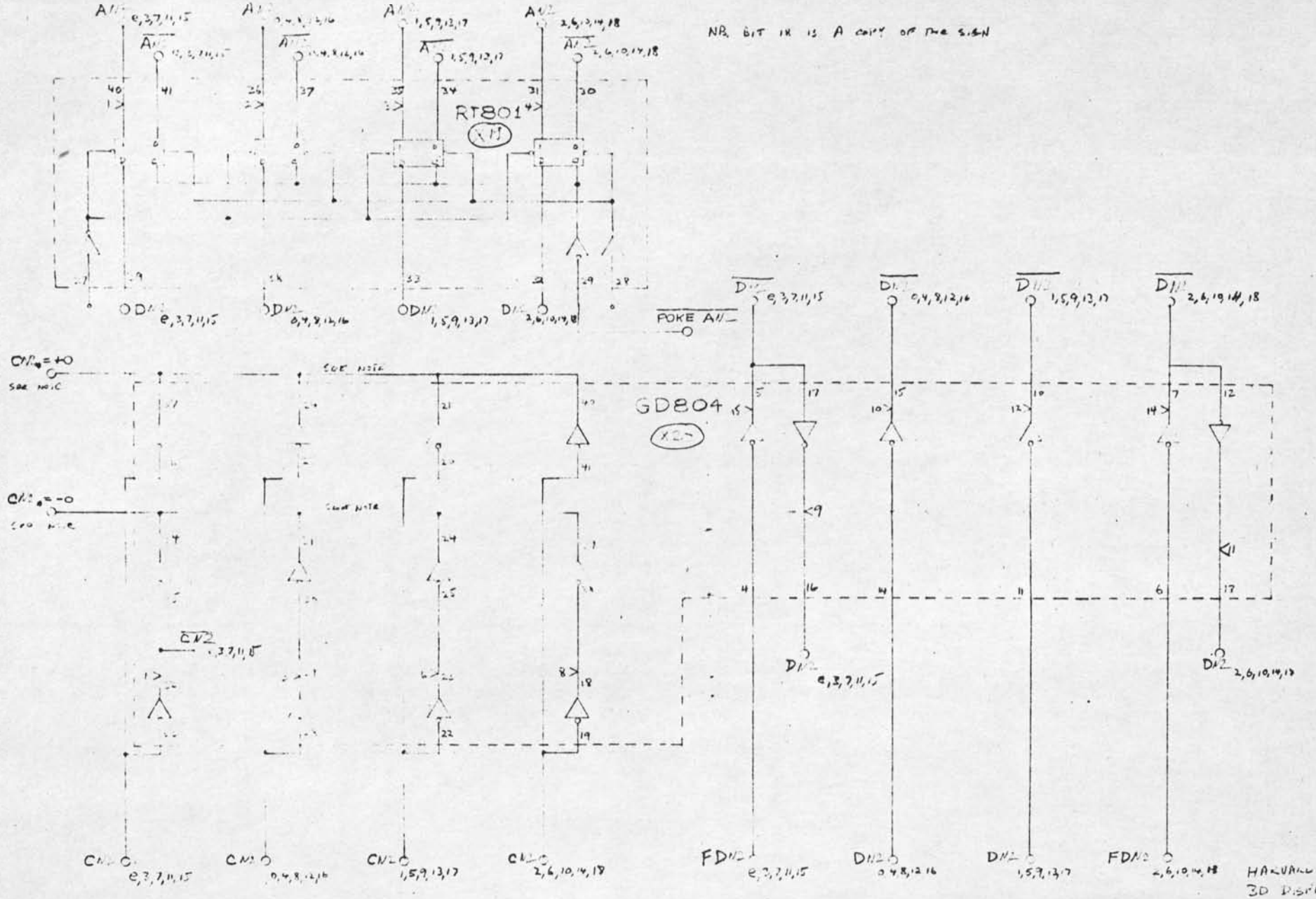


SHORT SYMBOL

NB. BECAUSE Δ FLIP FLOPS ARE REVERSED, WHAT GOES INTO THE SWITCH IS THE SAME AS WHAT COMES OUT OF Δ.

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER

ΔN2 REGISTER AND GATES

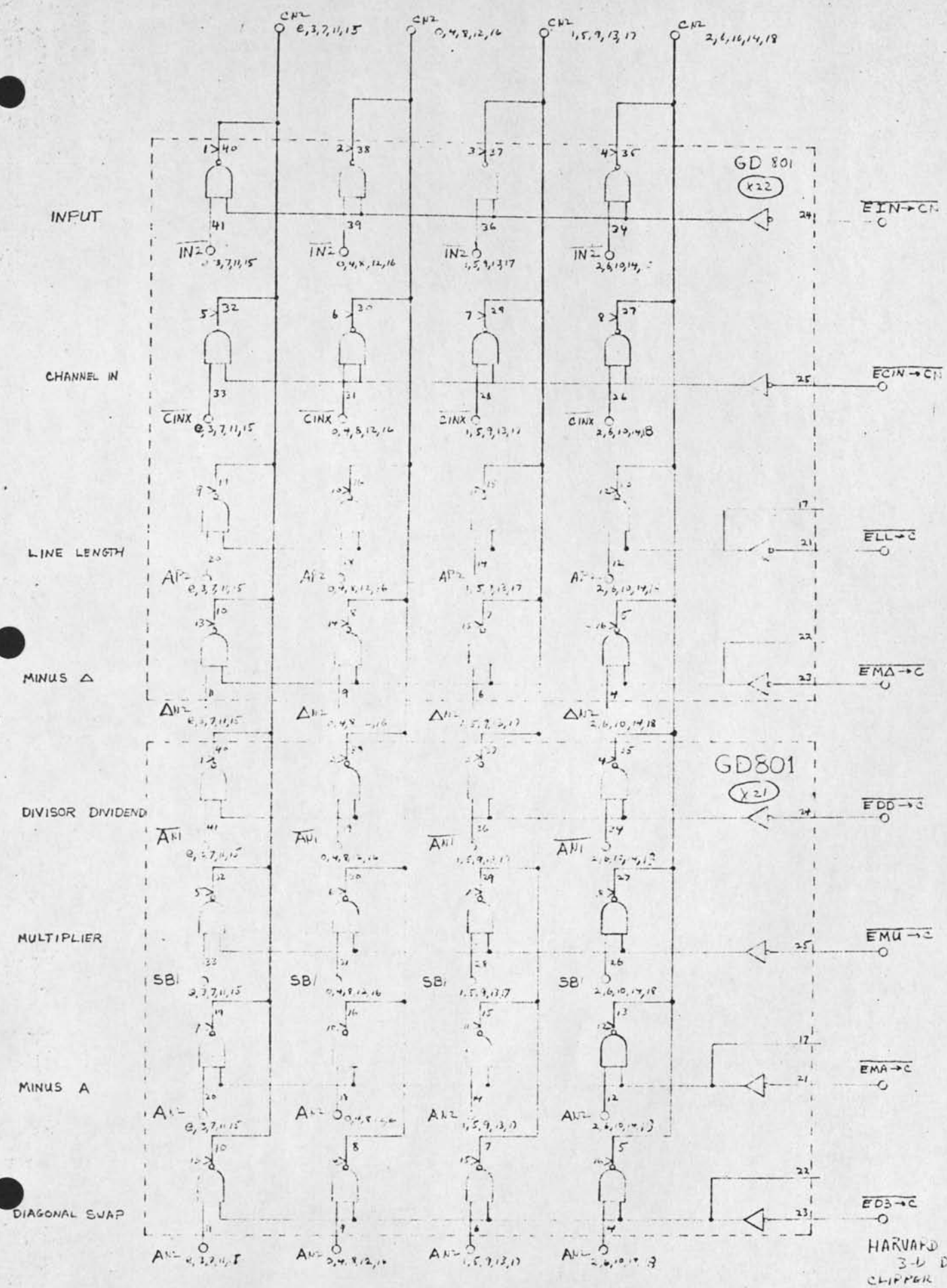


NB. $CN2_{*}=0$ ARE THREE LINES WITH SUBSCRIPTS e-5, 6-11, 12-17. E.G. $CN1_{e-5}=0$ THE FOUR WIRES ON THIS DIAGRAM MARKED 'SEE NOTE' ARE SCHEMATIC. ACTUALLY BITS 2-5 ARE WIRED TOGETHER AS ARE BITS 6-11, ETC. 12-17.

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER

A AND Δ REGISTERS
ZERO DETECTION NOTE

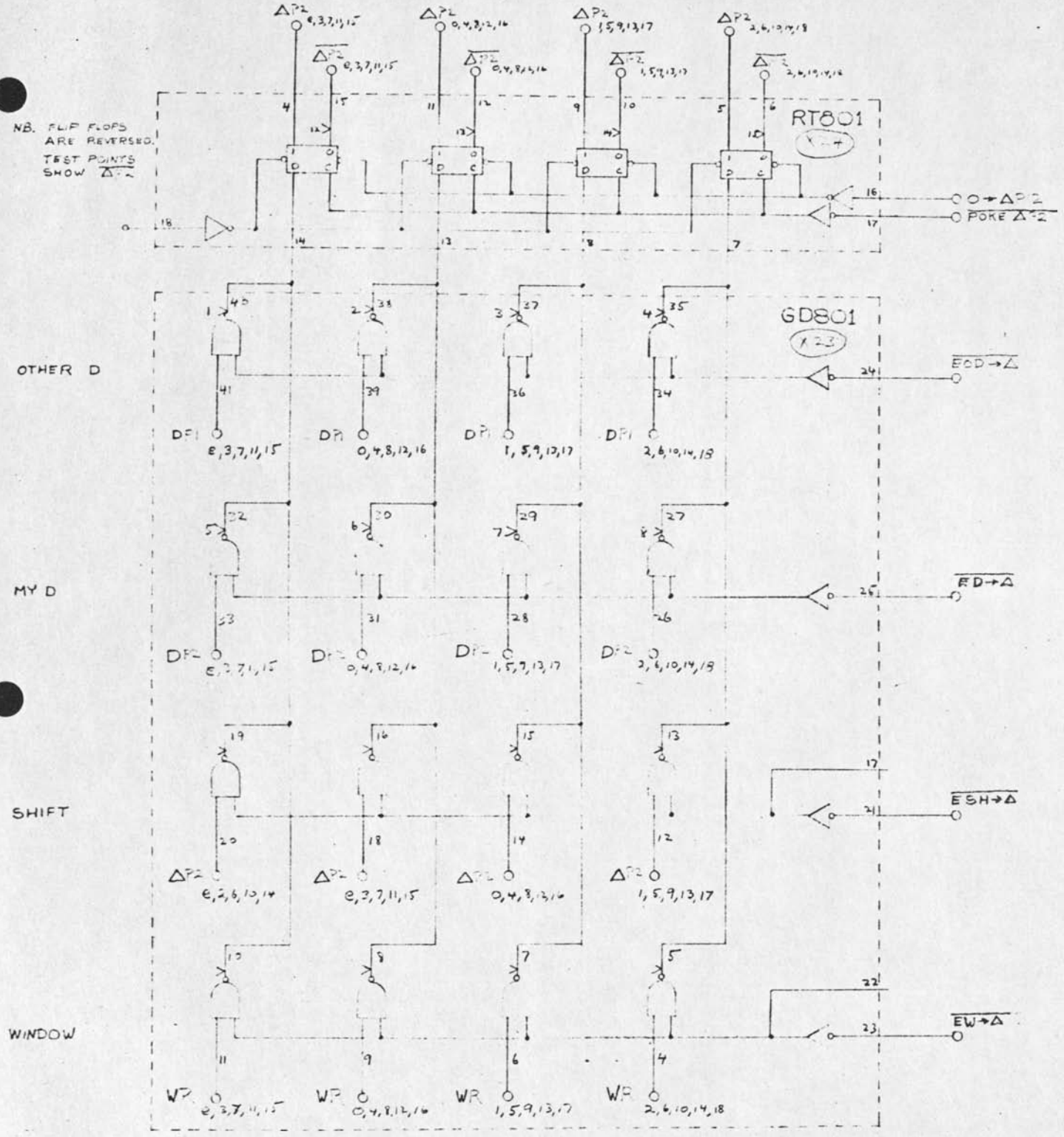
C8 C11 C14 C17

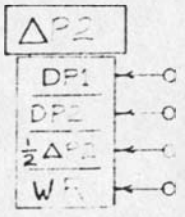


INPUTS ARE INVERTED, BUSS READS TRUE

REVISED 4NOV67

HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER
C-BUSS SWITCH
N2
25



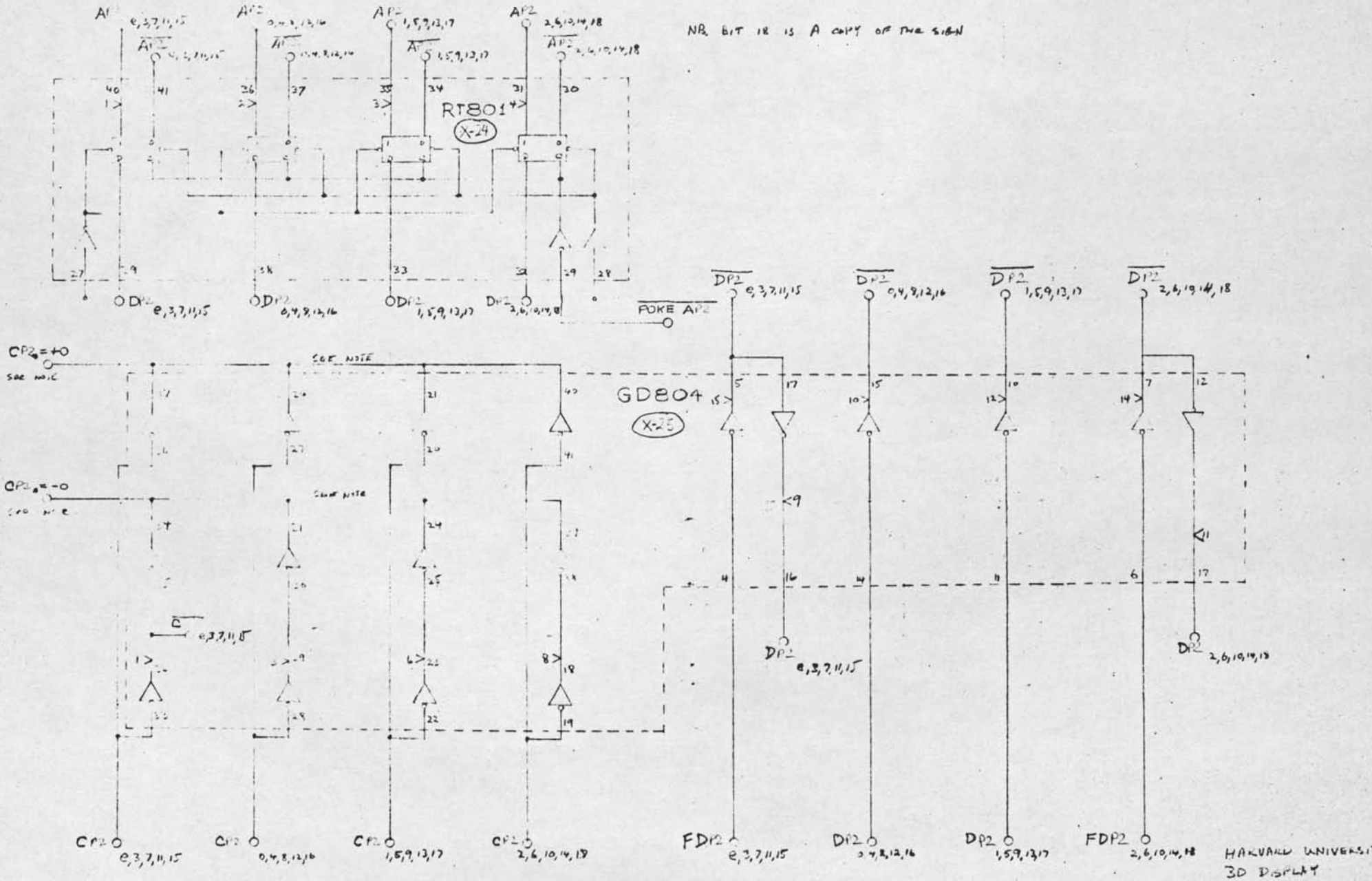


SHORT SYMBOL

NB. BECAUSE Δ FLIP FLOPS ARE REVERSED, WHAT GOES INTO THE SWITCH IS THE SAME AS WHAT COMES OUT OF Δ.

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER

ΔP2 REGISTER AND GATES

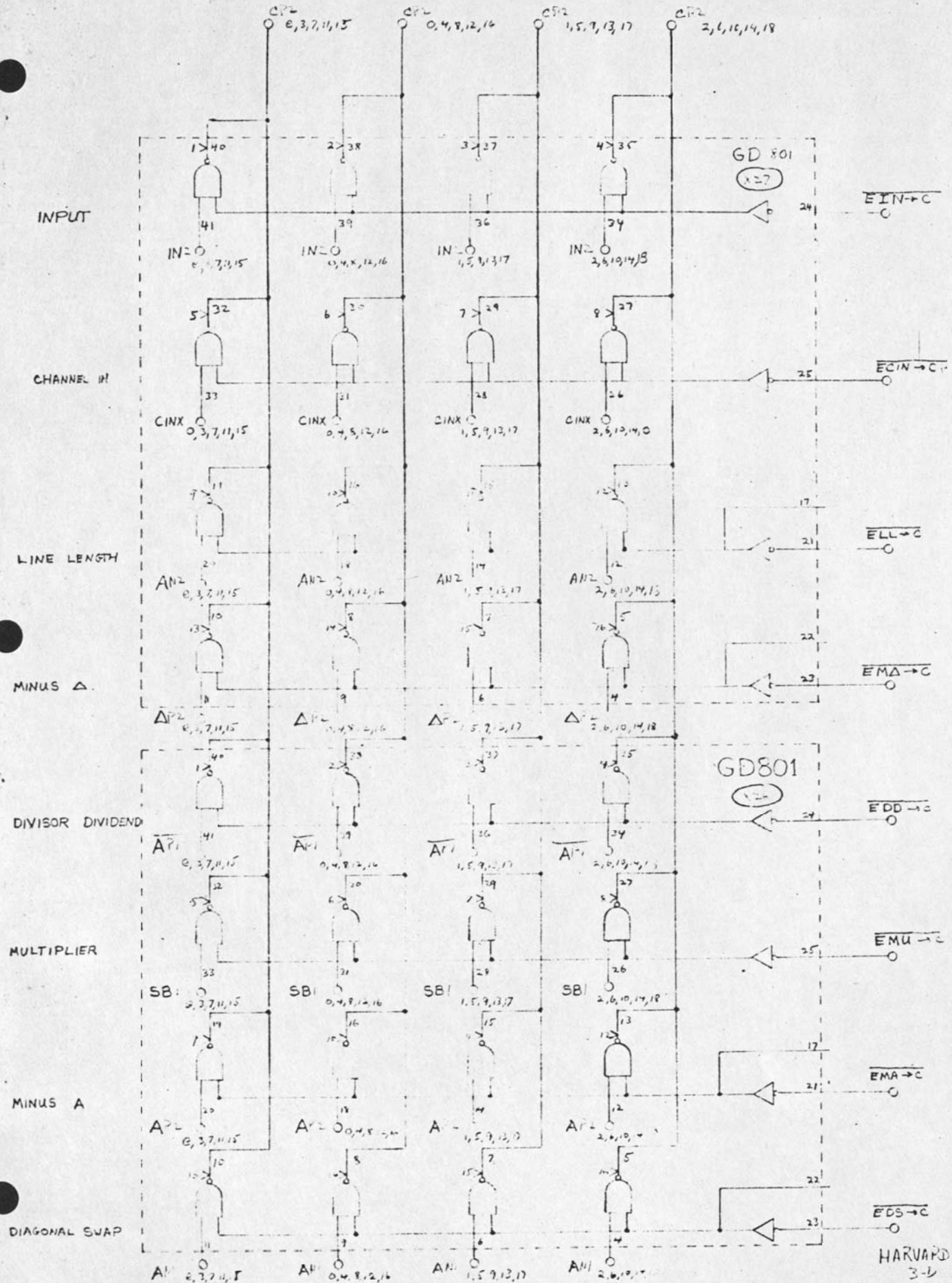


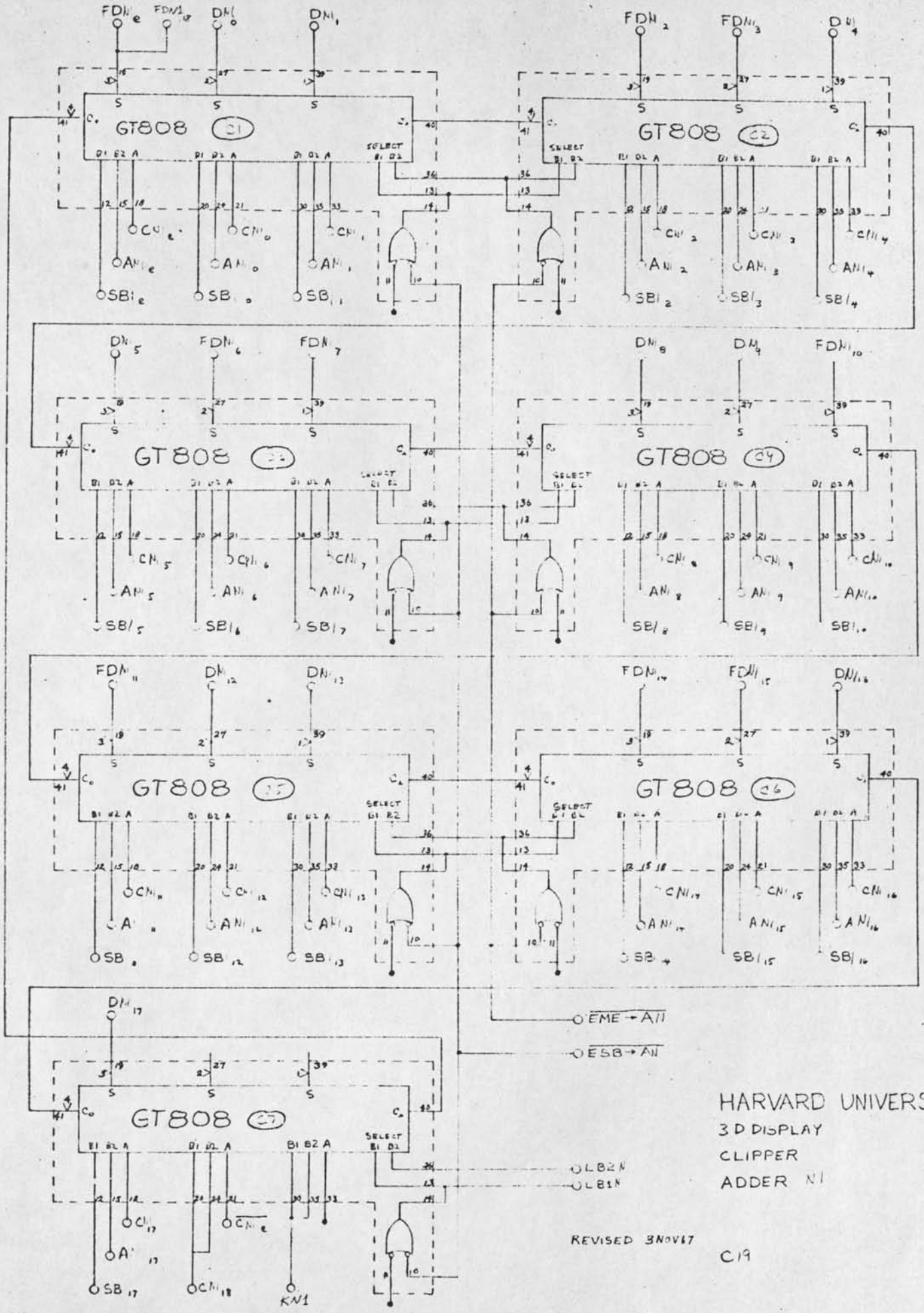
NB. C $_{*=0}$ ARE THREE LINES WITH SUBSCRIPTS e-5, 6-11, 12-17. E.G. $CN1_{e-5}=0$ THE FOUR WIRES ON THIS DRAWING MARKED "SEE NOTE" ARE SCHEMATIC. PHYSICALLY THEY ARE ONE WIRE THROUGH AS FOR BITS 6-11, AND 12-17.

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER

A AND Δ REGISTERS
ERROR DETECTION NOTE

C8 C11 C14 C17

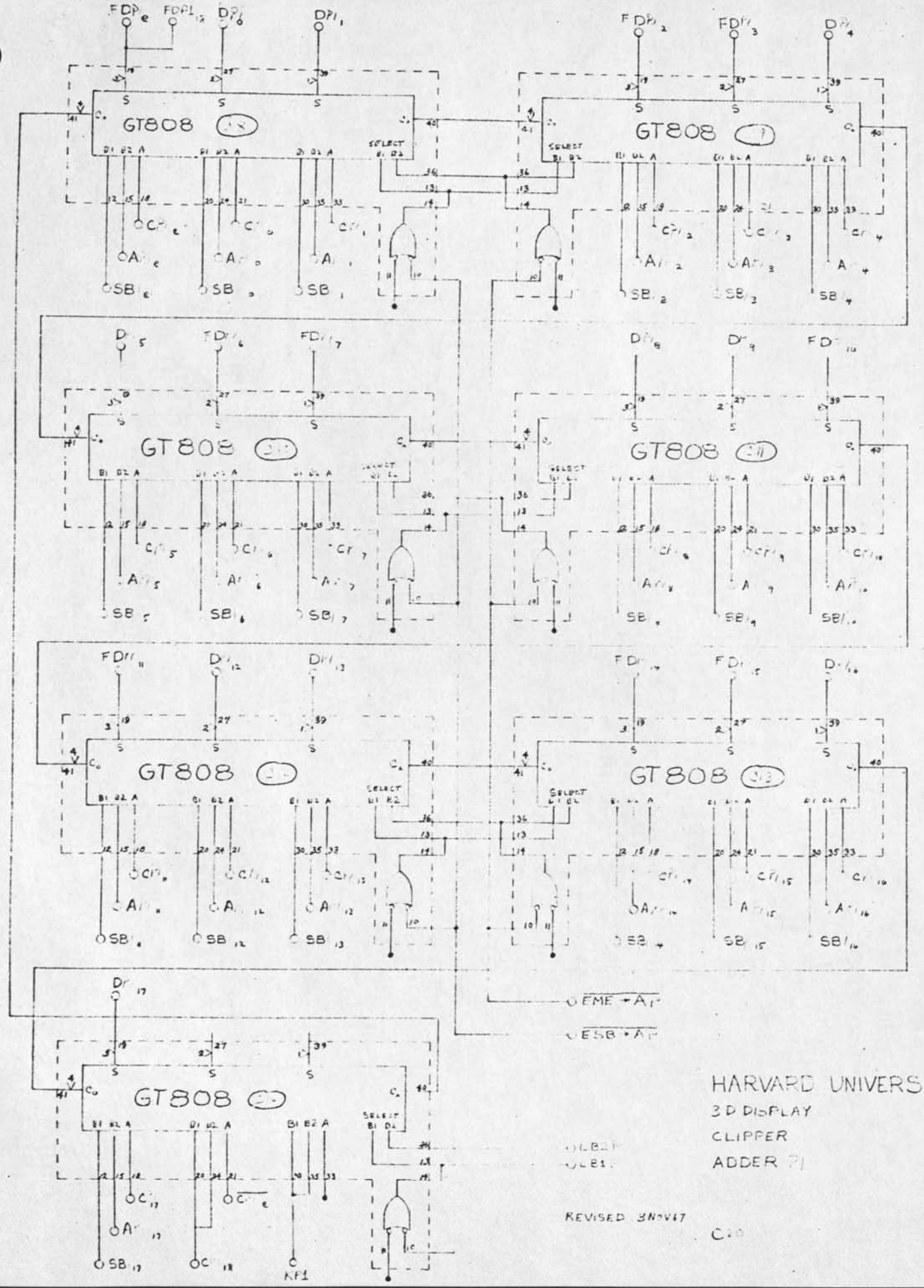

INPUT
CHANNEL IN
LINE LENGTH
MINUS Δ
DIVISOR DIVIDEND
MULTIPLIER
MINUS A
DIAGONAL SWAP
GD 801
GD801
EIN→C
ECIN→C
ELL→C
EMΔ→C
EDD→C
EMU→C
EMA→C
EDS→C
INPUTS ARE INVERTED, BUSS READS TRUE
REVISED 4NOV67
HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER
C-BUSS SWITCH
P2

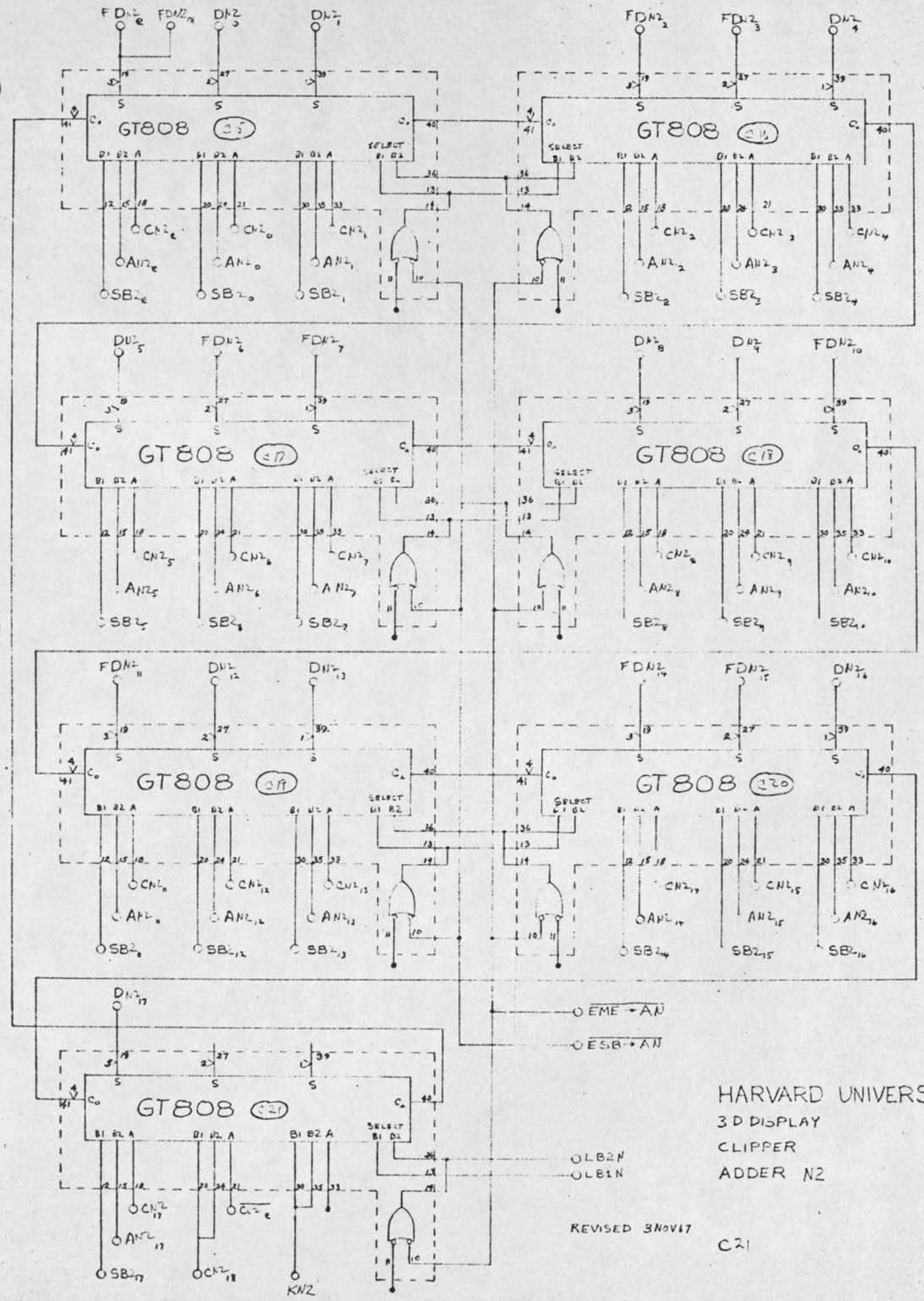


HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
ADDER N1

REVISED 3NOV87

C19

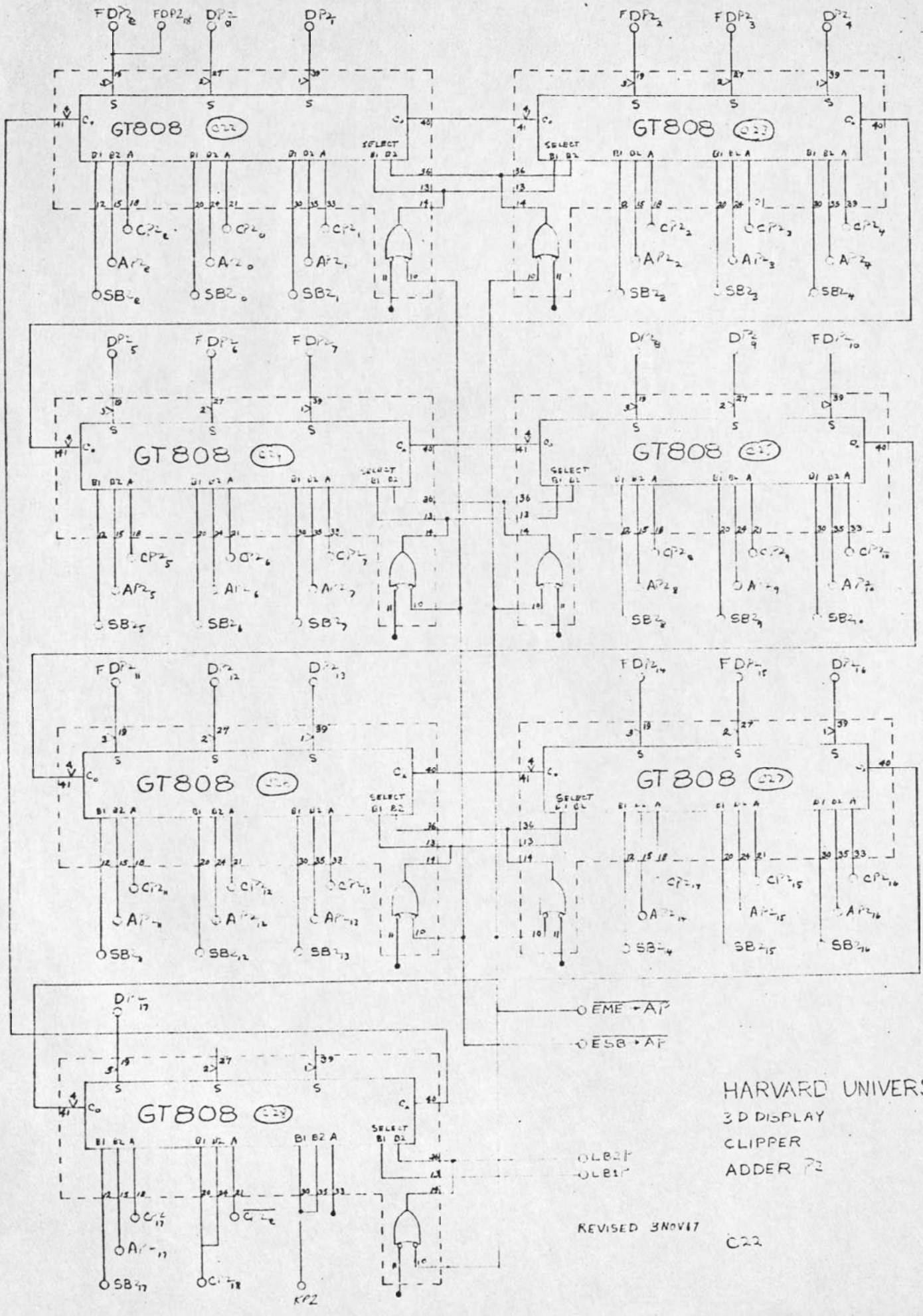

GT808
SELECT
S
SB
EME → AF
ESB → AF
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
ADDER P1
REVISED 3NOV67

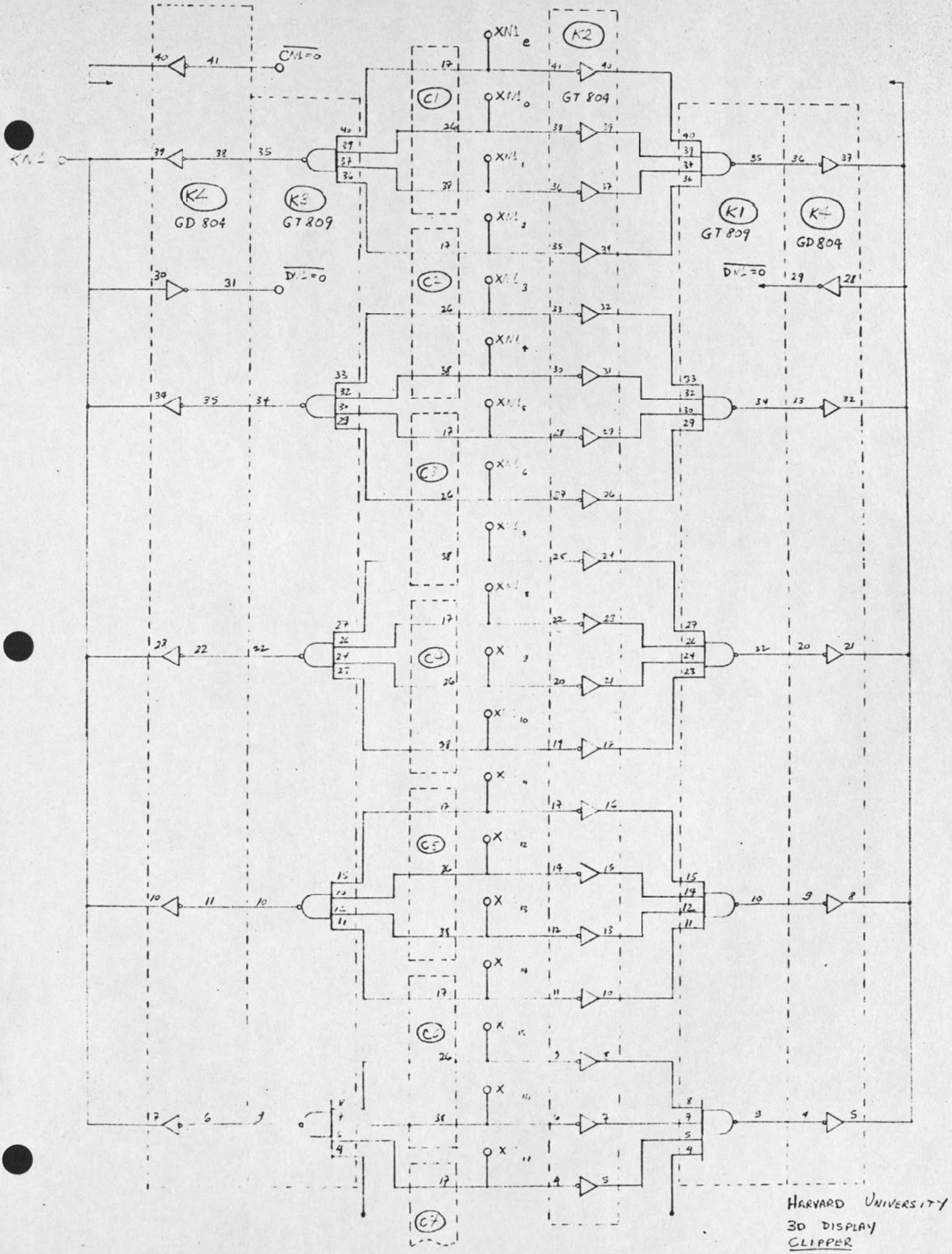


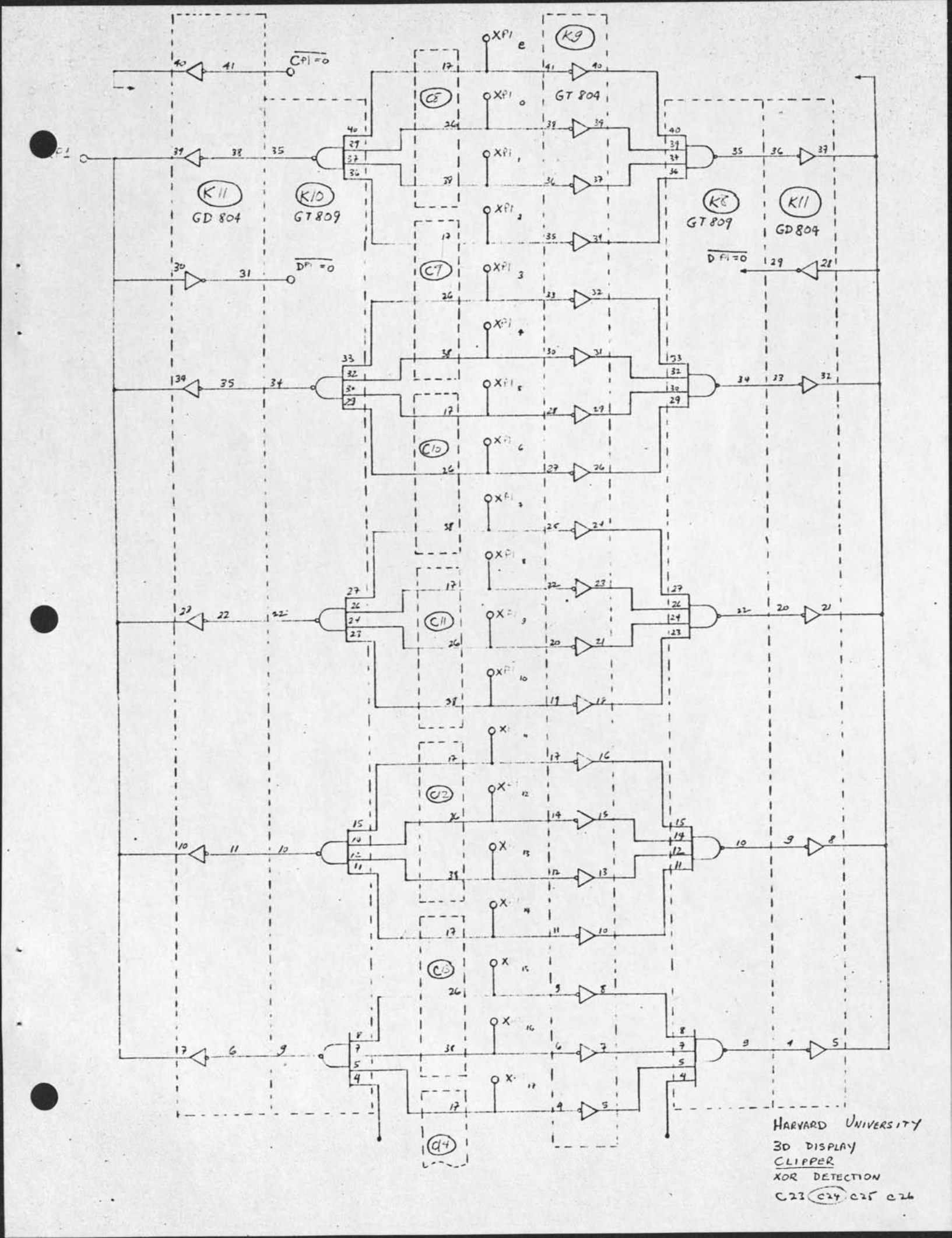
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
ADDER N2

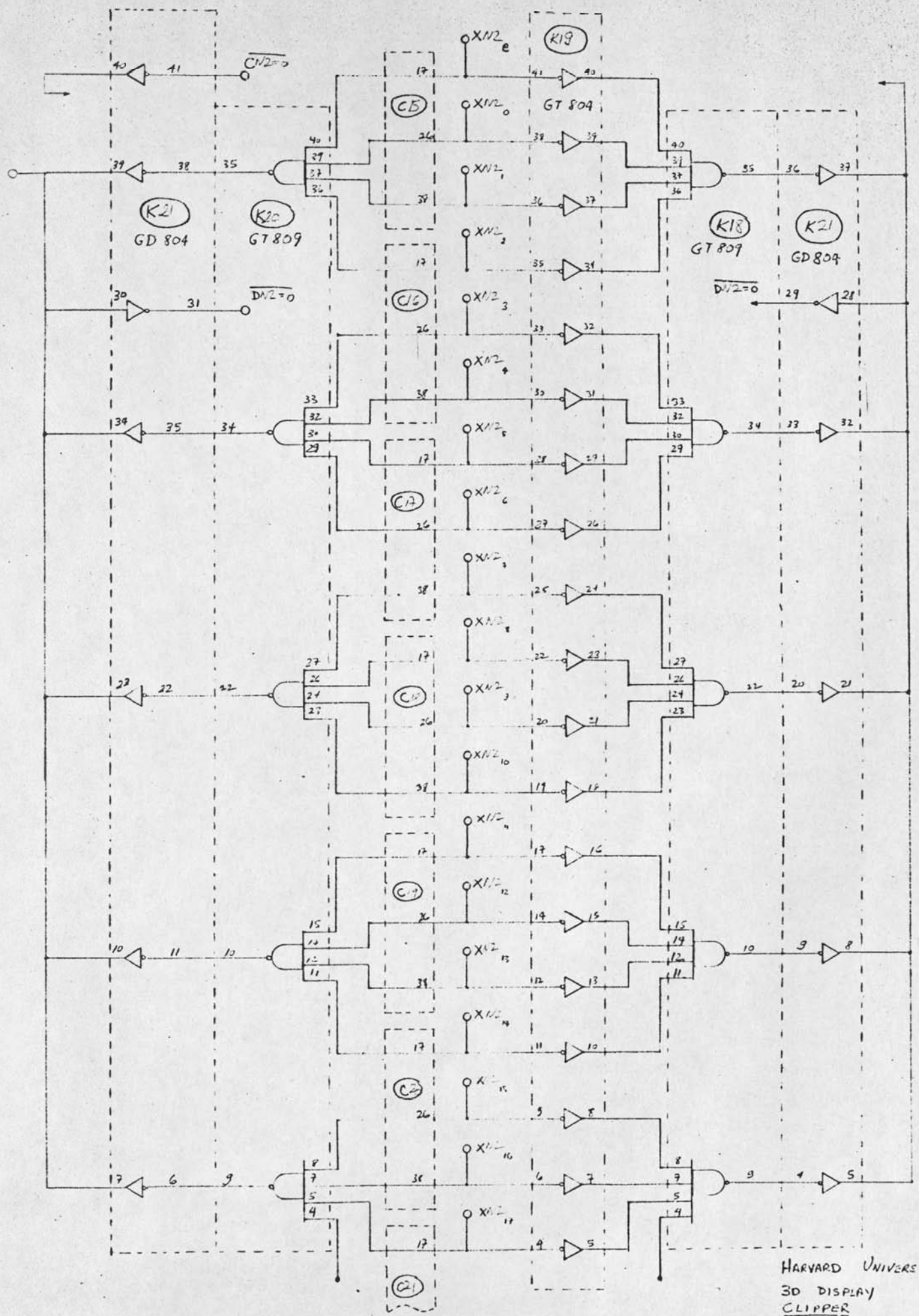
REVISED 3NOV17

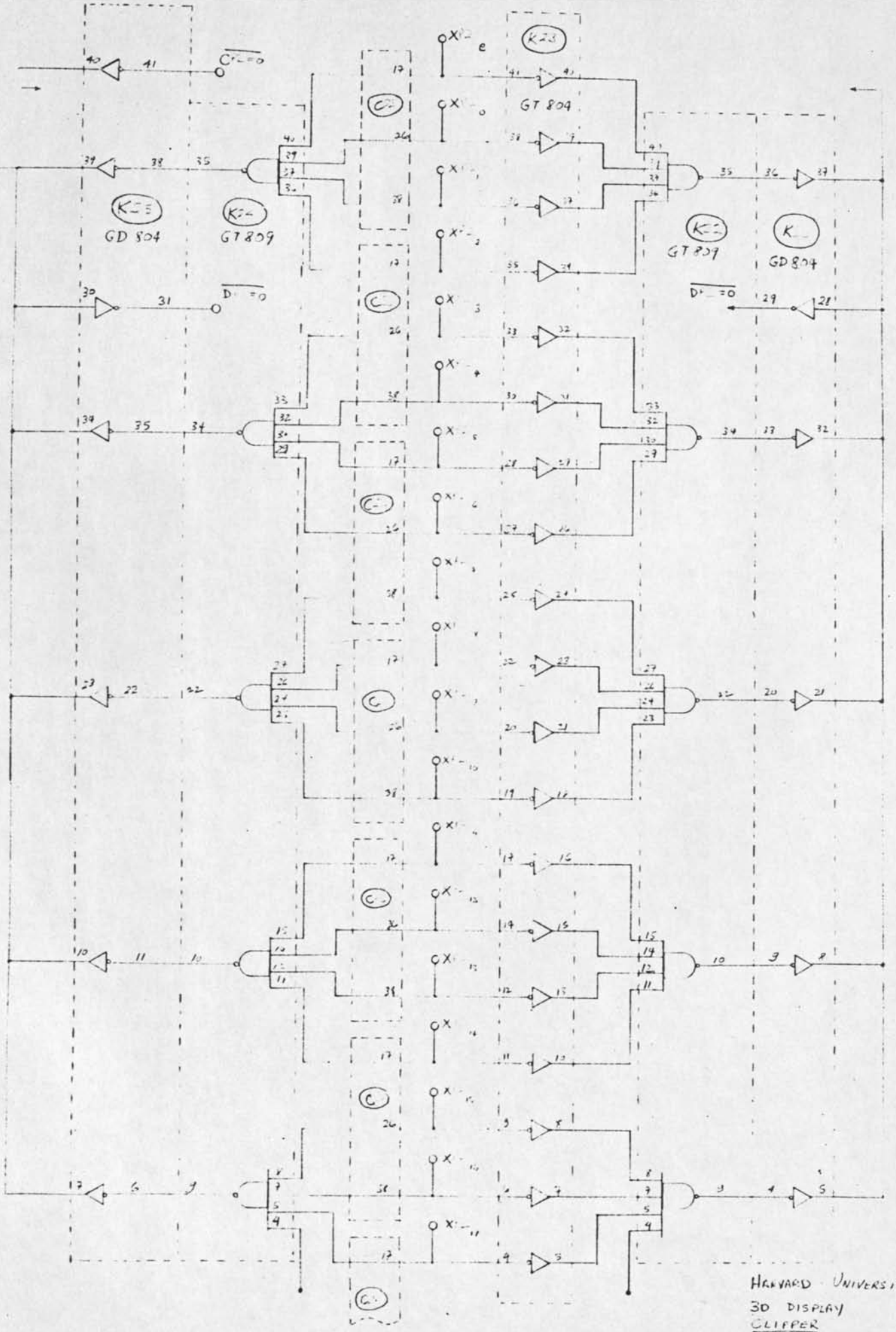
C21

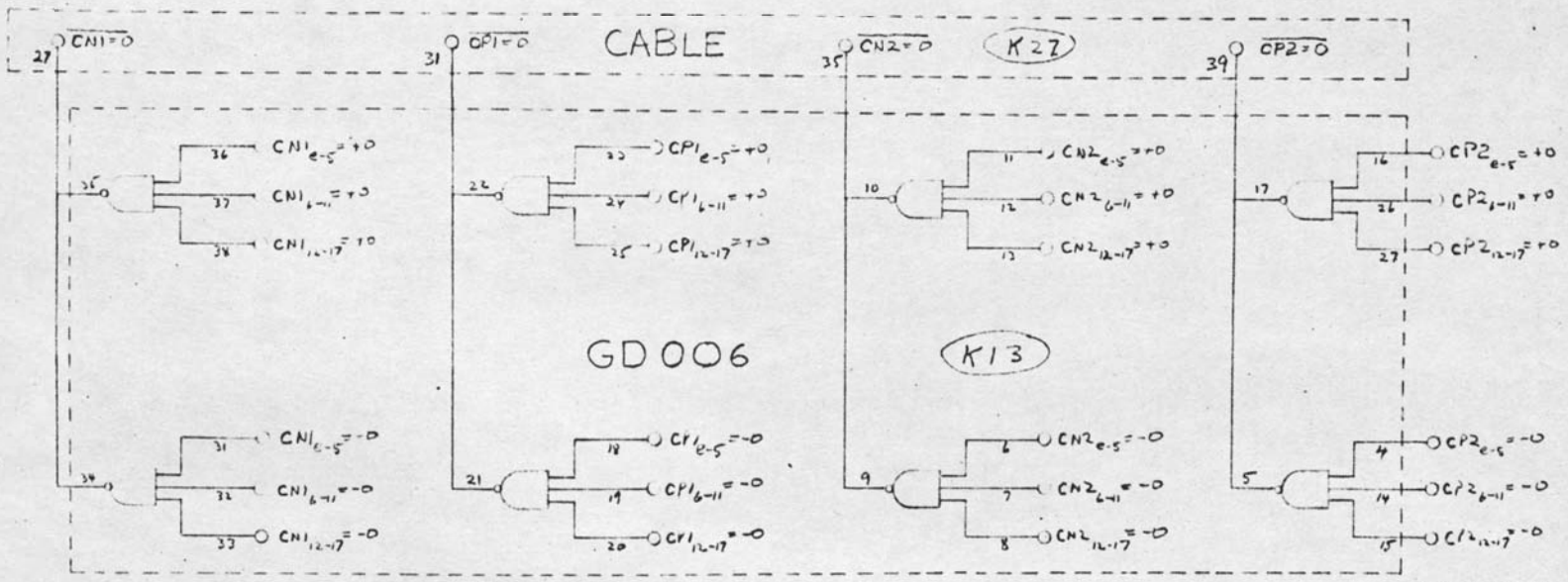

GT808
SELECT
S
$\overline{EME} \rightarrow AP$
$\overline{ESB} \rightarrow AP$
$LB2P$
$LB1P$
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
ADDER P2
REVISED 3NOV67
C22


K2
GT 804
K1
GT 809
K4
GD 804
K3
GT 809
K4
GD 804
C1
C2
C3
C4
C5
C6
C7
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
XOR DETECTION
C23 C24 C25 C26


K11
GD 804
K10
GT 809
K9
GT 804
K8
GT 809
K11
GD 804
C8
C7
C10
C11
C12
C13
C14
CP1 = 0
DP1 = 0
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
XOR DETECTION
C23 C24 C25 C26


K21
GD 804
K20
GT 809
K19
GT 804
K18
GT 809
K21
GD 804
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
XOR DETECTION
C23 C24 C25 C26


GD 804
GT 809
GT 809
GD 804
GT 804
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
XOR DETECTION
C23 C24 C25 C26



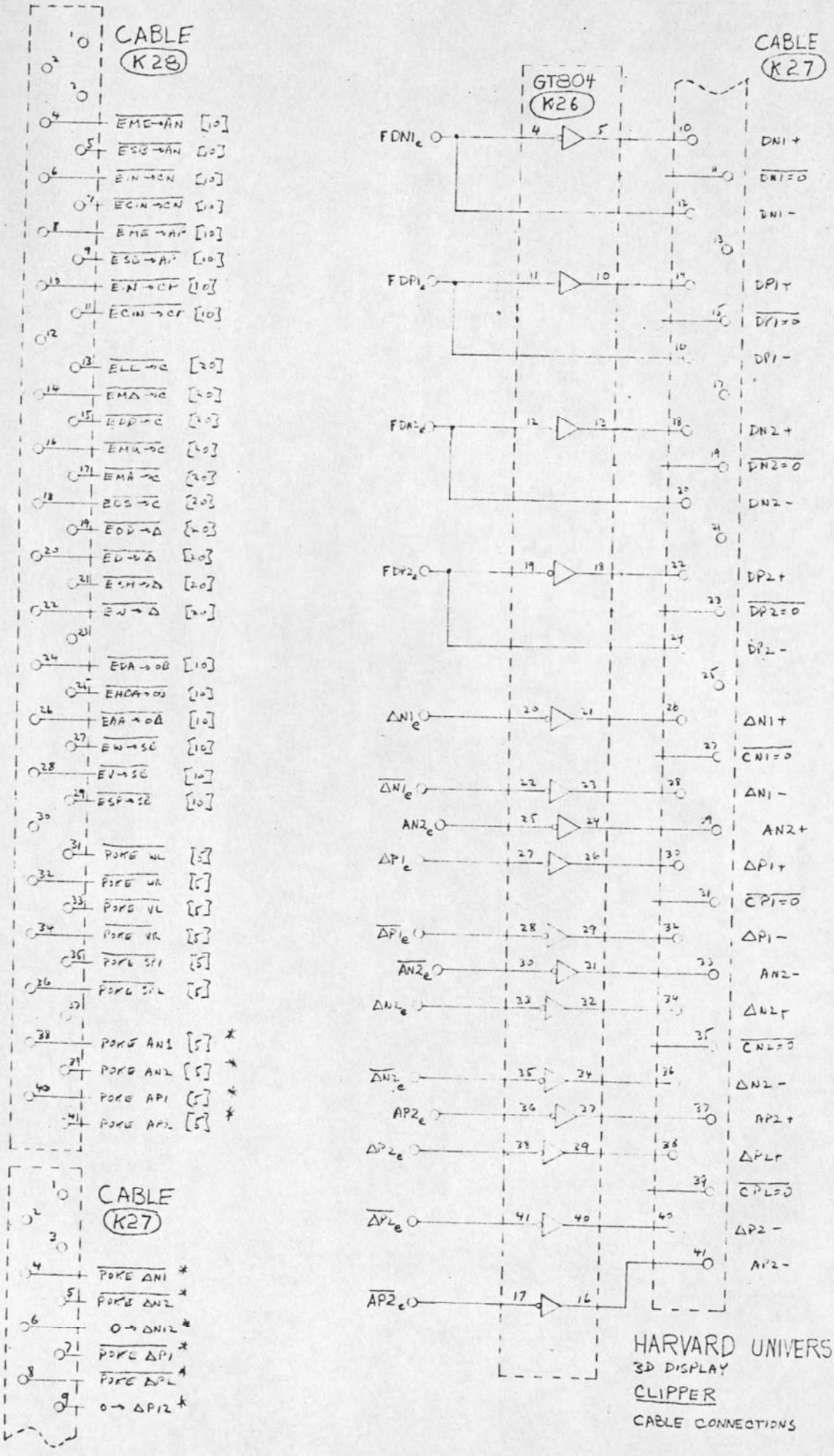
HARVARD UNIVERSITY

3D DISPLAY

CLIPPER

ZERO NETS

C27


CABLE
K28
CABLE
K27
GT804
K26
CABLE
K27
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER
CABLE CONNECTIONS
* THESE WIRES ARE NOT ACTUALLY CABLED 7/30/68
C28

APPENDIX 3

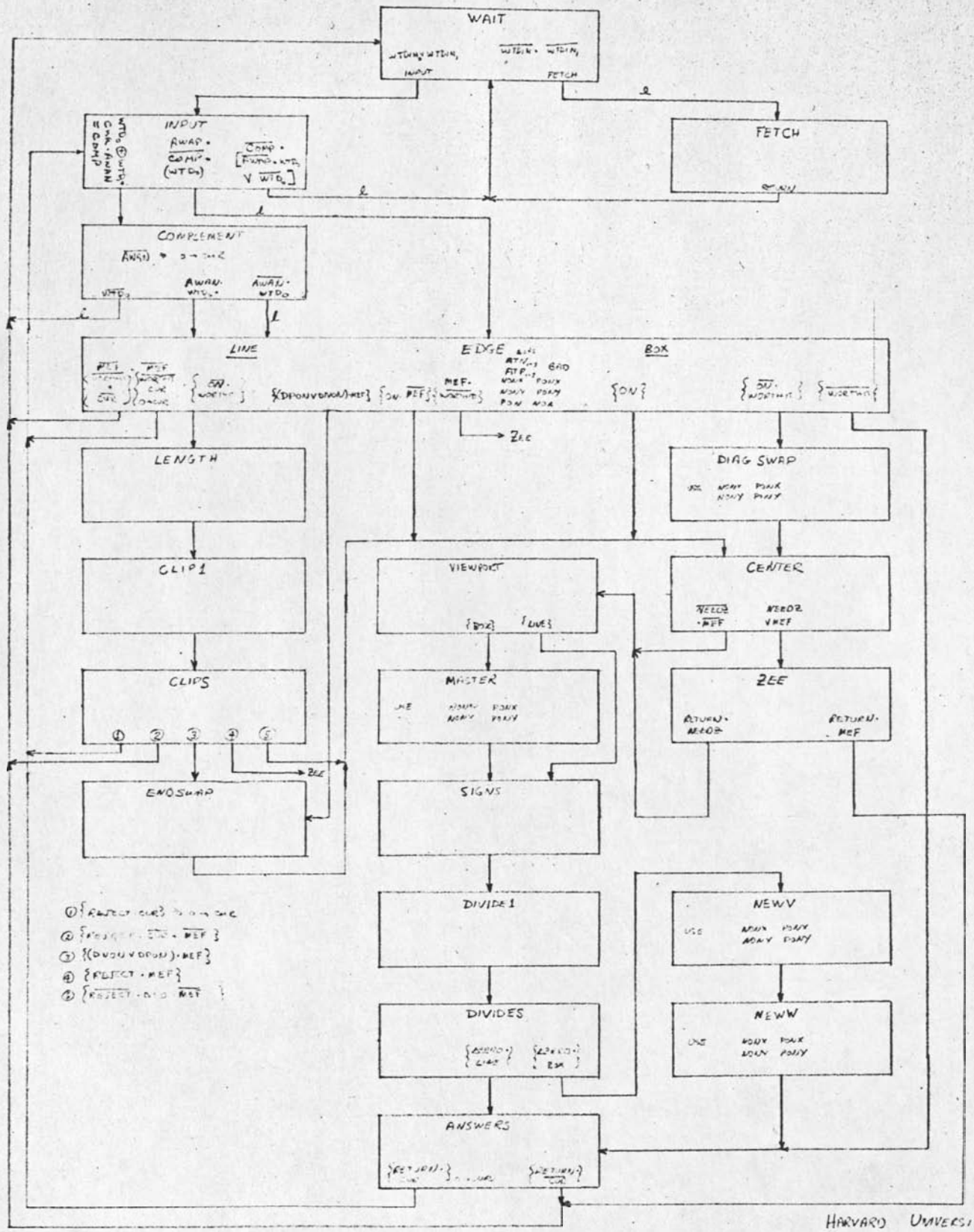
Prints for the Clipping Divider

Part 2 Clipper Control

Clipper Control

Print No.	Date	Title
CC1	7/30/68	flow diagram
CC2	7/30/68	mechanical layout
CC3	7/30/68	poke conditions
CC4	7/30/68	enable conditions
CC5	7/30/68	poke an
CC6	7/30/68	poke ap
CC7	7/30/68	accumulator poke logic
CC8	7/30/68	poke dn
CC9	7/30/68	poke dp
CC10	7/30/68	poke delta logic
CC11	7/30/68	zero-delta logic
CC12	7/30/68	storage register poke logic
CC13	7/30/68	gate enable logic
CC14	7/30/68	gate enable logic
CC15	7/30/68	gate enable logic
CC16	7/30/68	inequality nets
CC17	7/30/68	clip test nets
CC18	7/30/68	sign match logic
CC19	7/30/68	delta zero functions
CC20	7/30/68	combinational logic
CC21	7/30/68	combinational logic
CC22	7/30/68	combinational logic

CC23	7/30/68	combinational logic
CC24	7/30/68	combinational logic
CC25	7/30/68	combinational logic
CC26	7/30/68	state flip flops
CC27	7/30/68	state flip flops
CC28	7/30/68	state flip flops
CC29	7/30/68	state flip flops
CC30	7/30/68	state flip flops
CC31	7/30/68	angle detectors
CC32	7/30/68	angle adders
CC33	7/30/68	
CC34	7/30/68	directive input gate
CC35	7/30/68	storage register
CC36	7/30/68	storage register
CC37	7/30/68	state switches
CC38	7/30/68	extra tests
CC39	7/30/68	miscellaneous
CC40	7/30/68	strobe details
CC41	7/30/68	cables



HARVARD UNIVERSITY

3-D DISPLAY

CLIPPER CONTROL

FLOW DIAGRAM

SET OF 7/30/68

CC1

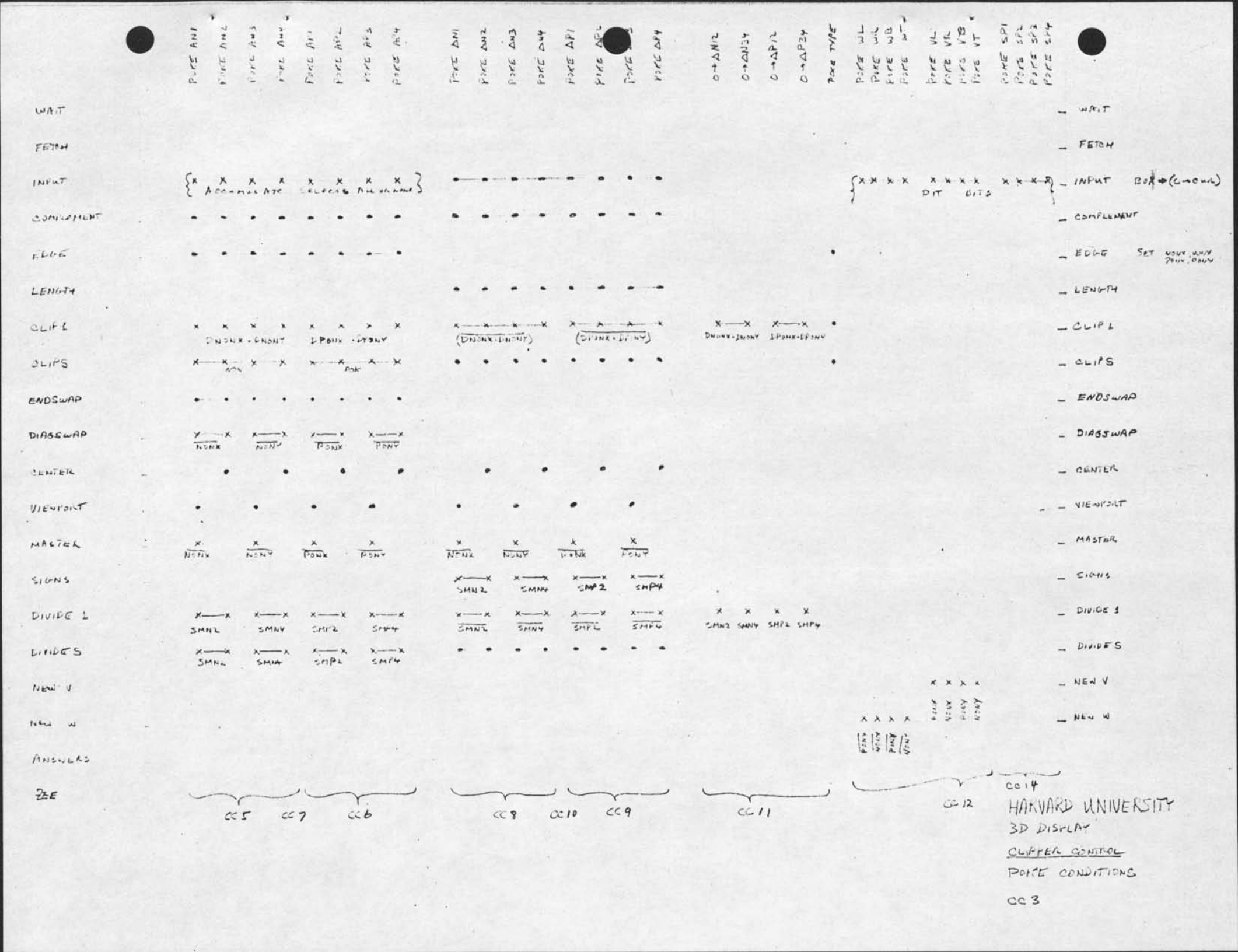
Slot	Q	Q cc	R	R cc	S	S cc
1	GD 801	CC 34	RT 801	CC 35	RT 801	CC 33
2	GD 801	CC 34	RT 801	CC 35,34	GT 808	CC 32
3	GD 801	CC 34	GT 813	CC 20,21,22,24	GT 808	CC 32
4			GT 809	CC 20,21,24,25	GT 808	CC 32
5			GT 810	CC 20,22,23,24	GT 802	CC 31,32
6			GD 804	CC 21,22,23	GT 813	CC 31
7			GD 813	CC 22,23,25	GT 809	CC 31
8			GT 809	CC 22,24	GT 809	CC 31
9			GT 810	CC 22,25	GT 801	CC 31
10			GT 802	CC 20,24	GT 809	CC 31
11			GD 802	CC 24,25	GT 804	CC 17,18,38
12			GD 809	CC 25	GT 813	CC 18
13			GD 802	CC 21,23	GD 810	CC 17,18,19
14			FT 801	CC 26	GD 810	CC 17,38
15			FT 801	CC 27	GD 813	CC 17
16			GT 804	CC 20,21,22,24,25	GD 813	CC 17
17			FT 801	CC 28	GD 809	CC 17
18			FT 801	CC 29	GD 804	CC 19,38
19	GT 809		FT 801	CC 30	GT 810	CC 16
20	GD 804		GT 802	CC 13,15	GT 810	CC 16
21	GT 810	CC 12	GT 810	CC 13,14,15		
22	GT 810	CC 12	GT 813	CC 14,15		
23	GT 809	CC 6	GT 809	CC 13,14	GT 802	CC 10
24	GT 809	CC 5	GT 805	CC 26-30	GD 810	CC 8,5,11
25	GT 813	CC 6	GT 805	CC 14,15	GT 813	CC 10,11
26	GT 813	CC 5,7	GT 805	CC 13,14,16	GD 813	CC 8,9
27	GT 810	CC 5,6,7			GD 813	CC 8,9
28	GT 802	CC 7			GD 813	CC 8,9

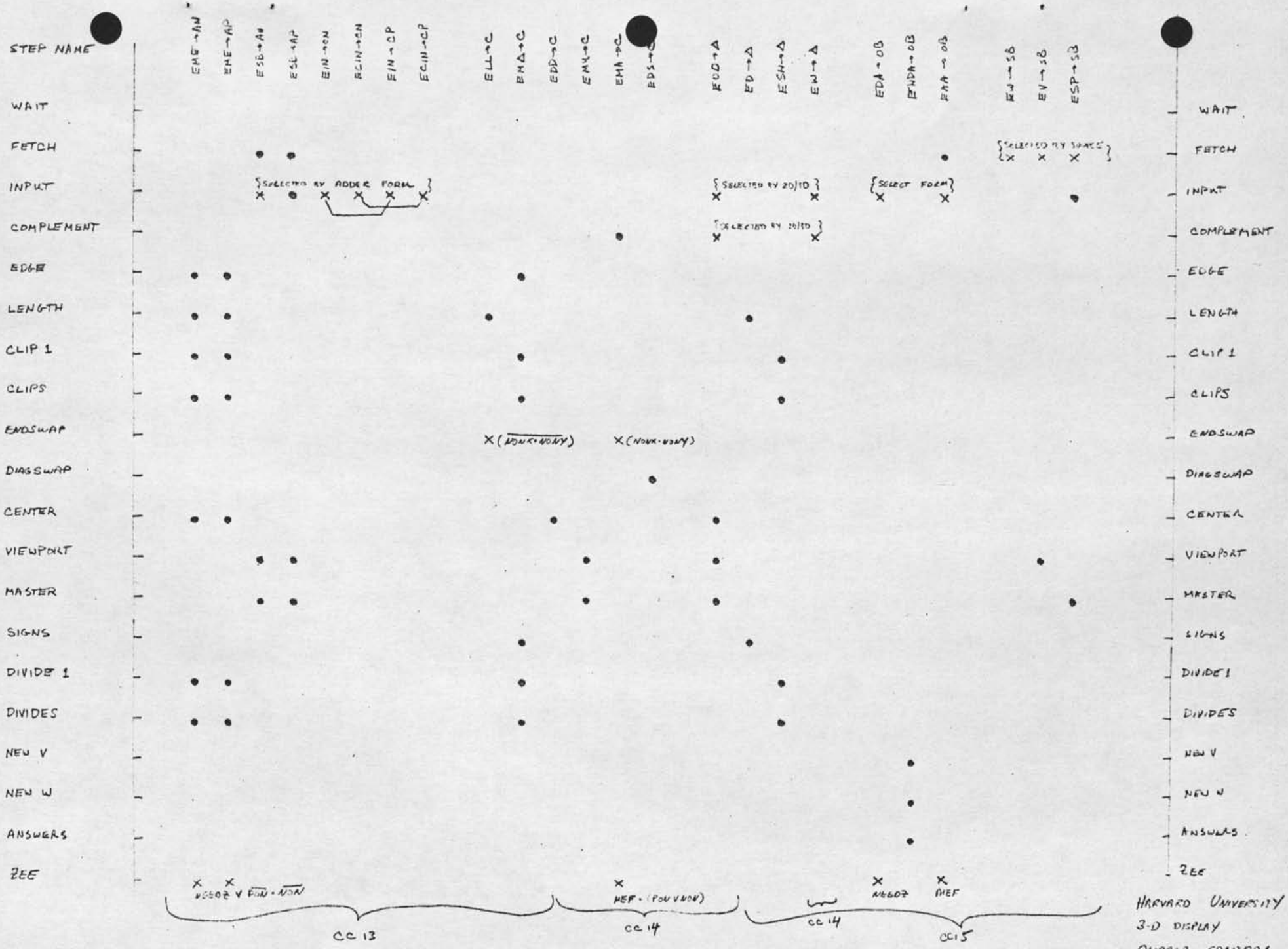
Row Q: POKE ACCUMULATOR (slots 19–28)

Row R: DIRECTIVE (slots 1–2); COMBINATIONAL LOGIC (slots 3–13); STATE FLIP FLOPS (slots 14–19); ENABLE LOGIC (slots 20–26); OUTPUTS (slots 27–28)

Row S: ANGLE (slots 1–10); TESTS (slots 11–20); INPUTS (slots 21–22); POKE DELTA (slots 23–28)

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
MECHANICAL LAYOUT
CC 2

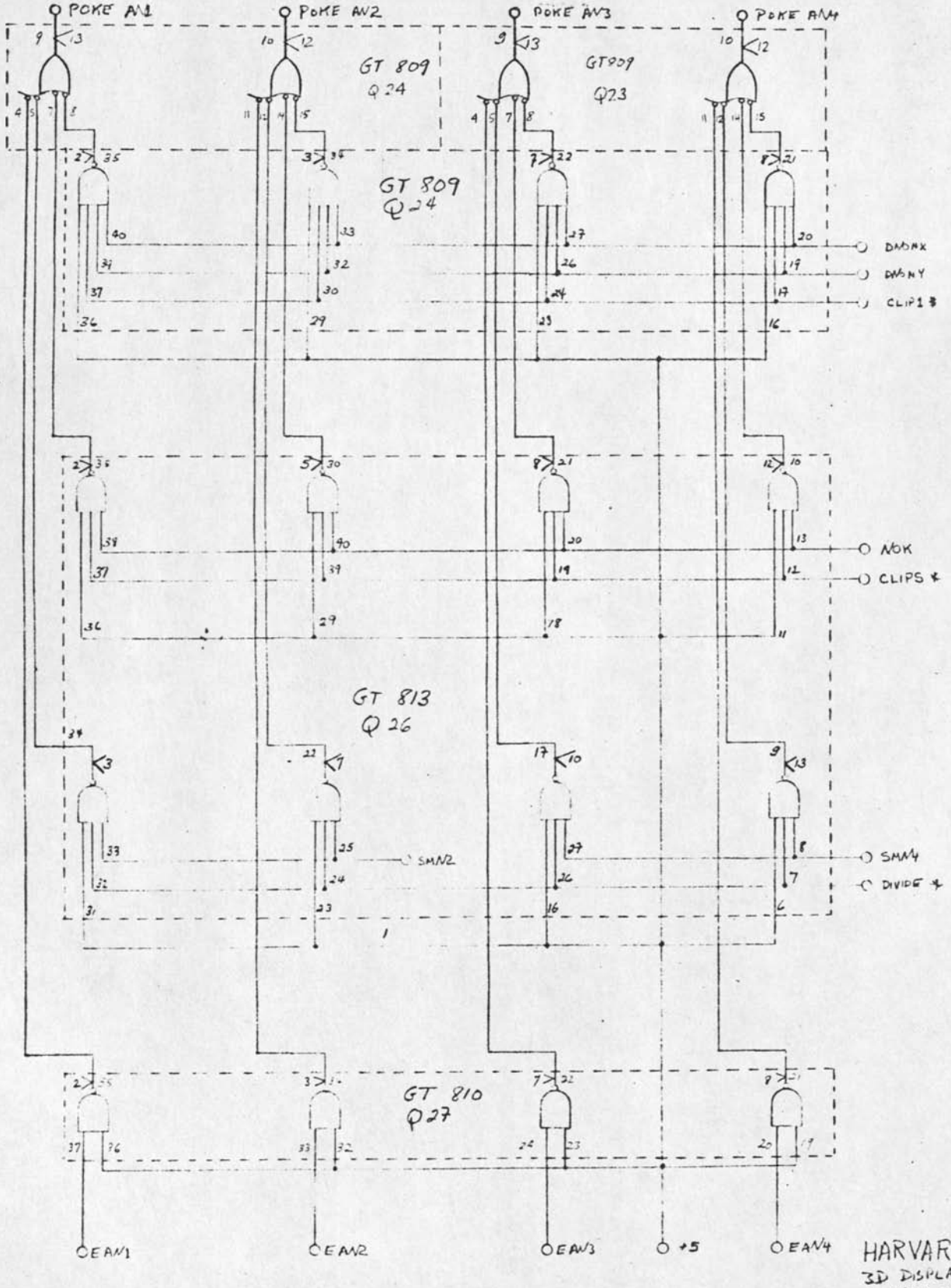

WAIT
FETCH
INPUT
COMPLEMENT
EDGE
LENGTH
CLIP L
CLIP S
ENDSWAP
DIAGSWAP
CENTER
VIEWPOINT
MASTER
SIGNS
DIVIDE 1
DIVIDE S
NEW V
NEW W
ANSWERS
ZEE
CC5
CC7
CC6
CC8
CC10
CC9
CC11
CC12
CC14
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
POKE CONDITIONS
CC3



HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER CONTROL

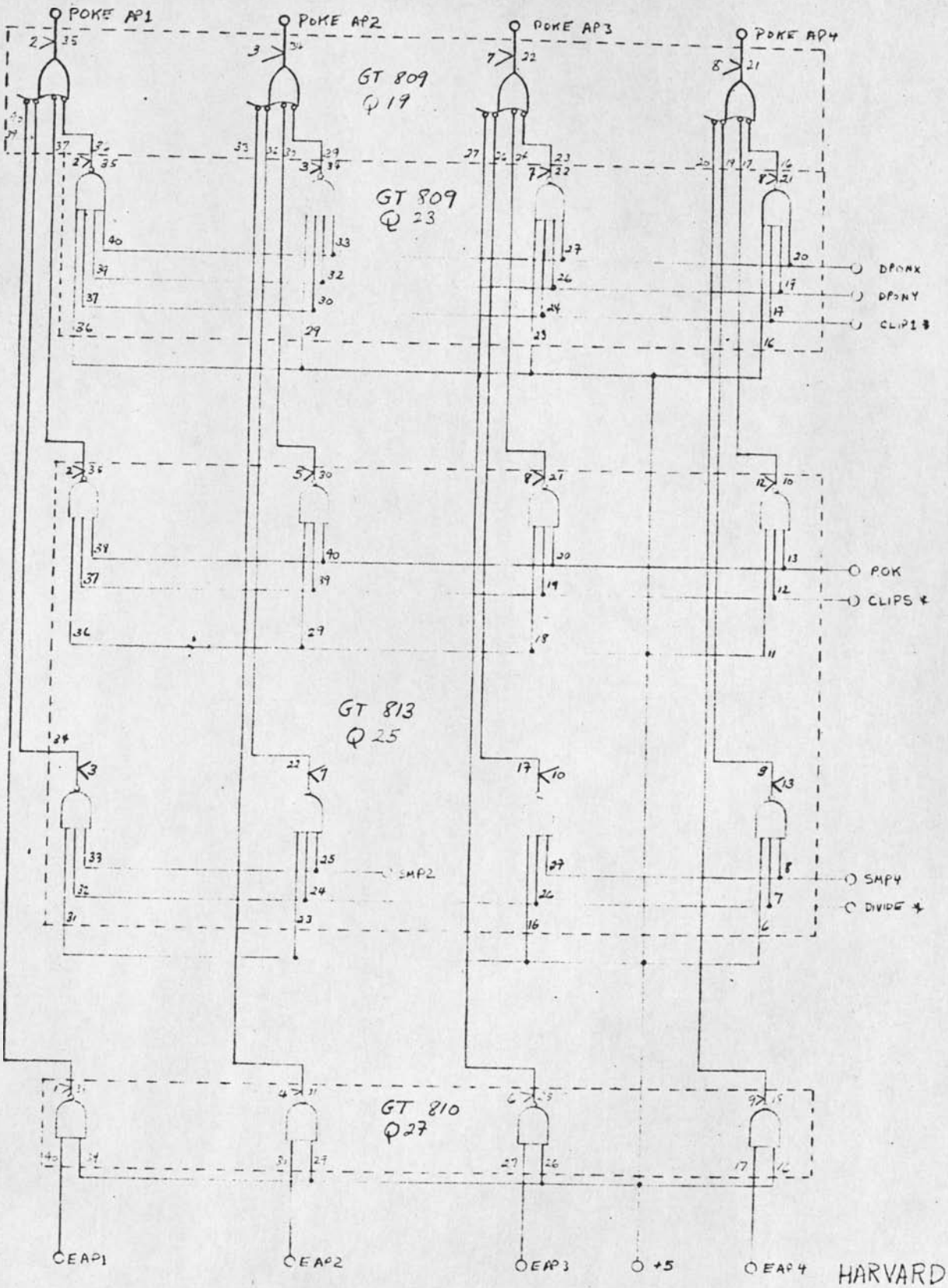
ENABLE CONDITIONS

CC A



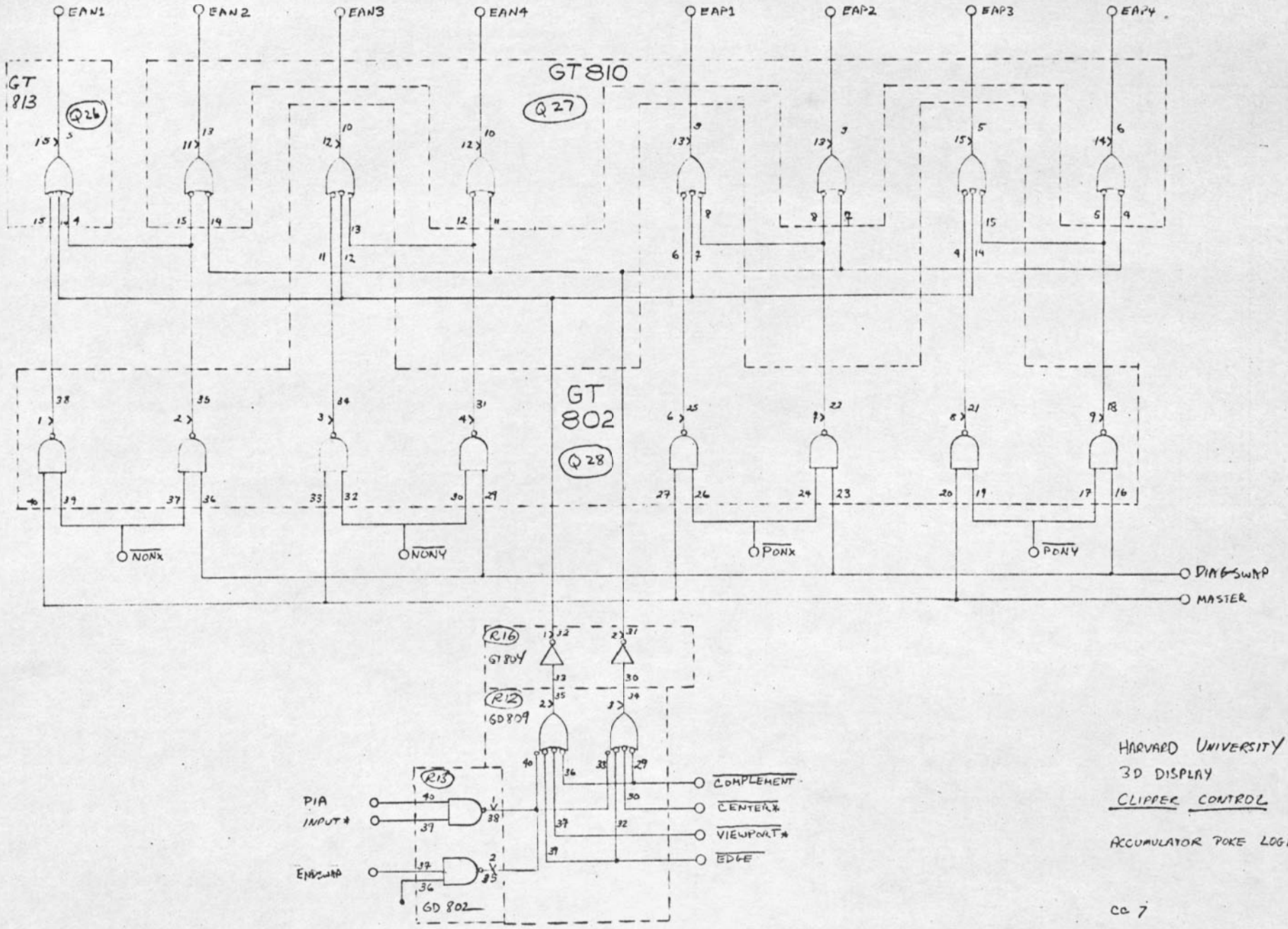
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
POKE AV

cc 5



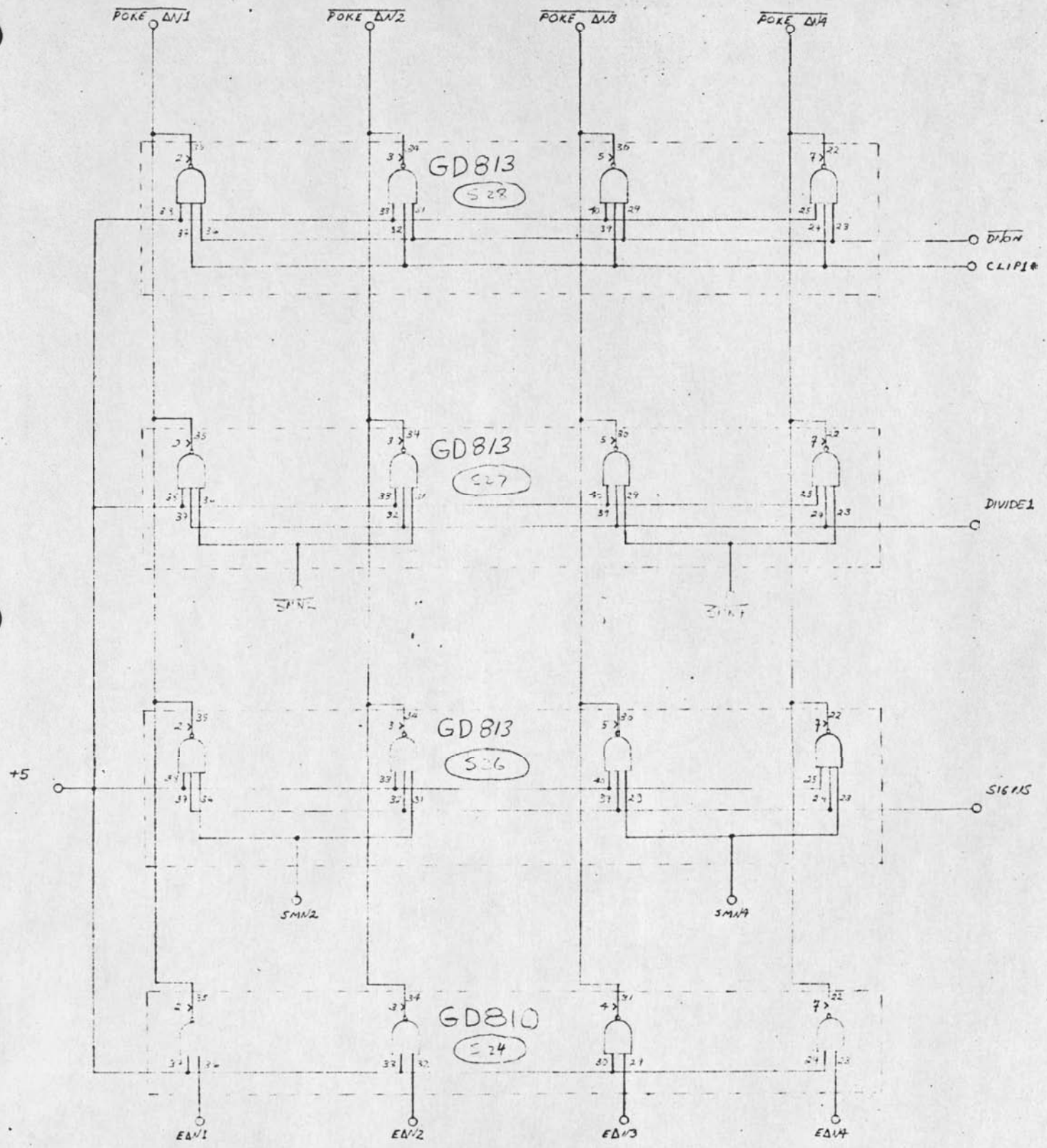
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
POKE AP

cc

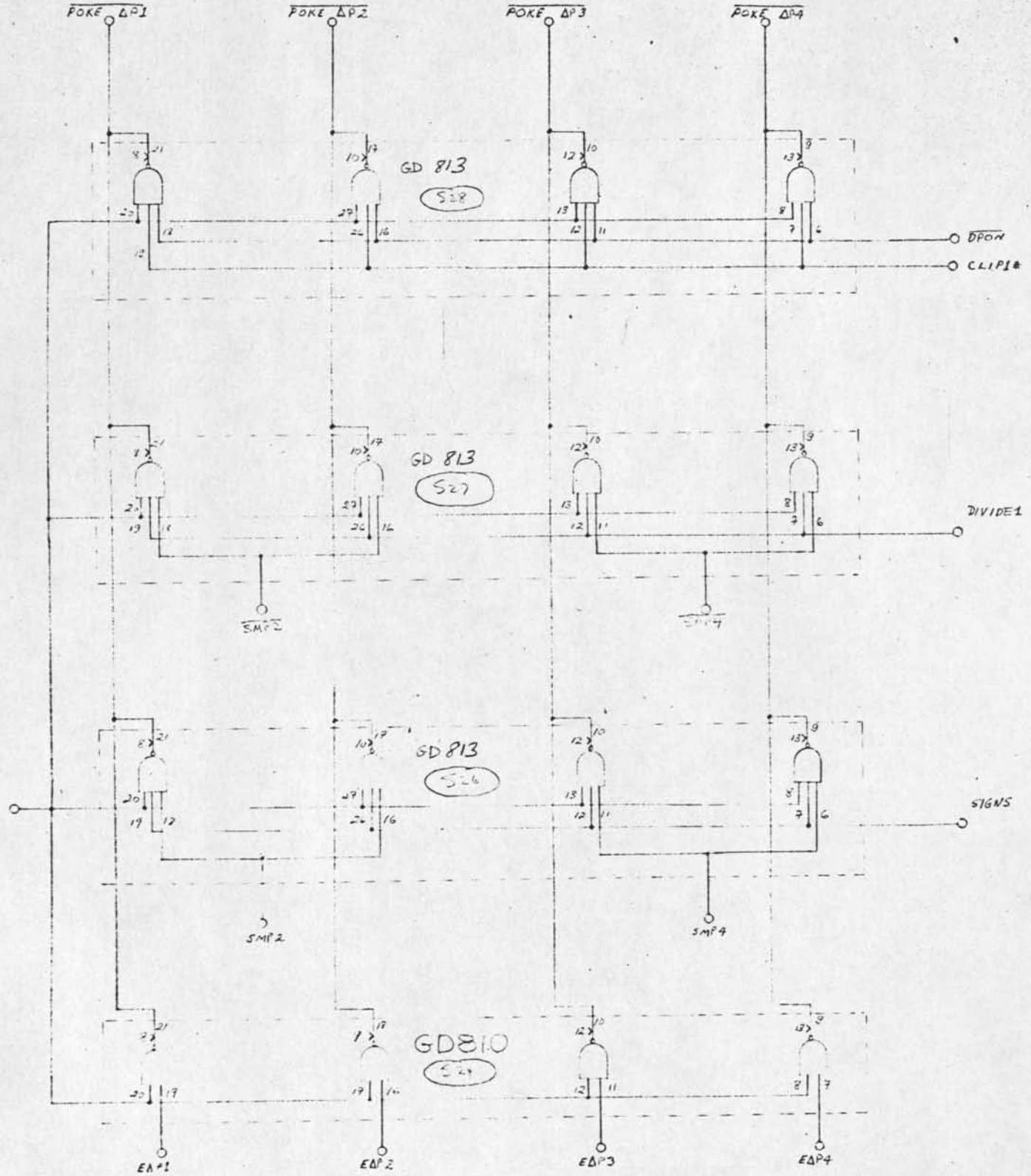


HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL

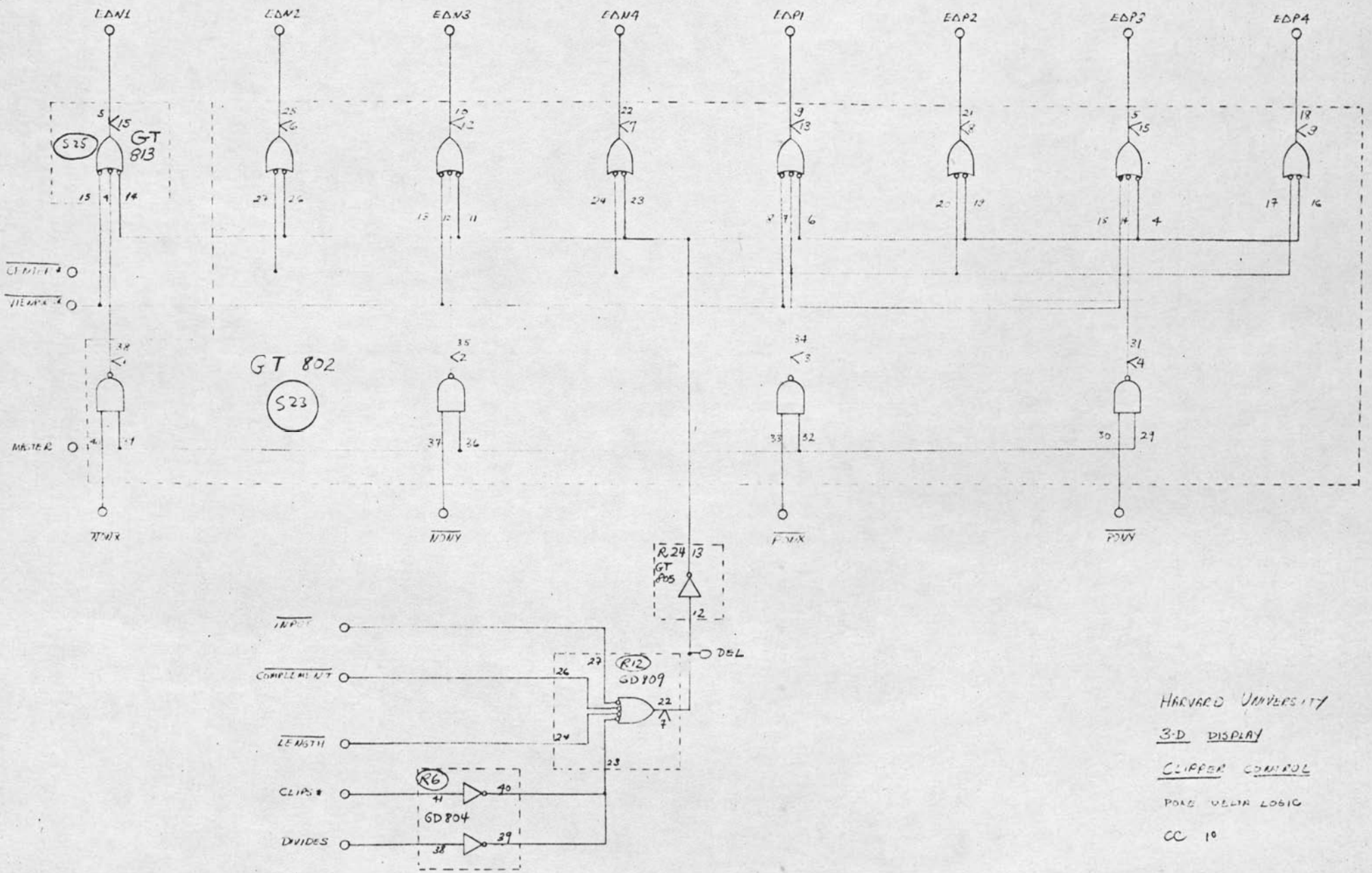
ACCUMULATOR POKE LOGIC



HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
POKE ΔN
CC 8



HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
POKE ΔP
CC 9



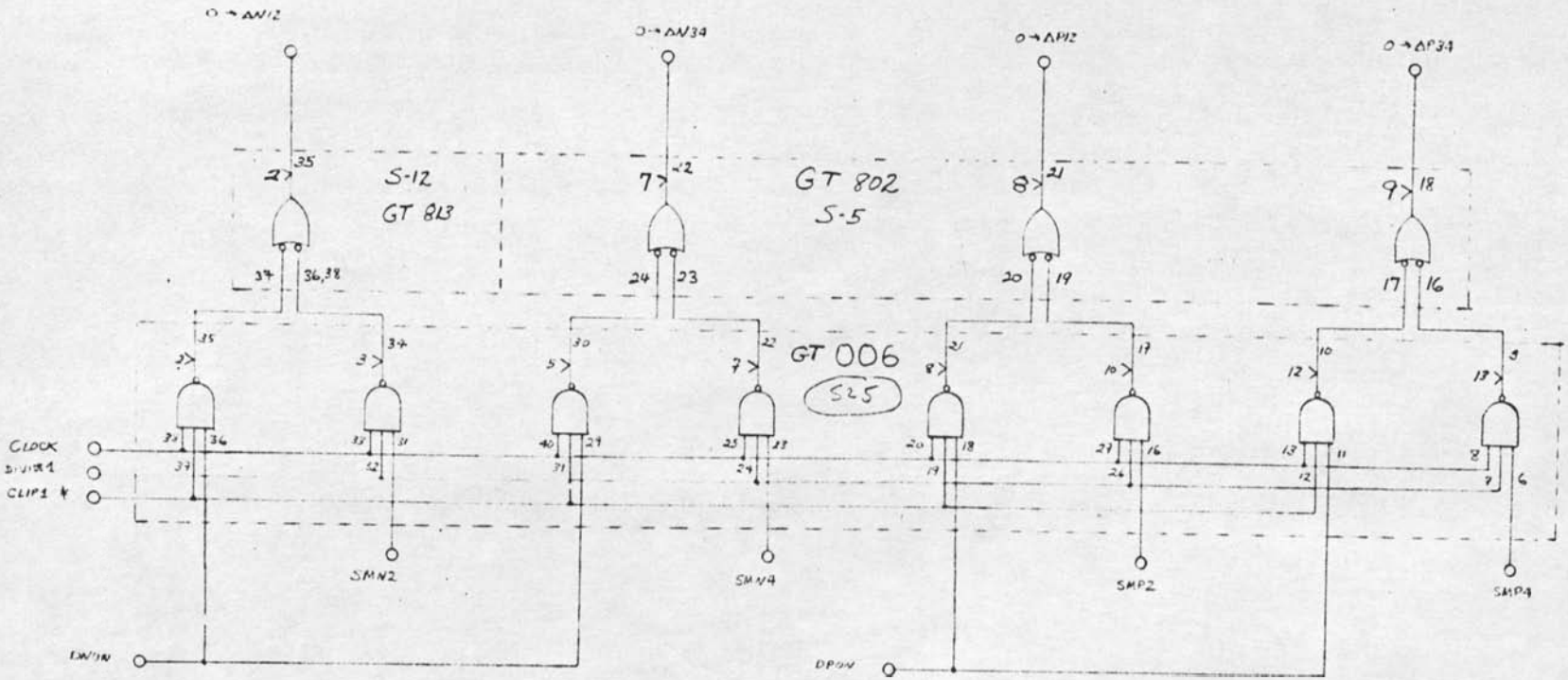
HARVARD UNIVERSITY

3-D DISPLAY

CLIPPER CONTROL

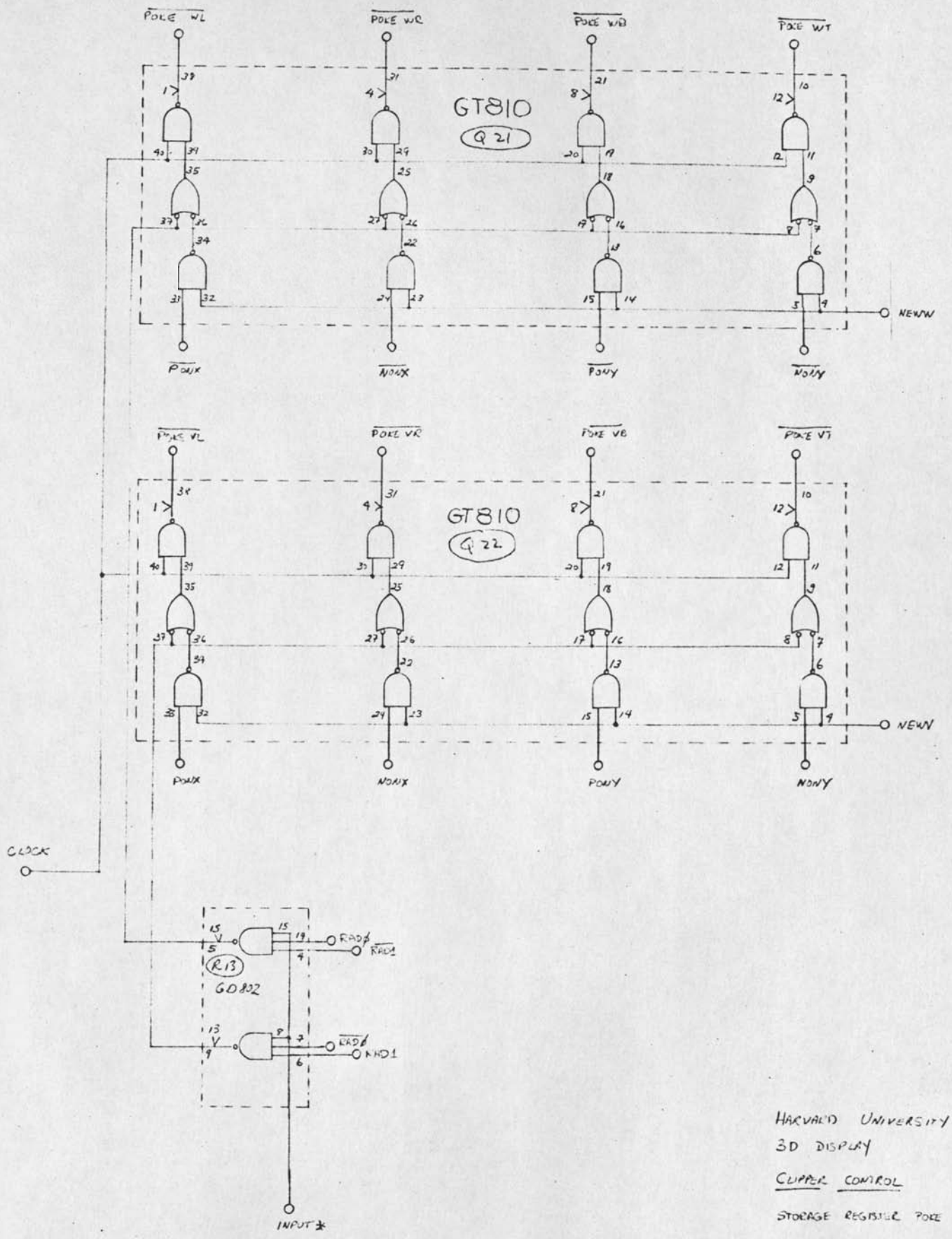
PDLG DELTA LOGIC

CC 10



HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER CONTROL
ZERO-DELTA LOGIC

CC 11

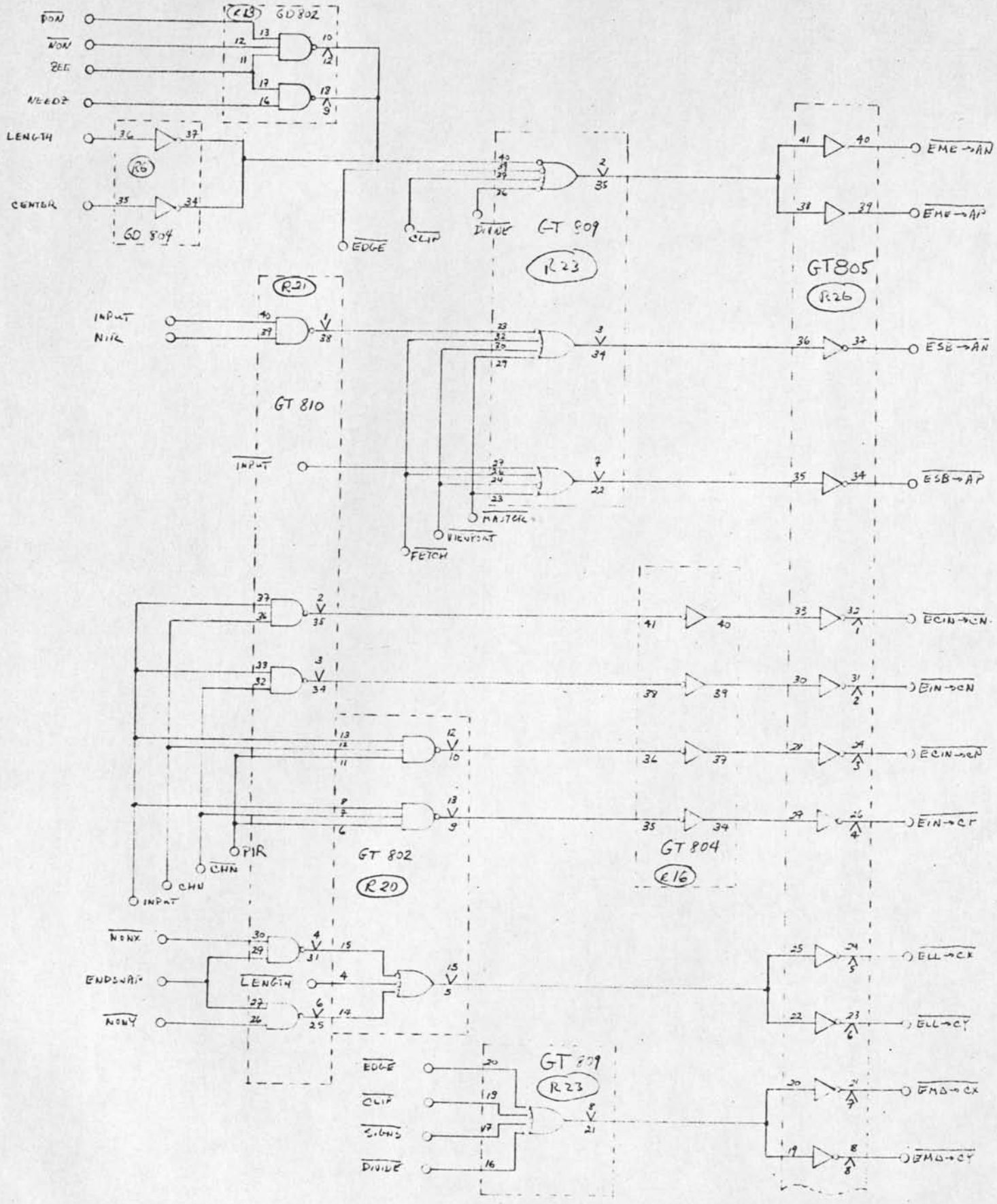


HARVARD UNIVERSITY

3D DISPLAY

CLIPPER CONTROL

STORAGE REGISTER POKE LOGIC


GD802
GT 809
GT805
GT 810
GT 802
GT 804
GT 809
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
GATE ENABLE LOGIC
CC 13

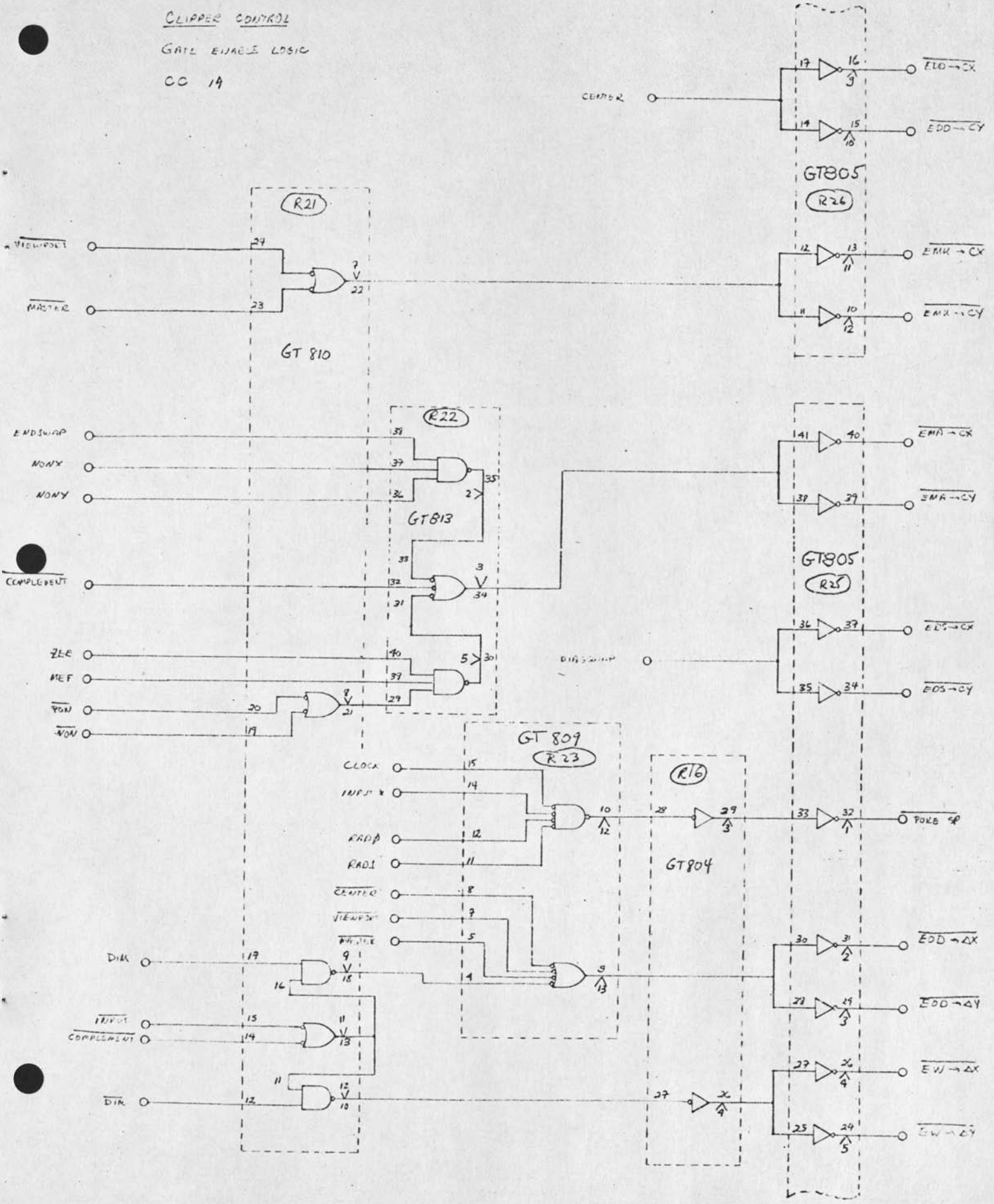
HARVARD UNIVERSITY

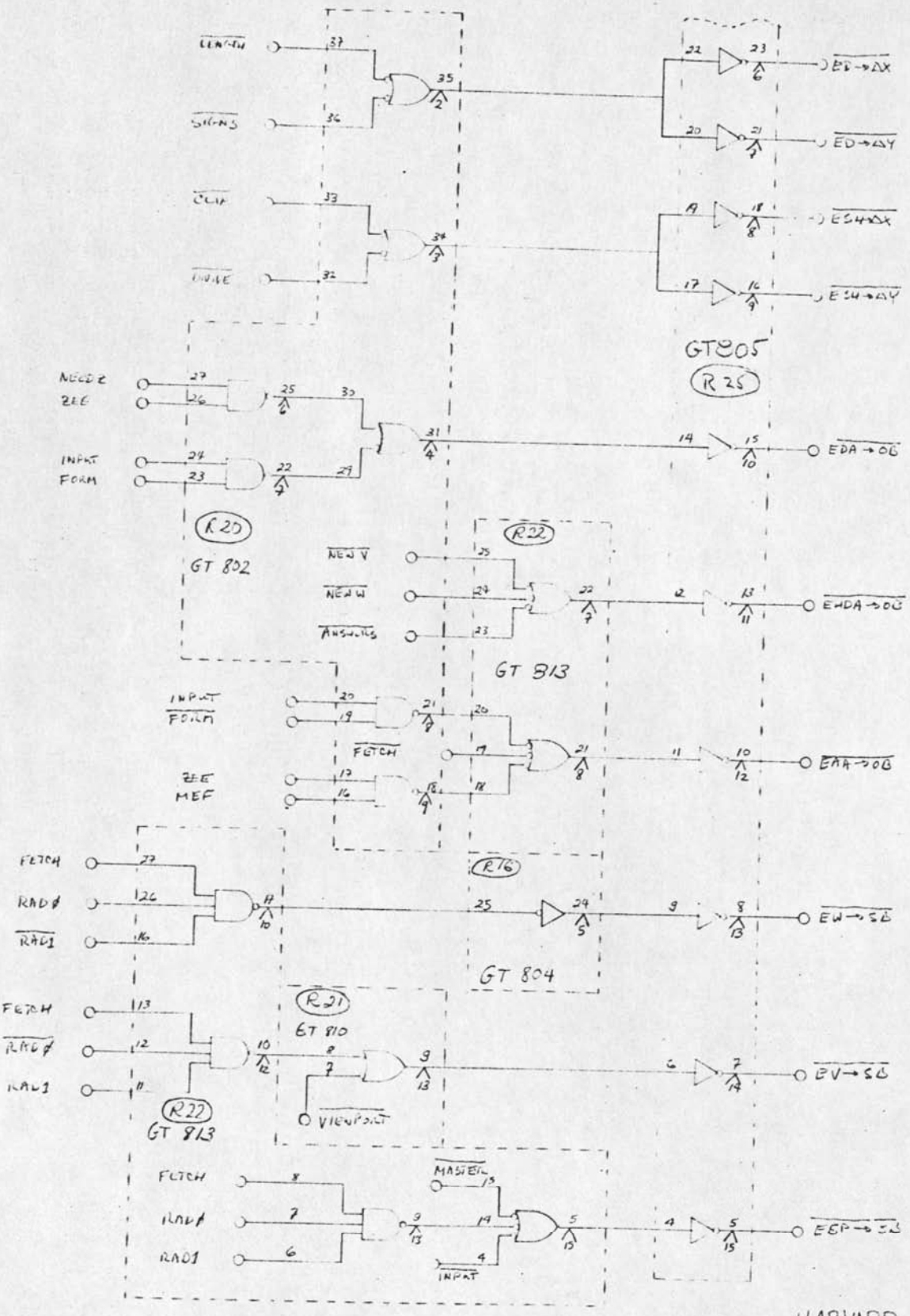
3D DISPLAY

CLIPPER CONTROL

GATE ENABLE LOGIC

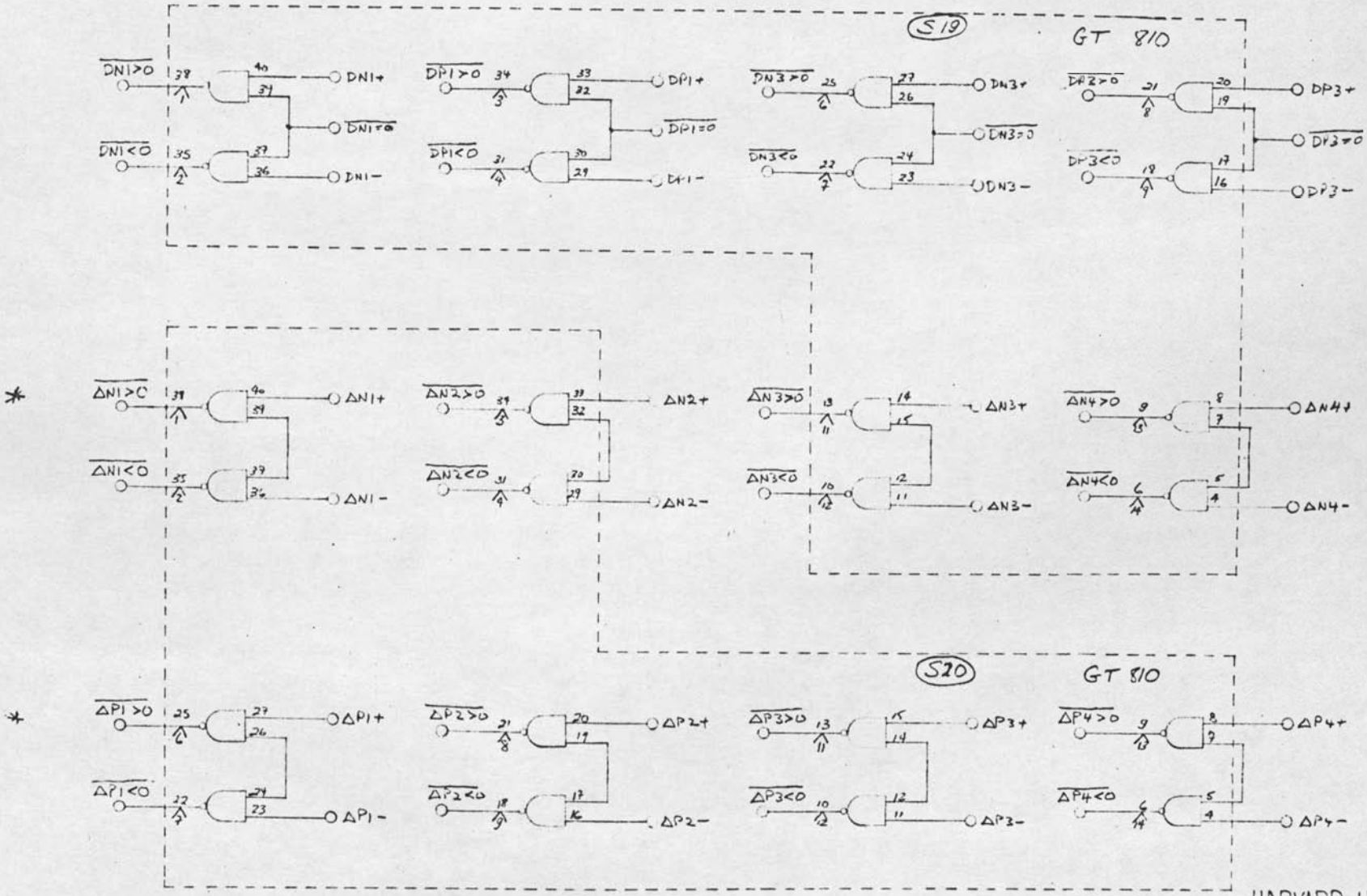
CC 14





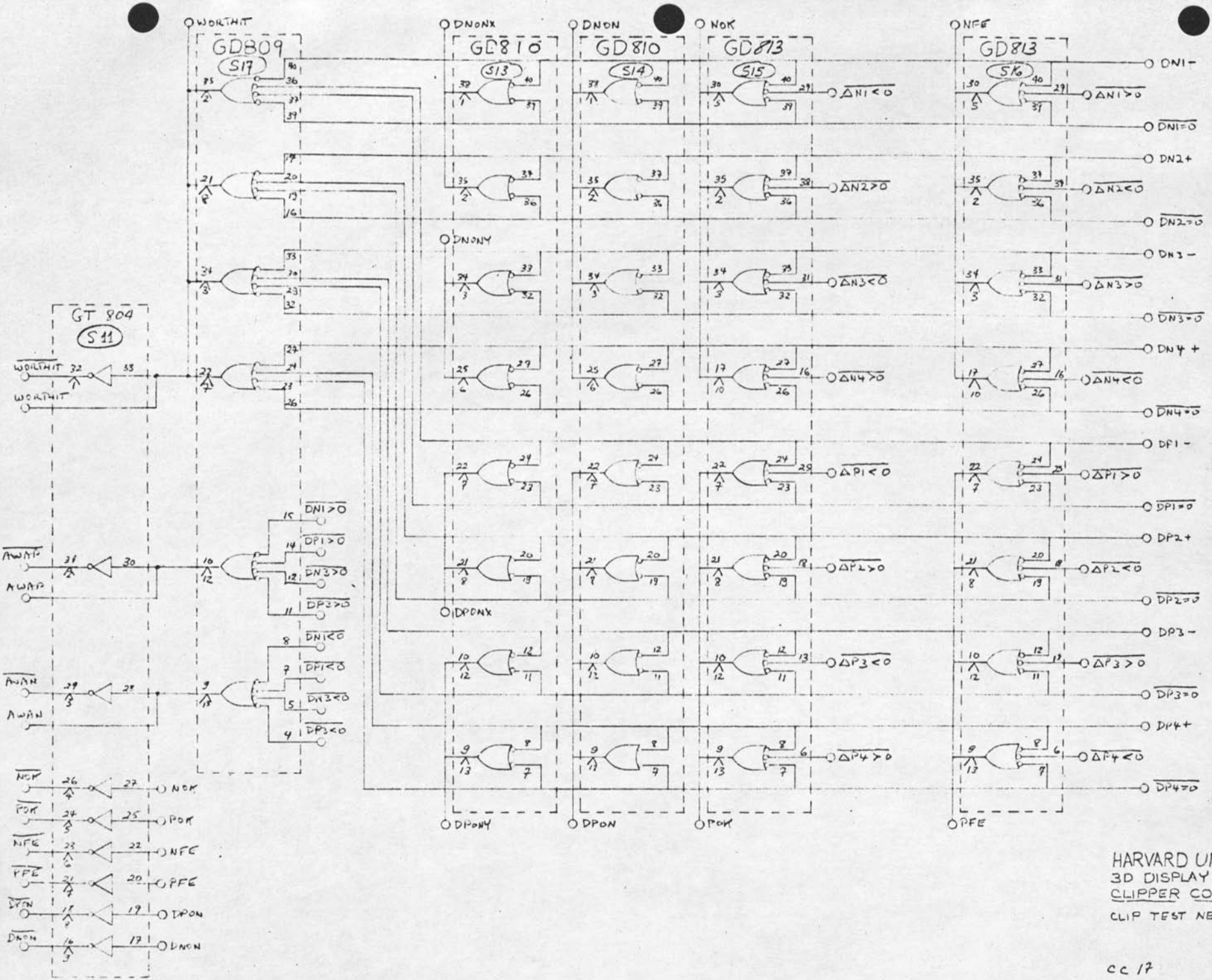
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
GATE ENABLE LOGIC

CCIT



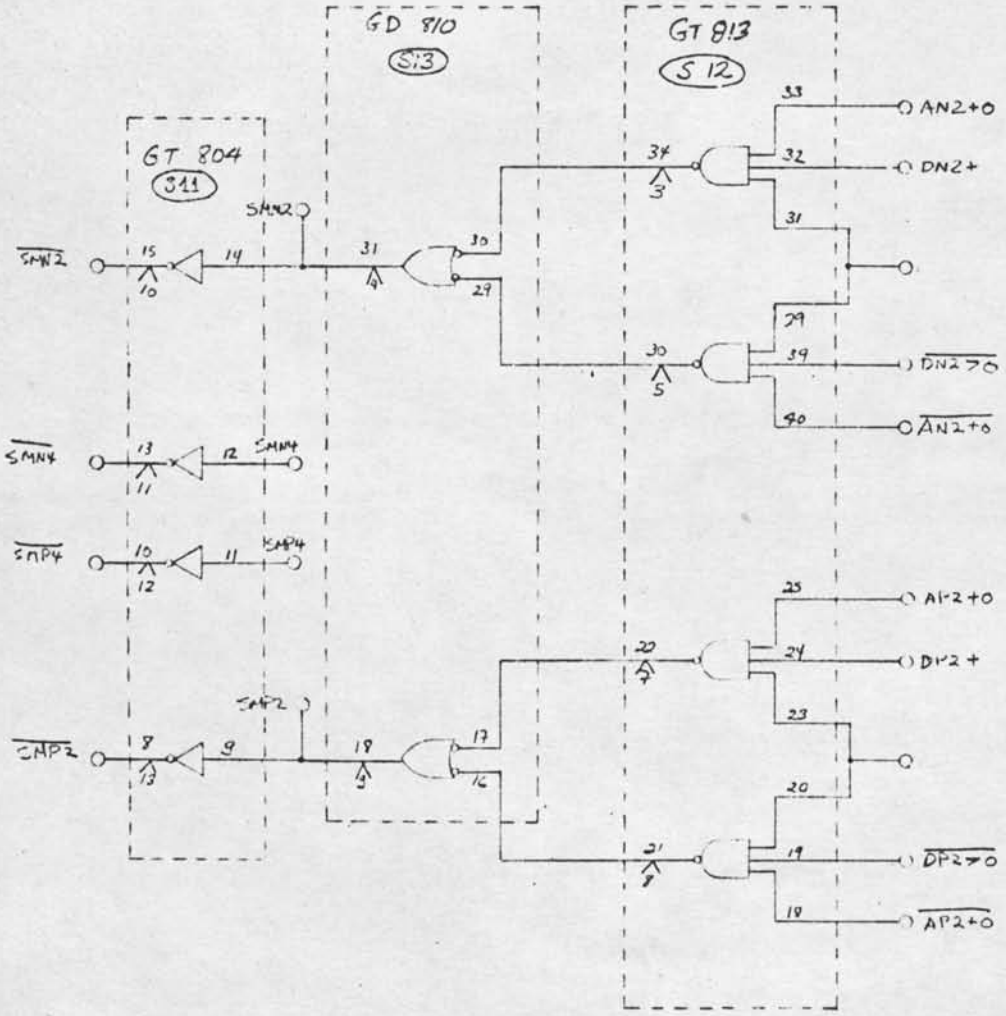
* NOTE THE WEIRD INPUTS, THESE ARE DUE TO CORRECTION OF AN ERROR

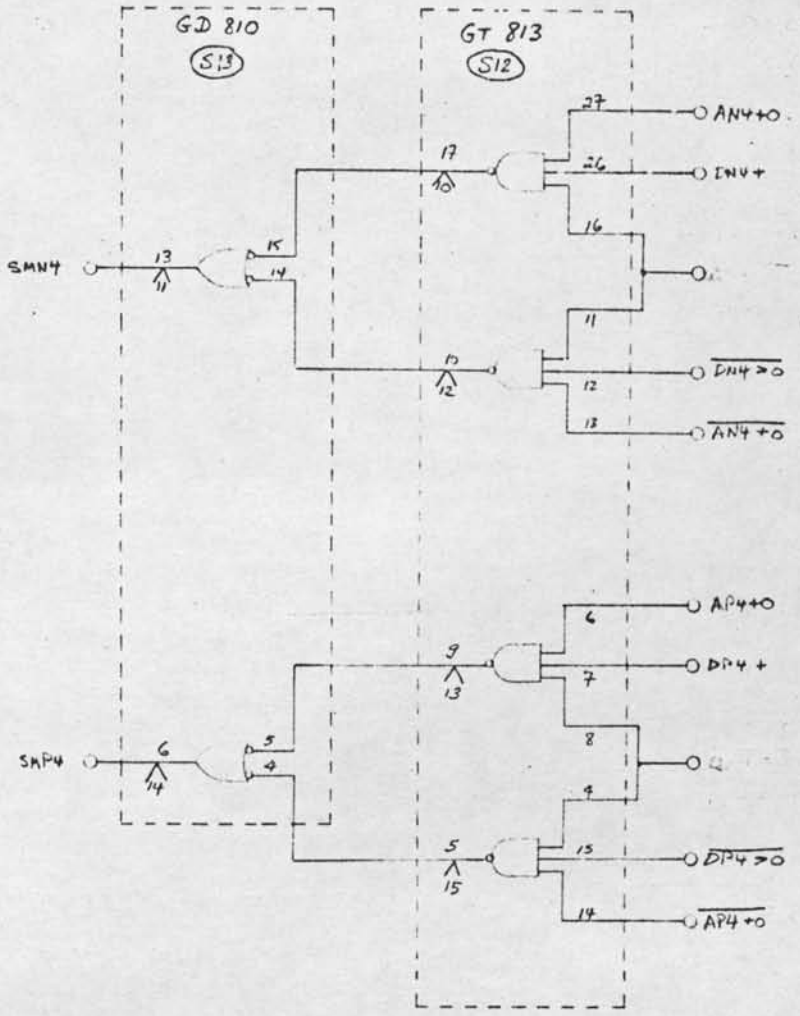
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
INEQUALITY NETS



HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
CLIP TEST NETS

cc 17





$SMN2 = \{(AN2+)\cdot(DN2+)\cdot(\overline{\Delta N2=0})\} \vee \{(AN2-)\cdot(DN2-)\cdot(\overline{\Delta N2=0})\}$

$SMN4 =$

$SMP2 =$

$SMP4 =$

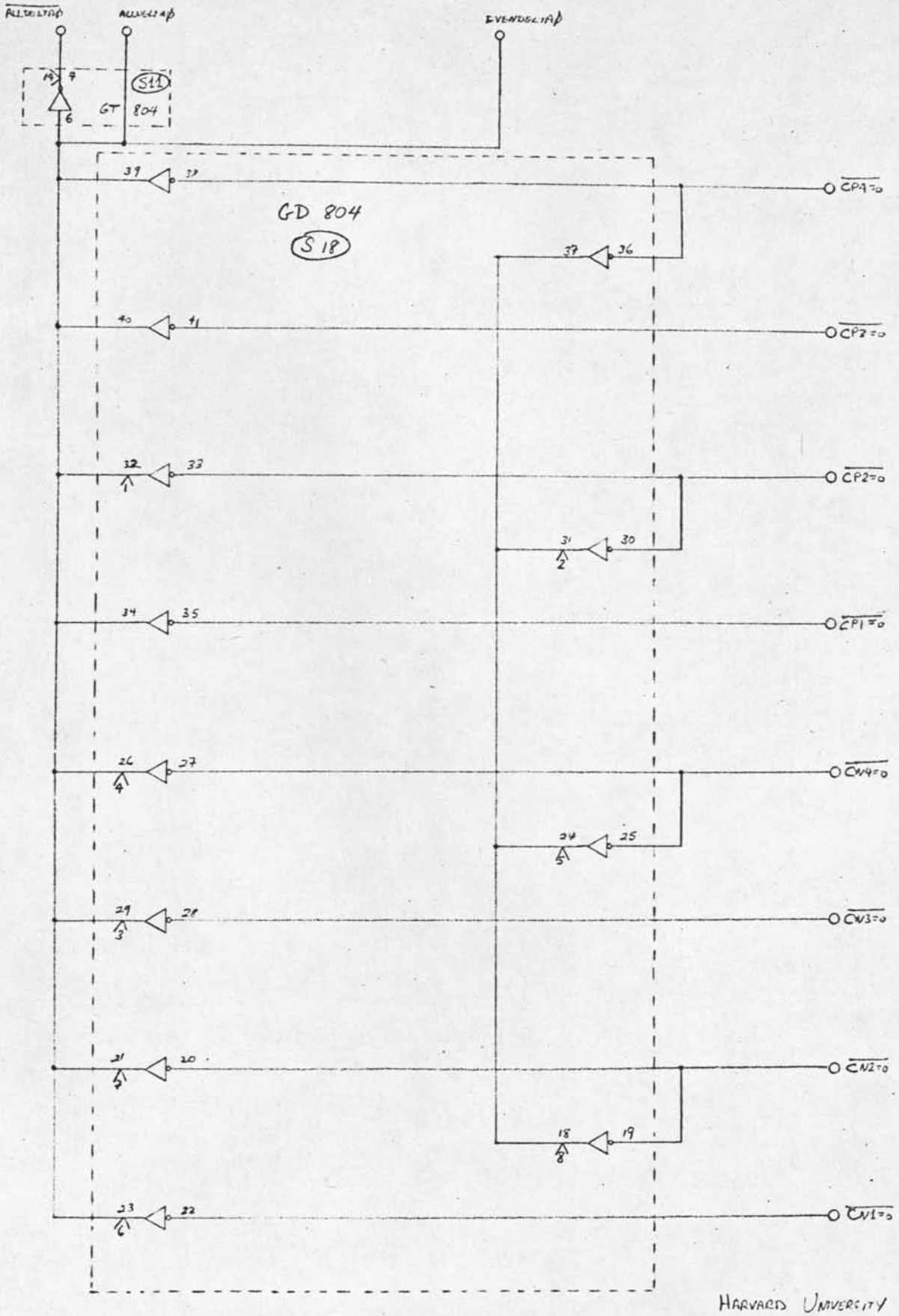
NB. ZERO DETECTION INCLUDED. WHEN $\Delta = 0$ SIGNS JUST DON'T MATCH. COMPLEMENT STEP INSURES THAT SIGN A = SIGN Δ.

HARVARD UNIVERSITY

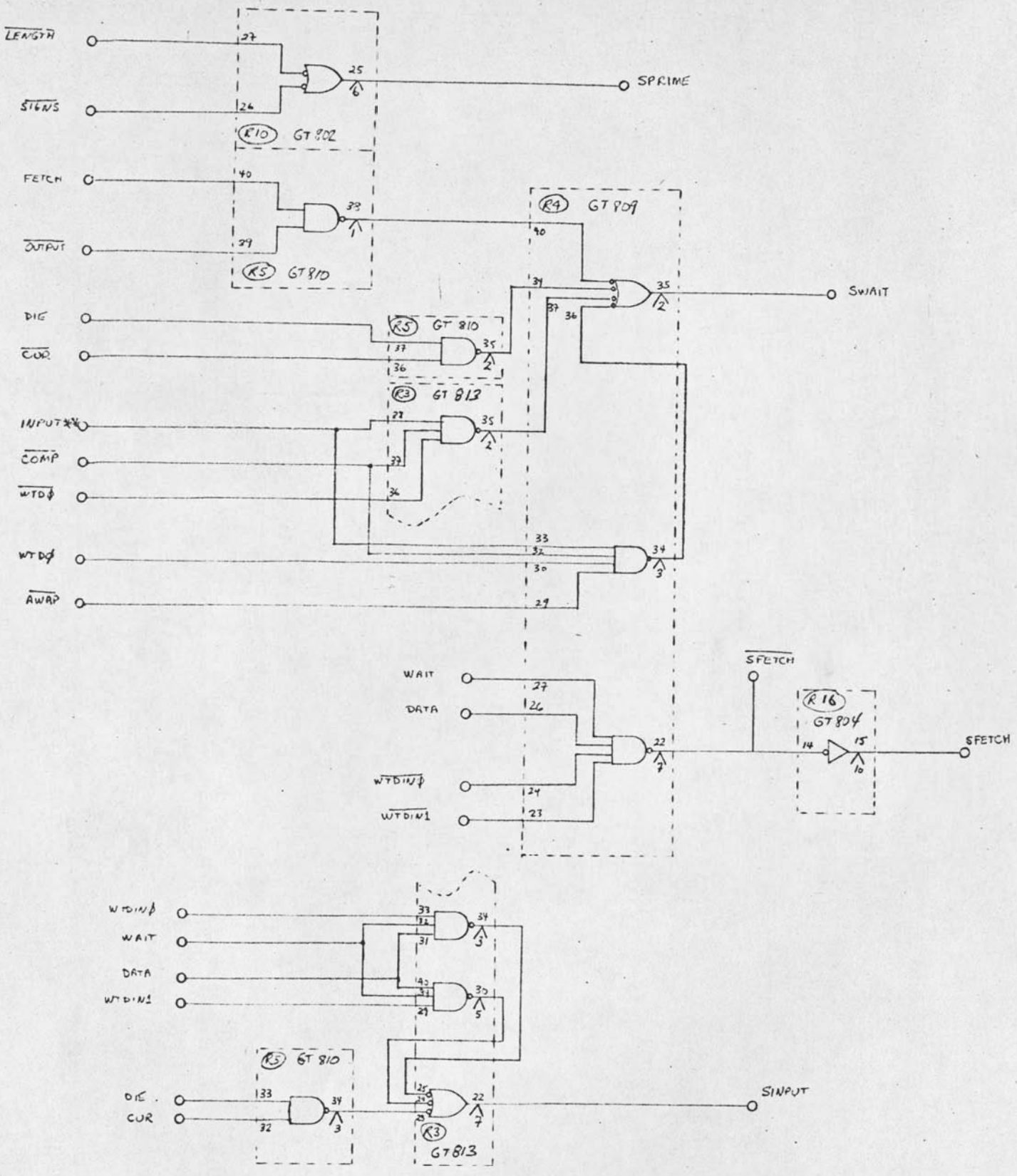
3D DISPLAY

CLIPPER CONTROL

SIGN MATCH LOGIC



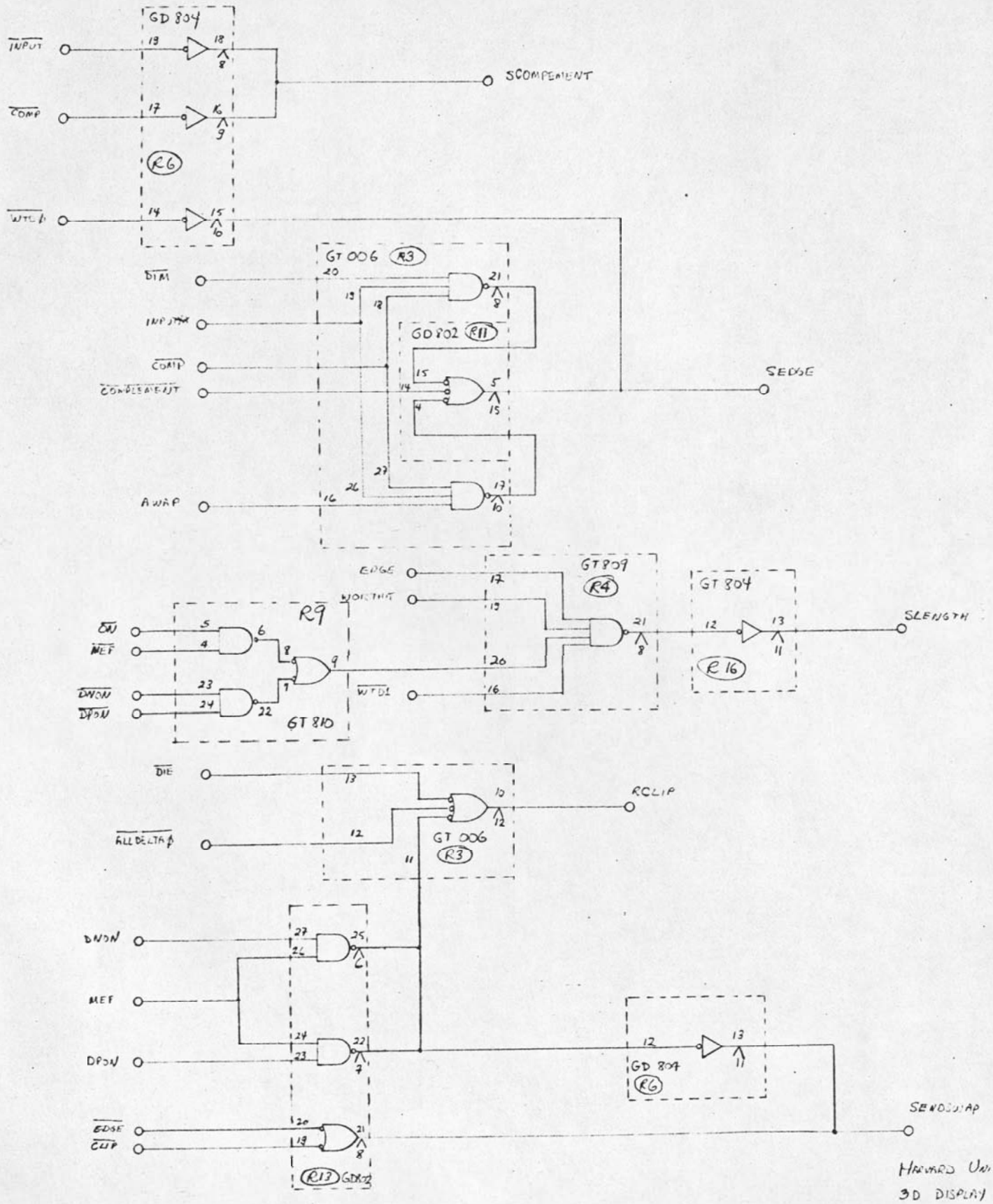
HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER CONTROL
DELTA ZERO FUNCTIONS
CC 19



HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER CONTROL

COMBINATIONAL LOGIC

CC '20

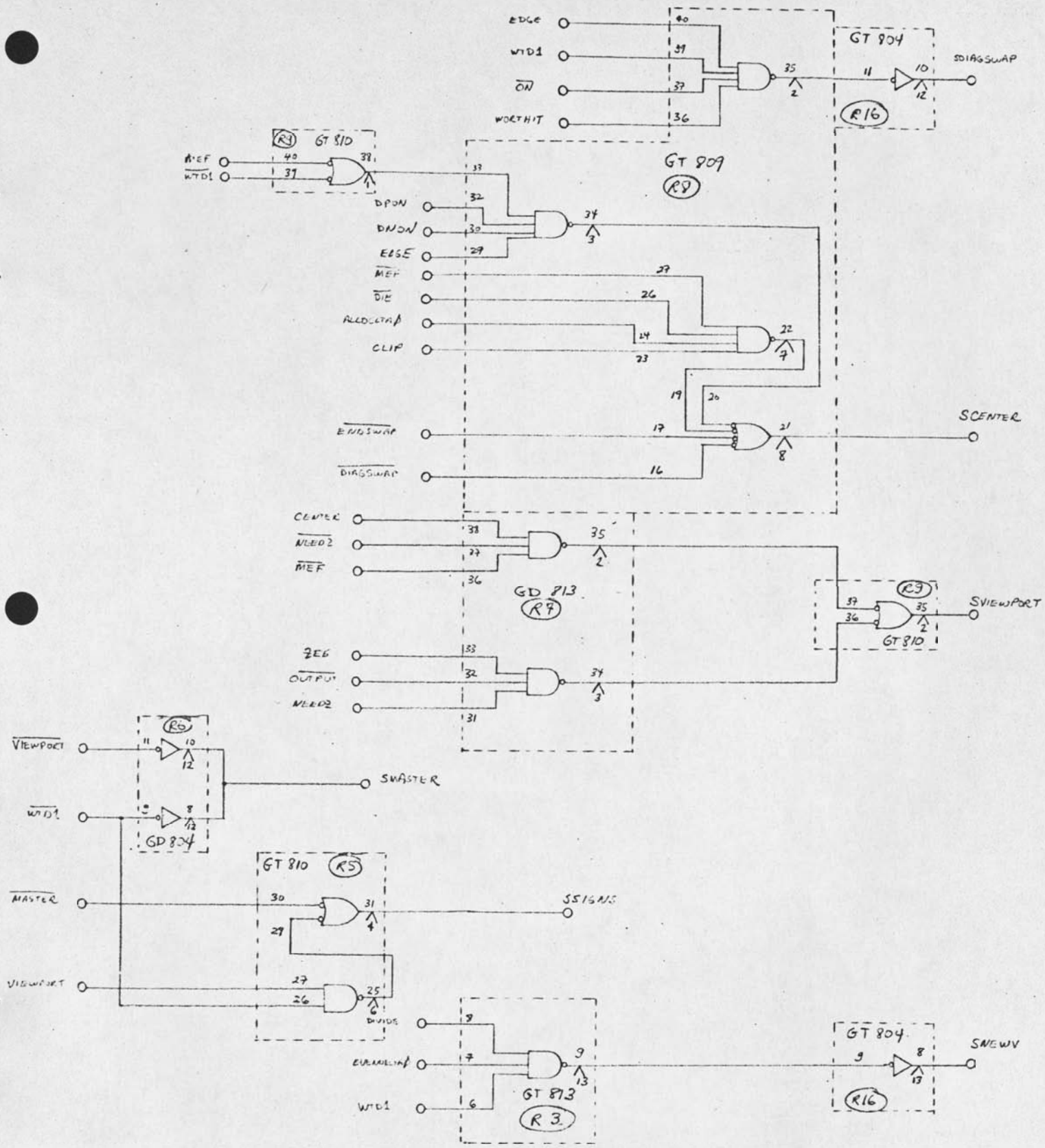


HARVARD UNIVERSITY

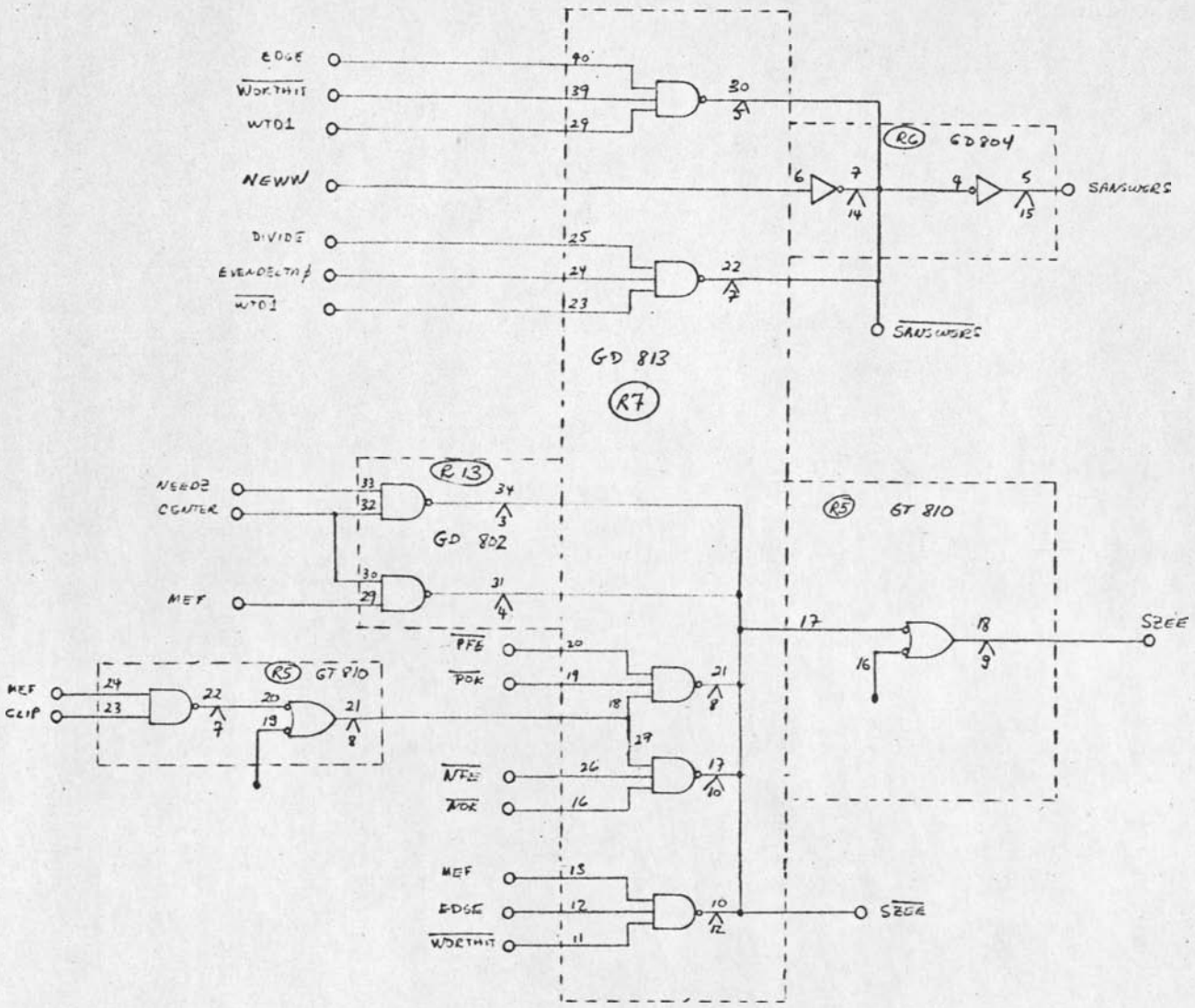
3D DISPLAY

CLIPPER CONTROL

COMBINATIONAL LOGIC



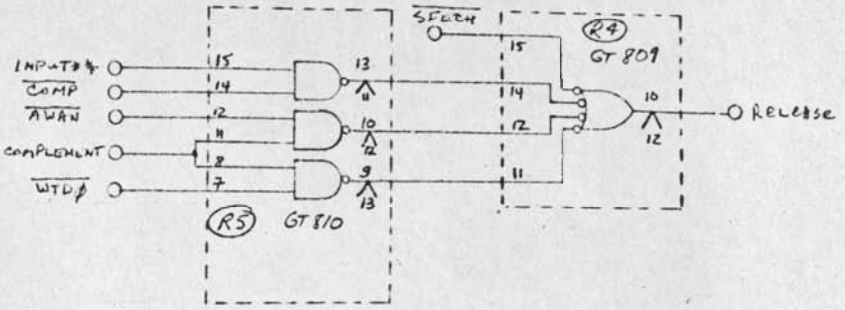
HARVARD UNIVERSITY
3 D DISPLAY
CLIPPER CONTROL
COMBINATIONAL LOGIC
CC 22

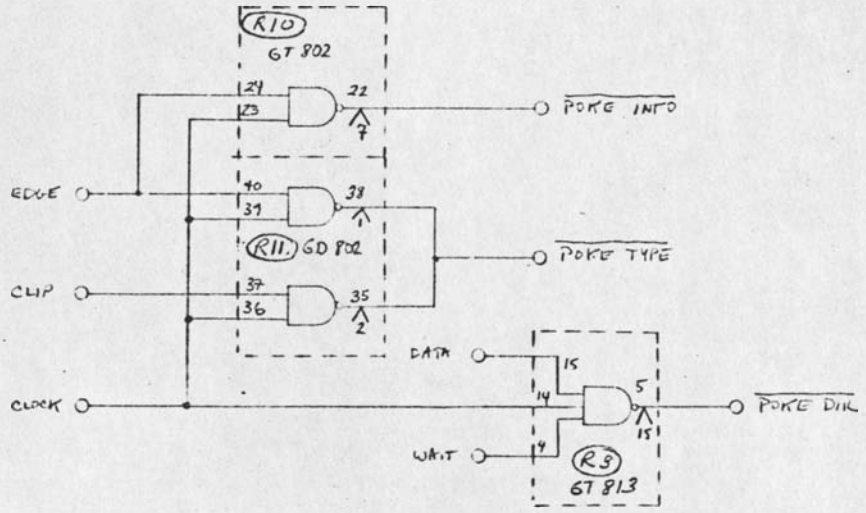


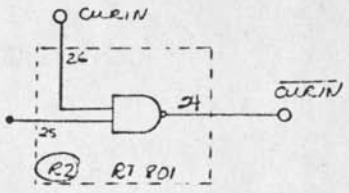
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL

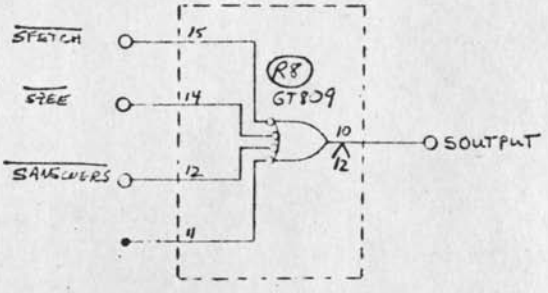
COMBINATIONAL LOGIC

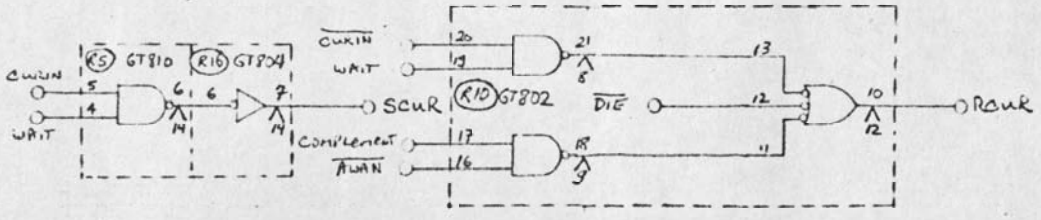
CC23





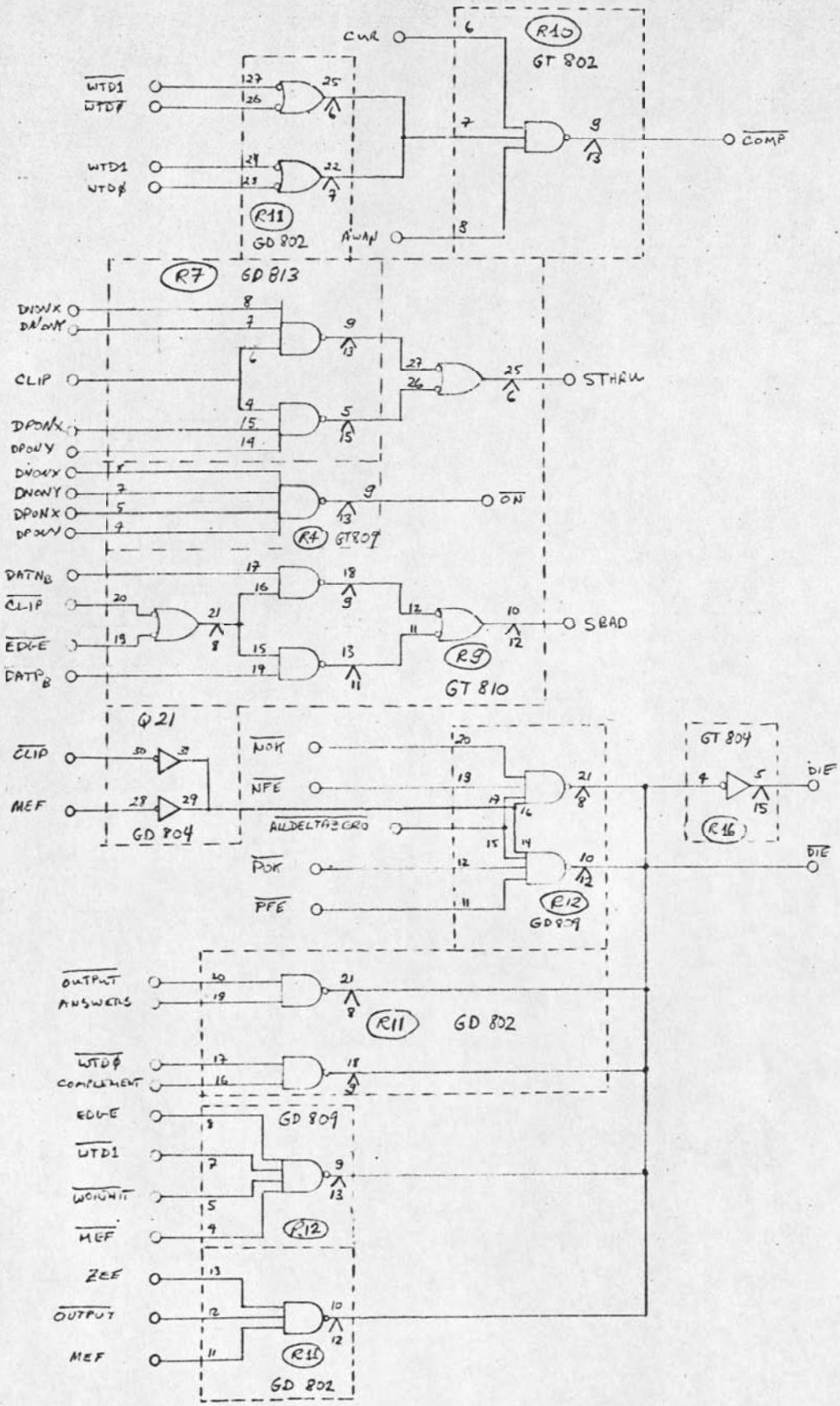






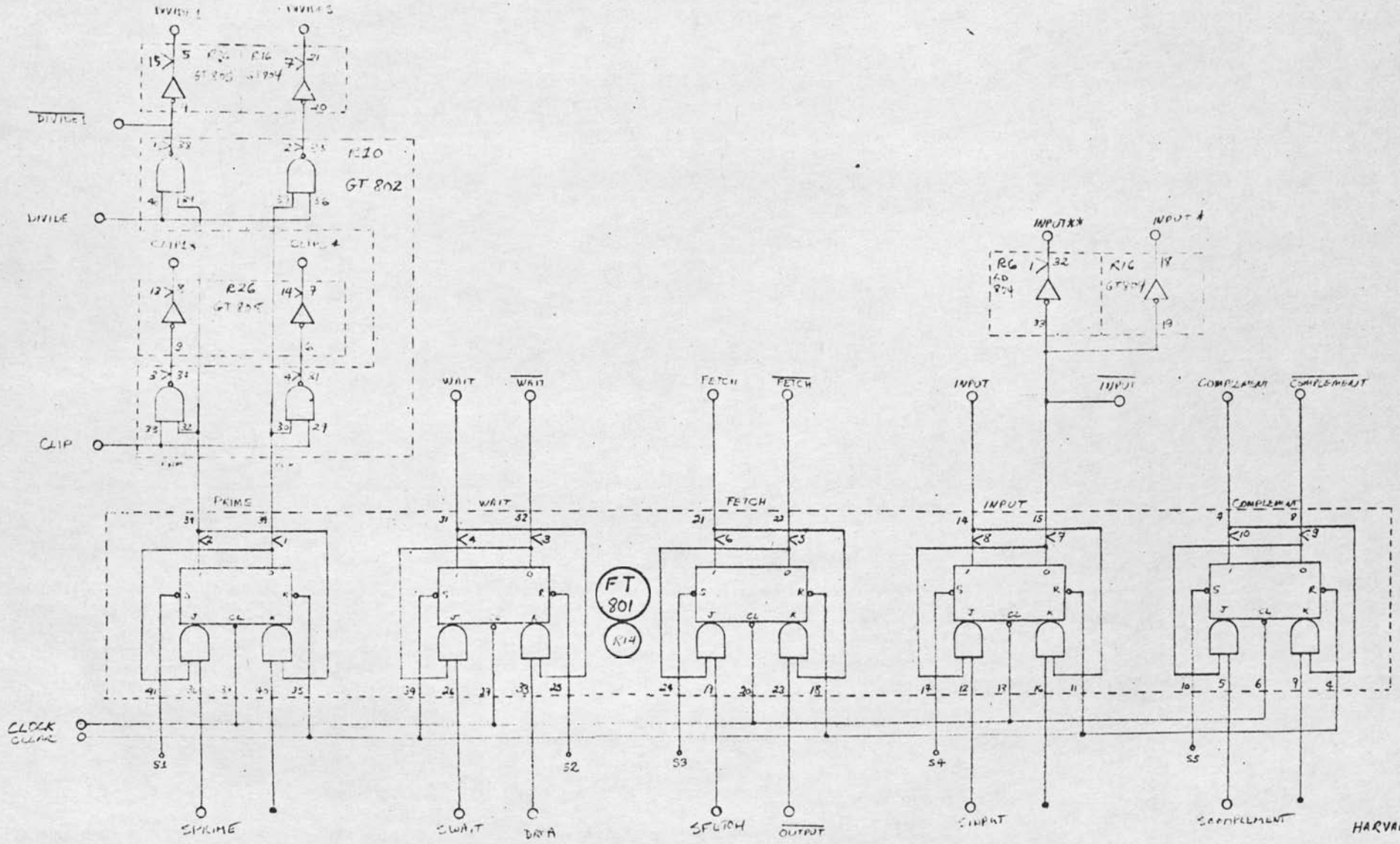
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL

COMBINITORIAL LOGIC



HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
COMBINATIONAL LOGIC

CC 25



HARVARD UNIVERSITY

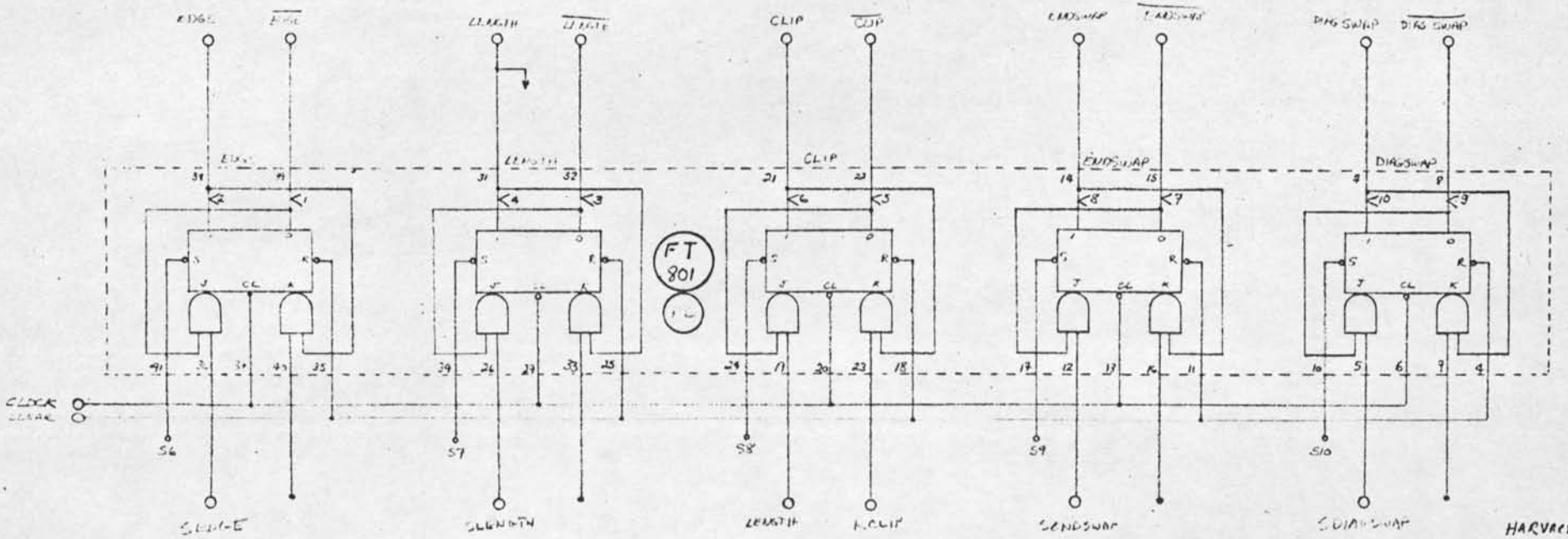
3D DISPLAY

CLIPPER CONTROL

STATE FLIP FLOPS

1 OF 5

CC 26



HARVARD UNIVERSITY

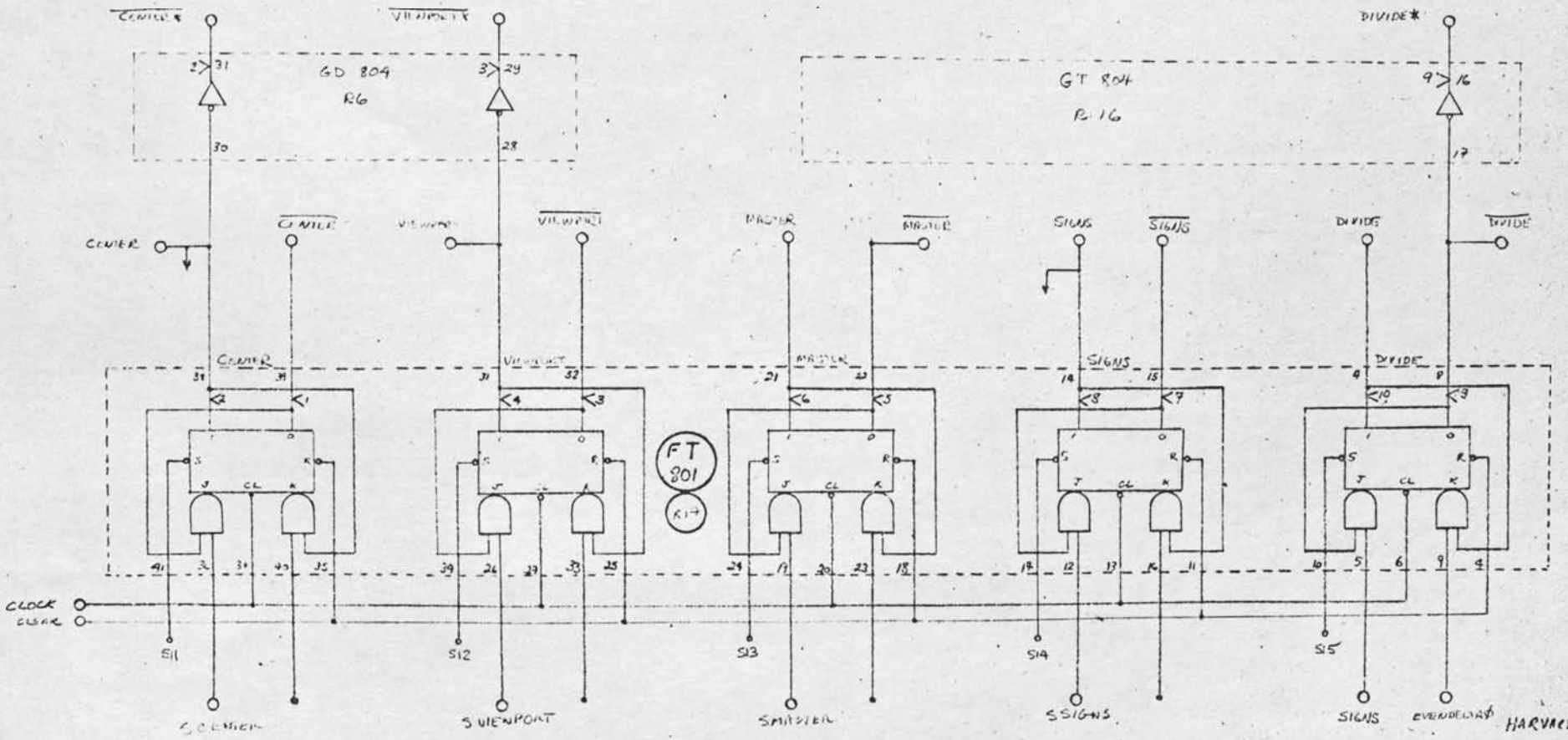
3D DISPLAY

CLIPPER CONTROL

STATE FLIP FLOPS

2 of 5

CC 27



HARVARD UNIVERSITY

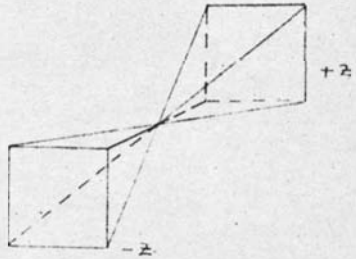
3D DISPLAY

CLIPPER CONTROL

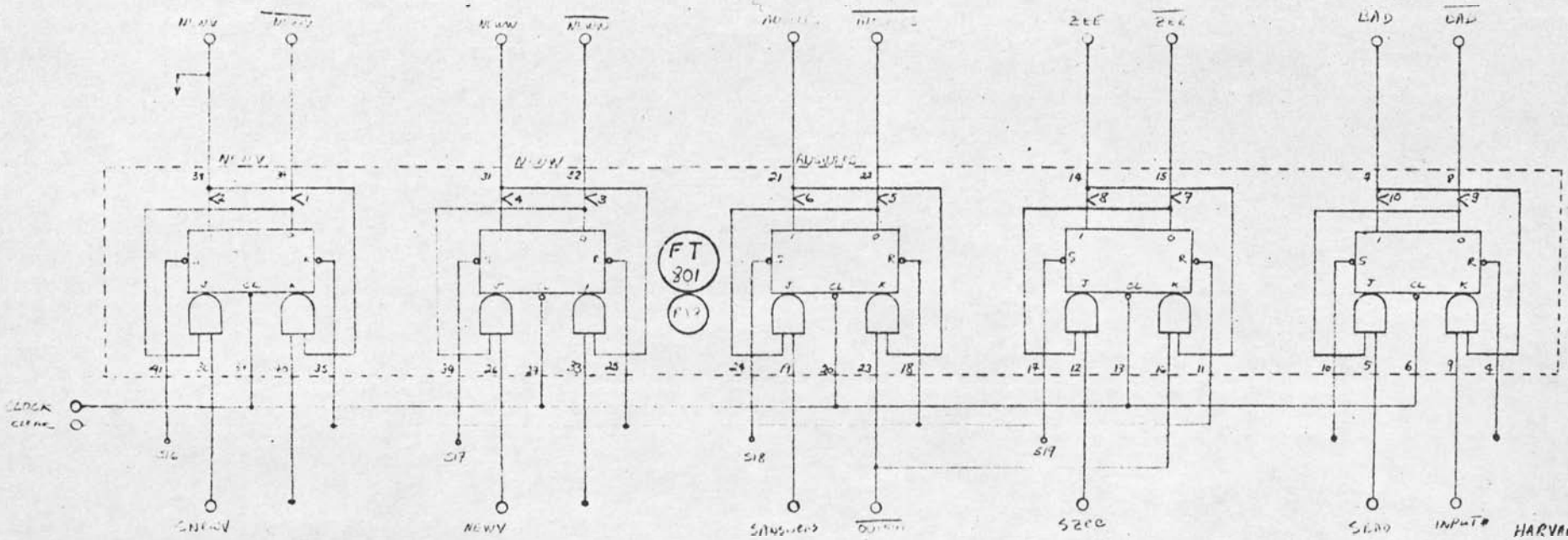
STATE FLIP FLOPS

3 OF 5

CC 28



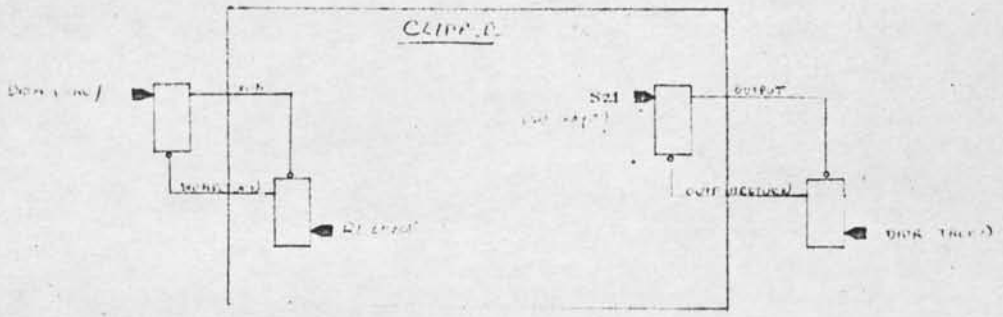
BAD = LINE SENT THRU THE -Z PYRAMID

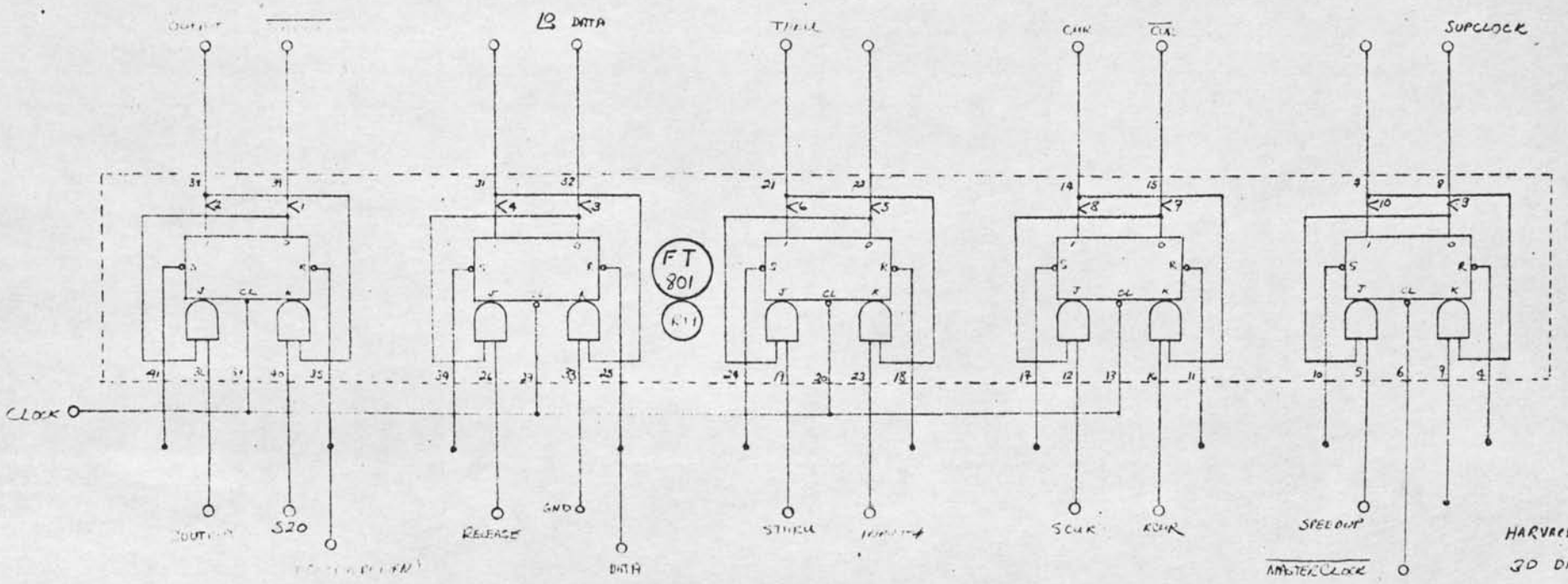


HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL

STATE FLIP FLOPS
4 OF 5
CC 29

SCHEMATIC OF INPUT OUTPUT





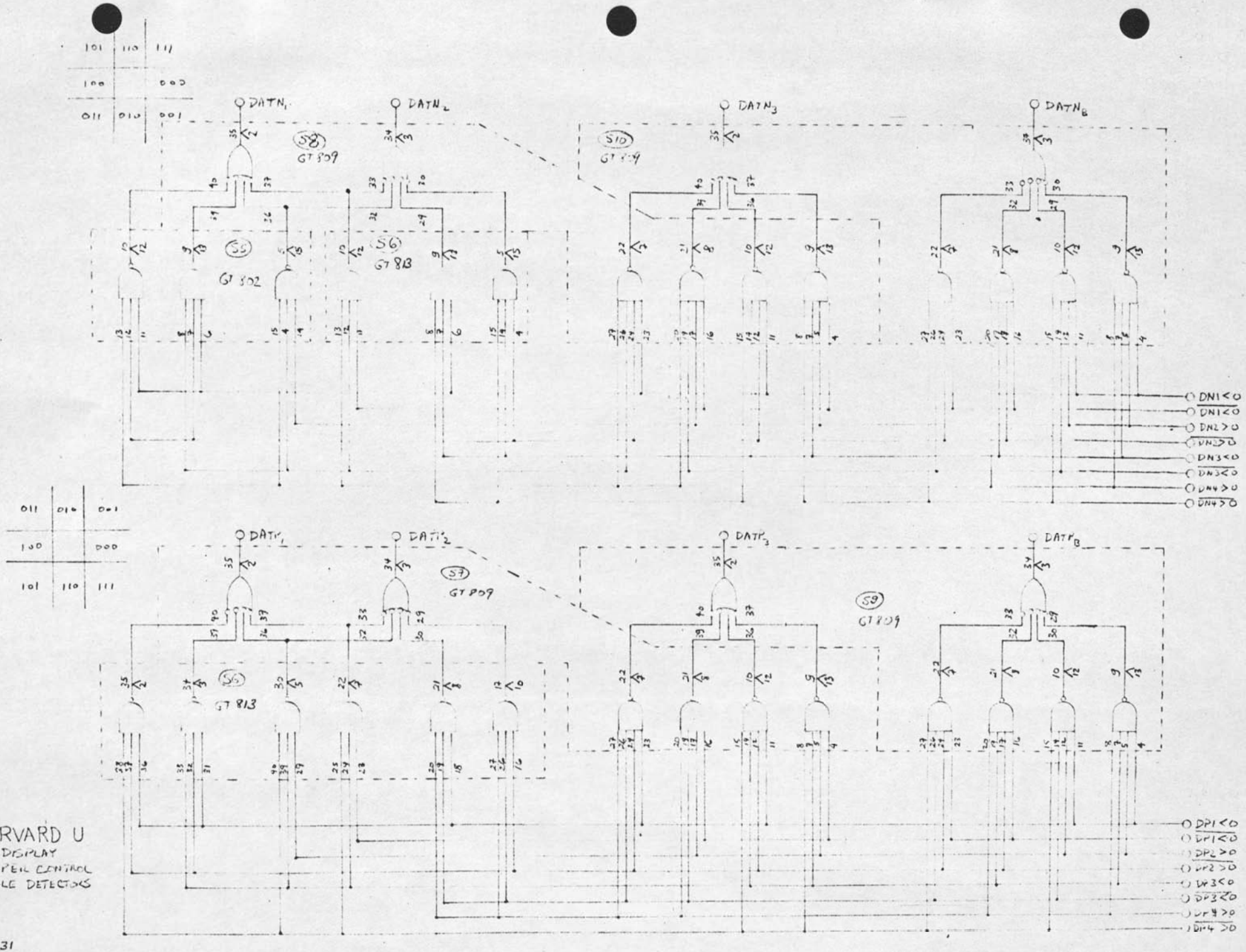
HARVARD UNIVERSITY

3D DISPLAY

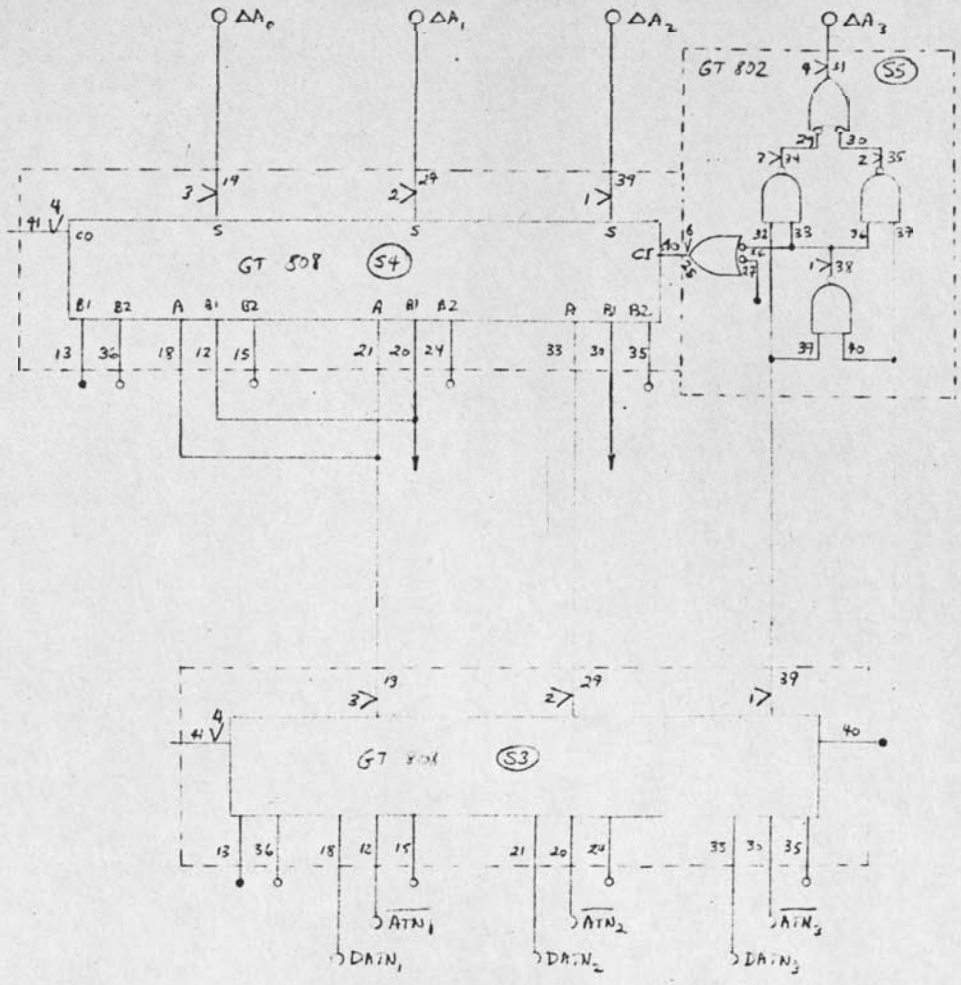
CLIPPER CONTROL

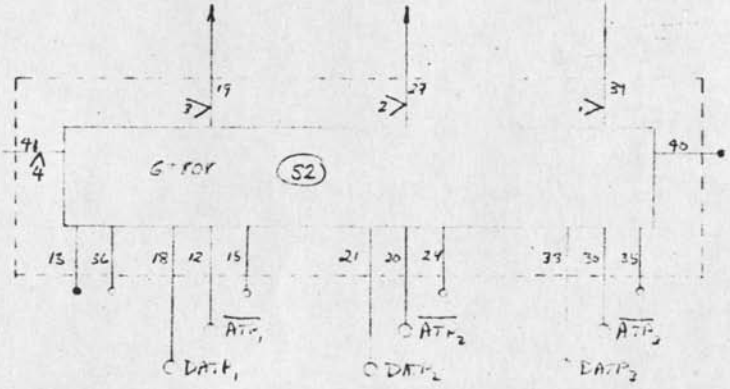
STATE FLIP FLOPS

CC 30



HARVARD U
3D DISPLAY
CLIPPER CONTROL
ANGLE DETECTORS





ANGLE OF POINT N

ANGLE OF REJECT POINT

ANGLE OF POINT P

$$\Delta A = \underbrace{[AN - AR]}_{\text{MOD 8}} + \underbrace{[AR - AP]}_{\text{MOD 8}}$$

$$-3 \leq \underbrace{[AN - AR]}_{\text{MOD 8}} \leq +3$$

SO LET DATN BE (= ANGLE OF NEW POINT). I.E. COULD ADD

101	110	111
100		000
011	010	001

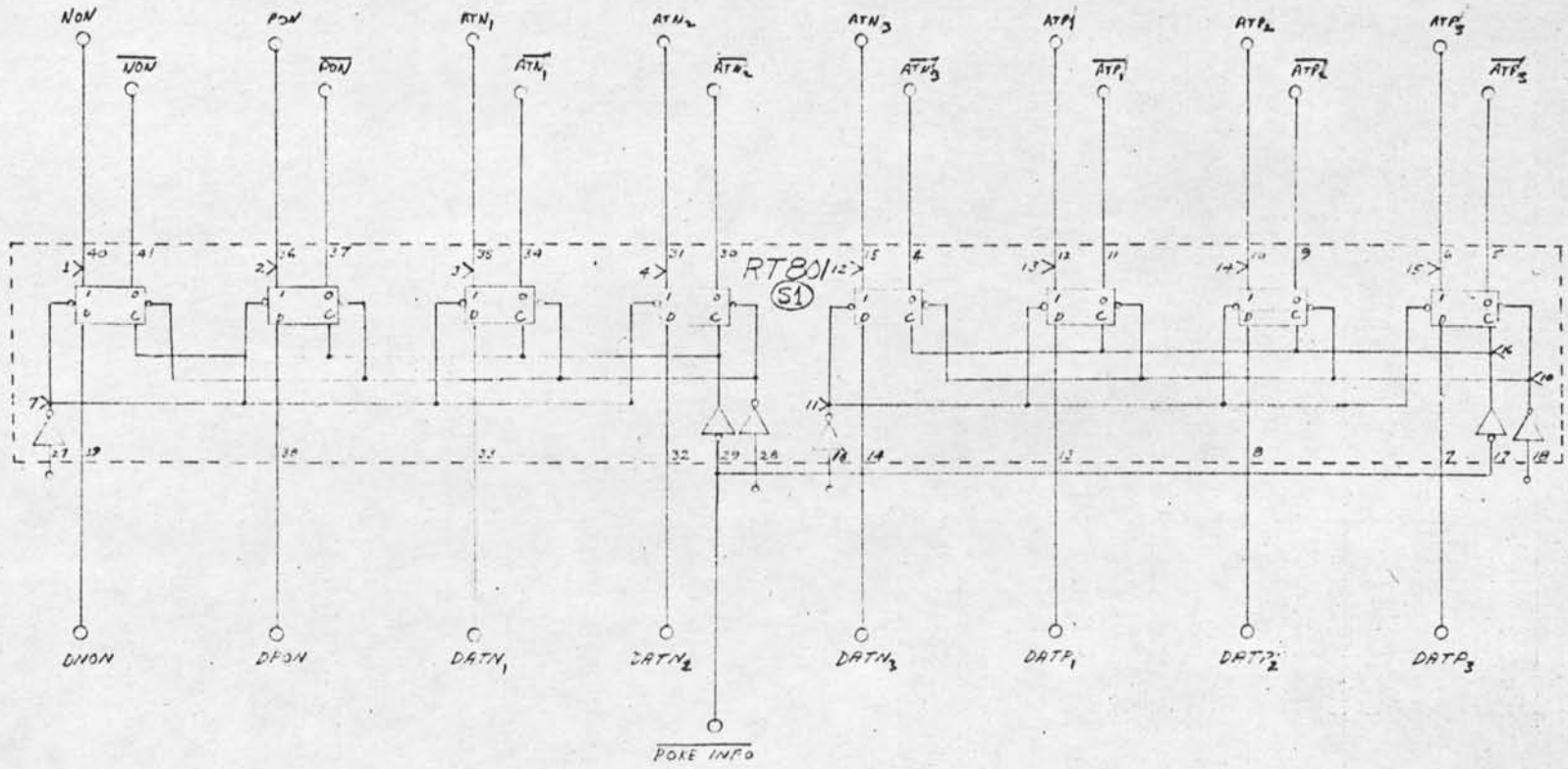
DATN

011	010	001
100		000
101	110	111

DATP

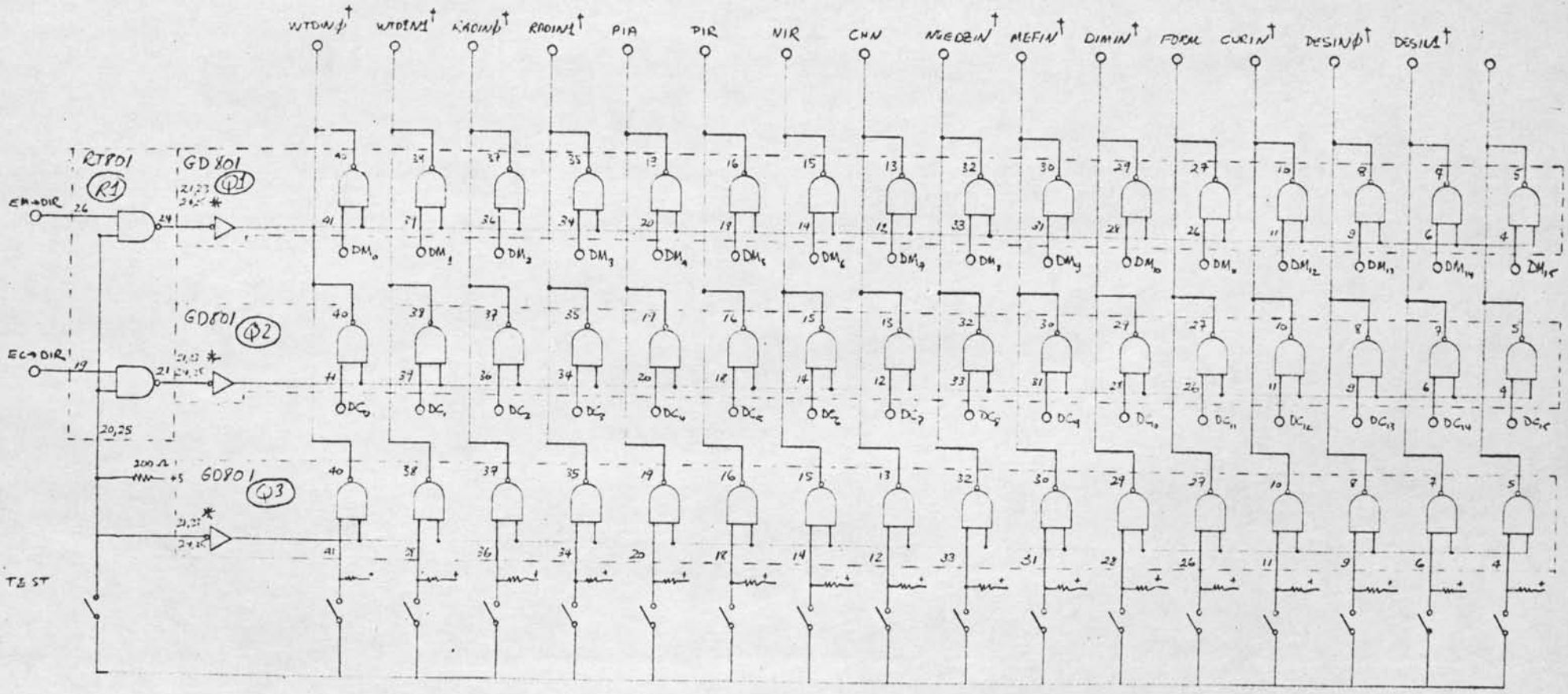
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
ANGLE ADDERS

CC 32



HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER CONTROL

CC 33



NOTES

† THESE ARE BUFFERED DIRECTIVE BITS

* SCHEMATIC ONLY- 4 INVERTERS ARE DRIVEN

— THE BUS READS TRUE; THE INPUTS ARE COMPLEMENTED

HARVARD UNIVERSITY
3-D DISPLAY
[CLIPPER CONTROL
DIRECTIVE INPUT GATE

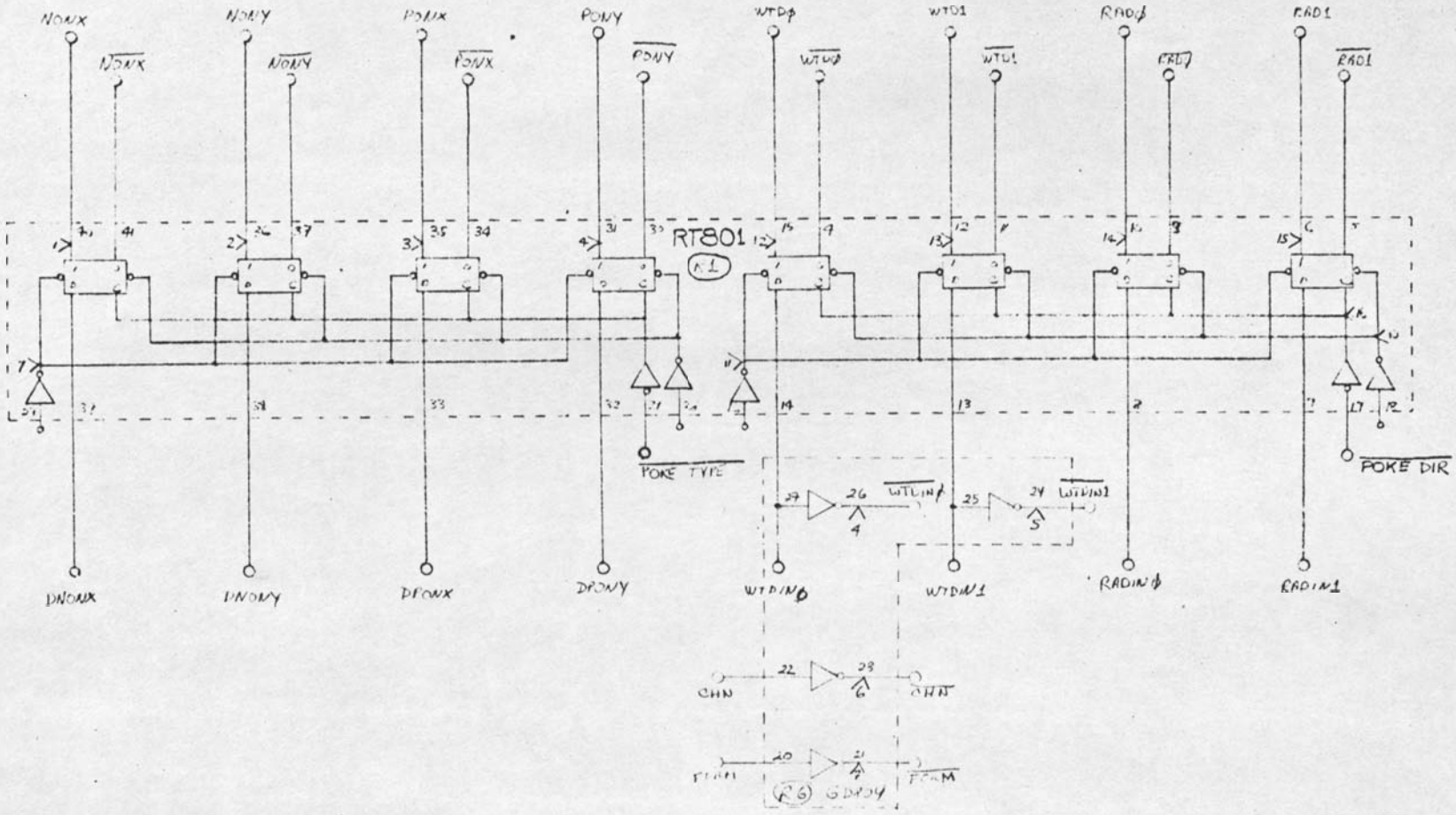
CC 34

WTD (WHAT to Do)

00	FETCH
01	INPUT
10	INPUT (LINE
11	INPUT (BOX

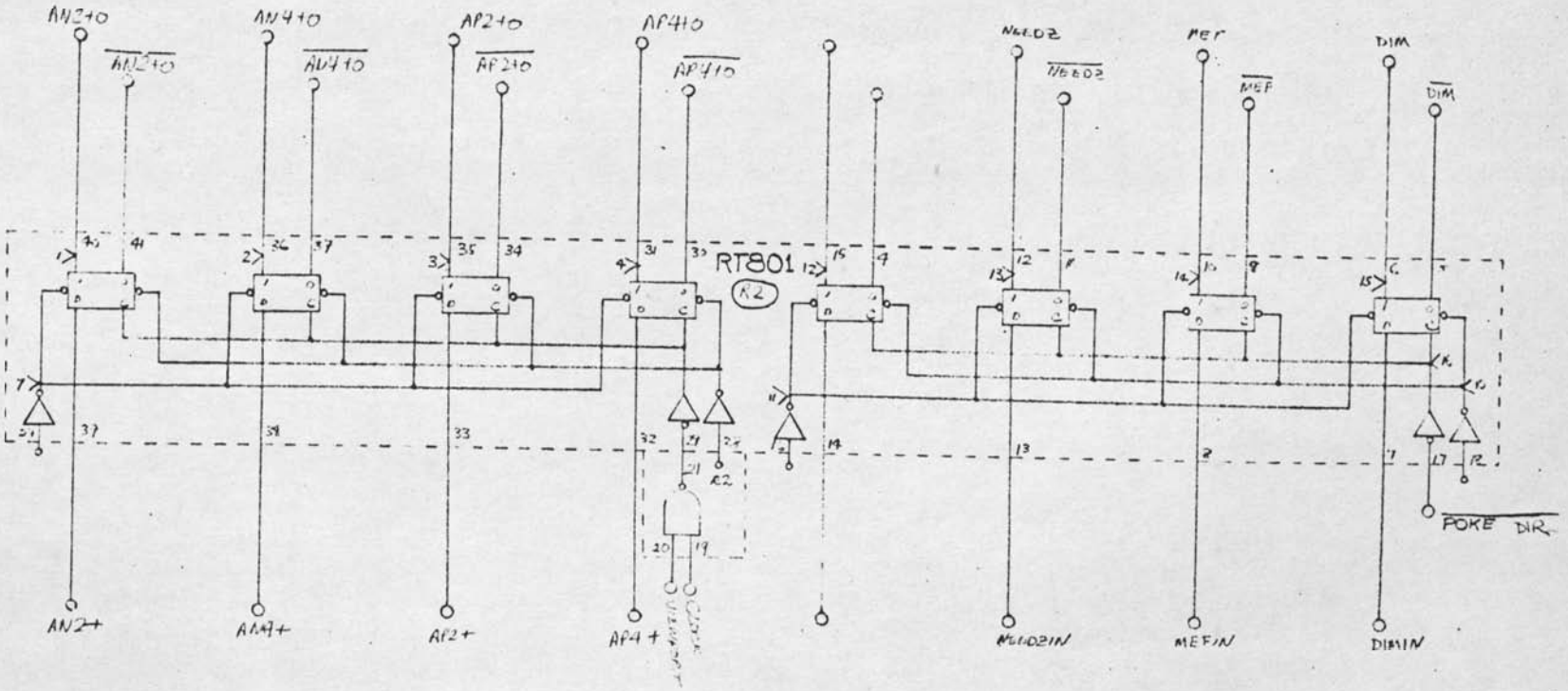
RAD (Register Address)

00	
01	VIEWPORT
10	WINDOW
11	SAVE POINT



HUMAN UNIVERSITY
3 D DISPLAY
CURSOR CONTROL

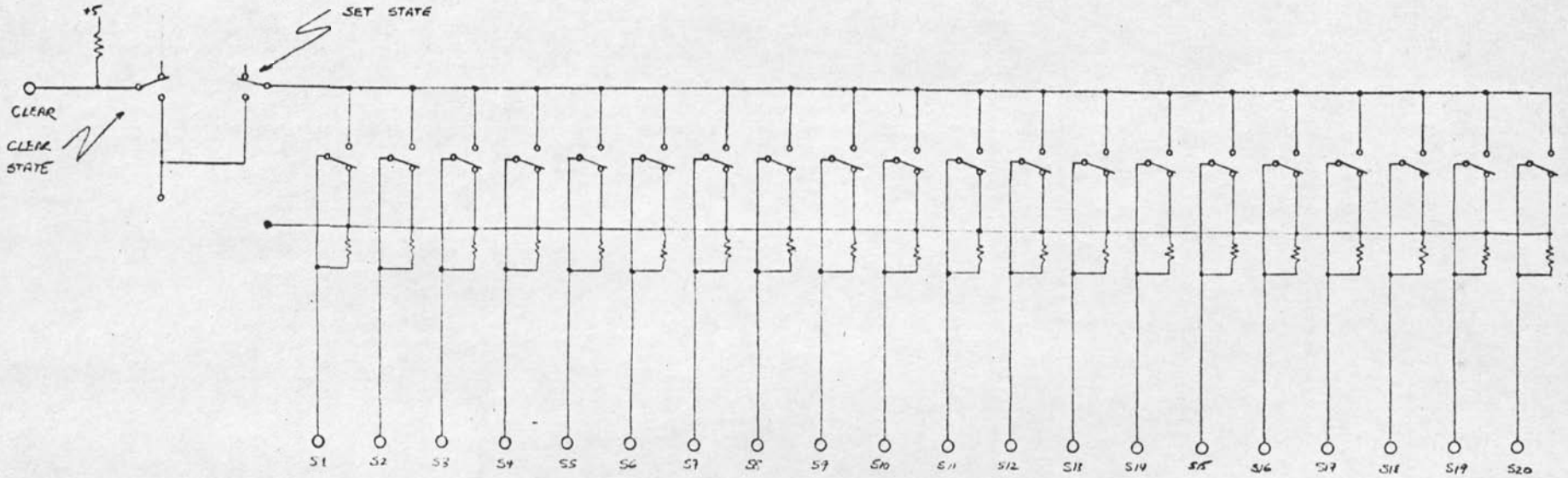
CC 35



HARVARD UNIVERSITY
3-D DISPLAY

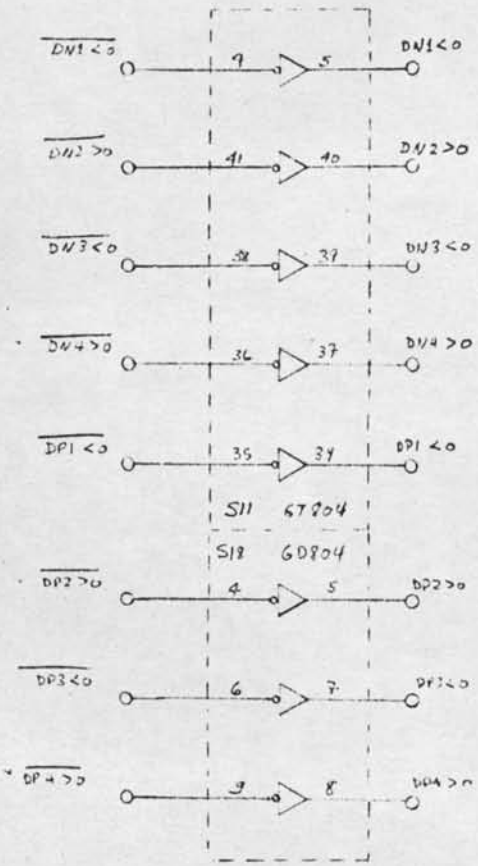
CLIPPER CONTROL

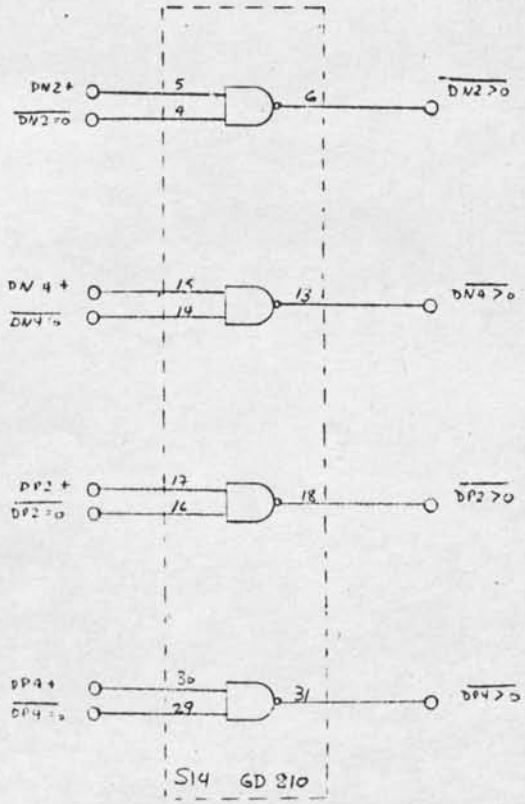
CC 36

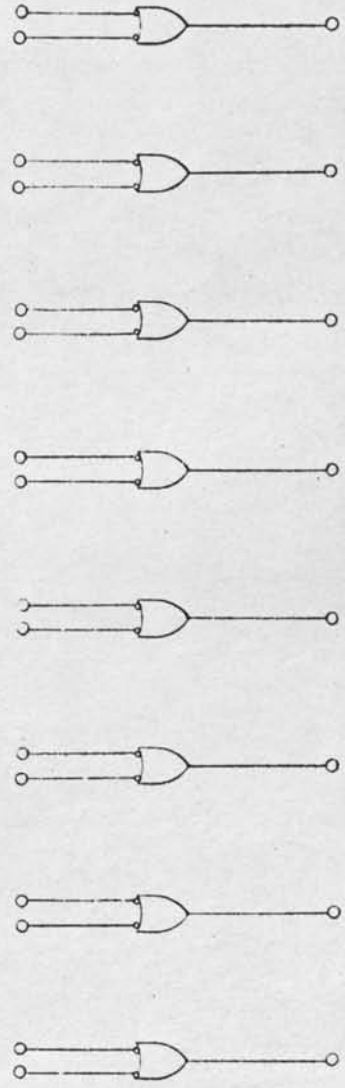


HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER CONTROL
STATE SWITCHES

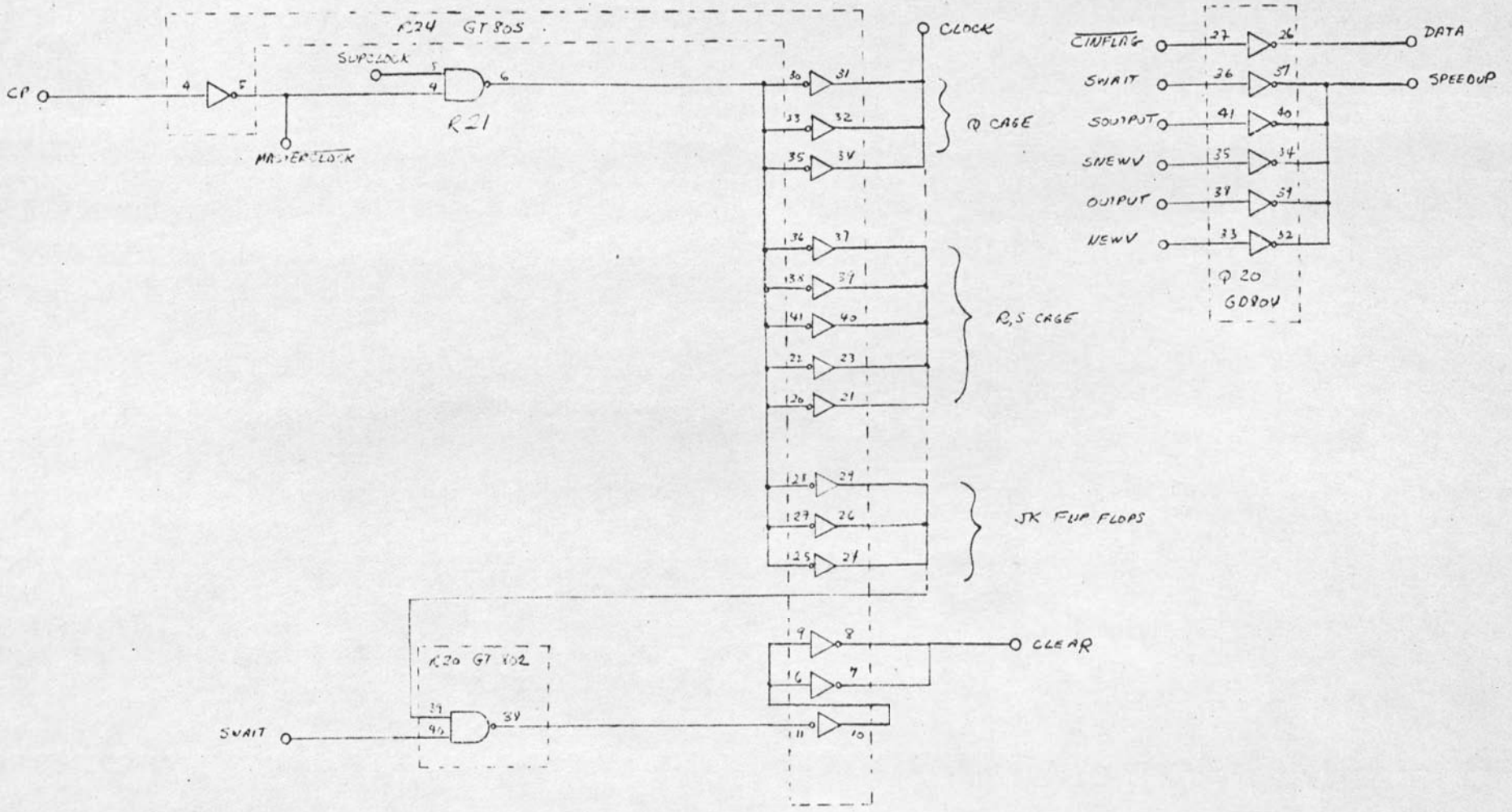
CC 37







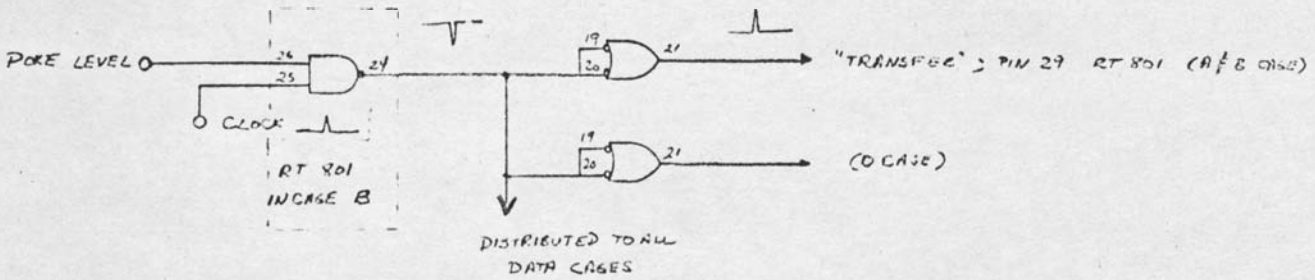
HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
EXTRA TESTS
CC 38



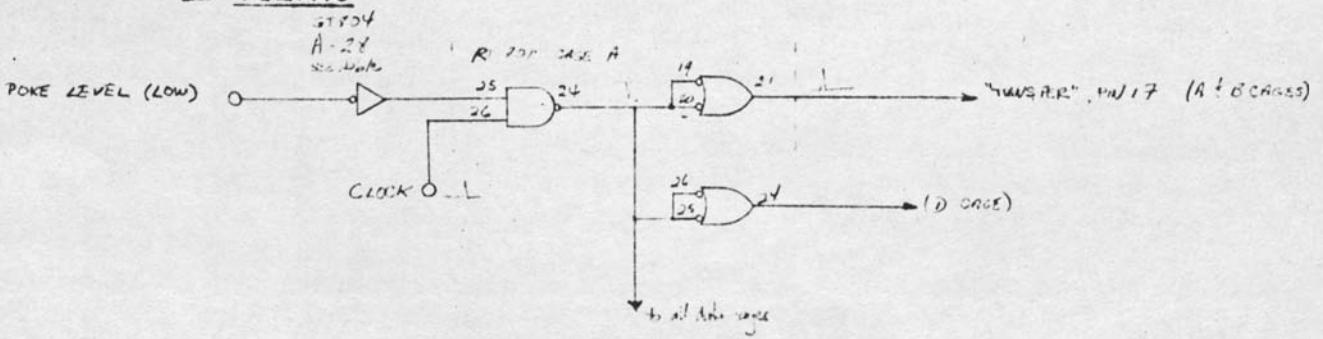
NOTE FOR WIRE LIST PURPOSES, ALL CLOCK DRIVERS ARE LISTED TOGETHER

HARVARD UNIVERSITY
3 D DISPLAY
CLIPPER CONTROL
MISCELLANEOUS
CC 39

I ACCUMULATORS



II DELTAS

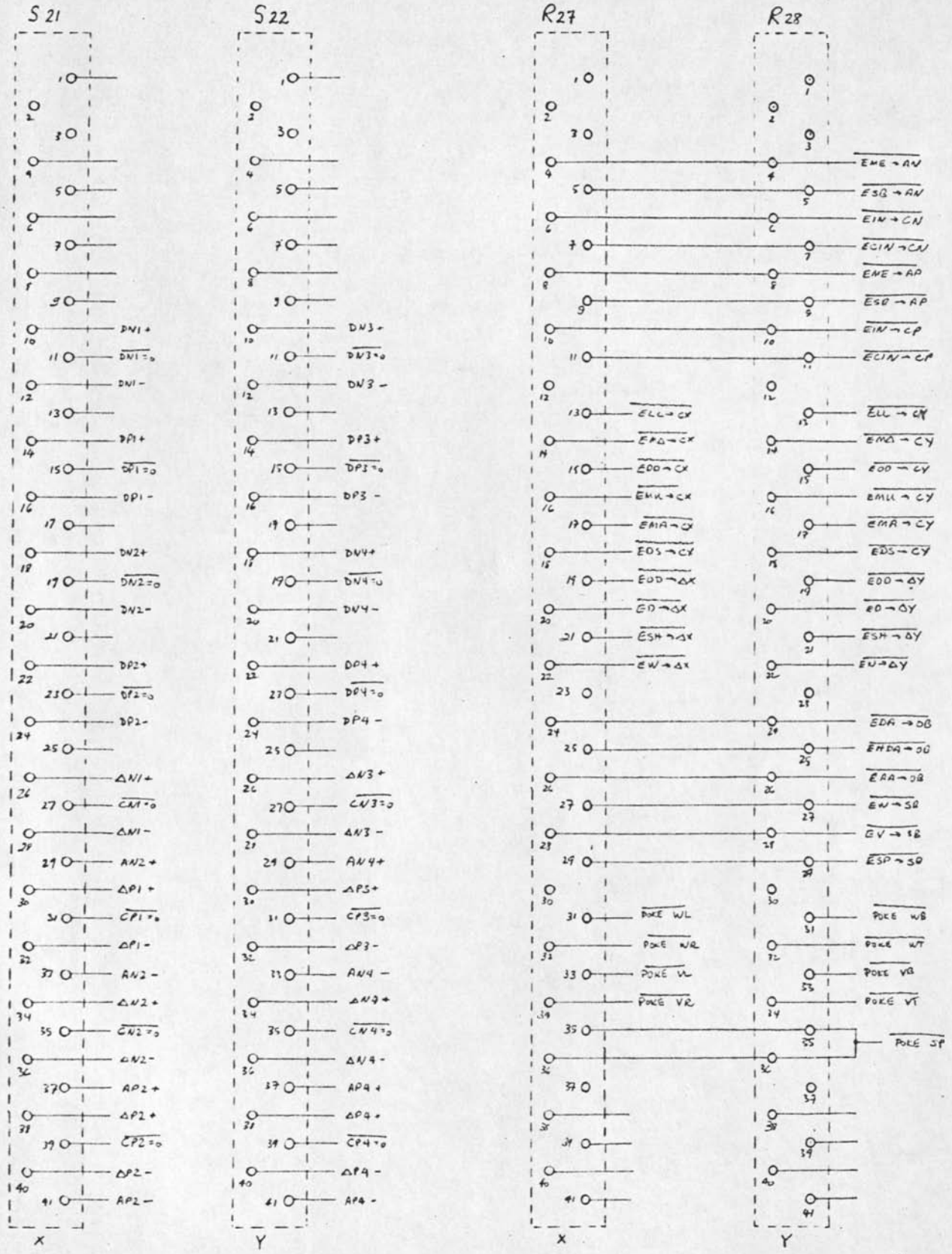


A 29

	in	out
LN1	4	5
LP1	6	7
LN2	9	8
LP2	11	10
LN3	12	13
LP3	14	15
LN4	17	16
LP4	19	18

HARVARD UNIVERSITY
3D DISPLAY
CLIPPER CONTROL
STROBE DETAILS

CC 40



INPUTS

OUTPUTS

HARVARD UNIVERSITY
3-D DISPLAY
CLIPPER CONTROL

CABLES
CC 41