UUCS-86-003
Rev. 1

PPL QUICK REFERENCE GUIDE (NMOS)

Steven R. Jacobs

Kent F. Smith

January 12, 1987

VLSI RESEARCH GROUP
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF UTAH
SALT LAKE CITY, UTAH 84112
(801) 581-8580

This work was supported in part by *Defense Research Projects Agency* under Contract Number DAAK1184K0017. All opinions, findings, conclusions or recommendations expressed in this document are those of the author(s) and do not necessarily reflect the view of DARPA.

Table of Contents

1. PPL Program Summary

Descriptions of valid ways to start PPL programs, and options available.

tiler (PPL editor)

tiler
tiler module (.ppl extension assumed)
tiler module.ext (explicit extension)

simpplex (PPL circuit extractor)

simpplex (prompts user for module name)
simpplex module (.ppl extension assumed)
simpplex module.ext (explicit extension)

options:

-t produce timing warnings for "slow" nodes (optional value specifies the minimum delay time that will produce a warning, example: -t25)

-l produce warnings for nodes with loads greater than specified value This is similar to -t option, but allows specifying minimum load value will that will produce a warning.

simppl (PPL logic simulator)

simppl (prompts user for module name)
simppl module (.ppl extension assumed)
simppl module.ext (explicit extension)

pplpr (PPL print formatter)

pplpr (uses stdin, stdout)
pplpr module
pplpr module.ext

options:

-d format for Display terminal (underline when possible, unless -u)
-h do not number rows and columns
-p Postscript device, font is scaled to fit module on one page
-n Narrow device (80 columns, default if no -p option)
-w Wide device (132 columns)
-u turn off all underlining, use characters to display breaks
-x unlimited width device (default for -p option)

ppl2cif (PPL CIF generator)

options:

-e extract named modules from library (must use -l also)
-f force generation of output even if errors are found
-O optimize output by removing unused connections between cells
-m module only, do not extract cells from library
-p Pseudo-cif output format
-s source file is a .sif pseudo-cif file, translate to real CIF
-w suppress warning messages for cells defined but not used
-C Consistency check of PPL library (must use -l also)
-l Specify PPL library name, example: -lNMOS
-9 keep user extension "9" lines in CIF, used to specify cell names

2. TILER Editor Command Summary

2.1. CONTROL commands

^A	move to first column on screen.
^B	move Backward (left) one or more columns.
^C	Suspend TILER. TILER may be re-entered without losing anything.
^D	Delete PPL cell (and breaks) at current cursor location.
^E	move to End of PPL (but remain on this screen).
^F	move Forward (right) one or more columns.
^G	Universal abort command.
^H	(backspace). Delete break information at the current cursor location.
^I	(Tab). Move to PPL grid location, given as two arguments.
^J	No op.
^K	Kill a window and surrounding breaks, place on the kill ring.
^L	Repaint the screen.
^N	move down to Next row.
^P	move up to Previous row.
^Q	Quote. Sneak past placement restrictions.
^R	Reverse search for a specified cell.
^S	Search for a specified cell.
^U	Universal argument.
^V	move down one or more screens.
^W	copy a Window onto the kill ring, without deleting it.
^X	command prefix for eXtended commands.
^Y	Yank a block of cells from the kill ring.
^Z	command prefix for META-CONTROL commands.
^[	(same as <esc>). Command prefix for META commands.

2.2. META commands

M-A	move to start of PPL (but remain on this screen).
M-C	incremental CIF check. Move to location of 'next' CIF error.
M-D	Set default Direction.
M-F	Fill window with cells. TILER prompts for the cell character.
M-L	Set screen Length.
M-M	Modify cell. Set the modifier value of the cell to the given argument.
M-O	Move to window. Same as M-[
M-Q	Toggle QUOTE mode. Always sneak past placement restrictions.
M-R	Toggle RIGID mode.
M-T	set Terminal display mode. Argument affects display of break characters.
M-U	Toggle USER mode.
M-V	move up one or more screens.
M-W	Set screen Width.
M-Y	Yank again, rotating through the kill ring.
M-Z	Toggle ZOOM mode. (Not yet implemented)

M-< move to top edge of PPL array, remaining in the same column.
M-> move to bottom edge of PPL array, remaining in the same column.
M-[Move to WINDOW (lower left corner).
M-] Move to WINDOW (upper right corner).
M-{ Set lower-left corner of WINDOW. Argument between 0 and 9 (Default 1).
M-} Set upper-right corner of WINDOW. Argument between 0 and 9 (Default 1).
M-~ Tell TILER to pretend that the current file has not been modified.
M-;,
M-: Toggle COORDINATE ECHO mode. Turns on/off the Rxx:Cxx in the status line.
M-/,
M-? Describe a TILER command or PPL cell. You type the command characters.

2.3. META-CONTROL commands

(use ^Z prefix, or <esc>-<control character>)

^Z-B move Backward (left) one or more screens. May be preceded by an argument.
^Z-C Suspend TILER. TILER may be re-entered without losing anything.
^Z-D Delete PPL cell at current cursor location, but keep surrounding breaks.
^Z-F move Forward (right) one or more screens. May be preceded by an argument.
^Z-G (aborted command, rings bell only)
^Z-I (META Tab). Set MARK. Argument between 0 and 9. (Default 1)
^Z-K Delete (cells from) WINDOW, but keep surrounding breaks. (Arg 0 to 9).
^Z-L Adjust screen, put cell under the cursor at upper-left corner of screen.
^Z-N move down one or more screens. May be preceded by an argument.
^Z-O move to origin of PPL array, changing screens if necessary.
^Z-P move up one or more screens. May be preceded by an argument.
^Z-W Write WINDOW (to PPL file). Argument between 0 and 9 (default 1).
^Z-Z Suspend TILER. TILER may be re-entered without losing anything.
<esc><esc>,
M-<esc>,
M-^[,
^Z-[(<esc><esc>) Escape to/from USER mode. (Prefix to switch keyboards)
^Z-_,
^Z-/
^Z-? Display the last error message encountered.

2.4. eXtended commands

^X-A move to Start of PPL. Moves to the first PPL cell or break on the row.
^X-B select Buffer. Switch to buffer with the specified name.
^X-C CIF checks. Creates .ERR file listing cell placement errors.
^X-D Define key. Assigns the current keyboard macro to the specified key.
^X-E Execute keyboard macro. May be preceded by an argument (repeat count).
^X-F list File. Displays contents of a text file on the screen.

^X-H Mark whole buffer. Selects the entire PPL array as the default WINDOW.
^X-I Insert PPL file. TILER prompts for the file name.
^X-K Describe User Key. TILER prompts for the key, shows keystroke definition.
^X-L show Location in PPL buffer. Useful when coordinate echo is off.
^X-M Modify window. Sets the modifier value of cells in the current window.
^X-O Move to PPL Origin. Place cursor at row 0, column 0.
^X-T new Technology. Start a new edit session with a different cell set.
^X-U Undefine key. Removes a key definition from the specified key.
^X-X eXecute a TILER command file (usually created by logging with ^X^L).
^X-= show Location in PPL buffer. Useful when coordinate echo is off.
^X-(Begin defining keyboard macro.
^X-) End of keyboard macro.
^X-[Move to origin (lower-left corner) of the PPL array.
^X-] Move to the upper-right corner of the PPL array.
^X-{ Set lower-left corner of PPL array. CAUTION: Reduces size of PPL array.
^X-} Set upper-right corner of PPL array. CAUTION: Reduces size of PPL array.
^X-/,
^X-? HELP!
^X^A move Backwards to the left edge of the PPL array.
^X^B list Buffers. Lists names of all buffers and associated files.
^X^C Exit TILER. Offers to save the file if changes have been made.
^X^E move to End of PPL. Move to the extreme right edge of the PPL array.
^X^F Find file. Edit a new file in its own buffer (or return to buffer).
^X^G (aborted command, rings bell only)
^X^I Insert PPL rows or columns (Argument specifies how many).
^X^K Delete PPL rows or columns (Argument specifies how many).
^X^L Log TILER commands. Start, pause, resume, or stop logging TILER commands.
^X^O move to PPL origin (row 0, column 0).
^X^R Read PPL file. Edit a new file, without changing to a new buffer.
^X^S Save changes in the current file.
^X^V Visit file. Same as ^X^F. (Visit PPL file).
^X^W Write file. TILER prompts for the file name.
^X^X Exchange POINT and MARK. Argument between 0 and 9 (Default 1).
^X^Z Suspend TILER. TILER may be re-entered without losing anything.

3. SIMPPL Simulator Command and Option Summary

3.1. Summary of SIMPPL commands

BREAK	Pause in execution of a command file.
CLOCK	Specify clock sequences associated with input nodes.
CONTINUE	Resume execution of a command file (after a BREAK).
COPY	Copy simulation output to a file.
CYCLE	Continue circuit simulation for a specified number of clock cycles.
EXIT	Exit the simulator program.
FORCE	Specify nodes to be forced to present (or specified) values.
HELP	Print helpful information about commands and options.
INIT	Initialize all circuit nodes to same state (0, 1, or X).
MAP	Find the node number of a specified PPL grid location.
OPTIONS	Set simulator options.
NAME	Name a circuit node (specified by PPL grid location).
NODE	Name a circuit node (specified by node number).
PAD	Name a pad cell (specified by PPL grid location).
PHASE	Continue circuit simulation for a specified number of clock phases.
PROBE	Name internal nodes of a cell (specified by cell PPL grid location).
PROMPT	Change the prompt issued by the simulator when ready for commands.
QUIT	Exit one level of simulation (end of command file, or exit).
RESTORE	Restore the circuit state from a file.
SAVE	Save the present state of the circuit in a file.
SET	Set the state value of input nodes and/or forced nodes.
SHOW	Show the current state of the specified nodes or the WATCH list.
SOURCE	Take simulator commands from a specified command file.
STATUS	Show state of a node, and interaction with other nodes.
STEP	Resume simulation for a specified number of unit steps.
TRACE	Add the specified nodes to the TRACE list.
TRAP	Add the specified nodes to the TRAP list.
UNCLOCK	Remove the specified nodes from the CLOCK list.
UNFORCE	Remove FORCE status from the specified nodes.
UNTRACE	Remove the specified nodes from the TRACE list.
UNTRAP	Remove the specified nodes from the TRAP list.
UNWATCH	Remove the specified nodes from the WATCH list.
VECTOR	Group several nodes together with a single name.
WATCH	Add the specified nodes to the WATCH list.
WIRES	Show the wire names used in the current cell set.
?	Give a list of available commands.

3.2. Summary of SIMPPL options

ASYNCH	Use the asynchronous model for simulation.
ECHO	Turn on/off echo of commands executed from a command file.
SHOWINPUTS	Show the initial state of the WATCH list when resuming simulation.
SYNCH	Use the synchronous model for simulation.
TRACE	Enable/disable tracing of nodes on the TRACE list.
TRAP	Enable/disable trapping of nodes on the TRAP list.
VERBOSE	Control amount of simulation data that is printed.

4. Special Rules: Placement and Sizing of NMOS Row Pullups

4.1. Introduction

The current NMOS PPL cell sets use the polysilicon layer for row wires. The relatively high resistance of polysilicon can lead to DC circuit problems if care is not taken in the placement of pullups which connect to row wires. An example of a problem circuit is shown below

0 1 1 0 0 ...(with a lot of blank columns in between)... + s r u

In this row of PPL, the '0' and '1' cells are pulldowns, the '+', 's', and 'r' cells are gates which detect the voltage on the row wire, and the 'u' cell is a row wire pullup which pulls the row high whenever all of the '0' and '1' pulldown transistors are off. When one or more of the pulldown transistors are on, the voltage at the left end of the row will be pulled near zero volts by the pulldown. However, because of the resistance of the polysilicon row wire, the voltage at the right end of the row could remain significantly above the turn-on voltage of the '+', 's', and 'r' transistors. This would prevent the '+', 's', and 'r' transistors from turning off properly, so that the circuit would fail in its intended operation.

Note that this circuit can be constructed properly by placing the 'u' cell near the '0' and '1' cells, as shown below.

0 1 1 0 0 u ...(with a lot of blank columns in between)... + s r

In this circuit, the voltage near the 'u' pullup cell can always be pulled near zero volts, and the voltage at the pullup will propagate down the row wire to the '+', 's', and 'r' cells. Although the resistance of the poly row wire is still high, there are no longer any DC currents flowing through the long poly wire. Thus after the capacitance of the row wire has been charged, no current will flow in the section of the row to the right of the 'u' cell, and that entire section of the row will be at the same voltage as the 'u' cell.

The proper placement of pullups becomes more complicated when pulldowns are mixed with sensing cells ('+', 's', 'r', etc.) in the same section of a row wire. Two sets of rules are outlined below for the proper placement of pullups on row wires. The "Quick and

Dirty method" given in section A is intended as a general rule of thumb that will always work, but is more restrictive than necessary in some cases. A more complicated, but less restrictive set of rules is given in section B.

The rules outlined in sections A and B below only apply to PPL circuits which occupy a single section of a single row. By using ohmic contacts, it is possible to have a circuit that is made up of several sections on several different rows. Such circuits complicate the analysis of the problem, and are not covered by the rules given below.

4.2. Simplified Placement Rules

The maximum distance between a row pullup and the "most distant" pulldown which will draw current from the pullup must be less than the value given in the table below. This value will be referred to as Dmax. In general, the safest place to put a row pullup is near the midpoint between the two "outer" pulldowns on the row segment. This will minimize the distance from the pullup and the most distant pulldown. If this is not possible, it may be necessary to use a "weak" pullup in order to increase the allowed Dmax value. Care should be taken to keep track of pulldown cells other than the '0' and '1' cell. Any cell which can affect the logic value of the row segment is a pulldown cell. Examples of other pulldown cells in the NMOS cell set are: 'X', 'E', 'w', 'j', 'y', 't', 'a', 'o', 'R', 'S', '6', '9', 'U', and 'D'. Ohmic contact which connect column wires onto a row wire can cause other cells to act as pulldowns on the row wire.

	Dmax (columns)		
pullup size	Safe	Risky	Unsafe
(standard)	12	20	30
1/2 strength	30	45	65
1/4 strength	65	100	130

In most cases, you are strongly advised to keep the distance between any row pullup and its most distant pulldown BELOW the "safe" value given in the table. This means that a row pullup may be safely placed in the CENTER of a string of pulldowns which is twice as wide as Dmax.

Distances which are only a few columns above the "safe" value should be avoided, but will very probably work with no problem. Distances near or above the "risky" value should be avoided at all cost. Distances near or above the "unsafe" value will be very likely to cause circuit failure for some circuits which are within the process parameters allowed by MOSIS.

4.3. Advanced Rules for Pullup Placement

The rules given in section A are meant as simple guidelines which will work in most cases. The rules of section A do not depend on the location of the sensing cells on the row. When the placement of the sensing cells is taken into account, less restrictions apply to the placement of pullups.

The proper operation of a circuit depends only on the ability of the pulldowns/pullups to correctly control the voltages AT THE SENSORS. Points in the circuit that are not connected to sensors may have wildly varying voltages on them without adversely affecting the operation of the circuit. The rules in section A are designed to ensure that all pulldowns on a row segment can control the voltages at all points on the segment. Rules in this section are designed to control the voltages only at the locations of the sensors. Points on a row segment that are not connected to sensors are allowed to remain at a high voltage even when some pulldowns on the row segment are on. This allows greater flexibility in placing pullups on row wires.

Three distinct cases may be considered separately to illustrate the rules for placement of pullups. Each case will be illustrated with an example and an explanation of the allowed range of placement positions for the pullup. The examples will show a line of PPL with the allowed pullup placement.

CASE 1: One pulldown, all sensors within Dmax of the pulldown.

```
Place pullup: <--------- (anywhere) ------------>
                    r s    + u 1    + r s
                  |<-- Dmax -->|<-- Dmax -->|
```

If all sensors are "close" to a single pulldown, then the pullup may be placed at any convenient location. The optimal location for the pullup is next to the pulldown. This

location will ensure that all locations on the row segment can be pulled to a low voltage by the pulldown.

If the pullup is placed a large distance from the pulldown, the pulldown will still be capable of making the voltage at all "nearby" locations low enough to turn off the sensing transistors in that area. The voltage near a "distant" pullup may remain high even when the pulldown is turned on, but this causes no harm since there are no sensing transistors in the areas where the voltage is too high to turn off a sensing transistor.

This case demonstrates the fact that sensors which are close to a single pullup cause no restrictions on the allowable location of the pullup. It is only when sensors are placed "far" from a pulldown that pullup placement becomes restricted. In the remaining examples, we will only need to worry about sensors that are placed "far" from a pulldown.

CASE 2: One pulldown, all "distant" sensors on one side of the pulldown

```
Place pullup:    <-- (any) -->|<-- Dmax -->|
                              u ↑                + r s          + r s
        (B)                  (A)      (D)                    (C)
```

The example above shows a PPL row which has row sensors which are all to the right of the pulldown. The voltage level at the sensors determines whether or not the circuit will operate correctly. The interaction of the pullup, the pulldown, and the resistance of the polysilicon between the pullup and pulldown determines what the voltage at the sensors will be.

First suppose that the pullup is located at its optimal position at (A). When the pulldown is turned on, the entire row segment will be pulled to a voltage near zero volts, turning off all of the sensors on the row. This is the normal mode of operation of the circuit.

Now suppose that the pullup is placed at (B). When the pulldown is turned on, current will flow from the pullup to the pulldown, passing through the polysilicon wire which connects the two transistors together. The voltage at (A) will be near zero volts, and the voltage at all points to the right of (A) will be at the same voltage as at (A). Moving to the left of (A), the voltage will increase slightly due to the current from the pullup flowing

through the resistance of the polysilicon wire. The voltage at the pullup will be the highest voltage on the row wire, and this same voltage will appear at all points to the left of the pullup. If the distance from (A) to (B) is small, this voltage will also be close to zero volts. But if the distance from (A) to (B) is large, the voltage at the pullup could be much greater than zero volts, and might even be high enough to turn on a transistor at that location. However, since there are no sensors on this part of the row wire, it does not matter how high the voltage is near the pullup. Thus when the pulldown at (A) is on, the voltage at all of the sensors is guaranteed to be near zero volts as long as the pullup is placed to the left of (A).

Now consider what happens then the pullup is placed at (C). In this case, the pulldown will pull the voltage near (A) close to zero volts, and the voltage on the row segment will increase between (A) and (C), reaching a maximum at (C). All points to the right of (C) will have the same voltage that appears at (C). If the distance from (A) to (C) is greater than Dmax, this maximum voltage will be too high to allow the sensors to the left side of (C) to turn off, causing a failure in the circuit. Also, any sensors to the right of (C) which are not closer to (A) than Dmax, will be left on. Thus the pullup at (C) is too far from the pulldown at (A). If the pullup is placed at (D) instead of at (C), then the pulldown will always be able to force the voltage at (D) to a low enough voltage to turn off all of the sensors on the row.

Pullup Law:

> If a row pullup is placed between a pulldown and a "distant" sensor, the pullup must be placed within Dmax columns of the pulldown. A "distant" sensor is any sensor that is further than Dmax columns away from the pulldown under consideration. The value of Dmax is determined by the strength of the pullup, and is given in the table in section A.

4.4. Multiple Pulldowns

Most useful circuits will have more that one pulldown on most of the row segments in the circuit. The proper placement of a pullup on a row segment which contains several pulldowns can be determined by considering each pulldown separately, and "combining" the allowed placement regions of the separate pulldowns into a single placement region for the group. The allowed placement region for a group of pulldowns, if it exists, is formed by taking the intersection of the allowed placement regions of all of the pulldowns in the group.

In many cases, the allowed placement region for a group of pulldowns is equal to the intersection of the placement regions of the TWO OUTERMOST PULLDOWNS in the group. Examples shown below demonstrate the method for finding the pullup placement region for several cases.

Example 1

```
                  u 1  0   1 1  0 0   + r $  +   +  +
<-- (any) ---->|<-- Dmax -->|                    (region for Left pulldown)
<-------------------- (any) ---->|<-- Dmax -->|  (region for Right pulldown)
<-- (any) ---->|<-- Dmax -->|                    (combined region)
```

Example 2

```
 + + r   $     u 1 0 1 1 0 1  + r $  +   +  +
    |<-- Dmax -->|<-- Dmax -->|                  (region for Left pulldown)
                 |<-- Dmax -->|<-- Dmax -->|    (region for Right pulldown)
                 |<-- allowed -->|               (combined region)
```

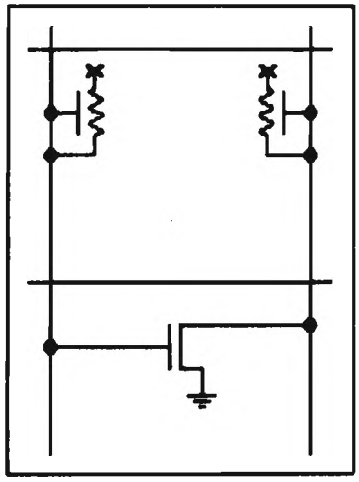
Example 3

```
               1 +      u     + 1
|<-- (any) -->|<-- Dmax -->|                    (region for Left pulldown)
                    |<-- Dmax -->|<-- (any) -->|  (region for Right pulldown)
                    |<- ok ->|                   (combined region)
```

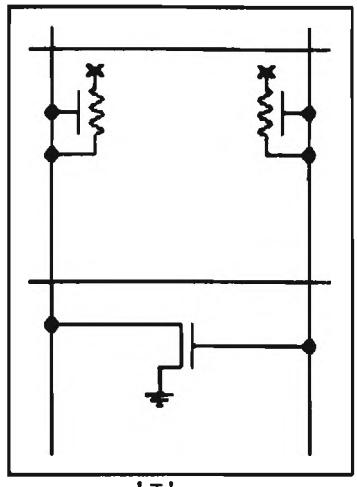
Example 4

```
               1 0 1 1 +       s   1 0     + 1
|<-- (any) -->|<-- Dmax -->|                    (region for Left pulldown)
  (region for Right pulldown)  |<-- Dmax -->|<-- (any) -->|
       (regions do not intersect, impossible to place pullup properly)
```

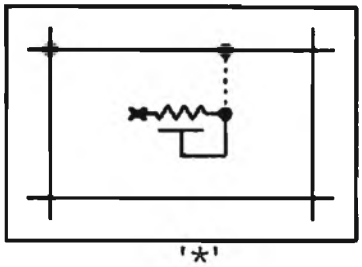
In Example 4 above, there is not a proper place to put a pullup. In this case, it would be necessary with to change the circuit so that it is not spread out so for, or try to use a weaker pullup in order to increase the size of the Dmax regions so that they overlap.



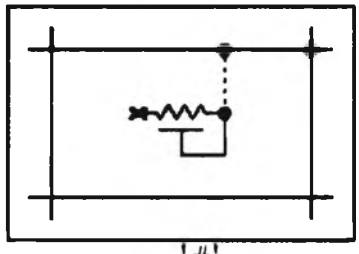
'V'



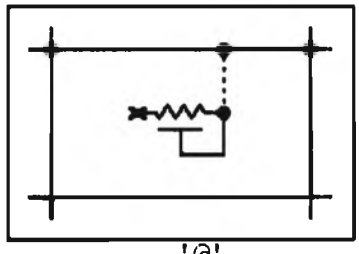
'I'



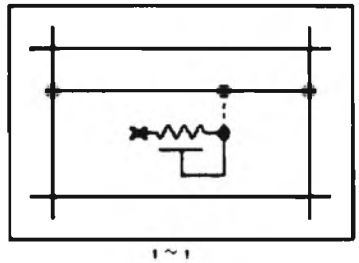
'*'



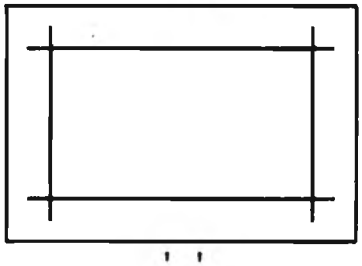
'#'



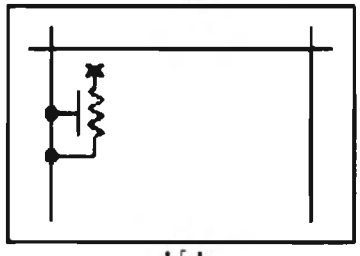
'@'



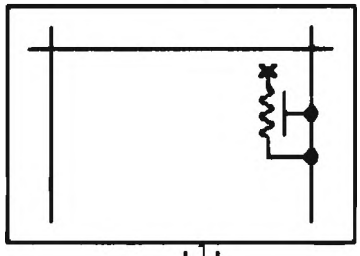
'~'



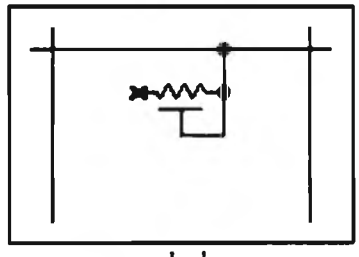
' '



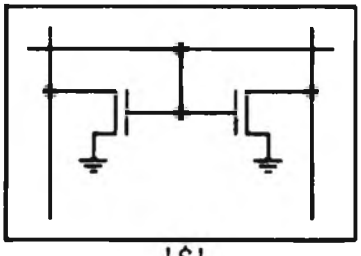
'['



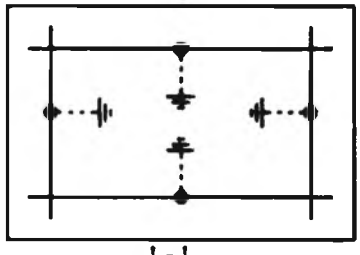
']'



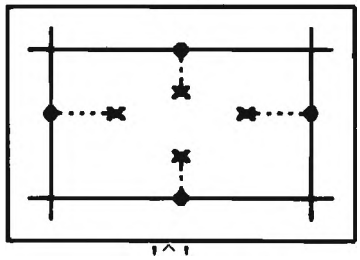
'u'



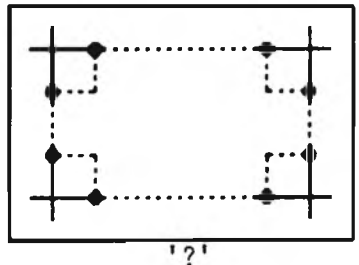
'$'



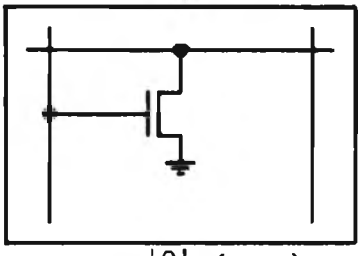
'g'



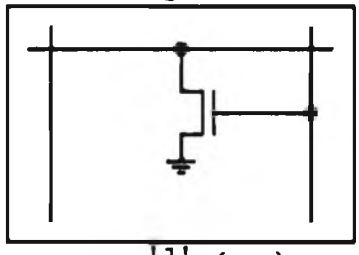
'^'



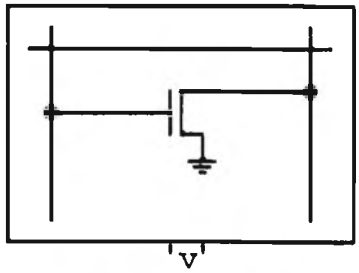
'?'



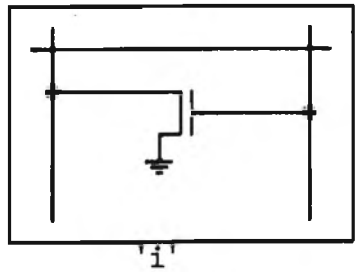
'0' (zero)



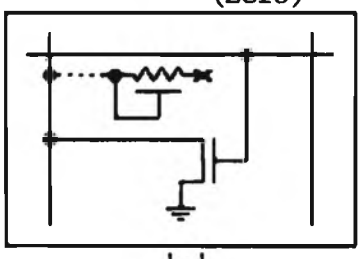
'1' (one)



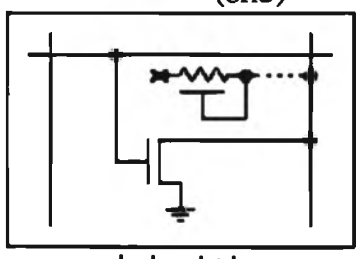
'v'



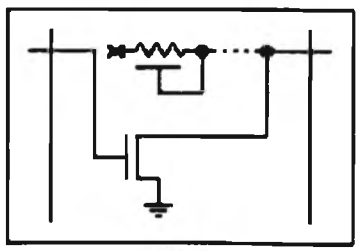
'i'



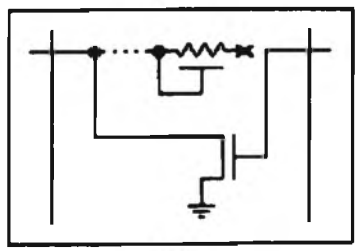
's'



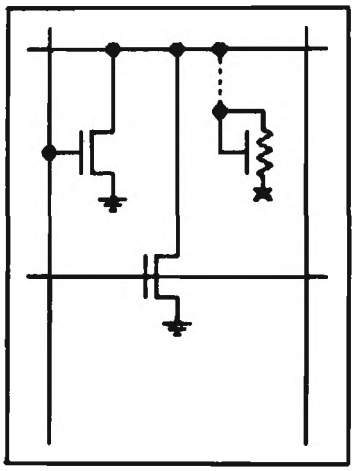
'r' '+'



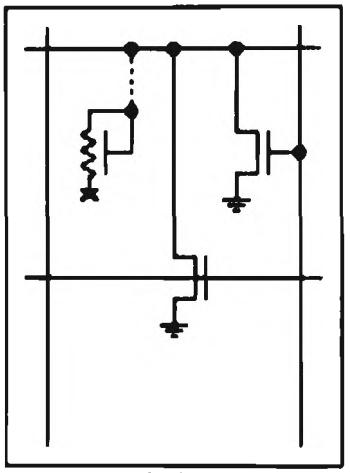
'w'



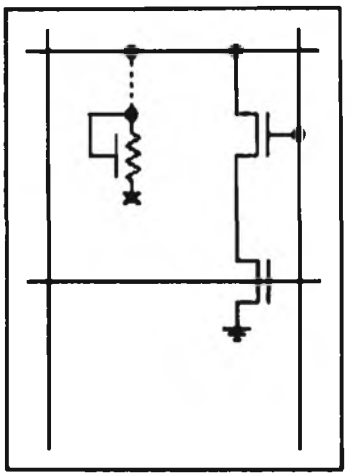
'j'



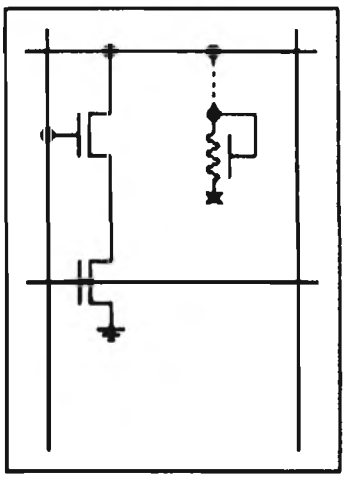
'y'



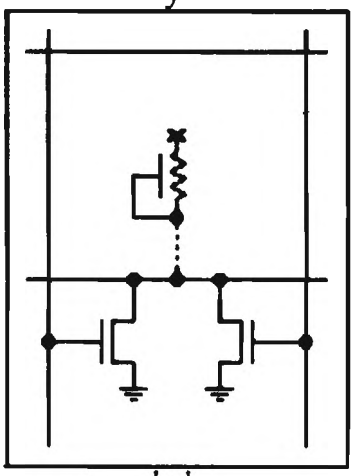
't'



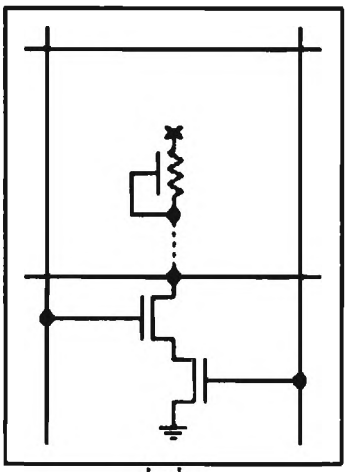
'l' (ell)



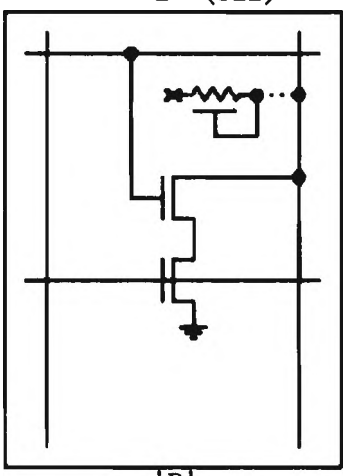
'0' (oh)



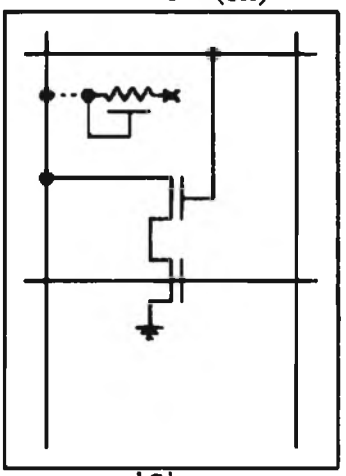
'a'



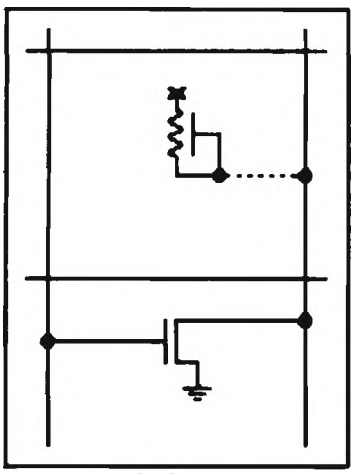
'o'



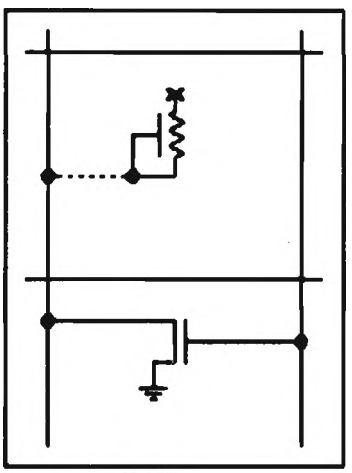
'R'



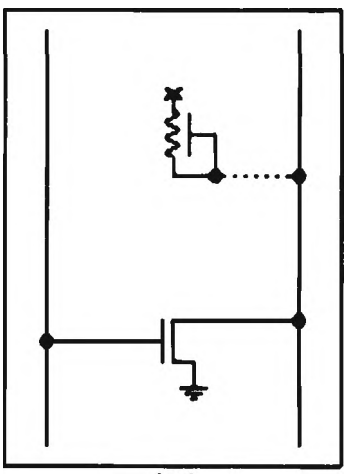
'S'



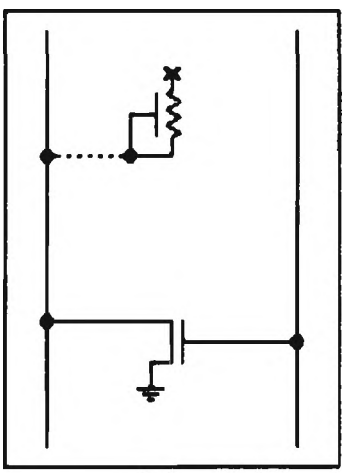
'V'



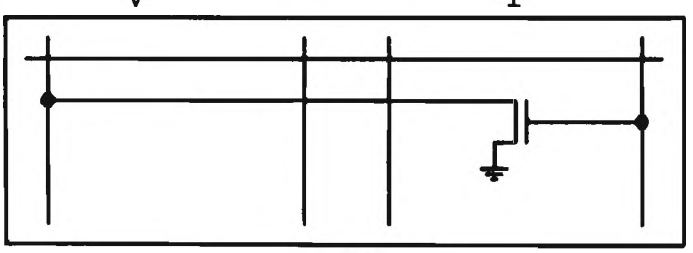
'I'



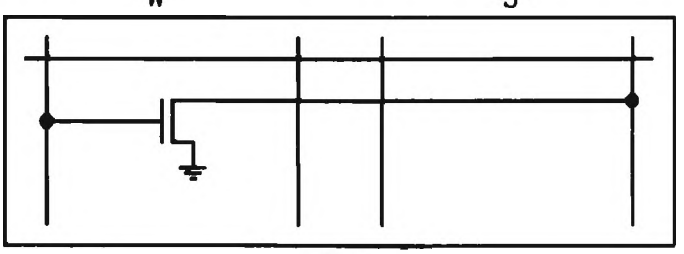
'W'



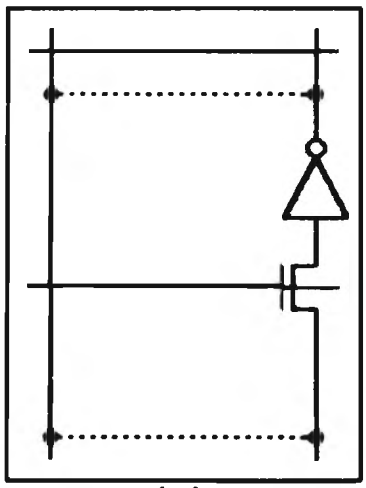
'J'



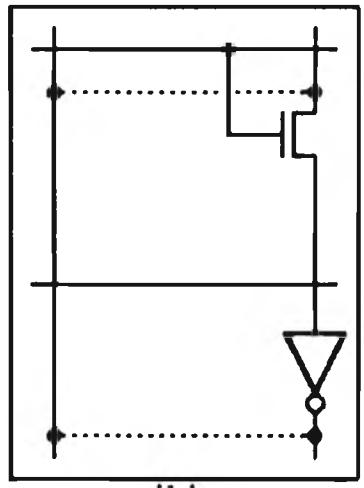
'k'



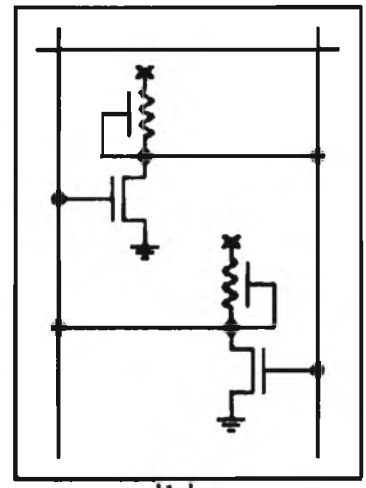
'n'



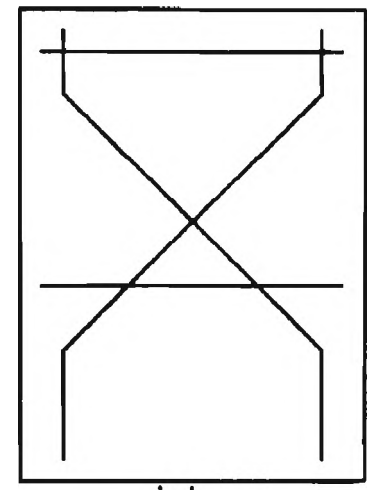
'p'



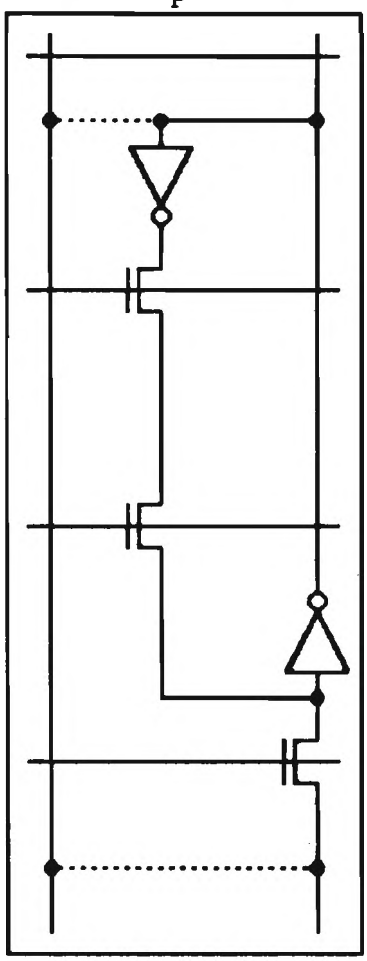
'b'



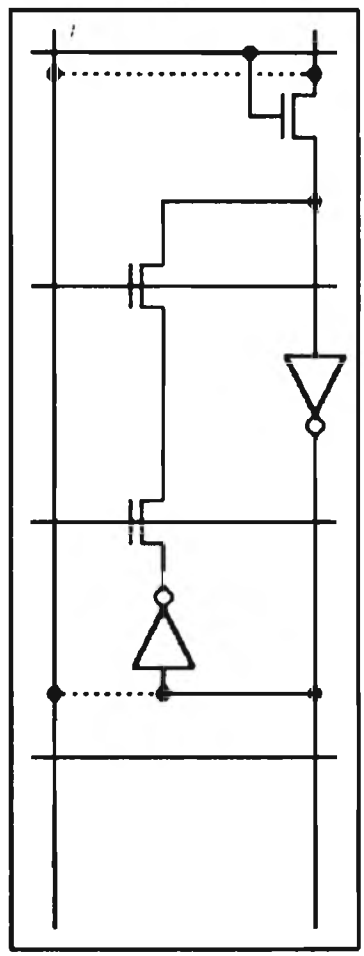
'h'



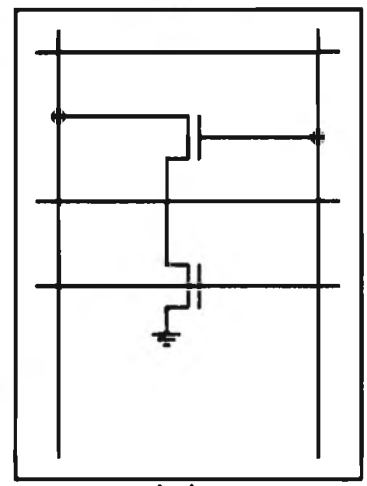
'x'



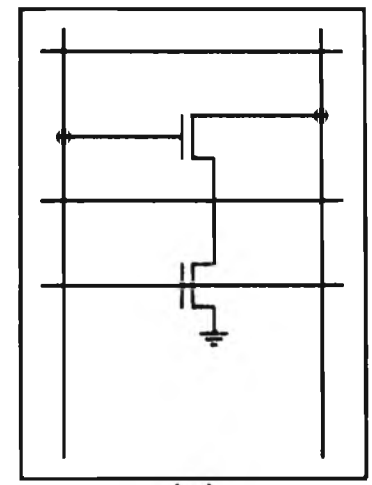
'P'



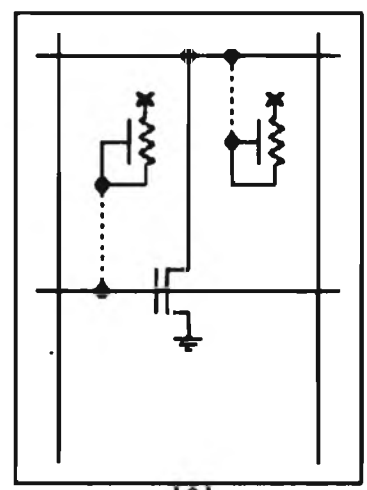
'B'



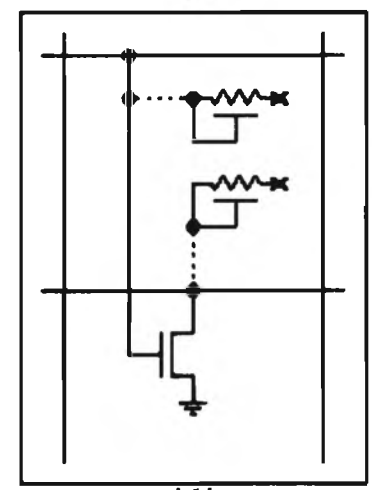
'<'



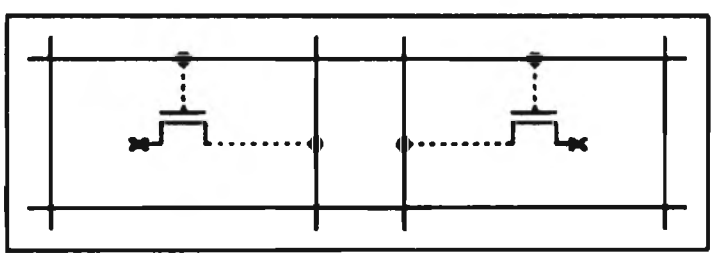
'>'



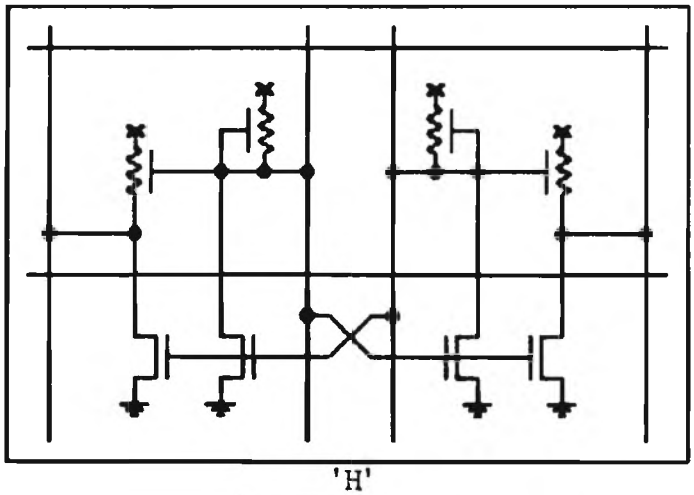
'9'



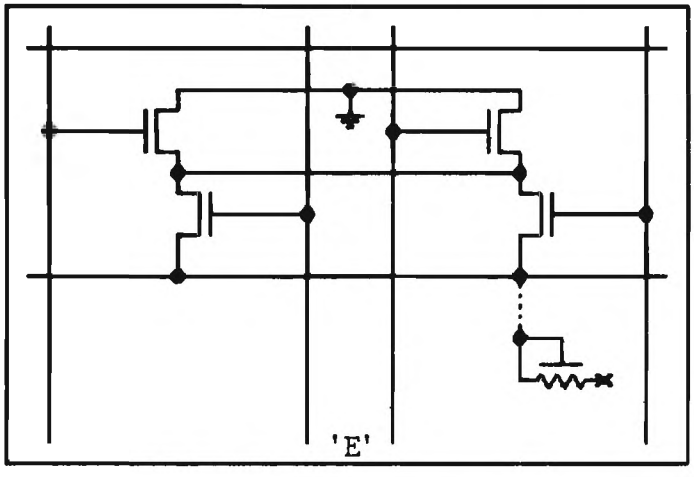
'6'



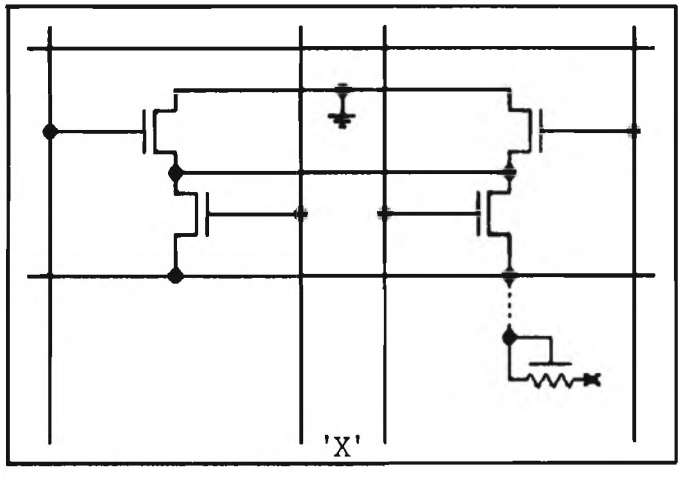
'm'



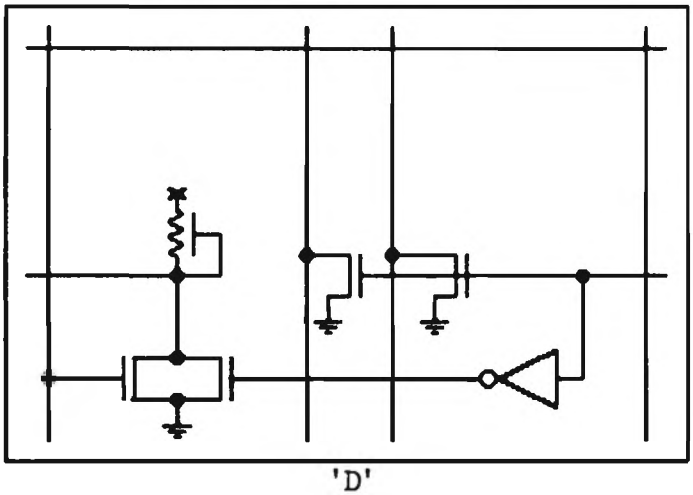
'H'



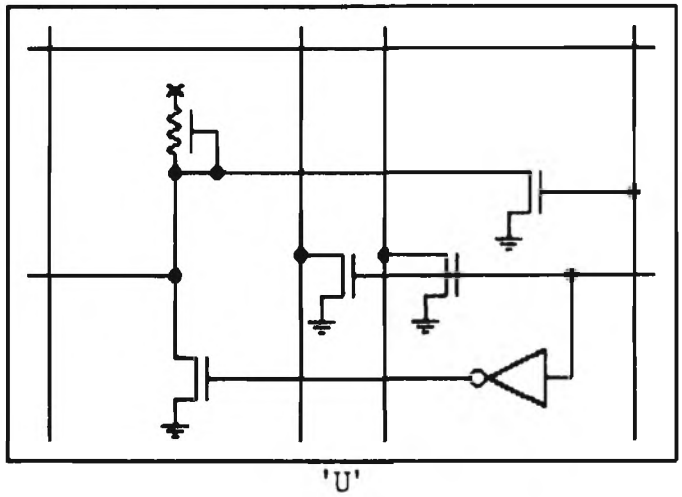
'E'



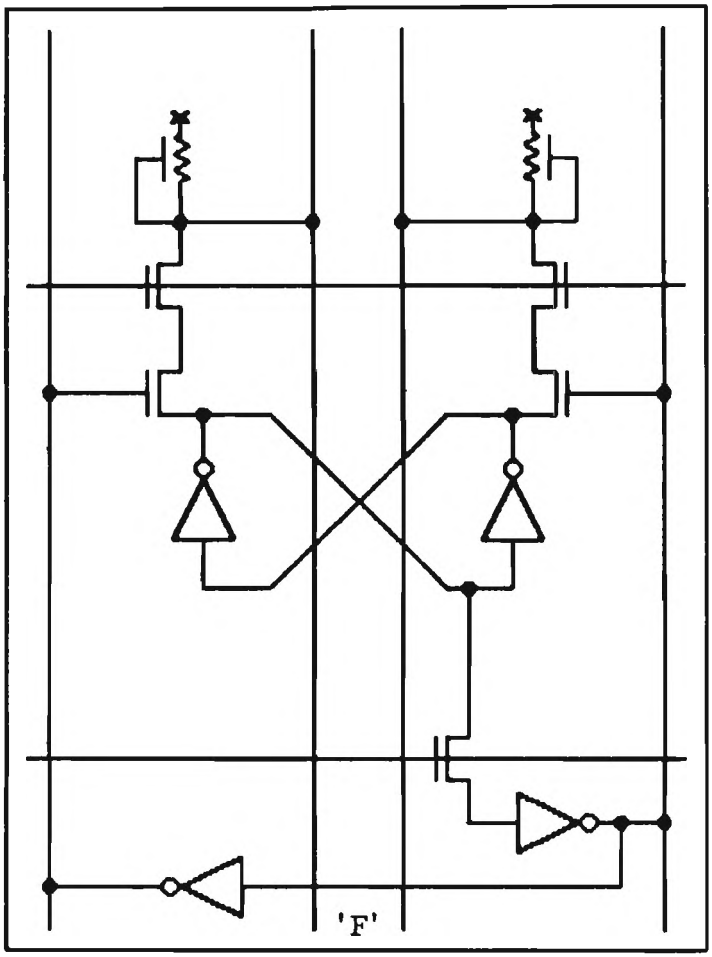
'X'

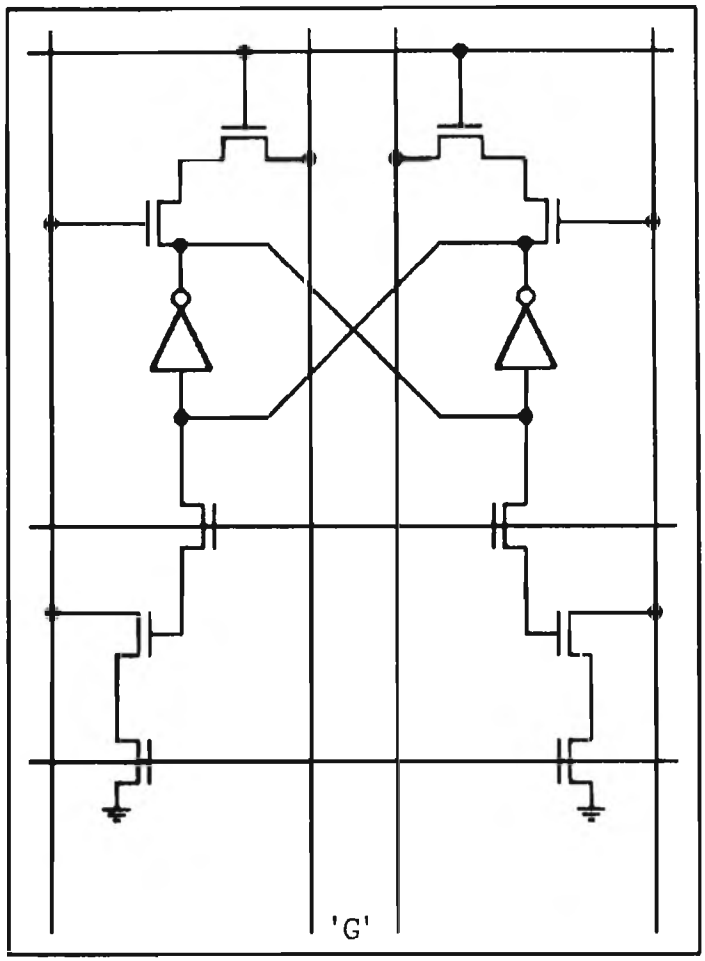


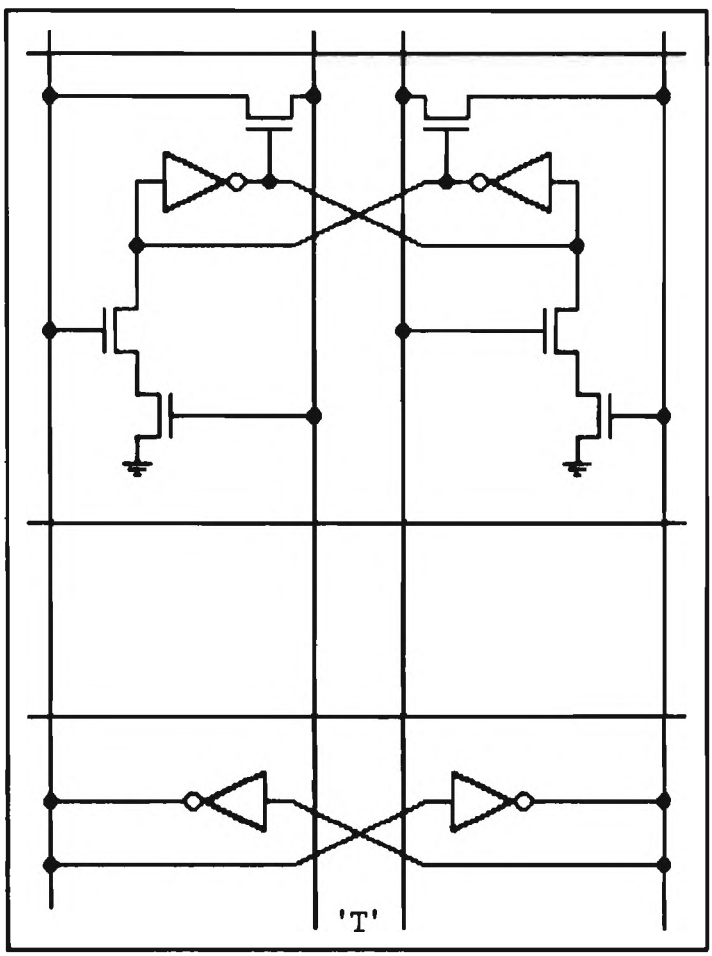
'D'

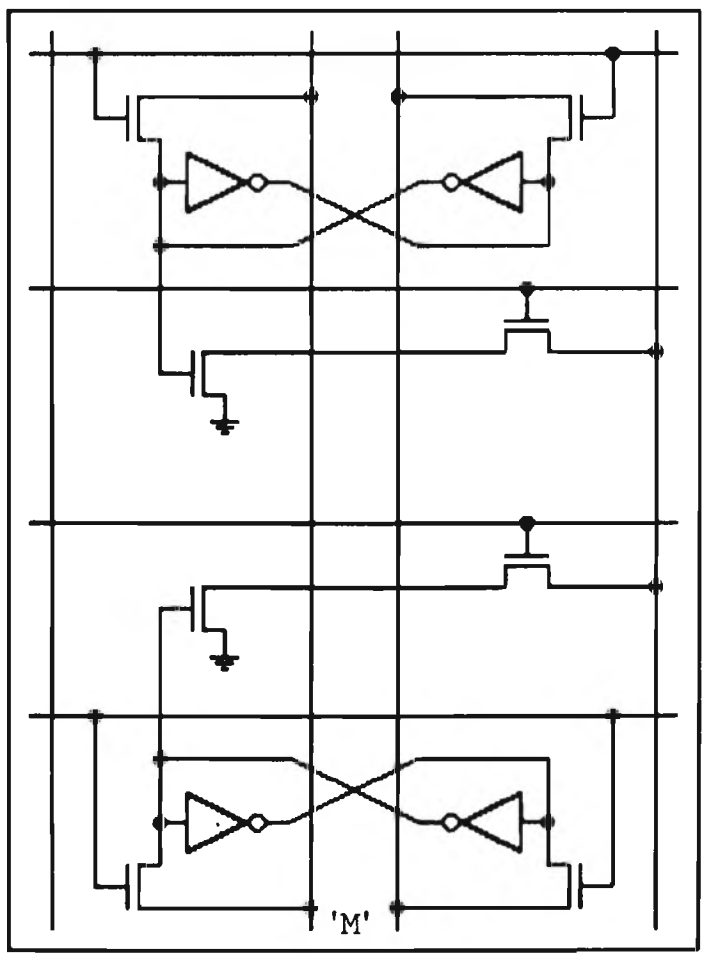


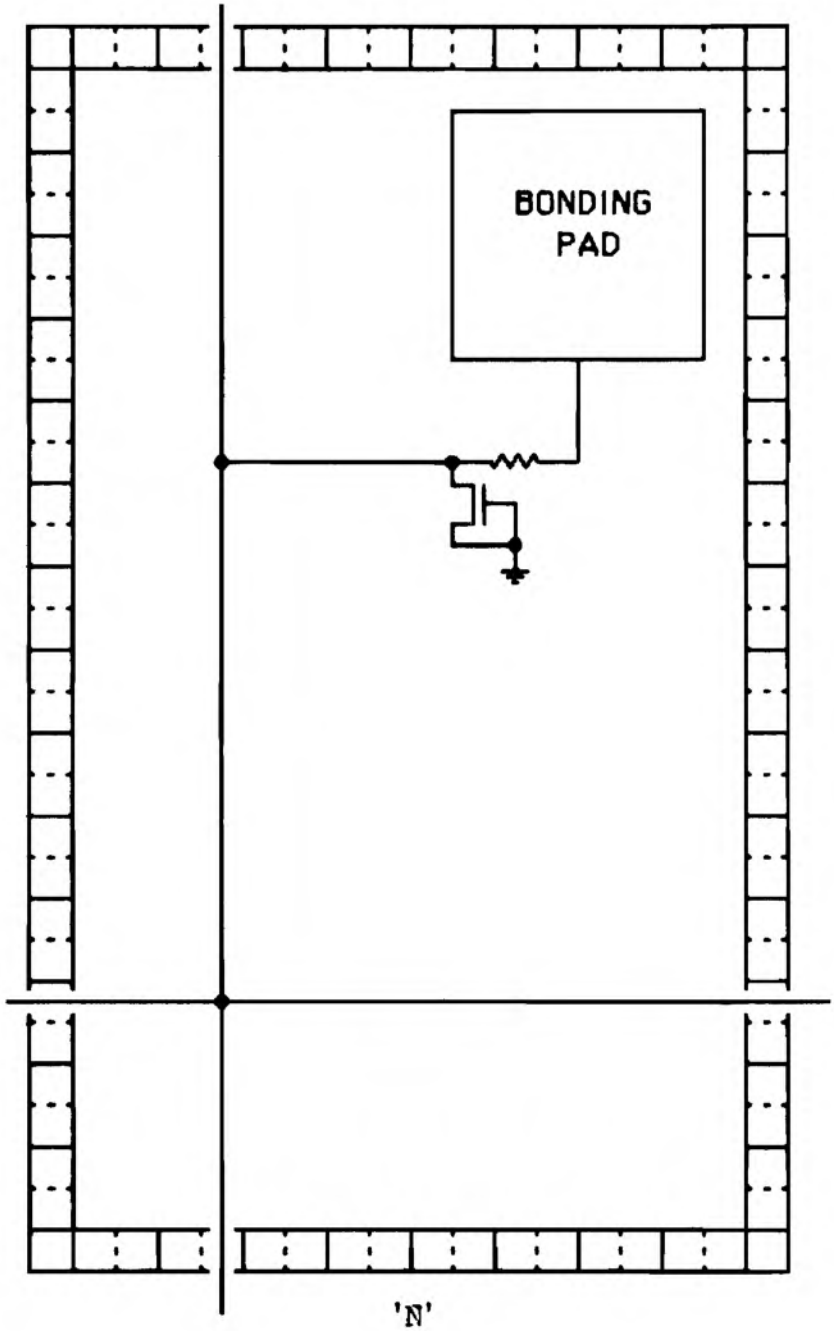
'U'

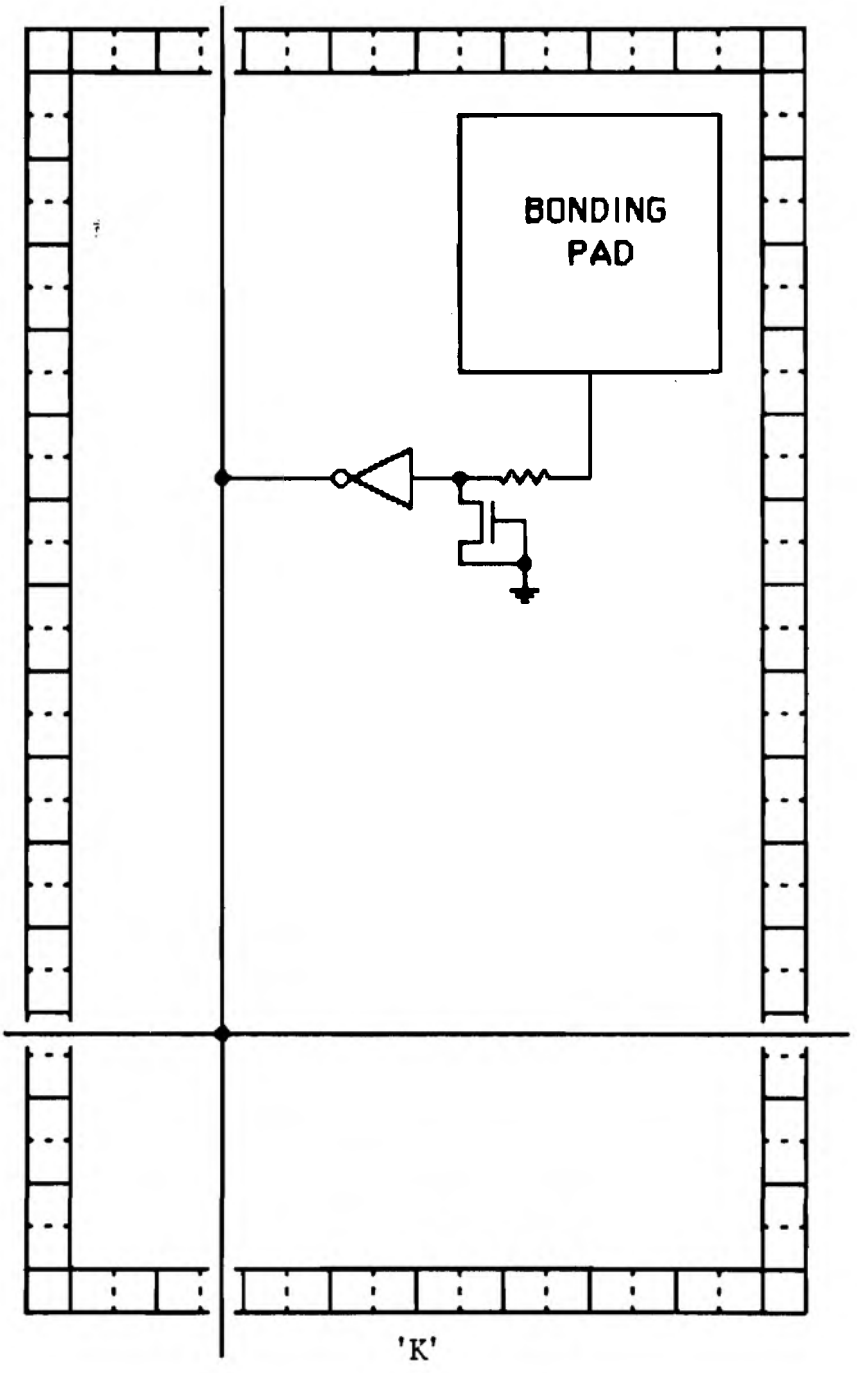

'F'

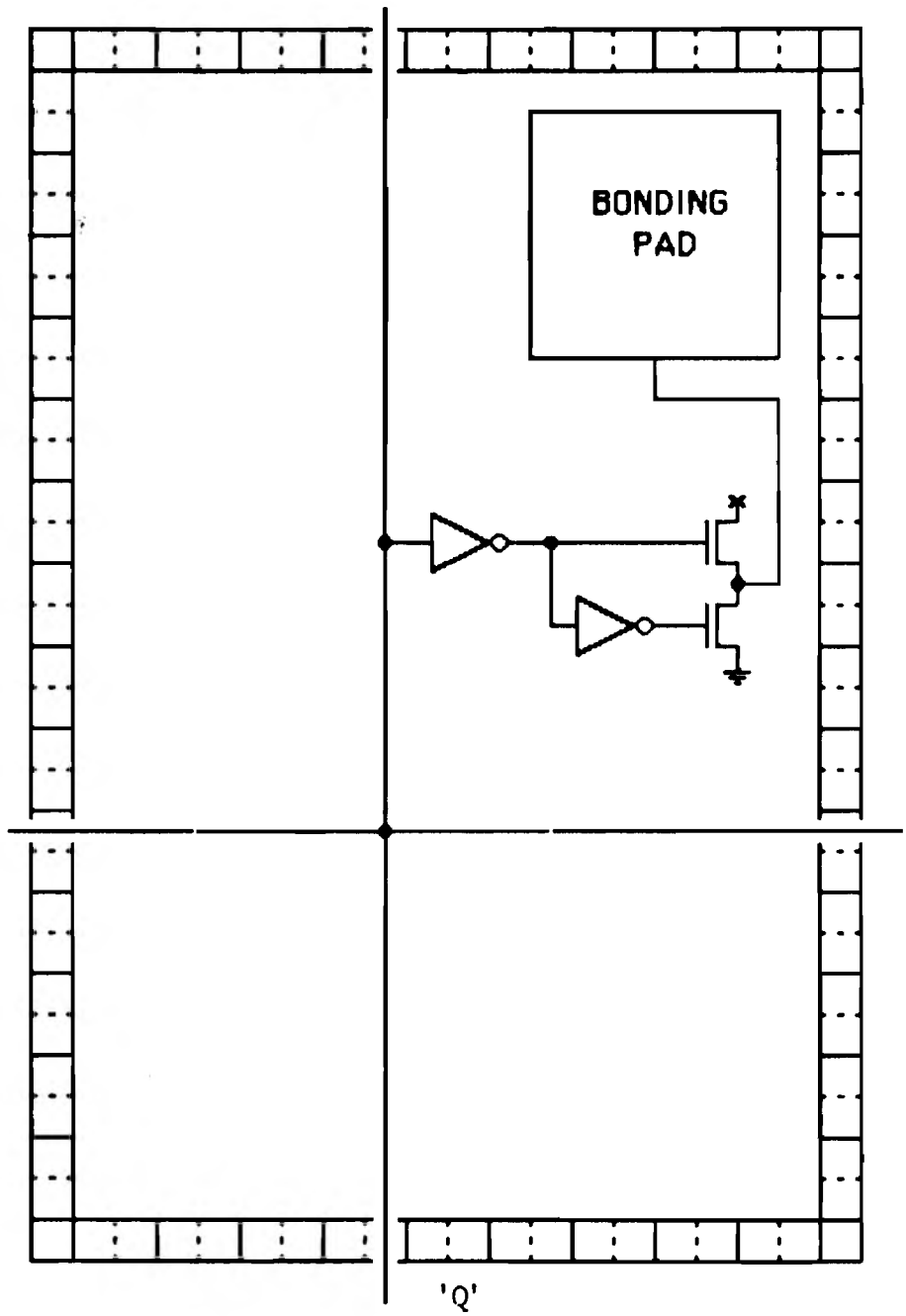

'G'

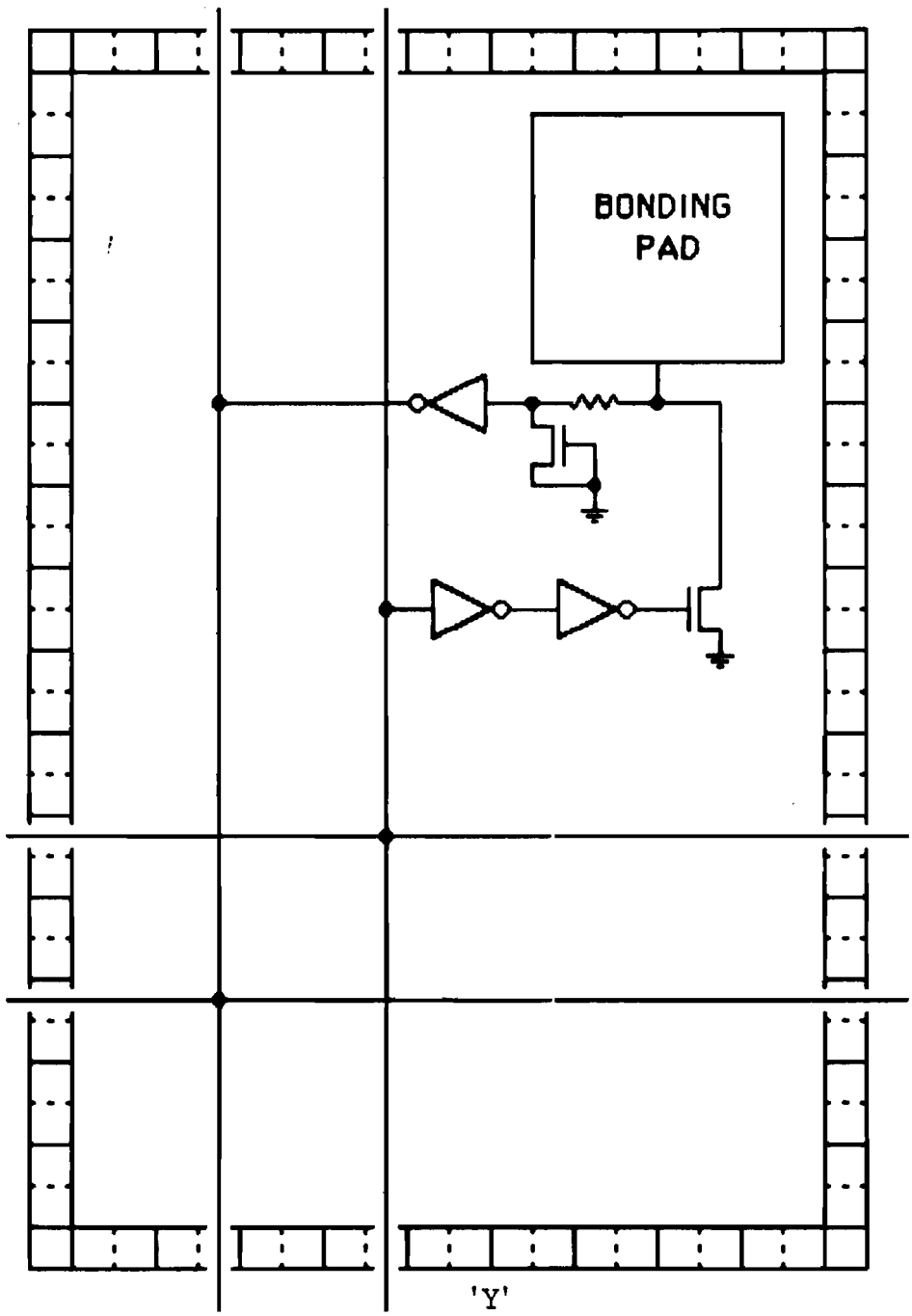

'T'

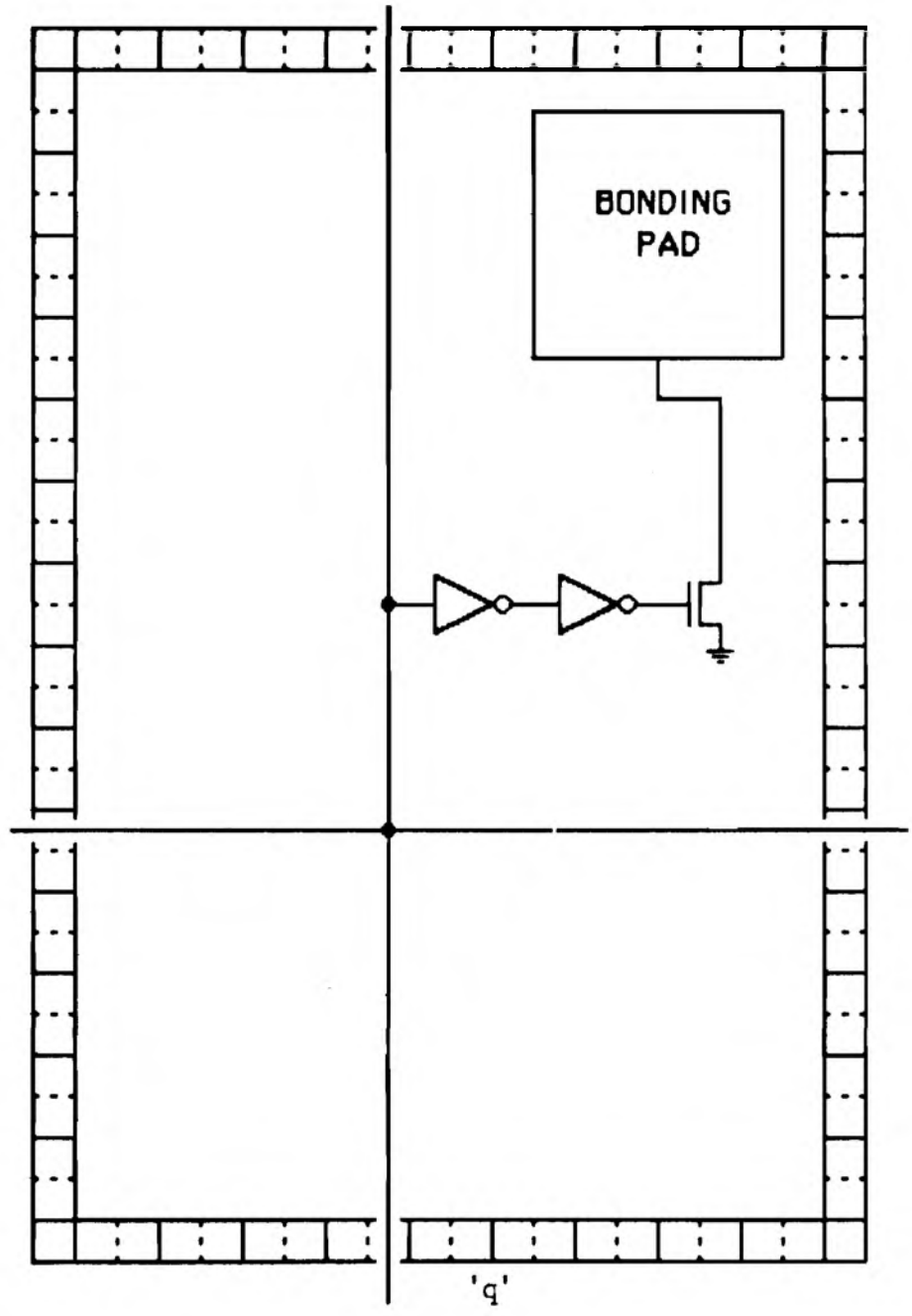

'M'


BONDING
PAD
'N'

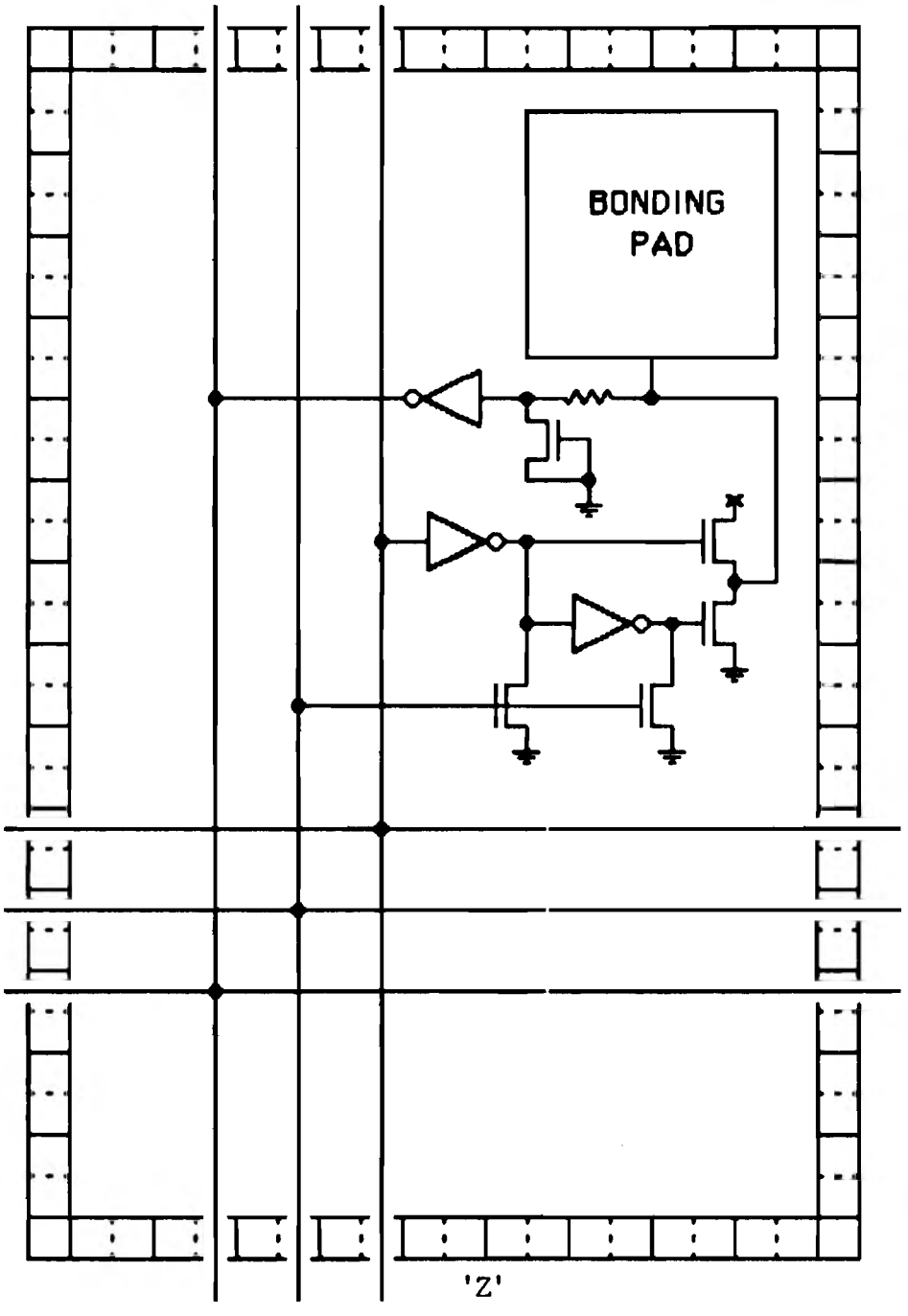

BONDING
PAD
'K'

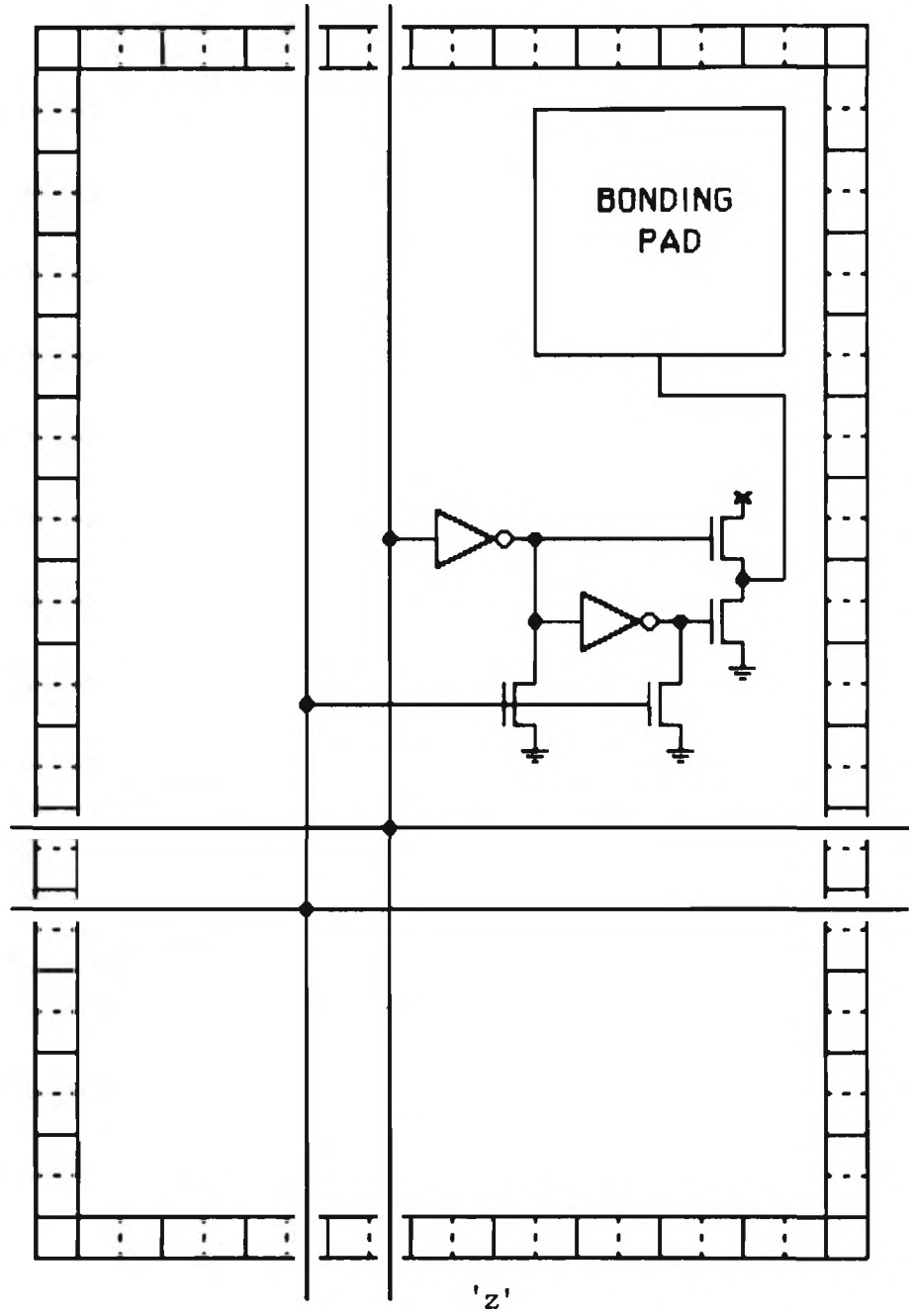

BONDING
PAD
'Q'


BONDING
PAD
'Y'



'q'


BONDING
PAD
'Z'


BONDING
PAD
'z'

5. NMOS PPL Cells

5.1. The PPL Methodology

Path Programmable Logic (PPL) is a cell based design system. When designing PPL circuits, a special text editor is used to place cells (represented by characters) on a regular rectangular grid. The PPL grid consists of many regularly spaced wires running vertically (column wires) and horizontally (row wires). A wire in the PPL grid normally extends the full height (or width) of the PPL array without connecting to any of the wires which run perpendicular to it. Special "routing" cells may be used to connect row wires to column wires. When cells that contain active devices are placed in the PPL array, the inputs and outputs of the cell are automatically connected to grid wires. Cells are connected together simply by positioning them so that they connect to the same grid wires. Cells placed in the same row are connected by the row wires of the grid, and cells placed in the same column are connected by the column wires of the grid. BREAKS may be placed between cells to split grid wires, so that connections between neighboring cells may be broken. Breaks are displayed in the editor as '|' and '_' characters. The '|' character indicates a broken row wire, and an underline indicates a broken column wire. Breaks that have been inserted into the PPL array may be removed by using the appropriate commands in the PPL editor.

PPL cells are almost always rectangular in shape, and always occupy a whole number of grid locations. The size of PPL cells is usually expressed in terms of rows and columns. Many of the cells require only a single grid location, meaning that they occupy one row and one column. Other cells may occupy a rectangle that is several rows high and several columns wide, depending on the complexity of the cell. The physical size of a single PPL row or column is fixed and does not vary between cells. Each PPL column will usually contain at least two column wires, and these are referred to as the "left" column (LCOL) and "right" column (RCOL) wires. Pairs of wires are used so that a signal and its compliment may be generated in the same column. PPL rows may contain one or more row wires, depending on the cell set. Complimentary pairs of signals are not usually created on row wires. Some cell sets may have a "primary" row wire which is used for signal inputs and outputs, along with a "secondary" row wire which may only be used for routing signals between columns.

When cells are placed in the PPL array, the pieces of circuitry in the cells can be thought of as being placed "under" the wires that make up the PPL grid. PPL cells usually cause some kind of interaction between signals on grid wires that pass through the cell. For example, a PPL cell might cause one of its row wires to be forced to a low voltage whenever a high voltage appears on a certain column wire, and let all other grid wires pass through the cell unaltered. Signals on "unused" wires are usually allowed to pass through the cell unaltered. Some PPL cells do not have enough "spare room" to allow all of the grid wires to pass through the cell. These cells cause "forced" breaks in the grid wires. Forced breaks are displayed in the same manner as breaks that are inserted by the user, but forced breaks cannot be removed from the grid except by removing the cell. Cells with forced breaks can be thought of as removing pieces of wire from the PPL grid in order to make room for the circuitry in the cell.

The characters used to represent PPL cells are usually chosen to reflect the logical function of the cell. The '1' and '0' characters might be used for cells that determine whether a signal is TRUE or FALSE, and the '+' character might be used to generate a logical OR of several signals. The 'F' character might be used to represent a flip-flop, and the 's' and 'r' characters might be used for cells which cause a flip-flop to be set or reset. The particular characters used for each cell will vary from cell set to cell set. A cell set designed for an NMOS process will probably be quite different than a cell set developed for CMOS.

Choosing appropriate characters to represent PPL cells allows PPL circuits to "look like" truth tables or state transition tables. Logical AND circuits are formed by placing several '0' and '1' cells together in a single row. This row wire acts as the output of a gate that uses column wires for inputs. The row wire then becomes TRUE when all of the column wire inputs to the '0' and '1' cells are TRUE. Logical OR circuits are formed by placing '+' cells together in a single column. This column acts as the output of a gate that uses row wires for inputs. The column then becomes TRUE when one or more of the row wire inputs to the '+' cells are TRUE. In this way, AND functions are generated on rows, while OR functions are formed on columns. Examples of a PPL exclusive-or module and a PPL full adder are shown below. In these modules, inputs enter on column wires and outputs exit on column wires, and the inputs and outputs all pass through to the top and bottom of the module. Row breaks are used to isolate the modules from each other. These

modules look very much like truth tables, when in fact the characters also represent the actual layout of the transistors needed to implement the logic functions. When PPL circuits are designed, both the logical function and the physical implementation of the circuits are created simultaneously.

```
|0 1 +|    |0 0 1 +  |
|1 0 +|    |0 1 0 +  |
           |1 0 0 +  |
           |1 1 1 +  |
           |  1 1   +|
           |1   1   +|
           |1 1     +|
```

The examples given above are not complete circuits, however, because only the cells used for the "logical" function of the module have been shown. Other cells are needed to implement the pullups needed by the row and column wires, and to generate the complimentary signals on the paired column wires. Below are the complete circuits, including the 'u' cell row pullups, and the '[' and ']' column pullups. The 'i' cells are inverters with inputs on the RCOL wire and outputs on the LCOL wires. Inputs to PPL modules are usually taken from the RCOL wires, because the '+' cell asserts the RCOL wire.

```
 [ [          [ [ [
 ] ] ]        ] ] ] ] ]
 i i          i i i
|0 1 + u|    |0 0 1 + u|
|1 0 + u|    |0 1 0 + u|
             |1 0 0 + u|
             |1 1 1 +  |
             |  1 1 u +|
             |1   1 u +|
             |1 1   u +|
```

Descriptions of all PPL cells in the NMOS cell set are included in this document.

5.2. Mixed Logic in PPL

The logical "meaning" that is associated with the physical voltages on a node determines the type of logic system in use. Nodes which use a HIGH voltage to represent a TRUE logic value (high-true) and a LOW voltage for a FALSE logic value are said to use POSITIVE logic. Nodes which use a LOW voltage as a TRUE logic value (low-true) and a

HIGH voltage as a FALSE logic value are said to use NEGATIVE logic. When the terms NAND and NOR are used in referring to gates, positive logic is implied so that NAND and NOR refer to the electrical function of the gate. Thus a "NAND gate" is a circuit that has a low voltage on its output whenever all of its inputs are at a high voltage, and a "NOR gate" is a circuit that has a low voltage on its output whenever any of its inputs is at a high voltage.

PPL cell sets are based on a "mixed" logic system. This is done so that the AND-OR structures described above may be physically implemented using only NAND gates or only NOR gates, but not both. The particular type of logic gate used depends on which process is used to fabricate the circuit (NMOS, CMOS, GaAs, etc.), and is based on circuit performance considerations. In general, NMOS circuits which are based on NOR gates can operate at higher speeds than similar NMOS circuits which are based on NAND gates. Thus NOR gates are preferred for circuits fabricated in NMOS. In most CMOS processes, NAND gates are preferred so that the faster n-channel transistors may be placed in series while the slower p-channel transistors are connected in parallel. This configuration can usually operate at higher speeds than the corresponding NOR gate which is constructed with series p-channel transistors and parallel n-channel transistors.

Cell sets which are based on electrical NOR gates (output low when any input is high) usually use positive logic on row wires while using negative logic on column wires. Cell sets which *are* based on electrical NAND gates (output low when all inputs are high) usually use positive logic on column wires while using negative logic on row wires. This allows a logical AND operation to be implemented by placing several '0' and '1' cells on the same row, and allows a logical OR operation to be implemented by placing several '+' cells on the same column. "Truth table" design of logic functions is performed in the conventional sum-of-products format found in most texts on logic design.

When describing the functions performed by the various cells in a PPL cell set, it is most convenient to describe the LOGICAL FUNCTION of the cells in a manner that is independent of the physical voltages used to represent the logic values involved. The cell descriptions which follow are usually worded in terms of the logical values (TRUE, FALSE) of the inputs and outputs of the cells rather than in terms of the electrical values (high, low) of the inputs and outputs. Since PPL designs use a mixed logic system,

knowing the voltage of a node is not equivalent to knowing the logic state of the node. On some nodes, a high voltage is used to represent TRUE, while on other nodes a high voltage is used to represent FALSE. Since there is no simple way to derive the logical state of a node from its voltage level, it would be confusing to try to describe the logical operation of the cells in terms of voltages. Occasionally, functional descriptions of cells are given in terms of voltages in order to clarify the operation of the cell under discussion. Functional descriptions of PAD cells will also include electrical information so that a proper interface to the "outside world" can be created.

The NMOS cell set is based on NMOS NOR gates. Most logic gates are formed by a single pullup transistor and one or more pulldown transistors. The pullup transistor pulls the output high when ALL of the pulldown transistors are turned off. When one or more of the pulldown transistors are turned on, the output is pulled to a low voltage. Thus any pulldown can force the output to a low voltage, but the output cannot be forced to a high voltage. Rather, the output may be "allowed" to remain high if none of the pulldowns are turned on. Occasionally, pulldown transistors are connected in series so that all of the transistors in the "stack" must be turned on in order to pull the output to a low voltage. Complex logic gates can be built by connecting the outputs of series stacks together, or by combining series stacks with "ordinary" pulldown transistors. In general, an output node will be at a high voltage only when all of its pulldowns and series stacks are "off", and the same node will be at a low voltage whenever one or more of its pulldowns and/or series stacks is "on".

In the NMOS cell set, positive (high-true) logic is mostly used on row wires, while negative (low-true) logic is mostly used on column wires. This implies that cells which affect the logic state of a row wire can either force the row wire to a FALSE logic value (low voltage), or allow the row wire to remain at a TRUE logic value (high voltage) if no other cells are forcing it false. In contrast, cells which affect the logic state of a column wire can either force the column wire TRUE (low voltage), or allow the column wire to remain FALSE (high voltage). The descriptions of the logical operation of the cells are phrased in terms of "forcing" signals TRUE/FALSE or "allowing" signals to remain TRUE/FALSE to help indicate the influence of other cells that are connected to the node. Also, a signal wire is said to be ASSERTED if it is in its TRUE state, regardless of the voltage levels involved. A signal is DEASSERTED or DISABLED when it is in its FALSE state.

5.3. NMOS Cell Set "Outline"

Basic PPL Cells

1) Wires, breaks, routing cells
   A) BLANK cell (wires everywhere)
   B) BREAK cells (breaking wires)
   C) Row/column interconnect cells
2) Input buffer pad, Output buffer pad cells, FILL cell ('K', 'Q', '')
3) Gates and Combinatorial logic
   A) Pullups (ratio'd logic only)
   B) Inverters ('i', 'v')
   C) AND elements ('0', '1')
   D) OR elements ('+', 'r', 's', '$')
4) Synchronous Flip Flops
   A) JK flip-flops ('F')
   B) Read enabled 'G' flip flop (stackable)

Intermediate PPL Cells

1) Universal Interconnect
   A) '?' cell
2) More inverters and buffers
   A) 'I', 'V', 'J', 'W' inverters and buffers
   B) 'k', 'n' Buffers for flip-flops
   C) 'j', 'w', '6', '9'
3) AND-OR and OR-AND elements
   A) 'a', 'o', 'E', 'X'
   B) '<', '>', 'O', 'I', 'S', 'R', 't', 'y'
4) Counters
   A) 'U', 'D' counter cells
5) Transparent Latches
   A) Dynamic ('p', 'b')
   B) Static ('P', 'B')
6) RAM cells
   A) Data elements ('M')
   B) Caps and Precharge elements ('m')
7) Asynchronous Latches ('h', 'H', 'T')
8) More Pad cells
   A) 'q', 'Y', 'Z'

Advanced PPL Cells (rarely used)

1) 'g', '^'
2) '(', ')', '\', '/'
3) '{', '}'

5.4. Basic PPL Cells

5.4.1. Default Cells

BLANK

Character: ' '
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) "normal", uses poly row wires, metal columns.
(1) *Z* use metal row wires, poly column wires.

Inputs: (none)
Outputs: (none)

The BLANK cell passes row and column wires which do not interact. This is the "default" cell that forms the wires in the PPL array.

NOTE: the *Z* modified version of this cell does not contain GND and VDD wires, and therefore can only be used in "special" sections of the PPL array that contain routing cells only. *Z* modified cells may only be placed next to other *Z* modified cells.

5.4.2. BREAK "cells"

BREAK cells are not actually cells, because breaks do not occupy a cell location in the PPL array. Breaks are placed between cells in the PPL array, and may be placed between any two adjacent cells. Breaks are represented by characters, and when a break character is placed at a cell location, the breaks take effect on the left and lower edges of the cell. Breaks are not always displayed as the character used to insert them. The PPL editor tries to display column breaks by underlining, and row breaks by the '|' character, whenever possible. On terminals that are not capable of underlining, other characters may be used to display break information.

ROWBRK

Character: '|'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks the ROW wire at left edge of cell.

AUXROWBRK

Character: ':'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks the AUXROW wire at left edge of cell.

LCOLBRK

Character: ',' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks left column (LCOL) wire at lower edge of cell.

RCOLBRK

Character: '.' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks right column (RCOL) wire at lower edge of cell.

BCOLBRK

Character: '=' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks right and left column wires at lower edge of cell.

LCOLRBRK

Character: '!' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks the left column wire and both row wires.

RCOLRBRK

Character: '%' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks the right column wire and both row wires.

ALLBREAK

Character: '&' (at left of cell character)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Breaks all wires at lower and left edges of cell.

5.4.3. Pseudo Cells

COVERED

Character: '"'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

The " character does not represent an actual cell. For cells which cover more than a single grid location, the " character is displayed at all grid locations covered by the cell except for the origin of the cell, which contains the character which represents the cell.

EMPTY

Character: '–'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

The empty cell prevents insertion of any layout data at the occupied grid location. This must not be confused with the BLANK cell, which contains row and column wires as well as power supply connections. The empty cell may be used to reserve space for insertion of custom designs into the PPL array.

5.4.4. Routing Cells

Routing cells are used to make connections between row and column wires. In order to conserve space, some routing cells may be "modified" to include a pullup that is connected to the intersecting wires. This allows some pullups to be combined with wire intersections so that no separate cell location is needed for the pullup. The connections allowed by cells in this section are adequate to handle most simple routing of signals between modules. Advanced users may want to use the '?' cell (universal intersection), which is a more flexible cell that allows complex connections to be 'built' with appropriate modifier values.

XTL

Character: '*'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) connection only, no pullup.
(1) pullup added to connection.
(2) 1/2 strength pullup added to connection.
(3) 1/4 strength pullup added to connection.
(4) 2 strength pullup added to connection.
(5) 4 strength pullup added to connection.
(6) 8 strength pullup added to connection.
(7) *Z* use metal row wires, poly column wires.

Inputs: (none)
Outputs: (none)

Intersection (ohmic contact) between the Top row wire and the Left column wire. This cell may be used for routing signals in the PPL array.

NOTE: the *Z* modified version of this cell does not contain GND and VDD wires, and therefore can only be used in "special" sections of the PPL array that contain routing cells only. *Z* modified cells may only be placed next to other *Z* modified cells.

XTR

Character: '#'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) connection only, no pullup.
(1) pullup added to connection.
(2) 1/2 strength pullup added to connection.
(3) 1/4 strength pullup added to connection.
(4) 2 strength pullup added to connection.
(5) 4 strength pullup added to connection.
(6) 8 strength pullup added to connection.
(7) *Z* use metal row wires, poly column wires.

Inputs: (none)
Outputs: (none)

Intersection (ohmic contact) between the Top row wire and the Right column wire. This cell may be used for routing signals in the PPL array.

NOTE: the *Z* modified version of this cell does not contain GND and VDD wires, and therefore can only be used in "special" sections of the PPL array that contain routing cells only. *Z* modified cells may only be placed next to other *Z* modified cells.

XTLTR

Character: '@'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) connection only, no pullup.
(1) pullup added to connection.
(2) 1/2 strength pullup added to connection.
(3) 1/4 strength pullup added to connection.
(4) 2 strength pullup added to connection.
(5) 4 strength pullup added to connection.
(6) 8 strength pullup added to connection.
(7) *Z* use metal row wires, poly column wires.

Inputs: (none)
Outputs: (none)

Intersection between the Top row wire and the Left and Right column wires. This cell may be used for routing signals in the PPL array.

NOTE: the *Z* modified version of this cell does not contain GND and VDD wires, and therefore can only be used in "special" sections of the PPL array that contain routing cells only. *Z* modified cells may only be placed next to other *Z* modified cells.

COLSHORT

Character: '~'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) connection only, no pullup.
(1) pullup added to connection.
(2) 1/2 strength pullup added to connection.
(3) 1/4 strength pullup added to connection.
(4) 2 strength pullup added to connection.
(5) 4 strength pullup added to connection.
(6) 8 strength pullup added to connection.
(7) *Z* use metal row wires, poly column wires.

Inputs: (none)
Outputs: (none)

Short together the Left and Right column wires. This cell may be used for routing signals in the PPL array.

NOTE: the *Z* modified version of this cell does not contain GND and VDD wires, and therefore can only be used in "special" sections of the PPL array that contain routing cells only. *Z* modified cells may only be placed next to other *Z* modified cells.

SWAPCOLS

Character: 'x'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Crossover of the right and left column wires.

5.4.5. Simple I/O Pads

IBUFPAD

Character: 'K'
Cell size: 13 rows, 8 columns. (NMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: None.

Inputs: P (external pad connection)
Outputs: Q (right column wire of first column to right of cell origin, or second row above cell origin)

Inverting input buffer pad. Drive capability is 10x standard size.

OBUFPAD

Character: 'Q'
Cell size: 14 rows, 9 columns. (NMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: None.

Inputs: D (right column wire of third column to right of cell origin, or fourth row above cell origin)
Outputs: P (external pad connection)

Inverting output buffer pad. Drive capability is one TTL load.

FILL

Character: "" (single quote)
Cell size: 1 row, 1 column.
Placement: Edges of circuit only.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Fill in between pad cells on edges of circuit. Must be placed in all locations that are not occupied by a pad cell. A FILL cell placed in the lower left corner of the PPL array is automatically converted in a GND pad. A FILL cell placed in the upper right corner of the PPL array is automatically converted into a VDD pad.

5.4.6. Pullup Cells

ROWPUP

Character: 'u'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) standard size pullup.
(1) 1/2 strength pullup.
(2) 1/4 strength pullup.
(3) 2 strength pullup.
(4) 4 strength pullup.
(5) 8 strength pullup.

Inputs: (none)
Outputs: Q (row wire)

ROW PullUP cells. Make Q true whenever all other cells allow Q to remain true.

LCOLPUP

Character: '['
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) standard size pullup.
(1) 1/2 strength pullup.
(2) 1/4 strength pullup.
(3) 2 strength pullup.
(4) 4 strength pullup.
(5) 8 strength pullup.

Inputs: (none)
Outputs: Q (left column wire)

Left column wire pullups.

RCOLPUP

Character: ']'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) standard size pullup.
(1) 1/2 strength pullup.
(2) 1/4 strength pullup.
(3) 2 strength pullup.
(4) 4 strength pullup.
(5) 8 strength pullup.

Inputs: (none)
Outputs: Q (right column wire)

Right column wire pullups.

5.4.7. Basic Logic Elements

INV1

Character: 'i'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: A (right column wire)
Outputs: Q (left column wire)

The Q output remains true until the A input becomes true. No internal pullups are provided.

INV0

Character: 'v'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: A (left column wire)
Outputs: Q (right column wire)

The Q output remains true until the A input becomes true. No internal pullups are provided.

ONE

Character: '1'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: ONE (right column wire)
Outputs: Q (row wire)

If ONE is true, Q is allowed to remain true. Otherwise, Q is made false. Other cells connected to the Q output may influence the value of the Q output. In particular, other '0', '1', and 'a' cells placed on the same row produce a logical AND of the column wires which act as inputs to these cells. Other cells such as the 'o', 'E', 'X', '>', and '<' cells may be combined with the '0' and '1' cells to produce more complex OR-AND type logic elements.

ZERO

Character: '0' (zero)
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (1) add pullup to output column.

Inputs: ZERO (left column wire)
Outputs: Q (row wire)

If the ZERO is true, Q is allowed to remain true. Otherwise, Q is made false. Other cells connected to the Q output may influence the value of the Q output. In particular, other '0', '1', and 'a' cells placed on the same row produce a logical AND of the column wires which act as inputs to these cells. Other cells such as the 'o', 'E', 'X', '>', and '<' cells may be combined with the '0' and '1' cells to produce more complex OR-AND type logic elements.

PLUS

Character: '+'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) normal strength cell.
(1) add pullup on output column.
(2) 2 strength cell, no pullup.
(3) 4 strength cell, no pullup.

Inputs: A (row wire)
Outputs: Q (right column wire)

Same as 'r' cell. If A is true, make the Q output true. Otherwise allow the Q output to remain false. Other cells connected to the Q output may influence the value of the Q output. In particular, several '+' cells placed in the same column produce a logical OR of the rows which act as inputs to the '+' cells.

RESET

Character: 'r'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) normal strength cell.
(1) add pullup on output column.
(2) 2 strength cell, no pullup.
(3) 4 strength cell, no pullup.

Inputs: A (row wire)
Outputs: Q (right column wire)

Same as '+' cell. If A is true, make the Q output true. Otherwise allow the Q output to remain false. Other cells connected to the Q output may influence the value of the Q output. In particular, several 'r' cells placed in the same column produce a logical OR of the rows which act as inputs to the 'r' cells.

SET

Character: 's'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: (0) normal strength cell.
(1) add pullup on output column.
(2) 2 strength cell, no pullup.
(3) 4 strength cell, no pullup.

Inputs: A (row wire)
Outputs: Q (left column wire)

If A is true, make the Q output true. Otherwise allow the Q output to remain false. Other cells connected to the Q output may influence the value of the Q output. In particular, several 's' cells placed in the same column produce a logical OR of the rows which act as inputs to the 's' cells.

SPLUS

Character: '$'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: A (row wire)
Outputs: S (left column wire)
R (right column wire)

Combination of an S cell and a PLUS (or RESET) cell. When the row is true, both the S and R outputs are made true. Otherwise, S and/or R are allowed to remain false. Other cells connected to the S and/or R outputs may independently influence the values of these outputs. In particular, other 's', 'r', '+', and '$' cells in the same column produce a logical OR of the rows which act as inputs to these cells.

5.4.8. Synchronous Flip-flops

JK

Character: 'F'
Cell size: 4 rows, 2 columns.
Placement: ODD row, ODD column.
Modifiers: (0) "normal", 0/R column on left, S/1 on right.
(1) mirror columns (S/1 on left, 0/R on right).
(2) mirror rows.
(3) mirror rows and columns.

Inputs: SET (left column wire of right column)
RESET (right column wire of left column)
PHI1 (upper row wire)
PHI2 (lower row wire)
Outputs: Q (right column wire of right column)
Qnot (left column wire of left column)

This synchronous JK flip flop changes its internal state when the PHI1 input becomes true. State changes are not passed to the outputs until the PHI2 input becomes true, which cannot occur while PHI1 remains true (the two clocks must be non-overlapping). The flip flop will become set (Q true and Qnot false) if the SET input is asserted during PHI1, and will become reset (Q false, Qnot true) if the RESET input is asserted during PHI1. If both SET and RESET are asserted during PHI1, the flip flop will toggle state.

NOTE: Pullups for all four column wires are built into this cell.

JKR

Character: 'G'
Cell size: 4 rows, 2 columns.
Placement: Even row, odd column.
Modifiers: (0) "normal", 0/R column on left, S/1 on right.
(1) mirror columns (S/1 on left, 0/R on right).
(2) mirror rows.
(3) mirror rows and columns.

Inputs: SET (left column wire of left column)
RESET (right column wire of right column)
PHI1 (upper row wire)
PHI2 (middle row wire)
READ (lower row wire)
Outputs: Q (right column wire of right column)
Qnot (left column wire of left column)

This is a stackable JK flip flop which operates in a manner similar to the JK cell. An additional READ input is used to enable the outputs of only one of the flip flops in the stack. In order for the flip flop to change state properly, the PHI1 input should only be made true while the READ input of the same cell is true, and only one cell in the stack should be enabled at any one time.

NOTE: External pullups of 1/2 normal strength (or less) must be used on all four column wires connected to this cell.

5.5. Intermediate PPL Cells

UNICON

Character: '?'
Cell size: 1 row, 1 column.
Placement: ANY row, ANY column.
Modifiers: [formed by summing values below]
(1) connect right column wire with main row wire.
(2) connect left column wire with main row wire.
(4) connect right column wire with aux. row wire.
(8) connect left column wire with aux. row wire.
(16) break right column wire, between rows.
(32) break left column wire, between rows.
(64) break main row wire, between columns.
(128) break auxiliary row wire, between columns.

Inputs: (none)
Outputs: (none)

Universal interconnect cell. Modifier value determines connections between the row and column wires, and/or internal breaks of row and column wires. Used for complex routing connections where ordinary *, #, ~, and cells are not adequate. A modifier value of ZERO causes no internal connections and no internal breaks, which is equivalent to a BLANK cell. Various combinations of internal connections and internal breaks may be formed by summing the modifier values listed above.

NOTE: Because of cell mirroring, internal wire breaks (modifier values greater than 15) should not be used when the cell is placed in EVEN rows or EVEN columns.

5.5.1. Inverters and Buffers

INVERT1

Character: 'I'
Cell size: 2 rows, 1 column.
Placement: Even row, any column.
Modifiers: (0) Standard sized inverter, pullups on A and Q.
(1) 5x sized inverter with 5x output pullup on Q.
(2) 5x sized inverter with no pullups.

Inputs: A (right column wire)
Outputs: Q (left column wire)

The Q output remains true until the A input becomes true. Modifiers are used to specify transistor sizes and pullups.

INVERT0

Character: 'V'
Cell size: 2 rows, 1 column.
Placement: Even row, any column.
Modifiers: (0) Standard sized inverter, pullups on A and Q.
(1) 5x sized inverter with 5x output pullup on Q.
(2) 5x sized inverter with no pullups.

Inputs: A (left column wire)
Outputs: Q (right column wire)

The Q output remains true until the A input becomes true. Modifiers are used to specify transistor sizes and pullups.

BIGBUF1

Character: 'J'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (0) 8x buffer with 8x pullup included.
(1) remove 8x pullup from Q output column.

Inputs: A (right column wire)
Outputs: Q (left column wire)

Large sized (8x) inverting buffer. The standard unmodified version of the cell includes both an 8x pulldown and an 8x pullup.

BIGBUF0

Character: 'W'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (0) 8x buffer with 8x pullup included.
(1) remove 8x pullup from Q output column.

Inputs: A (left column wire)
Outputs: Q (right column wire)

Large sized (8x) inverting buffer. The standard unmodified version of the cell includes both an 8x pulldown and an 8x pullup.

FFINV0

Character: 'n'
Cell size: 1 row, 2 column.
Placement: Any row, odd column.
Modifiers: (1) mirror columns.

Inputs: I (left wire of left column)
Outputs: Q (right wire of right column)

Invert the ZERO output of a flip-flop, with a 4x strength pulldown, onto the ONE output column of the flip-flop. Used for buffering the output of a flip-flop that must drive many '1' cells (or other inputs).

FFINV1

Character: 'k'
Cell size: 1 row, 2 column.
Placement: Any row, odd column.
Modifiers: (1) mirror columns.

Inputs: I (right wire of right column)
Outputs: Q (left wire of left column)

Invert the ONE output of a flip-flop, with a 4x strength pulldown, onto the ZERO output column of the flip-flop. Used for buffering the output of a flip-flop whose ZERO output must drive many '0' cells (or other inputs).

ROWINVRL

Character: 'j'
Cell size: 1 row, 1 column.
Placement: Any row, odd column.
Modifiers: (0) normal strength, pulldown only.
(1) add pullup to Q output.
(2) 4 strength, pulldown only.

Inputs: A (row wire on right edge)
Outputs: Q (row wire on left edge)

In-line row wire inverter with optional output pullup.

ROWINVLR

Character: 'w'
Cell size: 1 row, 1 column.
Placement: Any row, odd column.
Modifiers: (0) normal strength, pulldown only.
(1) add pullup to Q output.
(2) 4 strength, pulldown only.

Inputs: A (row wire on left edge)
Outputs: Q (row wire on right edge)

In-line row wire inverter with optional output pullup.

INVERTTB

Character: '6'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (0) inverter with no pullups
(1) inverter with output pullup
(2) inverter with input pullup
(3) inverter with input and output pullups

Inputs: A (row wire of upper row)
Outputs: Q (row wire of lower row)

Invert from top row wire to bottom row wire.

INVERTBT

Character: '9'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (0) inverter with no pullups
(1) inverter with output pullup
(2) inverter with input pullup
(3) inverter with input and output pullups

Inputs: A (row wire of lower row)
Outputs: Q (row wire of upper row)

Invert from the Bottom row onto the Top row wire.

5.5.2. Intermediate Logic Elements

AND01

Character: 'a'
Cell size: 2 rows, 1 column.
Placement: Even row, any column.
Modifiers: (1) add pullup to output row.
(2) mirror rows.
(3) mirror rows, add pullup to output row.

Inputs: A (left column wire)
B (right column wire)
Outputs: Q (lower row wire)

This cell leaves the Q output true if the A input is true AND the B input is true. Otherwise Q is made false. Other cells connected to the Q output may influence the value of Q.

OR01

Character: 'o'
Cell size: 2 rows, 1 column.
Placement: Even row, any column.
Modifiers: (1) add pullup to output row.
(2) mirror rows.
(3) mirror rows, add pullup to output row.

Inputs: A (left column wire)
B (right column wire)
Outputs: Q (lower row wire)

This cell leaves the Q output true if the A input is true OR the B input is true. Otherwise Q is made false. Other cells connected to the Q output may influence the value of Q.

EQUIV

Character: 'E'
Cell size: 2 rows, 2 columns.
Placement: Even row, any column.
Modifiers: (1) add pullup to output row.
(2) mirror rows.
(3) mirror rows, add pullup to output row.

Inputs: A (right column wire of left column)
A* (left column wire of left column)
B (right column wire of right column)
B* (left column wire of right column)
Outputs: Q (lower row wire)

The Q output is left true if (A and B) is true, or (A* and B*) is true. Otherwise, Q is made false. When A is the inverse of A*, and B is the inverse of B*, this performs an equivalence function on the A and B columns. However, the cell can perform more general and-or type logic when four distinct signals are used for inputs. In particular, several 'E' cells placed side by side form an arithmetic comparator which leaves Q true whenever the two input values are equal. Other cells connected to the Q output may influence the value of the Q output.

EXOR

Character: 'X'
Cell size: 2 rows, 2 columns.
Placement: Even row, any column.
Modifiers: (1) add pullup to output row.
(2) mirror rows.
(3) mirror rows, add pullup to output row.

Inputs: A (right column wire of left column)
A* (left column wire of left column)
B (right column wire of right column)
B* (left column wire of right column)
Outputs: Q (lower row wire)

The Q output is left true if (A and B*) is true, or (A* and B) is true. Otherwise, Q is made false. When A is the inverse of A*, and B is the inverse of B*, this performs an exclusive-or function on the A and B columns. However, the cell can perform more general and-or type logic when four distinct signals are used for inputs. Other cells connected to the Q output may influence the value of the Q output.

ENBINV1

Character: '<'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (1) mirror rows.

Inputs: ENABLE (row wire of lower row)
A (right column wire)
Outputs: Q (left column wire)

If ENABLE is true, the cell acts as an INV1 cell by inverting the A input onto the Q output.

ENBINV0

Character: '>'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (1) mirror rows.

Inputs: ENABLE (row wire of lower row)
A (left column wire)
Outputs: Q (right column wire)

If ENABLE is true, the cell acts as an INV0 cell by inverting the A input onto the Q output.

ENBLD1

Character: 'l' (lower case 'L')
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (1) add pullup to output row.
(2) mirror rows.
(3) mirror rows, add pullup to output row.

Inputs: ENABLE (row wire on lower row)
ONE (right column wire)
Outputs: Q (row wire on upper row)

If ENABLE is true and ONE is true, Q is allowed to remain true. Otherwise Q is made false. Other cells connected to the Q output may influence the value of Q.

ENBLD0

Character: 'O' (Upper case 'o')
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (1) add pullup to output row.
(2) mirror rows.
(3) mirror rows, add pullup to output row.

Inputs: ENABLE (row wire on lower row)
ZERO (left column wire)
Outputs: Q (row wire on upper row)

If ENABLE is true and ZERO is true, Q is allowed to remain true. Otherwise Q is made false. Other cells connected to the Q output may influence the value of Q.

ANDRESET

Character: 'R'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (1) add pullup to output column.
(2) mirror rows.
(3) mirror rows, add pullup to output column.

Inputs: A (row wire of upper row)
B (row wire of lower row)
Outputs: Q (right column wire)

If A is true AND B is true, make the Q output true. Otherwise allow the Q output to remain false. Other cells connected to the Q output may influence the value of the Q output.

ANDSET

Character: 'S'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (1) add pullup to output column.
(2) mirror rows.
(3) mirror rows, add pullup to output column.

Inputs: A (row wire of upper row)
B (row wire of lower row)
Outputs: Q (left column wire)

If A is true AND B is true, make the Q output true. Otherwise allow the Q output to remain false. Other cells connected to the Q output may influence the value of the Q output.

NORBR

Character: 't'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (1) add pullup to output row.
(2) mirror rows.
(3) mirror rows, add pullup to output row.

Inputs: A (row wire of lower row)
B (right column wire)
Outputs: Q (row wire of upper row)

This is a strange cell. If (A is true) OR (B is false), Q is made false. Otherwise Q is allowed to remain true.

In terms of voltages, if (A is high) OR (B is high), Q is made low. Otherwise Q is allowed to remain high. Other cells connected to the Q output may pull Q low.

NORBL

Character: 'y'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: (1) add pullup to output row.
(2) mirror rows.
(3) mirror rows, add pullup to output row.

Inputs: A (row wire of lower row)
B (left column wire)
Outputs: Q (row wire of upper row)

This is a strange cell. If (A is true) OR (B is false), Q is made false. Otherwise Q is allowed to remain true.

In terms of voltages, if (A is high) OR (B is high), Q is made low. Otherwise Q is allowed to remain high. Other cells connected to the Q output may pull Q low.

UPCOUNT

Character: 'U'
Cell size: 2 rows, 2 columns.
Placement: Any row, odd column.
Modifiers: (0) "normal" cell, carry in/out on lower row.
(1) mirror rows, carry in/out on upper row.
(2) mirror columns (use with mirrored flip-flops).
(3) mirror rows and columns.

Inputs: Cin (lower row wire, right edge)
ONE (right column wire of right column)
Outputs: Cout (lower row wire, left edge)
SET (left column wire of right column)
RESET (right column wire of left column)

When used with a JK flip flop, forms one bit of an up counter. The SET and RESET outputs are asserted whenever the Cin input is true. The Cout output becomes true when the Cin input is true and the ONE input is true. These cells may be placed side by side in order to feed the Cout from one cell to the Cin of the cell to the left. These cells may also be used in conjunction with the DOWNCOUNT cell to form up/down counters.

DOWNCOUNT

Character: 'D'
Cell size: 2 rows, 2 columns.
Placement: Any row, odd column.
Modifiers: (0) "normal" cell, carry in/out on lower row.
(1) mirror rows, carry in/out on upper row.
(2) mirror columns (use with mirrored flip-flops).
(3) mirror rows and columns.

Inputs: Bin (lower row wire, right edge)
ZERO (left column wire of left column)
Outputs: Bout (lower row wire, left edge)
SET (left column wire of right column)
RESET (right column wire of left column)

When used with a JK flip flop, forms one bit of a down counter. The SET and RESET outputs are asserted whenever the Bin input is true. The Bout output becomes true when the Bin input is true and the ZERO input is true. These cells may be placed side by side in order to feed the Bout from one cell to the Bin of the cell to the left. These cells may also be used in conjunction with the UPCOUNT cell to form up/down counters.

5.5.3. Static RAM

TWOBITRAM

Character: 'M'
Cell size: 4 rows, 2 columns.
Placement: Even row, odd column.
Modifiers: (1) mirror columns.

Inputs: R (right column wire of left column)
S (left column wire of right column)
WRITE-TOP (row wire at top of cell)
READ-TOP (row wire below WRITE-TOP)
WRITE-BOT (row wire at bottom of cell)
READ-BOT (row wire above WRITE-BOT)
Outputs: Q (right column wire of right column)

Two ram bits built into a single cell. Each ram bit has its own read and write strobes on row wires. Set and Reset column wires are used to write data into the ram, and optional precharge transistors in the RAMCAP cell may be used to improve the speed of write operations. A single data output is connected to the 1 wire of the right column. Read and write operations in different cells may overlap.

RAMCAP

Character: 'm'
Cell size: 1 row, 2 columns.
Placement: Any row, odd column.
Modifiers: (1) add precharge transistors on R and S columns.
(2) mirror columns.
(3) mirror columns, add precharge transistors.

Inputs: CHARGE (row wire) (only with modified cell)
Outputs: R (right column wire of left column) (only with modified cell)
S (left column wire of right column) (only with modified cell)

Cell to cap off top and bottom of RAM arrays. The modified cell also may be used as a column precharge on the S and R wires. When CHARGE is true, large enhancement transistors allow current to flow from VDD into the R and S column wires.

5.5.4. Transparent Latches

UPASINV

Character: 'p'
Cell size: 2 rows, 1 column.
Placement: Even row, any column.
Modifiers: (0) left column wire passes through unused.
(1) short output to left column wire.
(2) short input to left column wire.
(3) short input and output to left column wire.

Inputs: PHI (lower row wire)
D (right column wire, lower edge)
Outputs: Q (right column wire, upper edge)

Up PASs-transistor INVerter circuit. Half of a dynamic shift register which passes data up the right column wire. The pass transistor passes data from the D input to the output inverter when PHI is true. The output of the inverter is connected to Q.

DPASINV

Character: 'b'
Cell size: 2 rows, 1 column.
Placement: Even row, any column.
Modifiers: (0) left column wire passes through unused.
(1) short output to left column wire.
(2) short input to left column wire.
(3) short input and output to left column wire.

Inputs: PHI (upper row wire)
D (right column wire, upper edge)
Outputs: Q (right column wire, lower edge)

Down PASs-transistor INVerter circuit. Half of a dynamic shift register which passes data down the right column wire. The pass transistor passes data from the D input to the output inverter when PHI is true. The output of the inverter is connected to Q.

UQSTAT

Character: 'P'
Cell size: 4 rows, 1 column.
Placement: Even row, any column.
Modifiers: (0) left column wire passes through unused.
(1) short output to left column wire.
(2) short input to left column wire.
(3) short input and output to left column wire.

Inputs: PHI1 (lower row wire)
PHI2 (lower middle row wire)
PHI3 (upper middle row wire)
D (right column wire, lower edge)
Outputs: Q (right column wire, upper edge)

Up Quasi-STATic circuit. Half of a static shift register which passes data up the right column wire. The input pass transistor passes data from the D input to the output inverter when PHI1 is true. The output of this inverter is connected to Q and to the input of a feedback inverter. The output of the feedback inverter is passed back to the output inverter through two series connected pass transistors which are gated by PHI2 and PHI3. Normally, PHI2 and PHI3 are connected to the same feedback signal, usually the opposite of the PHI1 signal.

DQSTAT

Character: 'B'
Cell size: 4 rows, 1 column.
Placement: Even row, any column.
Modifiers: (0) left column wire passes through unused.
(1) short output to left column wire.
(2) short input to left column wire.
(3) short input and output to left column wire.

Inputs: PHI1 (upper row wire)
PHI2 (upper middle row wire)
PHI3 (lower middle row wire)
D (right column wire, upper edge)
Outputs: Q (right column wire, lower edge)

Down Quasi-STATic circuit. Half of a static shift register which passes data down the right column wire. The input pass transistor passes data from the D input to the output inverter when PHI1 is true. The output of this inverter is connected to Q and to the input of a feedback inverter. The output of the feedback inverter is passed back to the output inverter through two series connected pass transistors which are gated by PHI2 and PHI3. Normally, PHI2 and PHI3 are connected to the same feedback signal, usually the opposite of the PHI1 signal.

5.5.5. Asynchronous Latches

LATCH2

Character:	'h'
Cell size:	2 rows, 1 column.
Placement:	Even row, any column.
Modifiers:	(1) mirror rows.
Inputs:	SET (RIGHT column wire, same as Q output)
	RESET (LEFT column wire, same as Qnot output)
Outputs:	Q (right column wire)
	Qnot (left column wire)

This two wire asynchronous latch cell has built in pullups. To set the latch, the Q output is asserted. To reset the latch, the Qnot output is asserted. The two outputs should never be asserted simultaneously by external circuitry, as the resulting state of the latch will be unknown.

LATCH

Character:	'H'
Cell size:	2 rows, 2 columns.
Placement:	Even row, odd column.
Modifiers:	(0) "normal", 0/R column on left, S/1 on right.
	(1) mirror columns (S/1 on left, 0/R on right).
Inputs:	SET (left column wire of right column)
	RESET (right column wire of left column)
Outputs:	Q (right column wire of right column)
	Qnot (left column wire of left column)

This four wire asynchronous latch cell has internal pullups and buffered outputs. Asserting the SET input causes the Q output to become true and the Qnot output to become false. Asserting the RESET input causes the Q output to become false and the Qnot output to become true. The SET and RESET inputs should not be asserted simultaneously, as the resulting state of the latch will be unknown.

ETFF

Character: 'T'
Cell size: 4 rows, 2 columns.
Placement: Even row, odd column.
Modifiers: (0) "normal", 0/R column on left, S/1 on right.
(1) mirror columns (S/1 on left, 0/R on right).
(2) mirror rows.
(3) mirror rows and columns.

Inputs: SET (left column wire of right column)
RESET (right column wire of left column)
Outputs: Q (right column wire of right column)
Qnot (left column wire of left column)

This cell is a static asynchronous flip flop with edge triggered SET and RESET inputs. The flip flop changes from the RESET state to the SET state when the SET input is asserted, regardless of the value of the RESET input. Similarly, the flip flop changes from the SET state to the RESET state on a true-going change in the RESET input, regardless of the value of the SET input.

The flip flop will also act as an edge-triggered toggle flip flop if the SET and RESET inputs are physically connected so that they change at the same time. **CAUTION:** In order for the flip flop to function properly as a toggle flip flop, the SET and RESET inputs MUST be connected together by an ohmic contact placed close to the flip flop. Proper operation will not be guaranteed to occur if the SET and RESET inputs are left as two independent signals which are asserted "simultaneously", because variations in capacitive loading on the two independent inputs could cause them to change at different rates.

The Q and Qnot outputs may also be used as inputs under special conditions. When the flip flop is in the SET state with Q low/true and Qnot high/false, a pulldown which asserts the Qnot wire will cause the flip flop to change to the RESET state. When the flip flop is in the RESET state with Qnot low/true and Q high/false, a pulldown which asserts the Q wire will cause the flip flop to change to the SET state. **NOTE:** This method of changing the state of the flip flop should be used only to initialize the flip flop to a known state. The Q and Qnot wires MUST **NOT** be asserted at the same time, as this will leave the flip flop in an undetermined state. This method is equivalent to a level-sensitive preset/clear which overrides the normal SET and RESET inputs, and should not be used while the SET and/or RESET inputs are changing.

5.5.6. More Pad Cells

IPROPAD

Character: 'N'
Cell size: 13 rows, 8 columns. (NMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: None.

Inputs: P (external pad connection)
Outputs: Q (right column wire of first column to right of cell origin, or second row above cell origin)

Input protection pad with no buffer. Drive capability is determined by the external drive signal.

CAUTION: This pad is NOT TTL COMPATIBLE and should be used only when the externally applied signal is capable of producing 'high' voltages of VDD.

OCOBUFPAD

Character: 'q'
Cell size: 14 rows, 9 columns. (NMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: None.

Inputs: D (right column wire of third column to right of cell origin, or fourth row above cell origin)
Outputs: P (external pad connection)

Inverting output buffer pad with open collector output. Requires external pullup resistor. May be wire-or'ed with other open collector pads. Drive capability is one TTL load.

OCIOPAD

Character: 'Y'
Cell size: 14 rows, 9 columns. (NMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: None.

Inputs: Dout (right column wire of third column to right of cell origin, or fourth row above cell origin)
P (external pad connection)
Outputs: Din (right column wire of first column to right of cell origin, or second row above cell origin)
P (external pad connection)

This pad combines an open collector output pad with an inverting input buffer pad. Requires an external pullup resistor. May be wire-or'ed with other open collector pads. External drive capability is one TTL load. Internal drive capability of the input buffer is 10x standard size.

This pad is physically and logically equivalent to placing an OCOBUFPAD cell on top of an IBUFPAD pad cell.

TSIOPAD

Character: 'Z'
Cell size: 14 rows, 9 columns. (NMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: None.

Inputs: DISABLE (right column wire of second column to right of cell origin, or third row above cell origin)
Dout (right column wire of third column to right of cell origin, or fourth row above cell origin)
P (external pad connection)
Outputs: Din (right column wire of first column to right of cell origin, or second row above cell origin)
P (external pad connection)

This pad combines a tri-state output buffer pad with an inverting input buffer pad. The inverting input buffer is active at all times, and data at the Din connection is inverted with respect to data on the pad itself. When the DISABLE input is held low, the inverting output buffer is enabled so that data on the Dout input is inverted and placed on the external pad. When the DISABLE input is high, the output buffer is disabled so that external devices may place data on the pad.

Internal drive capability of the input buffer is 10x standard size. External drive capability of the output buffer is one TTL load.

TSOBUFPAD

Character: 'z'
Cell size: 14 rows, 9 columns. (NMOS30)
Placement: Edges of circuit only, not allowed in corners.
Modifiers: None.

Inputs: DISABLE (right column wire of second column to right of cell origin, or third row above cell origin)
Dout (right column wire of third column to right of cell origin, or fourth row above cell origin)
Outputs: P (external pad connection)

This pad is a tri-state output buffer pad. When the DISABLE input is held low, the inverting output buffer is enabled so that data on the Dout input is inverted and placed on the external pad. When the DISABLE input is high, the output buffer is disabled so that external devices may place data on the pad.

External drive capability of the output buffer is one TTL load.

5.6. Advanced PPL Cells (rarely used)

GNDCON

Character: 'g'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: [formed by summing values below]
(1) connect right column wire to ground.
(2) connect left column wire to ground.
(4) connect main row wire to ground.
(8) connect auxiliary row wire to ground.

Inputs: (none)
Outputs: (specified by modifier value)

Connect one or more wires directly to GND. Each wire is specified by the modifier value shown above. Combinations of wires are specified by a modifier that is the sum of the values specified for each wire that is to be grounded. A modifier value of ZERO will cause none of the wires to be grounded, which is equivalent to a BLANK cell.

VDDCON

Character: '^'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: [formed by summing values below]
(1) connect right column wire to VDD.
(2) connect left column wire to VDD.
(4) connect main row wire to VDD.
(8) connect auxiliary row wire to VDD.

Inputs: (none)
Outputs: (specified by modifier value)

Connect one or more wires directly to VDD. Each wire is specified by the modifier value shown above. Combinations of wires are specified by a modifier that is the sum of the values specified for each wire that is to be connected to VDD. A modifier value of ZERO will cause none of the wires to be connected to VDD, which is equivalent to a BLANK cell.

NOTE: Avoid this cell. Pullups are preferred over direct connections to VDD. Noise immunity and/or reliability problems may result from using this cell.

LCAP

Character: '('
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Caps off the left edge of a PPL module which is to be used in a custom circuit.

RCAP

Character: ')'
Cell size: 1 row, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Caps off the right edge of a PPL module which is to be used in a custom circuit.

ROWVBUS

Character: '/'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

VDD bus wire. Hidden in pad cells, this cell may be used for internal routing of VDD within the PPL array. Also used to cap off upper edge of PPL modules that are to be used in custom circuits.

Cell to route VDD bus along a row. Also used to cap off upper edge of PPL modules that are to be used in custom circuits. Included as part of pad cells automatically.

ROWGBUS

Character: '\'
Cell size: 2 rows, 1 column.
Placement: Any row, any column.
Modifiers: None.

Inputs: (none)
Outputs: (none)

Cell to route GND bus along a row. Also used to cap off lower edge of PPL modules that are to be used in custom circuits. Included as part of pad cells automatically.

COLGBUS

Character: '{'
Cell size: 1 row, 1 column.
Placement: Any row, even column.
Modifiers: (0) "normal", matches regular PPL on both sides.
(1) matches 'custom' (or *Z*) cells on the right, matches regular PPL cell on the left.

Inputs: (none)
Outputs: (none)

Cell to route GND bus along a column. Used to isolate an 'empty' region within a PPL array for inclusion of custom circuitry or *Z* modified routing cells.

NOTE: *Z* modified cells may only be placed next to other *Z* modified cells. *Z* modified cells use metal row wires and poly column wires, and contain no VDD or GND connections. They are used for critical routing of signals that cannot tolerate long poly row wires.

COLVBUS

Character: '}'
Cell size: 1 row, 1 column.
Placement: Any row, even column.
Modifiers: (0) "normal", matches regular PPL on both sides.
(1) matches 'custom' (or *Z*) cells on the left,
matches regular PPL cell on the right.

Inputs: (none)
Outputs: (none)

Cell to route VDD bus along a row. Used to isolate an 'empty' region within a PPL array for inclusion of custom circuitry or *Z* modified routing cells.

NOTE: *Z* modified cells may only be placed next to other *Z* modified cells. *Z* modified cells use metal row wires and poly column wires, and contain no VDD or GND connections. They are used for critical routing of signals that cannot tolerate long poly row wires.