1. DESCRIPTION

1.1 INTRODUCTION TO THE ZL30 and ZL30A

By the inclusion of link connections in certain component devices and subsequent programming of those links it is possible to program a device to give a desired output in response to certain input conditions.

Links are programmed by the application of high level programming voltages (super voltages) for defined periods of time in accordance with device manufacturers' instructions.

The ZL30 and ZL30A Logic programmers are instruments designed to program logic devices and is one of the system Z range of products. It is capable of programming PAL*, IFL, EPLD and GAL* devices in 20, 24, 28 and 40 pin packages.

The programming capability of the ZL30 and ZL30A is further enhanced by the availability of optional adaptor modules. The adaptors come in a variety of socket styles to support the following devices: Leadless chips, and 40 pin devices.

1.2 GENERAL DESCRIPTION

1.2.1 Specification

The ZL30 and ZL30A supports programmable logic devices from the following manufacturers:

Altera
AMD
Cypress
Fairchild
Harris
ITC
Lattice
MMI
National
Panatech/Ricoh
Signetics
Sprague
Texas Instruments
VTI

Programmable Parts: ZL30A expansion modules

Leadless devices
40 pin packages
Small outline devices
Surface mounted devices

The Keyboard

For data entry and operating programmer functions.

0 to 9	—	Numeric data entry keys
Load	—	Load RAM from a master device
Input	—	To activate the I/O to load RAM
Output	—	To output information to a computer terminal, printer or storage medium.
Set	—	Followed by a numeral to set the programmer for special functions: format selection, manufacturer and device selection etc.
Edit	—	Set the programmer to the edit mode
Exit	—	To exit from any routine or to terminate an entry.
Dec	—	To decrement the fuse counter to the previous fuse number.
Inc	—	To increment the fuse counter to the next sequential fuse number.
Test	—	To perform a vector test on the device.
Verify	—	To perform an array verify.
Empty	—	To perform an unprogrammed check.
Program	—	Set program sequence into operation.

Display

16 alphanumeric characters operating interactively to display the sequence of events, to supply programming or test parameters and to display the results of a stimulus.

I/O Formats

JEDEC JC 42.1 1981
Signetics standard formats
PALASM HEX
PAL X-Plot

Interface

RS232C with full handshake and Xon and Xoff control. Switches on the rear panel select baud rate, stop bits and local or remote control up to 38,400 baud.

IEEE STD 488

Universal Handler Interface. This works in conjunction with the top panel connector. (See Handler section 5.2).

Sockets for programmable devices

Four zero insertion force sockets to take 20, 24 and 28 pin devices.

Plus additional sockets on optional adaptor modules.

Four LED indicators adjacent to pin 1 of each socket to indicate the appropriate ZIF to be used for a pre-selected device.

Plus additional LEDS for sockets on an optional adaptor module.

Power

Power ON/OFF switch on the rear panel

110V, 60HZ or 240V, 50HZ supply

70 watts typical power consumption

Switching mode power supply with full protection for reliability and integrity.

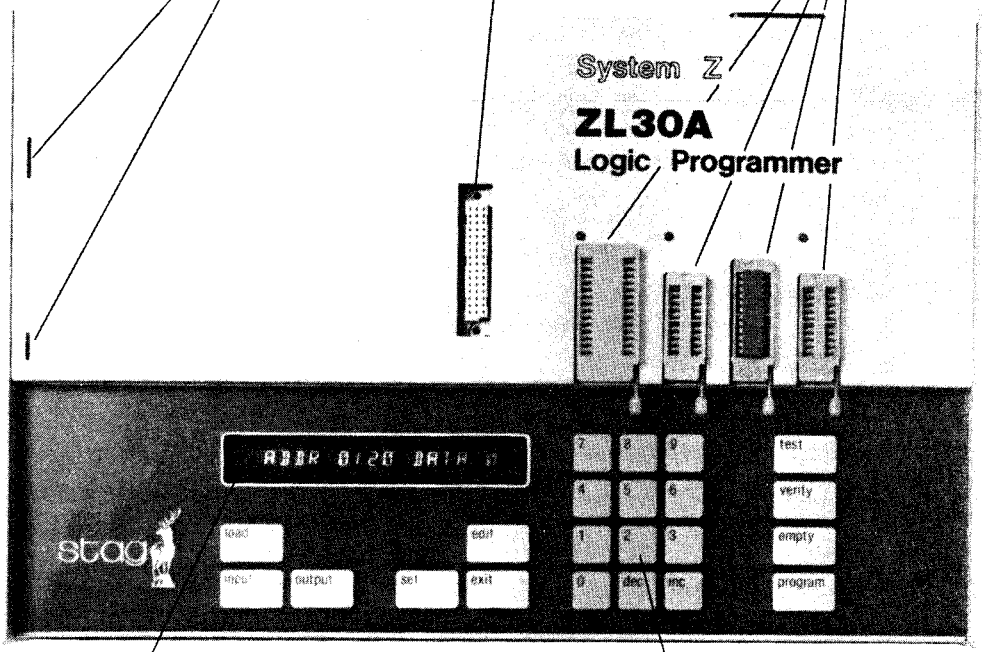
Dimensions: 355mm x 240mm x 82mm
Weight: 5Kg approx.

1.2.2 Mainframe Top Panel

Polarized socket to provide interconnection from the mainframe to the ZL30 and ZL30A modules. The socket also enhances the ZL30 and ZL30A's handler capability (See section 5.2)

Two locating notches to provide automatic module alignment.

4 Zero Insertion Force sockets for 20, 24 and 28 pin devices. (The sockets are removable to allow easy replacement of old or damaged ZIF's).



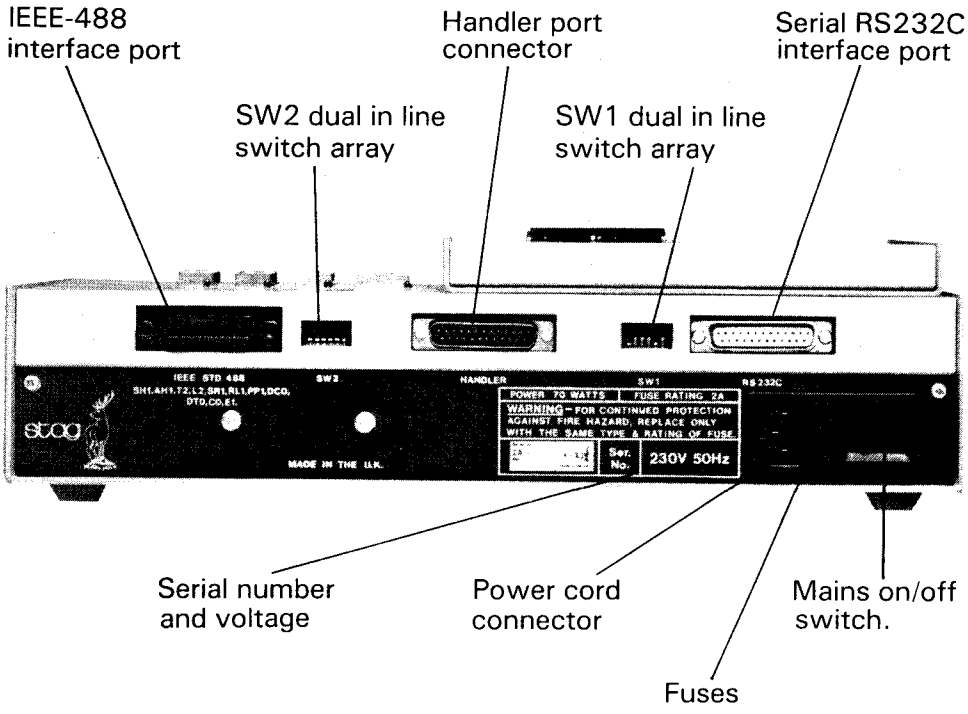
Display with 16 alphanumeric characters.

22 position touch keypad for local operation.

The display operates interactively with the programmer to indicate the sequence of events, to provide programming and test parameters and to display the result of a program cycle.

LEDs adjacent to pin 1 of each socket will illuminate to indicate which socket is to be used when the code number for a particular device has been entered.

1.2.3 **Rear Panel**



Note: Two versions of the ZL30 and ZL30A allow for a mains supply input of either 240V/50HZ or 110V/60HZ. The correct voltage application should be confirmed before use.

Interface Ports

The ZL30 and ZL30A may be used independently, under local keyboard control or can be operated remotely through one of two interfaces. These are the RS232C and the IEEE 4888 ports.

Associated with each interface is a dual in-line switch array (SW1 and SW2) which is used to configure the respective interface to the required input/output format. SW1 is used with the RS232C and SW2 is used with the IEEE-488.

It should be noted that the switches are only read by the instrument on power up so that if a change in settings is made after power is applied it will be necessary to switch the power off and then on again.

Another connector on the rear panel allows the ZL30 and ZL30A to be coupled up to an automatic handler thereby enabling devices to be programmed or tested remotely. (See Handler section).

1.3 **ZL30A expansion modules**

Device Support

The ZL30A adaptors have the capability to support advanced programmable logic devices including:

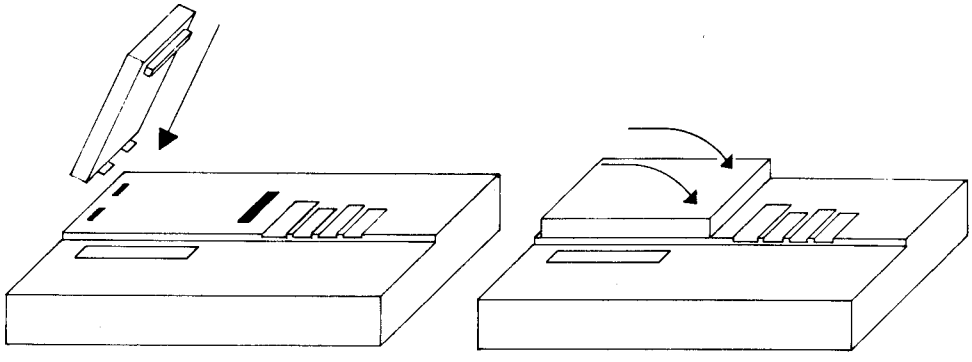
Leadless chip devices
40 pin packages

Setting-up procedure

1. Power-down the ZL30A with the on/off switch on the rear panel.

2. Insert the adaptor guide bars into the mainframe notches. Swing the adaptor down until the socket firmly engages the mainframe connector.

3. Power-up again.



When a device supported by the adaptor is selected a red LED will illuminate adjacent to the relevant socket.

Note: A second LED may illuminate on the ZL30A mainframe if the device selected for the adaptor was previously supported in a different package size.

Additional adaptor displays:

Adaptor ? This means that a pre-selected device is supported but an adaptor or the correct adaptor is required.

2. SETTING UP THE ZL30 and ZL30A

2.1 INTERFACE SWITCHES

2.1.1 Switch Settings

Before use, the ZL30 and ZL30A must be set up correctly. The two switches SW1 and SW2 on the rear panel should be set before power is applied to the instrument. Each switch has 6 poles and the function of each is given in Tables 1.1 and 1.2 (in the Tables D = down, U = up).

TABLE 1.1

Switch Settings for the RS232C Interface (SW1)

SW1/1 set to D = Local or U = Remote
SW1/2 set to D = 1 stop bit or U = 2 stop bits

SW1/3	SW1/4	SW1/5	SW1/6	BAUD RATE
D	D	D	D	45.5
D	D	D	U	50
D	D	U	D	75
D	D	U	U	110
D	U	D	D	134
D	U	D	U	150
D	U	U	D	300
D	U	U	U	600
U	D	D	D	1200
U	D	D	U	1800
U	D	U	D	2000
U	D	U	U	2400
U	U	D	D	4800
U	U	D	U	9600
U	U	U	D	19200
U	U	U	U	38400

TABLE 1.2

Switch Settings for the IEEE STD 488 Interface (SW2)

SW2/1 set to D = Local or U = Remote

SW2/2, SW2/3, SW2/4, SW2/5, SW2/6 — Set talker/listener address of the ZL30 and ZL30A on the IEE STD 488 bus. SW2/2 = MSB SW2/6 = LSB, D = Logic 0, U = Logic 1

The IEEE STD 488 interface has the following capabilities:

SH1	complete capability
AH1	complete capability
T2	basic talker + serial poll
L2	basic listener, no listen-only mode
SR1	complete capability
RL1	complete capability
PP1	parallel poll configured remotely
DC0	no device clear
DT0	no device trigger
C0	no controller capability
E1	open collector drivers

2.1.2 **Setting Up for Local Operation**

Switches SW1/1 and SW2/1 should both be set to the local (down) position.

2.1.3 **Setting Up for Remote Operation**

Either SW/1 or SW/2 should be set to the remote (up) position according to which interface standard is to be used. If the RS232C interface is being used then SW1/2 to SW1/6 must be set according to Table 1 to give the required baud rate and stop bits. If the IEEE STD 488 interface is to be used SW2/2 to SW2/6 must be set as in Table 1.2 to give the correct bus address of the ZL30 and ZL30A.

Note: Should both interface switches be set to remote the programmer will stay in local and display an error message.

When the setting up is completed, power may be applied.

3. OPERATION

3.1 LOCAL OPERATION OF THE ZL30 and ZL30A

SW1 and SW2 should be set as in paragraph 2.1.1. When the power is first switched on the ZL30 and ZL30A will go into a self test routine and display SELF TEST. At the end of this test, DEVICE CODE? will be displayed automatically.

3.1.1 Configuring the ZL30 and ZL30A

Before any programming or testing operations can be carried out with the programmer it is necessary to configure the instrument for the device type to be programmed or tested and for the correct input/output format to be selected. This configuration is carried out in the local mode by using the 'set' and numeric data entry keys on the keypad.

3.1.2 SET 0 Configure the ZL30 and ZL30A for the device type to be programmed

This function is used to configure the instrument for the device type to be programmed. A two-digit manufacturer code is obtained from Table 3.1 and a two or three digit device code is obtained from Table 3.2. Press 'set' '0'. When DEVICE CODE? is displayed key in the manufacturer code followed by the device code. These codes should appear on the display as they are entered: if they are correct the 'exit' key can be pressed and the display will show the manufacturer and device type. This configures the ZL30 and ZL30A for the device type.

Note: Should a mistake be made on keying in the codes, it is necessary only to key in the correct codes before pressing 'exit'. The second entry will overwrite the first.

TABLE 3.1

Manufacturer Codes

Code	Manufacturer
66	ALTERA 20 pin EPLD
67	ALTERA 20 pin EPLD
90	AMD 20 pin AmPAL
91	AMD 24 pin AmPAL
26	CYPRESS 20 pin EPLD
27	CYPRESS 24 pin PAL
83	Fairchild 28 pin FPLA
50	Harris 20 pin HPL
54	Harris 20 pin IFL equivalent
76	Lattice 20 pin RAL E^2PLD

Code	Manufacturer
20	MMI 20 pin PAL
21	MMI 24 pin PAL
22	MMI 20 pin BPAL
30	National 20 pin DMPAL
31	National 24 pin DMPAL
70	Panatech/Ricoh 20 pin EPLD
72	Panatech/Ricoh 20 pin EPLD
13	Signetics 28 pin IFL
14	Signetics 20 pin IFL
15	Signetics 24 pin IFL
40	Texas instruments 20 pin PAL
41	Texas instruments 24 pin PAL
60	VTI 20 pin EPLD

The above table will be subject to change in pursuit of a policy of continuous improvement.

*PAL is a registered trademark of Monolithic Memories Inc.

HPL is a registered trademark of Harris Corporation

IFL is a registered trademark of Signetics Corporation

RAL & GAL are trademarks of the Lattice Corporation.

TABLE 3.2 Device Codes

Device	Code	Type
82S100/1	00	28 PIN FPLA
82S102/3	01	28 PIN FPGA
82S104/5	02	28 PIN FPLS
82S106/7	03	28 PIN FPRP
82S150/1	04	20 PIN FPGA
82S152/3 (HPL 77153)	05	20 PIN FPLA
82S154/5	06	20 PIN FPLS
82S156/7	07	20 PIN FPLS
82S158/9	08	20 PIN FPLS
16V8	09	20 PIN GAL
18P8	10	20 PIN PLA
16RP8	11	20 PIN PLA
16RP6	12	20 PIN PLA
16RP4	13	20 PIN PLA
16P2	14	20 PIN PLA
16CP1	15	20 PIN PLA
14P4	16	20 PIN PLA
12P6	17	20 PIN PLA
10P8	18	20 PIN PLA
16RA8	19	20 PIN PLA

Device	Code	Type
10H8	20	20 PIN PLA
12H6	21	20 PIN PLA
14H4	22	20 PIN PLA
16H2	23	20 PIN PLA
16C1	24	20 PIN PLA
10L2	25	20 PIN PLA
12L6	26	20 PIN PLA
14L4	27	20 PIN PLA
16L2	28	20 PIN PLA
16L8 (HPL 77209)	29	20 PIN PLA
16R8 (HPL 77212)	30	20 PIN PLA
16R6 (HPL 77211)	31	20 PIN PLA
16R4 (HPL 77210)	32	20 PIN PLA
16X4	33	20 PIN PLA
16A4	34	20 PIN PLA
16H8 (HPL 77215)	35	20 PIN PLA
16LD8 (HPL 77317)	36	20 PIN PLA
16HD8 (HPL 77318)	37	20 PIN PLA
16P8 (HPL 77216)	38	20 PIN PLA
16LE8 (HPL 77319)	39	20 PIN PLA
16HE8 (HPL 77320)	40	20 PIN PLA
16RP8	41	20 PIN EPLD
16RP6	42	20 PIN EPLD
16RP4	43	20 PIN EPLD
16P2	44	20 PIN EPLD
10P8	45	20 PIN EPLD
14P4	46	20 PIN EPLD
12P6	47	20 PIN EPLD
16P8	48	20 PIN EPLD
EP300	49	20 PIN EPLD
12L10	50	24 PIN PLA
14L8	51	24 PIN PLA
16L6	52	24 PIN PLA
18L4	53	24 PIN PLA
20L2	54	24 PIN PLA
20C1	55	24 PIN PLA
20L8	56	24 PIN PLA
20R8	57	24 PIN PLA
20R6	58	24 PIN PLA
20R4	59	24 PIN PLA
20L10	60	24 PIN PLA
20X10	61	24 PIN PLA
20X8	62	24 PIN PLA
20X4	63	24 PIN PLA
12P10	64	24 PIN PLA
14P8	65	24 PIN PLA
16P6	66	24 PIN PLA
18P4	67	24 PIN PLA
20CP1	68	24 PIN PLA
20P2	69	24 PIN PLA
22V10	70	24 PIN PLA

The above table will be subject to change in pursuit of a policy of continuous improvement.

3.1.3 SET 1: Select data transfer formats

This function selects the data transfer format for inputs and outputs for the device under test (see Section 4, Data Transfer Formats). When power is switched on to the instrument the format adopted is JEDEC JC 42.1. If some other format is required, press 'set' '1' and then enter the code for the format required (see Table 3.3). When the correct code has been entered press 'exit' to set it into the instrument.

TABLE 3.3

Format Codes

Code	Format
0	JEDEC JC 42.1
1	SIGNETICS STANDARD (IFL ONLY)
3	PALASM HEX (20 pin ONLY)
3	FUSE PLOT (PALS ONLY)

3.1.4 SET 2 Enter the number of devices to be programmed on a handler

This function is used to set the ZL30 and ZL30A for the number of devices to be programmed and only operates when the instrument is being used with a handler. Press 'set' '2' and then enter the four digit number of devices to be programmed. When the correct number has been entered press 'exit' to set it into the instrument. If the instrument is connected to a handler, programming will stop when this number of devices has been correctly programmed. If for any reason the ZL30 and ZL30A programmer is stopped from completing its cycle (e.g. a fault on the handler), a time-out circuit (15 secs. approx.) will cause a display HANDLER TIMEOUT to be given.

3.1.5 SET 3 Program the device security fuse

Security fuses are contained in some manufacturers' devices. After programming, the security fuses are blown causing the fuse pattern to become isolated and cutting off all external access. The SET 3 function is used to instruct the programmer in the ZL30 and ZL30A that security fuses are to be blown in the automatic program cycle. To set the instrument to program security fuses press 'set' '3' followed by '1' (if the fuses are to be blown) or '0' (if the fuses are to remain intact), then press 'exit'. An instruction to blow the security fuses will mean that a default state will indicate that fuses are intact.

3.1.6 SET 4 Enter the unprogrammed state of the RAM

This function is used to set the ZL30 and ZL30A RAM contents to the unprogrammed state of the device for which the instrument is set up. To set the RAM to the unprogrammed state press 'set' '4' followed by 'exit'.

3.1.7 SET 5 Inhibit automatic in-program vector test

This function is used to inhibit the vector test in the automatic program sequence. (See Section 5.1—Vector Testing) Press 'set' '5' followed by '0' (to inhibit the in-program vector test) or '1' (to enable it), then press 'exit'.

3.1.8 SET 6 Display software version

This function will cause the ZL30 and ZL30A to display its internally stored software version. Press 'set' '6' for display. 30-42

Note: It is not necessary to press 'exit'.

3.1.9 Loading the ZL30 and ZL30A RAM

Before a device can be programmed the RAM in the ZL30 and ZL30A must first be loaded with the required fuse pattern. The RAM may be loaded in one of three ways:

(a) From a master device (i.e. a programmed device)

(b) From the keypad

(c) From a remote source through the RS232C interface.

Note: It is not possible in local operation to load from a remote source via the IEEE STD 488 interface.

(a) Loading from a Master Device

The master device should be placed in the correct socket (indicated by the associated LED). If the 'load' key is now pressed, the fuse pattern will be read from the master device and loaded into the programmer RAM. On completion of loading the display will indicate the number of blown fuses in the pattern.

Note: If the master device's security fuses have been blown it is not possible to load its fuse pattern into the RAM.

(b) Loading from the Keypad

The Programmer RAM may be loaded directly from the keypad. This is achieved using the JEDEC format fuse numbering system (see Section 4, Data Transfer Formats). To load or edit the RAM press 'edit' and then enter the 4-digit fuse number where editing is to commence. The fuse data is then entered as a 1 for a blown fuse or a 0 for an intact fuse. It is possible to move up and down the fuse numbering sequence using the 'inc' (increment) and 'dec' (decrement) keys. When editing is complete press 'exit' to return to the command mode.

(c) **Loading through the RS232C Interface**

To load the programmer RAM through the RS232C interface it is first necessary to ensure that the correct interface format is selected (see paragraph 3.1.3, SET 1) and that the interface switches are set correctly (see Section 2.1). Having set the required format, set the machine to load by pressing the 'input' key. Data can now be received by the programmer through the RS232C interface. The interface performs hardware handshaking and Xon/Xoff protocol.

Note: If the 'input' key is accidentally pressed without any interface connection, the operation can be aborted by pressing any other key.

3.1.10 Output from the ZL30 and ZL30A RAM

The contents of the programmer RAM may be output over the RS32C interface. The required interface format must be set (see paragraph 3.1.3, SET 1) and the interface switches must be set correctly (see Section 2.1) then press the OUTPUT key. The contents of the RAM will then be transmitted through the RS232C interface in the selected interface format. The RS232C interface responds to hardware handshake and Xon/Xoff protocol.

3.1.11 Device Related Keypad Functions

The remaining functions performed by the programmers are (a) TEST, (b) VERIFY, (c) EMPTY and (d) PROGRAM, which are described in the following paragraphs. All of these functions are prceded by an automatic continuity test on sockets and handlers and a reversed device check (device reversed in socket). A failed test is indicated on the display.

(a) **TEST Function**

The 'test' function tests a device by the application of test vectors (see Section 5.1—Vector Testing). Unlike the VERIFY operation this is a logical test to check the device's operation at normal operating voltages.

(b) **VERIFY Function**

This function compares the contents of the programmers RAM with the fuse pattern of a device that has been programmed. The device is placed in the socket indicated by the associated LED and the 'verify' key is pressed. The fuse pattern will be read out of the device and compared with the programmers RAM contents. If the two compare, the display will indicate PASS. If any fuse fails to agree with the RAM pattern the display will indicate VERIFY ERROR.

(c) **EMPTY Function**

This function checks that a device is unprogrammed. The device is placed in the socket indicated by the associated LED and the 'empty' key is pressed. The device will be checked and if all its fuses are intact the display will indicate PASS. If any fuse is blown the display will indicate NOT EMPTY.

(d) **PROGRAM Function**

This function initiates the automatic programming sequence. The device is first checked for illegal fuses (an illegal fuse is one where the RAM pattern requires an unblown fuse and the device has a blown one. Since fuses can only be blown and not created this is an illegal condition). The device is then programmed to the manufacturer's specification and verified against the RAM. At this point, if required, the security fuses are blown. The device is then tested by the application of test vectors loaded with the fuse pattern (JEDEC format only).

To program a device place it in the socket indicated by the associated LED and press 'program'. The automatic sequence will be executed and any failure indicated on the display.

3.2 REMOTE OPERATION OF THE ZL30 and ZL30A

3.2.1 Interface Definitions

The ZL30A programmer may be controlled remotely through either the RS232C or IEEE STD 488 interfaces. To set the machine to remote control first configure the interface to be used by means of SW1 or SW2 (see paragraph 2.1.), setting SW1/1 or SW2/1 to remote (up). Power up the machine.

Note: Should both interface switches be set to remote the programmer will stay in LOCAL and display an error message.

The programmer is now ready to be used in remote control. Command codes are given in Table 3.4.

TABLE 3.4

Remote Control Commands

Code	Command
S0 MMDD	Set the programmer for manufacturer MM, device DD
S1 I	Set the interface format to I
S2 NNNN	Set the number of devices to be programmed to NNNN
S3 0 or 1	Set the state of the security fuses
S4	Set the RAM to the unprogrammed state of device MMDD
R0	Read the manufacturer and device codes
R1	Read the interface format
R2	Read the number of devices to be operated on
R3	Read the security fuse setting
R4	Read the number of blown fuses in the RAM pattern
R6	Read the installed software revision number

TABLE 3.4 (Continued)

L	Load the programmer RAM from a device in the socket
X	Edit the programmer RAM from the remote interface
I	Load the programmer RAM from the remote interface
O	Output the RAM contents to the remote interface
E V T	Check NNNN devices to be unprogrammed Array verify NNNN devices Vector test NNNN devices
P0 P1 P2 P3 P4	Program NNNN devices with the full autosequence Program NNNN devices omitting the illegal bit check Program NNNN devices omitting the array verify Program NNNN devices without illegal bit and verify Program NNNN devices omitting vector test
H n D string	Sound horn n times Display string
Y	Read yield figures DDDD no. of devices handled NNNN no. of good parts BBBB no. of bit check failures VVVV no. of array verify failures TTTT no. of vector test failures
K	Wait for a programmer key to be pressed

(When the ZL30 or ZL30A is being controlled over the RS232C interface it will output a prompt (>) on completion of a command.)

3.2.2 Using the IEEE STD 488 Interface

(a) Status Byte

The status byte is the byte output by the programmer when a serial poll is conducted by the IEEE bus controller. Bit 7 of the byte, when set, indicates that the programmer is busy and cannot accept commands. While bit 7 is set the programmer will respond only to serial and parallel poll commands. If the programmer is addressed as a listener while bit 7 is set it will assert NRFD (NOT READY FOR

DATA) and lock up the bus until the current command is completed. If it is addressed as a talker it will not respond until the current command is completed. Bit 6 is the SRQ bit and if set indicates that the programmer is generating a service request. Bits 5 & 4 provide a two-bit code which, if non-zero, indicates the type of data the programmer has ready for transmission. The codes are as follows:

0000	—	no data
0001	—	read command data response
0010	—	RAM data
0011	—	yield figures

Bits 3 to 0 provide a four-bit code supplying error information. The codes are as follows:

0000	—	no error
0001	—	no blow
0010	—	verify error
0011	—	illegal fuse
0100	—	not empty
0101	—	connect error
0110	—	reversed part
0111	—	test array failure
1000	—	address out of range (JEDEC format only)
1010	—	illegal or unrecognised command
1011	—	error detected on input
1100	—	consecutive failures have occurred on the handler
1101	—	RAM fail
1110	—	Vector test fail
1111	—	Handler time-out

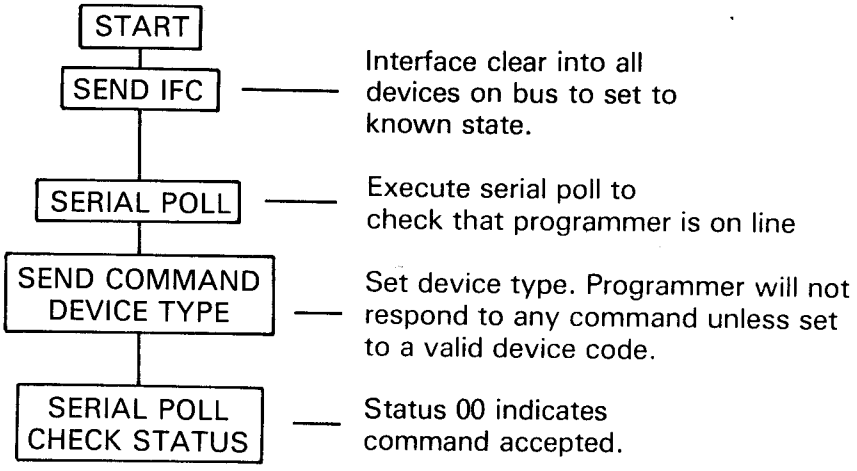
(b) **Service Request (SRQ)**

The programmer will generate an SRQ:

(i) When a command has been received which requires output onto the bus and the data is ready.

(ii) On completion of E, V, T & P commands (See Table 3.4).

(c) A typical flow chart for IEEE STD 488 remote control is:



In general, before sending a command, a serial poll should be executed to ensure that the programmer is able to accept a command, i.e. is not busy and does not have data ready to output onto the bus.

3.2.3 Using the RS232C Interface

(a) Command codes for the programmer are given in Table 3.4. On completion of a command the programmer responds by issuing a status code followed by a prompt (>). This complete response consists of 15 characters, viz:

CR, LF, null, null, null, null, S, S, CR, LF, null, null, null, null, — where S, S is the status code. The status codes are:

Code (Hex)	Status
00	Command Executed O.K.
01	No Blow
02	Verify Error
03	Illegal Fuse
04	Not Empty
05	Connect Error
06	Reversed Device
07	Test Array Failure
08	Out-of-Range Address (JEDEC Format only)
09	Correct adaptor not fitted
0A	Illegal or Unrecognised Command
0B	Load Error
0C	Consecutive Failures on Handle
0D	RAM Fail
0E	Vector Test Failure
0F	Handler Time Out

(b) When using the input command I, it is necessary to introduce a delay between transmission of the command and the start of the data to be sent. This is because, on receipt of the command I, the programmer clears its RAM and input buffer to eliminate extraneous characters. The recommended delay is 0.5 sec. RS232C Interface Connections are shown in Table 3.5.

TABLE 3.5

RS232C Interface Connections

Pin	Name	Function
1	GND.	SAFETY GROUND
2	TxD	SEND DATA. ZL30 and ZL30A SENDS DATA ON THIS PIN
3	RxD	RECEIVE DATA. ZL30 and ZL30A RECEIVES DATA ON THIS PIN
4	RTS	DRIVEN HIGH WHEN ZL30 and AL30A WANTS TO TRANSMIT
5	CTS	MUST BE HIGH OR OPEN TO PERMIT ZL30 and ZL30A TO SEND
6	DSR	IGNORED BY ZL30 and ZL30A
7	SIG.GND.	SIGNAL GROUND (S.G.)
8	DCD	MUST BE HIGH FOR THE ZL30 and ZL30A TO ACCEPT DATA
20	DTR	HELD HIGH BY ZL30 and ZL30A DROPPED TO INHIBIT DATA TRANSFER

TYPICAL CONFIGURATION

ZL30 and ZL30A			TERMINAL
Name	**Pin**	**Pin**	**Name**
S.G.	7 ———	7	S.G.
TxD	2 ———	3	RxD
RxD	3 ———	2	TxD
RTS	4 ———	8	DCD
DCD	8 ———	4	RTS
CTS	5 ———	20	DTR
DTR	20 ———	5	CTS
		6	DSR

Note: For systems using Xon/Xoff protocol a three-wire connections is sufficient. Resistors in the ZL30 and ZL30A will hold handshake lines in the correct polarity.

(c) **Editing**

When using the remote editor the terminal must be set to full duplex. To invoke the editor, key in X CR on remote keyboard. The terminal will display ENTER TERM NUMBER. Enter a term number. The terminal will display the current RAM contents and place an edit line underneath. RAM data may then be altered by entering the appropriate characters.

The cursor may be advanced using the CTRL L key and reversed by using the backspace key. The RETURN key may be used to complete entry of a term if the un-entered portion of the term is unchanged. To exit the edit mode type Q when the terminal asks for a term number.

Term numbers are numeric for PAL, with the exception that polarity is P

For IFL, term numbers are alphanumeric. In addition P (polarity) is used for active level and FF is used for flip-flop type.

4. DATA TRANSFER FORMATS

4.1 JEDEC JC 42.1 1981

The JEDEC format is a data transmission format for the transfer of information between a data preparation system and a device programmer and is independent of device architecture. The information is divided into five categories:

1. The design specification identifier.
2. The device to be programmed.
3. Fusing information for the implementation of the design specification.
4. Test vectors for structured functional testing.
5. Additional information.

4.1.1 **The Design Specification Identifier** (see para 4.1.7)

The design specification identifier consists of:

(i) ASCII STX (Start transmission).

(ii) User's name and company.

(iii) The design spec date, part number and revision.

(iv) The part number of the manufacturer's device.

(v) Other information.

(vi) An asterisk.

4.1.2 **The Device to be Programmed**

A field is defined to specify the device to be programmed. The device is specified by a four-digit code. The code is preceded by a D and terminated with an asterisk.

4.1.3 **Fusing Information** (see para 4.1.7)

Each fuse in a device is allocated a decimal fuse number. The state of each fuse is indicated by a '0' for an intact fuse and a '1' for a blown fuse. Fuse information is presented in three fields (F, L and C). The F field specifies the default state for fuses not otherwise defined. The L field specifies the states of the fuses. The C field is a checksum.

The F field must appear before the other fields. The F field commences with an F followed by 0 or 1 and terminates with an asterisk.

The L field defines the states of specified fuses in the device. It commences with an L followed immediately by decimal characters indicating the starting fuse number for a string of data. The first 0 or 1 is preceded by a space and the string is terminated with an asterisk. The data string may be of any convenient length and more than one L field may be used to specify a device. If a fuse is specified more than once the latest information will be used.

The C field is a checksum field. It commences with a C which is followed by four hex. Characters representing the checksum and is terminated with an asterisk. The checksum is calculated by the 16-bit addition of 8-bit words constructed from the specified state of each fuse in the device, bit 0 of word 0 being fuse number 0, bit 1 being fuse number 2 etc. The word containing the last fuse of the device is constructed by setting to zero all bit locations corresponding to fuse numbers higher than the last fuse in the device.

4.1.4 **Structured Test Information**

A field is defined to allow the loading of test vectors for functional testing of the device. A test vector defines a combination of input and output states for a device. For detailed information on test vectors see Section 5.1. A test vector must be preceded by a V. This is followed immediately by decimal characters representing the number of the test vector. This number is followed by a space and then the test variables. Each test vector is terminated by an asterisk. Vectors are applied in numerical order. If a test vector number appears more than once the latest test vector will be used.

4.1.5 **Additional Information**

Additional fields may be defined in the future using other letters.

4.1.6 **Ending the Format**

The transmission must be terminated by ETX (End Transmission). A checksum of four hex. characters must follow immediately. This checksum is the 16-bit sum of the ASCII values of the transmitted characters between, and including, the STX and ETX. The parity bit must be excluded during this calculation.

4.1.7 **Example**

Design Specification	STX Peter Collins, STAG, 22 02 83,	user's name user's company design spec. date
Identifier	DS123-2345 PN SIG. IFL 82S153*	design spec. part number device manufacturer's part no.
	D1405*	device to be programmed
Fusing Information	F0* L0000 00110101001010* L0014 0111100010*	fusing default state states of fuses
	CABFD*	fuse checksum
	V0001 101010101 XHLHLHLHLHX*	test vector
	ETX12FE	transmission checksum

4.2 SIGNETICS IFL FORMATS

4.2.1 82S100 FPLA

A typical format for the 82S100 is as shown below:

```
                                    * AL HL HL HL H
*P 00 *IHHHHHHHHHHHHHHHH * F. A. A. A. A
*P 01 *IHHHHHHHHHHHHHHHH * F. A. A. A. A
*P 02 *IHHHHHHHHHHHHHHHH * F. A. A. A. A
```

The data field starts with STX (^B). This is followed by the active level identifier *A and then the states of the active levels F7 to F0. The P-terms are then entered by first entering the P-term identifier *P followed by a space and then the P-term number. The input variable identifier *I is then followed by the input variables I15 to I0 and then the output function *F followed by F7 to F0. The data field is terminated by ETX (^C). It is not necessary to enter unused P-terms, these will automatically be set to the unprogrammed state.

4.2.2 82S102 FPGA

A typical format for the 82S102 is shown below:

```
STX
*G 00 *AL * I HHHHHHHHH - HHHHHH
*G 01 *AH * I LLLLLLLLLLLLLLLL
*G 02 *AL * I HHHHHHHHHHHHHHHH
ETX
```

The data field commences with STX followed by the term identifier *G followed by the term number. This is followed by the active level identifier *A and the active level data. The input variable identifier *I is then followed by the input variable data I15 to I0. The data field is terminated with ETX. It is not necessary to enter data for unused gates.

4.2.3 82S104 FPLS

A typical format for the 82S104 is shown below:

```
STX
*AH
*T 00 *C. * I HHHHHHHHH - HHHHHH* P HHHHHH* NHHHHHH* F HHHHHHHH
*T 01 *C. * I HHHHHHHHHHHHHHHH* P HHHHHH* NHHHHHH* F HHH- HHHH
*T 02 *C. * I HHHHHHHHHHHHHHHH* P HHHHHH* NHHHHHH* F HHHHHHHH
ETX
```

The data field is started with STX. The preset/enable option is then entered using the identifier *A followed by H or L for the required state. The transition terms are then entered by first selecting the term using the term identifier *T followed by the term number. The input variables for the term are entered with the input variable identifier *I followed by I15 to I0. The present state of the flip-flops is then entered with *P and P5 and P0 followed by the next state *N and N5 to N0. The output function is then input as *F followed by F7 to F0. The data field is terminated with ETX.

4.2.4 **82S106**

The format for the 82S106 is similar to that for the 82S100 FPLA. It differs in that, since the active level is fixed, the entry for the active level must always be *AHHHHHHHH. A typical format for the 82S106 is shown below:

```
STX(B)
* AHHHHHHHH
*P 00 * I HHHHHHHHH - HHHHHH * F . A. A. A. A
*P 01 * I HHHHHHHHHHHHHHHH * F . A. A. A. A
*P 02 * I HHHHHHHHHHHHHHHH * F . A. A. A. A
ETX(C)
```

4.2.5 **82S150/151**

A typical format is shown below:

```
STX
*POL.HHHHHHHHHHHH
*E.............
*P 00 *I--------  *BI----------------
*P 01 *I--------  *BI------------0000
*P 02 *I000000    *BI0000000000000
*P 03 *I000000    *BI0000----------
*P 04 *I--------  *BI----------------
*P 05 *I--------  *BI0000000000000
*P 06 *I000000    *BI0000000000000
*P 07 *I------00  *BI----------------
*P 08 *I--------  *BI--------------00
*P 09 *I000000    *BI0000000000000
*P 10 *I000000    *BI000000--------
*P 11 *I--------  *BI----------------
*P D2 *I--0000    *BI----------------
*P D1 *I000000    *BI0000000000000
*P D0 *I--------  *BI--0000000000
ETX
```

The start of the data field is marked with STX. The output polarity is then entered starting with the identifier *POL followed by the required states of the outputs B11 to B0.
The information for the enables is then entered as *E followed by the required states A = I/O or · = Input.
The P-Terms are then entered using the P-Term identifier *P followed by the P-Term number. Note that the control term numbers start with D. The input variables are then entered using the identifier *I followed by data for I7 to I0. The I/O variables are then entered with the identifier *BI and data B11 to B0.
The field is terminated with ETX.

4.2.6 **82S152/153**

A typical format for the 82S152/153 is as shown below:

```
STX
*POLHLLLHLLLLH
*P 00 *I - - - - HH- -  *BI - - - - - - - - - -  *BO. . . A. . . . . .
*P 01 *I - - - - HL- -  *BI - - - - - - - - - -  *BO. . A. . . . . . .
*P 02 *I - - - - LL- -  *BI - - - - - - - - - -  *BO. A. . . . . . . .
*P D1 *I - - - - - - - -  *BI - - - - - - - - - -
*P D0 *I - - - - - - - -  *BI - - - - - - - - - -
ETX
```

The start of the data field is marked with STX. The output polarity is then entered starting with the identifier *POL followed by the required states of the outputs B9 to B0. The P-terms are then entered using the P-term identifier *P followed by the P-term number. Note that the control term numbers start with D. The input variables are then entered using the identifier *I followed by the data for I7 to I0. The I/O variables are then entered with the identifier *BI and data B9 to B0. The output functions are then entered identified by *BO and data B9 to B0. The field is terminated with ETX. Note that the control terms have no output function.

A typical JEDEC format for the 82S152/153 is shown below:

```
*D1405
*G0
*L0000 11110101111111111111111111111111111111110111111
*L0046 11111001111111111111111111111111111111101111111
*L1796 00000000000000000000000000000000000000
*L1832 0111011110
*C71D7
*A040
```

4.2.7 **82S154/155/156/157**

A typical format for this device is shown below:

```
STZ
*E-- *F/F.... *POLHHHHHHHHH
*T 00 *C- *I----- *BI---------- *OP----- *ON----- *P.. *R.. *BO.......
*T 01 *C0 *I----- *BI0000000000 *OP000- *ON0000 *PAA *RAA *BOAAAAAAAAA
*T 02 *C- *I0000 *BI--------00 *OP0000 *ON----- *P.. *R.. *BO.......
*T FC *C- *I----- *BI---------- *OP-----
*T LB *C- *I0000 *BI-----H000 *OP0000
*T LA *C. *I0000 *BI0000000000 *OP0000
*T D7 *C. *I----- *BI000L----- *OP-----
*T D6 *C- *I----- *BI---------- *OP-----
*T D5 *C- *I0000 *BI--H000000 *OP0000
*T D4 *C. *I0000 *BI0000000000 *OP0000
*T D3 *C. *I----- *BI0L-------- *OP-----
*T D2 *C- *I----- *BI---------- *OP-----
*T D1 *C- *I0000 *BIH0000000 *OP0000
*T D0 *C. *I0000 *BI0000000000 *OP0000
ETX
```

The start of the data field is marked with STX. The information for the enables is then entered as *E followed by the required states O = idle, A = control, • = enable or – = disable. This is followed by the flip-flop type information and the output polarities.
The transition terms are then entered as *T and the T-Term number followed by the complement *C and the input data *I, I3 to I0.
The I/O variables are then entered using the identifier *BI and data B7 to B0 followed by the present state *Q and the next state *QN.
The field is terminated with ETX.

4.2.8 **82S158/ 159**

A typical format for the 82S158/159 is as shown below.

```
STX  *EAA *F/FA.A.A.A.*POLLHLH
*T 00 *C.   *I HHHH * BI HHHH * QP HHHHH- HH * QNHHHHHHHH * BOA. A.
*T 01 *C.   *I HHHH * BI HHHH * QP HHHHHHHH * QNHHHHHHHH * BOA. A.
*T 02 *C-   *I HHHH * BI HHHH * QP HHHHHHHH * QNHHHHHHHH * BOA. A.
*T FC *C.   *I HH-H * BI HHHH * QP HHHHHHHH
*T PB *C-   *I LLLL * BI LLLL * QP LLLLLLLL
*T RB *C.   *I HHHH * BI HHHH * QP HHHHHHHH
*T LB *C-   *I LLLL * BI LLLL * QP LLLLLLLL
*T PA *C-   *I LLLL * BI LLLL * QP LLLLLLLL
*T RA *C.   *I -HHH * BI HHHH * QP HHHHHHHH
*T LA *C.   *I HHHH * BI HHHH * QP HHHHHHHH
*T D3 *C-   *I LLLL * BI LLLL * QP LLLLLLLL
*T D2 *C.   *I HHHH * BI HHHH * QP HHHHHH-H
*T D1 *C-   *I LLLL * BI LLLL * QP LLLLLLLL
*T D0 *C.   *I HHHH * BI HHHH * QP HHHHHHHH
EXT
```

The data field is started with STX. The information for the enables is then entered as *E and then EB and EA. This is followed by the flip-flop type information and the output polarities. The transition terms are then entered as *T and the T-term number followed by the input data, the I/O data, the present state and the next state. The field is terminated with ETX.

4.3 **PALASM HEX FORMAT**

The PALASM hex format is the output of the PALASM Boolean fuse assembler for 20 pin PLAs. The ZL30 can accept this output directly.

4.4 **FUSE-PLOT**

The fuse-plot format is provided to give a readable format for 20 and 24 pin PLAs. It follows closely the logic diagram of these circuits.

A typical X—PLOT is shown below:

```
P NO   0123 4567 8901 2345 6789 0123 4567 8901
00     .... .... .... .... .... .... .... ....
01     .... .... .... .... .... .... .... ....
02     .... .... .... .... .... .... .... ....
03     .... .... .... .... .... .... .... ....
04     .... .... .... .... .... .... .... ....
05     .... .... .... .... .... .... .... ....
06     .... .... .... .... .... .... .... ....
07     .... .... .... .... .... .... .... ....
32     XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
33     ---- ---- -X-- ---- ---- ---- -X-- ----
34     XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
35     ---- ---- ---- ---- ---- ---- ---- ----
36     XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
37     ---- ---- ---- ---- ---- ---- ---- ----
38     XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
39     ---- ---- ---- ---- ---- ---- ---- ----
```

20 pin PLA

The fuse matrix is described as a number of P-terms. Each P-term has a two-digit decimal number followed by 32 characters representing the required conditions of the 16 input lines. The characters are X for an intact fuse, - for a blow fuse and . for a "phantom fuse". A typical X-plot input is shown below:

```
STX
00 X - X - - - - - - - X - X - X - X - - - - - - X - - - - - X X -
ETX
```

The data field is started with STX (B̂). This is followed by the P-term number and the fusing data for that P-term. The data field is terminated with ETX (Ĉ). It is not necessary to provide data for unused terms.

24 pin PLA

The format for the 24 pin devices is similar to that for the 20 pin devices. The main difference is the number of fuses in the P-term, this being 40 for the 24 pin devices.

5. MISCELLANEOUS INFORMATION

5.1 VECTOR TESTING

(a) WHY TEST PROGRAMMABLE LOGIC?

The fuses in programmable logic devices are programmed and read by the application of supervoltages (voltages greater than TTL levels) to the device pins. Also, because of the way the fuses are addressed and read, not all of the device circuitry is used during the verification process. For these reasons it is possible to have a PLD (Programmable Logic Device) which contains the desired fuse pattern but which does not function correctly at operational voltages. It is desirable therefore to perform logic tests on a device to ensure its correct operation.

(b) WHAT IS VECTOR TESTING?

Vector testing is a method of testing the logic function of a device. A sequence of test vectors is applied to the device and its pins monitored for the expected states. A test vector defines voltages to be applied to device pins and the expected conditions of other pins. A typical test vector is shown below.

V0001 C1011110011NHLHLZZHHLHLN

V0001 is the test vector number. The rest of the vector describes input and output conditions for the test, each character representing a device pin. The first character is for pin 1, the second for pin 2 etc. Characters are interpreted as follows:

0	—	drive pin low
1	—	drive pin high
2	—	drive pin to super voltage 1
3-9	—	drive pin to super voltage 2-8
C	—	drive pin low, high, low
K	—	drive pin high, low, high
N	—	power pin
L	—	test pin low
H	—	test pin high
Z	—	test pin for high impedance
X	—	don't care
F	—	float pin

In the example shown the test vector is for a 24-pin PAL and so pins 12 and 24 have N characters as these are the power pins.

Test vectors are applied in numerical sequence and steady levels are applied before the C and K clock tests.

(c) WHY VECTOR TEST?

With combinational logic such as the Signetics 82S100 where the input and output pins are defined by the device architecture, it is simple to perform a verification. If the tester applies all 64K combinations of the 16 input pins and compares the 8 output pins with states computed from the fuse pattern, the device will be tested. This is called Logic Verification and takes typically 6 minutes. Many of the 64K input combinations will be providing redundant tests. By applying test vectors which are relevant to the device's function the number of tests may be reduced thus reducing testing time with no loss of integrity of the test procedure.

Most PALs and the 20-pin IFL devices have pins which may be inputs or outputs depending on the fuse pattern programmed into the device. It is impossible for a standard automatic test procedure to cope with this as a pin's function is dependent on the state of other pins, e.g. I/O pins and tristate enables. This results in inconsistent results when logic verification is attempted.

With registered PALs and FPLSs the sequence in which inputs are applied becomes important as the internal registers must be tested for correct sequencing. This is impossible to test automatically (e.g. by applying pseudo-random sequences to the device) since if, for instance, a RESET line is low then the registers will not be sequenced. To make matters even worse in an FPLS the flip-flop type is controlled by a P-term and can change from J-K to D-type or vice-versa depending on input conditions and the states of the internal flip-flops.

It is therefore essential to test a logic device in a manner tailored to its function. The best way of achieving this is to apply test vectors which are generated with consideration of the logical function of the device.

Vector testing can reduce testing time and properly test all types of programmable logic devices. It should be noted that vector testing is not confined to programmable logic devices but is a technique which can be applied to all logic devices.

5.2 HANDLERS

A handler is an electro-mechanical device which is loaded with parts to be tested and sorted). The ZL30 and ZL30A may be operated with most common handlers. Coded cables indicate to the programmer the type of handler in use, the correct drive signals and sorts for the handler are then selected automatically. When used with a handler the ZL30 and ZL30A will abort the operation should three consecutive device failures occur and an error message CONSECUTIVE FAIL will be displayed.

Handler	Control Cable	Sort Categories
Trigon 2010	20-1001	Left channel: connect error reversed part, bit fail. Centre channel: good devices Right channel: no blow, verify fail, test fail.
Exatron 800B	20-1003	1 pass 2 fail.
IEEE STD P849	20-1004	1 connect error, reversed part, bit fail 2 good devices 3 no blow, verify fail, test fail.
MCT 2608/4	20-1005	
EMS	20-1006	
DATMARC	20-1007	

5.3 ACCESSORIES

The following accessories are currently available for use with the ZL30A:

Part Number	Description
20-1000	IEEE STD 488 interface cable
20-1001	Trigon 2010 control cable
20-1003	Exatron 800B control cable
20-1004	IEEE STD P849 control cable
20-1005	MCT 2608 control cable

5.4 JEDEC FUSE NUMBERING—Index

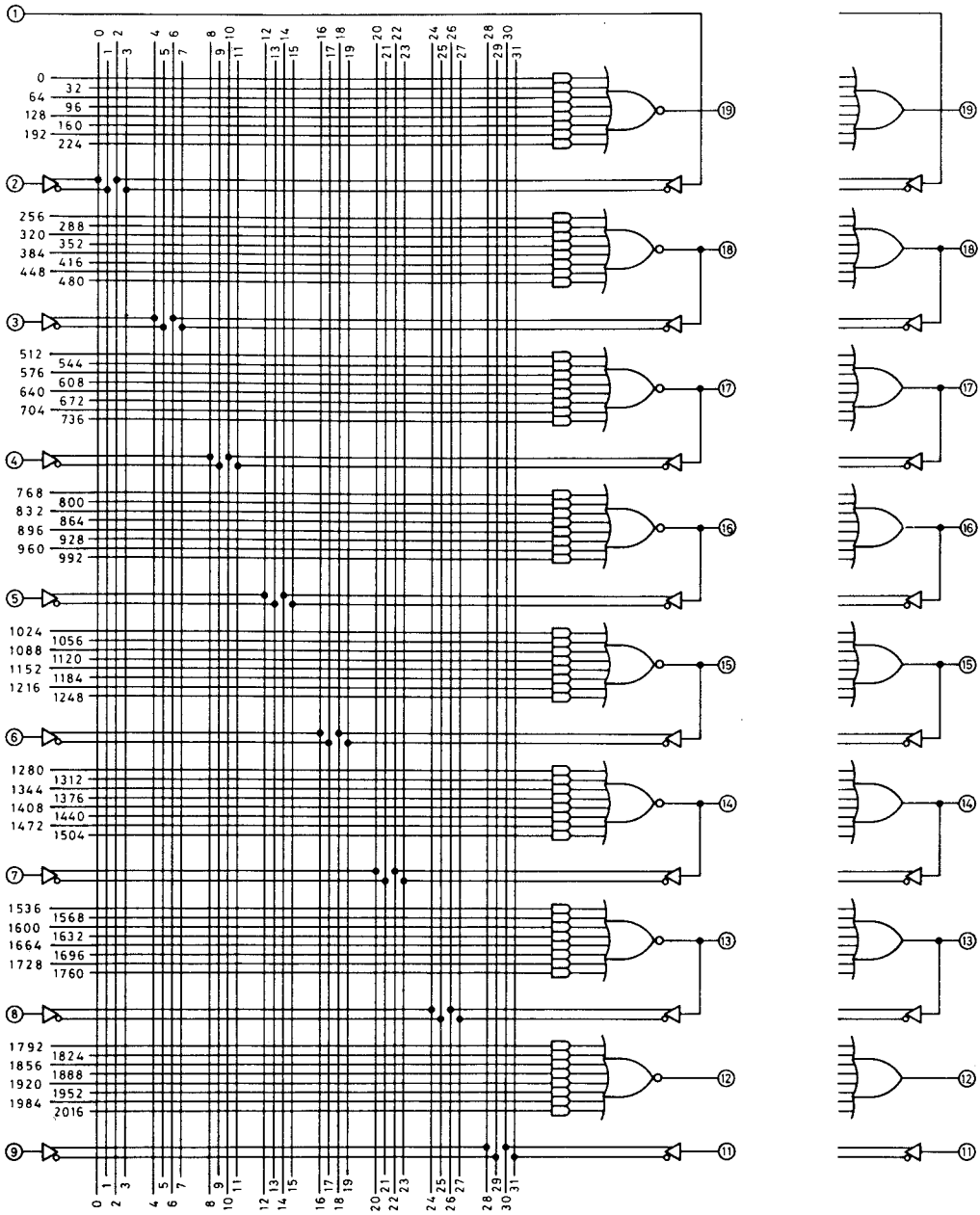
77209(16L8)	
77215(16H8)	5.52
77216(16P8)	
22V10	5.53

The above table will be subject to change in pursuit of a policy of continuous improvement.

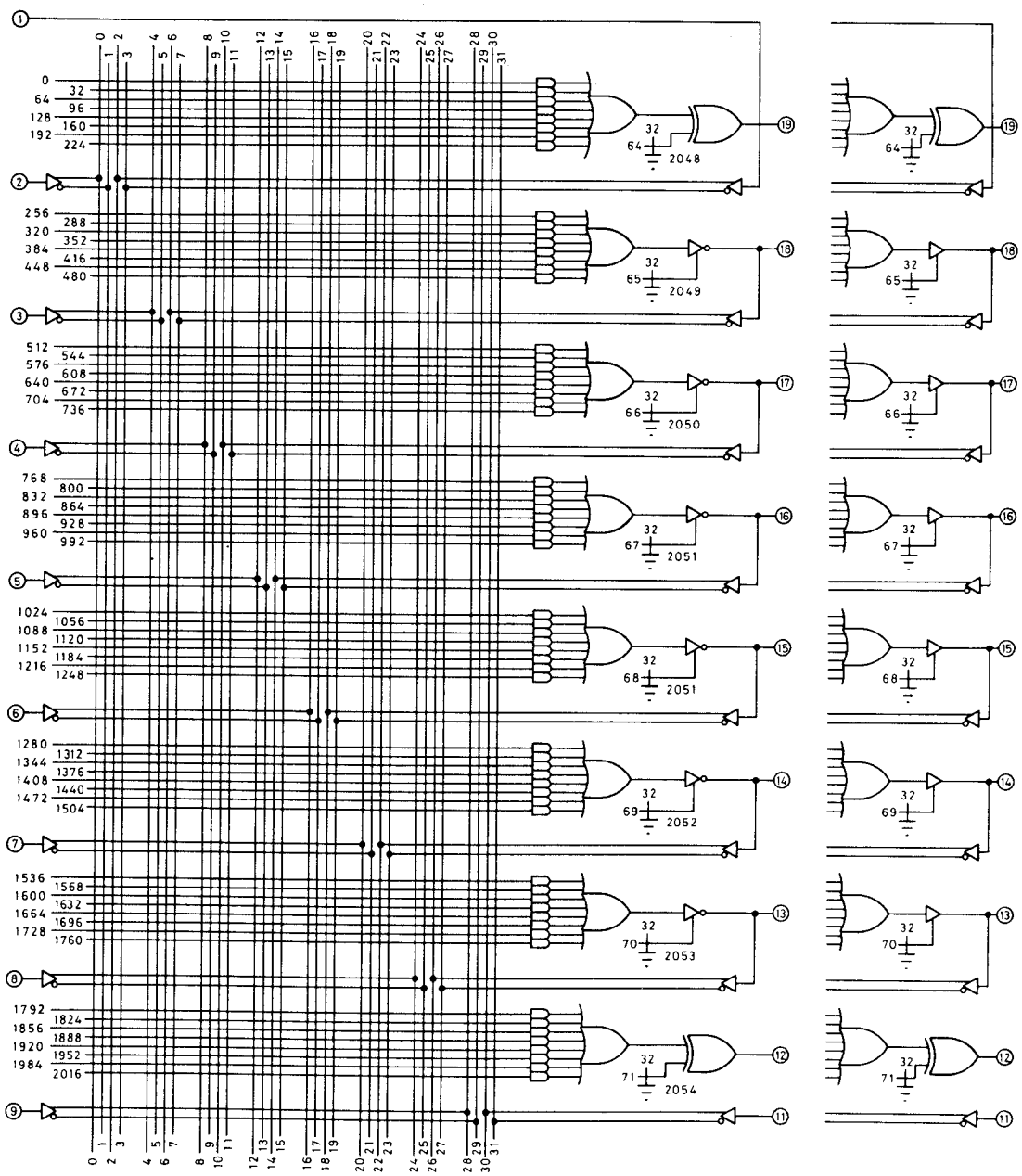
5.5 LIST OF POSSIBLE DISPLAYS

The top panel display can indicate the following:

ADAPTOR?
BIT FAIL
CHECKING
CLEAR MEMORY?
CONNECT ERROR
CONSECUTIVE FAIL
DEVICE CODE
DUMPING
END OF RUN
FORMAT CODE
HANDLER TIMEOUT
ILLEGAL CODE
LOADING
LOAD ERROR
"MIF" "Type"
NO BLOW
NO DEVICES 0001
NO VECTORS
NOT EMPTY
0001 BLOWN FUSES
PASS
PROGRAMMING
RAM FAIL
REMOTE
REVERSED DEVICE
SECURITY BLOW
SECURITY INTACT
SELF TEST
SELF TEST FAIL
SET
TEST ARRAY FAIL
TESTING
VECTOR TEST FAIL
VECTOR TEST OFF
VECTOR TEST ON
VERIFY
VERIFY ERROR
VERSION

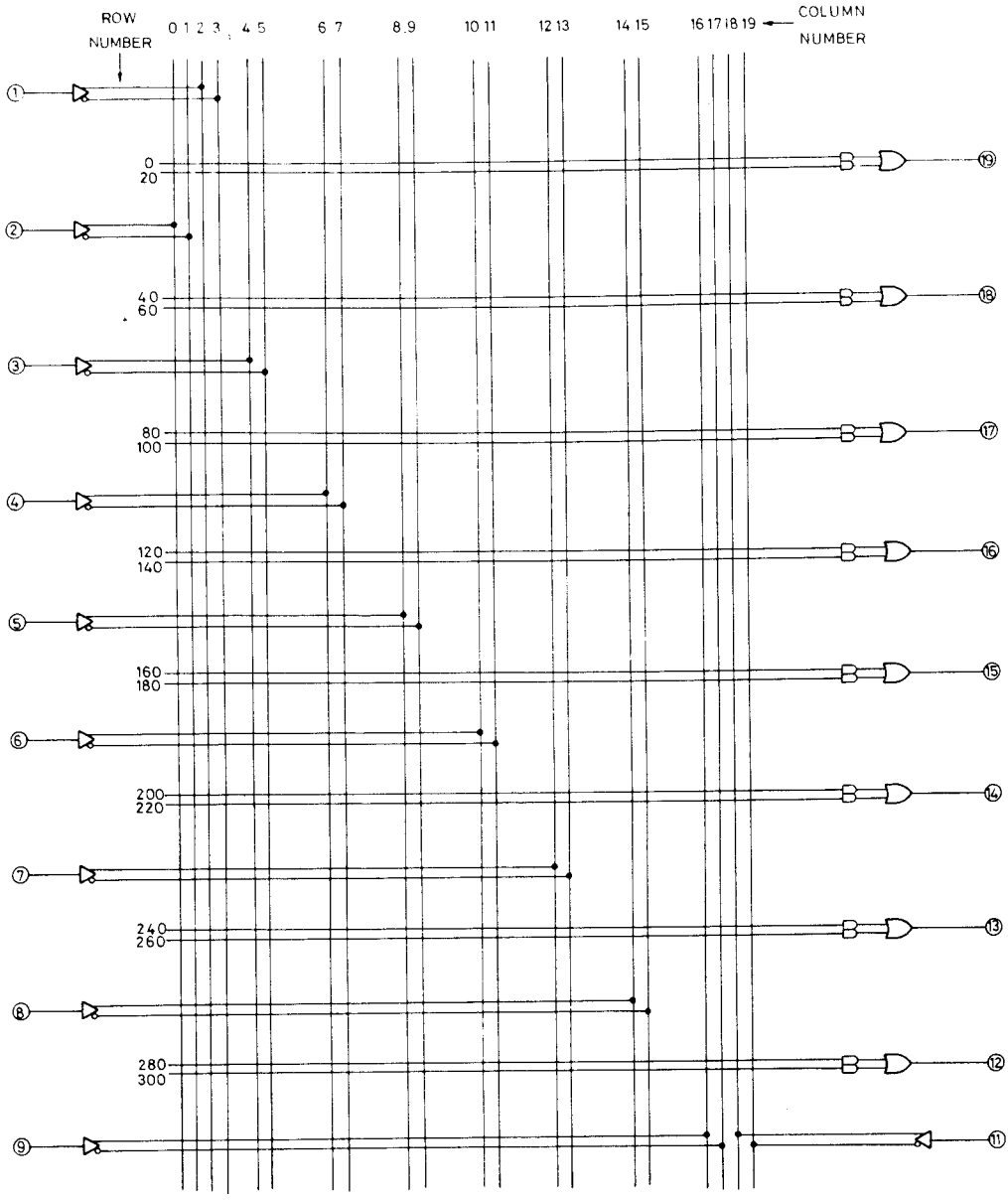


FUSE No. = ROW No. + COLUMN No.



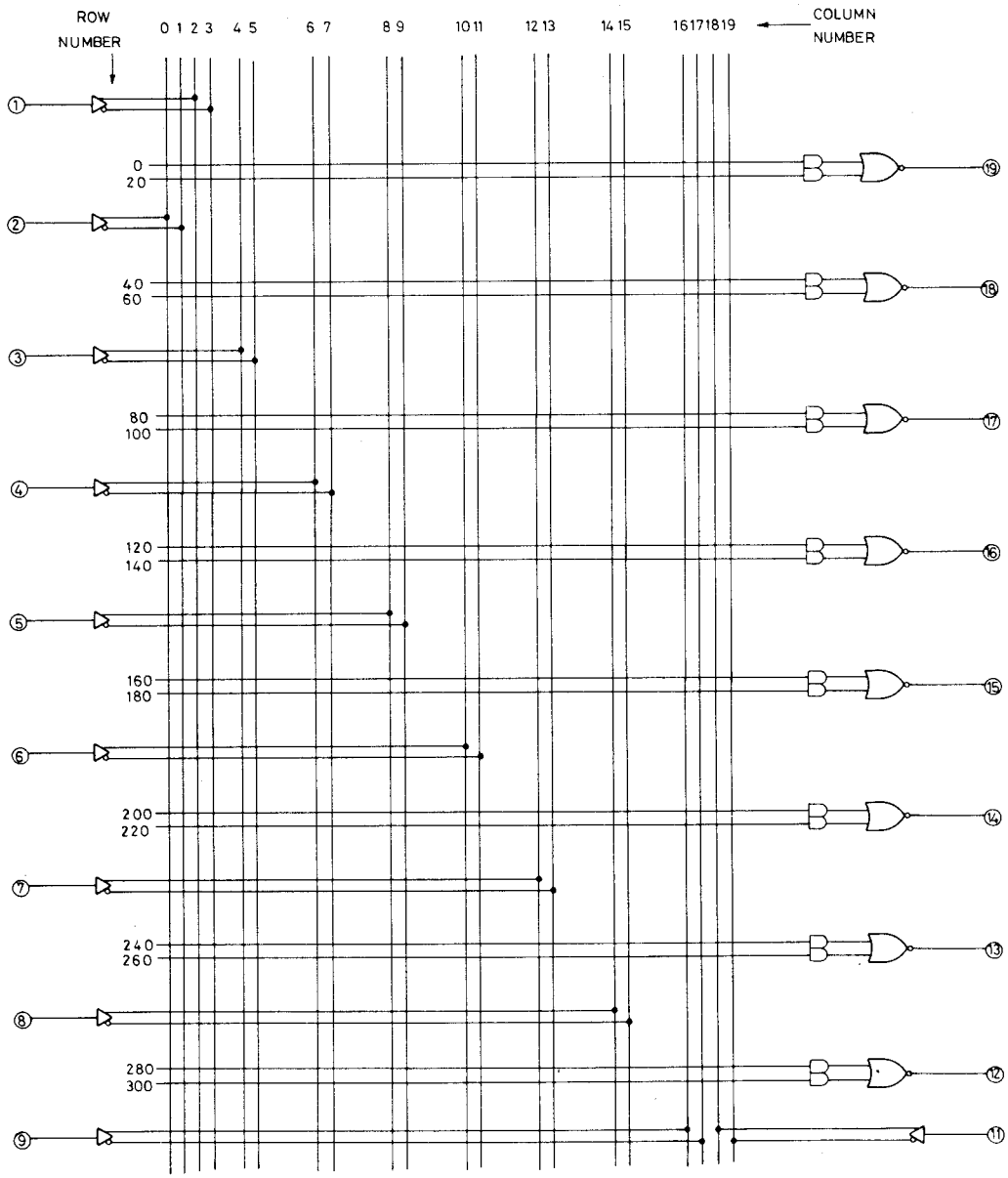
FUSE No = ROW No • COLUMN No

FUSE NUMBER DIAGRAM PLA 10H8



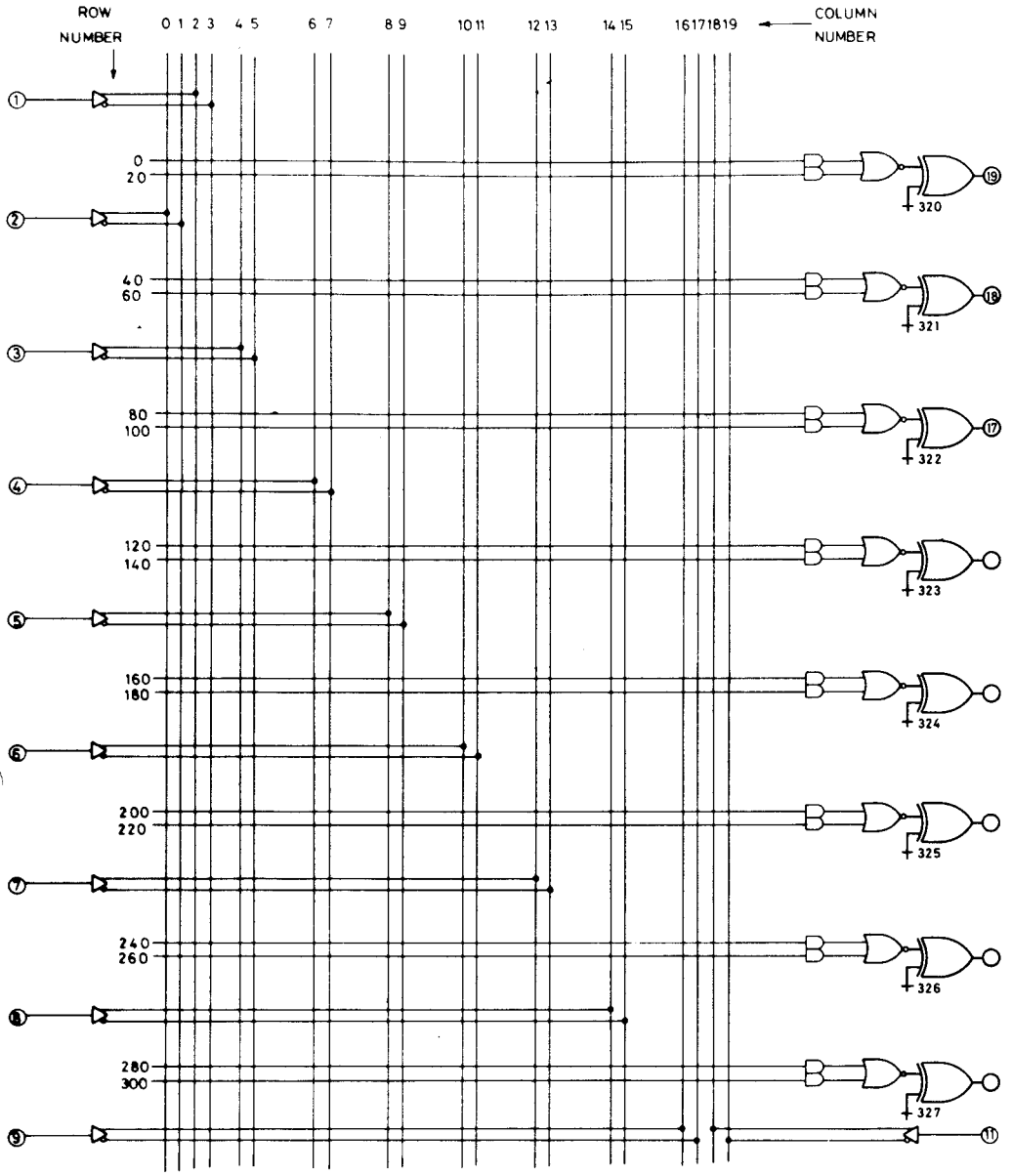
FUSE No. = COLUMN No. • ROW No.

FUSE NUMBER DIAGRAM PLA 10L8



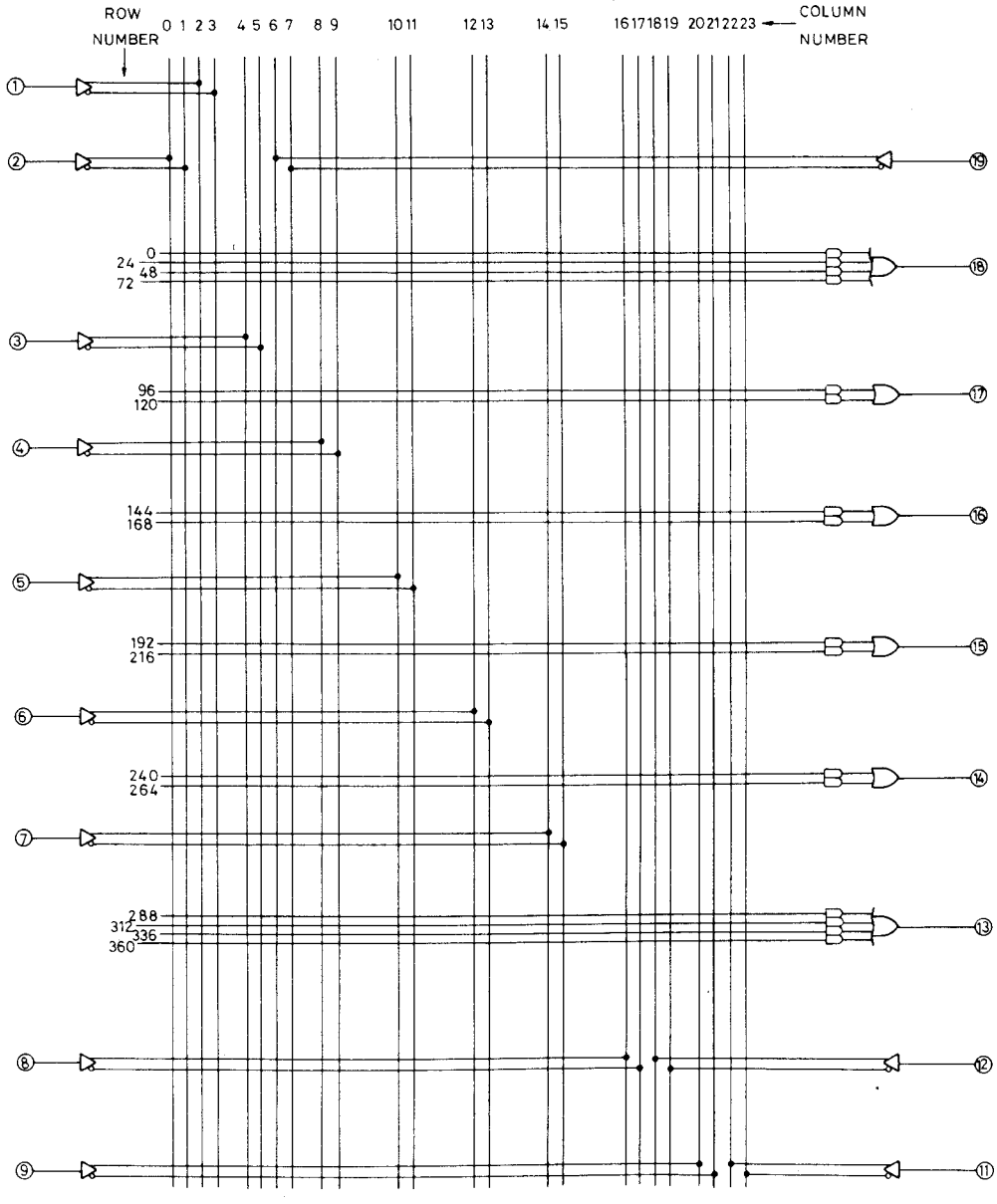
FUSE No. = COLUMN No. • ROW No.

FUSE NUMBER DIAGRAM PLA 10P8



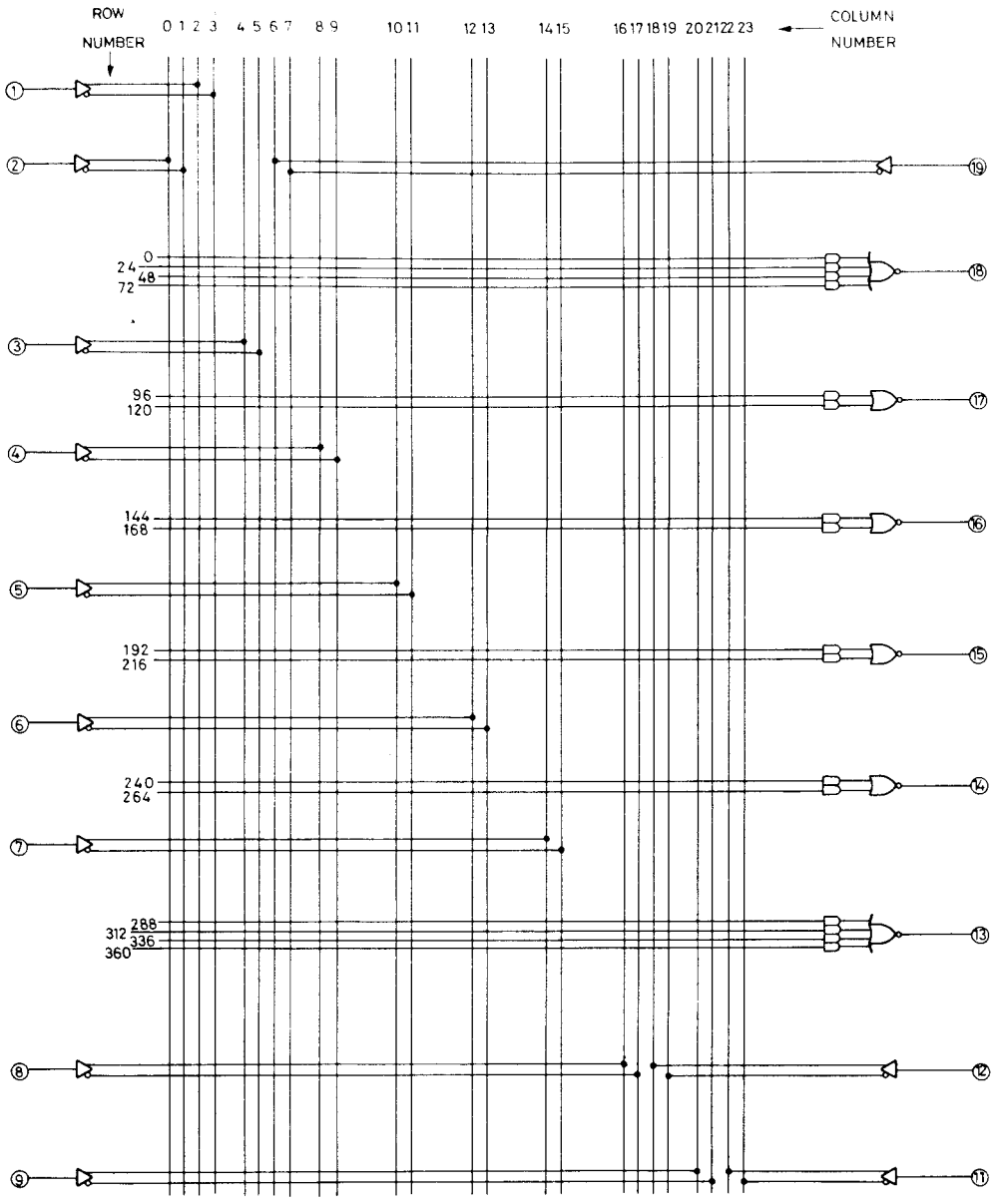
FUSE No. = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM PLA 12H6



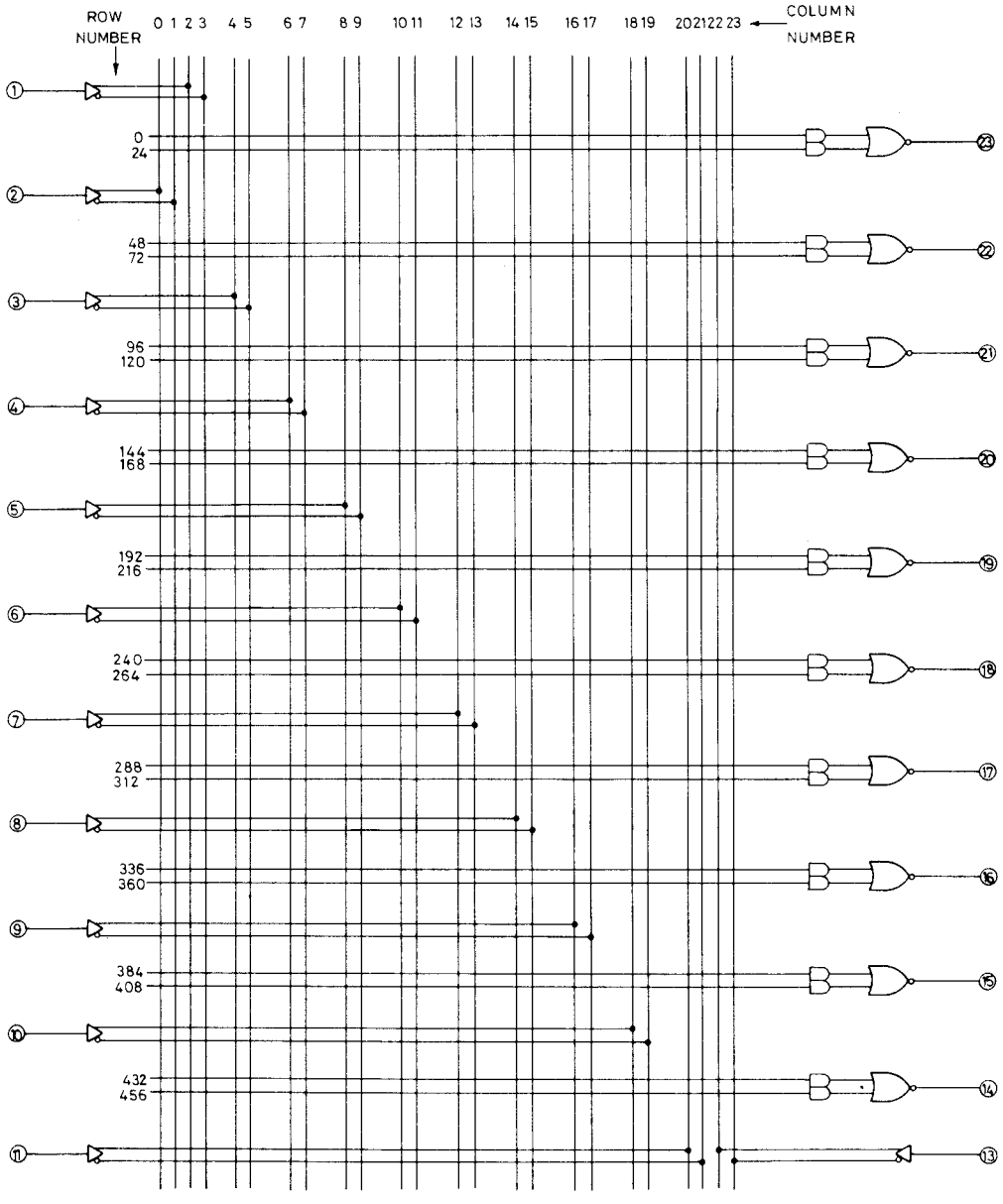
FUSE No = COLUMN No + ROW No

FUSE NUMBER DIAGRAM PLA 12L6



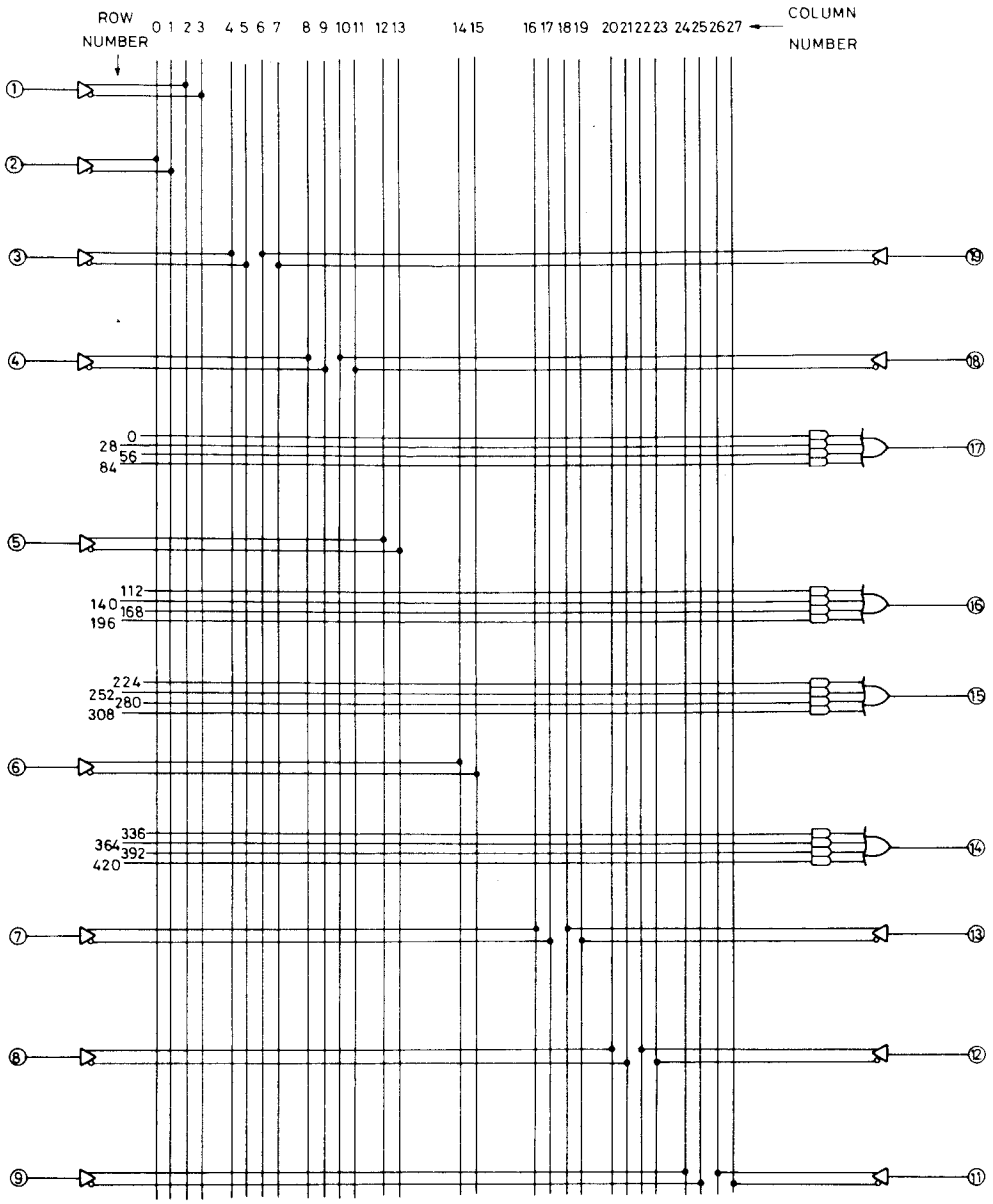
FUSE No = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM PLA 12L10



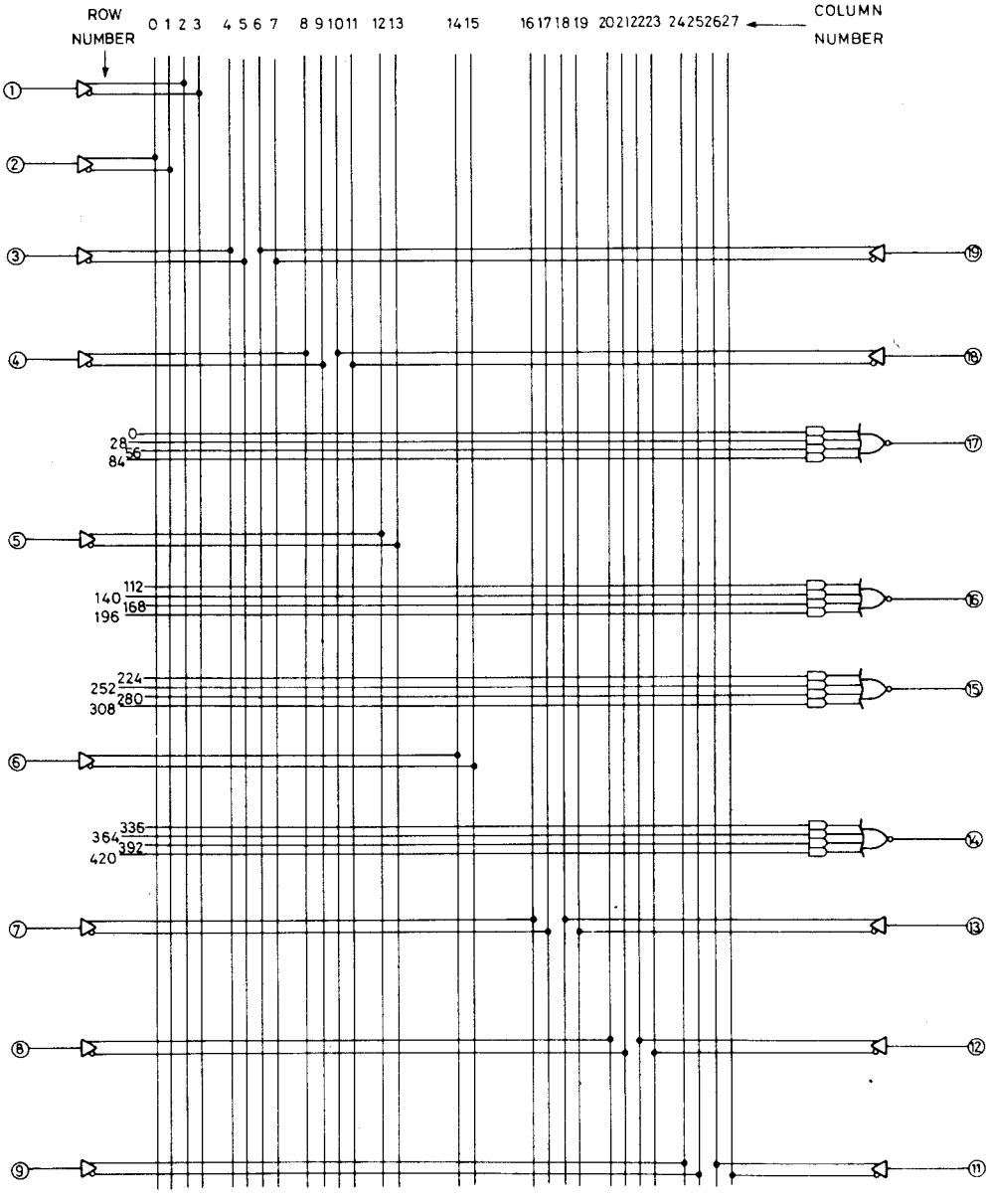
FUSE No. = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM PLA 14H4



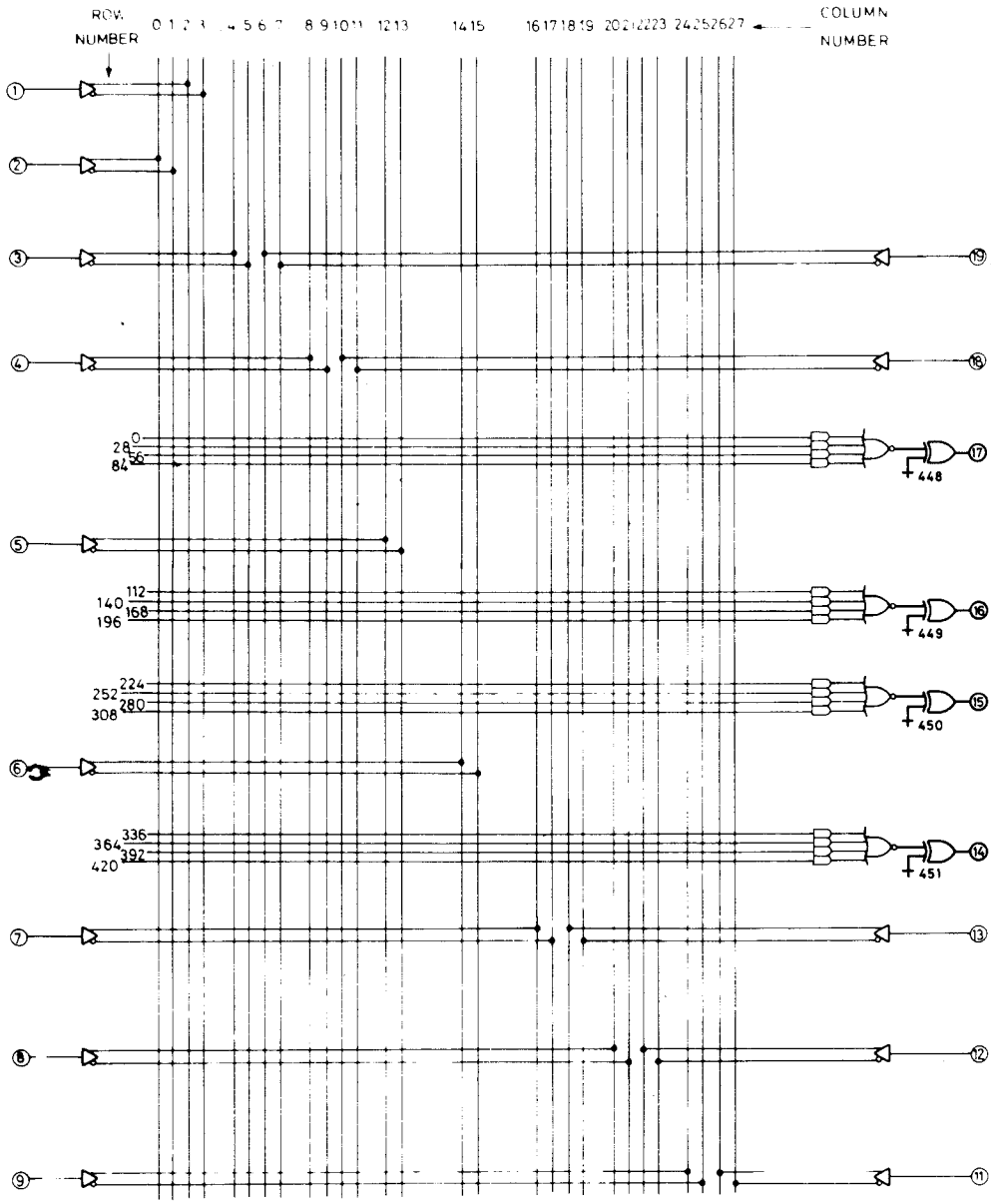
FUSE No. = COLUMN No. • ROW No.

FUSE NUMBER DIAGRAM PLA 14L4



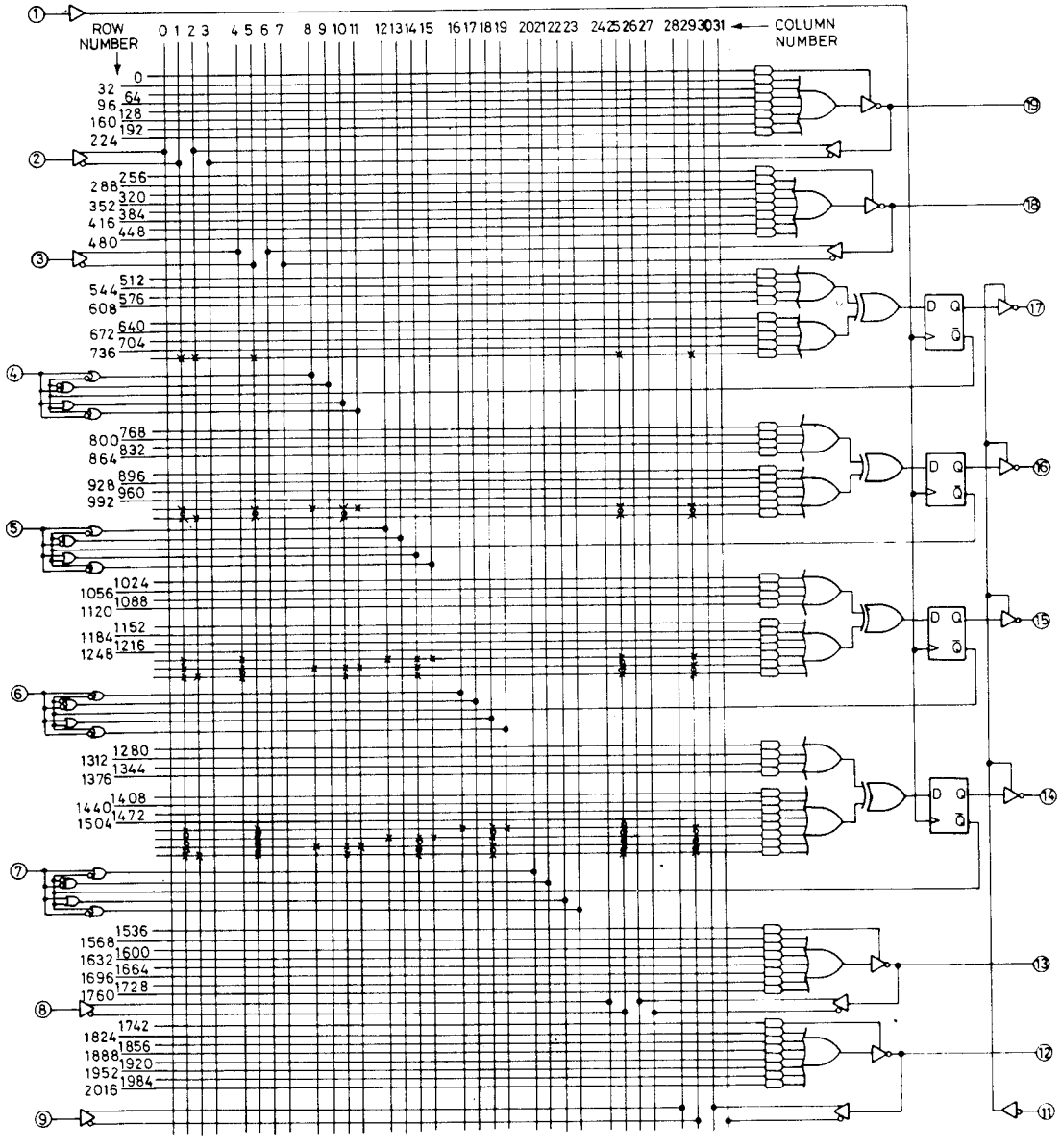
FUSE No = COLUMN No + ROW No

FUSE NUMBER DIAGRAM PLA 14P4



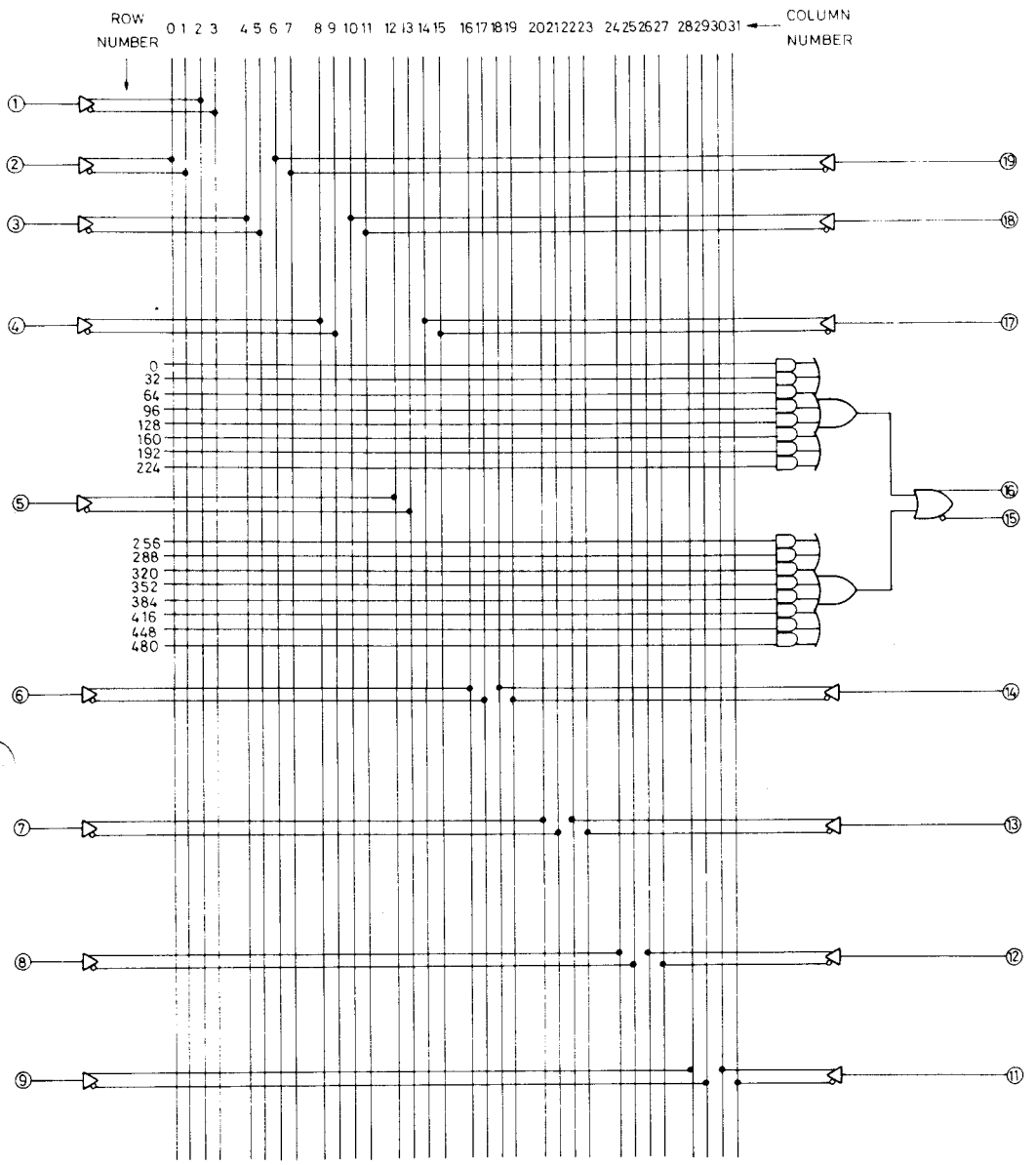
FUSE No = COLUMN No. * ROW No

FUSE NUMBER DIAGRAM PLA 16A4



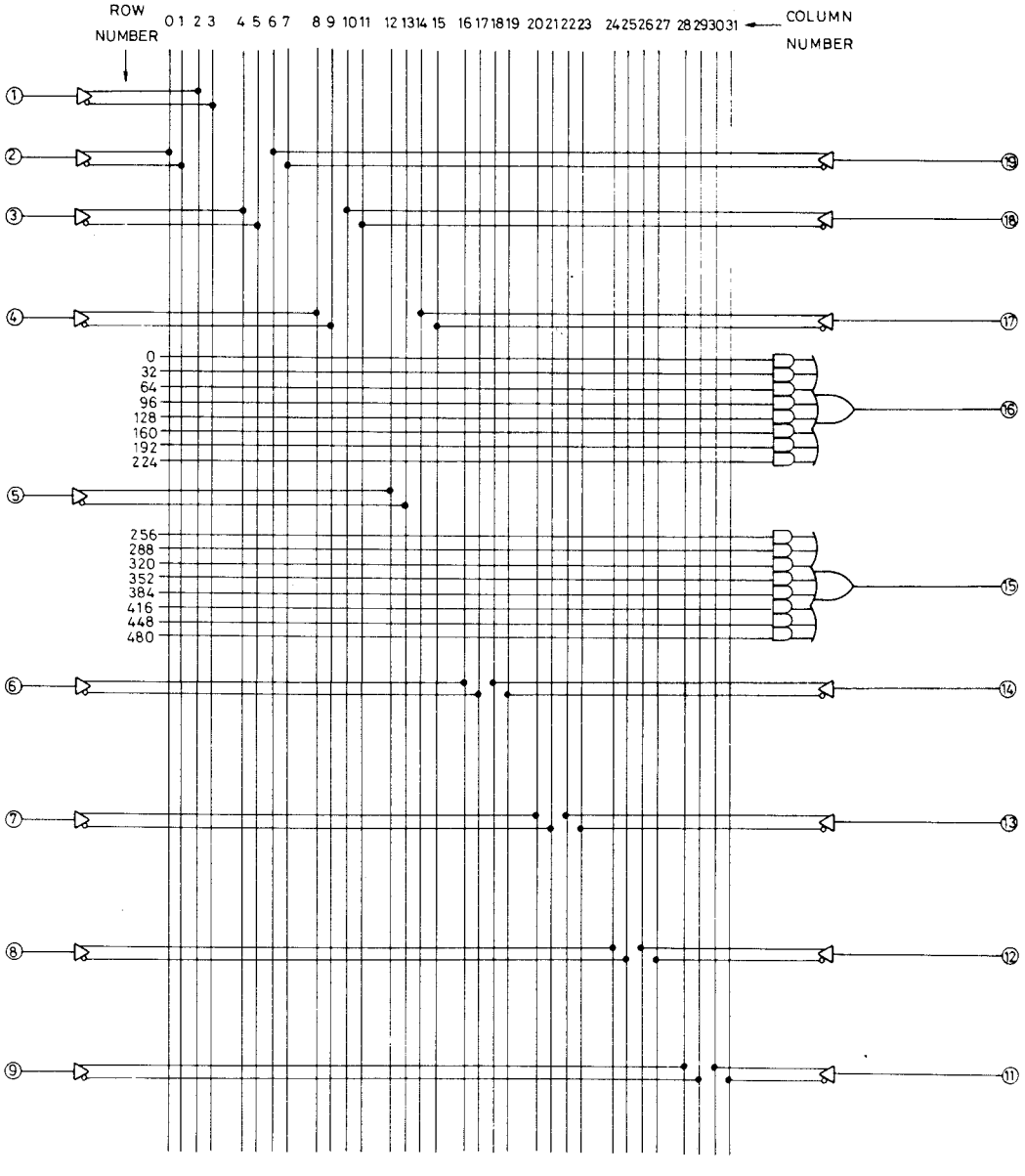
FUSE No. = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM PLA 16C1



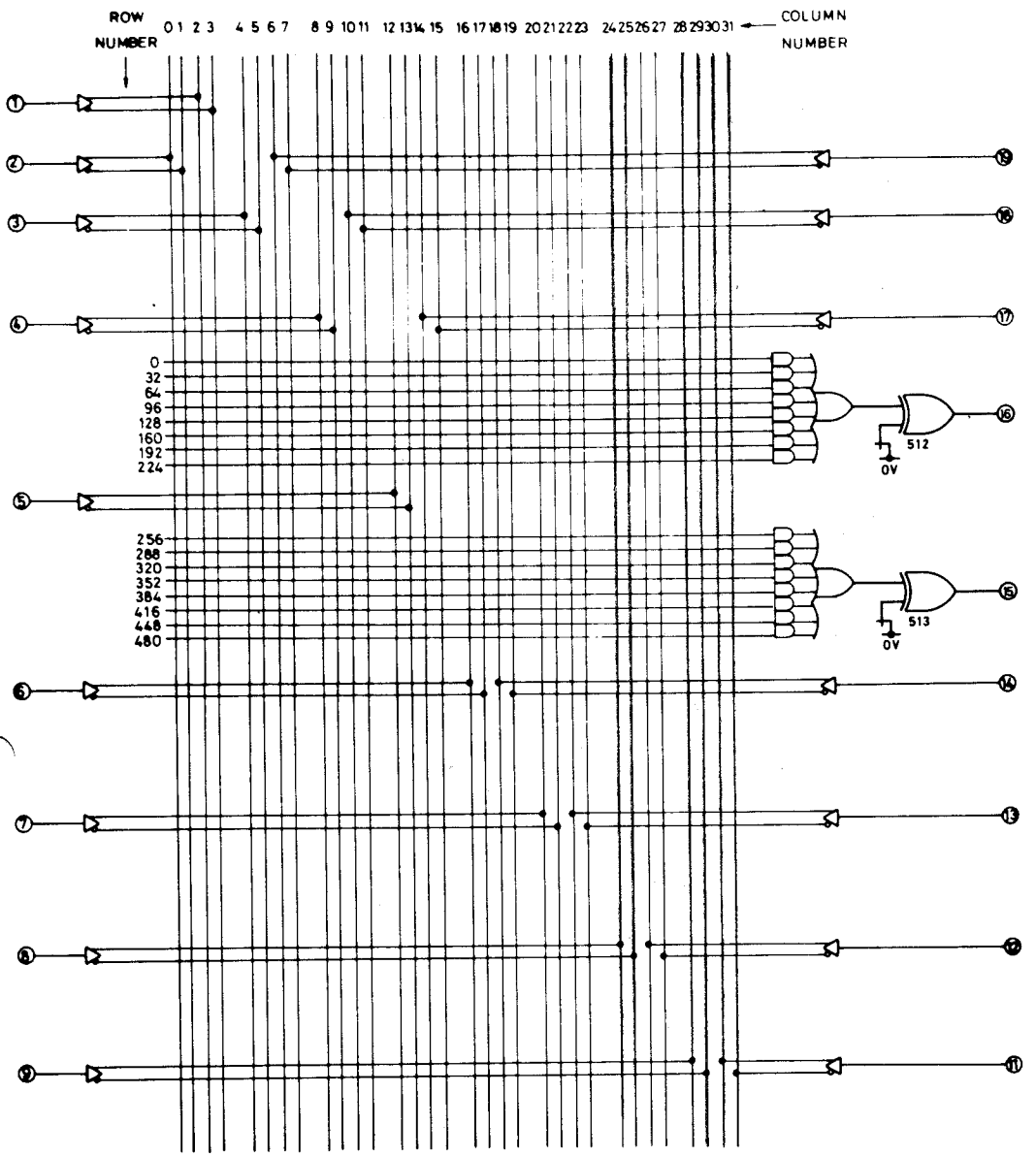
FUSE No = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM PLA 16H2



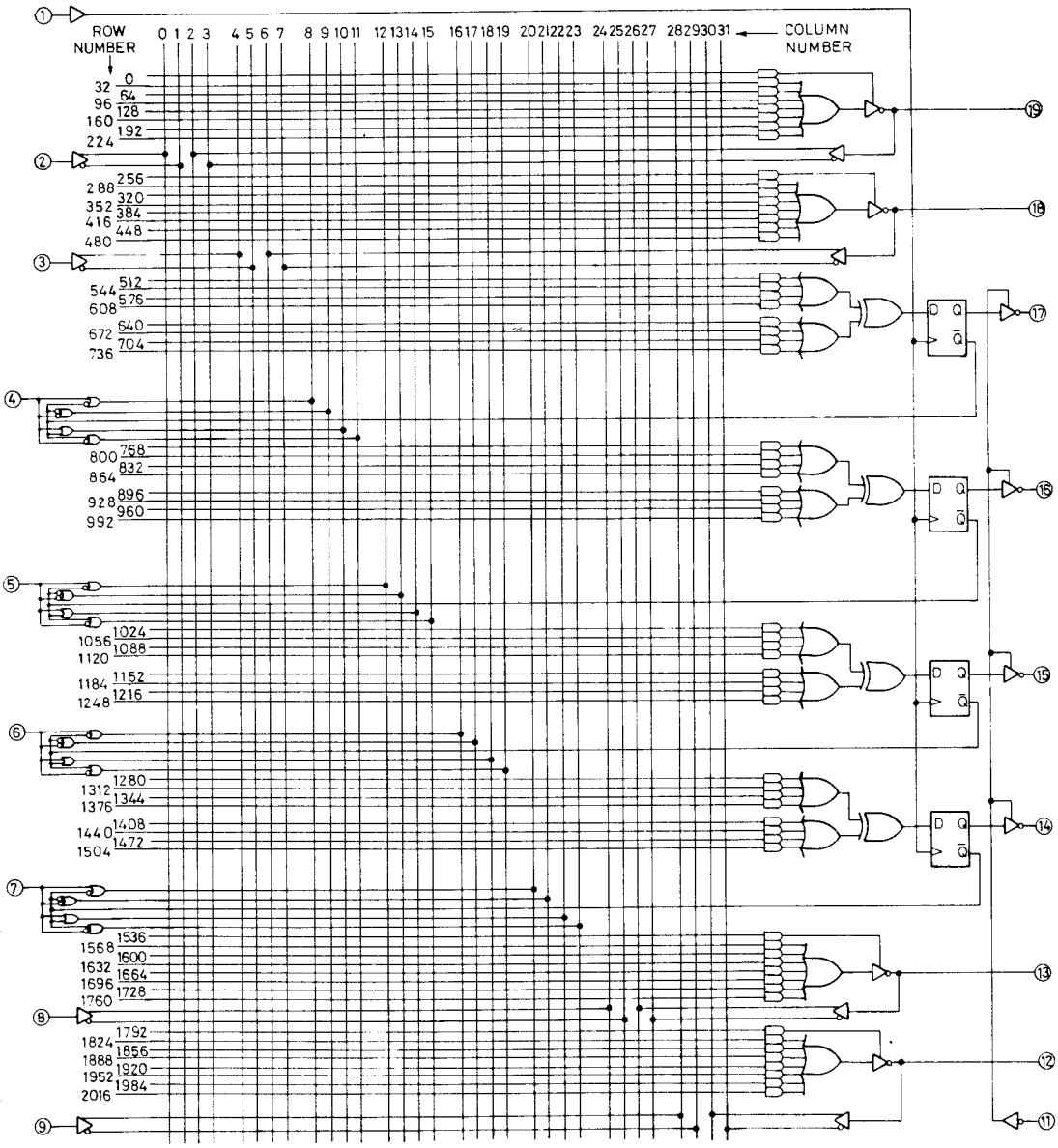
FUSE No = COLUMN No + ROW No

FUSE NUMBER DIAGRAM PLA 16P2



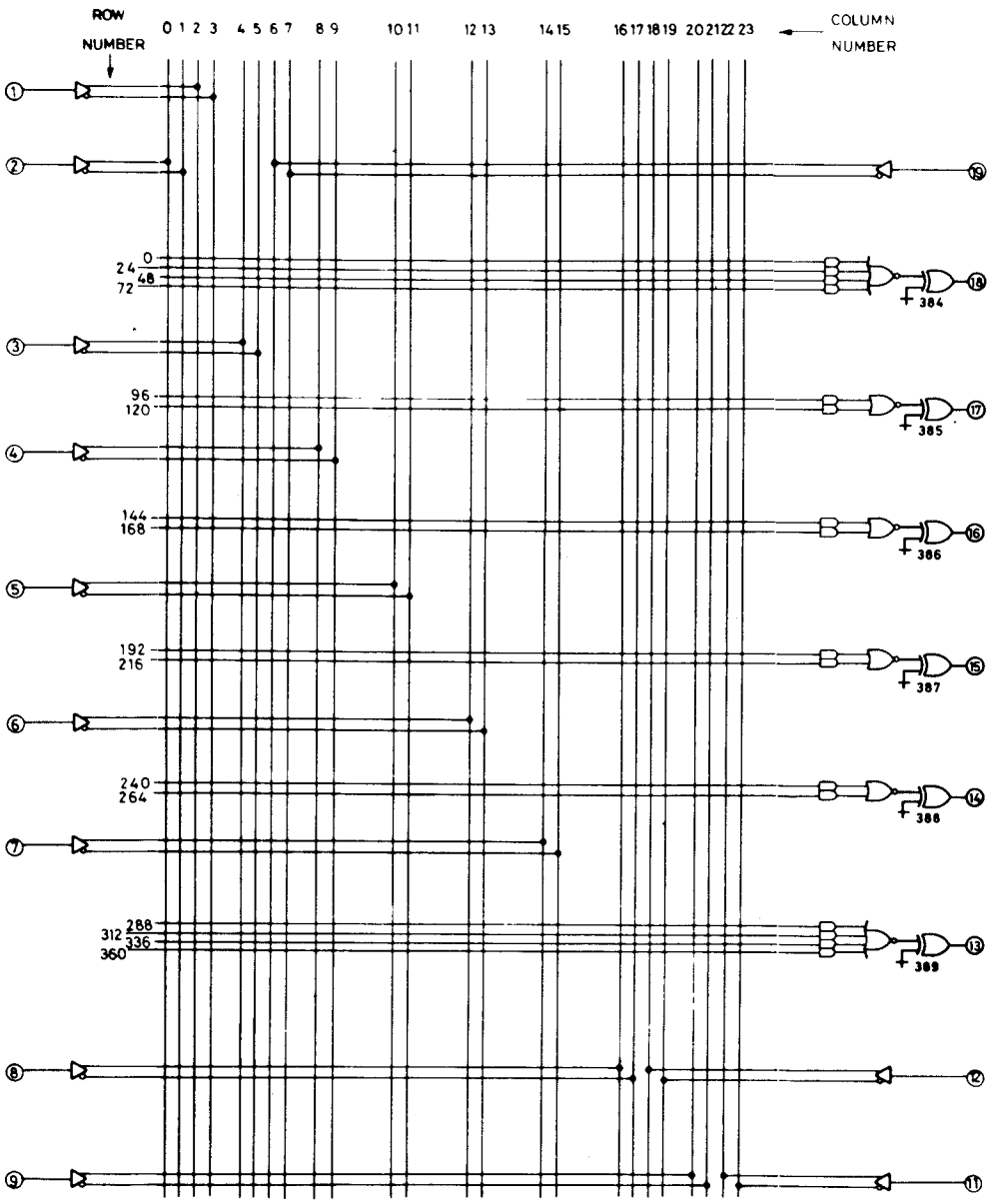
FUSE No. = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM PLA 16X4



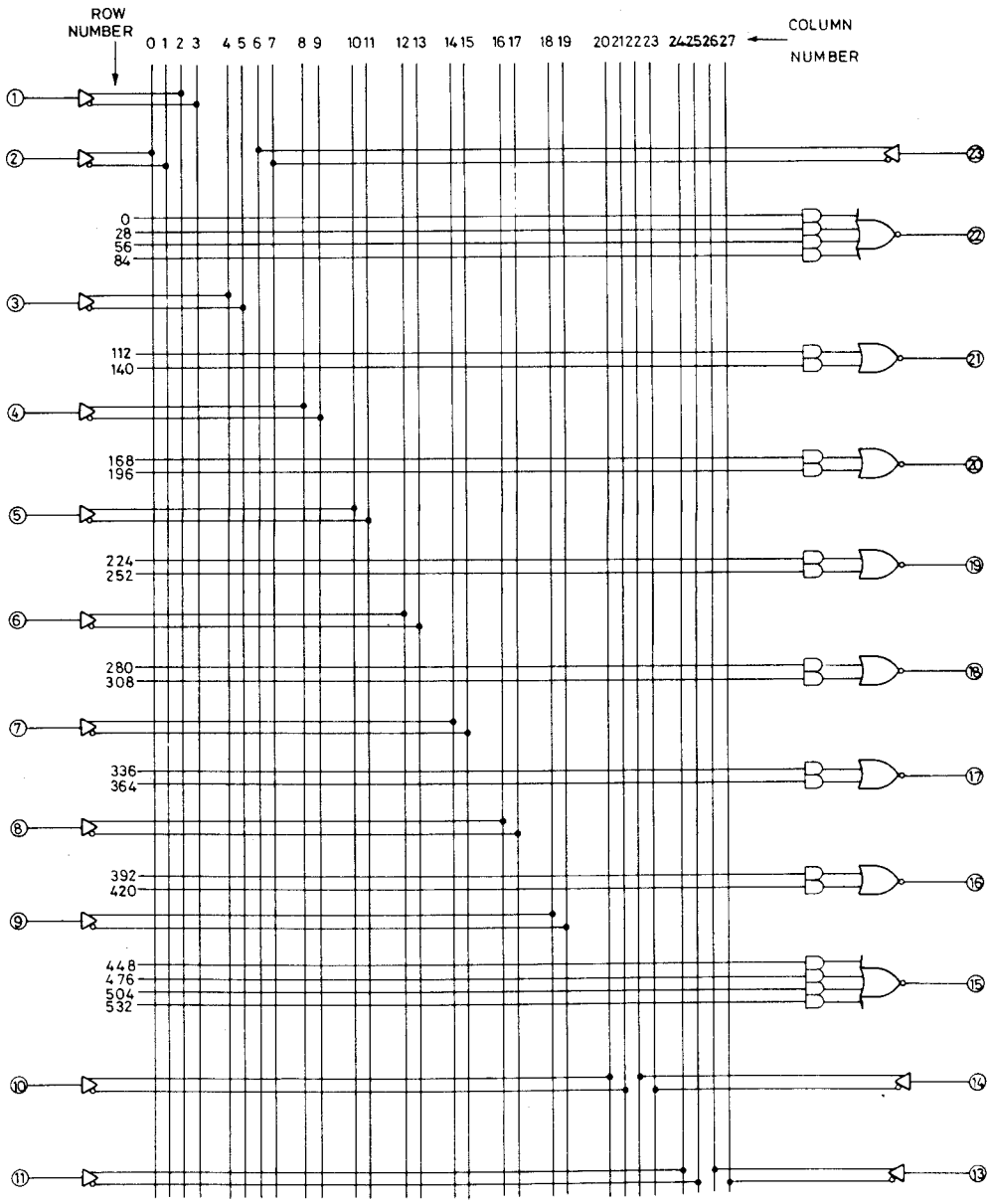
FUSE No = COLUMN No • ROW No

FUSE NUMBER DIAGRAM PLA 12P6



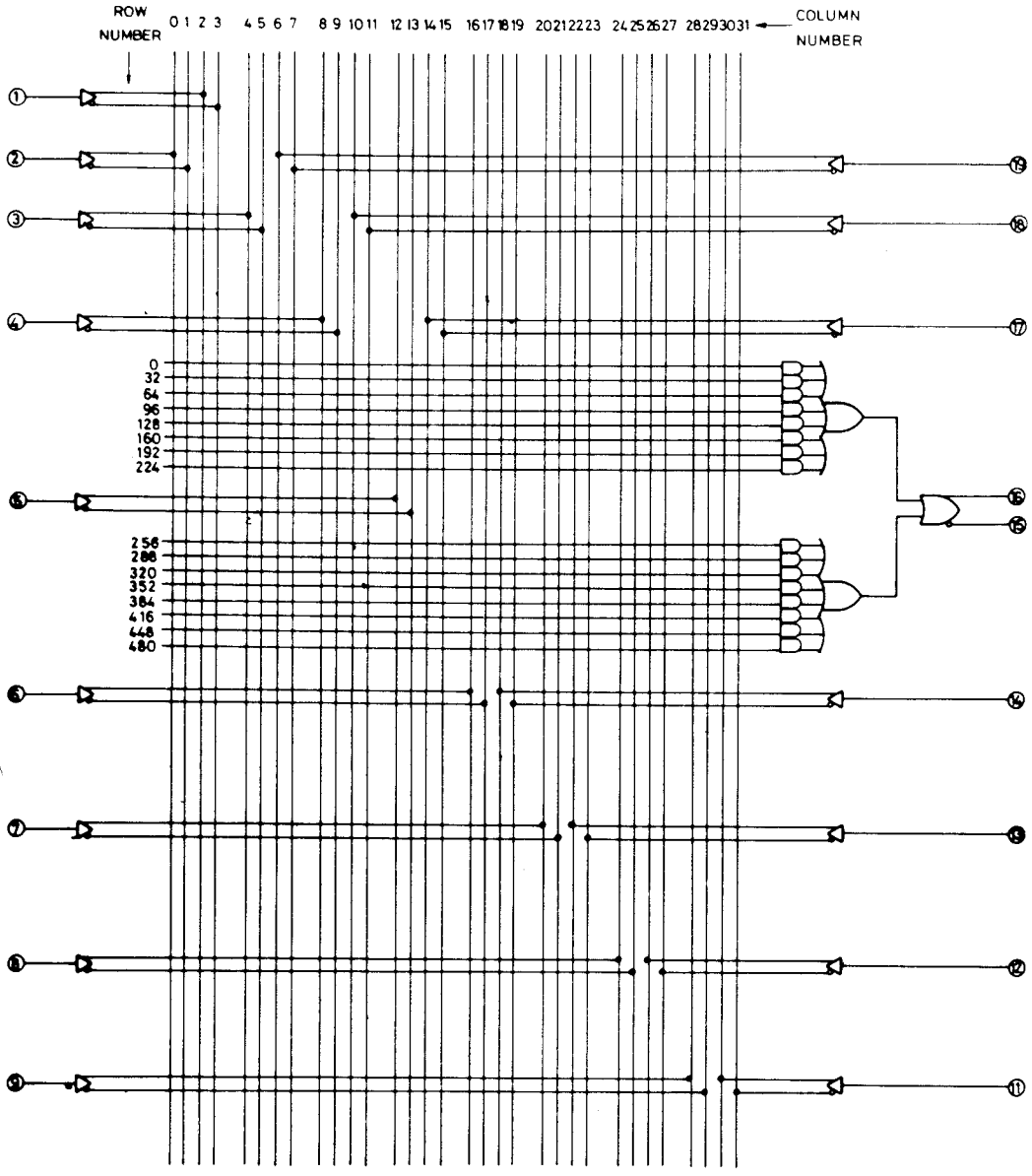
FUSE No. = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM PLA 14L4



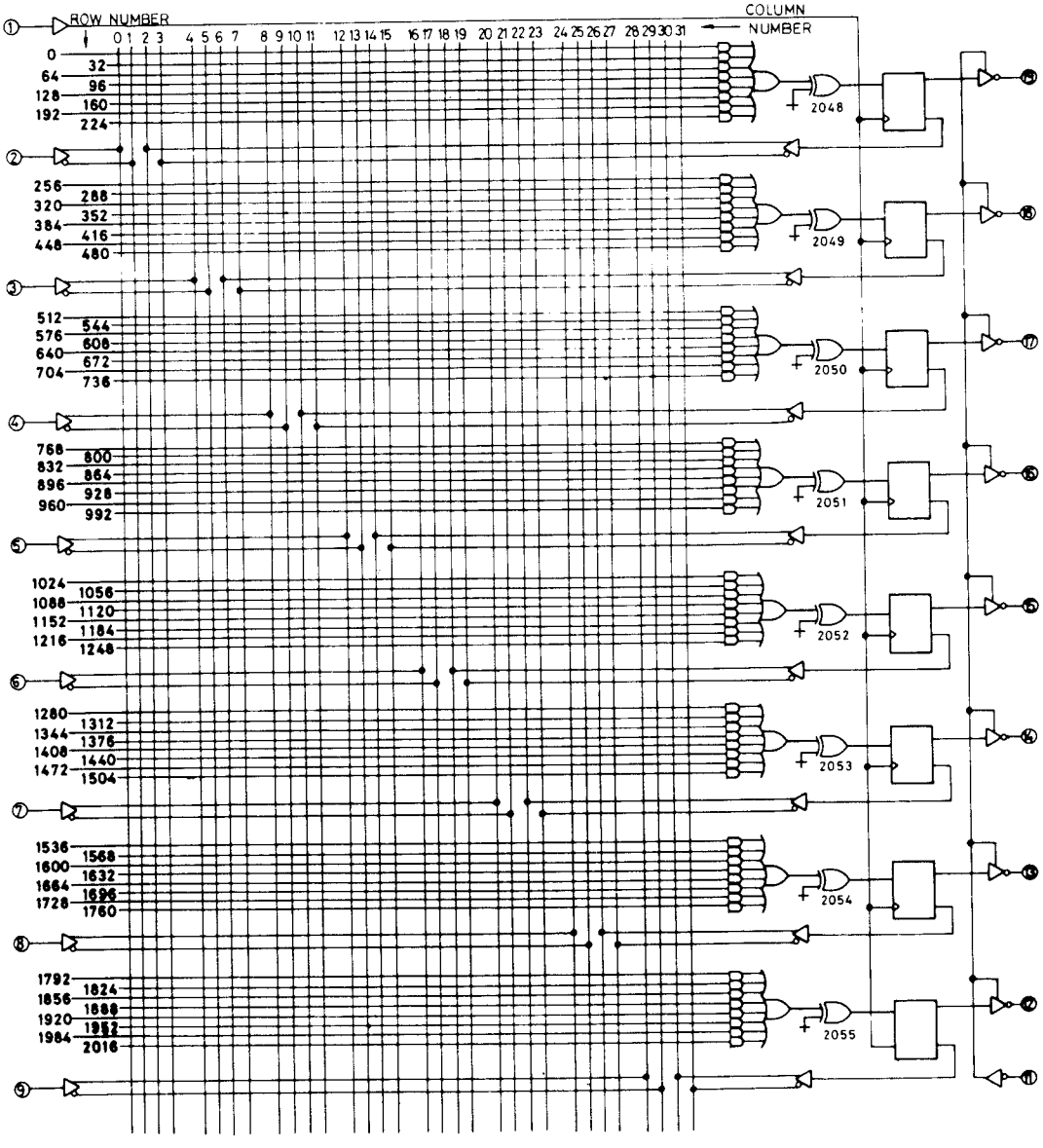
FUSE No.= COLUMN No • ROW No.

FUSE NUMBER DIAGRAM PLA 16CP1



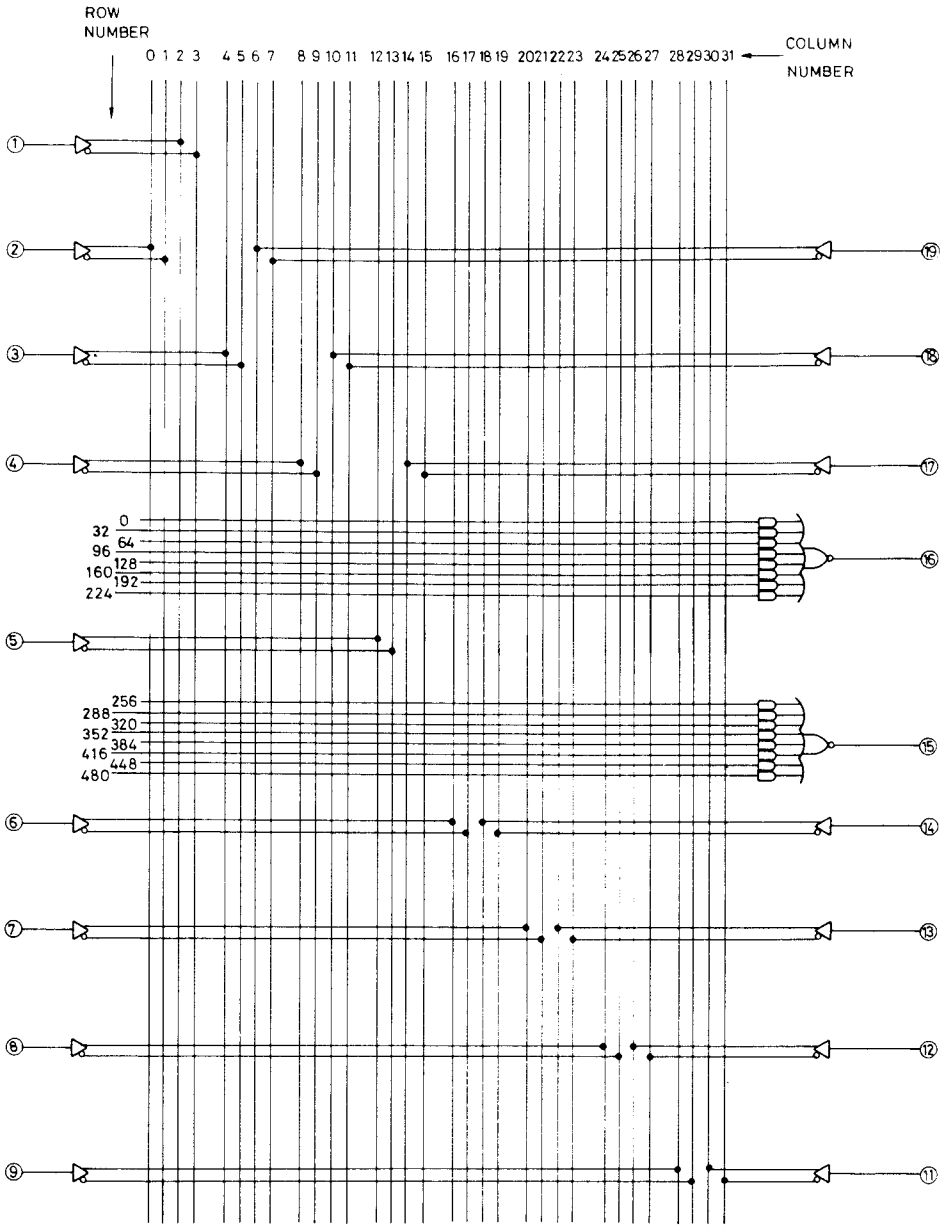
FUSE No. = COLUMN No + ROW No.

FUSE NUMBER DIAGRAM 16CP8



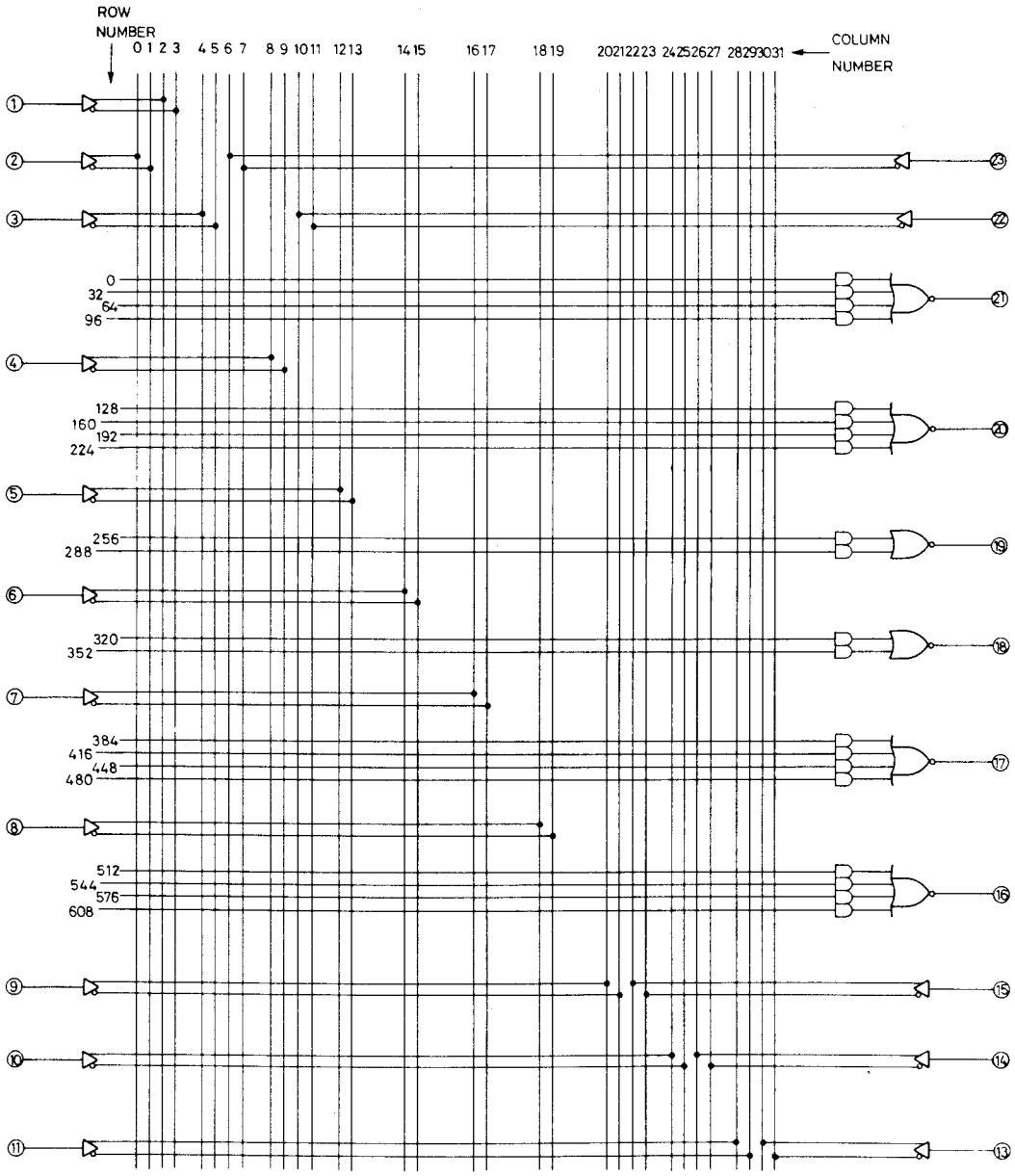
FUSE No = COLUMN No + ROW No

FUSE NUMBER DIAGRAM 16L2



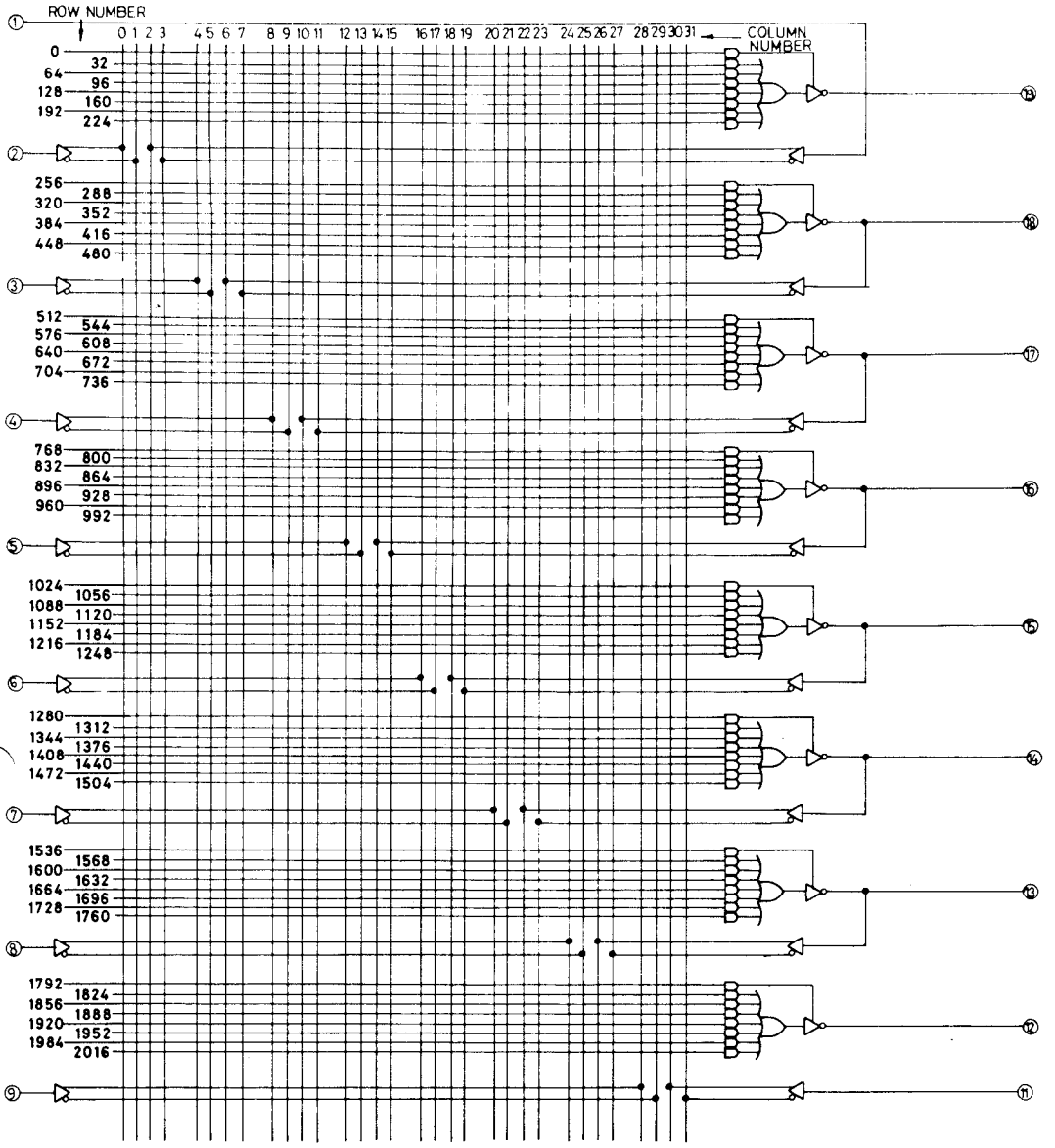
FUSE No = COLUMN No + ROW No

FUSE NUMBER DIAGRAM 16L6



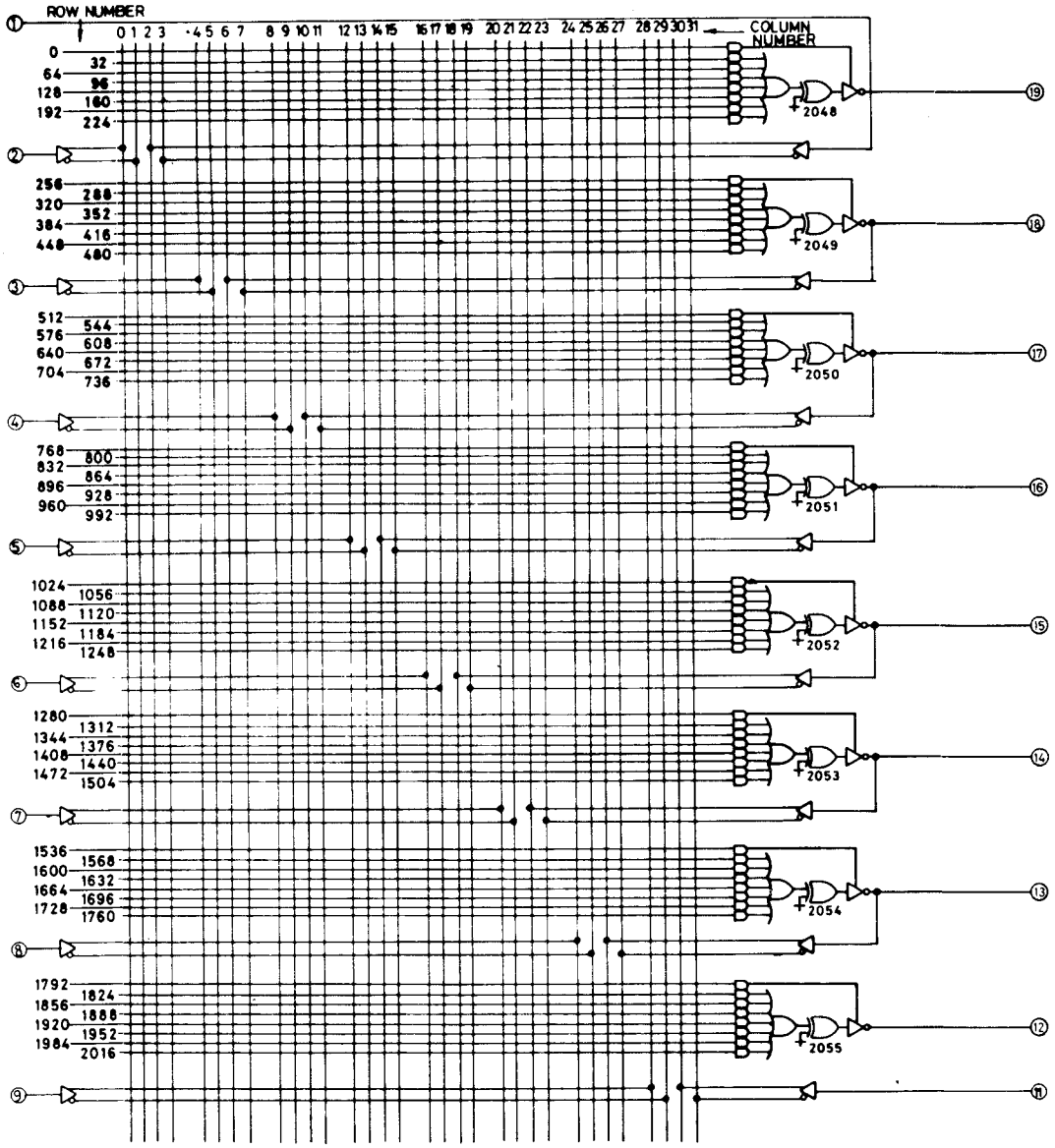
FUSE No = COLUMN No. + ROW No

FUSE NUMBER DIAGRAM 16L8



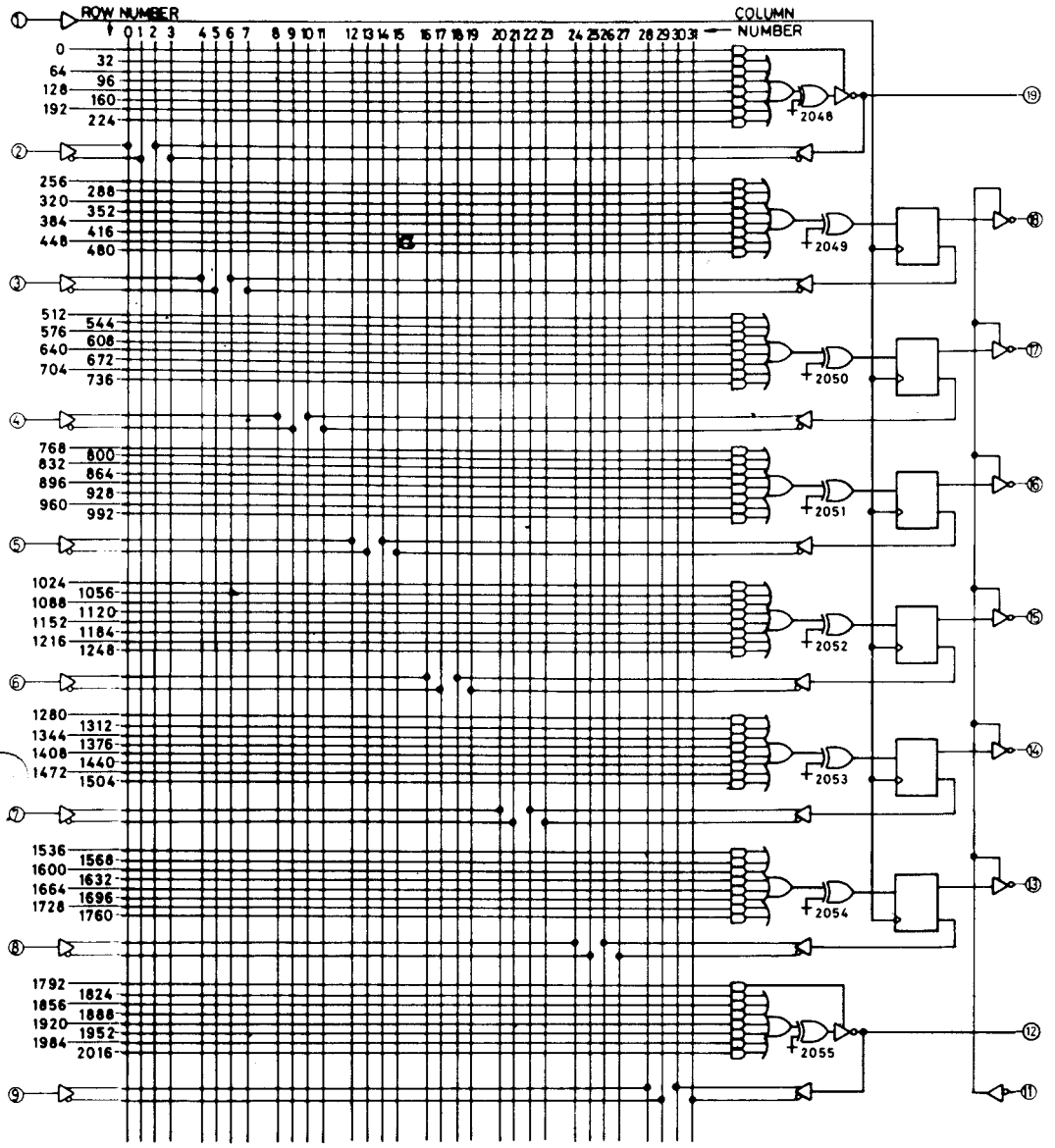
FUSE No = COLUMN No • ROW No.

FUSE NUMBER DIAGRAM 16P8

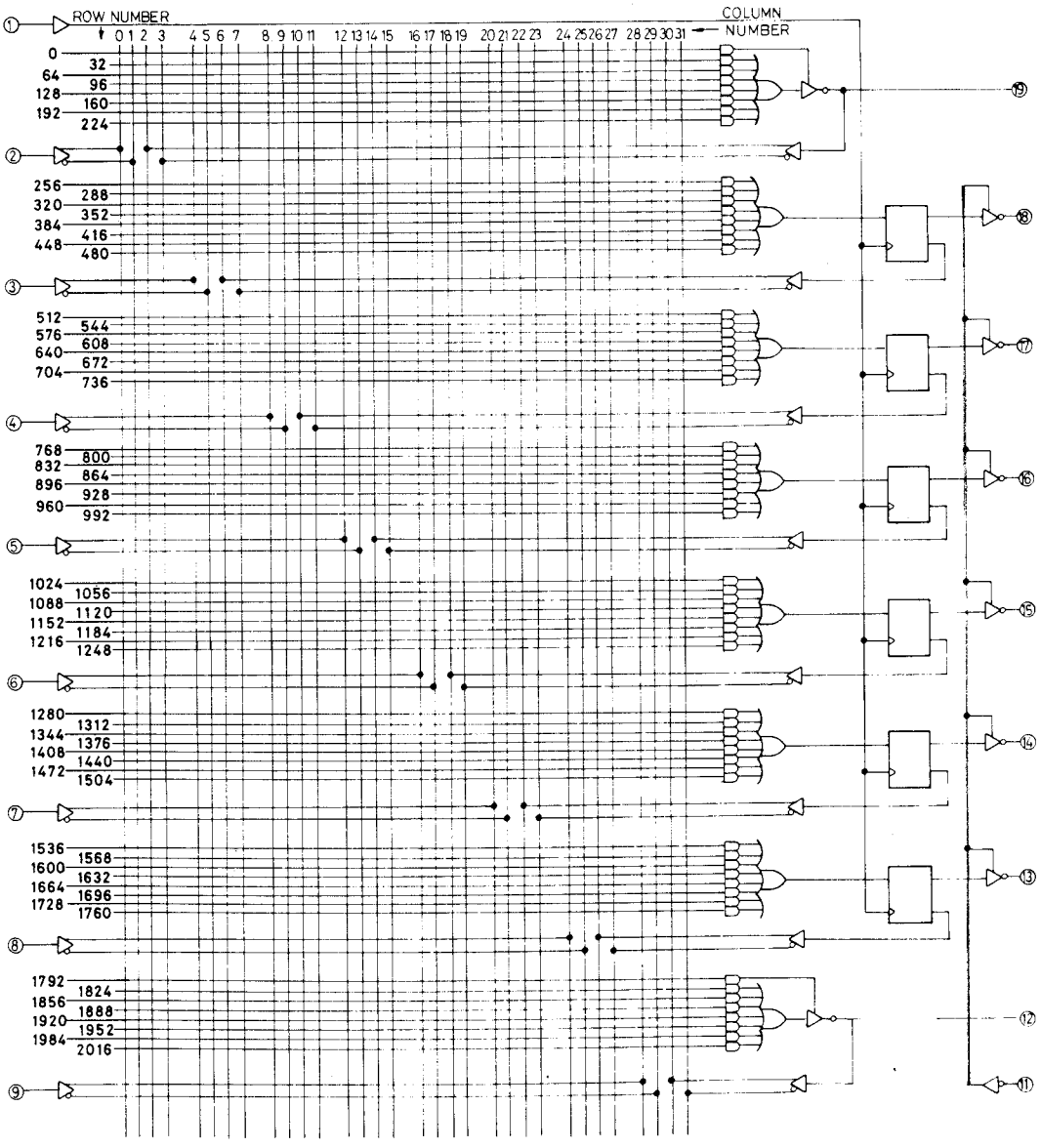


FUSE No = COLUMN No. • ROW No

FUSE NUMBER DIAGRAM 16RP6

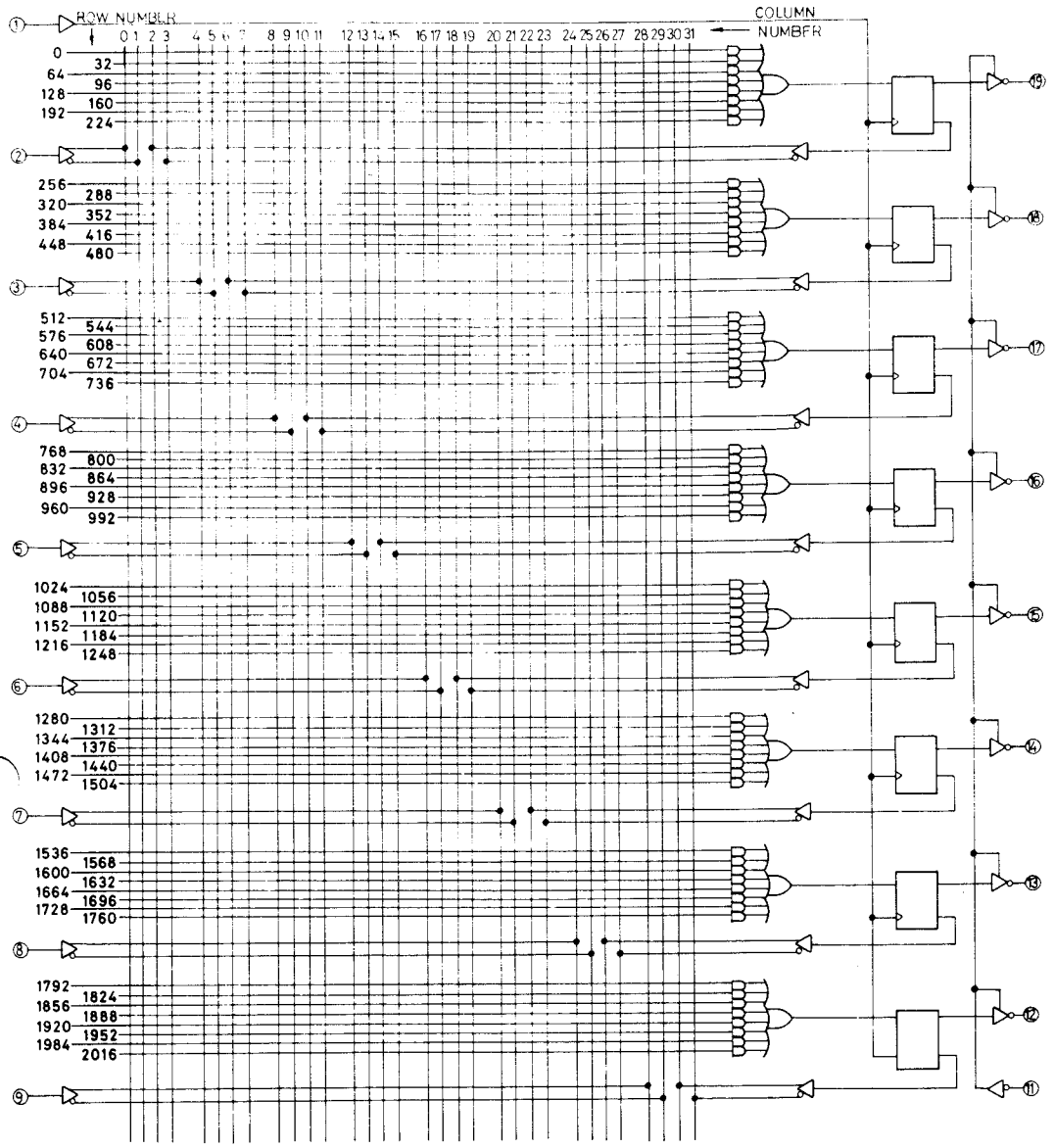


FUSE No = COLUMN No * ROW No



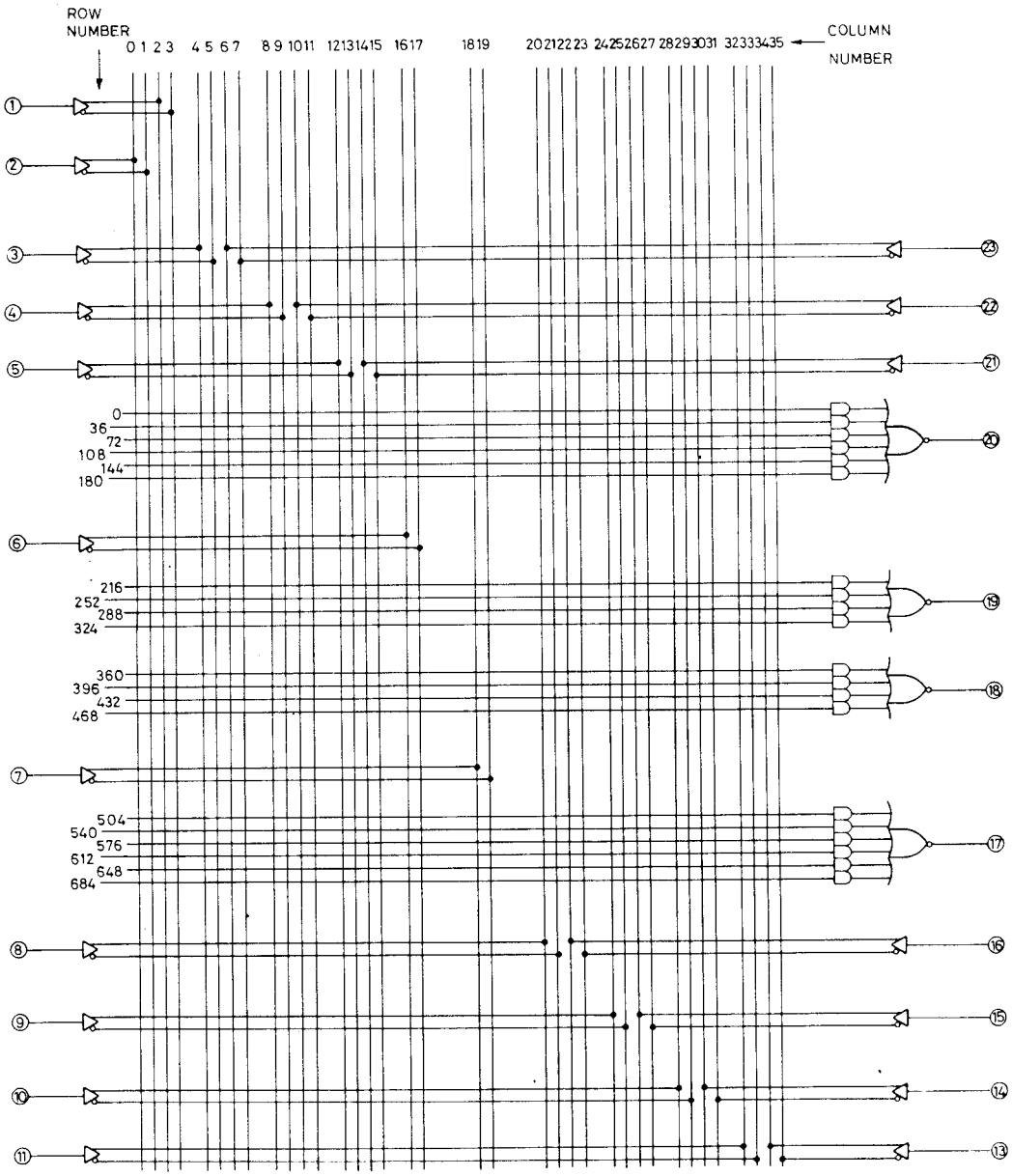
FUSE No = COLUMN No * ROW No

FUSE NUMBER DIAGRAM 16R8



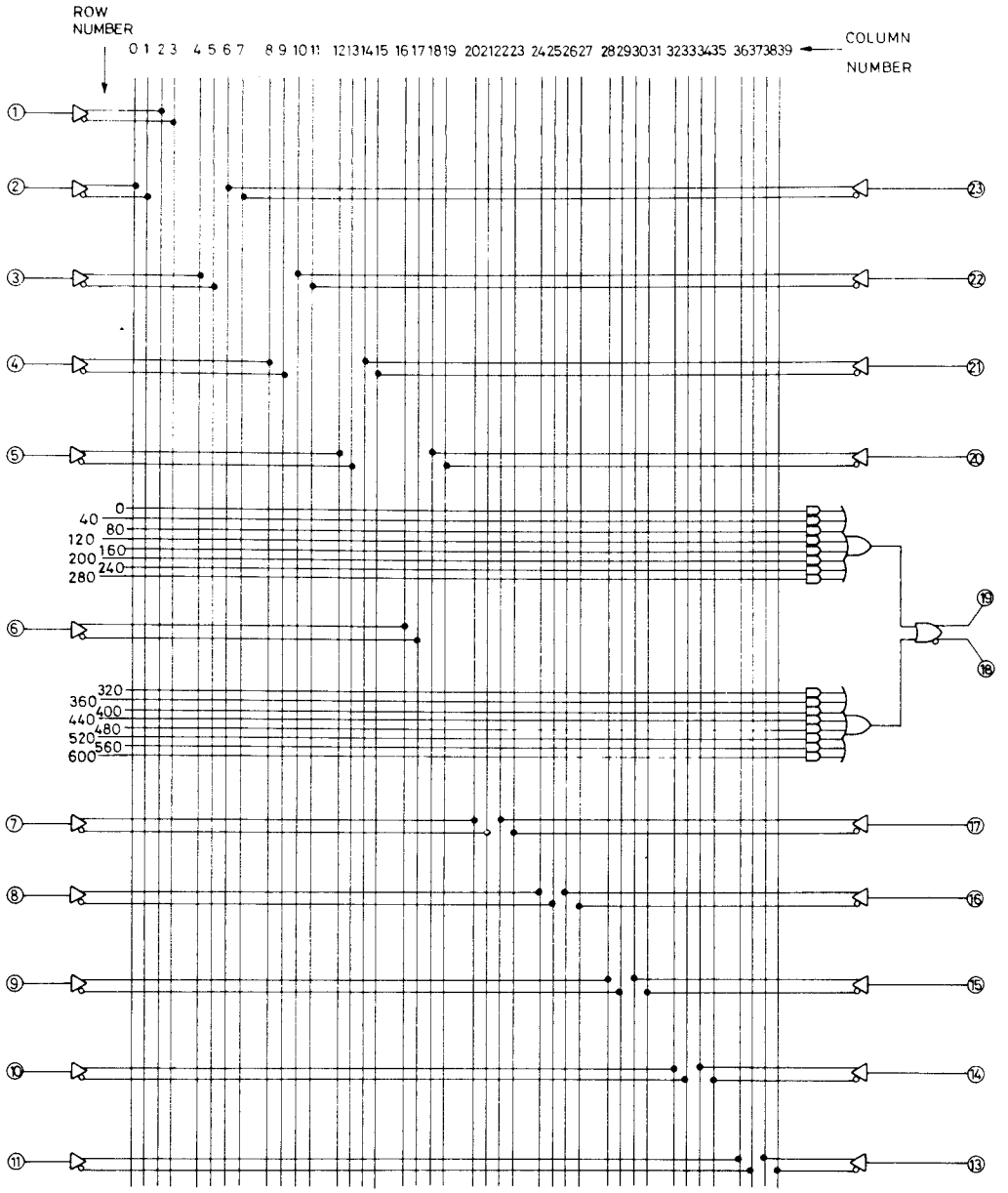
FUSE No = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM 18L4

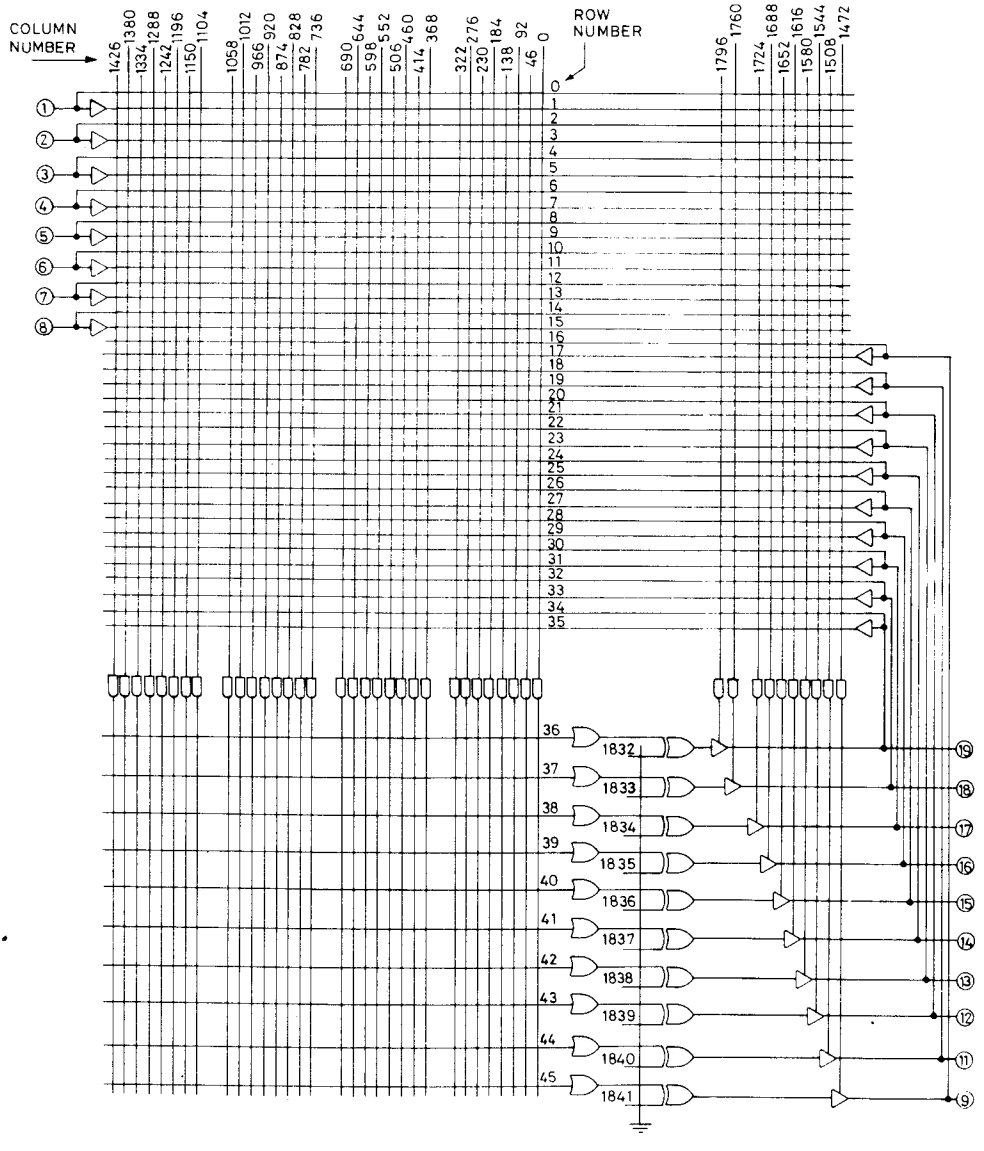


FUSE No = COLUMN No • ROW No

FUSE NUMBER DIAGRAM 20C1

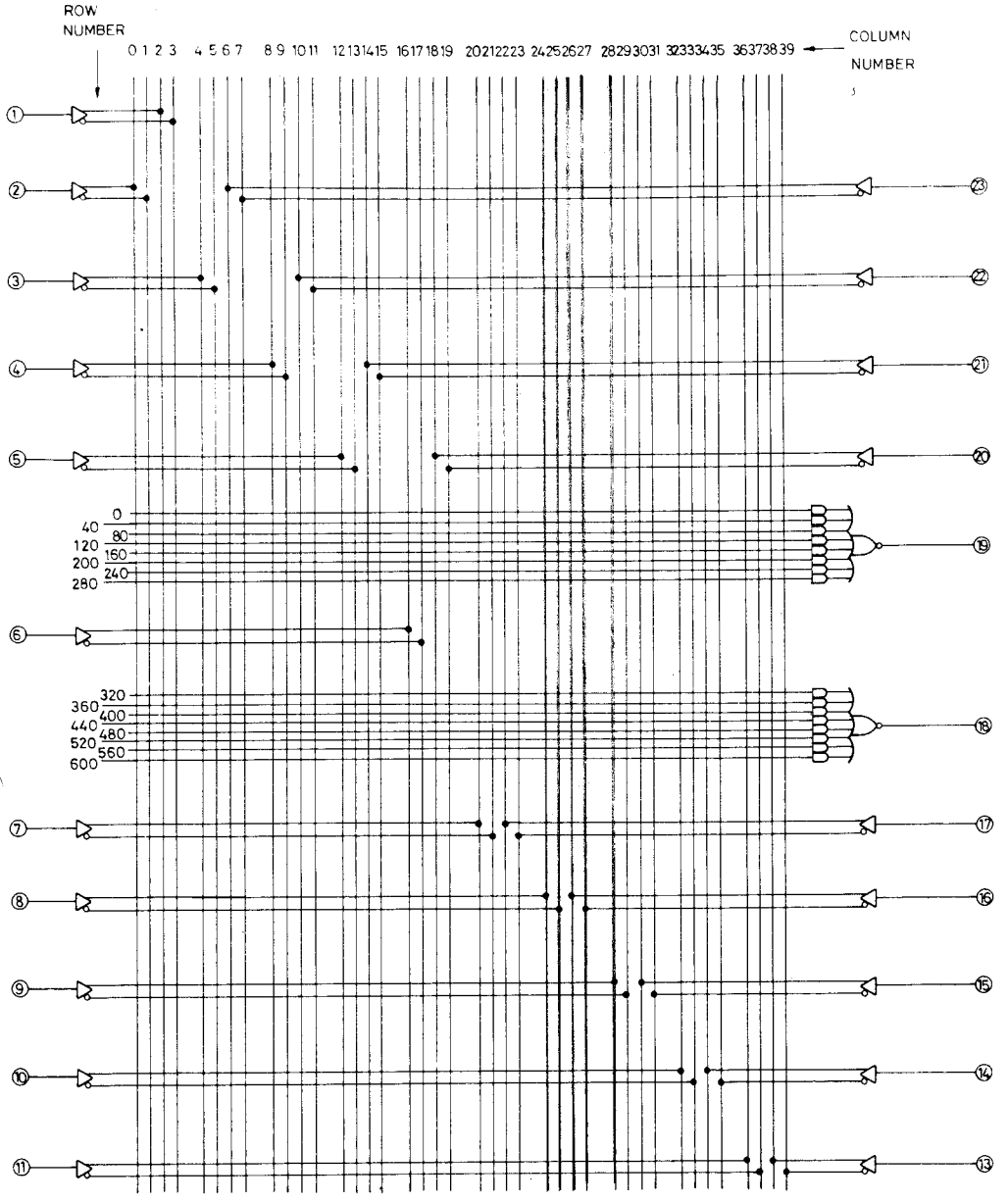


FUSE No = COLUMN No. + ROW No.



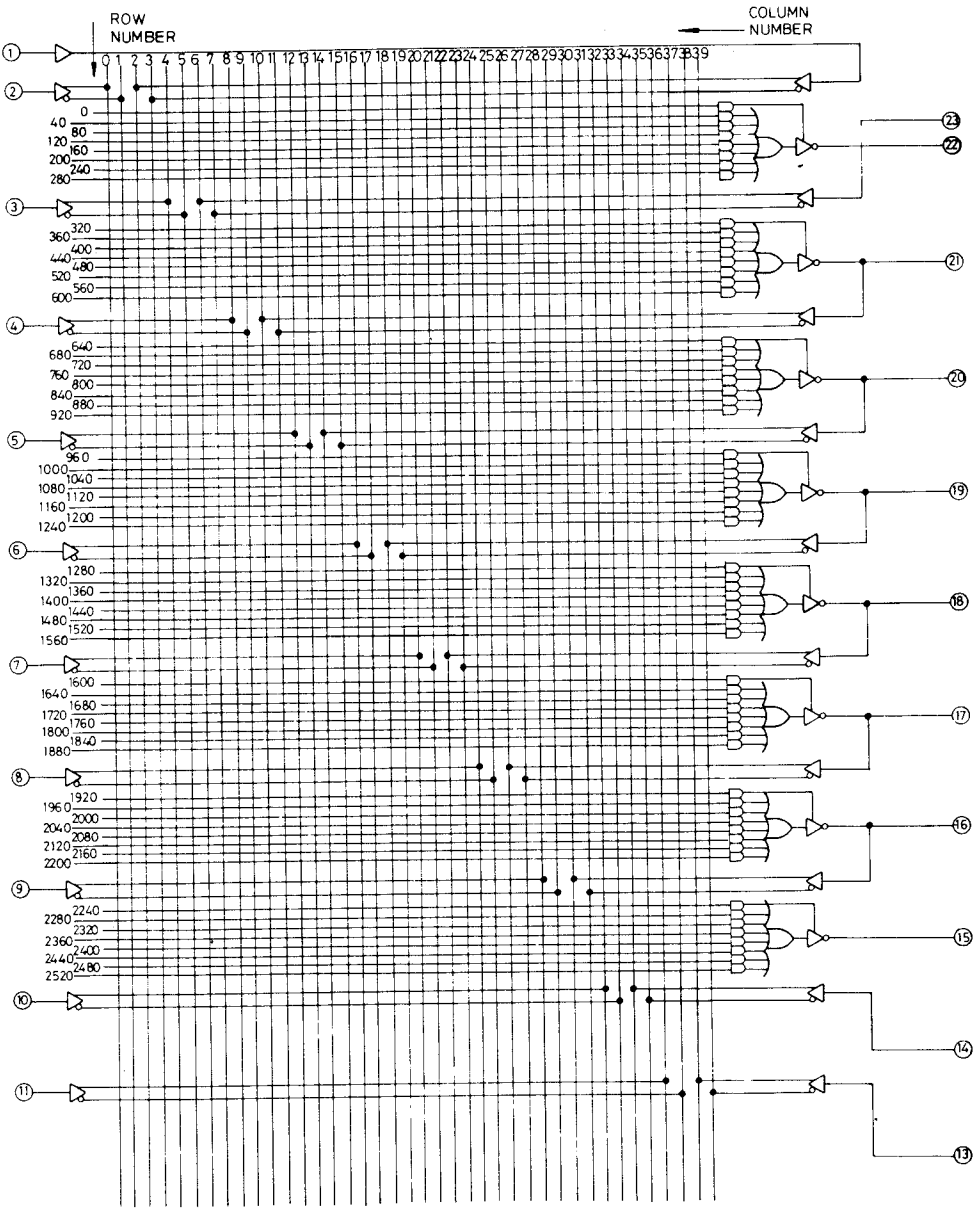
FUSE No = COLUMN No + ROW No

FUSE NUMBER DIAGRAM 20L2



FUSE No. = COLUMN No. + ROW No

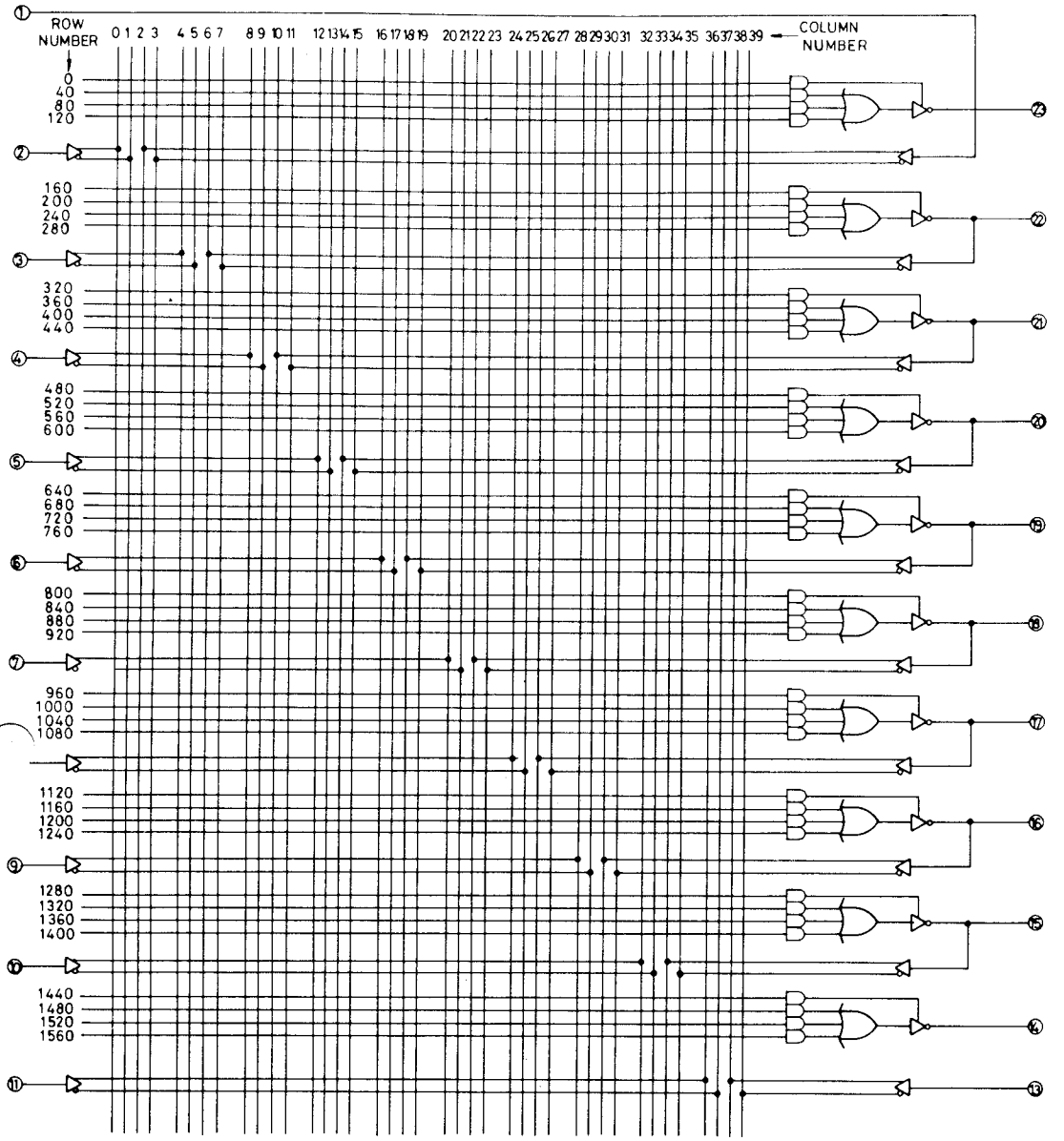
FUSE NUMBER DIAGRAM 20L8

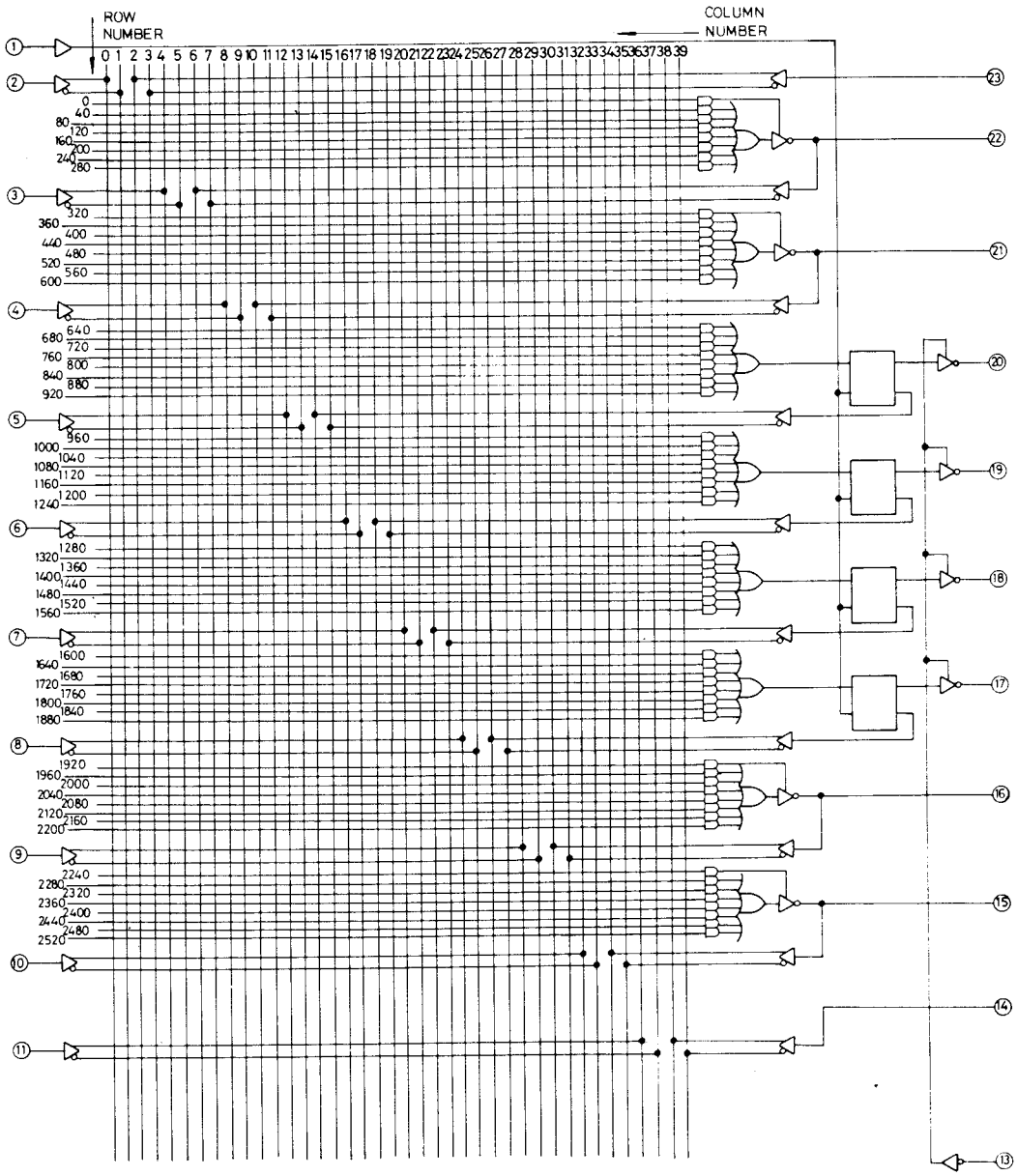


FUSE No = COLUMN No • ROW No

FUSE NUMBER DIAGRAM 20L10

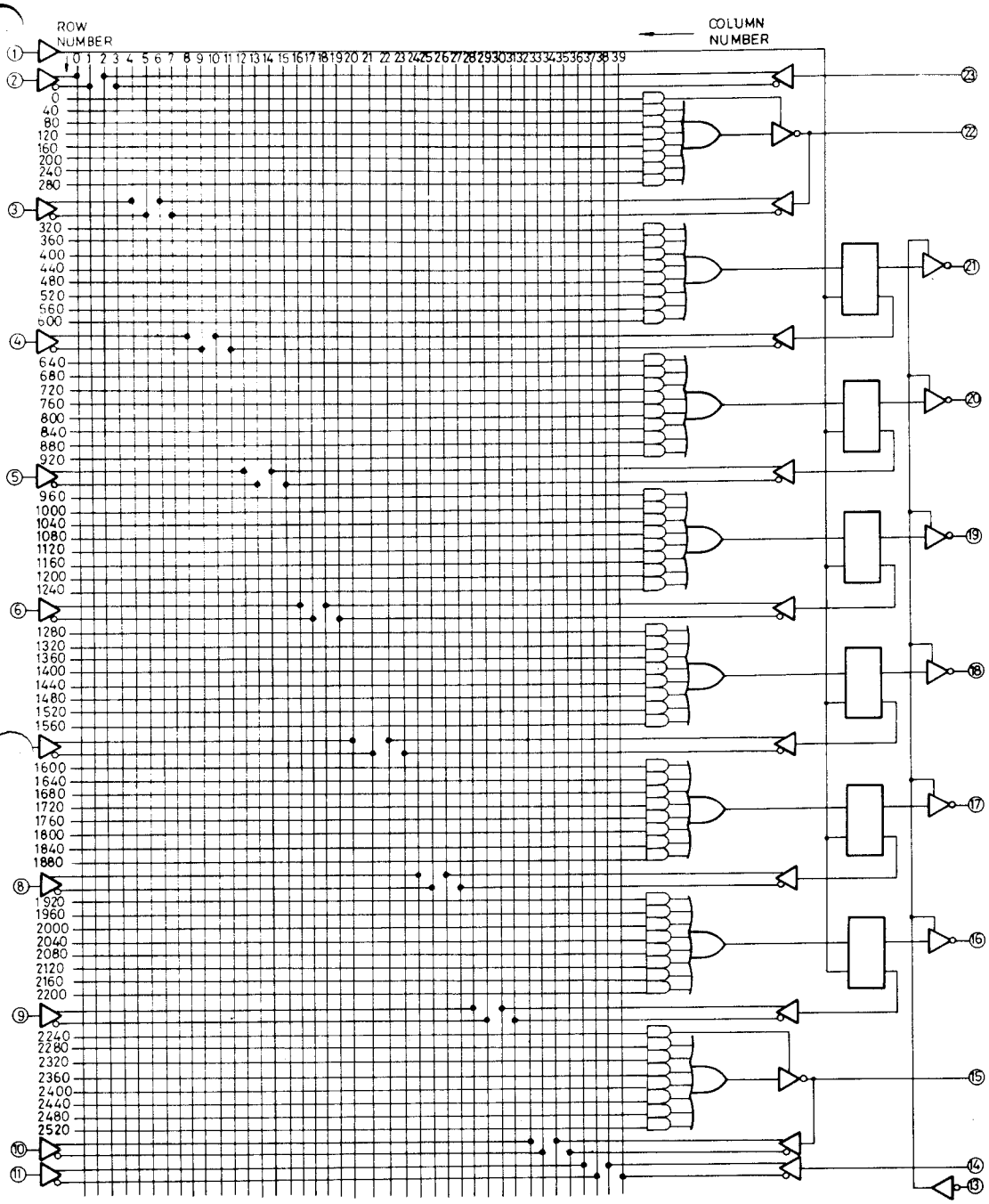
FUSE NUMBER DIAGRAM 20L10





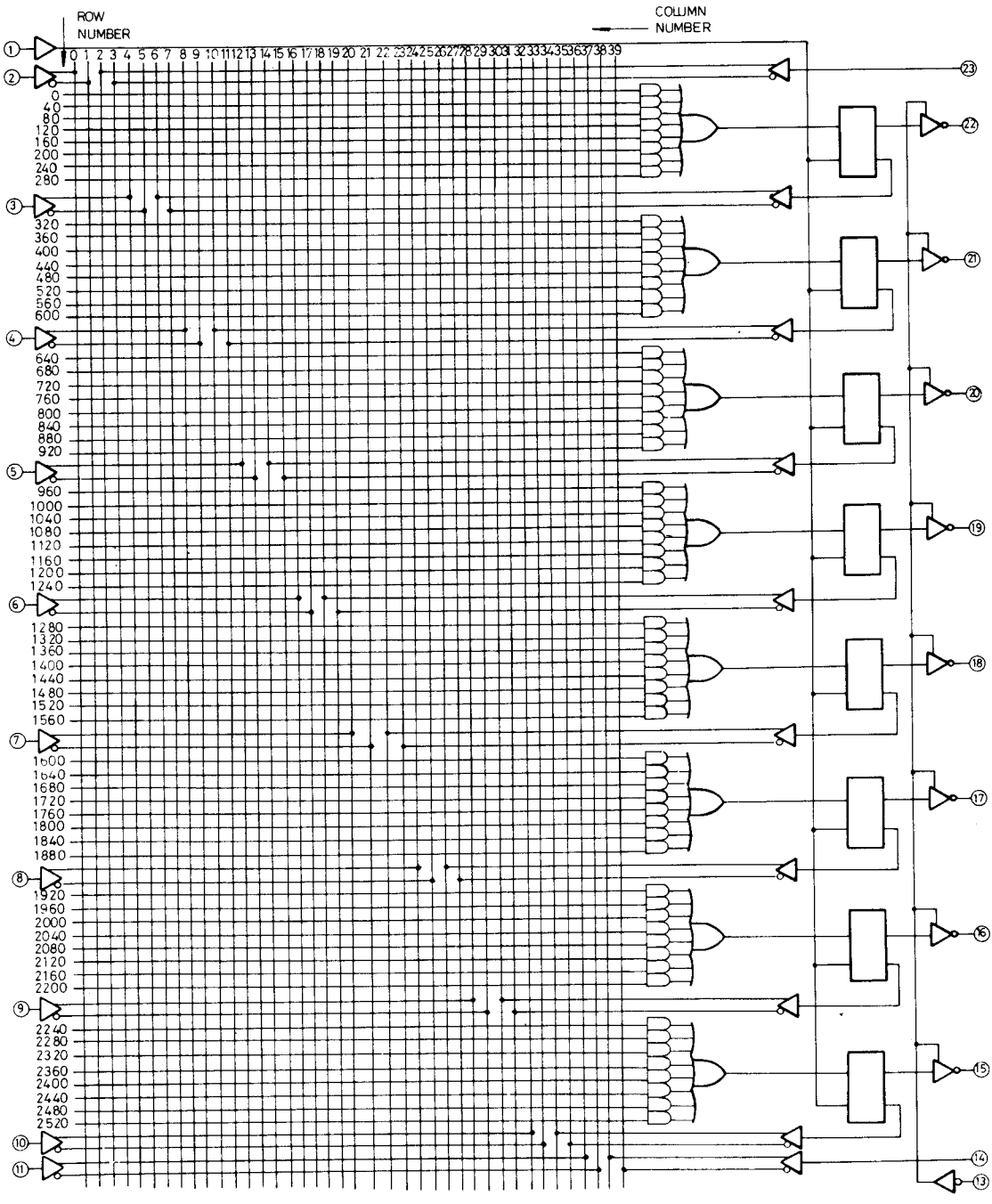
FUSE No. = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM 20R6



FUSE No = COLUMN No. • ROW No.

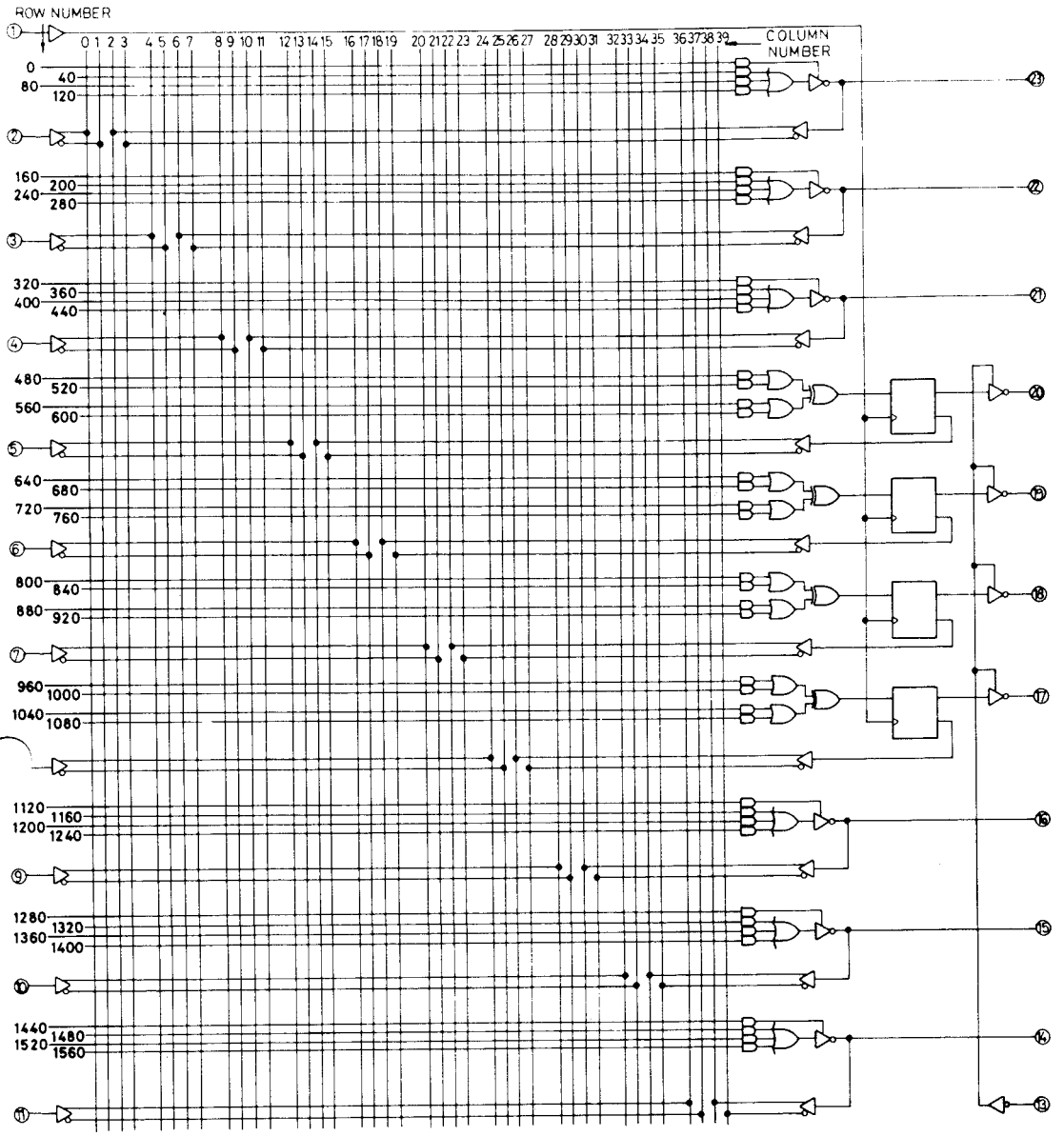
FUSE NUMBER DIAGRAM 20R8



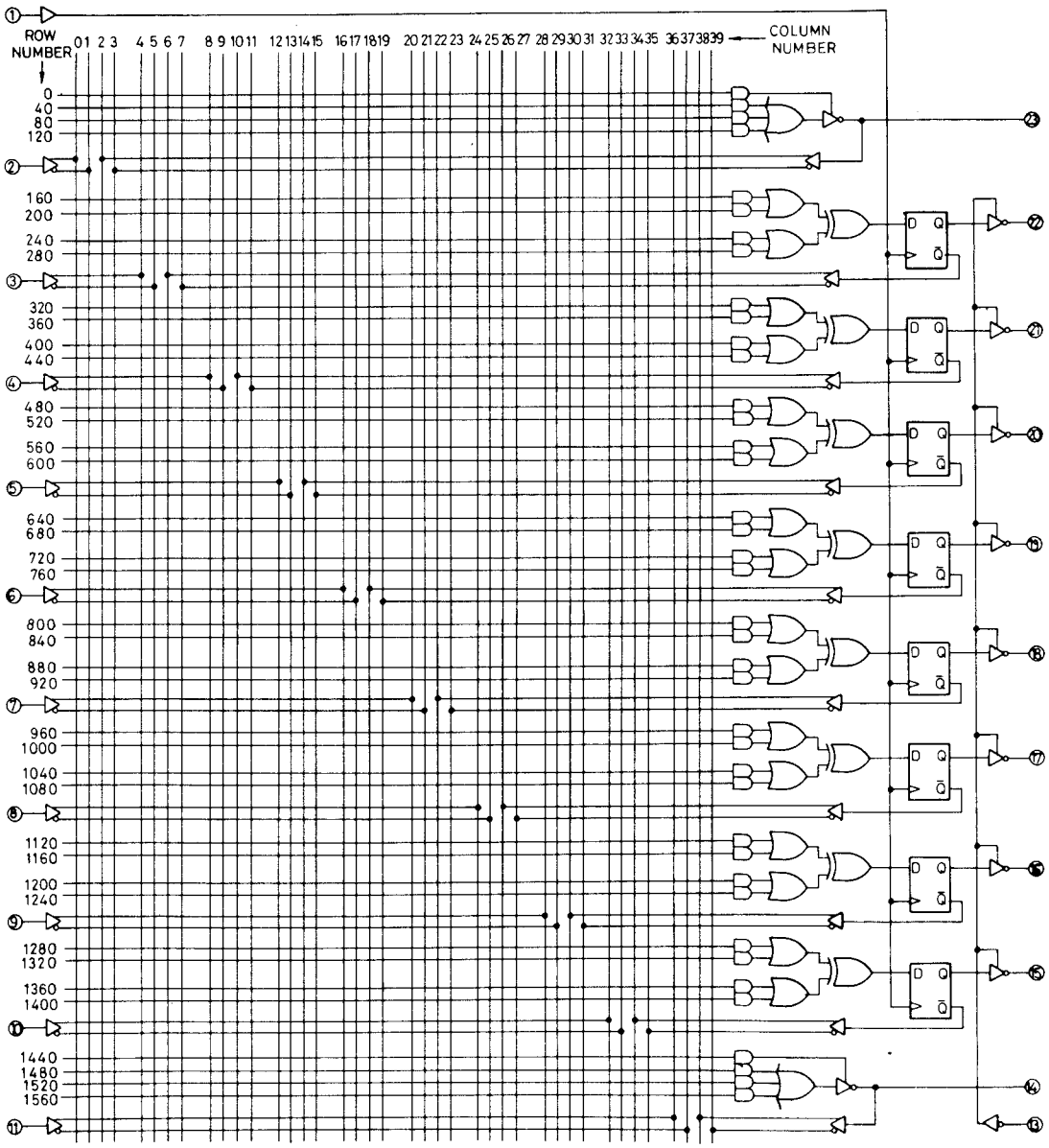
FUSE No = COLUMN No + ROW No.

FUSE NUMBER DIAGRAM 20 x 4

FUSE NUMBER DIAGRAM 20 X 4

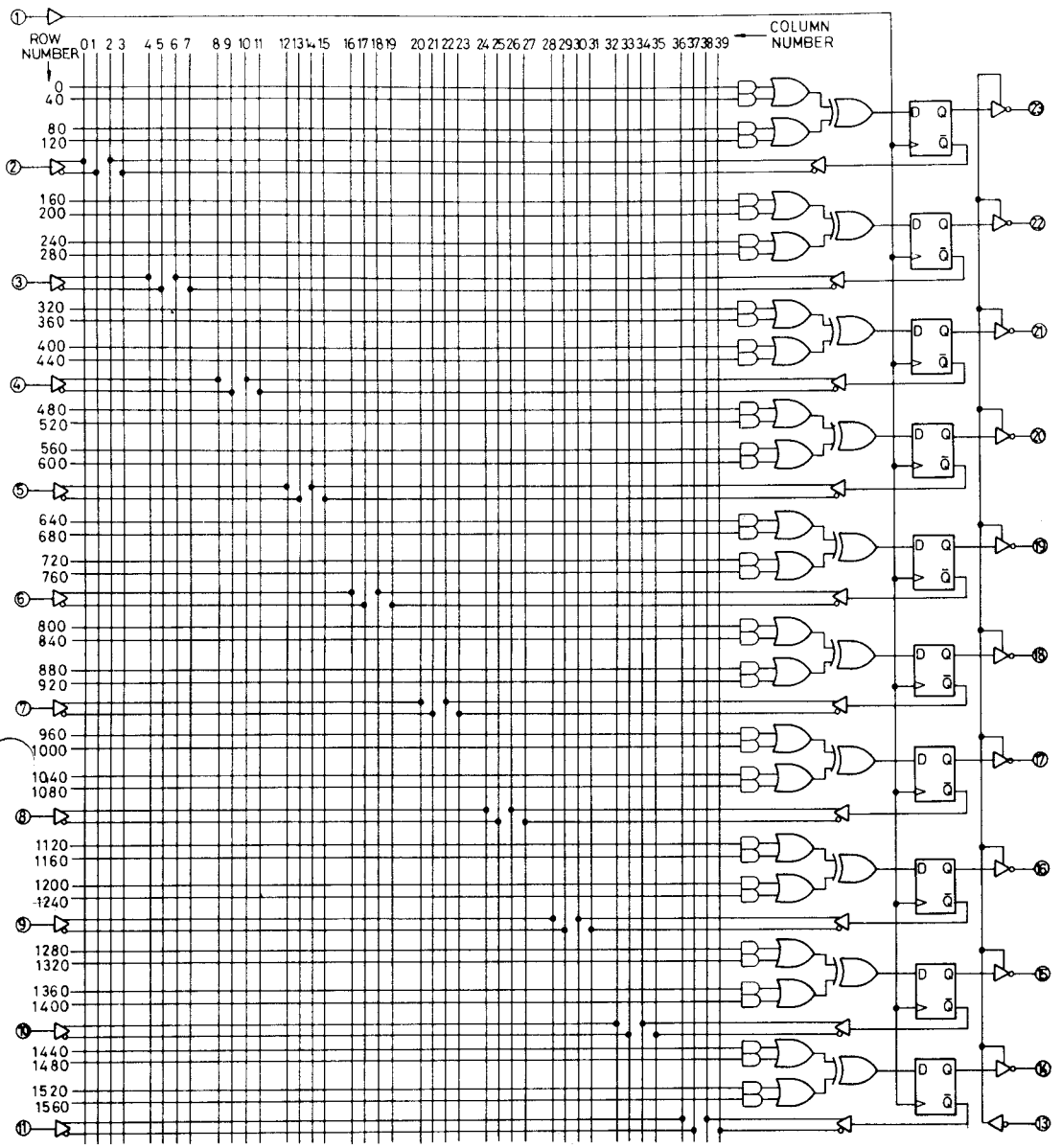


FUSE No. = COLUMN No. + ROW No.

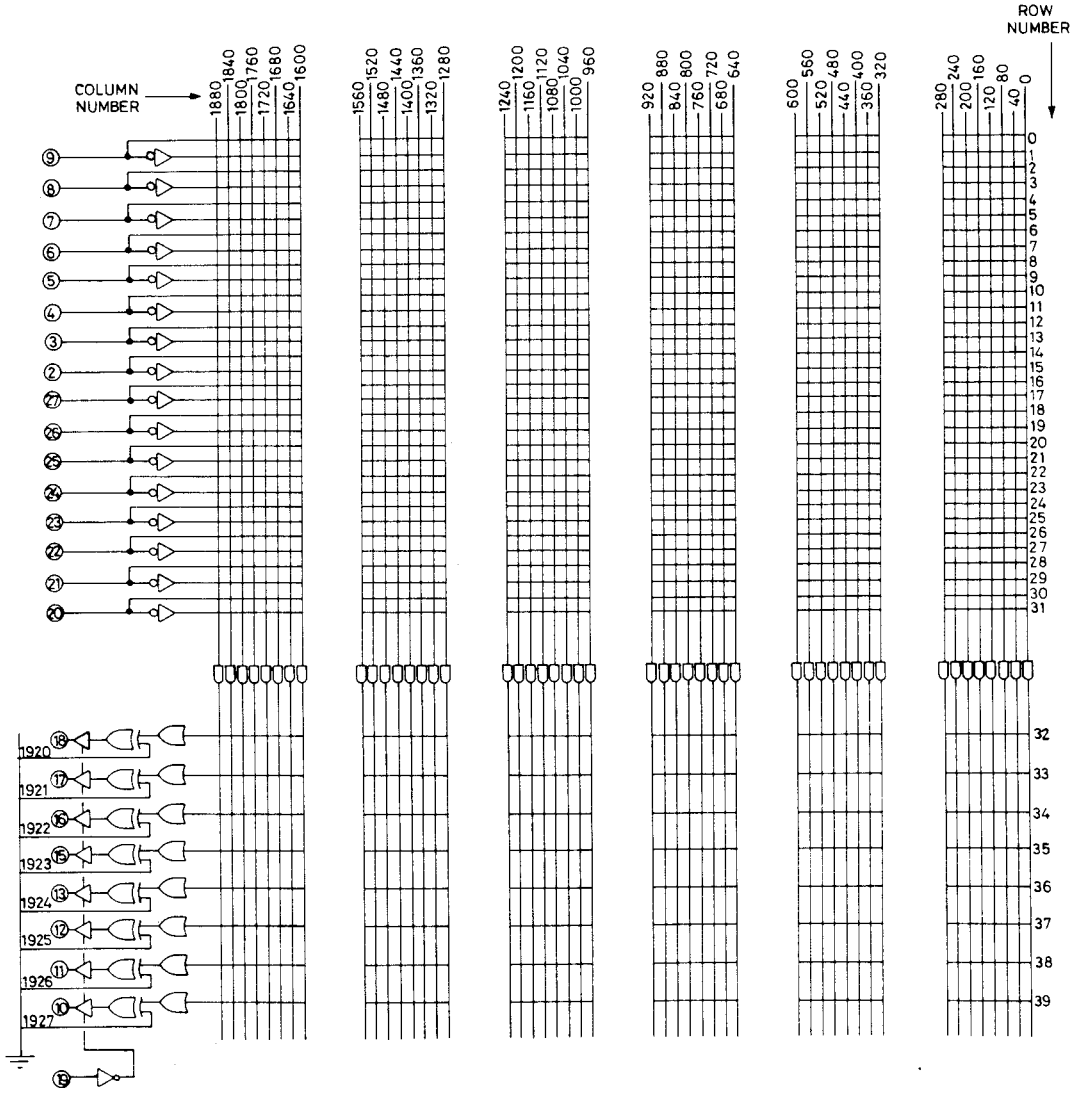


FUSE No = COLUMN No • ROW No

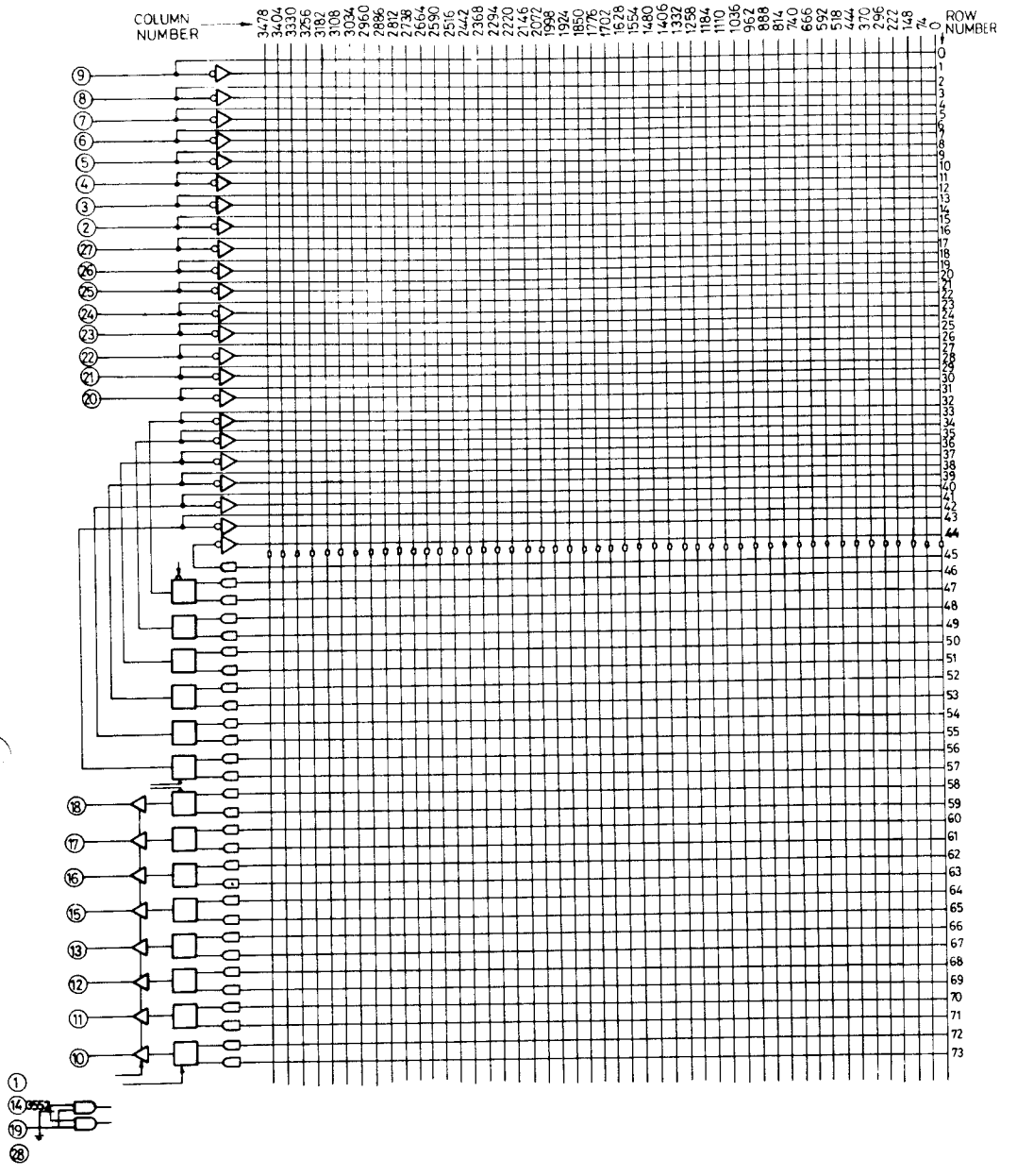
FUSE NUMBER DIAGRAM 20 x 10



FUSE No. = COLUMN No. + ROW No.

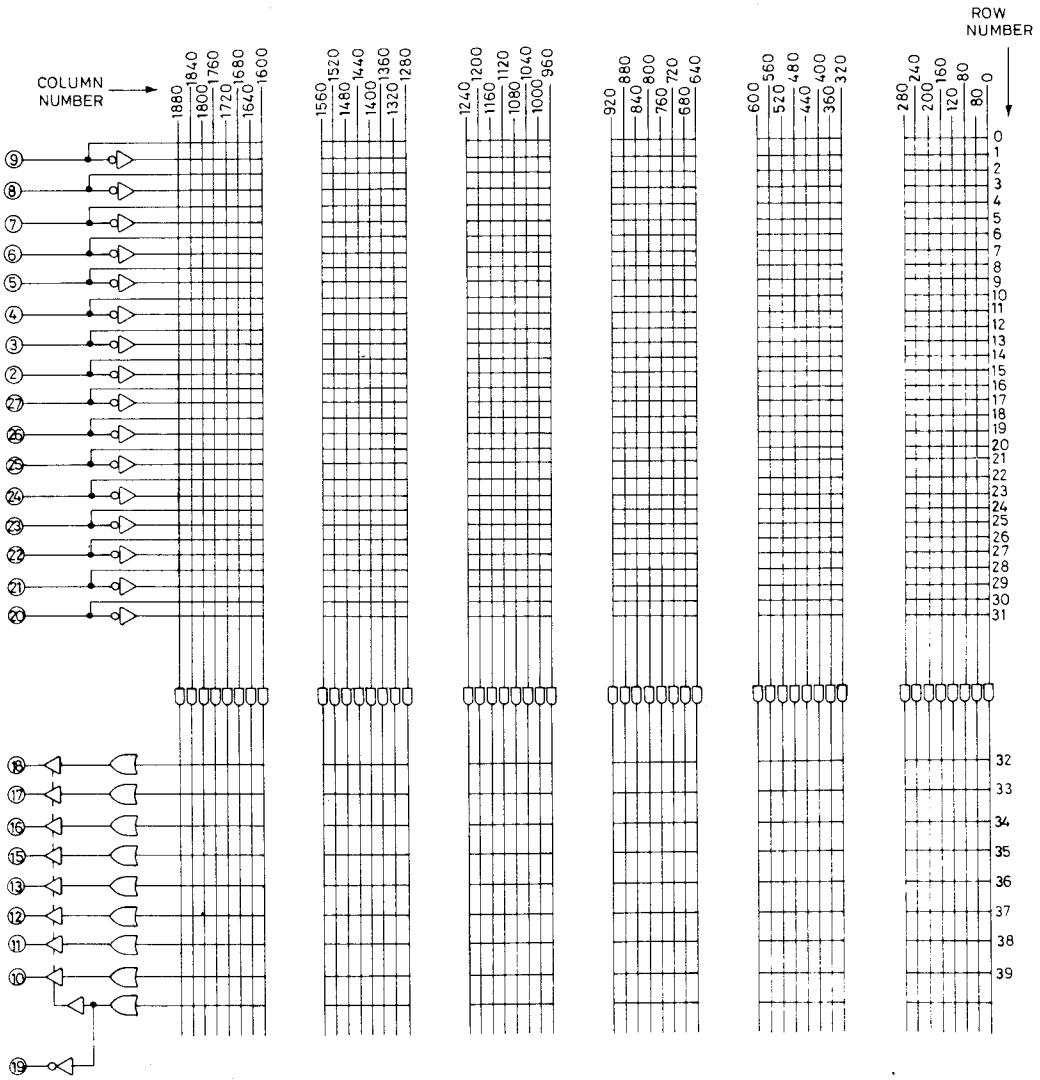

COLUMN NUMBER
ROW NUMBER
1880 1840 1800 1760 1720 1680 1640 1600
1560 1520 1480 1440 1400 1360 1320 1280
1240 1200 1160 1120 1080 1040 1000 960
920 880 840 800 760 720 680 640
600 560 520 480 440 400 360 320
280 240 200 160 120 80 40 0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
9 8 7 6 5 4 3 2 27 26 25 24 23 22 21 20
1920 19
1921 17
1922 16
1923 15
1924 13
1925 12
1926 11
1927 10
18

FUSE NUMBER DIAGRAM SERIES 28 FPLS



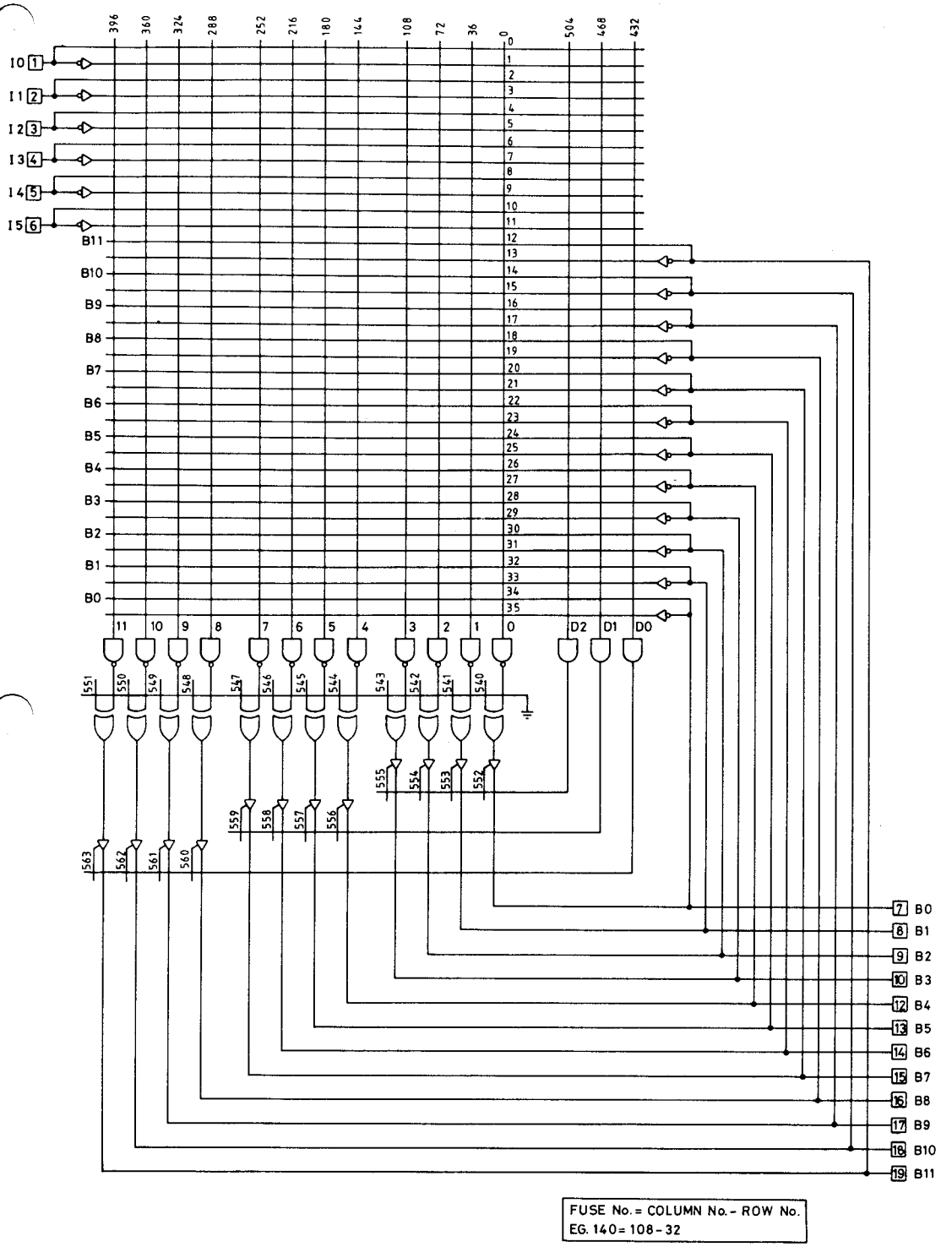
FUSE No. = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM SERIES 28 FPRP



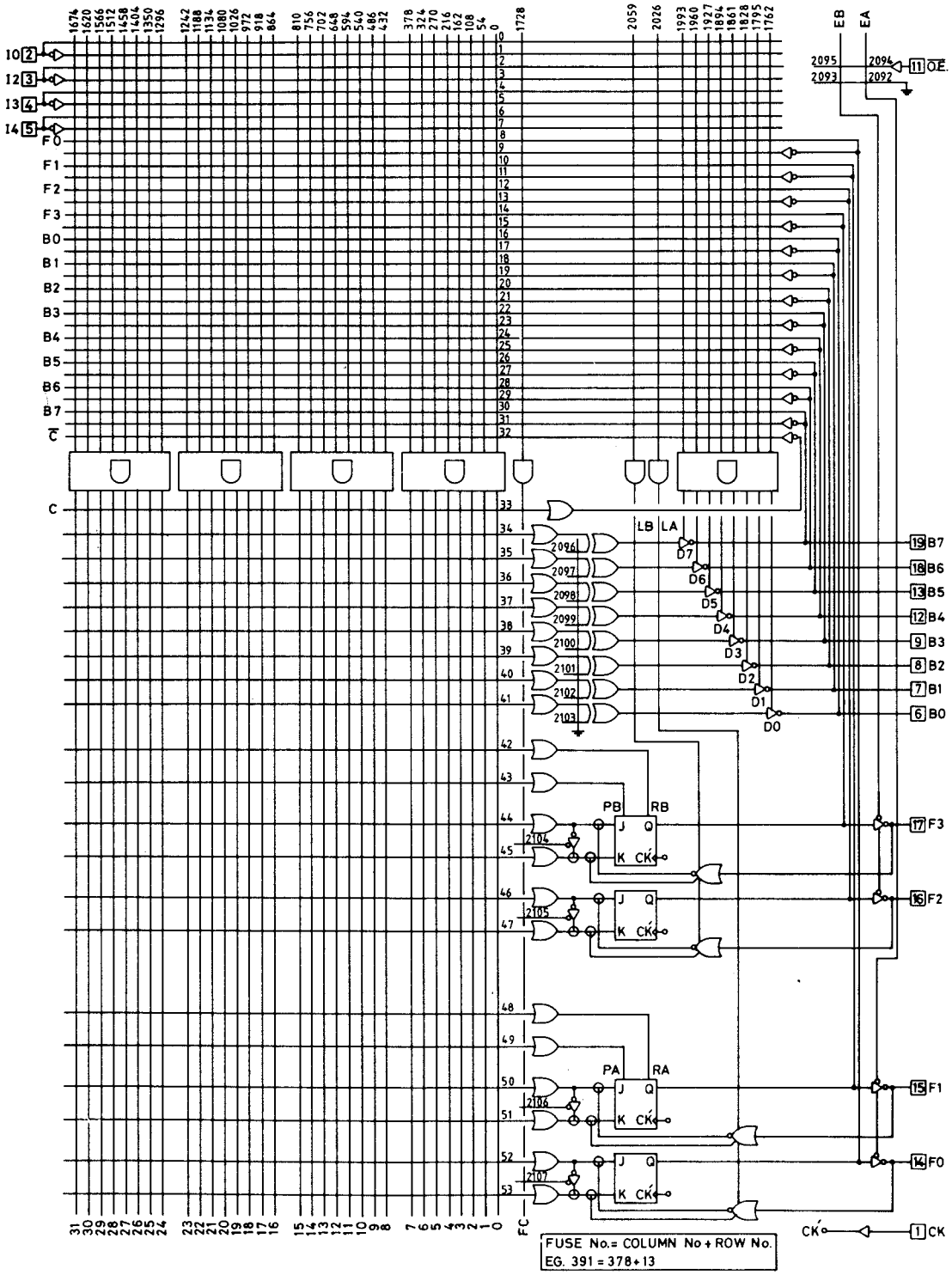
FUSE No = COLUMN No + ROW No

82S150/1 FPGA LOGIC DIAGRAM SHOWING JEDEC FUSE NUMBERS

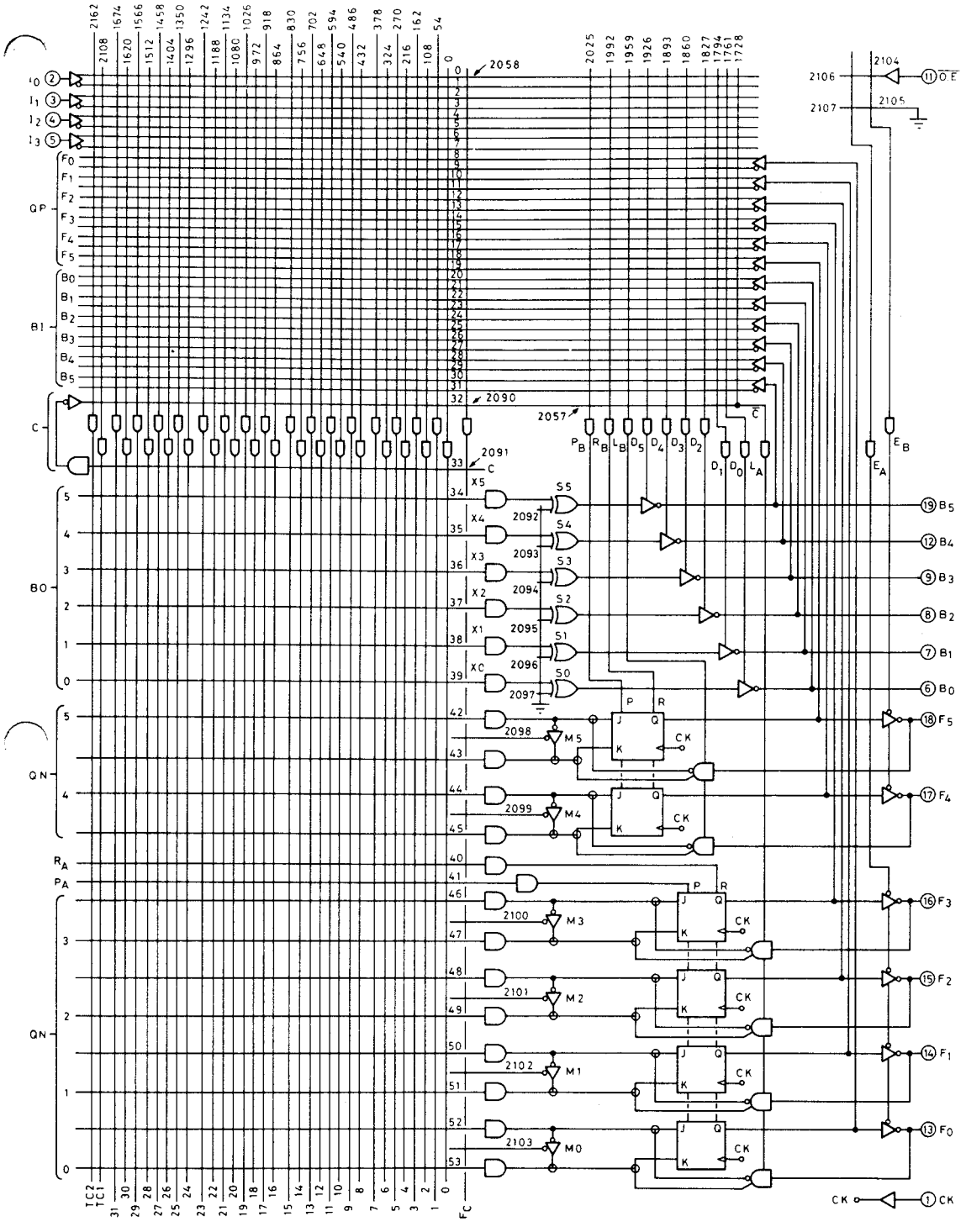


FUSE No. = COLUMN No. - ROW No.
EG. 140 = 108 - 32

82S154/5 FPLS LOGIC DIAGRAM SHOWING JEDEC FUSE NUMBERS



FUSE NUMBER DIAGRAM 82S 157



NOTES :-

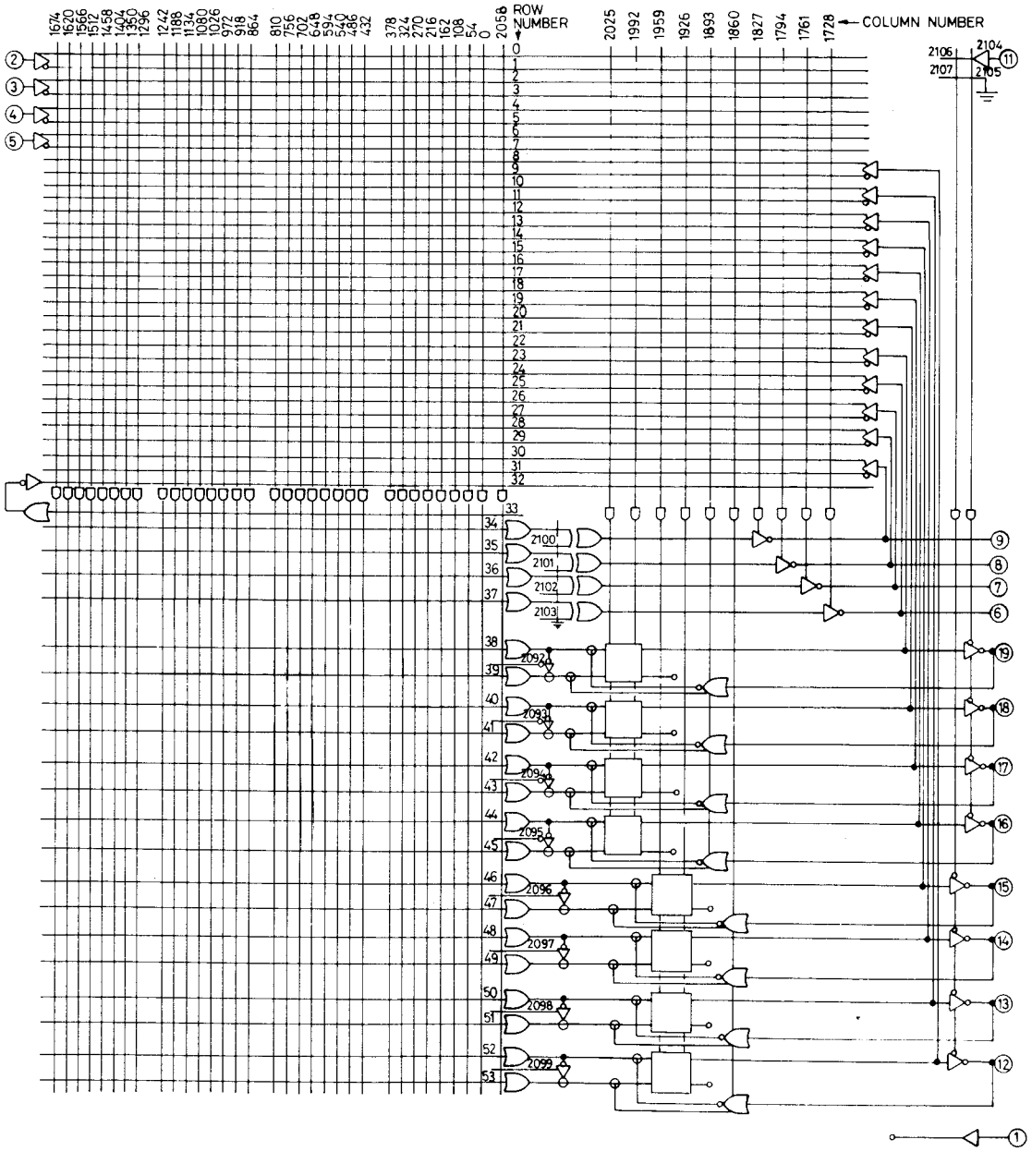
1 ALL 'OR' GATES 'R' GATE INPUTS WITH A BLOWN LINK FLOAT TO LOGIC "0"

2. ALL OTHER GATHER GATES AND CONTROL INPUTS WITH A BLOWN LINK FLOAT TO LOGIC "1"

3 PROGRAMMABLE CONNECTION

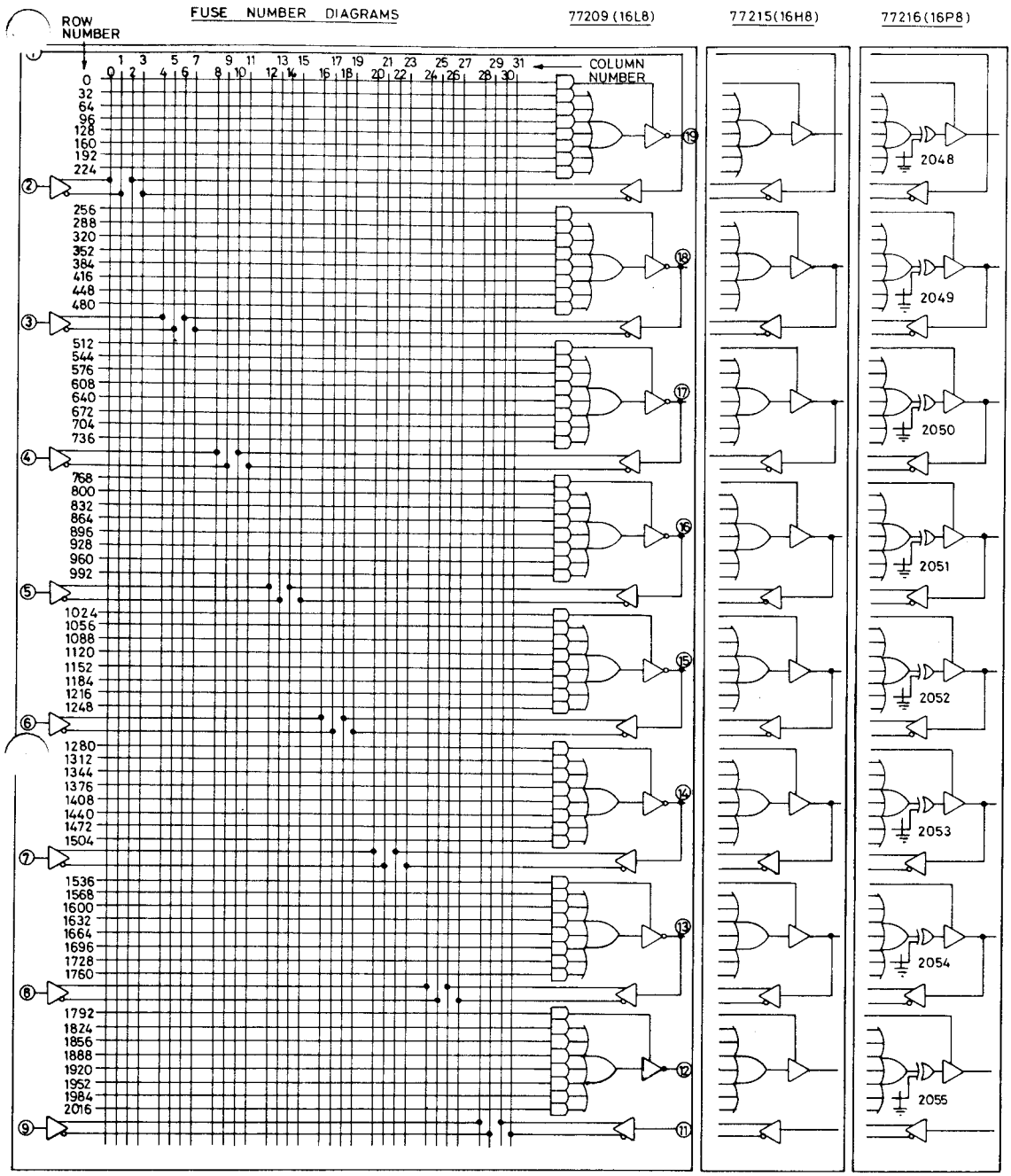
4 ⊕ DENOTES 'NOTES' WIRE-OR'

FUSE No. = ROW No. + COLUMN No.



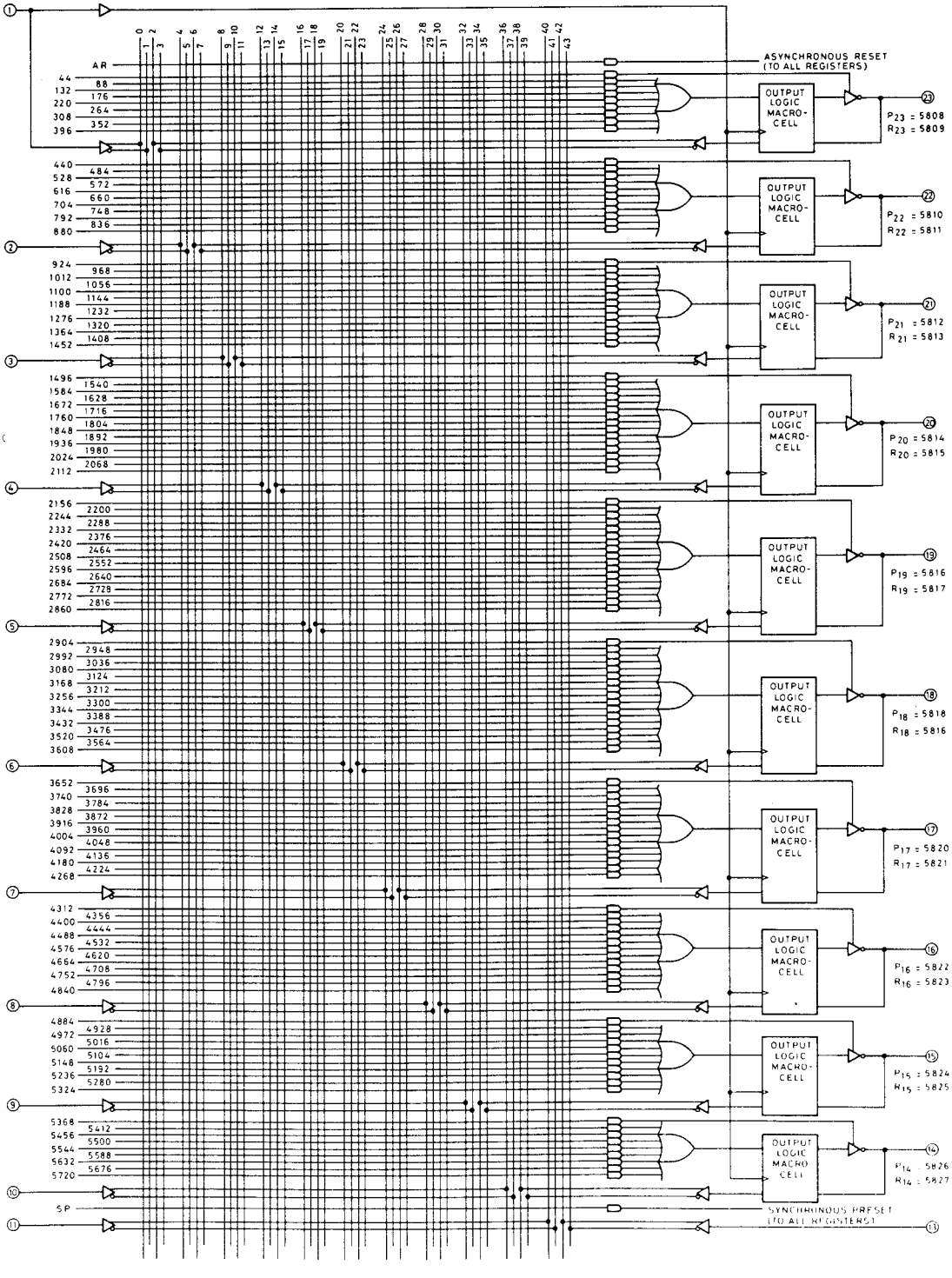
FUSE No = COLUMN No + ROW No.

FUSE NUMBER DIAGRAMS



FUSE No. = COLUMN No. + ROW No.

FUSE NUMBER DIAGRAM AmPAL22V10



FUSE No = COLUMN No + ROW No