technical memorandum Daresbury Laboratory

DL/CSE/TM28

USE OF PROGRAMMABLE ARRAY LOGIC IN CAMAC MODULES

by

I. SUMNER, Daresbury Laboratory.

NOVEMBER, 1983

637ACO

Science & Engineering Research Council

Daresbury Laboratory

Daresbury, Warrington WA4 4AD

CONTENTS

		Page
1.	INTRODUCTION	2
2.	PAL ARCHITECTURE	2
3.	PALASM	3
4.	CAMAC DECODER	6
5.	CAMAC DECODER PALASM LISTING	8
6.	CAMAC MASK REGISTER	9
7.	CAMAC MASK REGISTER PALASM LISTING	10
8.	CONCLUSIONS	12

1. INTRODUCTION

Programmed Array Logic (PAL) integrated circuits provide a way of designing random logic into a "Universal" integrated circuit.

These circuits are easy to program and non-volatile.

PALs can be used to advantage to replace several separate integrated circuits in decoding logic. They can provide a 6 to 1 chip count reduction over random logic. They have glitch free operation as an advantage over Programmed Read Only Memory (PROM) based look-up table decoders.

For use in CAMAC modules, functions that immediately come to mind are a complete CAMAC decoder in one IC and a bit set/bit clearable LAM mask and demand register in one IC.

2. PAL ARCHITECTURE

This section is a brief description of the internal layout of PAL ICs, for a more complete description the reader is referred to the PROGRAMMABLE ARRAY LOGIC HANDBOOK from Monolithic Memories and the PAL DATABOOK from NATIONAL SEMICONDUCTOR.

The PAL architecture is based on AND gates with their inputs fuse programmable to the true and inverse of up to 20 input and feedback terms. The outputs of the AND gates are then grouped into OR terms. Figure 1 shows a partial logic of 2 inputs 2 AND gates and 1 OR gate.

The notation used in PALS is shown in fig.2 which shows the conventional symbol for a three input AND gate and also how this is represented in PAL notation. Figure 3 shows the PAL notation for the partial logic shown in fig.1.

Figure 4 shows a random logic function and fig.5 shows how this function is represented on a PAL logic diagram. The intact fuses are represented by X at the junction and blown fuses are represented by a plain crossover. It is therefore easy to follow the logic by following the

inputs through to the outputs they control.

It is now possible to use the PAL notation to show the architecture of the various types of PAL.

Figure 6 shows a section of a gate array PAL. In this type of PAL the inputs are buffered and inverted and bussed to the inputs of all the AND gates such that each AND gate can see each input. The AND gates are then grouped to OR functions so that each output can be activated by several input combinations. There are a wide range of these PALs with various combinations of inputs and outputs. To select a PAL from this range to replace random logic it is normally only necessary to count the number of input and output connections and select the PAL accordingly.

Figure 7 shows a section of the next most complex type of PAL. This type has a three-state output controlled by one AND gate (note this can be permanently enabled by blowing all the fuses on the input to the control gate). The output pin is also connected back into the array thus allowing feedback, input or bi-directional use of this terminal.

Figure 8 shows the registered PAL. This type of PAL has an externally clocked register and feedback. These PALs have their registers clocked simultaneously and most of them have separate non registered outputs that can be externally connected to give a clock from several sources.

Figure 9 shows the XOR registered PAL which is a variation on the registered PAL with 2 OR functions XORed before the register. This allows an easy implementation of a HOLD function.

3. PALASM

PALASM is a FORTRAN IV program which assembles the PAL Design Specification translating the logic equation to a PAL fuse pattern.

PALASM also contains a simulator which exercises the Function Table vectors and reports any inconsistencies between the table and the

equations.

The Function Tables also form the basis for a test program for the blown PAL.

PALASM is incorporated in a small microprocessor based PAL programmer the SD20/24. This can be used to develop PAL specifications as well as to blow the PAL.

A PALASM program consists of three or four parts. These are: 1 setup conditions, 2 the Boolean logic equations, 3 a Function Table and, 4 a description. Stages 3 and 4 are optional but for use with the SD20/24 one or other must be present.

The setup is five lines long. The first line is the PAL type number. The next three lines are the part number, the part title and the programmers name and location. The 5th line is the pin name list. The pin list is a sequence of symbolic names separated by one or more spaces on one or more lines in order of the device pin numbers. Each symbolic name is unique. All pins including power and ground must be named. Names may use any printable character except the operators: "*+/()". The prefix "/", may be used to logically complement the name.

The next part of the program starts here and continues until the words FUNCTION TABLE or DESCRIPTION are encountered. This section consists of the Boolean equations expressing the transfer function of the device. These equations are of one of three forms.

1. SYMBOL=EXPRESSION
2. IF(PRODUCT)SYMBOL=EXPRESSION
3. SYMBOL:=EXPRESSION

Equation 1 is used for gate array PALs. Equation 2 is used for GATE array PALs with Three-State outputs. Equation 3 is used for the registered outputs of registered PALs.

The terms are defined:

SYMBOL Pin name with optional prefix,"/".

PRODUCT A sequence of SYMBOLS separated by the AND operator,"*".

IF Tri-State operator.

EXPRESSION A sequence of SYMBOLS separated by operators.

OPERATORS

=	Equality, sum of products.
:=	Becomes after the low to high clock transition (for use with registered outputs).
/	Complement, operator prefix to variable name.
*	AND operator.
+	OR operator.
:+:	XOR operator.
:*:	XNOR operator.
;	Comment prefix. All characters following ";" are ignored until carriage return.

The program now continues with the optional Function Table.

The Function Table begins with the key word FUNCTION TABLE. It is followed by a pin list which may contain any number of SYMBOLS which may be in a different order and polarity from the pin list in line 5. VCC and GND cannot be listed. The pin list is followed by optional comments (;XXXXXX), then a dashed line; e.g., --------- (length optional), which in turn is followed by a list of vectors, one vector per line, each vector containing one CONSTANT for each item in the pin list. CONSTANTS are optionally separated by spaces and followed by an optional comment.

Definition of Function Table States:

H High level

L Low level

X Irrelevant

C Transition from low level to high level (clock)

Z Off (HI-Z)

The vector list is followed by another dashed line.

The final part of the program is again optional. It is started with the key word DESCRIPTION. The device operation and application are

described here.

For use with the SD20/24 the file then closes with a control-z.

4. CAMAC DECODER

One problem with CAMAC is the large number of command and timing lines. There are a total of 14 command lines input to a normal CAMAC module; these are 5 function lines, 4 sub-address lines, 3 timing lines (BUSY S1 S2), 1 module address line and an initialise line. Handling these is less of a problem to PAL decoders than to other deocder systems.

Choosing a suitable PAL for a CAMAC decoder is tackled by considering the operation. First the decode is a gating function so that gate array PALs are suitable and registered PALs are not suitable. Then there have to be 14 inputs so examination of the available PAL types shows that PAL14H4, PAL14L4, and PAL16L8 PALs in the 20 pin range to be suitable. The PAL16H2, PAL16L2 and PAL16C2 types would have unused inputs which wastes pins. However it will be seen that there can only be four output pins left with a 20 pin device.

If we now look at the 24 pin PALs we find 3 possible devices; PAL14L8, PAL20L8 and PAL20L10. These can give 8 independant outputs. If more than 8 outputs are needed it is possible to connect more than one PAL to the same inputs thus increasing the number of outputs by connecting two or more PAL's in parallel.

The following example is a CAMAC decode PAL for a module that responds to the following commands:

RD1 A00

WT1 A00

WT1 A01

WT2 A00

WT2 A01

WT2 A02

XEQ A15

Z.S2

The five write commands set up various output registers and control registers, the read command reads out one of the registers and the XEQ command initialises the registers as does the Z command.

All commands respond with an X response.

It will be seen that 8 separate outputs are required, but that only the X response and initialise outputs are required to decode more than one function. Apparently the X response requires 7 separate functions decoding, however examination of the functions shows several that differ by only one bit setting. These can be decoded by the same AND function with the differing bit being, "do not care" i.e. both fuses being blown. In practice four functions are decoded by one AND function. These are the WT1 A00, WT1 A01, WT2 A00, WT2 A01 commands, examination will show that these differ only in the setting of the F1 and A1 lines, therefore not decoding these lines allows the one AND function to decode all four of these functions for the X response. The other functions do not have such single bit commonality and thus must be decoded separately, therefore requiring four OR terms to decode the X response.

Thus the PAL requires 14 input pins, 8 output pins, 1 output term to have 4 OR functions, 1 output term to have 2 OR functions and the remaining 6 terms to have one function. Examining the specifications in the PAL handbook the PAL14L8 is seen to be suitable.

The PALASM program is shown together with a function table to test the blown PAL.

5. CAMAC DECODER PALASM LISTING

```
PAL14L8
PART 1
CAMAC DECODER
SERC Daresbury  Ian Sumner  20/05/83

N B S1 S2 Z A1 A2 A4 A8 F1 F2 GND
F4 F8 INIT WT10 WT11 WT20 WT21 WT22 RD10 X F18 VCC
/INIT =  /S2  */Z
         ;Z Initialise
         +/F16 */F8 * F4 * F2 */F1 */A8 */A4 */A2 */A1 */S2 */B */N
         ;OR XEQ A15
/WT10 =   /F16 * F8 * F4 * F2 * F1 * A8 * A4 * A2 * A1 */S1 */B */N
         ;WT1 A00 pulses low at S1 time
/WT11 =   /F16 * F8 * F4 * F2 * F1 * A8 * A4 * A2 */A1 */S1 */B */N
         ;WT1 A01 pulses low at S1 time
/WT20 =   /F16 * F8 * F4 * F2 */F1 * A8 * A4 * A2 * A1 */S1 */B */N
         ;WT2 A00 pulses low at S1 time
/WT21 =   /F16 * F8 * F4 * F2 */F1 * A8 * A4 * A2 */A1 */S1 */B */N
         ;WT2 A01 pulses low at S1 time
/WT22 =   /F16 * F8 * F4 * F2 */F1 * A8 * A4 */A2 * A1 */S1 */B */N
         ;WT2 A02 pulses low at S1 time
/RD10 =    F16 * F8 * F4 * F2 * F1 * A8 * A4 * A2 * A1 */B */N
         ;RD0 A00 output low for whole of CAMAC cycle
/X    =   /F16 * F8 * F4 * F2 * A8 *A4 * A2 */B */N
         ;X response for WT2 A00, Wt2 A01, WT1 A00, WT1 A02
         +/F16 * F8 * F4 * F2 */F1 * A8 * A4 * /A2 * A1 */B */N
         ;OR WT2 A02
         +/F16 */F8 * F4 * F2 */F1 */A8 */A4 */A2 */A1 */B */N
         ;OR XEQ A15
         + F16 * F8 * F4 * F2 * F1 * A8 * A4 * A2 * A1 */B */N
         ;OR RD1 A00
```

```
FUNCTION TABLE
F16 F8 F4 F2 F1 A8 A4 A2 A1 S2 S1 B N Z WT10 WT11 WT20 WT21 WT22 RD10 X INIT
;FFFFF AAAA SS BNZ  WWWWWRXI
;18421 8421 21      TTTTTD N
;6                  112221 I
;                   010120 T
----------------------------
 HHHHH HHHH LH HHL  HHHHHHHL    INITIALISE Z.S2
 XXXXX XXXX HX XHH  HHHHHHHH    NOT ADDRESSED NO ACTION
 XXXXX XXXX HX HXH  HHHHHHHH    NO B NO ACTION
 HHHHH HHHH XX LLH  HHHHHLLH    RD1 ADD
 LHHHH HHHH XH LLH  HHHHHHLH    WT1 ADD X RESPONSE
 LHHHH HHHH XL LLH  LHHHHHLH    WT1 ADD S1
 LHHHH HHHL XH LLH  HHHHHHLH    WT1 A01 X RESPONSE
 LHHHH HHHL XL LLH  HLHHHHLH    WT1 A01 S1
 LHHHL HHHH XH LLH  HHHHHHLH    WT2 A00 X RESPONSE
 LHHHL HHHH XL LLH  HHLHHHLH    WT2 A00 S1
 LHHHL HHHL XH LLH  HHHHHHLH    WT2 A01 X RESPONSE
 LHHHL HHHL XL LLH  HHHLHHLH    WT2 A01 S1
 LLHHL LLLL HX LLH  HHHHHHLH    XEQ A15 X RESPONSE
 LLHHL LLLL LX LLH  HHHHHHLL    XEQ A15 S2 INITIALISE
----------------------------
```

NOTE CAMAC LINES are active low therefore the input lines are inverse logic.

Figure 10 shows the resulting fuse pattern from this program.

The numbers at the end of the fuse matrix lines are used to address the matrix during programming and are automatically generated by the PALASM programme.

6. CAMAC MASK REGISTER

This example is a CAMAC LAM mask and demand register which can be bit set and bit cleared, the mask read and the demand pattern read whilst it gives a demand output.

As the example contains a register it is obvious that a registered PAL is required. The example chosen has three maskable interrupt sources, so the register must be at least three bits long. The demand pattern must also be three bits long, requiring three non registered outputs. The demand output also requires a non registered output, so there must also be four non-registered outputs. There need to be eight inputs and a clock, there are the two control bits SET and INITIALISE the three write bits and the three condition bits. These requirements are fulfilled by a PAL16R4.

The CAMAC decoder provides a clock at S1 time of every appropriate CAMAC command (bitset, bitclear and initialise). It also provides two control lines INIT and SET which are low active for the duration of the Initialise and Bitset commands respectively, Bitclear is decoded by default.

On examination of the logic diagram of the PAL it will be seen that the register outputs are fedback into the logic array as well as being three-state buffered as outputs. The three-state outputs are obvious outputs for reading the LAM mask. The other outputs are then available for reading the requests and the LAM output.

The PALASM program for the resister is shown below.

7. PALASM LISTING FOR CAMAC MASK REGISTER

```
     PAL16R4
     PART 2
     CAMAC LAM MASK AND REQUEST REGISTER
     SERC Daresbury  Ian Sumner  20/05/83
CLKMASK W1 W2 W3 SET INIT DM1 DM2 DM3 GND /READM LAM R3 I4 M3 M2 M1 R2 R1 VCC
/M1 := /INIT
       + W1 * SET
       +/W1 */M1
/M2 :-  /INIT
       + W2 * SET
       +/W2 */M2
```

```
/M3 :-  /INIT
       + M3 * SET
       +/M3 */M3
IF (VCC) /R1 = /M1 + DM1
IF (VCC) /R2 = /M2 +/DM2
IF (VCC) /R3 = /M3 +/DM3
IF (VCC)/LAM = /R1 */R2 */R3

FUNCTION TABLE
CLKMASK INIT SET W1 W2 W3 DM1 DM2 DM3 READM R1 R2 R3 M1 M2 M3 LAM
;--INPUTS--  OUTPUTS
;CISWWWDDDR RRRMMML
;LNE123MMME  123123A
;KIT   123A        M
;MT       D
;A        M
;S
;K                      COMMENTS
-----------------------------------
 CLXXXXHLLL  LLLLLLL    INITIALISE
 HLHLLLHLLH  LLLZZZL    TRI STATE THE OUTPUTS
 HLHLLLLHHL  LLLLLLL    CHECK NO DEMANDS
 CHLLLLLHHL  LLLLLLL    SET BUT NO W LINES SO MASK STAYS CLEAR
 CHLHLLLHHL  HLLHLLH    SETS M1 R1 AND LAM
 CHLLLLHLLL  LLLHLLL    LEAVES MASK SET
 CHHLLLLHHL  HLLHLLH    LEAVES MASK SET + LAM
 CHHHLLLHHL  LLLLLLL    CLEARS M1
 CHLLHLLHHL  LHLLHLH    SETS M2 R2 AND LAM
 CHLLLLHLLL  LLLLHLL    LEAVES MASK SET
 CHHLLLLHHL  LHLLHLH    LEAVES MASK SET + LAM
 CHHLHLLHHL  LLLLLLL    CLEARS M2
 CHLLLHLHHL  LLHLLHH    SETS M3 R3 AND LAM
 CHLLLLHLLL  LLLLLHL    LEAVES MASK SET
 CHHLLLLHHL  LLHLLHH    LEAVES MASK SET + LAM
 CHHLLHLHHL  LLLLLLL    CLEARS M3
-----------------------------------
NOTE  The outputs are all active low
      The WRITE and DM1 inputs are active high
```


The DM2, DM3, SET and INIT inputs are active low

Figure 11 shows the fuse pattern resulting from this program

The INIT line unconditionally clears the mask at the clock time. The SET line (corresponding to bitset and bitclear) together with the associated write line controls the outcome at other clock times. If the write line is active (high), the set line controls the mask register input, setting or clearing as appropriate. If the write line is low the mask register is fedback on itself, such that it stays in the same state.

8. CONCLUSIONS

The examples above show areas where CAMAC modules can have their IC count significantly reduced by comparison with conventional random logic approaches, thus saving valuable board area.

PAL design has the advantage over PROM based decoders of only decoding the inputs relevant to each output, thus eliminating glitches due to an irrelevant input signal sequence. The PAL will also have more input terms than a PROM, of equivalent speed and power dissipation.

PALs being universal parts, that are programmed before use, will have a large fraction of unused internal components. The effect of this is a high power dissipation. However, it must be borne in mind that this dissipation in one IC has to be set against the power dissipated in the several ICs that it has replaced. PROMs also suffer from a high power dissipation for the same reason.

PALs are available in several families each with a different speed/power trade off. If it is assumed that a PAL replaces 2 to 3 gate layers of random logic then the standard family is comparable to LS TTL logic for propogation delays.

These examples are just two of many areas where PALs can beneficially be used to reduce chip count and complexity in logic designs. One area of interest is in replacing the random logic "glue" that holds circuits

together. Another area of interest is in designing simple sequencers for the control of units.

The Laboratory has a Structured Design SD20/24 PAL programmer which allows a designer to write a PAL specification, store it on a tape cartridge and to blow the fuse pattern into a PAL.

FIGURE CAPTIONS

Fig.1 Basic PAL architecture

Fig.2 Logical element and equivalent PAL schematic

Fig.3 Schematic for unblown PAL shown in Fig.1

Fig.4 Random logic implementation of exclusive or function

Fig.5 PAL implementation of exclusive or function using PAL in Fig.1

Fig.6 Partial schematic for a gate array PAL

Fig.7 Partial schematic for a gate array PAL with feedback

Fig.8 Partial schematic for a registered PAL

Fig.9 Partial schematic for an exclusive OR registered PAL

Fig.10 PAL schematic for the CAMAC decoder

Fig.11 PAL schematic for the CAMAC mask register

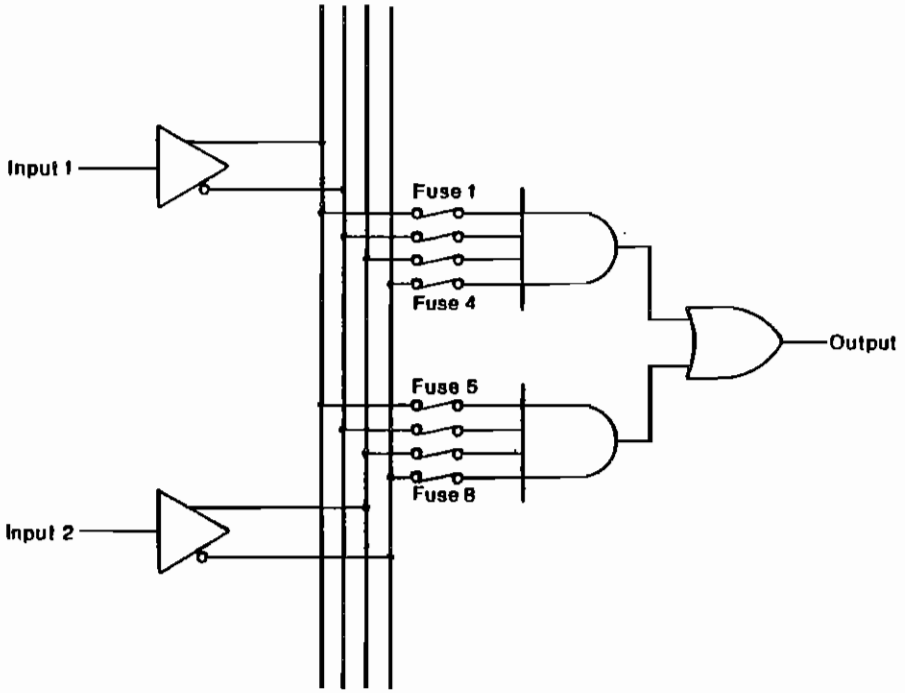

Fig. 1

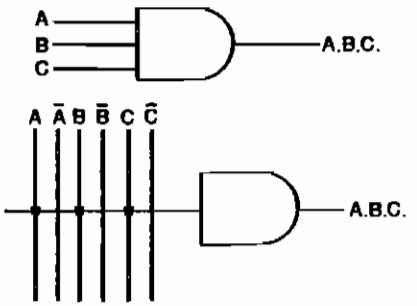

Fig. 2

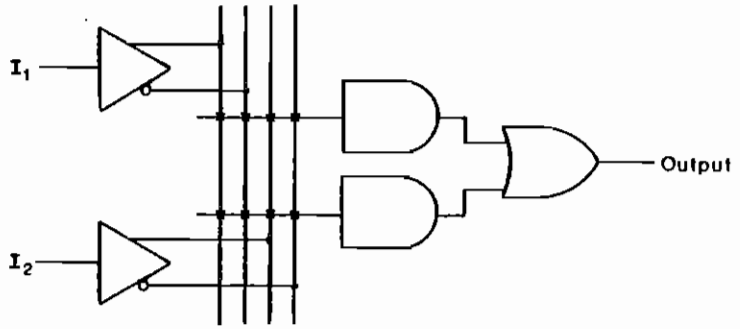

Fig. 3

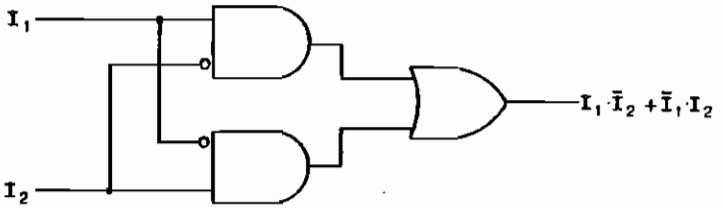

Fig. 4

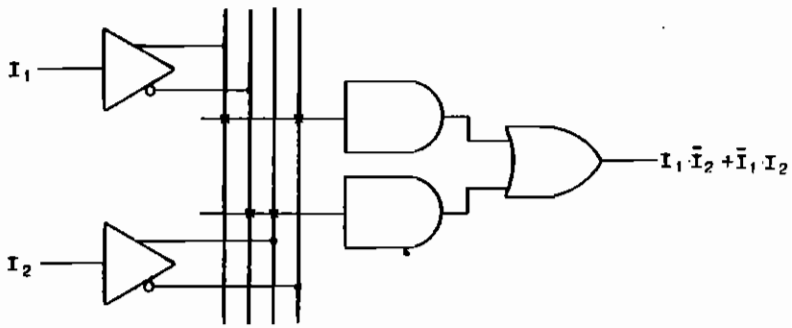

Fig. 5

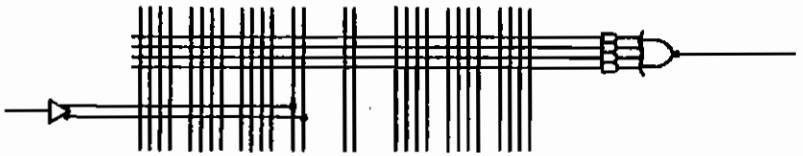

Fig. 6

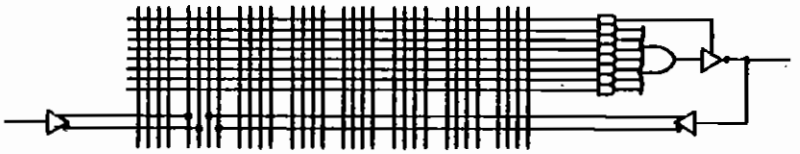

Fig. 7

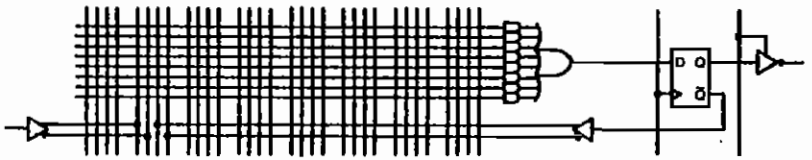

Fig. 8

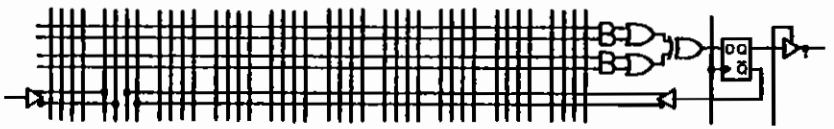

Fig. 9

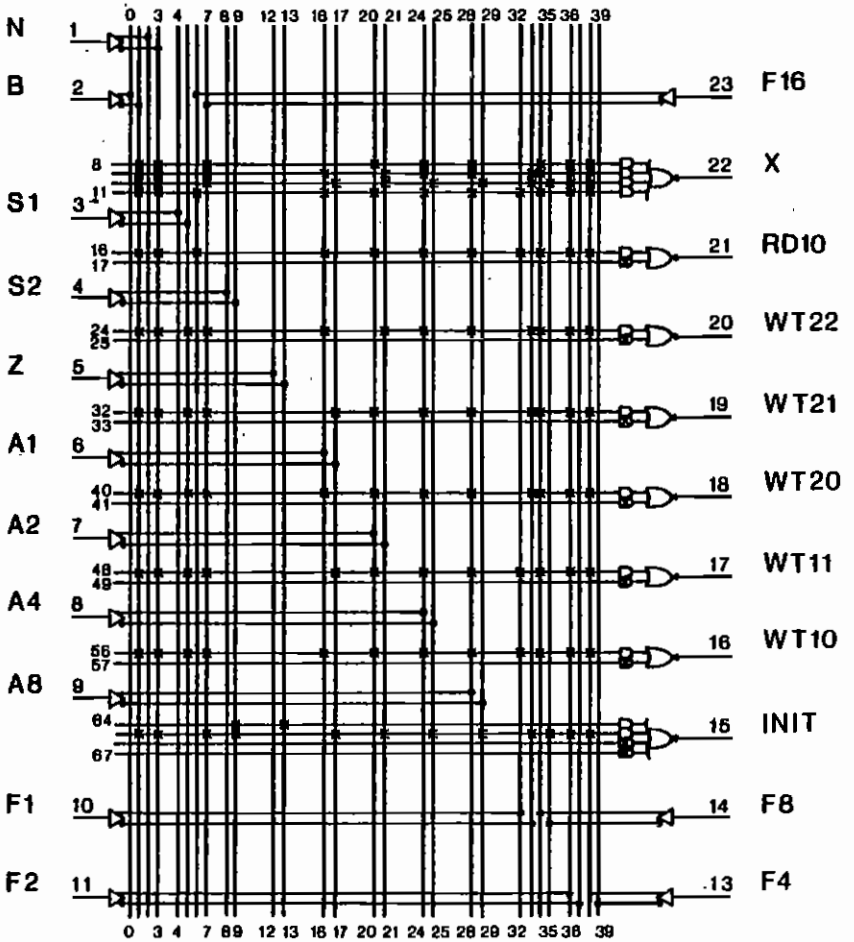

Fig.10

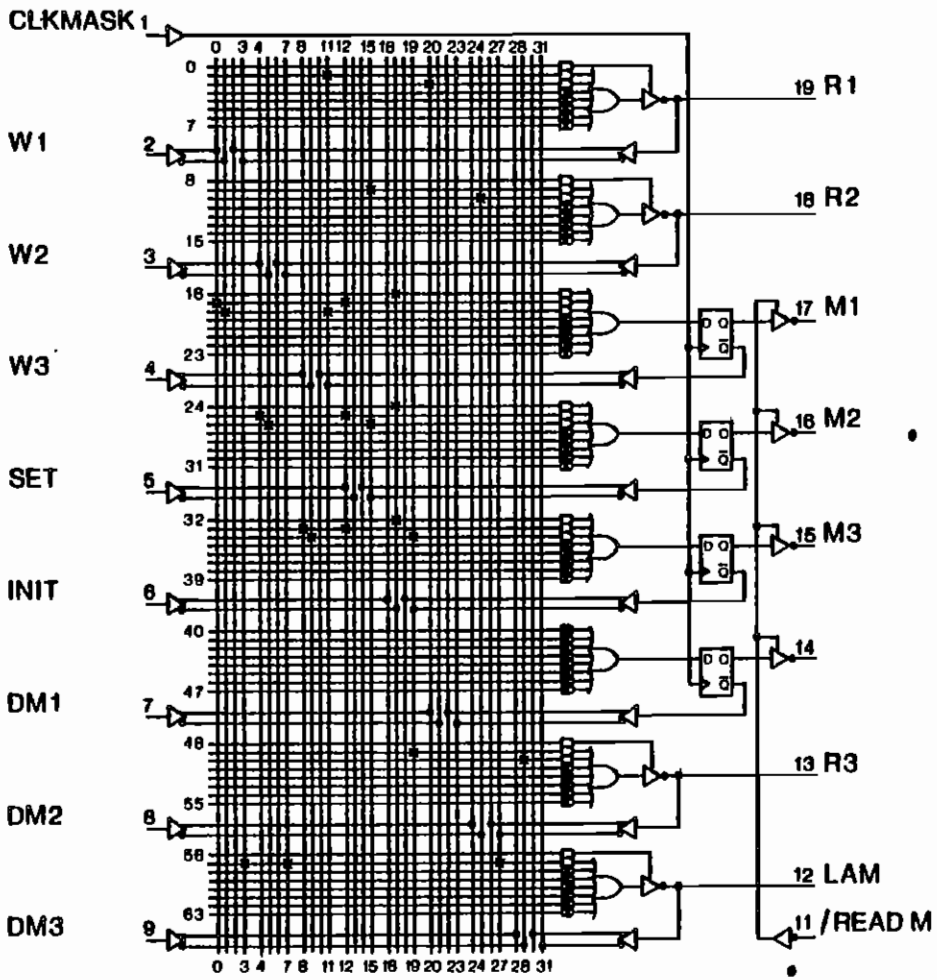
CLKMASK 1
W1 2
W2 3
W3 4
SET 5
INIT 6
DM1 7
DM2 8
DM3 9
0 3 4 7 8 11 12 15 16 19 20 23 24 27 28 31
19 R1
18 R2
17 M1
16 M2
15 M3
14
13 R3
12 LAM
11 /READ M

Fig.11