

CERN-Data Handling Division
DD/77/14



CM-P00059720

A NEW FAST AND PROGRAMMABLE TRIGGER LOGIC

A. Fucci

CERN, Geneva, Switzerland

S.R. Amendolia, E. Bertolucci, U. Bottigli, C. Bradaschia,
L. Foà, A. Giazotto, M. Giorgi, M. Givoletti, A. Menzione,
D. Passuello, M. Quaglia, L. Ristori, L. Rolandi, P. Salvadori,
A. Scribano, R. Stanga, A. Stefanini and M.L. Vincelli

Istituto di Fisica dell'Università, Pisa, Italy

INFN, Sezione di Pisa, Italy

ABSTRACT

The NAL (FRAMM) experiment, under construction for the CERN-SPS North Area, deals with more than 1,000 counter signals which have to be combined together in order to build sophisticated and highly selective triggers. These requirements have led to the development of a low cost, combinatorial, fast electronics which can replace, in an advantageous way, the standard NIM electronics at the trigger level. The essential performances of the basic circuit are:

- i) programmability of any desired logical expression;
- ii) trigger time independent of the chosen expression;
- iii) reduced cost and compactness due to the use of commercial RAM's, PROM's, and PLA's;
- iv) short delay, less than 20 nsec, between input and output pulses.

Geneva - 19 January 1977

(Submitted to Nuclear Instruments and Methods)

1. INTRODUCTION

In the FRAMM¹⁾ experimental apparatus about 1,000 counters constitute the elements of various hodoscopes for charged particles and photons. Signals from both sets of detectors can be treated in the same way for selecting particular events and for generating a trigger within a delay of ~ 200 nsec. For instance, a trigger signal is required when a fixed number of counters is fired, thus selecting events of fixed multiplicity. The selection criteria (in this case the multiplicity) need to be changed via computer in order to avoid errors due to manual operations. These requirements have been fulfilled by means of a logical unit which makes full use of integrated circuits normally used in computer technology. More generally, the proposed circuit can be used to build, in a modular way, the full fast logic of an on-line counter experiment, replacing the standard NIM electronics, with an important reduction in cost and volume^{*)}.

In the following, we first compare the classical and the new approaches in order to clarify the differences of the two solutions, and then we describe two units (one using NIM mechanics and one CAMAC standard) which correspond to two different practical needs. A further improved version of the CAMAC module, including INHIBIT and LAM functions, is described in the last paragraph.

2. GENERAL DESCRIPTION

Given a set of N input signals A_i , the output trigger can be expressed as a sum of product terms; algebraically this is written as

$$\text{OUTPUT} = (A_1 * A_2 * \dots) + \dots + (A_i * A_{i+1} * \dots) + \dots + (\dots A_{N-1} * A_N) .$$

Via NIM logic this signal is obtained as shown in Fig. 1.

Only two levels of logic appear in this solution, which is therefore inherently fast. Nevertheless, the system provides solutions tailored to the individual problems, so that changes in the decision equations, while entirely possible, are often difficult to perform owing to space, time, cabling, and financial considerations.

To overcome these NIM limitations, a general solution could be the flexible logical array shown in Fig. 2. M combinations of the N input variables may be selected by switches as well as the product terms which are to be added up together. These switches could be hardwired links, preset hardware masks over an integrated circuit array or, ideally, programmable memory cells. Unfortunately, the latter, although conceivable in principle, are not yet commercially available. Masks are

*) A first step in this direction has been made in the NICE experiment at Serpukhov by the IHEP-Karlsruhe-Pisa-Vienna Collaboration²⁾.

convenient only when applied on a large scale; and moreover, their initial setting cannot be changed during the operations. Hardwiring becomes cumbersome as N and/or M tend to large numbers.

The solution to this practical problem can be found by generating all the product terms which may access, through an equal number of presettable gates, to a summing operator as shown in Fig. 3. In such a scheme, the N input variables are used as bits of an address which is fully decoded. Each output of the detector is obviously one out of the 2^N possible combinations of the N inputs.

Each decoded address acts to enable the output from a presettable memory cell. The OR of all possible signals is presented as the final output. This circuit is in fact a schematic representation of a fully decoded Random Access Memory (RAM). Such circuits are commercially available and are characterized by low prices, high speed, and compactness. Depending on the chosen technology, a single chip may provide, at most, 4 functions from 10 input variables (MOS, TTL) or 1 function from 10 input variables (STTL, ECL).

Summarizing, for any combination of input signals a "switch" is available which can be preset via (external) computer or manual control. No simultaneous provision of A and \bar{A} signals is needed for generating different triggers, since all combinations (with and without A) are decoded. For the same reason, "no care" conditions, which are intrinsic in the NIM logic, do not exist in this approach, and all possible combinations must have a foreseen response.

To appreciate the difference between the NIM and the RAM schemes let us consider, for example, the trigger requirement of choosing any two, and only two, out of four input variables. The expression defining the output signal is

$$\text{OUT} = (A*B*\bar{C}*\bar{D}) + (A*\bar{B}*C*\bar{D}) + (A*\bar{B}*\bar{C}*D) + \\ (\bar{A}*B*C*\bar{D}) + (\bar{A}*B*\bar{C}*D) + (\bar{A}*\bar{B}*C*D)$$

Figure 4 shows the two solutions. The NIM approach requires an AND-gate for each product term together with an OR-gate as wide as the number of product terms. The RAM solution requires only one chip suitably programmed. If we now add the requirement of an additional term, for instance $A*B*C*\bar{D}$, the output is defined as

$$\text{OUT1} = \text{OUT} + A*B*C*\bar{D} .$$

In the NIM scheme this requirement is fulfilled by using an extra AND-gate, and an OR circuit, one input wider than the previous one. The RAM scheme, on the contrary, requires only a new setting of programmable memories and no hardware modifications. Table 1 shows five examples of data which have to be loaded into a 16-word RAM for particular logical combinations of the four inputs.

3. PROTOTYPE CIRCUITS

We discuss, in the following, two prototypes suitable for different applications in the hardware of on-line experiments. The first one is a 16-word module which can replace the logical NIM circuits and should be used whenever a single trigger function, implying only few counters, is desired, or when computer or CAMAC facilities are limited. With this aim in view, input and output stages are NIM compatible. The memory loading is performed by means of normal switches.

The second module has a 1024-word capacity and allows all logical expressions of 10 input variables. This circuit is a CAMAC unit, loaded via computer, and can solve complicated trigger requirements when many counters are involved.

3.1 The four-input circuit (MB 4/4)

A block diagram of this circuit is shown in Fig. 5. The core of the circuit is a Motorola ECL memory, type 10145; a 16-word \times 4-bit memory with fully decoded inputs. NIM to ECL and ECL to NIM translators at the input/output stages allow working with standard NIM pulses. As shown below, the provision of a fast strobe improves the performances of the circuit. Before the use of the unit, its memory has to be loaded with a sequence of 16 steps in order to specify to which output each particular input configuration should contribute. This is performed by manually selecting each address in sequence by means of switches I_1 - I_4 and its corresponding content via switches I_5 - I_8 . The loading of each memory bit then takes place by pushing the button P.

An obvious drawback of this circuit is the "loss of memory" in case of power failure. The addition of a -6 V accumulator can remove this trouble.

Even without pushing the test to the extreme limits, we have checked the performances of the circuit under severe experimental conditions. It works properly with a pulse length ≥ 4 nsec and at a repetition rate of ~ 100 MHz, the delay between input and output pulses being 18 nsec, independently of the chosen logical function (see Fig. 6).

If several output configurations are simultaneously selected, the timing and the equalization of the input signals must be carefully checked, in order to avoid ghost outputs. For example, a circuit is programmed to detect separately three functions: $F1 = A \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot B \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C$; $F2 = A \cdot B + A \cdot C + B \cdot C$; $F3 = A \cdot B \cdot C$. A good event generates $A \cdot B \cdot C$ simultaneously and $F3$ is output. However, if signal B is advanced with respect to A and C, as shown in Fig. 7, there will be three successive outputs of $F1$, $F3$, and $F2$. Similar generation of ghost outputs will occur if the signals are of different lengths.

The timing precision of the input pulses and the maximum rate depend critically on the value of resistors R, because presently available RAM's show a residual pulse also on the undesired output lines (edge effects); this pulse is normally cut by the threshold of the subsequent ECL/NIM translator. Since any mistiming between incoming pulses increases these unwanted signals, we can increase at will the sensitivity of this threshold effect. In the present configuration, two input signals can be 3 nsec out of time without producing a response at the undesired outputs.

If the particular experimental conditions allow the use of pulses 20 nsec or greater, then advantage may be taken of the coincidence stage between the RAM output signals and a fast strobe pulse (6-10 nsec wide). This coincidence selects the central part of the RAM signals, disregarding the edge effects.

3.2 The 10-input circuit (MB 10/4)

As the number of programming steps required is 2^N , manual programming becomes cumbersome for N larger than 4 or 5. On the other hand, the largest number of variables possible per module is often needed. With today's technology it is the memory size that fixes the practical limit. Currently, the largest ECL memory available is 1024-word \times 1-bit (produced by at least two manufacturers). A memory of this size will evidently need to be loaded under computer control. This additional requirement is well satisfied by choosing a CAMAC framework.

The loading procedure is the same as in the four-input circuit, except that the switches and buttons are replaced by CAMAC logic. The circuit, containing four RAM's in order to provide four independent outputs, is shown in Fig. 8, to the left of the dashed line. Elements to the right of the dashed line are discussed later in Section 4.

A CAMAC C or Z cycle resets the address counter. Then, by sending an F(16) to the module, the contents of the Write lines W_{11} - W_{14} are written into the current address during S1, and the address counter is incremented during S2. In order to check the memory contents, the READ function has also been provided. By sending an F(0) to the module the output of the current address is presented on the Read lines R_{11} - R_{14} during N, and then the address counter is incremented during S2. One C or Z cycle and 1024 CAMAC cycles are needed to fully load or read out the memory.

The possibility of loading the memory quickly is important in a multipurpose experiment because it allows the use and proper scheduling of several trigger configurations.

Figure 9 shows the listing of a program written in HP-Basic language that provides for four different logical functions, chosen as examples.

4. CIRCUIT EXTENSIONS

As a result of the tests performed on the prototypes, several extensions have been proposed which greatly increase the potential of the circuit. To take advantage of the CAMAC interrupt system, some kind of LAM generation is necessary. However, the module can generate up to four triggers and so masking logic will also be needed.

4.1 LAM logic

The setting of the LAM MASK, which is loaded by $F(17) \cdot A_5 \cdot S_1$, enables one or more of the outputs to generate a LAM. Since the computer needs a certain amount of time to recognize a LAM signal and to service it, some or all of the counting rates should be stopped during this time. This operation is fulfilled by setting bits in the INHIBIT MASK using the CAMAC operation $F(17) \cdot S_1 \cdot (A_0 - A_4)$.

Having received the LAM, the computer can read the four flip-flops to discover which of the triggers did set the LAM. At the end of the LAM service routine, the computer sends a LAM reset signal and the memory outputs are once again enabled.

In order that a module, once having sent a LAM, may inhibit other modules from doing the same, an external inhibit is foreseen which enters the INHIBIT MASK register independently.

Loading the module now takes place in three stages: firstly the memory is loaded with the required logical functions; secondly the outputs that may generate a LAM are selected; and finally the inhibition of outputs by LAM's are preset.

4.2 True random access

During the loading of the trigger program in the memory, it has been foreseen that sequential access only could be a limitation. In order to overcome this difficulty, a load function is provided. $F(17) \cdot A_6$ will load the address on lines $W_1 - W_{10}$ into the counter. This location may then be read or written into, as described earlier.

Acknowledgements

The authors would like to thank M. Martin for his assistance.

REFERENCES

- 1) The FRAMM Collaboration, CERN/SPSC/74-15 (1974), CERN/SPSC/74-83 (1974) and CERN/SPSC/76-23 (1976).
- 2) Yu.B. Bushnin, R.N. Krasnokutskij, Yu.V. Mikhailov and R.S. Shuvalov, An economic nanosecond logic system, IHEP preprint (1976).

Table 1

Inputs				Word	Data to be loaded for particular logic combinations					
1	2	4	8		$1 \cdot 2 \cdot 3 \cdot 4$	$1 + 2 + 3 + 4$	$(1 \cdot 2 \cdot 3 \cdot \bar{4})$	$1 \cdot 2 \cdot (3 + 4)$	Any two out of four	
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	1	0	1	0	0	0	
0	1	0	0	2	0	1	0	0	0	
1	1	0	0	3	0	1	0	0	1	
0	0	1	0	4	0	1	0	0	0	
1	0	1	0	5	0	1	0	0	1	
0	1	1	0	6	0	1	0	0	1	
1	1	1	0	7	0	1	1	1	0	
0	0	0	1	8	0	1	0	0	0	
1	0	0	1	9	0	1	0	0	1	
0	1	0	1	10	0	1	0	0	1	
1	1	0	1	11	0	1	0	1	0	
0	0	1	1	12	0	1	0	0	1	
1	0	1	1	13	0	1	0	0	0	
0	1	1	1	14	0	1	0	0	0	
1	1	1	1	15	1	1	0	1	0	

Figure captions

- Fig. 1 : NIM logic.
- Fig. 2 : Logic array.
- Fig. 3 : Fully decoded RAM.
- Fig. 4 : Differences between NIM and RAM solutions.
- Fig. 5 : Block diagram of the four-input decision box.
- Fig. 6 : The resolution and speed of the circuit are illustrated here. The upper trace is a 5 nsec signal applied to two of the four inputs. The memory has been programmed for a coincidence at these inputs, i.e. $A \cdot B$. The output 0 of the memory is seen on the lower trace.
- Fig. 7 : Timing diagram of A, B, and C variables.
- Fig. 8 : Block diagram of the ten-input decision box.
- Fig. 9 : Listing of a loading program.

NIM LOGIC

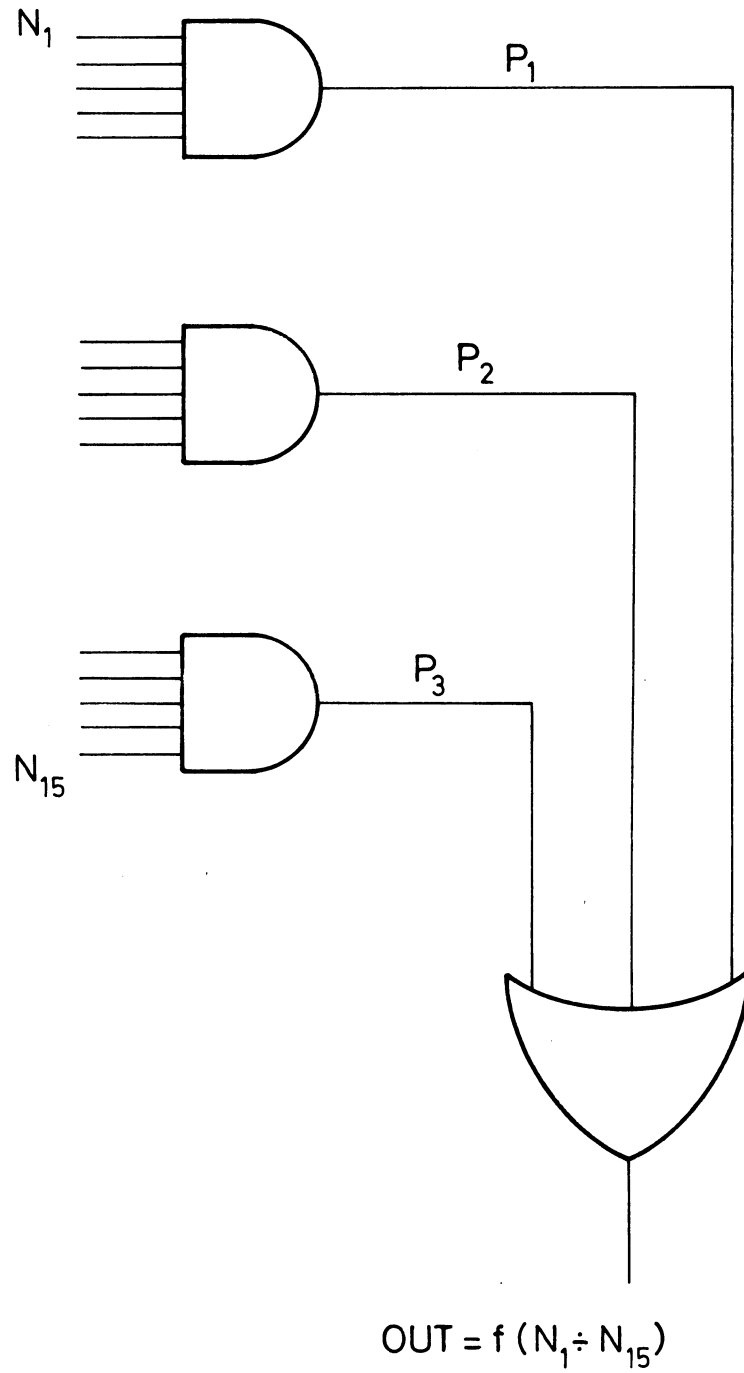
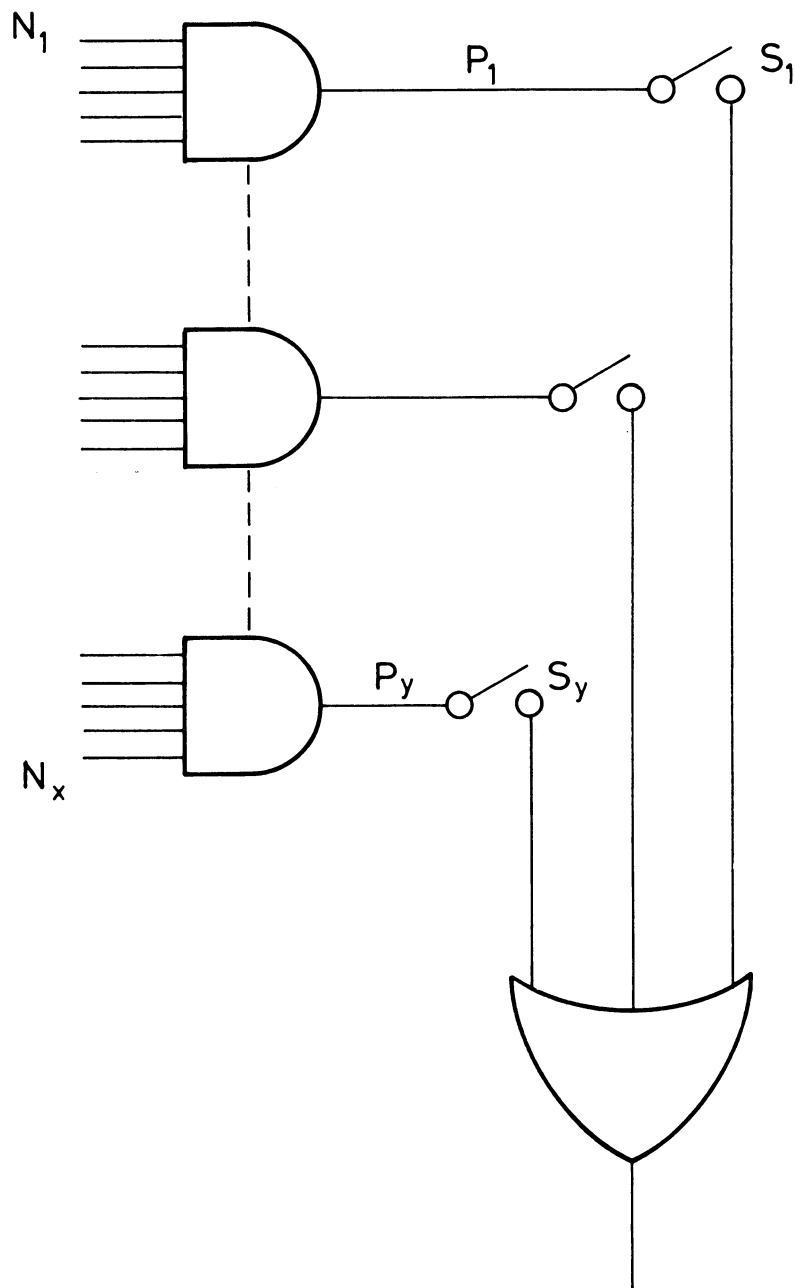


Fig. 1

LOGIC ARRAY



$$\text{OUT} = f(N_1 \div N_{x1}, S_1 \div S_y)$$

Fig. 2

FULLY DECODED RAM

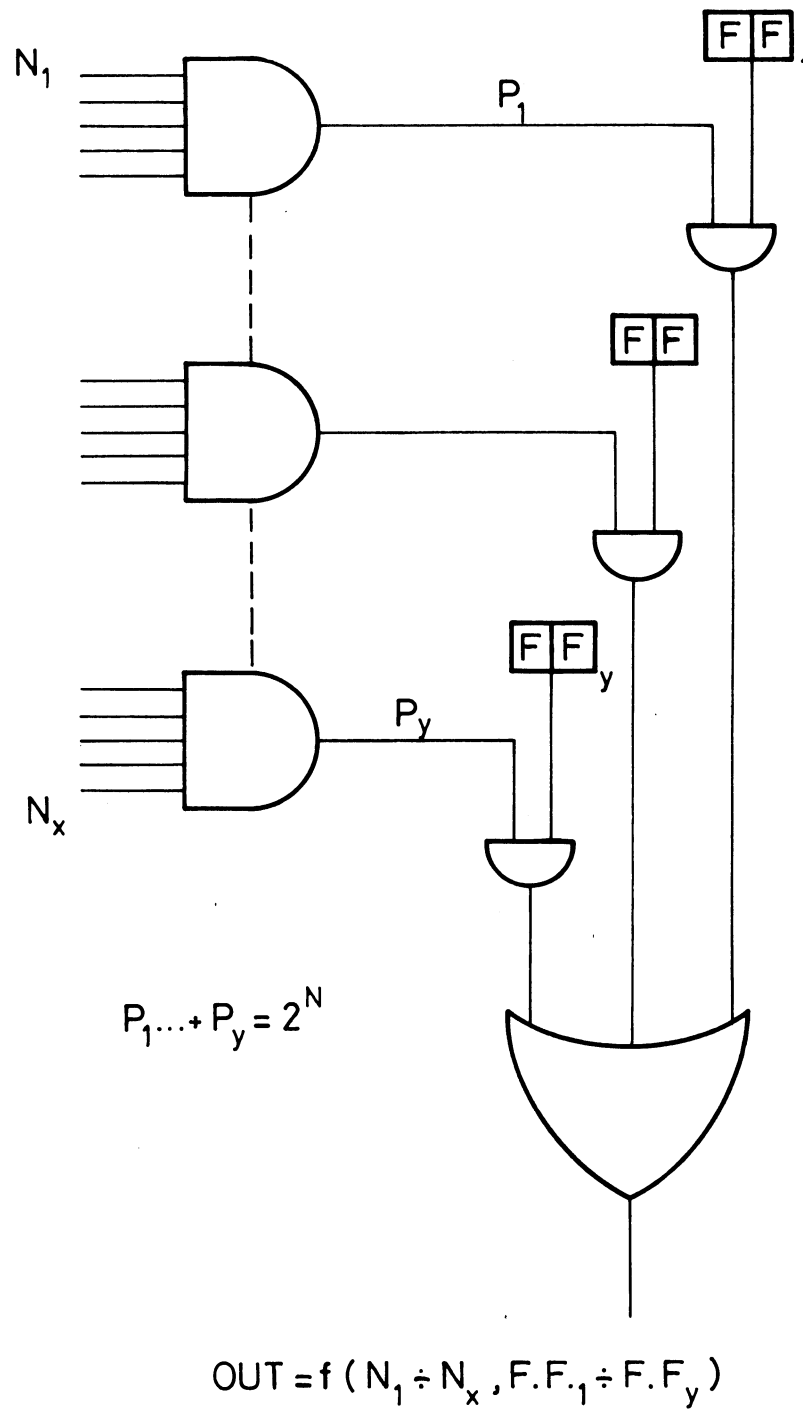


Fig. 3

NIM SOLUTION

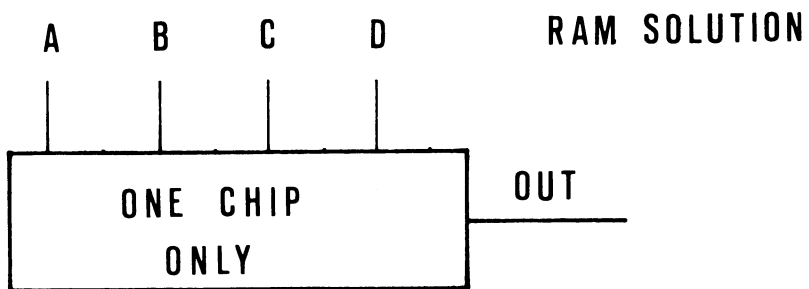
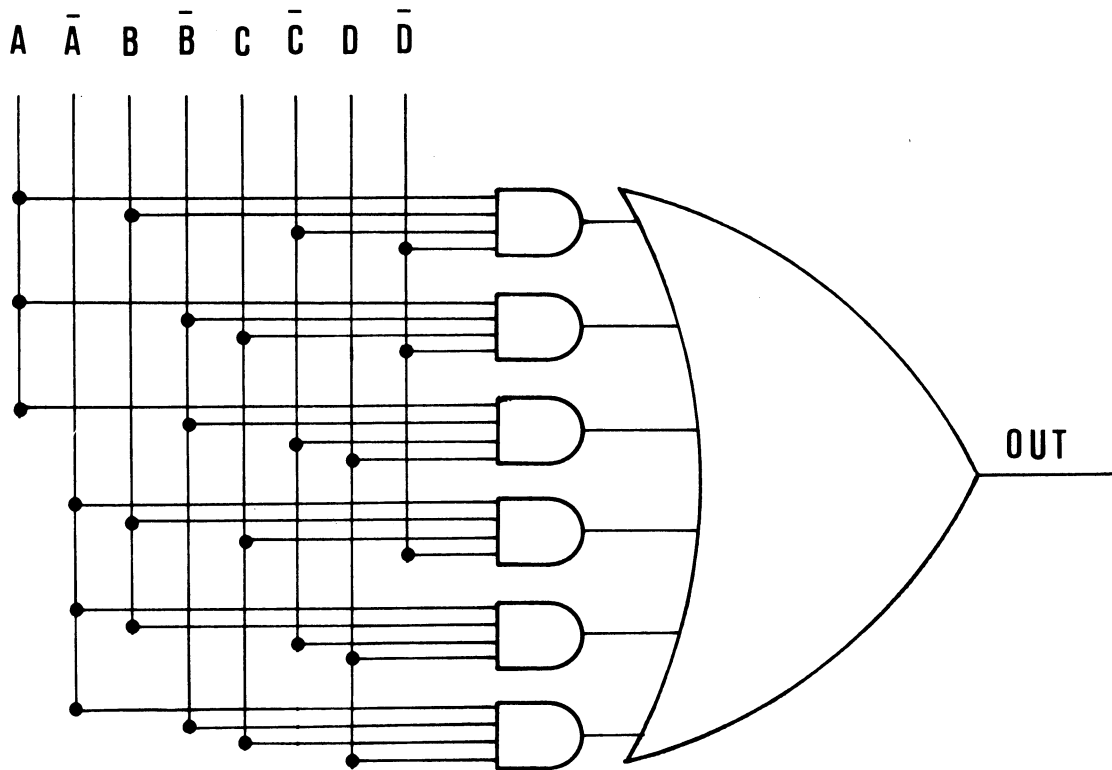
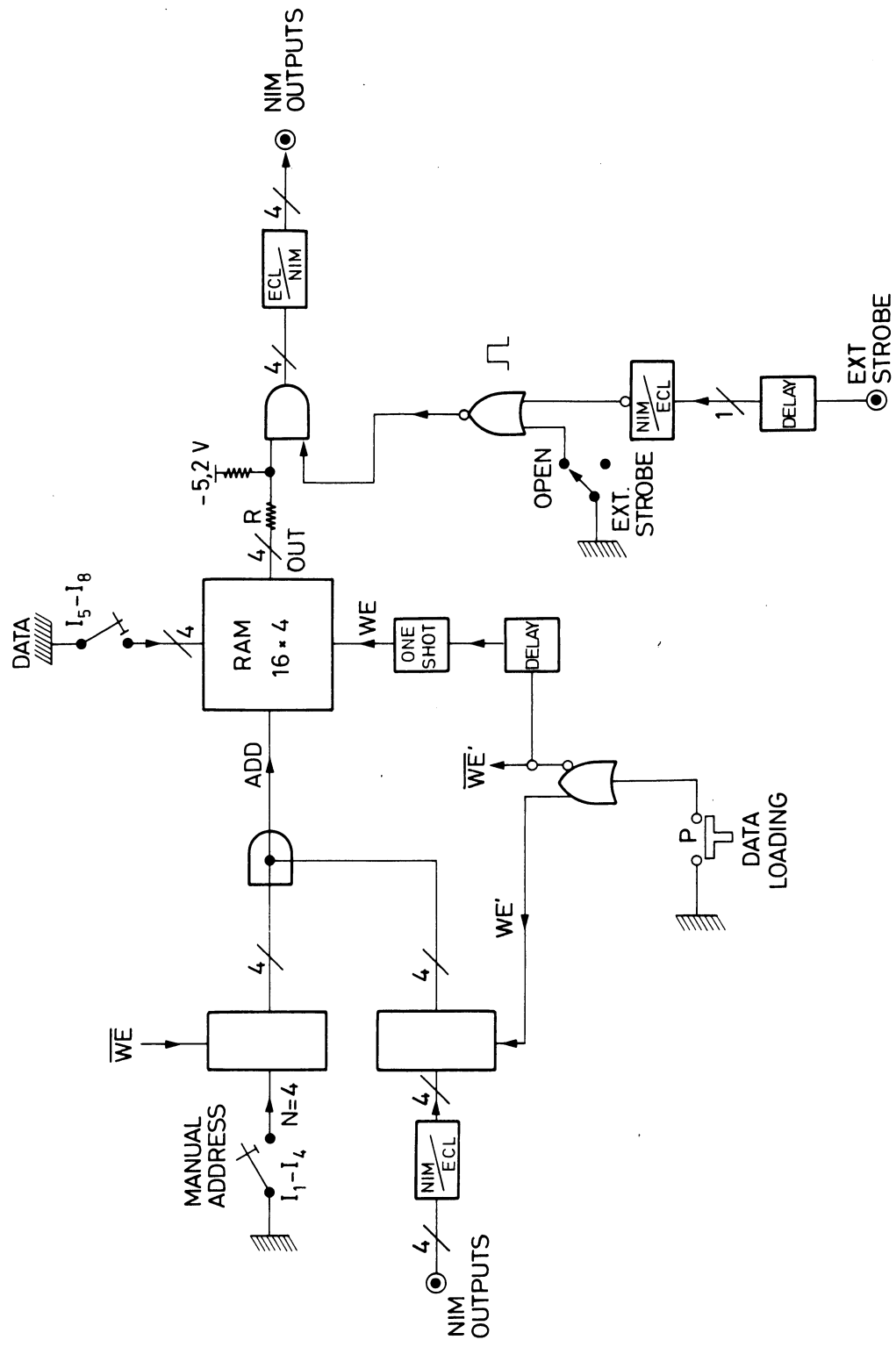


Fig. 4



4 INPUT DECISION BOX

Fig. 5

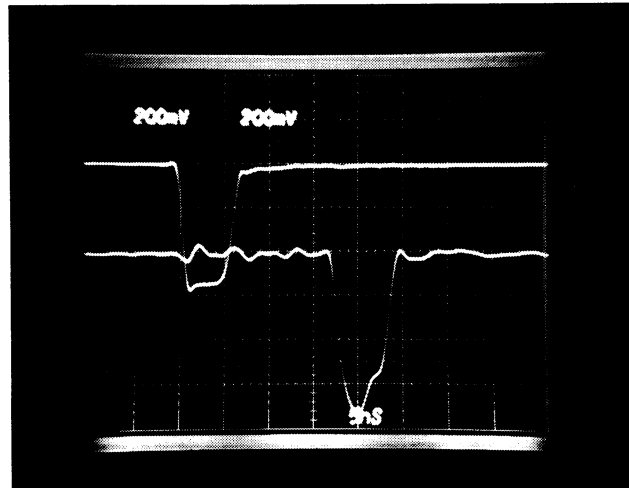


Fig. 6

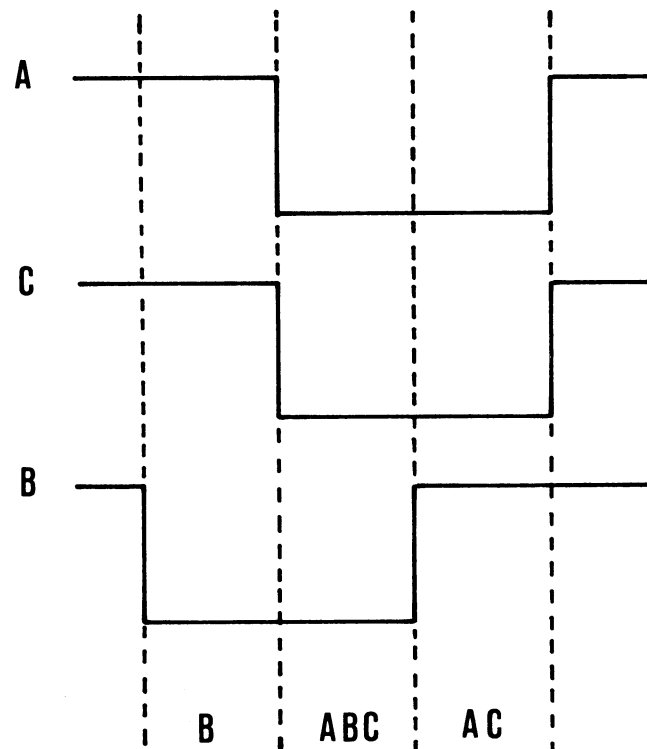


Fig. 7

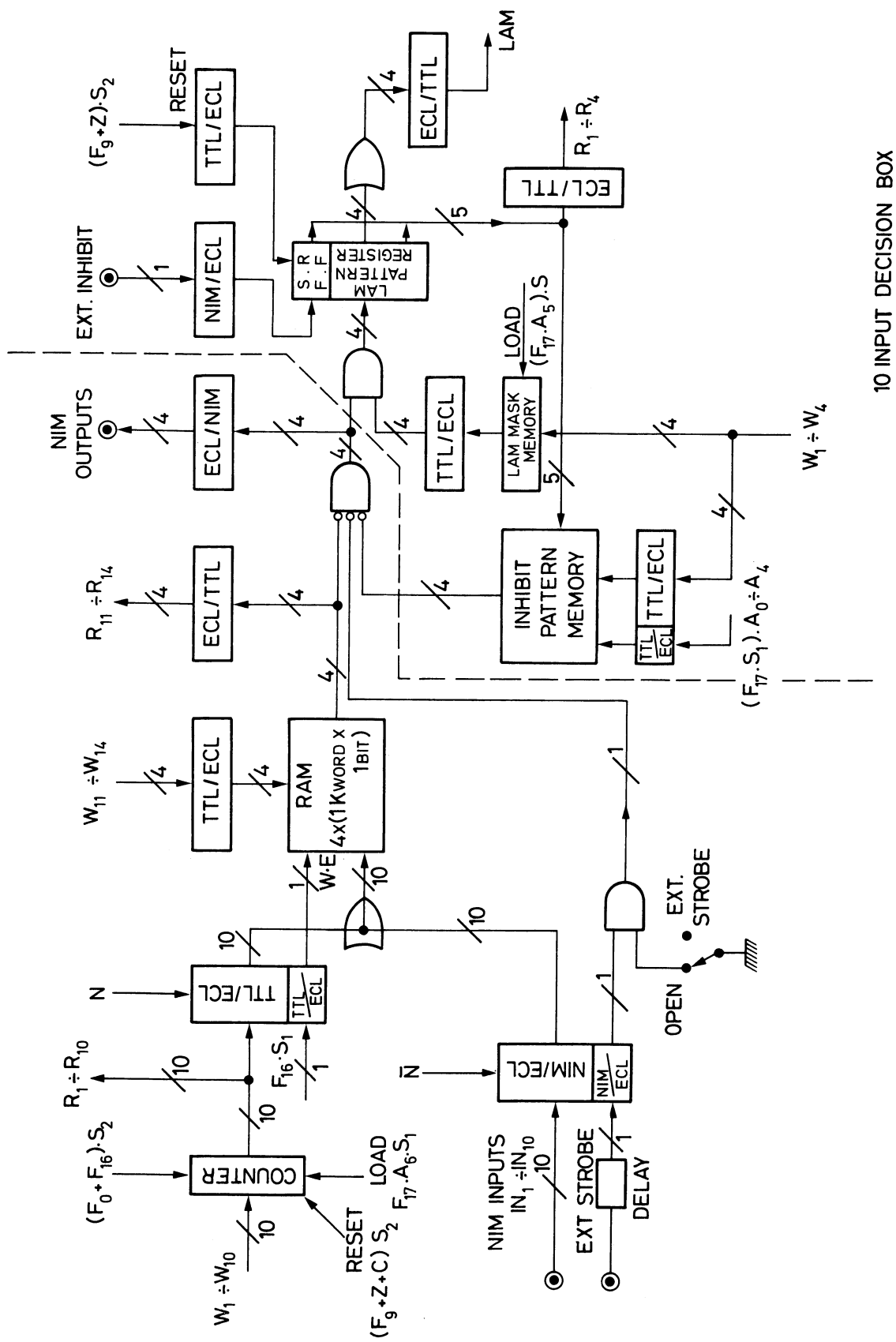


Fig. 8


```

1.  REM **LOADING OF THE 10 INPUT CIRCUIT**
10. DIM I(10), O(4)
20. PRINT "GIVE AN ADDRESS"
30. INPUT N
40. REM **SEND A CAMAC CLEAR**
50. CALL (4)
60. REM **START LOOP ON MEMORY ADDRESSES**
70. FOR J=0 TO 1023
80. LET A=J
90. REM **DECODE MEMORY ADDRESS IN BINARY**
100. FOR K=1 TO 10
110. LET I(K)=A - INT(A/2)
120. LET A=INT(A/2)
130. NEXT K
140. REM **DEFINE WANTED INPUT LOGICAL FUNCTION**
141. REM **O(1,...,4)=1 ONLY WHEN SECOND TERMS SATISFIED**
150. LET O(1)=I(1) AND I(2) AND I(3) AND I(4)
160. LET O(2)=I(1) AND I(2) AND I(3) AND I(4) AND I(5) AND (NOT I(6))
170. LET O(3)=I(1) AND (I(2) OR I(3))
180. LET S=0
190. FOR K=1 TO 10
200. LET S=S+I(K)
210. NEXT K
220. LET O(4)=(S=3)
230. LET P=(O(1) + O(2)*2+O(3)*4+O(4)*8)*2+10
240. REM **WRITING OF THE MEMORY**
250. CALL (1,O,N,O,16,P,1)
260. NEXT J
270. PRINT "THE MEMORIES HAVE BEEN WRITTEN"
280. STOP
290. END

```

Fig. 9