COMPUTER SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORD UNIVERSITY · STANFORD, CA 94305

DESIGN FOR AUTONOMOUS TEST

Edward J. McCluskey, Saied Bozorgui-Nesbat

CRC Technical Report No. 81-l

(CSL TR No. 182)

June 1981

This work was supported in part by
National Science Foundation Grant Number MCS-7904864
and in part by Army Research and Development Command
Contract No. DAAK20-80-C-0266

Center for
Reliable
Computing

DESIGN FOR AUTONOMOUS TEST

Edward J. McCluskey, Saied Bozorgui-Nesbat

CRC Technical Report No. 81-1
(CSL TR No. 182)
June 1981

ERRATA

Page	Line	Error	Correction
7	11	23%	40%
7	13	22%	37%
16 & 17	24/1	Replace "a two input multiplexer" with "select gates"	
17	2	one gate	two gates
17	6	11	19
17	7	23%	40%
17	22	22%	37%

Figure 3b is being replaced by the attached, corrected figure.

Center for Reliable Computing
Computer Systems Laboratory
Departments of Computer Science and Electrical Engineering
Stanford University, Stanford, CA 94305

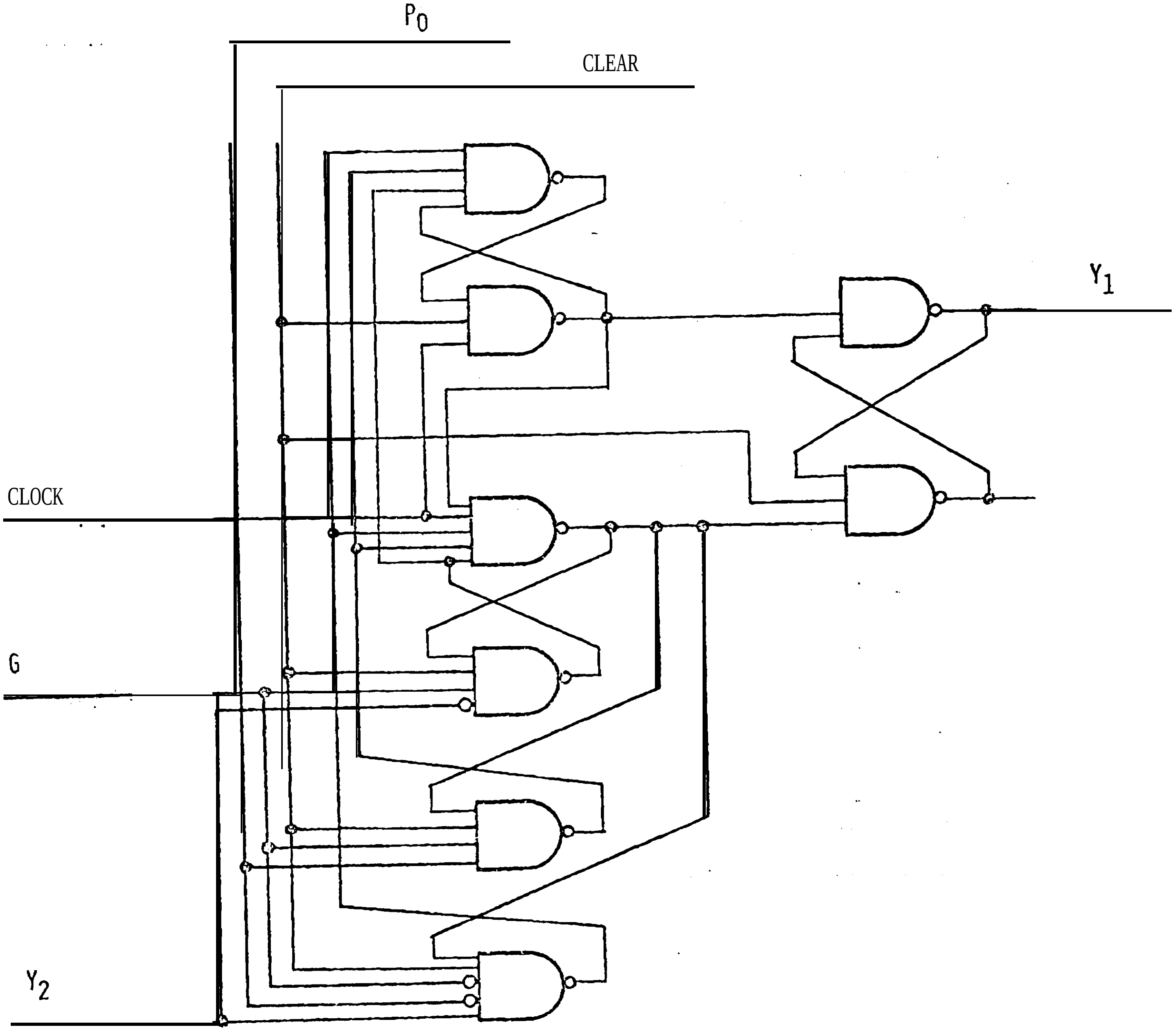


FIGURE 9.

Center for
Reliable
Computing

DESIGN FOR AUTONOMOUS TEST

Edward J. McCluskey, Saied Bozorgui-Nesbat

CRC Technical Report No. 81-1

(CSL TR No. 182)

June 1981

Center for Reliable Computing
Computer Systems Laboratory
Departments of Computer Science and Electrical Engineering
Stanford University, Stanford, CA 94305

This work was supported in part by National Science Foundation Grant Number MCS-7904864 and in part by Army Research and Development Command Contract No. DAAK20-80-C-0266.

DESIGN FOR AUTONOMOUS TEST

Edward J. McCluskey, Saied Bozorgui-Nesbat

CRC Technical Report No. 81-1
(CSL TR No. 182)

June 1981

Center for Reliable Computing
Computer Systems Laboratory
Departments of Computer Science and Electrical Engineering
Stanford University, Stanford, CA 94305

ABSTRACT

A technique for modifying networks so that they are capable of self test is presented. The major innovation is partitioning the network into subnetworks with sufficiently few inputs that exhaustive testing of the subnetworks is possible.

Procedures for reconfiguring the existing registers into modified linear feedback registers (LFSR's) which apply the exhaustive (not pseudo-random) test patterns or convert the responses into signatures are described. No fault models or test pattern generation programs are required. A method to modify CMOS circuits so that exhaustive testing can be used even when stuck-open faults must be detected is described. A detailed example using the 74181 ALU is presented.

Table of Contents

Page

List of Figures

INTRODUCTION

Production and usage of digital integrated circuits involve a variety of testing problems: wafer probe, wafer sort, preship screen, incoming test of chips and boards, test of assembled boards, system test, periodic maintenance, repair testing, etc. Traditional test techniques for chips and populated boards require derivation of input test sets with associated output responses. These tests are stored in a fault dictionary and programmed into an automatic tester which applies the signal to the unit under test and checks that the correct response is produced. For more details on this subject see p. 16 in [Breuer,76]. There are a number of difficulties with this approach:

1. A fault model is required. In VLSI circuits the classical assumption that only single stuck-at-faults need be modelled may no longer be valid. More complex models are possible but they substantially increase the difficulty of test pattern generation.

2. Test pattern generation is required. Automatic test pattern generation is very costly and typically does not provide sufficiently high fault coverage. Manual test pattern generation has the added disadvantage of inserting a long delay into the production cycle. For sequential circuits automatic generation may break down completely and manual generation can be very lengthy and produce poor results.

3. An expensive tester is required. When test generation produces many patterns the tester is tied up for a long time so that many testers must be used.

Current approaches to alleviate testing problems fall into two categories. One uses the ad hoc method of designing the circuitry so that test pattern generation is economically feasible. These approaches can be strictly ad hoc as described in [Turino,79] or can make use of a program to develop a "testability measure" which serves as a guide for redesign to make the circuit more testable [Stephenson,76].

The other approach is to include, in the original design, structures to facilitate testing. These structures reduce the sequential portions of the circuits to combinational networks for testing, and allow test patterns to be scanned in and out of the circuit flip-flops [Williams,73; Eichelberger,77].

While these scan designs provide substantial improvements over the designs that do not include such explicit testability features, a number of problems still remain: The chips with scan circuits introduce additional complexity into the design process. It is still necessary to do test pattern generation for the combinational portions of the network. The testing time is substantially increased because of the time taken to scan patterns in and out of the network. The problems of fault modelling remain. Expensive and sophisticated tester equipment is needed. Both of these approaches can reduce the problems cited above, but do not offer any fundamental remedies.

AUTONOMOUS TEST

In recent years there has been increasing interest in Built-in Test (BIT) techniques. With these techniques sufficient extra circuitry is added to a module or chip so that external testing equipment is not required. The internal test circuitry generates and applies the test inputs, and also checks that the response is correct. Generation of the test inputs is typically done by a counter: either pseudo-random inputs (RANDOM) are derived from a Linear Feedback Shift Register (LFSR) or all 2^n inputs (for an n-input circuit) (EXHAUSTIVE) are formed. The use of a ROM to store the input patterns is also possible. The output response is typically checked with an LFSR used as a signature analyzer. For multi-output circuits it is possible to use one LFSR for all of the outputs by connecting each output to an XOR gate feeding a different stage of the LFSR. This technique will be called parallel signature. Another possibility for checking the response is to make use of the checking circuits which are present for concurrent checking of the circuits during normal operation (parity checks, modulo code checkers, etc.) The following table summarizes some of the schemes for BIT.

Table 1. BIT Schemes

Reference	Test input generation	Output response check
[Benowitz,75]	RANDOM	Signature
[Bozorgui,80]	EXHAUSTIVE	Parallel Signature
[Eiki,80]	Shift register (SR) with CUT output connected to SR input	Parallel SR
[Fasang,80]	RANDOM	Parallel Signature
[Konemann,80]	RANDOM	Parallel Signature
[McCluskey,80]	EXHAUSTIVE	Parallel Signature
[Sedmak,79],	EXHAUSTIVE	On-line concurrent
[Sedmak,80]		checking circuits.

The use of pseudo-random test inputs is convenient in that the calculation circuitry is simple and it is not necessary to do test pattern generation. A significant feature is its applicability to sequential as well as combinational logic. The drawbacks are that the fault coverage is prohibitively difficult to obtain, [Shedletsky,77], and very long test sequences must therefore be applied. Parallel signature analysis appears to detect many errors, but calculating its detection capabilities has been intractable except in very special situations, [Smith,80]. Thus, there is increasing concern about its use for self-test.

Another response checking technique, syndrome analysis, has been proposed; but this scheme relies on all input combinations being applied and is only guaranteed to detect single stuck-at faults, [Savir,80]. The use of the circuitry already present in a circuit to provide concurrent checking is a very promising approach since it uses circuitry which already exists in some designs, thus avoiding part of the additional overhead for BIT. Its weakness lies in the fact that the coverage of the concurrent checks is relatively low and may not be sufficient for manufacturing test during which many faults may be present.

The application of all 2^n input combinations avoids test pattern generation. It is restricted to designs which have been decomposed with a technique such as that used for LSSD, [Eichelberger,77]. However, for such designs it is unnecessary to calculate fault coverage since the entire truth table of the circuit is verified. The only faults which would go undetected are those that are missed by the output checking circuitry. No faults in the circuit under test would fail to be tested by the inputs applied. The major difficulty with using all input combinations has been the concern that some circuits would have so many inputs that very long test times would be required. An approach which has been tried to avoid this difficulty is to trace back from each circuit output to determine the actual number of inputs which drive the output, [Bottorff,77]. This will sometimes reduce the number of input combinations sufficiently so that exhaustive testing is feasible.

The technique presented in this paper relies on exhaustive testing, but divides the circuit into segments or partitions in order to avoid excessively long input test sequences. It differs from previous attempts along these lines in that the partitions may divide the signal path through the circuit rather than just separating the signal paths from one another.

Any built-in-test scheme should be evaluated in terms of several criteria:

1. Design difficulty - If the design of the additional BIT circuitry is difficult then the technique will probably not gain acceptance. Our experience with the examples considered thus far is that the scheme presented here is no more difficult to implement than the LSSD technique which is now widely used. For the circuits studied it has been straightforward to manually develop the partitions. However, in order to guarantee generality a formal technique, suitable for use in a design automation system, is being specified. This will be presented in a subsequent paper.

2. Performance penalty - The BIT circuitry should not have a significant negative impact on the performance of the function to which it is added. This is discussed in detail in connection with the design example, but it appears that BIT can be achieved with negligible performance degradation.

3. Cost penalty - The BIT circuitry should require minimal additional circuitry or chip area. This is a difficult criterion to discuss quantitatively. The most convincing demonstration would be to

have a large number of varied designs and to measure the circuitry required for BIT. We do not have such data available yet (and are unaware of the existence of such data for any of the other methods to use as comparisons).

A technique will be shown which does not require any additional circuitry in the functional part of the original design. The registers at the input and out put of the functional circuitry do have to be modified (these are the same registers that are modified for LSSD). For the case where these registers use TTL D flip-flops a design will be shown which requires 37%additional circuitry for the input register and 23% additional circuitry for the output register. Extra logic to control the BIT adds 23 gates for this example. The overhead for the entire circuit is approximately 22%. Our feeling is that this should be regarded as more of an upper bound than a typical example since the circuit (74181) used in the example had fewer gates (63) than would be expected for typical custom LSI or VLSI. Note that as the number of gates in the functional circuitry increases the overhead percentage should decrease.

Designs for built-in test typically operate in two modes, normal and test. The technique to be described here also employs these two modes :

NORMAL MODE: In this mode the additional diagnostic circuitry is transparent to the user and the fault free circuit functions according to the specifications.

TEST MODE: In the test mode the circuit is automatically partitioned and each subcircuit is tested exhaustively. The response of each subcircuit is verified by the diagnostic circuitry. The verification circuitry must be capable of detecting any fault that can change the input and output relationship of any subcircuit. Two techniques to implement the partitioning will be described in the next section.

PARTITIONING

Since the bigger and the more complex a circuit becomes, the more difficult it will be to test, it seems reasonable to try to test a complex circuit by parts rather than as a whole. In general, a circuit can be partitioned into two or more subcircuits. If partitioning can be carried out such that the number of the input lines to each subcircuit is significantly fewer than that in the original circuit, it will be possible to test each subcircuit exhaustively. The time required to test all subcircuits exhaustively will be less than the time required to test the whole circuit. In fact, it is just necessary to have each partition output function depend on a sufficiently small number of input variables. This question is discussed in [McCluskey,81]. An implementation algorithm is given for exhaustive testing in terms of this functional dependence rather than the total number of input variables.

In order to exhaustively test each subcircuit, all subcircuit inputs must be controllable at the input of the circuit and all subcircuit outputs must be observable at the circuit outputs. This can be achieved in two ways: (1) hardware partitioning and (2) sensitized partitioning. Next we describe these two partitioning methods.

MULTIPLEXER PARTITIONING

Access to the embedded inputs and outputs of the subcircuit under test can be achieved by inserting multiplexers and connecting the embedded inputs and outputs of each subcircuit to those primary inputs and outputs that are not used by the subcircuit under test [Bozorgui,80]. Figure 1a depicts this hardware partitioning scheme. By controlling the multiplexers, all the inputs and outputs of each subcircuit can be accessed using primary input and output lines. For example, to test subcircuit G_1 the multiplexers can be controlled as depicted in Figure 1b to completely access all the inputs and outputs (including the embedded intermodule lines).

Example of the Multiplexer Partitioning

To demonstrate this method, partitioning of 74181ALU/FUNCTION GENERATOR, [TI,81], has been carried out. Figure 2a shows the circuit diagram of this IC. The partitions used are shown in this Figure. The IC is partitioned into five subcircuits, four of which are identical structurally. Figure 2b depicts the block diagram of these partitions. Figure 3depicts the arrangement of the multiplexers added to activate this partitioning. By properly controlling these multiplexers all five subcircuits can be accessed and tested:

1. In order to test the N_1 subcircuits, the output multiplexers can be set to connect the HI and LI lines to the output pins. Since HI depends only on AI, BI, S_2, and S_3 it can be verified by applying all 16 combinations of these 4 inputs. By applying the same signal to all of the AI and also driving all of the BI in common, all four HI functions can be verified simultaneously. The situation for the LI functions is the same except for the replacement of S_2 by S_0 and S_3 by S_1. A total of 32 patterns are sufficient to verify all of the HI and LI circuits.

2. To test subcircuit N_2, all output multiplexers can be reset and all intermediate multiplexers set so that the AI and BI lines can control the input lines to $N_{2'}$ By applying all possible input patterns to lines A3, B3, A2, B2, A1, B1, A0, B0, M, and C_n the subcircuit N_2 can be tested exhaustively with $2^{10} = 1024$ input patterns.

The total number of patterns needed to test the circuit exhaustively is 1056 (1024 + 32). Without partitioning, 2^{14} or 16k input patterns would be needed to verify the circuit. The four gate level delay introduced by the multiplexers can be reduced by embedding the multiplexers in the subcircuits. The actual gate level design has been carried out [Bozorgui,80]. Based on the number of gates, the fraction of the hardware added is estimated to be approximately 30%.

SENSITIZED PARTITIONING

A considerable portion of the diagnostic circuitry of the previous section is comprised of the routing multiplexers. These multiplexers reduce the speed of the operation and are costly to implement. However it is possible to achieve the same testing discipline without actually inserting any multiplexers at all. Thus, the previous discussion should be regarded as an introduction to the technique, to be described next, which is the actual suggested implementation.

Circuit partitioning and subcircuit isolation can be achieved by applying the appropriate input pattern to some of the input lines. The effect achieved is similar to that of hardware partitioning: paths from the primary inputs to the subcircuit inputs and paths from the subcircuit output to the primary output can be sensitized. Using these paths each subcircuit can be tested exhaustively.

Example of Sensitized Partitioning

The same IC, 74181 ALU/FUNCTION GENERATOR, has been used as an example of sensitized partitioning. To test the N_1 blocks we set M=1 and set $S_2=S_3=0$. This sets all HI lines to 1 and hence LI' (LI complement) will appear at the corresponding output F_i lines (Fig. 4a). Then all LI functions can be tested by connecting all AI lines together and all BI lines together and cycling through all 16 [AI, BI, S_0, and S_1] combinations.

With a similar procedure all HI functions can be tested: setting M=1 and $S_0=S_1=1$ will make the HI lines appear at the corresponding F_i output lines. Then all AI lines can be connected together and all the BI lines can be connected together and by cycling through all possible values of [AI, BI, S_2, S3}], that is 1 6 patterns, all HI functions can be tested (Fig.4b).

To test subcircuit N_2 we note that in normal operation mode the lines HI and LI are only subject to [II, 10, 00] values. Hence, for exhaustive testing of this subcircuit, we only have to drive these lines into 3 pairs of values. Setting $S_0=S_1=S_2=0$ and $S_3=1$ results in:

HI'=(AI)(BI) and LI'=AI

By cycling all AI and BI through [00,10,11] in parallel, the lines HI and LI will be driven to [11,10,00]. All possible input patterns are applied to N2 by cycling AI and BI through 3 combinations in parallel and cycling M and C_n through their 4 combinations for each of the AI, BI combinations. The total number of tests needed for exhaustive testing of N2 is thus:

$$(3^4)\,(2^2) = 324 \text{ patterns.}$$

TEST GENERATION AND RESPONSE VERIFICATION

An autonomously testable circuit in the test mode must have provisions to generate test inputs for itself, to verify its own correct output, and give notification of any fault occurrence. For exhaustive testing, a device capable of generating all possible input combinations is needed. An n-bit Linear Feedback Shift Register (LFSR) is suitable to generate all but one of the possible n-bit input patterns used to test a circuit exhaustively [Gschwind,75]. A modification of the drive circuitry for the LFSR can include the all zero state and hence generate all the possible input combinations for testing an n-input circuit. Other forms of counter could also be used, but the modified LFSR appears to be more economical than a synchronous binary counter and does not produce the spikes typical of ripple counters.

An LFSR can also be used to compact the output pattern of the circuit under test by generating a "signature" that is compared with the precomputed signature of the fault-free circuit, [Benowitz,75]. Multiple outputs can be compacted in parallel by connecting each output into a different stage of the LFSR through an XOR gate as suggested in [Konemann,80].

Figure 5 shows a 3-bit reconfigurable LFSR module. The two inputs N and S control the reconfigurability. With N=1, the module behaves as a register. The input data X_1, X_2, and X_3 can be simultaneously clocked into the flip-flops and can be read out on the I_1, I_2, and I_3 lines (Fig. 6a).

With N=0 and S=0, the module cycles through all the possible 3-bit patterns. In this mode the inputs are disconnected from the circuit (Fig. 6b). The output of this module in this mode provides the input patterns for the subcircuit being verified.

With N=0 and S=1, the module functions as a parallel signature analyzer (Fig. 6c). In this mode, the module linearly combines the incoming inputs and produces a compact signature of the output pattern. In the event of a fault that creates an error in the output pattern, the signature would be different from the fault free signature with a high probability [Smith,80].

Such a module can replace the intermediate registers or the input/output flip-flops. In the test mode such a module can be reconfigured to aid the testing of the other parts of the circuit.

Using this procedure, many faults within the reconfigurable circuit can also be detected. However, if a module is only tested during its test (reconfigured) mode, when its interconnections are different from its normal mode interconnections, faults that only affect the normal operation of this circuit may remain undetected. These reconfigurable circuits are used to test the correct operation of other modules; the normal operation interconnection of these modules must be checked with the aid of other diagnostic circuits.

As an example of a checker fault, consider a circuit under test and its associated diagnostic circuitry. A fault that permanently connects the diagnostic circuitry to the circuit under test may be untestable by the diagnostic circuitry alone, since to test for this fault the diagnostic circuitry may have to be disconnected from the circuit under test, hence the contradiction. The external test for such a condition is often a simple continuity test.

Autonomously Testable Design of 74181

An example of the design details of the self-test circuitry for the 74181 ALU will be shown next. The block diagram of this design is shown in Fig. 7. Blocks N_1 and N_2 refer to the partitioning blocks of figures 2a and 2b. The design also includes two sets of registers, an input register and an output register. Internal control signals are derived from the added circuit: testing control circuit. Two pins are added to the circuit: an input Test pin, which initiates the autonomous testing process, and an output pin OK, which upon the completion of the test verifies the correctness of the response of circuit under test. The input and the output registers have been modified to function as an exhaustive test pattern generator and a signature analyzer, respectively. The method of sensitized partitioning is employed to partition the ALU into five blocks. Under the control of the testing control circuit some of the input signals will be set to the appropriate values to implement the partitioning.

The autonomous testing process takes place in three test phases. In the first two phases (P_1, P_2), HI and LI functions of all N_1 blocks will be verified. In the third phase block N_2 is tested exhaustively. Hence, the input register must be reconfigured to provide three input test pattern generators, one for each testing phase.

The design of this reconfigurable input register is shown in Fig. 8a. The boxes marked MFF are modified, edge-triggered D flip-flops. They function as a four input or a two input multiplexer and a D flip-flop. During the normal mode of the operation (P_0), these flip-flops are loaded with input pin values (denoted by C_n, M, A0, B0, . . . etc.). During the three test phases the input flip-flops are loaded with appropriate inputs, provided by three feedback circuits, to generate the test patterns and input values necessary to test all the partitions exhaustively. Lines marked Z_1, Z_2, and Z_3 are used by the control circuitry to detect the end of each test phase.

By redesigning the input register so that the multiplexer functions are incorporated in each flip-flop, the delay which would be introduced by a separate multiplexer can be eliminated. Also, the number of additional gates is reduced to a minimum. Figures 8b depicts the gate level design of the MFF block labled (b) on figure 8a. The multiplexer delay is eliminated since the multiplexing stage is incorporated in the first stage of the flip-flop. Only three additional gates are needed to implement the four input multiplexer. Not all the MFFs of figure 8a need a four input multiplexer, hence three additional gates. Figure 8c depicts the gate level design of the Mff block labled (c) on Figure 8a. Using preset and clear lines for loading "1" and "0" in P_1 and P_3, respectively, only a two input multiplexing capability is required. This can be achieved with no extra delay and only one extra gate, as shown in figure 8c.

Figure 9a shows the design of the output register-signature analyzer. The MFF boxes represent the combination of a two input multiplexer and a D flip-flop. This combination requires the addition of one gate per output flip-flop. During normal operation (P_0), the output circuit functions as a register. During the three test phases, the output circuit is reconfigured to a signature analyzer, using the feedback circuit provided. Figure 9b shows the gate level design of the MFF block labled (b) in figure 9a. The hardware overhead is 11 gates, or 23% of the exsisting hardware. Again this reconfiguration does not add any delay to the unmodified circuit.

Figure 10 shows the testing control circuit . This circuit includes a four phase counter that provides the phase signals P_0 (normal operation phase), P_1, P_2, and P_3 (test phases) for reconfiguring the input and output registers. The transition from P_0 to P_1 is provided by the input TEST pin. The AND gates 1, 2, and 3 detect the completion of each test phase and advance the test phase counter to the next phase. Upon completion of the test the counter will return to P_0 and the signature in the output register will be decoded by AND gate 4 (this is done by connecting Y_i or Y_i complement, as appropriate, to the inputs of AND gate 4). If the signature in the output register is correct the OK output signal will be high, indicating the correctness of the ALU output.

The number of gates added to make this circuit autonomously testable is 22% of the gate count of the original design. The normal speed of operation is unaffected.

MODIFICATIONS FOR CMOS CIRCUITS

The design technique presented above is restricted to combinational circuits or circuits which can be restructured as combinational for testing purposes. It also requires that faults do not change a combinational circuit into a sequential circuit. This assumption is valid for most faults and is commonly used in all automatic test pattern generation systems. However it is known that CMOS circuits are subject to stuck-open faults which do introduce sequential behavior into combinational circuits, [Wadsack,78]. A technique for modifying CMOS circuits so that the autonomous testing technique can be used will be presented next.

C/D Tester For CMOS Stuck-Open-Faults

The logic output of a CMOS gate, depending on its input, is either connected to a power source (pulled up) or is connected to ground voltage (pulled down). Normally, appropriate circuitry is provided to do the gate function. Consider the CMOS NOR gate depicted in Fig. 11. The two P-Channel Transistors (1) and (2) are pull-up transistors; they connect the output to V_{dd} if and only if A=B=0. Similarly transistors (3) and (4) are pull-down transistors; They connect the output to ground if either A or B is 1. For such a gate the output is either connected to the ground or to V_{dd}.

A stuck-open-fault, however, will cause the output to be neither connected to V_{dd} nor to the ground voltage, for some input patterns. For example , suppose line (4), in Fig. 11, is broken; then for input AB=01 neither the pull-up circuit nor the pull-down circuit will be active. However the output will remain in the previous state, charged or discharged. Since the output of the previous pattern may be the same as the output for the 01 input pattern, the effect of the fault on the output may or may not be observed. Therefore, depending on the order of the input patterns, the stuck-open-faults may go undetected. To detect such faults in CMOS, special considerations must be given to the arrangement and the order of the test patterns; this, in general, is a formidable and difficult task.

Wadsack [Wadsack,78] has proposed models for stuck-open-faults in CMOS circuits which require the introduction of a latch at each gate output. Use of these models increases the complexity of test pattern generation so as to make its practicality questionable. Exhaustive testing is not suitable for stuck-open-faults. However, by use of the C/D technique to be described next, a CMOS circuit can be modified so that it can be tested exhaustively (or by any other technique which relies on the circuit remaining combinational in the presence of a fault).

Using the C/D technique, additional test circuitry is placed on the CMOS chip to charge or discharge the stuck-open lines. Figure 12 shows a modified NOR circuit. Two additional lines, TEST and C/D, are used to control the charging and discharging gate (5) while testing for stuck-open-faults. The technique consists of charging and discharging the output line after the application of each test pattern. If the output line is not stuck-open, the output logic value will return to its correct value after the charging or discharging is ceased. However, if the line is stuck-open, it will remain charged or discharged after charging or discharging (for the charging cycle TEST input line of the C/D tester must have a higher voltage than the C/D line).

The C/D testing method can be utilized to detect the stuck-open-faults on all the lines of a CMOS combinational circuit. The block diagram of a CMOS circuit with its C/D testing circuitry is shown in Fig. 13. Again after the application of each test pattern, C/D and TEST lines are used to detect the stuck-open faulty lines.

This method can be used in any CMOS combinational circuit. It allows the use of any suitable set of test patterns regardless of the order of the patterns. In particular, using the C/D method, CMOS circuits can be exhaustively tested.

SUMMARY AND CONCLUSIONS

A procedure for designing general logic networks in a structure which is capable of testing itself is presented. When a test mode signal is applied to the chip, an automatic restructuring is carried out such that:

1. The registers in the network are reconnected in the form of modified linear feedback shift registers. Some of these registers are

ɔounters which cycle through all states. Others receive the output of the combinational portions of the network and compute a compacted network response (signature) which is automatically compared with the response of a fault-free network. All other feedback paths in the network are broken.

2. The network is automatically partitioned so that the resulting combinational subnetworks can be cycled through all possible input combinations in the time allowed for testing.

This technique is an improvement over the scan techniques used for improving network testaoility in that:

1. Test pattern generation, fault modelling, and storage of input test patterns and output responses are not required since the subnetworks are cycled through all possible input patterns.

2. The time to shift in the test pattern and shift out the network response is eliminated, thereby reducing the testing time. This reduction is compensated, in part, by the need to apply all input patterns to the subnetworks, but this time can be controlled by choosing the size of the subnetworks.

3. The need for expensive external testing equipment is eliminated.

Other techniques for self test have relied on pseudo-random test inputs. These have the disadvantages of uncertainty about what is a sufficient test length and the fault coverage obtained. By contrast, the exhaustive test discipline used here will detect all faults which either increase the number of states nor disturb the structure assumed when partitioning is necessary. Since the partitioning is done in terms of very large blocks, it should not be difficult to study the faults which can invalidate the partitions, if such study seems necessary. A specific technique for handling that class of faults which is known to convert combinational CMOS circuits to sequential circuits has been presented. It is our belief that a competitive technique for implementing self-testing designs has been presented. The partitioning (or segmenting) procedure should be applicable for test pattern generation in situations where self test is not considered necessary or economical. The advantages of not requiring detailed fault models and detection of almost all faults rather than just stuck-at faults would still be retained.

Several interesting questions which could lead to improvements in the technique described here have been identified:

1. What is the best set of test patterns for exhaustive testing of a multiple-output network in which none of the outputs depend on all of the network inputs? An algorithm for answering this question has been developed and is presented in [McCluskey,81].

2. How can a good (measured by short test length, high fault coverage, minimal additional circuitry for self-test, etc.) segmentation (or partition) of a network be obtained automatically? Some preliminary results on this question have been obtained, but it is currently the subject of ongoing research.

3. What are the best techniques for compacting the output responses? This question is currently being studied in an attempt to develop more accurate estimates of the coverage and cost of the various possible alternatives.

4. What are the best techniques for guaranteei o the integrity of the internal test circuitry and the connections to the chip pins? Several promising approaches are currently under study.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation Grant Number MCS-7904864 and in part by Army Research and Development Command Contract No. DAAK20-80-C-0266. The authors wish to thank the members of the Center for Reliable Computing at Stanford University for their helpful criticism. The authors also wish to thank Lydia Christopher for her help in preparing this paper.

REFERENCES

[Benowitz,75] N. Benowitz, D.F.Calhoun, G.E. Alderson, J.E. Bauer and C.T. Joeckel, "An advanced fault isolation system for digital logic," IEEE Trans. Comput., Vol. C-24, No. 5, pp. 489-497, May 1975.

[Bottorff,77] P.S. Bottorff, R.E. France, N.H. Garges, and E.J. Orosz, "Test generation for large logic networks," Proc. 14th Design Automation Conf., June 1977, pp. 479-485.

[Bozorgui,80] S. Bozorgui-Nesbat and E.J. McCluskey, "Structured design for testability to eliminate test pattern generation,"Proc. 1980 Int'l Fault-Tolerant Computing Symp., Oct. 1980, pp. 158463.

[Breuer,76] M.A. Breuer and M.D. Friedman, Diagnosis and Reliable Design of Digital Systems, Computer Science Press, Woodland Hills, Ca., 1976.

[Eichelberger,77] E.B. Eichelberger and T.W. Williams, "A logic design structure for LSI testability,"Proc. Π14thgn______ Automation Conf., June 1977, pp. 462-468.

[Eiki,80] H. Eiki, K. Inagaki, and S. Yajima, "Autonomous testing and its application to testable design of logic circuits,"Proc. 1980 Int'l Fault-Tolerant Computing Symp., Oct. 1980, pp. 173-178.

[Fasang,80] P.P. Fasang, "BIDCO, built-in digital circuit observer, Proc. 1980 IEEE Test Conference, Nov. 1980, pp.261-266.

[Gschwind,75] H.W. Gschwind and E.J. McCluskey, Design of Digital Computers, Springer-Verlag, New York, 1975.

[Konemann,80] B. Konemann, J. Mucha, and G. Zwiehoff, "Built-in logic block observation technique," IEEE J. of Solid State Circuits, June 1980.

[McCluskey,80] E.J. McCluskey and S. Bozorgui-Nesbat, "Design for autonomous test,"Proc. 1980 IEEE Test Conf., Nov. 1980, pp. 15-21.

[McCluskey,81] E.J.McCluskey, "Verification Testing", Center for Reliable Computing Technical Report 81-3, Stanford University.

[Savir,80] J. Savir, "Syndrome-testable design of combinational circuits," IEEE Trans. Comput., Vol. C-29, No. 6, pp. 442-451, June 1980.

[Sedmak,79] R.M. Sedmak, "Design for Self-Verification: an approach for dealing with testability problems in VLSI-based designs," Proc. 1979 IEEE Test Conf., Nov. 1980, pp.112-124.

[Shedletsky,77] J.J. Shedletsky, "Random Testing: practicality vs. verified effectiveness," Proc. ____ Int'l Fault-Tolerant Computing Symp., June 1977, pp. 175-179.

[Smith,80] J.E. Smith, "Measure of the effectiveness of fault signature analysis," IEEE Trans. Comput., Vol. C-29, No. 6, pp. 510-514, June 1980.

[Stephenson,76] J.E. Stephenson and J. Grason, "A testability measure for register transfer level digital circuits," Proc. Int'l Fault-Tolerant Computing Symp., pp. 101-107, June 21-23, 1976.

[TI,81] The Engineering Staff of Texas Instruments Incorporated, Semiconductor Group, The TTL Data Book ___ Design Engineers, Second Edition, Texas Instruments Inc., 1981.

[Turino, 19791 J. Turino, Design for Testability., Logical Solutions Inc., Campbell, California, 1979.

[Wadsack, 1978] R.L. Wadsack, "Fault modeling and logic simulation of CMOS and MOS integrated circuits," Bell System Tech. J., May-June 1978, pp. 1449-1473.

[Williams, 1973] M.J.Y. Williams and J.B. Angel, "Enhancing testability of large scale integrated circuits via test points and additional logic," IEEE Trans. on Comput., Vol. C-22, No. 1, pp. 46-60, Jan. 1973.

[Williams, 19791 T.W. Williams and K.P. Parker, "Testing logic networks and designing for testability," Computer, pp. 9-21, Oct. 1979.

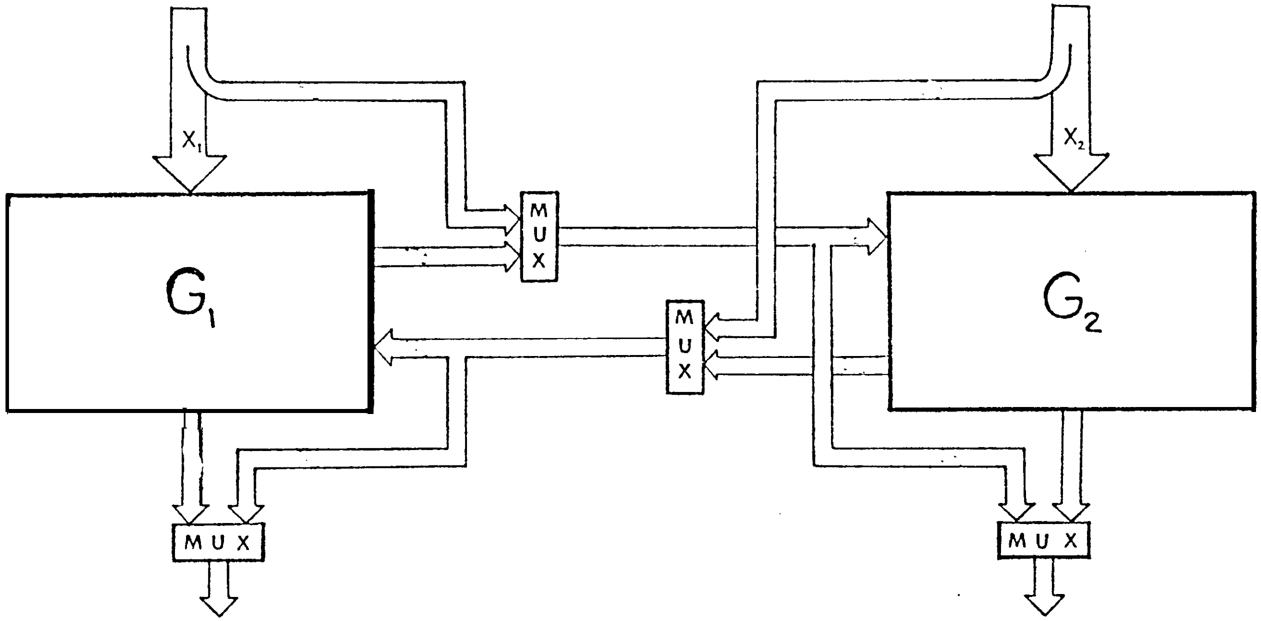


Figure la. General hardware partitioning scheme using multiplexers.

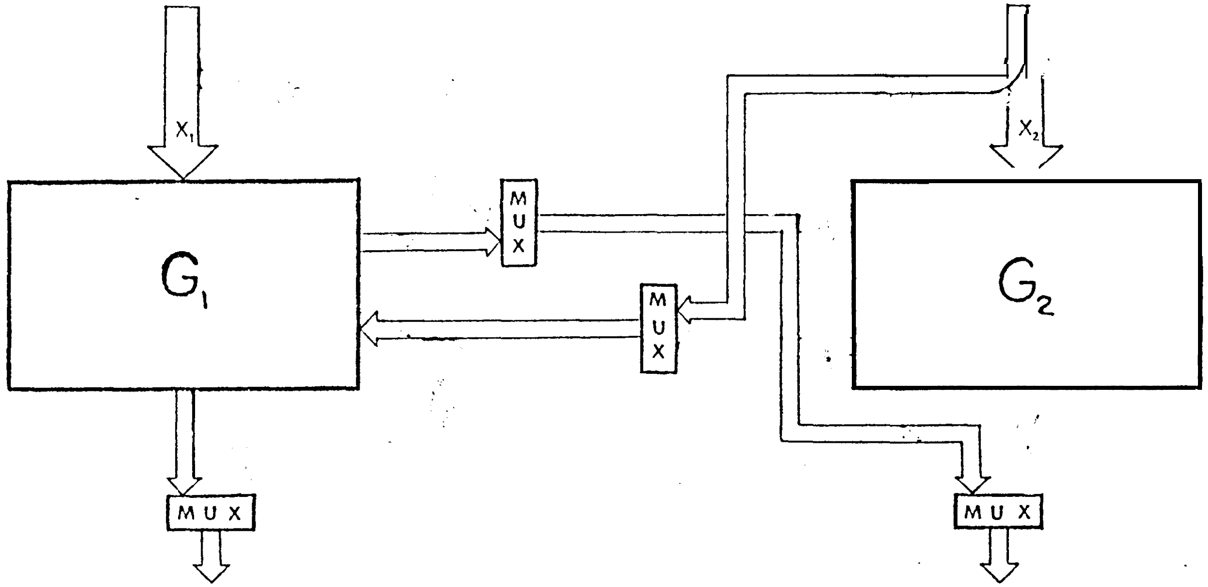


Figure 1b Configuration to test subcircuit G_1

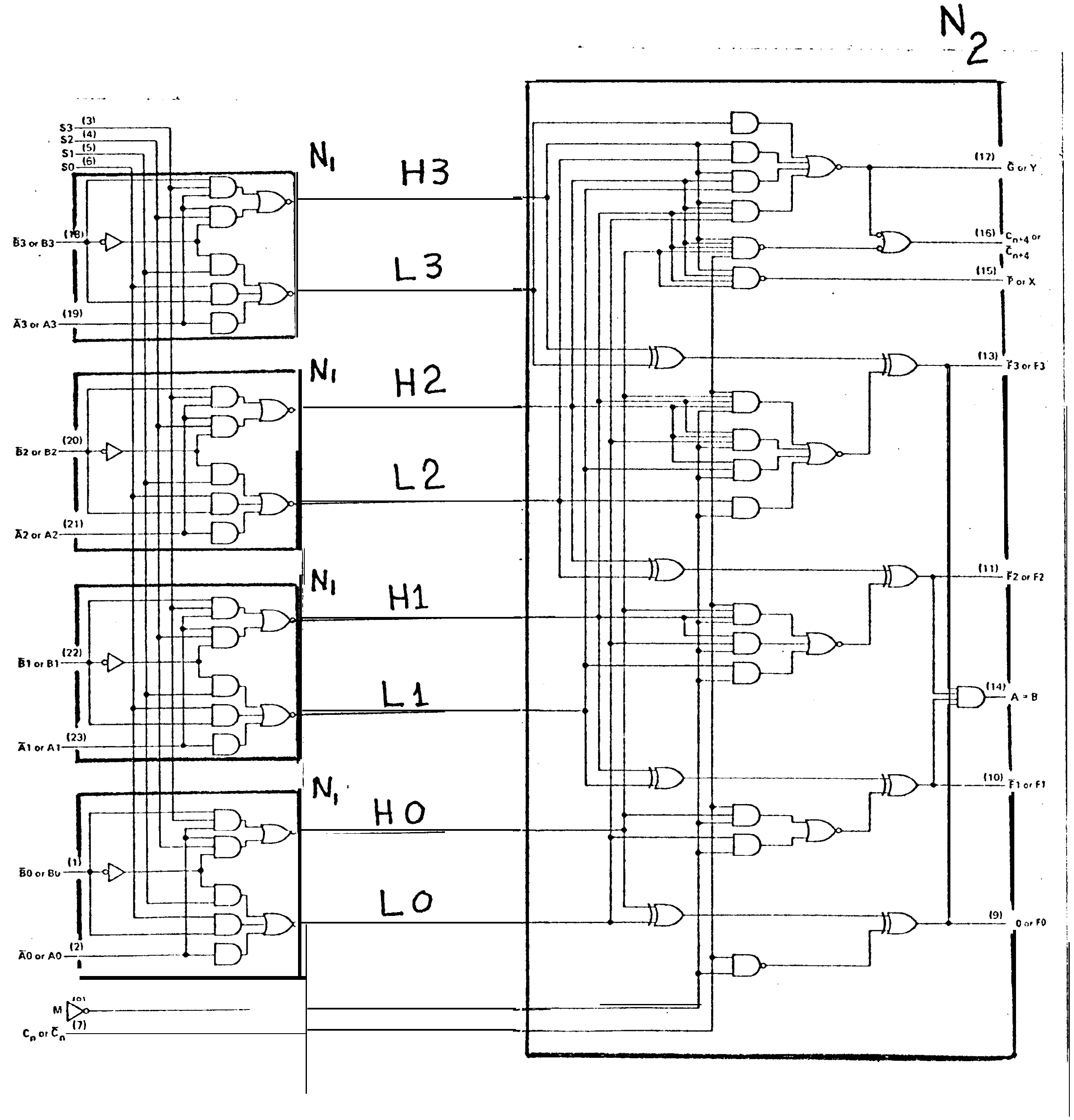


Figure 2a. 74181 ALU/FUNCTION GENERATOR and the partition blocks.

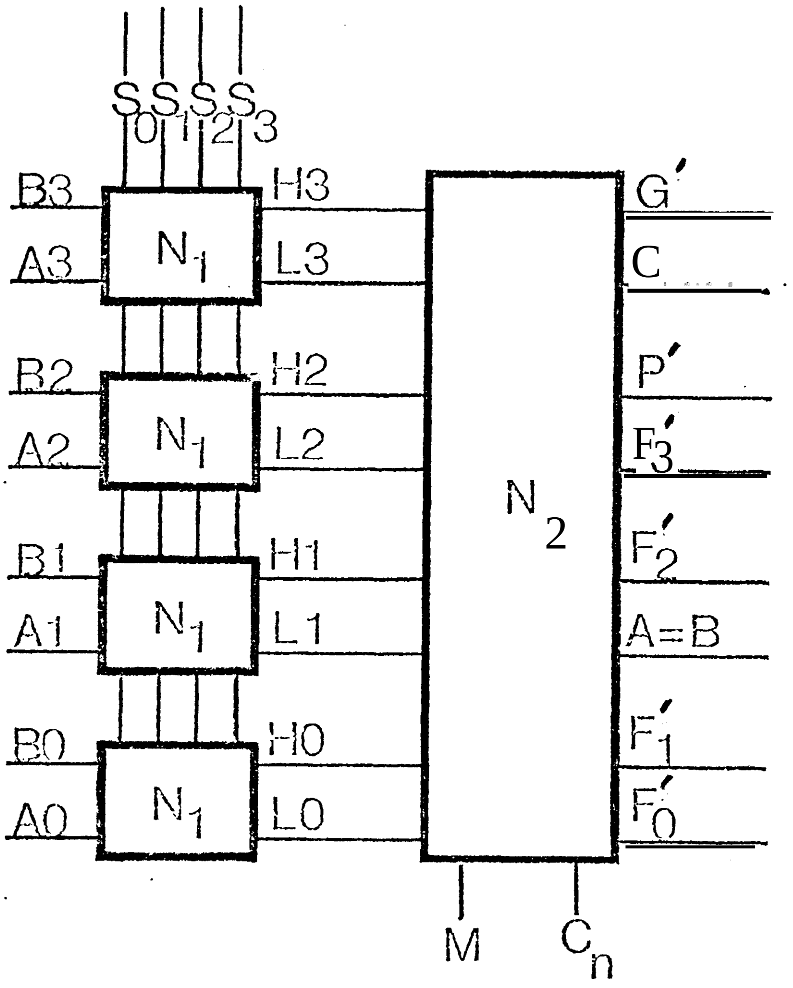
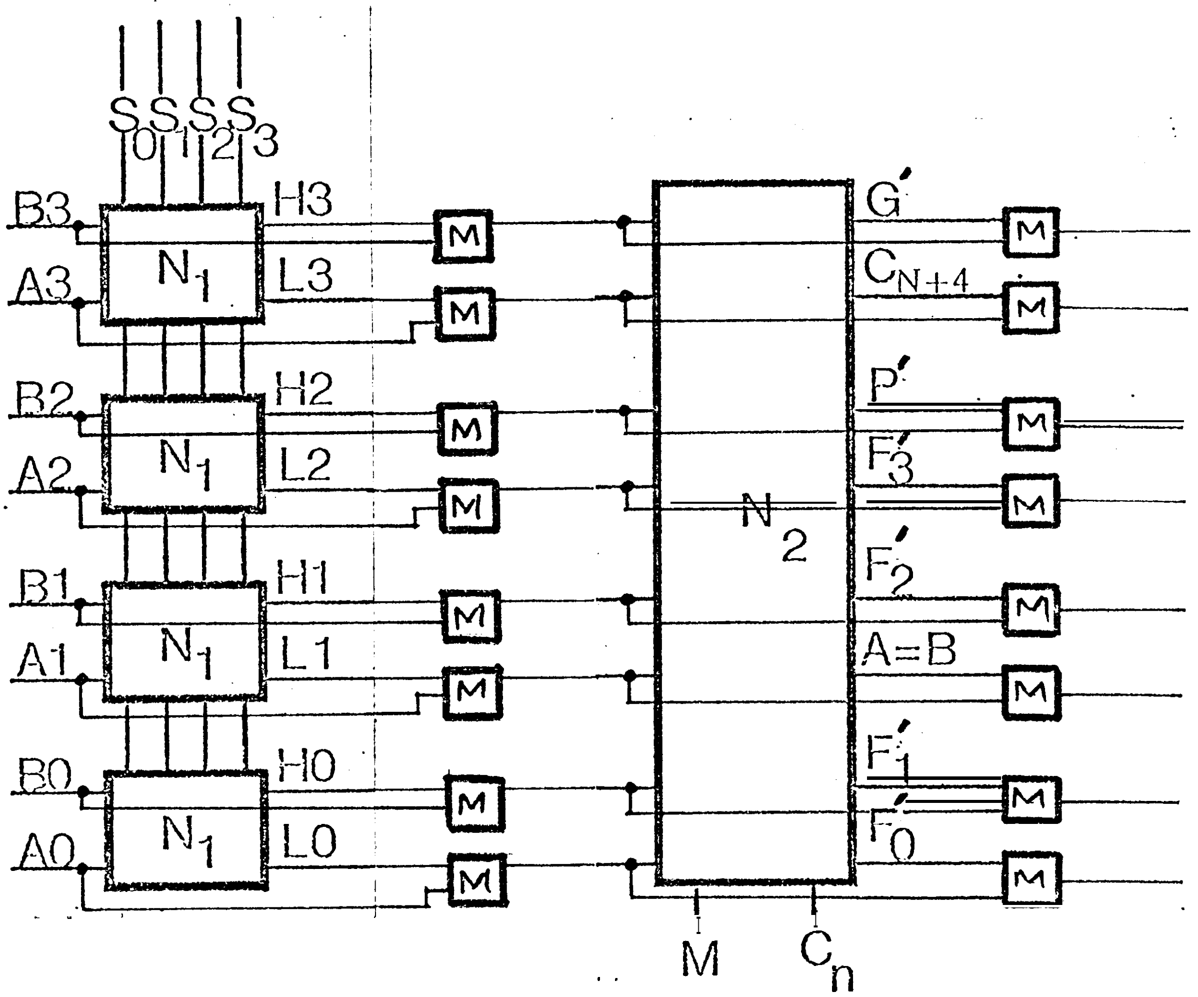


Figure 2b. Block diagram of the partition blocks.



M= Multiplexer

Figure 3. Multiplexer partitioning of 74181 ALU/FUNCTION GENERATOR.

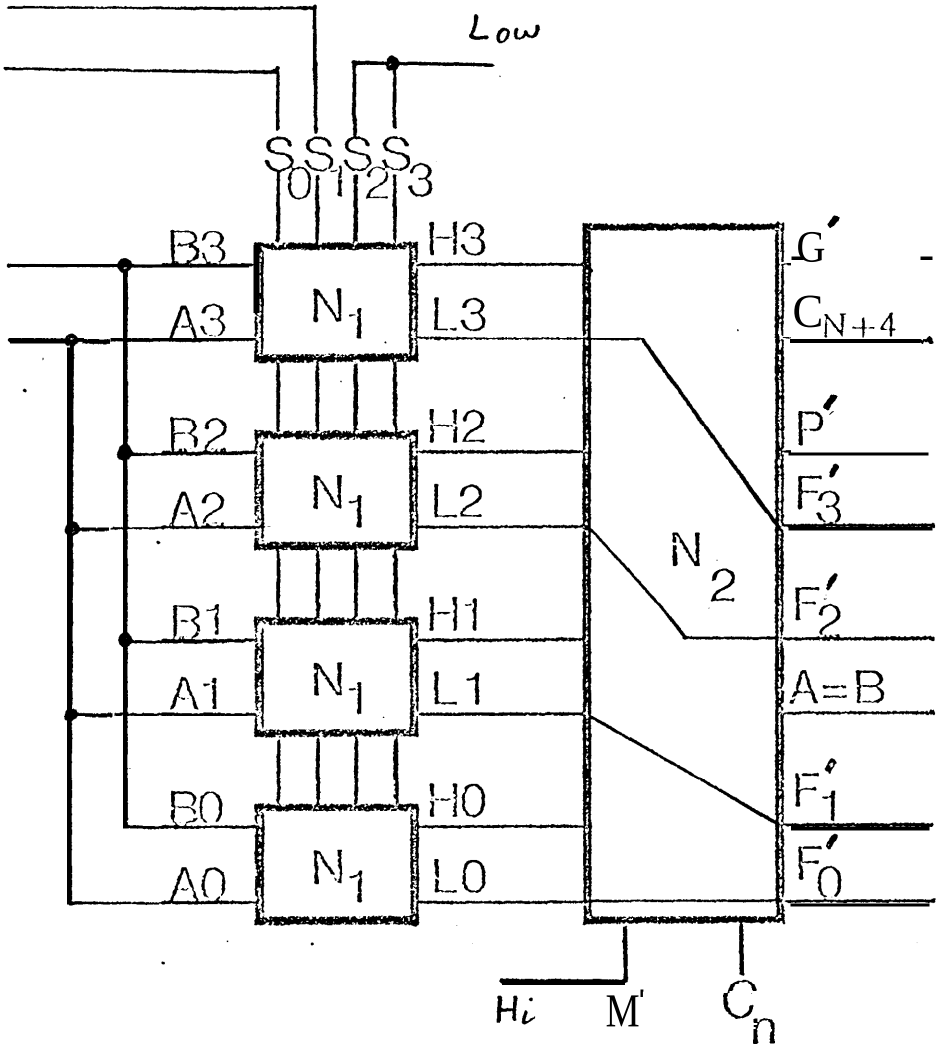


Figure 4a. Sensitized partitioning arrangement to test LI function.

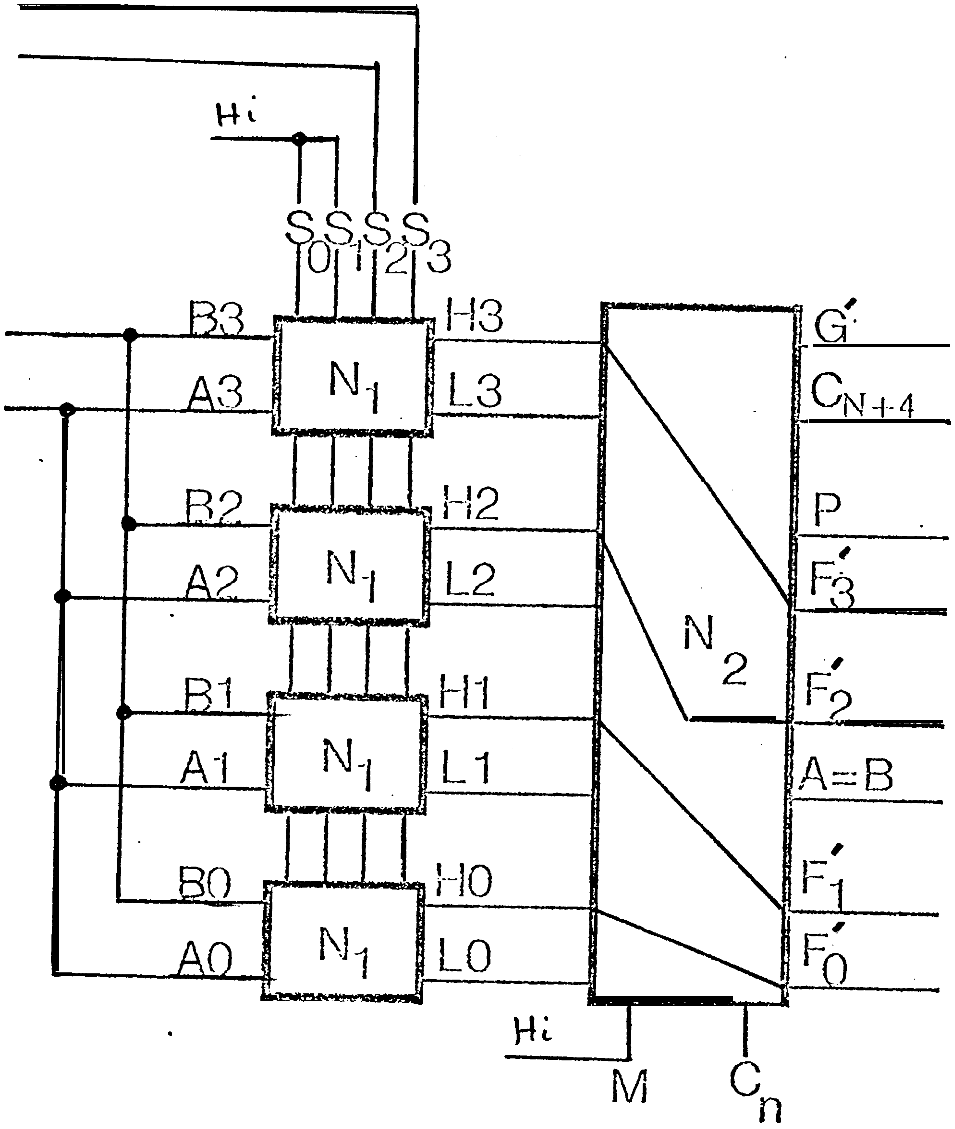


Figure 4b. Sensitized partitioning arrangement to test HI functions.

	N	S	MODE
M =	1	X	NORMAL OPERATION
	0	0	INPUT GENERATOR
	0	1	SIGNATURE ANALYZER

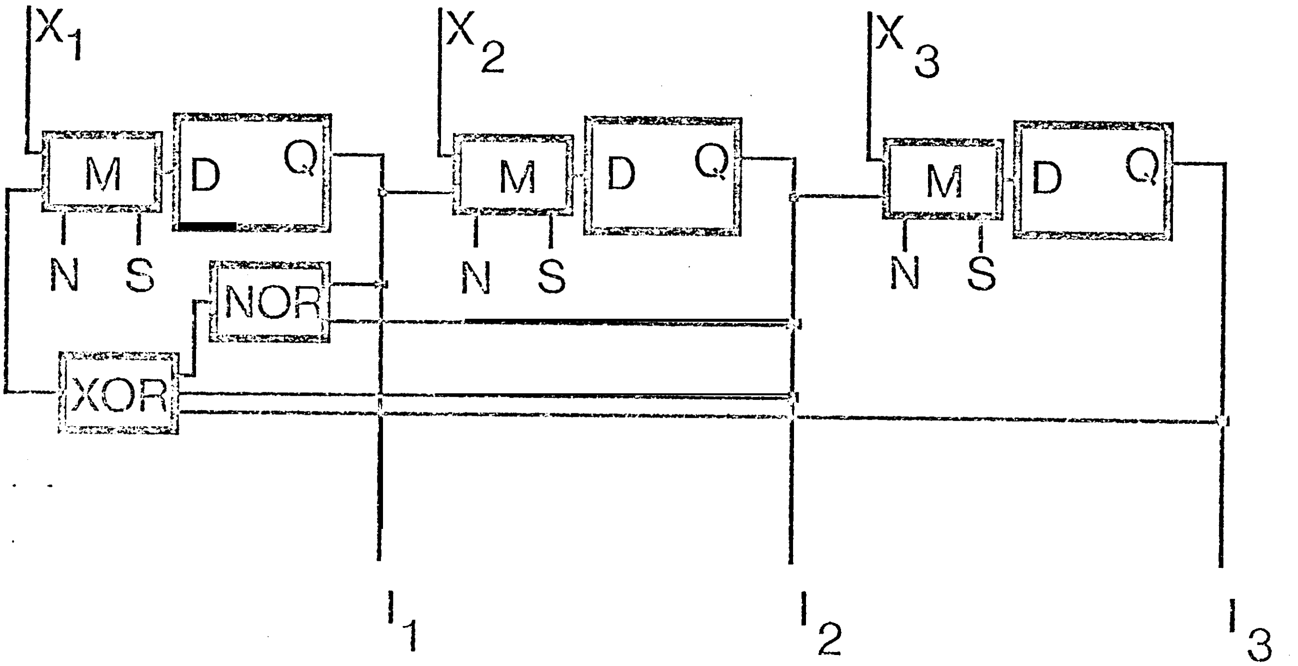


Figure 5. A reconfigurable 3-bit LFSR module.

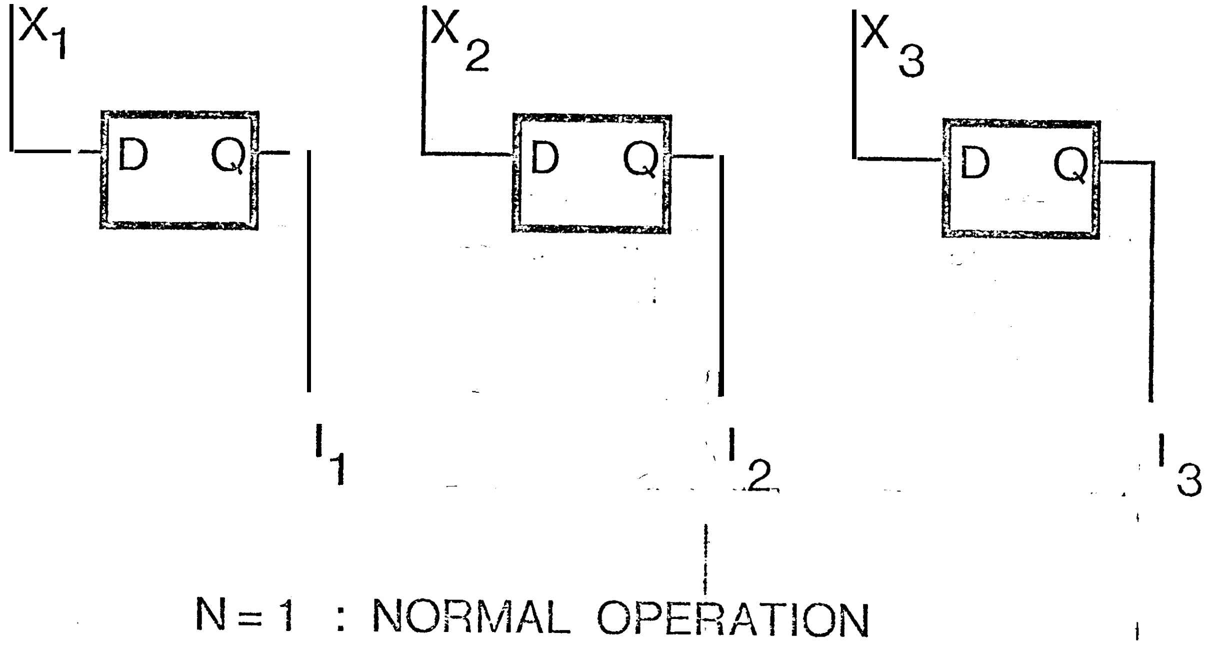


Figure 6a. A reconfigurable 3-bit LFSR module in register mode

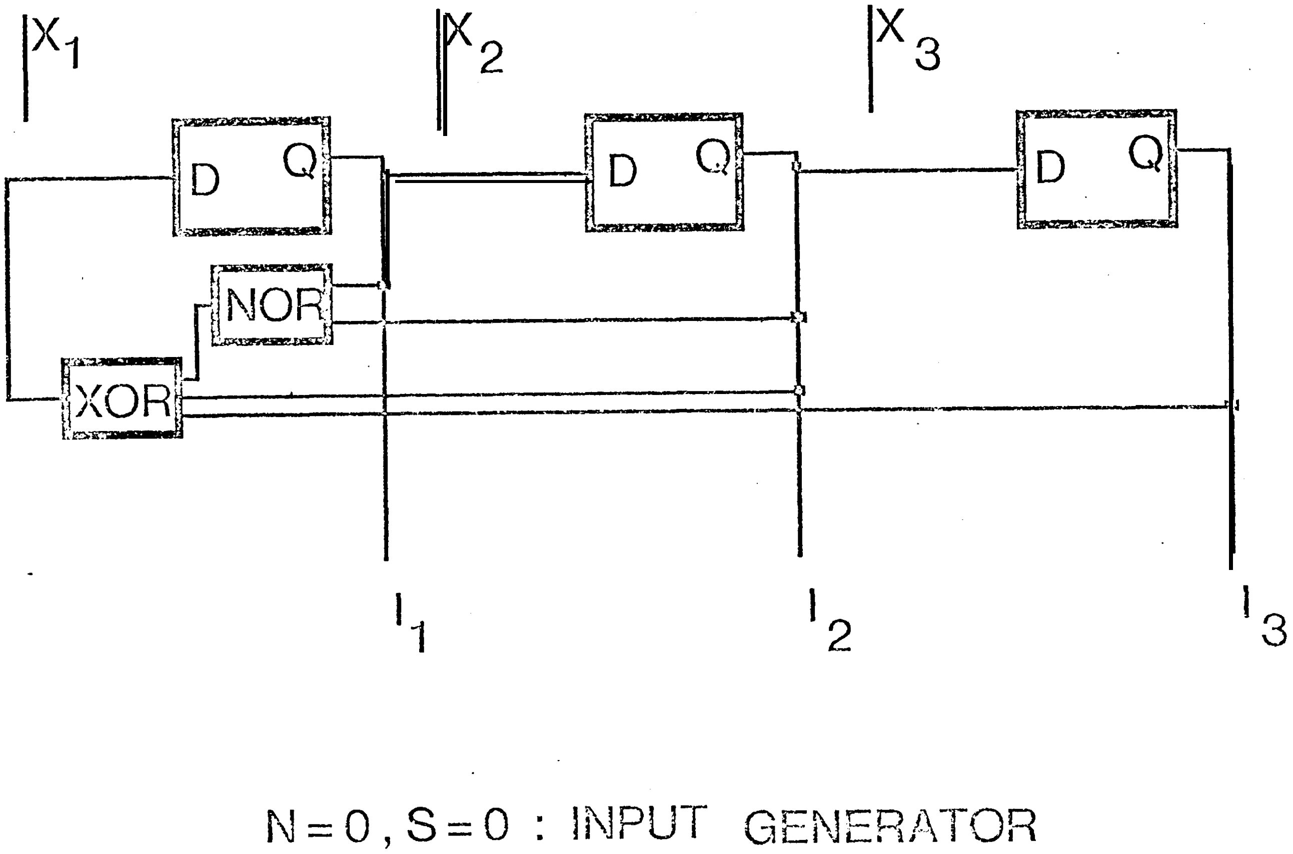
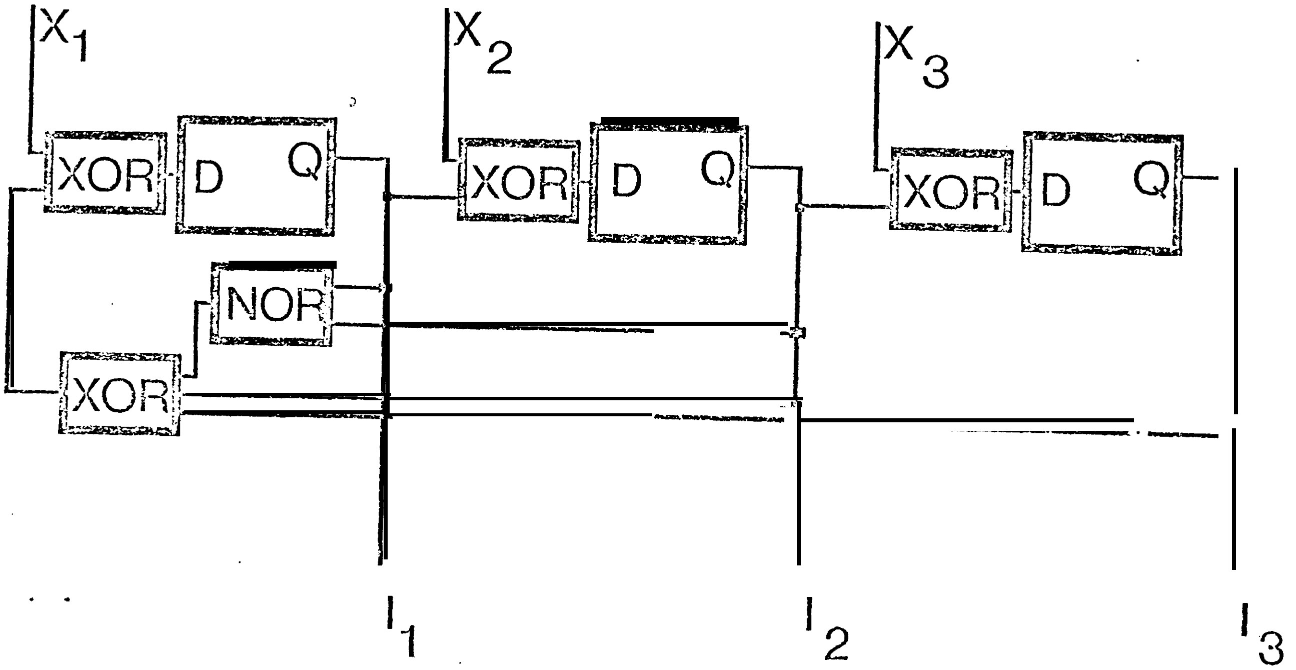


Figure 6b A reconfigurable 3-bit LFSR module in test generation mode



N = 0 , S = 1: OUTPUT SIGNATURE ANALYZER

Figure 6c. A reconfigurable 3-bit LFSR module in signature analyzer mode.

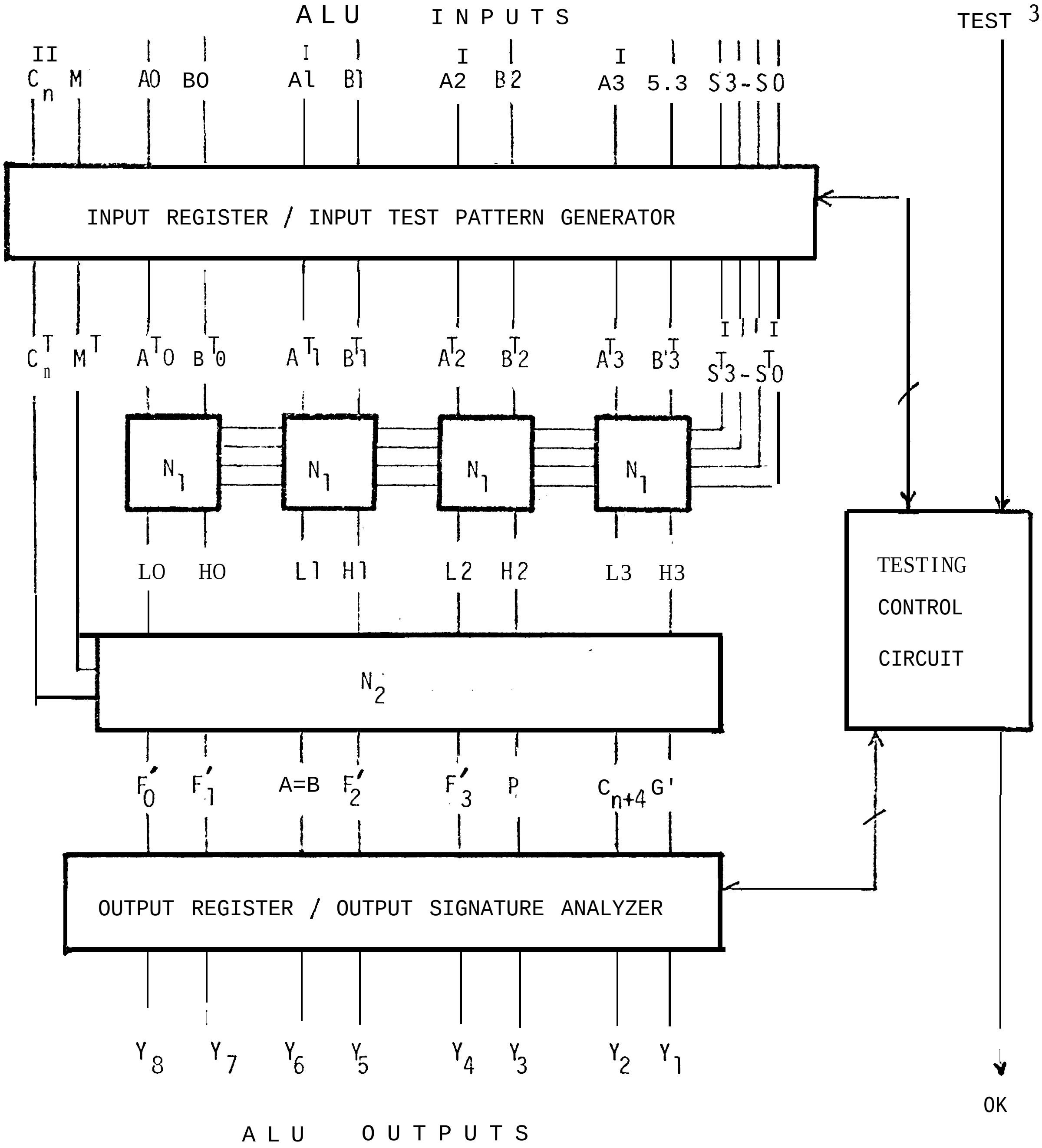

ALU INPUTS
TEST
C_n M A0 B0 A1 B1 A2 B2 A3 B3 S3-S0
INPUT REGISTER / INPUT TEST PATTERN GENERATOR
C_n^T M^T A^T_0 B^T_0 A^T_1 B^T_1 A^T_2 B^T_2 A^T_3 B^T_3 S^T_3-S^T_0
N_1
N_1
N_1
N_1
L0 H0 L1 H1 L2 H2 L3 H3
N_2
TESTING
CONTROL
CIRCUIT
F_0' F_1' A=B F_2' F_3' P C_{n+4} G'
OUTPUT REGISTER / OUTPUT SIGNATURE ANALYZER
Y_8 Y_7 Y_6 Y_5 Y_4 Y_3 Y_2 Y_1
OK
ALU OUTPUTS

Figure 7. Block diagram of the autonomously testable 74181 ALU/FUNCTION GENERATOR.

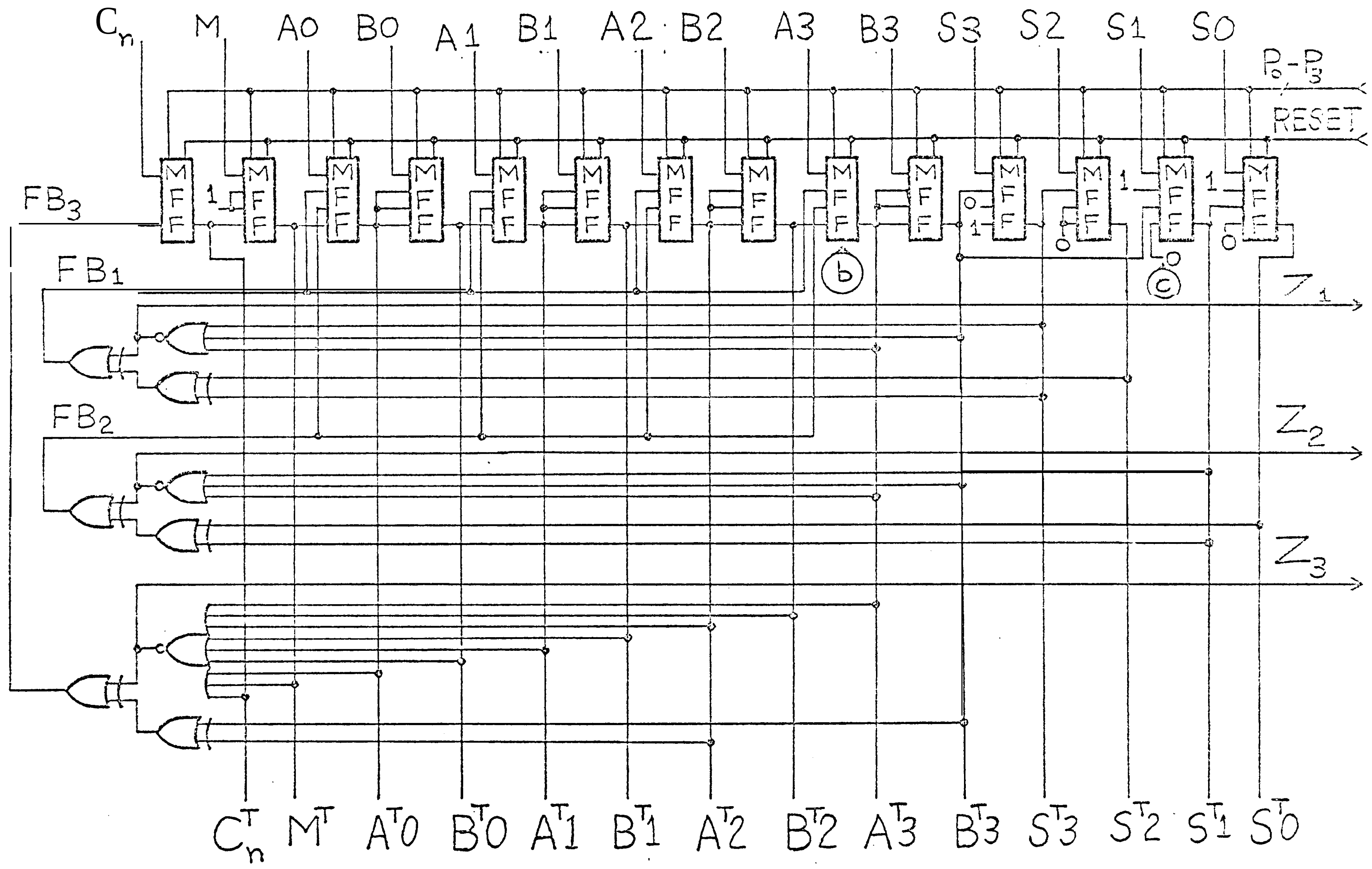


Figure 8a. Input register/input test pattern generator.

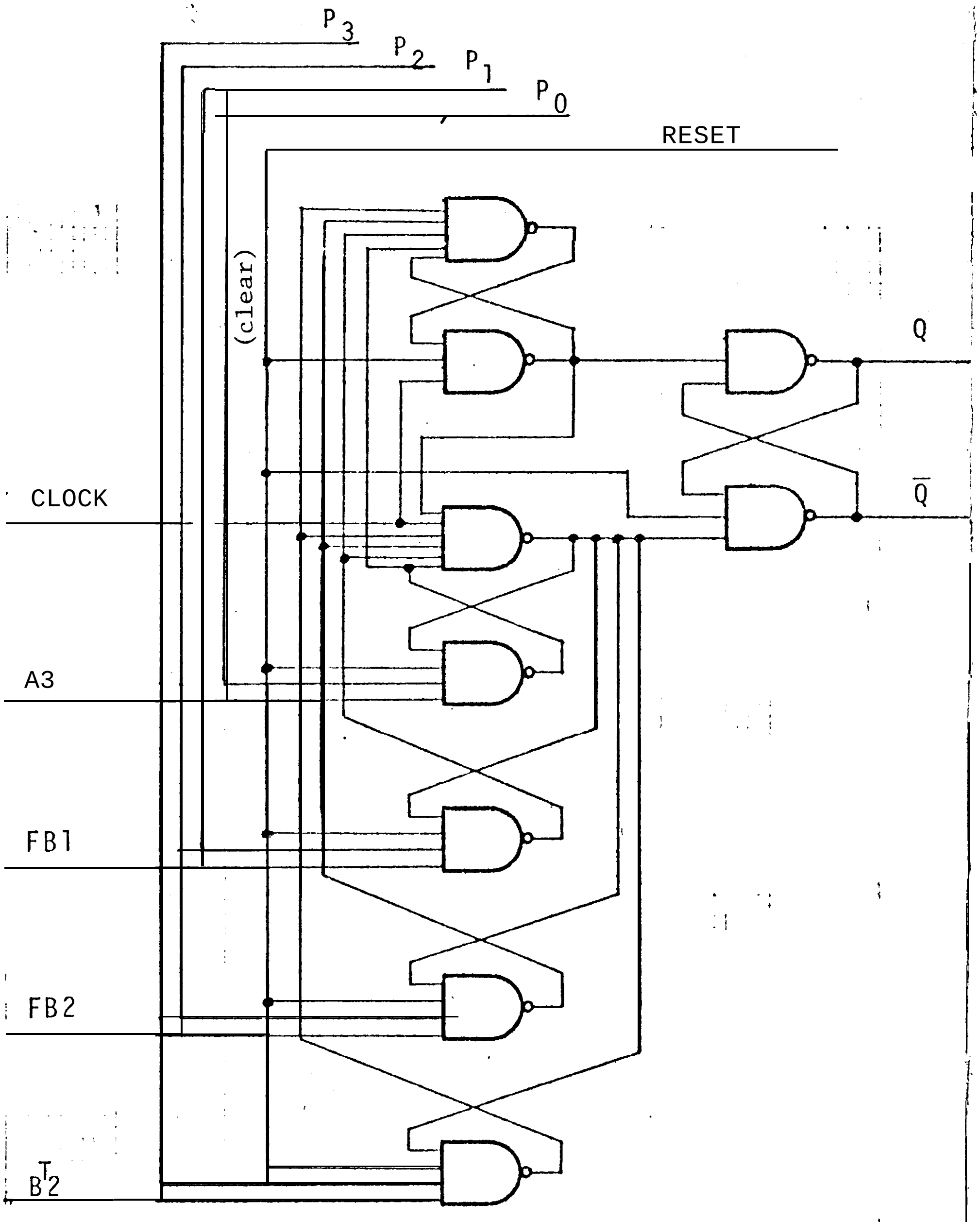


Figure 8b. Gate level design of MFF labeled (b) in Figure 8a.

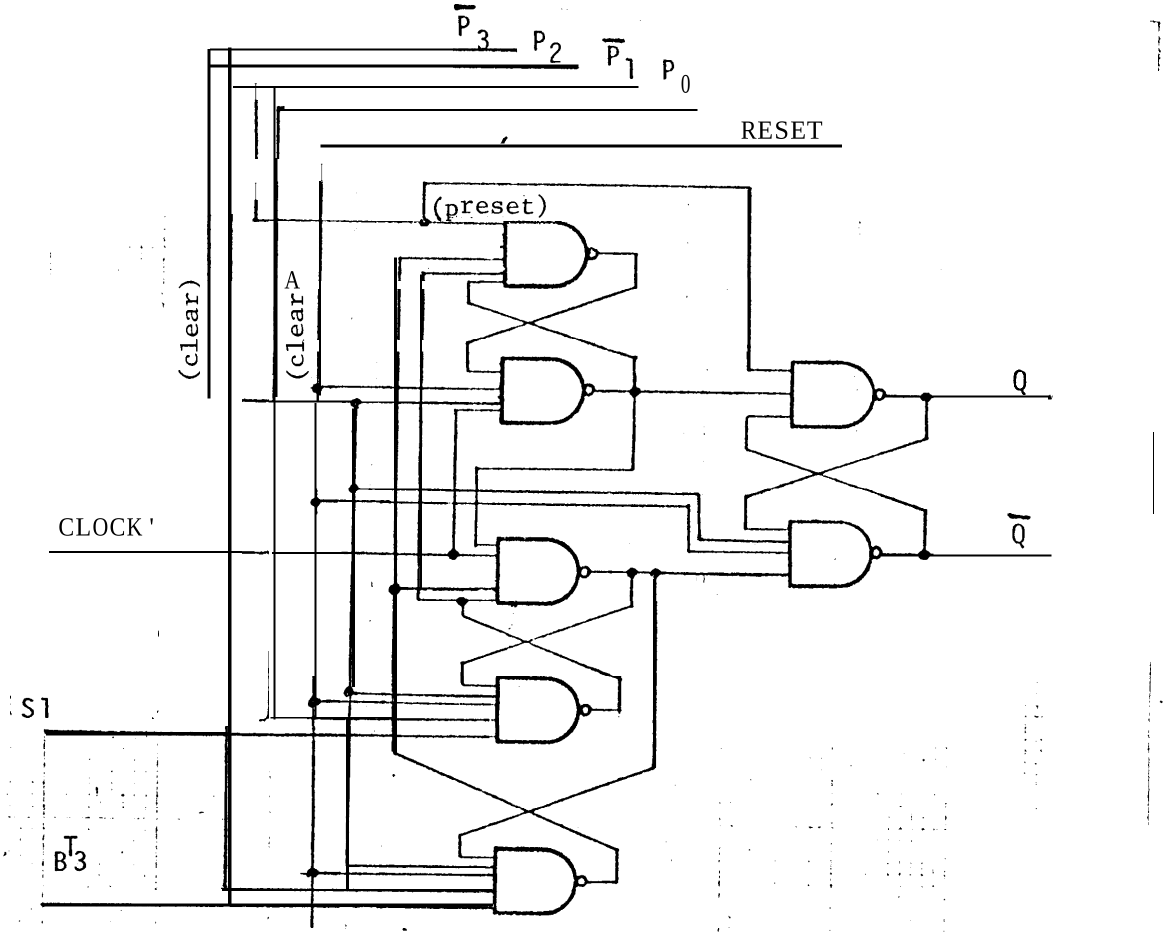


Figure 8c Gate level design of MFF labeled (c) in Figure 8a.

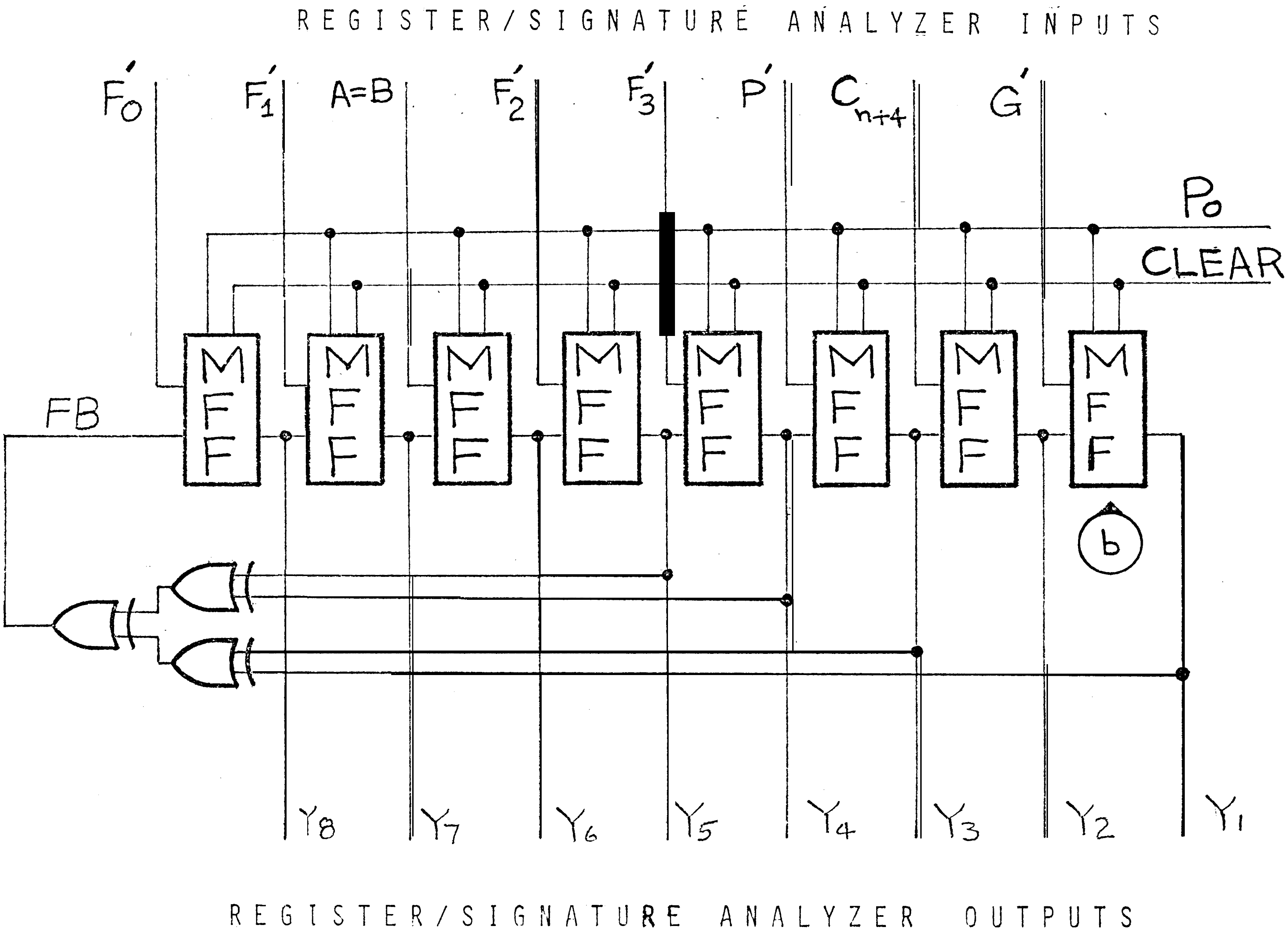


Figure 9a. Output register/output signature analyzer.

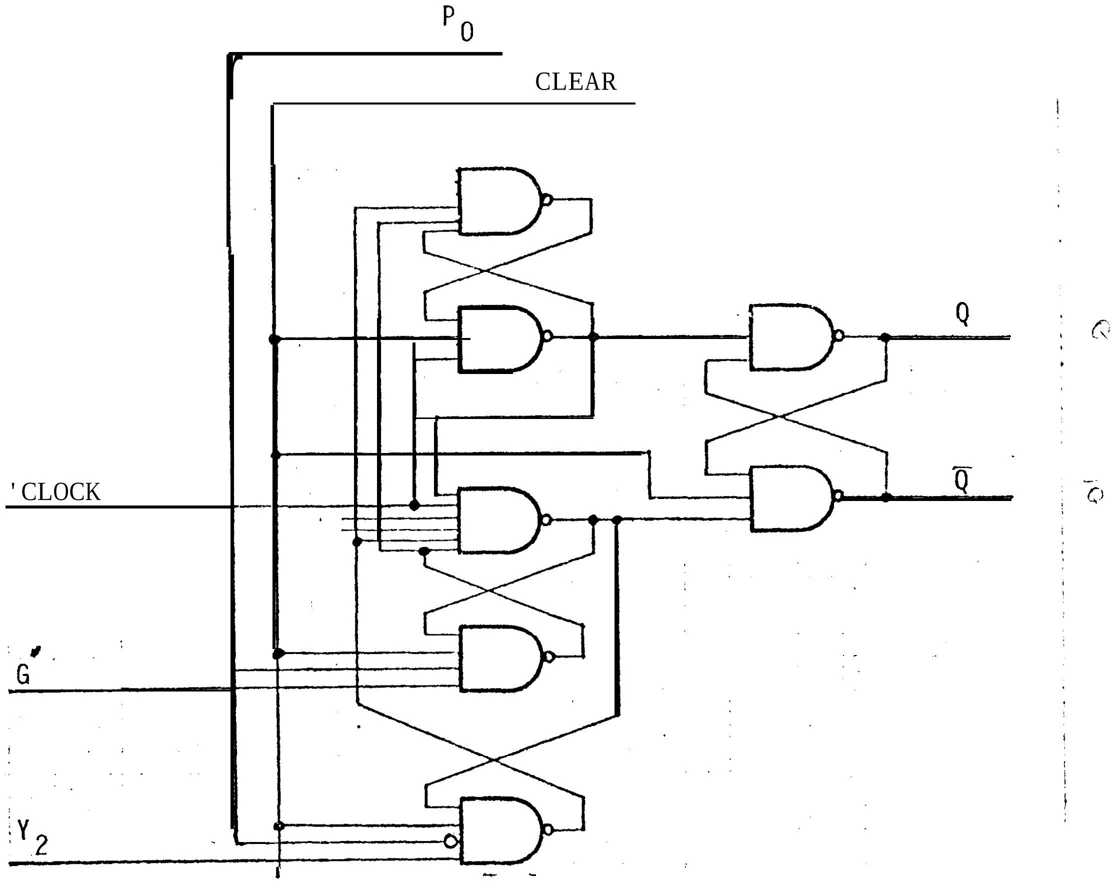


Figure 9b. Gate level design of MFF labeled (b) in Figure 9a.

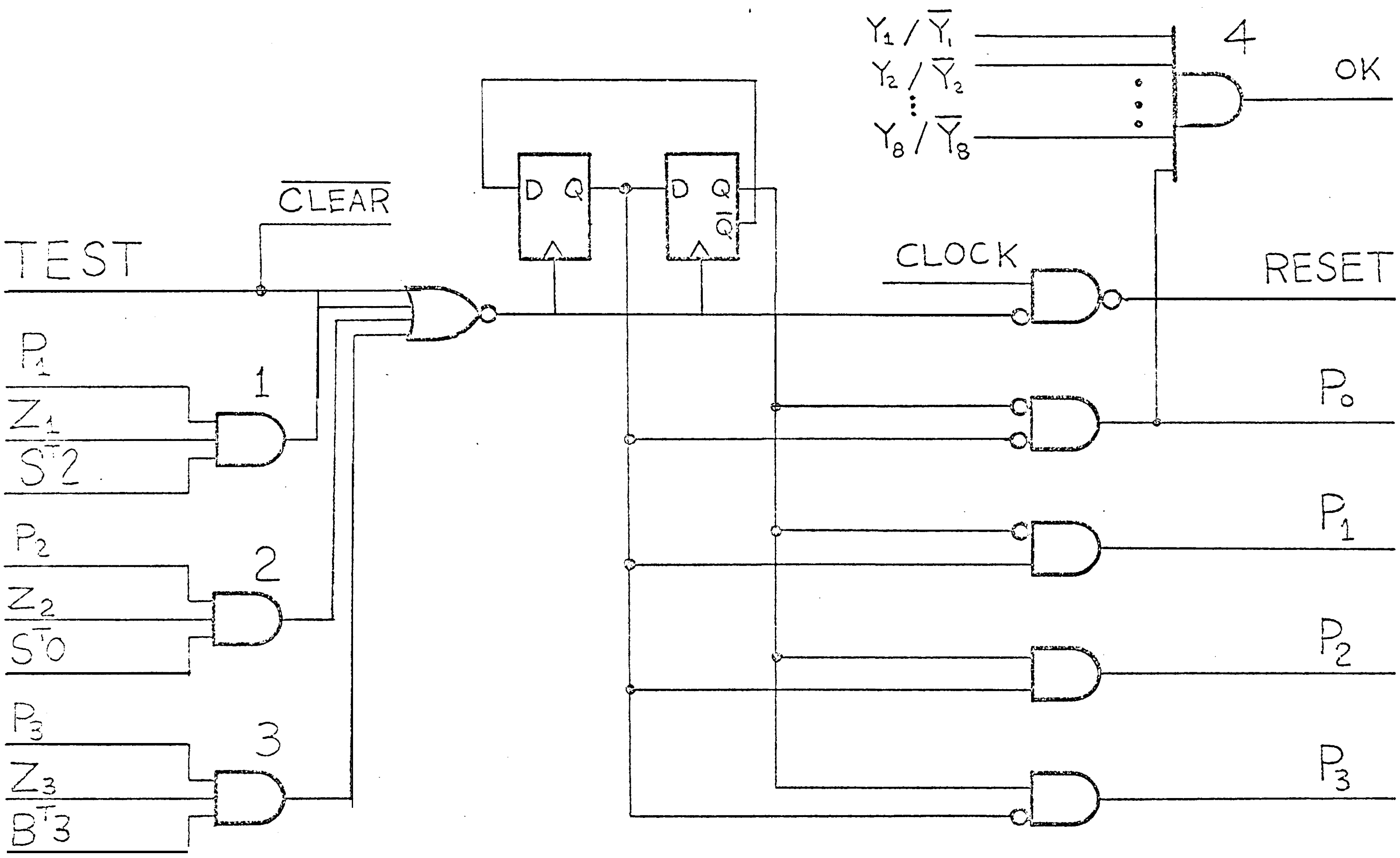


Figure 10. Testing control

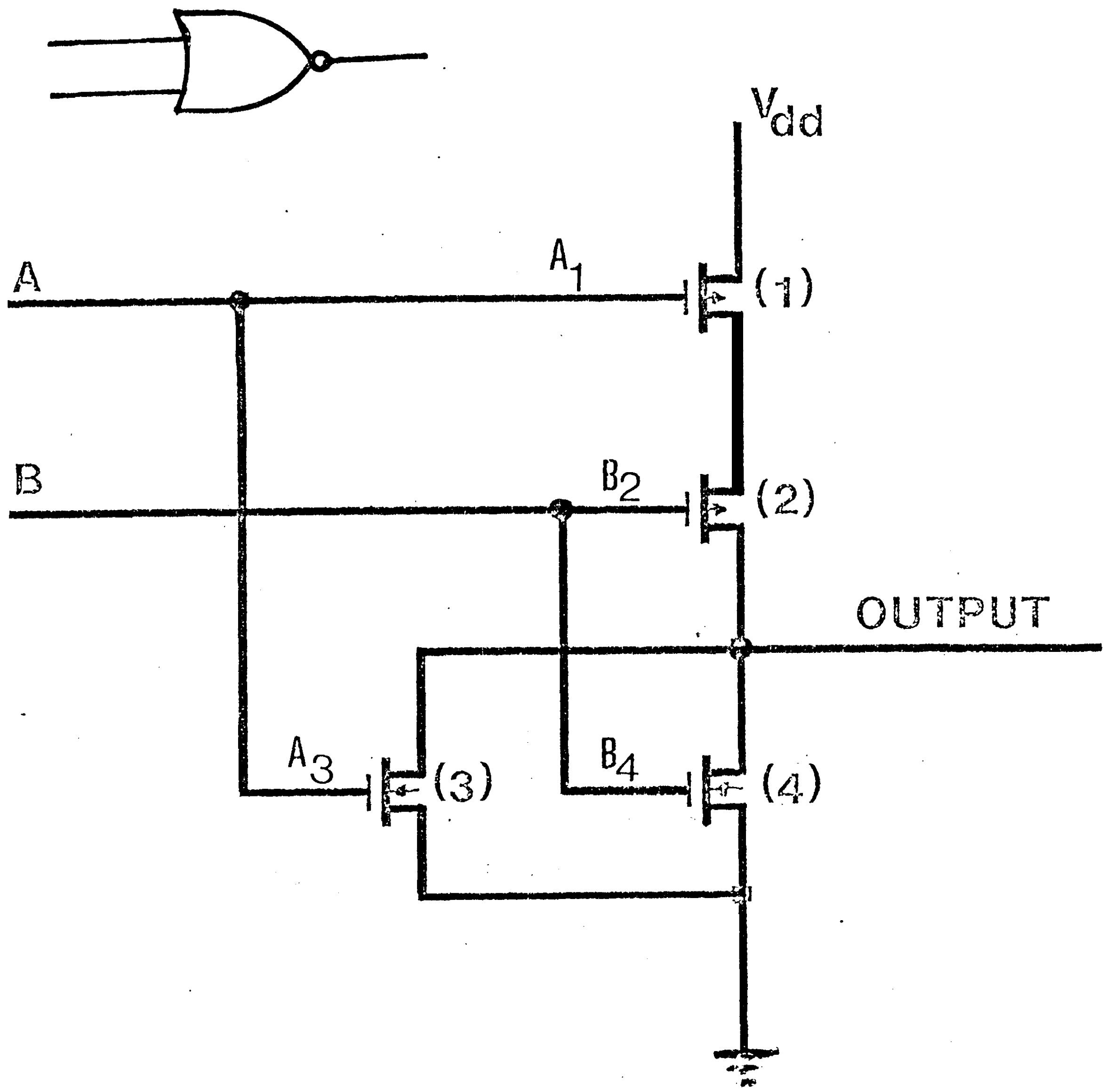


Figure 11. CMOS NOR gate.

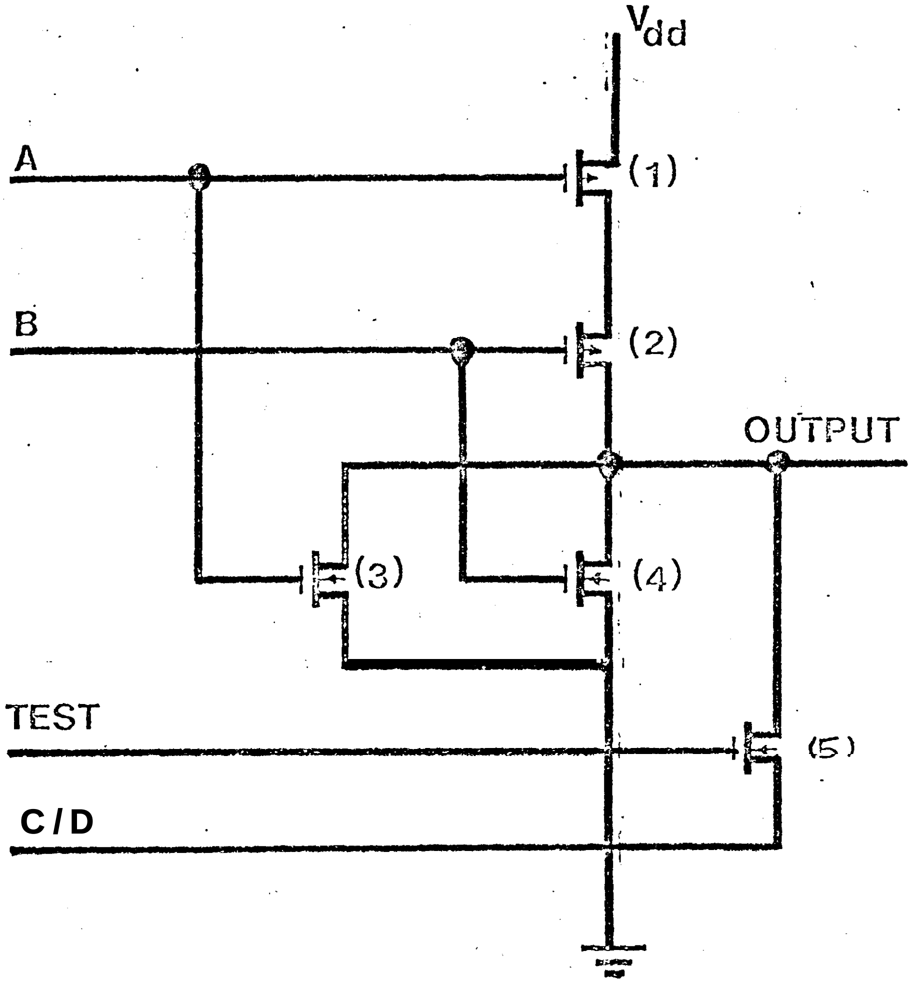


Figure 12. C/D tester for CMOS NOR *gate* stuck-open faults.

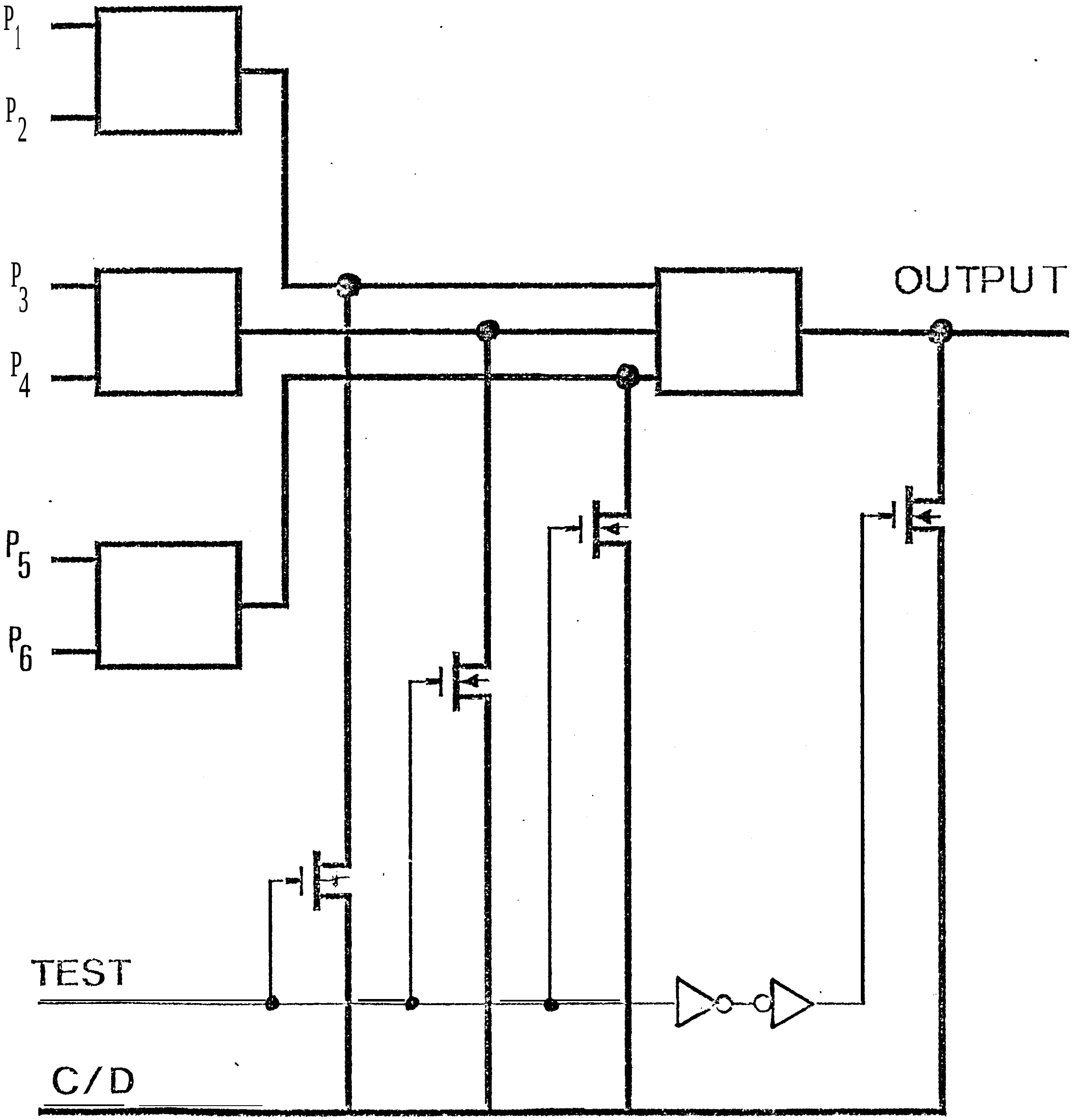


Figure 13. Organization of C/D tester circuitry for a CMOS.