Proc. Design Automation Conf. 1980.

[illegible]

DETECTING BRIDGING AND STUCK-AT FAULTS AT INPUT AND OUTPUT PINS OF STANDARD DIGITAL COMPONENTS*

Mark Karpovsky and Stephen Y. H. Su
Research Group on Design Automation
and Fault-tolerant Computing
School of Advanced Technology
State University of New York, Binghamton, NY 13901

ABSTRACT

Due to the advances in the integrated circuit technology, there is an increasing importance in testing bridging (short circuit) failures in digital networks. Unfortunately, very little work has been done in this area. This paper presents the schemes for the detection of feedback bridgings between the inputs and outputs through the observation of oscillation and asynchronous behavior of sequential networks created by bridging faults. The lower and upper bounds on the number of tests for detecting all feedback bridging faults are given. Conditions for the undetectability of input bridgings are given and a method for testing input bridgings is presented. The results are generalized to detect bridging and stuck-at faults in the input and output lines of a multiple-output network. Finally, the complete test sets are given for detecting input, output and feedback bridgings as well as stuck-at faults at the input and output pins of the standard integrated circuit chips including shift registers, counters, decoders, multiplexers, adders/subtracters, multipliers, dividers and RAM. Future unsolved problems in this area are also given.

I. INTRODUCTION

The testing of digital systems has become increasingly important in recent years. Unfortunately, almost all published research papers deal only with the stuck-at faults. A bridging (short circuit) fault is a fault in which two or more leads in the circuit are shorted (wired) together. Technology has moved into VLSI (very large scale integration) where hundreds of thousands of digital components are being fabricated into a tiny IC (integrated circuit) chip. For example, the whole CPU (central processing unit) of a computer was fabricated into a single chip called a microprocessor. Now microprocessor chips of greater complexity are being built. The trend for fabricating more and more components into an IC chip is continuing and this increases the chance of short circuit failures between components and interconnecting wires. Bridging faults may arise in various levels.

* This work is supported in part by the Division Mathematical and Computer Science of National Science Foundation under grant No. MCS78-24323.

1. Inside the chip, they may arise due to either the failure of insulation between adjacent layers of metalization or the bridging of two conductors in the same layer as a result of improper masking or etching.

2. Integrated circuit packaging provides more bridging faults due to the manual labor involved in bonding between the tiny solder pads on the chip and the pins of package. Two neighboring wires may come into contact when one shakes loose between pads.

3. At the circuit board level, shorts may be caused by human error, defective printed circuit traces and feedthroughs, loose, or excess bare wires or solder bridges.

4. Interboard communication is normally done by cabling, backplane wire-wrapping, or printed circuit motherboards. Solder bridges, excess bare wires, defective traces, etc., remain to be the major causes of bridging faults and so do the easily bent wire-wrap pins. Also, the insulation of cable wires may crack in severe conditions and result in bridging faults.

Compared to the stuck-at faults, bridging faults pose more difficulties. These are given below.

1. Bridging faults may change a non-redundant network to a redundant one, which may in turn invalidate a stuck-at fault detection test set. This, therefore, poses a practical problem.

2. Bridging faults may introduce a feedback loop in a combinational logic circuit, causing the circuit to oscillate or to behave as an asynchronous sequential circuit. This violates the usual assumption that a fault does not change a combinational circuit into a sequential circuit and complicates the analysis of the fault behavior.

3. In the case of stuck-at faults, if there are n lines in the circuit, then there are 2n possible single stuck-at faults, and 3^n-1 possible multiple stuck-at faults. Whereas in the case of bridging faults, if we have to consider b dging faults between any s lines in a circuit, then the number of single bridging faults alone will be $\binom{n}{s}$ and the number of multiple bridging faults will be very much larger [5].

In the very recent paper by these authors [19], conditions for feedback bridging (short circuit) faults to generate oscillation and asynchronous behavior are given for short circuits among input lines and the primary output. The lower and upper bounds on the number of tests for detecting all feedback bridging faults are given. Necessary and sufficient conditions for the undetectability of input bridgings are presented. It is found that any test detecting single bridging fault e will also detect all multiple bridgings containing e. Complete test sets for locating either all input or all feedback bridgings of any multiplicities for networks implementing several classes of functions are given.

In this paper, we shall concentrate on the testing of bridging and stuck-at faults among the input and output pins of standard digital components such as counters, registers, decoders, multiplexers, demultiplexers, adders/subtracters, multipliers, dividers, RAM etc. Such tests are important for the IC (integrated circuit) chip users who use the IC's to design a digital system. Once the digital equipment is delivered to the field, faults (bridging and stuck-at) often occur in the input and output pins of IC chips. Detecting such types of faults is very important for maintenance testing (field testing). We shall use the methods described in [19] to generate tests for these components. These tests detect all single faults and about 95% of multiple faults. An AND-type (OR-type) bridging is defined to be the bridging fault which causes two or more wires to form the AND-(OR) operation. In this paper, only AND-type bridgings will be tested. The OR-type bridgings can be tested in a similar fashion by trivial modification of the methods given in this paper. The following five types of AND (OR) bridgings will be considered:

(1) bridgings among input lines
(2) bridgings among output lines
(3) feedback bridgings between input and output lines
(4) stuck-at faults at input lines
(5) stuck-at faults at output lines

For components with memory (registers, counters, RAM, etc.), we assume that their contents are directly readable and writable through the I/O pins of the IC chips.

Section II of this paper reviews the earlier work in bridging faults. Sections III and IV present theorems with practical examples on the detection of feedback and input bridgings respectively. Section V outlines the generalization of fault detection technique to handle multiple-output networks. In Section VI, we apply the research results shown in Sections III to V to the generation of complete test set for simultaneously detecting the aforementioned five types of faults in standard digital components. The final section outlines the future work on this topic.

II. REVIEWS OF EARLIER WORK

There are very few papers in the area of bridging faults. This is partly because the research work on this topic started only recently and the treatment of bridging faults is much more complex than the treatment of stuck-at faults. Two types of bridging faults are considered in the literature—namely, the AND type and OR type bridging faults. The AND and OR types of bridging faults mean that two or more lines are short-circuited to form AND and OR logical operations. The available techniques for bridging fault detection have been approached through the existing procedures for testing stuck-at faults since a great deal of work has already been reported on stuck-at faults.

The well-known fundamental paper on D-algorithm by Roth [1] was the first to treat bridging faults. In his paper, Roth introduced a D-notation to express a fault (stuck-at, bridging or other types of fault) and presented an algorithm to generate a test for the fault. Repeated applications of the algorithm allows one to find a complete set of tests for detecting any fault in the given combinational logic network [15]. The algorithm is suitable for computer execution and has widely been used. In a short note [2], Roth mentioned the detection of feedback bridgings for the odd number of inversions in the feedback loop. D-algorithm is general since it is independent of the types of faults. If one is interested in only bridging faults, we believe that algorithms which uses smaller memory space and shorter computing time can be developed and should be investigated.

Chang [3] proposed a fault model for short input diode fault and describes a two-pass evaluation for fault simulation. He also discusses the induced oscillations caused by backward propagation of a shorted-input diode fault and the design method to eliminate these oscillations.

Friedman [4] has shown that in a fanout-free network, a single stuck-at fault test set will also detect all bridging faults. Furthermore, it has been established that there exists a test for some stuck-at faults which can also detect an AND (OR) type bridging fault at the two input leads of an OR (AND) gate, provided that at least one of the inputs does not have any fanout. However, it is still not known whether a bridging fault between two input leads with fanouts is detectable by a single fault detection test set or even detectable at all. Also, the possibility exists for such an undetectable fault to invalidate a single fault detection test set. Friedman [4] has modified single fault detection test sets for detecting all detectable bridging faults at the inputs of a gate.

Mei [5] considered feedback bridging faults in fanout-free combinational networks using NAND gates. He showed that single stuck-at fault detection test set is also capable of detecting a feedback bridging fault if there is an odd number of inverters in the path between the two points that are bridged. If there is an even number of inverters between the two bridging points, then the detection is relatively more difficult. These faults are referred to as non-inverting feedback bridging faults (NFBF). He showed that the detection of these faults would require a certain amount of ordering of the test vectors.

Pradhan and Kodandapani [6] gave an algebraic equation to check whether a non-feedback bridging fault of two lines was detectable. (We shall generalize their results for the case of bridging of an arbitrary number of lines (Theorem 4)). They also prove that in a nonredundant two-level AND-OR (OR-AND) network, all intragate bridging faults are detectable.

Iosupovicz [7] proved that for two-level AND-OR unate function, a minimal fault detection set for single stuck-at faults could be found which would also detect all single bridging faults. He also gives a procedure, called maximum desensitization procedure, for finding that minimal fault test set.

Danilov, Karpovsky and Moskalev [8] considered the problem of detection and location of short circuits and open circuits of an arbitrary multiplicity in non-oriented graphs and contact networks. The exact bounds for the minimal number of tests for both detection and location were given. In [9] they applied the method developed for the contact networks to the detection of stuck-at errors in an arbitrary combinational network with many outputs.

In the paper by Lin and Su [10], single feedback bridging faults between two lines in general combinational networks were examined. Sufficient conditions independent of circuit implementations were derived to determine the effects of feedback bridging faults which either caused a logic network to oscillate or converted it to a sequential network. Test patterns for detecting these faults were generated and dominance relations between feedback bridging faults were studied.

III. DETECTION OF FEEDBACK BRIDGING FAULTS

In this section, first we shall present our results on the detection of feedback bridging (short circuit) faultsbetween the primary output and the primary inputs in a single-output logic network. The results on the detection of bridging faults among the primary inputs shall be given in the next section.

Instead of considering the bridgings between two lines, we shall consider the general case of the bridging among the primary output and s primary input lines, called feedback bridging of multiplicity s. Similarly, the input bridging among s input lines is called input bridging of the multiplicity s. Without loss of generality, we assume that for a network implementing function $F(x_1, x_2, \ldots, x_n)$, if the s input lines which are bridged together (either with the primary output or among themselves only) are known, then these lines are $x_1, x_2, \ldots, x_s$. $(Y\, x_1x_2\ldots x_s)$ and $(x_1x_2\ldots x_s)$ denote feedback and input bridgings of multiplicity s, respectively.

Let us consider a combinational network implementing $F(x_1,x_2,\ldots,x_n)$. If the AND-type bridging fault exists between the primary output and s input lines $x_1,x_2,\ldots x_s$ then the faulty primary output Y_e is equal to the AND function of the original output of the network and $x_1,x_2,\ldots,x_s$.

Each one of the first s primary inputs becomes $Yx_1x_2\ldots x_s$. This can be represented by the model shown in Fig. 1. Such a model will be used throughout the paper for feedback bridging faults.

The following definitions can be found in [8].
Definition 1. A circuit oscillates under certain input combination (pattern), if the output of the circuit at the next instant is the complement of current output, i.e., $Y^i = \bar{Y}^{i-1}$ where Y^i is the output at time i.

Definition 2. A circuit has asynchronous behavior under certain input combination if the circuit is stable and the present output is a function of its previous inputs and $Y^i = Y^{i-1}$.

The proofs and the examples for the following theorems can be found in the very recent paper by these authors [19].

Theorem 1 Under feedback bridging $(Y\, x_1x_2\ldots x_s)$ any network N implementing $F(x_1,x_2,\ldots,x_n)$ oscillates if the binary input n-tuple $(x_1,\ldots,x_n)$ satisfied the following condition:

$$x_1x_2\ldots x_s F(0,0,\ldots,0,x_{s+1},\ldots,x_n) \bar{F}(1,1,\ldots,1,x_{s+1},\ldots,x_n)=1. \quad (1)$$

N will have the asynchronous behavior if

$$x_1x_2\ldots x_s \bar{F}(0,0,\ldots,0,x_{s+1},\ldots,x_n) F(1,1,\ldots,1,x_{s+1},\ldots,x_n)=1. \quad (2)$$

Corollary 1. Under feedback bridging $(Y\, x_1)$, any network N implementing $F(x_1,x_2,\ldots,x_n)$ oscillates if the binary input n-tuple $(x_1,x_2,\ldots,x_n)$ satisfies

$$x_1\bar{F}(1,x_2,\ldots,x_n)F(0,x_2,\ldots,x_n)=1 \quad ; \quad (3)$$

N has asynchronous behavior if

$$x_1F(1,x_2,\ldots,x_n)\bar{F}(0,x_2,\ldots,x_n)=1 \quad . \quad (4)$$

Note that for the network to oscillate, the total number of inversions in the feedback loop must be odd. If the number of inversions in the feedback loop is even, the bridging faults in the network can be detected by utilizing the asynchronous behavior property. If Eq.(2) is satisfied, then

$$F(0,0,\ldots,0,x_{s+1},\ldots,x_n)=0 \quad ,$$

which means that the circuit output can be reset to 0 as long as the first s input variables are 0's. After reseting the output to 0, if we apply an input pattern v such that F(v)=1, then from the model shown in Fig. 1, the output response to v is 1 for the fault-free network and 0 for the network with feedback bridging of a multiplicity s.

From Theorem 1, if there exists $x_{s+1},\ldots,x_n \varepsilon \{0,1\}$ such that $F(0,0,\ldots,0,x_{s+1},\ldots,x_n) \neq F(1,1,\ldots,1,x_{s+1},\ldots,x_n)$, then either Eq. (1) or Eq. (2) is satisfied for an input pattern with the first s variables equal to 1's. Therefore, the bridging $(Y\, x_1x_2\ldots x_s)$ can be detected by observing the oscillation or asynchronous behavior of the faulty network.

It is interesting to note that sometimes the same test pattern can detect oscillation for the bridging of the given multiplicity s but cannot detect bridgings of multiplicity less than s. Formally speaking, if test pattern t satisfies Eq. (1) (or (2)) for the bridging$(Y\, x_1\ldots x_s)$, then t not necessarily satisfies Eq. (1) (or (2)) for a bridging $(Y\, x_1\ldots x_{s-q})$ $(q=1,\ldots,s-1)$ [19].

Theorem 1 is devoted to the conditions of the oscillation or of the asynchronous behavior and to the detection of the given feedback bridgings. The following theorem will be devoted to the case when we don't know which input lines are bridged and only the multiplicity s of a fault is given. Let $|x|$ denote the number of 1's in the binary n-tuple $x=(x_1,\ldots,x_n)$, then

$$|x| = \sum_{i=1}^{n} x_i \text{ and } F(|x|=i)=F(x)\big|_{|x|=i} \quad . \quad (5)$$

We note that, as it follows from the model of a faulty network (see Fig. 1), if t is a single test pattern generating the oscillation for all possible bridgings of the given multiplicity s $(1\leq s\leq n)$, then $t=\underline{1}=(1,\ldots,1)$.

Theorem 2 (i) The single test vector $\underline{1}=(1,1,\ldots,1)$ detects all possible feedback bridgings of the given multiplicity s by oscillation in a network realizing $F(x_1,x_2,\ldots,x_n)$ if

$$F(|x|=n)=0,$$
$$F(|x|=n-s)=1. \quad (6)$$

(ii) The test sequence $(R,\underline{1})$ detects all possible feedback bridgings of the given multiplicity by asynchronous behavior in a network realizing $F(x_1,x_2,\ldots,x_n)$ if

$$F(|x|=n)=1$$
$$F(|x|=n-s)=0, \quad (7)$$

where R is the input pattern which resets the output to 0.

Example 1. (i) For the parity checker $F = \bigoplus_{i=1}^{n} x_i$ and n=even, $\underline{1}=(1,1,\dots,1)$ generates the oscillation for any feedback bridging of odd multiplicity.

(ii) For a K-bit parallel adder $F=X+Y$ $0\le X,Y<2^K$

$X = \sum_{i=0}^{K-1} x_{i+1}2^i$, $Y = \sum_{i=0}^{K-1} y_{i+1}2^i$, $F = \sum_{i=0}^{K} F_{i+1}(x_1,\dots,x_K,y_1,\dots,y_k)2^i$, $F_{i+1}(0,\dots,0,0,\dots,0)=0$

where $i=0,1,\dots,K$, and $F_{i+1}(1,\dots,1,1,\dots,1)=1$ where $i=1,2,\dots,K$; thus, by Theorem 2 the test sequence $(\underline{0},\underline{1})$ detects all possible feedback bridgings between primary inputs and primary outputs $F_{i+1}(i=1,2,\dots,K)$, since for a faulty network there exists at least one 0 in the output vector $(F_2,\dots,F_{K+1})$. See Fig. 1.

Theorem 2 deals with the case when the multiplicity of bridgings is known. The following result is for the general case where the multiplicity of the bridgings is unknown.

Theorem 3. Let $N_{d,f}(n)$ be the minimum number of tests for detecting feedback bridgings of any multiplicity in any network implementing a function F of n variables (the subscripts d and f denote "detection" and "feedback" respectively). Then

$$1\le N_{d,f}(n)\le n \quad . \qquad (8)$$

Since 50% of the function has the property $F(\underline{0})=1$, feedback bridgings in networks implementing half of the functions can be detected by applying only one test pattern $\underline{0}$.

Example 2. For a K-bit comparator $F(X,Y)=F(x_1,\dots,x_K,y_1,\dots,y_K)=1$ iff $X=Y$ $(0\le X,Y<2^K$, $X= \sum_{i=0}^{K-1} x_{i+1}2^i$, $Y= \sum_{i=0}^{K-1} y_{i+1}2^i)$ and since $F(0,\dots,0,0,\dots,0)=1$, the test pattern $\underline{0}$ detects all possible feedback bridgings (the output will be 0 if there is any feedback fault.) See Fig. 1.

Let t be a test pattern (input binary n-tuple) and $\bar{t}$ be its complement, and $F(t)=F(\bar{t})=1$. Then in a network realizing F any feedback bridging of any multiplicity can be detected by applying just two test patterns: t and $\bar{t}$.

Example 3. For a K-bit adder (see Example 1(ii)) $F=X+Y$. $X=(x_1,\dots,x_K)$, $Y=(y_1,\dots,y_K)$, $F=(F_1,\dots,F_{K+1})$ and two input patterns $X=\underline{0},Y=\underline{1}$ and $X=\underline{1},Y=\underline{0}$ detect any feedback bridging between primary inputs and $F_1,\dots,F_K$.

Theorem 4. For function $F(x_1,x_2,\dots,x_n)$, any sequence of input patterns $(t^1,t^2,\dots,t^N)$, such that $F(t^i)=1$ $(i=1,2,\dots,N)$ and $\prod_{i=1}^{N} t^i=\underline{0}$ ($\prod_{i=1}^{N} t^i$ is the component-wise multiplication of vectors t^i) is the test sequence for detecting all possible feedback bridgings in any network implementing $F(x_1,x_2,\dots,x_n)$.

Example 4. For a K-to-1 multiplexer with K data inputs $(x_1,x_2,\dots,x_K)$ and $\alpha=\log_2 K$ select lines $S_1,\dots,S_\alpha$ we have $F(x_1,\dots,x_K,S_1,\dots,S_\alpha)=x_{i+1}$ where $i=\sum_{j=0}^{\alpha-1} S_{j+1}2^j$ and $K=2^\alpha$. Choose $t^1=(\overbrace{1,0,0,\dots,0}^{},\overbrace{0,0,\dots,0}^{})$ and $t^2=(\overbrace{0,1,0,\dots,0}^{K},\overbrace{1,0,\dots,0}^{\alpha})$. Then $F(t^1)=F(t^2)=1$, $t^1\cdot t^2=\underline{0}$ and by Theorem 4, the test sequence (t^1,t^2) detects all possible feedback bridgings in a K-to-1 multiplexer since t^1 (t^2) detects feedback bridgings involving all x_i's except $x_1(x_2)$.

IV. DETECTION OF INPUT BRIDGINGS

The logical model for the AND-type bridging of multiplicity s among lines $x_1,x_2,\dots,x_s$ in a network implementing $F(x_1,x_2,\dots,x_n)$ is given in Fig. 2. The next theorem presents the necessary and sufficient conditions for undetectability of an input bridging of any given multiplicity.

Theorem 5. For any network implementing $F(x_1,x_2,\dots,x_n)$, the bridging between input lines $x_1,x_2,\dots,x_s$ is undectable if and only if for every $x_{s+1},x_{s+2},\dots,x_n\in\{0,1\}$ and every $A=(a_1,a_2,\dots,a_s)\ne\underline{0},\underline{1}$ where $\underline{0}=(\underbrace{0,0,\dots,0}_{s})$ and $\underline{1}=(\underbrace{1,1,\dots,1}_{s})$,

$$F(a_1,a_2,\dots,a_s,x_{s+1},\dots,x_n) = F(0,0,\dots,0,x_{s+1},\dots,x_n). \qquad (9)$$

Example 5. $F(x_1,x_2,x_3)=\bar{x}_1(\bar{x}_2 v \bar{x}_3) v\ x_1x_2x_3$. (Symbol "v" stands for logical OR). The bridging between x_2 and x_3 is undectable since $F(x_1,0,1)=F(x_1,1,0)=F(x_1,0,0) = \bar{x}_1$ and Eq. (9) is satisfied. However, since $F(0,0,x_3)=1$ and $F(0,1,x_3)=\bar{x}_3$, the bridging between x_1 and x_2 is detected by the test (0,1,1).

Corollary 2. An input pattern $t=(t_1,t_2,\dots,t_n)$ detects the bridging $(x_1x_2\dots x_s)$ if, and only if, $(t_1,t_2,\dots,t_s)\ne\underline{0},\underline{1}$ and

$$F(0,0,\dots,0,t_{s+1},\dots,t_n)\ne F(t_1,t_2,\dots,t_n). \qquad (10)$$

Example 6. For the detection of all input bridgings in a 8-bit parallel adder (see Example 1 (ii)) it is sufficient to apply the four test patterns t^1,t^2,t^3,t^4 $(t^i=(x_1,\dots,x_8,y_1,\dots,y_8)$ where $i=1,2,3,4)$ which are the rows of the following matrix(T).

$$(T)=\left[\begin{array}{cccc} 0000 & 0000 & 1111 & 1111 \\ 0000 & 1111 & 0000 & 1111 \\ 0011 & 0011 & 0011 & 0011 \\ 0101 & 0101 & 0101 & 0101 \end{array}\right]\begin{array}{c} t^1 \\ t^2 \\ t^3 \\ t^4 \end{array} ;$$

For this test and any input bridging there exists an output $F_i(x_1,\dots,x_8,y_1,\dots,y_8)$ such that equation (10) is satisfied for at least one test pattern. In general if the number of bits for an adder is equal to $K=2^\alpha$, then $\log_2 2K$ test patterns such that all columns of the corresponding matrix(T)are different, will detect all possible input bridgings in an adder, and this test set is minimal.

It is well known that two stuck-at faults may compensate each other [9]. It is interesting to see that the situation is different for bridging faults. The next theorem states that two AND-type (or OR-type) input bridgings can not compensate each other.

Theorem 6. For a network implementing $F(x_1,x_2,\dots,x_n)$, if there does not exist any test for detecting the double input bridgings $e_1=(x_1x_2\dots x_s)$ and $e_2=(x_{s+1}\dots x_{s+q})$, $s+q\le n$, then no test pattern can detect either bridging.

Note that the generalizations of the results for locating feedback and input bridgings are also given in [19].

V. FAULT DETECTION IN MULTIPLE-OUTPUT NETWORKS

Our previous results were mostly devoted to the case of networks with single output. In this section, we consider the problem of test generation for multiple-output networks. The problem of the detection of input, output and feedback bridging and stuck-at faults in these networks is very important for the maintenance testing of remote networks, e.g. blocks of an on-board computer.

Let us consider a combinational network with n inputs and m outputs implementing the following system of m functions

$$Y_i = F_i(x_1,\dots,x_n) \text{ where } i=1,\dots,m \tag{11}$$

We shall consider the following five classes of faults: (1) Feedback bridgings; (2) Input bridgings; (3) Output bridgings; (4) Input stuck-at faults; (5) Output stuck-at faults.

5.1 Feedback Bridgings

First let us consider feedback bridgings. If $F_i(\underline{0})=1$ (where $i=1,\dots,m$; $\underline{0}=(0,\dots,0)$), then input vector $\underline{0}$ detects all feedback bridgings (since output vector for a faulty network contains zeros). Similarly, if there exists $t=(t_1,\dots,t_n)$ where $t_j \in \{0,1\}$ such that $F_i(t)=F_i(\bar{t})=1$ where $i=1,\dots,m$, then two input vectors $t,\bar{t}$ detect all feedback bridgings. In general, any sequence $(t^1,t^2,\dots,t^N)$ of input patterns such that $F_i(t^j)=1$ where $i=1,2,\dots,m$ and $j=1,\dots,N$ and $\prod_{j=1}^{N} t^j=\underline{0}$ is the test sequence detecting all feedback bridgings.

Another approach for generating tests for feedback bridgings is based on the observation of the oscillation or asynchronous behavior. In this case, we may use the corresponding results from Section III.

Let T_i be a minimal test for a part of the network implementing $F_i(x_1,\dots,x_n)$. Then we always can use the set $T=\bigcup_{i=1}^{m} T_i$ as a test set for the detection of all feedback bridgings. Thus from Theorem 3, we have the following bounds for the minimal number $N_{d,f}(n,m)$ of tests for detecting all feedback bridgings in any network with n inputs and m outputs.

$$1 \le N_{d,f}(n,m) \le \left|\bigcup_{i=1}^{m} T_i\right| \le n\, m \tag{12}$$

For an IC chip with 10-input, 10-output variables, in the worst case, it takes only 100 test patterns to detect the feedback bridgings. We suspect that the upper bound smaller than nm can be obtained and we plan to look into this.

5.2 Input Bridgings

Let us consider now input bridgings. First we generalize the conditions of undetectability of input bridgings to the case of multiple-output. The bridging $(x_1 \dots x_s)$ is undetectable iff

$$F_i(\alpha_1,\dots,\alpha_s,x_{s+1},\dots,x_n) = F_i(0,\dots,0,x_{s+1},\dots,x_n) \tag{13}$$

for all $\alpha=(\alpha_1,\dots,\alpha_s)$, $\alpha \ne \underline{0},\underline{1}$;

$x_{s+1},\dots,x_n \in \{0,1\}$, $i=1,\dots,m$.

We note that there exist systems $F(x)=\{F_i(x)\}$, $i=1,\dots,m$ such that, each one of the functions of the system cannot detect all input bridgings of the given multiplicity, but the whole system detects them. In other words, there exists a multiple-output network such that not all input bridgings can be detected by observing only one output but all input bridgings can be detected by observing all outputs.

Example 7 . Let us consider single input bridgings in a network computing F_1^1, F_2^1 (see Table 1, here n=3, m=2).

x_1	x_2	x_3	F_1^1	F_2^1	F_1^2	F_2^2	F_3^2	F_4^2	F_1^3	F_2^3	F_3^3	F_4^3
0	0	0	0	0	0	1	0	1	0	0	0	1
0	0	1	1	0	0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1	1	0	0	0
1	0	0	0	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1

Table 1

It follows from (13) that the function F_1^1 cannot detect bridging (x_1x_2) and the function F_2^1 cannot detect (x_2x_3), but the whole system detects all single input bridging, e.g., by the test set $T=\{(0\underline{1}\underline{1}),(0\underline{0}\underline{1})\}$ since $(0\underline{1}\underline{1})$ test the bridging between x_1 and x_2 or x_3 and $(0\underline{0}\underline{1})$ test bridging between x_3 and x_1 or x_2.

If $T=(t^1,\dots,t^N)$ is a test sequence for detecting all input bridgings then in the $N \times n$ binary input matrix T with rows $t^1,t^2,\dots,t^N$, all columns must be different.

We note also that, if T_i is a minimal test set for detecting input bridgings by observing output F_i only, then we always can use T_i as a test sequence for a whole network. Thus, if $N_{Inp}(n,m)$ is the minimal number of tests, for input bridgings, then we have the $\log_2 n \le N_{Inp}(n,m) \le \min_i |T_i|$.

Since for a m-output network, there are m instead of one observation points, the actual number of tests will be smaller than the smallest $|T_i|$.

5.3 Output Bridgings

For output bridgings we note first that all these bridgings are detectable iff our network is irredundant (nonredundant). Indeed, t detects output bridging (F_iF_j) if $F_i(t) \ne F_j(t)$ (since for a network with an AND-bridging, $F_i(t)=F_j(t)=0$).

We note that $T=(t^1,\dots,t^N)$ is a test sequence for all output bridgings, if and only if in the $(m \times N)$ binary output matrix

$$F(T)=\begin{bmatrix} F_1(t^1)\dots F_m(t^1) \\ \vdots \qquad \vdots \\ F_m(t^N)\dots F_m(t^N)\end{bmatrix}$$ all columns are different, so we have the following bounds for the minimal number of tests $N_{out}(m)$ for detecting output bridgings

$$\log_2 m \le N_{out}(m) \le m-1. \qquad (15)$$

Example 8. In Table 1 lower bound is reached for F_1^2, F_2^2, F_3^2, F_4^2 with $T = \{(0,0,0),(0,0,1)\}$ and upper bound is reached for F_1^3, F_2^3, F_3^3, F_4^3 with $T = \{(0,0,0),(0,0,1),(0,1,0)\}$.

It follows from the bounds (12), (14), (15) that with the increase of the number of outputs, m, the minimal number of tests increase linearly for the feedback and output bridgings and decreases for input bridgings, but testing practical digital components may be carried out in a reasonably short time. See the next section.

5.4 Input and Output Stuck-at Faults

The input stuck-at faults can be detected by a test sequence $T = (t^1,\dots,t^N)$ where $t^i=(t_1^i,\dots,t_n^i)$ and $t_j^i \in \{0,1\}$ such that in matrix(T) with rows $t^1,\dots,t^N$, all columns are neither $\underline{0}$ nor $\underline{1}$. Furthermore, for any $j\in\{1,\dots,n\}$ there exists $i\in\{1,\dots,N\}$ and $p \in\{1,\dots,m\}$ such that $F_p(t_1^i,\dots,t_{j-1}^i,0,t_{j+1}^i,\dots,t_n^i) \ne F_p(t_1^i,\dots,t_{j-1}^i,1,t_{m+1}^i,\dots,t_n^i)$. (16)

Note that for many standard digital components the last condition is satisfied for all $(t_1^i,\dots,t_{j-1}^i,t_{j+1}^i,\dots,t_n^i)$ (see Section VI).

For the detection of output stuck-at faults it is necessary and sufficient that the output matrix

$$F(T)=\begin{bmatrix} F_1(t^1) \dots F_m(t^1) \\ \vdots \qquad \vdots \\ F_1(t^N) \dots F_m(t^N)\end{bmatrix}$$ does not contain columns $\underline{0}$ and $\underline{1}$. Therefore, for a minimal number of tests, $N(n,m)$, for detecting bridging and stuck-at faults in any networks with $N=2^p$ inputs and $m=2^q$ outputs.

$$N(n,m) \ge \max(p,q)+1 \qquad (17)$$

Example 9. Let us construct the test for detecting all feedback, input and output bridgings and stuck-at faults in a 7-bit algebraic array multiplier $F=XY$, $0\le X=\sum_{i=1}^{7} x_i 2^{7-i}$, $Y=\sum_{i=1}^{7} y_i 2^{7-i} <2^7$ (see item No. 2 in Table 2). Using the previous results it is easy to check that test patterns $t^1,t^2,\dots,t^8$ (t^i=(sign X, $x_1,\dots,x_7$, Sign Y, $y_1,\dots,y_7$)) expressed by the rows of the following input matrix will detect all bridging and stuck-at faults.

$$(T)=\begin{bmatrix}
1111 & 1111 & 0000 & 0000 \\
1111 & 1111 & 1111 & 1111 \\
0000 & 1111 & 0000 & 0001 \\
0011 & 0011 & 0000 & 0001 \\
0101 & 0101 & 0000 & 0001 \\
1100 & 0000 & 0000 & 1111 \\
1100 & 0000 & 0011 & 0011 \\
1100 & 0000 & 0101 & 0101
\end{bmatrix}.$$

For the corresponding output matrix F(T) we have

$$F(T)=\begin{bmatrix}
0000 & 0000 & 0000 & 0000 \\
0111 & 1111 & 0000 & 0001 \\
0000 & 0000 & 0000 & 1111 \\
0000 & 0000 & 0011 & 0011 \\
0000 & 0000 & 0101 & 0101 \\
1000 & 0111 & 1000 & 0000 \\
1001 & 1001 & 1000 & 0000 \\
1010 & 1010 & 1000 & 0000
\end{bmatrix},$$

If $F(t^i)=Z$ then $Z=(\text{Sign}Z,Z_1,\dots,Z_{14})$, $|F(t^i)|=\sum_{i=1}^{14} Z_i 2^{14-i}$.

The general case of K-bit algebraic multiplier is considered in Table 2.

Note also that the approach to the test generation described above for combinational networks may also be applied for networks with memory if storage elements are directly readable and writable from primary inputs and outputs, which is the standard requirement in the design for testability. Several examples of test sequences for the devices with memory (registers, counters, RAM) are considered in Table 2.

VI. APPLICATIONS: TESTING STANDARD COMPONENTS

In this section we shall apply our previous results to the simultaneous detection of bridging and stuck-at faults at primary inputs and outputs. The testing of faults in input/output pins is an important part of field testing. Only AND-type bridgings will be considered (but the results may easily be modified to the case of OR-type bridgings). The following five types of faults will be detected: (1) stuck-at faults at input lines; (2) stuck-at faults at output lines; (3) bridgings among input lines; (4) bridgings among output lines; (5) feedback bridgings. Detection of feedback bridgings will be based on observation of the asynchronous behaviour (model in Fig. 1) and Theorem 4.

Table 2 contains tests for detecting all single stuck-at and bridging faults for shift registers (shifter) and counters with parallel load , decoders, algebraic adders, multiplexers, dividers and RAM. These tests detect all single faults and about 95% of multiple faults.

Table 2 also contains N, the number of test patterns as a function of I, the number of input pins for each component. (Input pins for power supply, clock and enable are not included). Numbers in arithmetical devices (adders, multipliers, dividers) are represented in the form of "sign and magnitude", i.e. for K-bit device $X=(SX,x_1,\dots,x_{K-1})$ where SX is the sign bit for X (SX=0 if $X\ge0$, SX=1 if X<0) and

$$X = \sum_{i=1}^{K-1} x_i 2^{K-1-i}$$ is the decimal equivalent of the binary number. For devices with memory (registers, counters, RAM) we assume that all flip-flops are directly writable and readable from the corresponding input and output pins.

Table 2 also contains examples of test sequences for K-bit arithmetical and logical devices with a small K. The following notations are used in Table 2.

1) I is a number of input pins.
2) N is a number of tests for detecting any single fault in the input and output pins of a component.
3) For a test sequence $T = (t^1, t^2, \ldots, t^N)$ where $t^i = (t_1^i, \ldots, t_I^i)$, $t_1^i \in \{0,1\}$, (T) is the $(N \times I)$ binary input matrix with the rows $t^1, t^2, \ldots, t^N$.
4) For a device with I input lines and R output lines implementing the system of Boolean functions, $Z_i = F_i(x_1, x_2, \ldots, x_I)$ where $i = 1, 2, \ldots, R$, we denote F(T) the $(N \times R)$ binary output matrix with the element in the ith rows and the jth column denotes the jth output response to the ith test.
5) M_α is binary $(\alpha \times 2^\alpha)$-matrix containing all possible binary vectors with α components per column; M_α^T is the transposed matrix for M_α; M_α-$\underline{1}$ (M_α-$\underline{0}$) is the matrix M_α without the all-one (all-zero) column; $\overline{M_\alpha\text{-}\underline{1}}$ is the matrix obtained by the negation of all entries in M_α-$\underline{1}$; E is the identity matrix.
6) Let A and B be two (pxq)-binary matrices with rows $A_1, \ldots, A_p$ and $B_1, \ldots, B_p$; the (2pxq)-matrix with rows $A_1, B_1, A_2, B_2, \ldots, A_p, B_p$ is denoted by $A \circ B$. (Note $A \circ B \neq B \circ A$)

We now outline the approach we took for constructing Table 2 and explain why the tests (shown in the Table) detect the aforementioned five types of faults.

First, let us consider the counter (device No. 2). The first rows in matrices (T) and F(T) in Table 2 show that when counter control C=1, the counter counts up and its content changes from (1,1,1,1,1,1) to (0,0,0,0,0,0). When C=0 the content of the counter remains unchanged. Matrix (T) is selected in such a way that all columns of (T) are distinct and not equal $\underline{0}$ or $\underline{1}$. (Since if column x_i contains all 0's (1's) then x_i s-a-1 (s-a-0) cannot be detected.) Therefore, any bridging fault between any two inputs will change at least one column and hence change the value(s) of output response for at least one test pattern. For instance, if there is an AND-bridging fault between C and x_1 both the C column and x_1 column will become (1,0,0,0) which changes the last row of F(T) from (1,0,1,0,1,0) to (0,0,1,0,1,0) and thus the fault is detected. A similar principal is applied for detecting output bridging faults. For example, if there is an AND-bridging between z_1 and z_6, then the first and the last column of F(T) will change to all 0's. The first test pattern will detect any feedback bridging (of any multiplicity) between any otuput and any input (including input signal C) because any feedback bridging between x_i and z_j where $i,j = 1,2,\ldots,6$ will change x_i from 1 to 0 (see the model in Fig. 1) and hence change the outputs. For instance, if there is a bridging between x_6 and z_1 then the first row of (T) will become (1,1,1,1,1,1,0) and thereby changes the first row of F(T) to (1,1,1,1,1,1).

For stuck-at faults, the first test will detect any s-a-0 faults at inputs or s-a-1 at outputs. The second test will detect C, x_1, x_2 or x_3 s-a-1 or z_4, z_5 and z_6 s-a-0. The last two tests detect the remaining stuck-at faults.

Based on the two matrices in this example, we develop a general form of (T) and F(T) for a counter with any number of bits. Note that in this example, the columns of the submatirx of (T) consisting of rows 2 to 4 contain all possible combinations of three variables except (1,1,1). We denote it by M_α-$\underline{1}$ in the general from for T in Table 2. Similarly, M_α-$\underline{0}$-$\underline{1}$ in F(T) denotes a matrix containing all possible combinations in its columns except $\underline{0}=(0,0,\ldots,0)$ and $\underline{1}=(1,1,\ldots,1)$. Finally, the number of tests, N, for a counter with $I = 2^\alpha - 1$ inputs (including control signal C) is $\log_2(I+1) + 1 = \alpha + 1$.

Let us now look at the adder/subtracter in Table 2. Each row of matrix (T) contains the sign SX and magnitude $(x_1, x_2, \ldots, x_7)$ of the first number X and the sign SY and the magnitude $(y_1, y_2, \ldots, y_7)$ of the second number Y. The sum X+Y is given in the corresponding row in F(T), the matrix for the sum. Again we choose (T) such that all columns are distinct and not equal $\underline{0}$ or $\underline{1}$. This yields the F(T) with distinct columns not equal $\underline{0}$ or $\underline{1}$. Hence all input and output bridgings are detected. The feedback bridging is detected by using the asynchronous behavior property of sequential circuits (Fig. 1 and Theorem 1). The output response to the first test is all 0's. Now if there is a feedback bridging between x_i and z_j, for any i and j, then the bridging will change the second test from $\underline{1} = (1,1,\ldots,1)$ to a pattern with all 1's except a 0 at the ith position. This in turn will change the sum shown in the second row of F(T) and hence the fault is detected. Finally, the first test in (T) detects any stuck-at-1 faults at any input or output. The second test detects all stuck-at-0 faults on the inputs and all stuck-at-0 faults except one (i.e. z_8) at the outputs. z_8 s-a-0 is detected by the third test. The test patterns for an adder with any number of bits are given and the total number of tests is $\log_2 I + 2$ where $\log_2 I$ is the number of rows in $M_{\alpha+1}$, as shown in Table 2. With a simple modification, Table 2 can be used for generating tests IC's with any number of inputs I.

By the same approach, the complete test set and the total number of tests for other standard digital components can be found.

The number of tests for detecting any single fault in different K-bit devices and RAM is given in Table 3.

No. of bits / Device	8	16	32	64
Shift Register	6	7	8	9
Counter	5	6	7	8
Up & Down Counter	6	7	8	9
Algebraic Adder	6	7	8	9
Multiplier	8	10	12	14
Divider	8	10	12	14
RAM	8	10	12	14

Table 3. The number of tests for K-bits devices and RAM for detecting input, output and feedback bridgings and stuck-at faults.

As we can see from Table 3, for the testing of arithmetical devices and RAM of practical size with respect to input, output and feedback bridgings and stuck-at faults, we need very few tests.

From Table 2 we can see that the complete detection set can be stored in a small PROM and these test patterns can rapidly be applied to the network-under-test to determine whether there is a fault.

VII. CONCLUSION AND FUTURE WORK

With the advances in LSI and VLSI the problems of testing birdging faults will become worse unless efficient testing strategies are developed very soon. This paper and the very recent paper by these authors [19] presents an initial step toward solving this important problem. Some theorems and methods for testing the bridging faults at the input and output pins of integrated circuit chips for some standard digital components have been presented. The complete test set for detecting bridging and stuck-at failures in these components have been generated. Research funds are being sought to continue performing research on the following topics in this important area.

1) Generalization of our results for testing basic standard digital components to the test generation for detecting input, output and feedback bridgings and stuck-at faults for ALU, PLA, CPU and some commercially available microprocessors (maintenance testing).
2) Algorithms for generating efficient tests for detecting all internal bridgings (manufacturing testing).
3) Lower and upper bounds for the minimal number of tests for detecting all internal bridgings (testing time for internal bridgings), and the mixture of bridging and stuck-at faults.
4) Techniques and algorithms for the construction of efficient test sets for detecting all bridging and stuck-at faults in multiple-output digital networks.
5) Computer implementation of algorithms developed in items 1, 2 and 4, i.e. program the algorithms for generating tests for detecting internal bridging and the mixture of stuck-at and bridging faults.

REFERENCES

[1] J.P. Roth, "Diagnosis of Automata Failures: A calculus and a method," IBM Journal of Research and Development, Vol. 10, pp. 278-291, July 1966.

[2] J.P. Roth, "Method of Testing for Shorts," IBM Technical Disclosure Bulletin, pp. 3108-3109, February 1976.

[3] H.Y. Chang, "A Method for Digitally Simulating Shorted Input Diode Failures," Bell System Technical Journal, Vol. 48, pp. 1957-1966.

[4] A.D. Friedman, "Diagnosis of Short-Circuit Faults in Combinational Circuits," IEEE Transactions on Computers, Vol. C-23, pp. 746-752, July 1974.

[5] K.C.Y. Mei, "Bridging and Stuck-at-Faults," IEEE Transactions on Computers, Vol. C-23, pp. 720-727, July 1974.

[6] K.L. Kodandapani and D.K. Pradhan, "Undetectability of Bridging Faults and Validity of Stuck-fault Test Set," IEEE Transactions on Computers, Vol. C-29, pp. 55-59, January, 1980.

[7] A. Iosupovicz, "Optimal Detection of Bridging Faults and Stuck-at Faults in Two-Level Logic," IEEE Transactions on Computers, Vol. C-27, pp. 452-455, May 1978.

[8] V.V. Danilov, M.G. Karpovsky, E.S. Moskalev, "Tests for Nonoriented Graphs," Automat. and Remote Control, Vol. 31, No. 4, pp. 656-665, April 1970, (translated from: "Automatika i Telemekhanika," No. 4, pp. 160-168, 1970, Russian).

[9] V.V. Danilov, M.G. Karpovsky, E.S. Moskalev, "Tests for Digital Devices," pp. 164-167, In Monograph: "Technical Diagnostics," Prof. Parkhomenko Ed., Moscow, 1972, (Russian).

[10] C.H. Lin and S.Y.H. Su, "Feedback Bridging Faults in General Combinational Networks," Proceedings of the 8th International Symposium on Fault-tolerant Computing, June 1978.

[11] R.P. Capece, "Tracking the Very Large-Scale Problems of VLSI: a Special Report," Electronics, Vol. 51, No. 24, pp. 111-125, November 1978.

[12] W.W. Peterson, E.J. Weldon, "Error Correcting Codes, 2nd Edition, MIT Press, Cambridge, Mass. 1972.

[13] F.W. Clegg, and E.J. McCluskey, "Algebraic Properties of Faults in Logic Networks," Techn, Rep. No. 4, SU-BSEL 69-078, Digital Systems Laboratory, Stanford University, Stanford, California.

[14] A.D.R. Schertz, "Application of a Fault Class Representation of Diagnosis," Proc. of 1973 International Symposium on Fault-Tolerant Computing, June 1973, Palo Alto, California.

[15] J.P. Roth, W.G. Bouricius and P.R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," IEEE on Computers, EC-16, pp. 567-579, October 1967.

[16] A.N. Airapetian, J.E. McDonald, "Improved Test Generation Algorithm for Combinational Circuit Control" (to appear).

[17] M.A. Breuer and A.D. Friedman, "Diagnosis and Reliable Design of Digital Systems", Computer Science Press, Inc., 1976.

[18] S.Y.H. Su, "Logic Design and Its Recent Development Part V: Fault Diagnosis in Digital Networks", Computer Design, pp. 87-92, February 1974.

[19] M. Karpovsky, S.Y.H. Su, "Detection and Location of Input and Feedback Bridging Faults in Combinational Networks", IEEE Transactions on Computers, June, 1980.

[20] S.Y.H. Su and Y.C. Cho, "A New Approach to the Fault Location of Combinational Circuits," IEEE Transactions on Computers, Vol. C-21, pp. 21-30, January 1972.

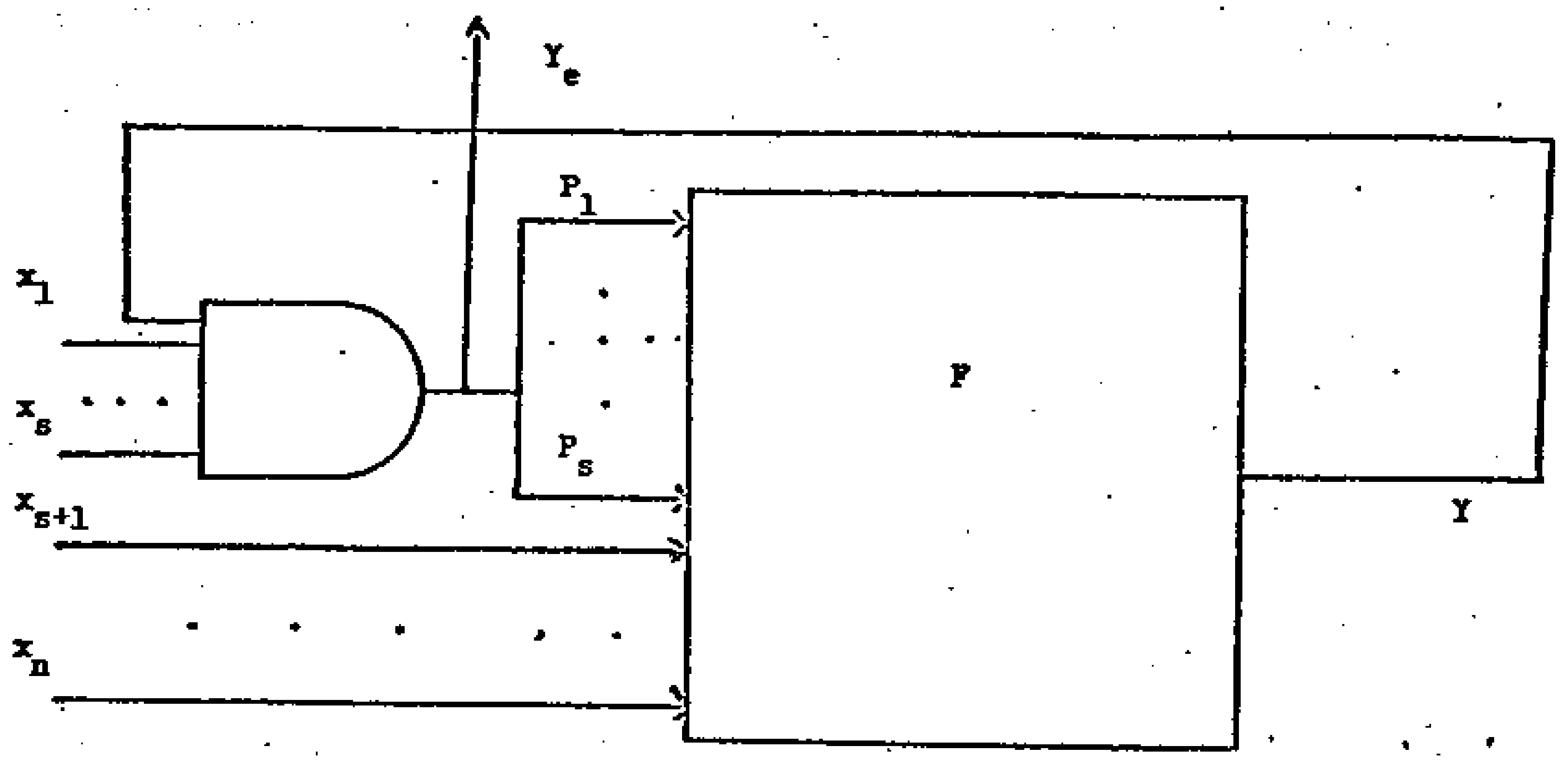

Fig. 1. Logical model of feedback bridging ($Yx_1 \ldots x_s$)

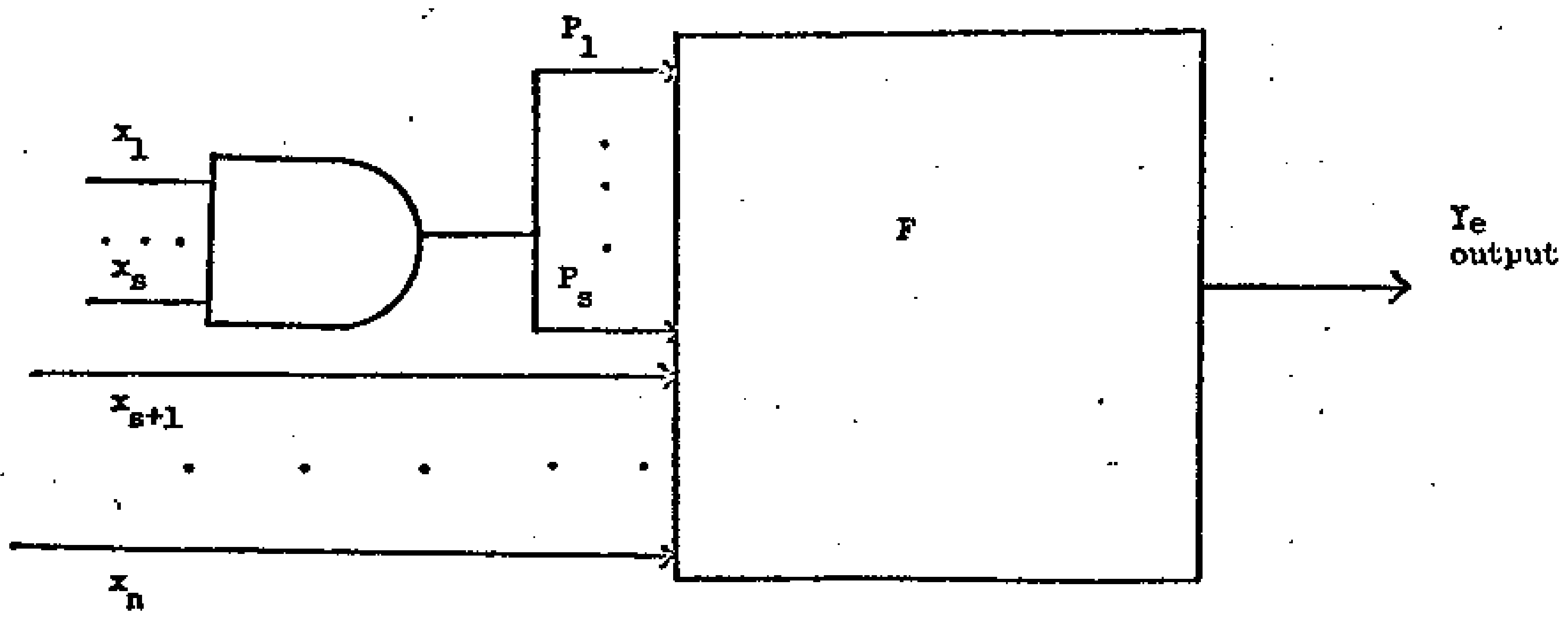

Fig. 2. Logical model of input bridging ($x_1 \ldots x_s$)

DEVICE	DEFINITION	(T)	F(T)	Number N of tests	Example (T)	Example F(T)	
NO. 1 K-bit register with parallel load or shifter	$F(Shl,Shr,x_1,x_2,\dots,x_{K-1},x_K)=$ $(x_1,x_2,\dots,x_{K-1},x_K)$ if Shl=Shr=0; $(x_2,x_3,\dots,x_K,0)$ if Shl=1, Shr=0; $(0,x_1,\dots,x_{K-2},x_{K-1})$ if Shl=0, Shr=1; $K=2^{\alpha}-2$.	$\left[\begin{array}{cc} 00 & 00\dots0 \\ 10 & 11\dots1 \\ \hline 01 & \\ 01 & \\ \vdots & M_{\alpha}-\underline{0}-\underline{1} \\ 01 & \end{array}\right]$ $\underbrace{\qquad}_{2^{\alpha}-2}$	$\left[\begin{array}{ccccc} 00 & \dots & & & 00 \\ 11 & \dots & & & 10 \\ \hline & & 1 & & 1 \\ M_{\alpha} & - & \vdots & - & \vdots \\ & & 1 & & 1 \\ & & 0 & & 1 \end{array}\right]$ $\underbrace{\qquad}_{2^{\alpha}-2}$	$\log_2 I+2$	$\begin{array}{cc} \left[\begin{array}{cc} 00 & 000000 \\ 10 & 111111 \\ \hline 01 & 000111 \\ 01 & 011001 \\ 01 & 101010 \end{array}\right] \\ Shl\ Shr & x_1\dots x_6 \end{array}$	$\begin{array}{c} \left[\begin{array}{c} 00000\ 0 \\ 11111\ 0 \\ \hline 00001\ 1 \\ 00110\ 0 \\ 01010\ 1 \end{array}\right] \\ Z_1 Z_2\dots Z_6 \end{array}$	K=6 I=8 N=5
NO. 2 K-bit counter with parallel load	$\vert Z\vert=F(C,X)=$ $C+\vert X\vert \pmod{2^K}$ $C\epsilon\{0,1\}$ $X=\sum_{i=1}^{K} x_i 2^{K-i}; K=2^{\alpha}-2$; $X=(x_1,\dots,x_K)$; $Z=(z_1,\dots,z_K)$.	$\left[\begin{array}{c} 111\dots1 \\ \hline M_{\alpha}-\underline{1} \end{array}\right]$ $\underbrace{\qquad}_{2^{\alpha}-1}$	$\left[\begin{array}{c} 000\dots0 \\ \hline M_{\alpha}-\underline{0}-\underline{1} \end{array}\right]$ $\underbrace{\qquad}_{2^{\alpha}-2}$	$\log_2(I+1)+1$	$\begin{array}{c} \left[\begin{array}{cc} 1 & 111111 \\ \hline 0 & 000111 \\ 0 & 011001 \\ 0 & 101010 \end{array}\right] \\ C\ x_1\dots x_6 \end{array}$	$\begin{array}{c} \left[\begin{array}{c} 000000 \\ \hline 000111 \\ 011001 \\ 101010 \end{array}\right] \\ Z_1\dots Z_6 \end{array}$	K=6 I=7 N=4
NO. 3 K-bit Up and down counter with parallel load	$\vert Z\vert=F(C_{up},C_{down},X)$ $=\vert X\vert+C_{up}-C_{down} \pmod{2^K}$ $C_{up},C_{down}\ \epsilon\{0,1\}$ $X=\sum_{i=1}^{K} x_i 2^{K-i}; K=2^{\alpha}-2$; $X=(x_1,\dots,x_K)$; $Z=(Z_1,\dots,Z_K)$.	$\left[\begin{array}{cc} 0 & 100\dots0 \\ 1 & 011\dots1 \\ \hline 0 & \\ 0 & M_{\alpha}-\underline{1} \\ \vdots & \\ 0 & \end{array}\right]$ $\underbrace{\qquad}_{2^{\alpha}-1}$	$\left[\begin{array}{c} 111\dots1 \\ 000\dots0 \\ \hline M_{\alpha}-\underline{0}-\underline{1} \end{array}\right]$ $\underbrace{\qquad}_{2^{\alpha}-2}$	$\log_2 I+2$	$\begin{array}{c} \left[\begin{array}{cc} 01 & 000000 \\ 10 & 111111 \\ \hline 00 & 000111 \\ 00 & 011001 \\ 00 & 101010 \end{array}\right] \\ C_{up}\ C_{down}\ x_1\dots x_6 \end{array}$	$\begin{array}{c} \left[\begin{array}{c} 111111 \\ 000000 \\ \hline 000111 \\ 011001 \\ 101010 \end{array}\right] \\ Z_1\dots Z_6 \end{array}$	K=6 I=8 N=5
NO. 4 $K\times 2^K$ Decoder-Demultiplexer	$F(Enable,x_1,\dots,x_K)=$ $(F_1,\dots,F_m)$; $F_s=1$ iff Enable = 1 and $\sum_{i=1}^{K} x_i 2^{K-i}=S-1$; $m=2^K$.	$\left[\begin{array}{cc} 0 & 00\dots0 \\ \hline 1 & \\ 1 & M_{\alpha}^{T} \\ \vdots & \\ 1 & \end{array}\right]$ $\underbrace{\qquad}_{\alpha}$	$\left[\begin{array}{c} 00\dots0 \\ \hline E \end{array}\right]$ $\underbrace{\qquad}_{2^{\alpha}}$	$2^{I-1}+1$	$\begin{array}{c} \left[\begin{array}{cc} 0 & 000 \\ \hline 1 & 000 \\ 1 & 001 \\ 1 & 010 \\ 1 & 011 \\ 1 & 100 \\ 1 & 101 \\ 1 & 110 \\ 1 & 111 \end{array}\right] \\ enable\ x_1 x_2 x_3 \end{array}$	$\begin{array}{c} \left[\begin{array}{c} 00000000 \\ \hline 10000000 \\ 01000000 \\ 00100000 \\ 00010000 \\ 00001000 \\ 00000100 \\ 00000010 \\ 00000001 \end{array}\right] \\ F_1\dots F_8 \end{array}$	K=3 I=4 N=9

Table 2 Complete Detection Setsfor Standard Digital Components

NO. 8

K-bit
Divider

F (X,Y) =
$X \div Y = Z + R/Y$

$|R| < |Y|$

$F(X,Y)=(SF,z_1,\ldots,z_{K-1},R_1,\ldots,R_{K-1})$,

$X = (SX,\ x_1,\ldots,x_{K-1})$,

$Y = (SY,y_1,\ldots,y_{K-1})$,

$Z = (z_1,\ldots,z_{K-1})$,
$R = (R_1,\ldots,R_{K-1})$,
$K = 2^\alpha, |X|= \sum_{i=1}^{K-1} x_i 2^{K-1-i}$,

$|Y|= \sum_{i=1}^{K-1} y_i 2^{K-1-i}$, $|Z| = \sum_{i=1}^{K-1} z_i 2^{K-1-i}$,

$|R| = \sum_{i=1}^{K-1} R_i 2^{K-1-i}$.

$$\left[\begin{array}{c|c|c|c|c}
\multicolumn{5}{c}{00\ldots0010\ldots01} \\
\multicolumn{5}{c}{11\ldots1111\ldots11} \\
\hline
0 & & 1 & & 1 \\
0 & & 1 & & 1 \\
\vdots & \overline{M_\alpha-\underline{1}} & \vdots & 0 & \vdots \\
0 & & 1 & & 1 \\
\hline
0 & & 0 & \multicolumn{2}{c}{} \\
0 & & 0 & \multicolumn{2}{c}{} \\
\vdots & 0 & \vdots & \multicolumn{2}{c}{\overline{M_\alpha-\underline{1}}} \\
0 & & 0 & \multicolumn{2}{c}{} \\
\end{array}\right]
\begin{array}{c} \\ \end{array}$$
$$\qquad \underbrace{\qquad}_{2^\alpha-1} \qquad \underbrace{\qquad}_{2^\alpha-1}$$

$$\left[\begin{array}{c|c|c}
1 & 0\ldots00 & 00\ldots01 \\
0 & 0\ldots01 & 00\ldots00 \\
\hline
1 & & \\
1 & & \\
\vdots & \overline{M_\alpha-\underline{1}} & 0 \\
1 & & \\
\hline
0 & & \\
0 & & \\
\vdots & 0 & \overline{M_\alpha-\underline{1}} \\
0 & & \\
\end{array}\right]$$
$$\qquad \underbrace{\qquad}_{2^\alpha-1} \qquad \underbrace{\qquad}_{2^\alpha-1}$$

2 $\log_2 I$

$$\left[\begin{array}{c|c|c|c}
0 & 0000000 & 1 & 0000001 \\
1 & 1111111 & 1 & 1111111 \\
\hline
0 & 1111000 & 1 & 0000001 \\
0 & 1100110 & 1 & 0000001 \\
0 & 1010101 & 1 & 0000001 \\
\hline
0 & 0000000 & 0 & 1111000 \\
0 & 0000000 & 0 & 1100110 \\
0 & 0000000 & 0 & 1010101 \\
\end{array}\right]
\qquad
\left[\begin{array}{c|c|c}
1 & 0000000 & 0000001 \\
0 & 0000001 & 0000000 \\
\hline
1 & 1111000 & 0000000 \\
1 & 1100110 & 0000000 \\
1 & 1010101 & 0000000 \\
\hline
0 & 0000000 & 1111000 \\
0 & 0000000 & 1100110 \\
0 & 0000000 & 1010101 \\
\end{array}\right]$$
$$SX\ \ x_1\ldots x_7\ \ SY\ \ y_1\ldots y_7 \qquad\qquad SF\ \ z_1\ldots z_7 R_1\ldots R_7$$

K=8 I=16 N=8

~~22~~ 6c

NO. 9

RAM K-bits
$N=2^n$ words
*Before the test all memory cells reset to 0.
**Stuck-at faults at address lines are undetectable.

$Z = F(C,A_1,\ldots,A_n,x_1,\ldots,x_K)$

C = 1 for Read mode,
  0 for Write mode;
$(A_1,\ldots,A_n)$ - address;
$n = 2^\beta$,
$(x_1,\ldots,x_K)$ - input data ;
$K = 2^\alpha$, $\beta \le \alpha$,
$Z = (z_1,\ldots,z_K)$.
For any $A_1,\ldots,A_n$
and $x_1,\ldots,x_K$
$F(0,A_1,\ldots,A_n,x_1,\ldots,x_K)=$
$(0,\ldots,0)$.

$$\left[\begin{array}{c|c|c}
0 & & \\
1 & M_\beta \circ M_\beta & \\
0 & & \\
1 & & M_\alpha \circ 0 \\
\cline{2-2}
\cdot & 1010\ldots10 & \\
\cdot & 1010\ldots10 & \\
\vdots & \vdots & \\
\cline{3-3}
0 & 1010\ldots10 & 1010\ldots10 \\
1 & 1010\ldots10 & 0000\ldots00 \\
\end{array}\right]
\left[\begin{array}{c}
\\
\\
0 \circ M_\alpha \\
\\
\hline
0000\ldots00 \\
1010\ldots10 \\
\end{array}\right]$$
$$\qquad \underbrace{\qquad}_{2^\beta} \qquad \underbrace{\qquad}_{2^\alpha} \qquad \underbrace{\qquad}_{2^\alpha}$$

$2\alpha+2 \le$

$2\lfloor \log_2 I \rfloor$

$$\left[\begin{array}{c|c|c}
0 & 0011 & 00001111 \\
1 & 0011 & 00000000 \\
0 & 0101 & 00110011 \\
1 & 0101 & 00000000 \\
\cline{2-2}
0 & 1010 & 01010101 \\
1 & 1010 & 00000000 \\
\cline{3-3}
0 & 1010 & 10101010 \\
1 & 1010 & 00000000 \\
\end{array}\right]
\qquad
\left[\begin{array}{c}
00000000 \\
00001111 \\
00000000 \\
00110011 \\
00000000 \\
01010101 \\
\hline
00000000 \\
10101010 \\
\end{array}\right]$$
$$C\ \ A_1\text{--}A_4\ x_1\ldots x_8 \qquad\qquad z_1\ldots\ \ z_8$$

K=8 β=2 I=13 N=8

Table 2 (Cont'd) Complete Detection Sets for Standard Digital Components