3-1977

DUAL-MODE SEQUENTIAL LOGIC FOR FUNCTION INDEPENDENT FAULT-TESTING

Sumit DasGupta
Syracuse University

Carlos R.P. Hartmann
Syracuse University, chartman@syr.edu

Luther D. Rudolph
Syracuse University

DUAL-MODE SEQUENTIAL LOGIC FOR FUNCTION INDEPENDENT FAULT-TESTING

Sumit DasGupta
Carlos R. P. Hartmann
Luther D. Rudolph

March 1977

SCHOOL OF COMPUTER AND INFORMATION SCIENCE
SYRACUSE UNIVERSITY

DUAL-MODE SEQUENTIAL LOGIC FOR FUNCTION INDEPENDENT FAULT-TESTING

Sumit DasGupta

Carlos R. P. Hartmann

Luther D. Rudolph

School of Computer and Information Science

Syracuse University

Syracuse, New York 13210

Tele. (315) 423-2368

This work was supported in part by the Air Force Systems Command's Rome Air Development Center, Griffiss AFB, New York and by IBM under the Resident Study Program.

ABSTRACT

This paper presents a method of using hardware redundancy to ease the problem of fault testing in sequential logic networks. Sequential logic networks are constructed using two kinds of dual-mode logic gates, one of which is specifically required to initialize a feedback loop to some logic value. Initially, it is shown that these networks can be tested for all single stuck-at-faults with six function-independent tests. Next, this method is generalized to detect large classes of multiple faults with six function-independent tests. In both cases, the network must have the proper number of extra inputs.

I. Introduction

This paper presents the concept of function-independent fault testing of sequential networks. This is an extension of the theory of function-independent fault testing [1] previously presented by the same authors for combinational logic only. While one needs to consider signal flow in one direction only when deriving test patterns for a combinational network, the process is severely complicated by the presence of feedback loops in sequential networks. Thus, though existing techniques [2,3,4] work adequately for small networks with few feedback loops, they do not lend themselves well to large sequential networks where derivation of a single pattern may require multiple iterations through each loop in the network to determine a consistent set of input values.

In this paper, however, we present a method that fits into the present specter of increasing complexity of test pattern generation and decreasing component costs. This method uses redundancy, as in [1], to create a network structure where the tests for the network are no longer dependent on its functional definition or layout, thus eliminating the need to study signal propagation to determine the tests and prove their consistency. Since this paper is based primarily on the results of [1], we will briefly discuss its highlights here as a prelude to what is to be presented in this paper.

The basic idea of function-independent testing lies in the use of dual-mode logic gates. As Figure 1 shows, these gates have two distinct kinds of inputs - one set for data inputs and another set for control inputs which determine the mode of operation, i.e., test or

normal mode. These gates will be referred to as 2T(r) gates, where the number in parenthesis refers to the number of controls in that gate. As in [1], all gates in any network will be assumed to share identical controls to minimize the number of extra inputs but no control input of a gate can be fed from the same source as one of its data inputs.

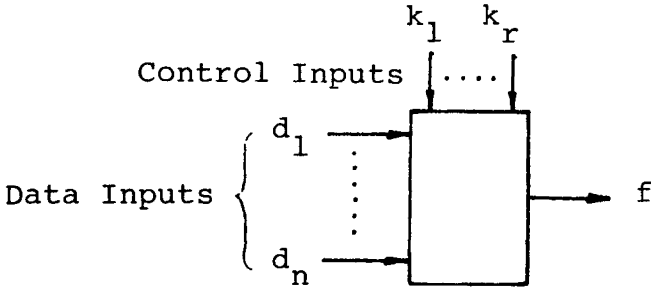

Figure 1. A 2T(r) gate

Although these gates can perform varied functions in normal mode, in test mode all the gates perform either an OR operation or an AND operation of the data inputs. So in test mode, all networks degenerate to either a massive OR function or an AND function of the data inputs, depending on the logic values at the control inputs. Thus, in the former condition, an all-0 pattern at the data inputs will test for any pattern of faults containing at least one stuck-at-1 (s-a-1) fault that can propagate to a network output. Likewise, in the latter condition, an all-1 pattern at the data inputs will test for any pattern of faults containing at least one stuck-at-0 (s-a-0) fault that can propagate to a network output. Furthermore, as Table I shows, the same tests would detect faults in control inputs independent of faults in the data inputs, where the maximum number, t, of control

TABLE I

k_1 ... k_t	... k_{r-t+1}	... k_r	d_1 ... d_n	f	
a_1 ... a_t	... a_{r-t+1}	... a_r	0 ... 0	0	
a_1 ... a_t	... a_{r-t+1}	... a_r	0 ... 0 1	1	TEST MODE -
a_1 ... a_t	... a_{r-t+1}	... a_r	0 ... 1 0	1	OR function
a_1 ... a_t	... a_{r-t+1}	... a_r	1 ... 1	1	
a_1 ... a_t	... a_{r-t+1}	... $a_{r-1}\bar{a}_r$	x ... x	1	
$\bar{a}_1 a_2$ ... a_t	... a_{r-t+1}	... a_r	x ... x	1	
a_1 ... a_t	... $a_{r-t}\bar{a}_{r-t+1}$	... $\bar{a}_r$	x ... x	1	
$\bar{a}_1$ ... $\bar{a}_t a_{t+1}$	... a_{r-t+1}	... a_r	x ... x	1	
a_1 ... a_t	... $a_{r-t-1}\bar{a}_{r-t}\bar{a}_{r-t+1}$	... $\bar{a}_r$	NORMAL MODE FOR LOGIC FUNCTIONS	?	
a_1 ... $a_t a_{t+1}\bar{a}_{t+2}$	... $\bar{a}_{r-t+1}$	... $\bar{a}_r$			
$\bar{a}_1$ ... $\bar{a}_t$	... $\bar{a}_{r-t}a_{r-t+1}$	... a_r	x ... x	0	
a_1 ... $a_t\bar{a}_{t+1}$	... $\bar{a}_{r-t+1}$	... $\bar{a}_r$	x ... x	0	
$\bar{a}_1$ ... $\bar{a}_t$	... $\bar{a}_{r-t+1}$	... a_r	x ... x	0	
$a_1\bar{a}_2$ ... $\bar{a}_t$	... $\bar{a}_{r-t+1}$	... $\bar{a}_r$	x ... x	0	
$\bar{a}_1$ ... $\bar{a}_t$	... $\bar{a}_{r-t+1}$	... $\bar{a}_r$	0 ... 0	0	
$\bar{a}_1$ ... $\bar{a}_t$	... $\bar{a}_{r-t+1}$	... $\bar{a}_r$	0 ... 0 1	0	TEST MODE -
$\bar{a}_1$ ... $\bar{a}_t$	... $\bar{a}_{r-t+1}$	... $\bar{a}_r$	1 ... 1 0	0	AND function
$\bar{a}_1$ ... $\bar{a}_t$	... $\bar{a}_{r-t+1}$	... $\bar{a}_r$	1 ... 1	1	

x ≡ Don't Care

input faults detectable in a network with r controls is expressed by the relation,

$$t = \left\lfloor \frac{r-2}{2} \right\rfloor$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. In fact, these two tests will detect any pattern of faults as long as there are no more than t faults appearing at the control inputs of any 2T(r) gate in any network.

In this paper, we use the 2T(r) gates to build sequential networks because of its unique property that two function-independent tests can test for all data input and output faults and upto a certain number of control input faults. As we shall show later, 2T(r) gates alone are not enough to make a sequential network testable with a function-independent test set since this gate does not have the capability of presetting a memory loop prior to application of a test. Thus, we shall develop another gate to do just that, specifying how and where it can be used and subsequently show that any network built with these two gates can be tested with six function-independent tests for at least all single faults in any network, extending this concept later to cover large classes of multiple faults.

In Section II, we discuss the problem of building sequential networks with 2T(r) gates only and then develop the theory of the dual-mode sequential network and the new gate to preset a memory loop. Section III contains conclusions and discussions.

II. The Dual-Mode Sequential Logic Network

In this section we will develop the dual-mode sequential (DMS) network so as to extend the concept of function-independent testing [1] to sequential networks. Since we have proven the characteristics of the 2T(r) gate, shown in Table I, regarding function-independent testing [1], we will assume that the DMS network is built with 2T(r) gates. In the combinational sections of the network, use of 2T(r) gates satisfies the requirements for function-independent testing but problems arise when feedback loops are built with 2T(r) gates only.

To understand the problem, consider the simple loop of Fig. 2. From [1], the two tests for the network would appear to be:

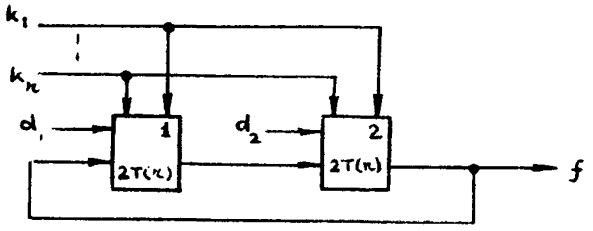

Fig. 2. Simple memory loop with 2T(r) gates

$(k_1,\ldots,k_r,d_1,d_2) = (a_1,\ldots,a_r,0,0) = \underset{\sim}{T}$ with expected output 0

and

$(k_1,\ldots,k_r,d_1,d_2) = (\bar{a}_1,\ldots,\bar{a}_r,1,1) = \bar{\underset{\sim}{T}}$ with expected output 1.

Now if test $\underset{\sim}{T}$ is applied and there exists a 1 at the output of one of the gates, (say gate 1), then as Table I indicates, that 1 will force a 1 at the output of gate 2 and thus give the indication of an error. But the 1 at the output of gate 1 may not be the result of

a stuck-at-fault (s-a-fault) but rather the result of the inherent memory of the loop. Thus, $\underset{\sim}{T}$ alone cannot test for s-a-1 faults in the loop. A similar case can be made for test $\bar{\underset{\sim}{T}}$ also. The first step, therefore, in testing a sequential network is to initialize it, i.e., before the all-0 pattern can be applied to the data inputs of the network, we must ensure that all gates in all fault-free loops have output 0. Similarly, all gates in fault-free loops must have output 1 before the all-1 pattern can be applied to the data inputs of the network.

It is appropriate at this point to present and discuss the properties required of the initializing patterns:

Property 1: The initializing pattern for any test must have the same expected output as the test.

Property 2: All feedback loops must be initialized independent of the data inputs.

Property 3: The initializing pattern and its related test pattern must have the same values specified for the data inputs of the network.

Property 4: If the initializing pattern is Hamming distance two or more away from its related test pattern, then all patterns in between must have the same expected output as the initializing and test patterns.

Property 1 follows from the fact that before the test with the all-0 pattern in the inputs can be applied, the network has to be initialized,

i.e. all gates in the loops must have output 0 so that the all-0 pattern can then be applied. But this test has output 0, hence the property. Similarly, the output for the initializing pattern preceding the test with the all-1 pattern in the data inputs must be the same as the test that it precedes. This property also implies that the initializing pattern and its related test pattern cannot be from the same segment of the truth table as specified by Lemma 1 for p = n in [1].

Property 2 follows from the argument that if initializing depended on a specific pattern in the data inputs of the gates in a loop, then any other pattern caused by the memory of the loop would prevent initialization. Hence there must be a pair of control conditions, one for each test, which would initialize the loops independent of the data inputs.

Though Property 2 gives near complete independence in selecting an initializing pattern for a particular test, Property 3 specifies that both should have the same pattern for the data inputs so as to prevent critical races from occurring when changing from the initializing pattern to the test pattern. For example, if $(a_1,\ldots,a_k,a_{k+1} \cdots a_r,0,\ldots,0,0)$ is one of the tests and $(\bar{a}_1,\ldots,\bar{a}_k,a_{k+1},\cdots a_r,0,\ldots,0,1)$ is the initializing pattern, then if in switching from the initializing pattern to the test pattern, an intermediate state is $(a_1,\ldots,a_r,0,\ldots,0,1)$, then as Table I shows, this pattern will circulate a 1 around a loop and give the indication of an error.

Property 4 implies that if the initializing pattern in Hamming distance [5] two through, say, k away from its related test pattern

then all patterns Hamming distance one through (k-1) from the initializing pattern and also Hamming distance (k-1) through one from the test pattern must have the same output. For example if $(a_1,\ldots,a_k,a_{k+1} \cdots a_r,0,\ldots,0)$ is the test and $(\bar{a}_1,\ldots,\bar{a}_k,a_{k+1},\ldots,a_r,0,\ldots,0)$ is the initializing pattern both with expected output 0, then if $(\bar{a}_1,\ldots,\bar{a}_{k-1},a_k,a_{k+1},\ldots,a_r,0,\ldots,0)$ has output 1 and is the intermediate state in changing from the initializing pattern to the test, then the intermediate state would force a 1 into the loop and destroy the result of initialization.

A study of the 2T(r) gate of Table I clearly proves that it cannot be used to initialize a network. Thus, a new gate is needed which will henceforth be referred to as a $4T(\hat{r})$ gate, where $\hat{r}$ is the number of control inputs in the gate. In what follows, a bound will be developed for the minimum number of control inputs that a DMS network requires to detect all single s-a-faults with a minimal set of function-independent tests. Then, the truth table of a $4T(\hat{r})$ gate will be presented that satisfies the above bound and which later will be shown to follow a bound similar to the bound in [1] for multiple faults. Later, another $4T(\hat{r})$ gate will be presented whose use in a DMS network helps to reduce the number of required control inputs and tests under certain restrictive assumptions.

The following theorem develops a lower bound for the number of control inputs required by a DMS network built with $4T(\hat{r})$ gates, where required, and 2T(4) gates, (as defined in Table II), where the 2T(4) gate is the gate with the minimum number of controls that permits single fault detection with a function-independent test set. Note that here, as in [1],

2T(r) gates are assumed to share corresponding controls, some of which may be used by $4T(\hat{r})$ gates also:

Theorem 1: A DMS network built with 2T(4) gates and $4T(\hat{r})$ gates, where necessary, requires at least five control inputs so that all single faults can be detected with a function-independent test set.

Proof: Since 2T(4) gates have four control inputs, a DMS network must have at least four control inputs. Now suppose that it is possible to have a DMS network with four controls such that all single faults can be detected with a function-independent test set. Therefore, a $4T(\hat{r})$ gate exists with four or fewer control inputs such that it shares all these controls with those of the 2T(4) gates so that the network can have four controls in all. It will now be shown that this limit is not achievable.

From [1], it is evident that the two function-independent tests for the 2T(4) gates are:

$$\underset{\sim}{T} = (k_1,k_2,k_3,k_4,d_1,\ldots,d_m) = (a_1,a_2,a_3,a_4,0,\ldots,0)$$

with expected output 0 and

$$\bar{\underset{\sim}{T}} = (k_1,k_2,k_3,k_4,d_1,\ldots,d_m) = (\bar{a}_1,\bar{a}_2,\bar{a}_3,\bar{a}_4,1,\ldots,1)$$

with expected output 1. Then three choices exist for each of the initializing patterns for the two tests. With respect to test $\underset{\sim}{T}$, these choices are:

(i) The control values for $\underset{\sim}{T}$ and the initializing pattern are the same.

TABLE II

k_1	k_2	k_3	k_4	d_1	d_n	f
a_1	a_2	a_3	a_4	0 ...	0	0
a_1	a_2	a_3	a_4	0 ... 0	1	1
a_1	a_2	a_3	a_4	0 ... 1	0	1
				.		.
				.		.
				.		.
a_1	a_2	a_3	a_4	1 ...	1	1
a_1	a_2	a_3	$\bar{a}_4$	x ...	x	1
a_1	a_2	$\bar{a}_3$	a_4	x	x	1
a_1	$\bar{a}_2$	a_3	a_4	x ...	x	1
$\bar{a}_1$	a_2	a_3	a_4	x ...	x	1
$\bar{a}_1$	$\bar{a}_2$	a_3	a_4	One or		
$\bar{a}_1$	a_2	$\bar{a}_3$	a_4	more logic		
a_1	$\bar{a}_2$	$\bar{a}_3$	a_4	functions		
$\bar{a}_1$	a_2	a_3	$\bar{a}_4$			
a_1	$\bar{a}_2$	a_3	$\bar{a}_4$			
a_1	a_2	$\bar{a}_3$	$\bar{a}_4$			
a_1	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	x ...	x	0
$\bar{a}_1$	a_2	$\bar{a}_3$	$\bar{a}_4$	x ...	x	0
$\bar{a}_1$	$\bar{a}_2$	a_3	$\bar{a}_4$	x ...	x	0
$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	a_4	x ...	x	0
$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	0 ...	0	0
$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	0 ... 0	1	0
				.		.
				.		.
				.		.
$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	1 ... 1	0	0
$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	1 ...	1	1

x ≡ Don't Care

(ii) The control values for $\underset{\sim}{T}$ and the initializing pattern are Hamming distance one apart.

(iii) The control values for $\underset{\sim}{T}$ and the initializing pattern are Hamming distance two or more apart.

Case (i): If $\underset{\sim}{T}$ and its related initializing pattern are the same, then $\underset{\sim}{T}$ will not detect any s-a-1 fault at the data inputs of the $4T(\hat{r})$ gates since its output is independent of data inputs when the initializing pattern is applied. Then, extra test is required with control values at least Hamming distance one away from those for $\underset{\sim}{T}$. But as Table II shows, a s-a-1 fault propagates through a 2T(4) gate as an erroneous logic value only when $(k_1,k_2,k_3,k_4) = (a_1,a_2,a_3,a_4)$. Hence, no fault is detected.

If $\underset{\sim}{T}$ and the initializing pattern differ in data inputs, as Table II shows, a pattern that is not all-0 in the data inputs may force a 1 at the output of one or more 2T(4) gates and may not be able to initialize all loops in all networks.

Case (ii): With reference to $\underset{\sim}{T}$, no control condition Hamming distance one from $(k_1,k_2,k_3,k_4) = (a_1,a_2,a_3,a_4)$ can be used to initialize a loop since as Table II shows, the output of a 2T(4) gate is 1.

Case (iii): Finally, no control condition Hamming distance two or more from $(k_1,k_2,k_3,k_4) = (a_1,a_2,a_3,a_4)$ can be used to initialize a loop to 0 since in changing from the initializing pattern to test $\underset{\sim}{T}$, one intermediate control condition may be Hamming distance one away from that of $\underset{\sim}{T}$ for which, as Case (ii) shows, all 2T(4) gates have output 1. Thus, the effect of initializing is destroyed.

Hence, none of these options listed above are applicable. A similar case can be made for test $\bar{\underset{\sim}{T}}$ also.

Q.E.D.

The above theorem shows that a DMS network needs at least five controls for function-independent testing. The following theorem develops a lower bound for the number of function-independent tests, a bound that will subsequently be shown to be achievable:

Theorem 2: DMS networks built with 2T(4) gates and 4T($\hat{r}$) gates, where necessary, and containing at least five control inputs cannot be tested with less than six tests.

Proof: Let the five controls be denoted k_1,k_2,k_3,k_4,k_5 of which k_1,k_2,k_3,k_4 drive all 2T(4) gates. Control k_5 and one or more of the remainder drive all 4T($\hat{r}$) gates. Since as far as 2T(4) gates are concerned, one of the tests for the network is

$$\underset{\sim}{T}_0 = (k_1,k_2,k_3,k_4,d_1,\ldots,d_m) = (a_1,a_2,a_3,a_4,0,\ldots,0)$$

with expected output 0, the related initializing sequence is either

$$(k_1,k_2,k_3,k_4,k_5,d_1,\ldots,d_m) = (a_1,a_2,a_3,a_4,0,0,\ldots,0)$$

or

$$(k_1,k_2,k_3,k_4,k_5,d_1,\ldots,d_m) = (a_1,a_2,a_3,a_4,1,0,\ldots,0)$$

with expected output 0 in either case. Let the initializing sequence, without loss of generality, be

$$\underset{\sim}{T}_i = (k_1,k_2,k_3,k_4,k_5,d_1,\ldots,d_m) = (a_1,a_2,a_3,a_4,0,0,\ldots,0)$$

with expected output 0. Then the complete specification for $\underset{\sim}{T}_0$ above is

$$\underset{\sim}{T}_0 = (k_1,k_2,k_3,k_4,k_5,d_1,\ldots,d_m) = (a_1,a_2,a_3,a_4,1,0,\ldots,0)$$

with expected output 0 also. Since $\underset{\sim}{T}_0$ will test all 2T(4) gates for s-a-1 faults, an erroneous logic value must be able to propagate through a $4T(\hat{r})$ gate. Hence $\underset{\sim}{T}_0$ will test all s-a-1 faults at the data inputs of the $4T(\hat{r})$ gates. However if k_5 is s-a-0, it will appear as if $\underset{\sim}{T}_i$ is applied, irrespective of whether $\underset{\sim}{T}_0$ or $\underset{\sim}{T}_i$ is applied, and the output will be 0 since $\underset{\sim}{T}_i$ is the initializing pattern and hence the fault will be undetectable. Again, if k_5 is s-a-1, the network cannot be initialized since it will appear as if $\underset{\sim}{T}_0$ is applied irrespective of whether $\underset{\sim}{T}_0$ or $\underset{\sim}{T}_i$ is applied, i.e. the network cannot be initialized to 0. Hence, the fault will be detected if an erroneous value appears in the loop because of its memory, i.e. this fault cannot be detected with certainty and therefore is an undetectable fault. A similar argument can be made to show that the complement pair of patterns, $\underset{\sim}{\bar{T}}_0$ and $\underset{\sim}{\bar{T}}_i$, where one is to initialize loops to 1 and the other to test 2T(4) gates for s-a-0 faults, cannot detect any fault in k_5. Hence, two more tests are required - one to test k_5 for a s-a-0 and another to test for a s-a-1 fault - thus requiring a total of at least six tests.

Q.E.D.

A $4T(\hat{r})$ gate is now presented such that the DMS network in which it is used satisfies the bounds set by Theorems 1 and 2. Also, the properties of a DMS network that uses this $4T(\hat{r})$ gate will be discussed. Evidently, this $4T(\hat{r})$ gate and all others like it must have at least two control inputs to satisfy the minimum requirement

of two initializing modes - one to set all loops to 0 and another to set all loops to 1 - and at least one condition for normal mode, assuming that normal mode can somehow be used to test this gate also.

So consider the truth table of Table III. Since the lower bound for single fault detection is five controls, one of the control inputs must be shared with one of the controls of 2T(4) gates in the network. Since the controls for 2T(4) gates have to be in test mode during initializing also, the control input of the 4T(2) gate that must be shared with 2T(4) gates must have the property that it cannot change in transferring from initializing to test mode. As Table III shows, $\hat{k}_1$ is the control input that can be so shared. Control $\hat{k}_2$ then becomes the control that switches the 4T(2) gate and the network from initializing to its related test mode. Table III also shows that in either test mode, i.e. $(\hat{k}_1,\hat{k}_2) = (\hat{a}_1,\hat{a}_2)$ and $(\bar{\hat{a}}_1,\bar{\hat{a}}_2)$, the output of the gate is the same as the data input. Thus, in test mode, this gate will propagate the logic value appearing at its data input, which is the property required to propagate a fault to an output. Note also that either test mode can represent normal mode.

At this stage, it is necessary to define a property to be applied to all DMS networks in this chapter since the 4T(2) gates in this chapter are basically of the same nature. The subsequent discussion justifies the need for this property.

TABLE III

k_1	k_2	d	f
$\hat{a}_1$	$\hat{a}_2$	0	0
$\hat{a}_1$	$\hat{a}_2$	1	1
$\hat{a}_1$	$\bar{\hat{a}}_2$	x	0
$\bar{\hat{a}}_1$	$\hat{a}_2$	x	1
$\bar{\hat{a}}_1$	$\bar{\hat{a}}_2$	0	0
$\bar{\hat{a}}_1$	$\bar{\hat{a}}_2$	1	1

Property 5: Every feedback loop must have one and only one $4T(\hat{r})$ gate to initialize the loop so that a logic value in propagating once along the shortest path in any loop encounters one and only one $4T(\hat{r})$ gate in that loop.

It is evident from what has been presented so far that every loop must have a $4T(\hat{r})$ gate to initialize it. At the same time, a loop cannot have more than one $4T(\hat{r})$ gate since then a fault in one $4T(\hat{r})$ gate will be blocked by a fault-free $4T(\hat{r})$ gate in the same loop and which will also prevent storing that information during initializing in order to transmit that information to an output at a later time. This also explains why an all-purpose gate, formed by merging the properties of $2T(r)$ and $4T(\hat{r})$ gates, cannot be used to build sequential networks since that would amount to having multiple gates with the properties of $4T(\hat{r})$ gates in the same loop.

The need for all loops, not just memory loops, i.e. those loops that store information, to have $4T(\hat{r})$ gates fits well with the general Huffman model of a sequential network shown in Figure 2 where all loops are considered alike. Property 5 is also necessary to initialize all fault-free loops in any DMS network configuration. For example, if in the network of Figure 3, the loop formed by the memory loops feeding each other did not contain a $4T(2)$ gate, then it would be impossible to initialize any of the loops. As specified by Property 5, one loop is formed by gates 1, 2 and 3, a second loop with gates 5, 6 and 7 and a third loop with gates 2, 3, 4, 6, 7, 8 and 9.

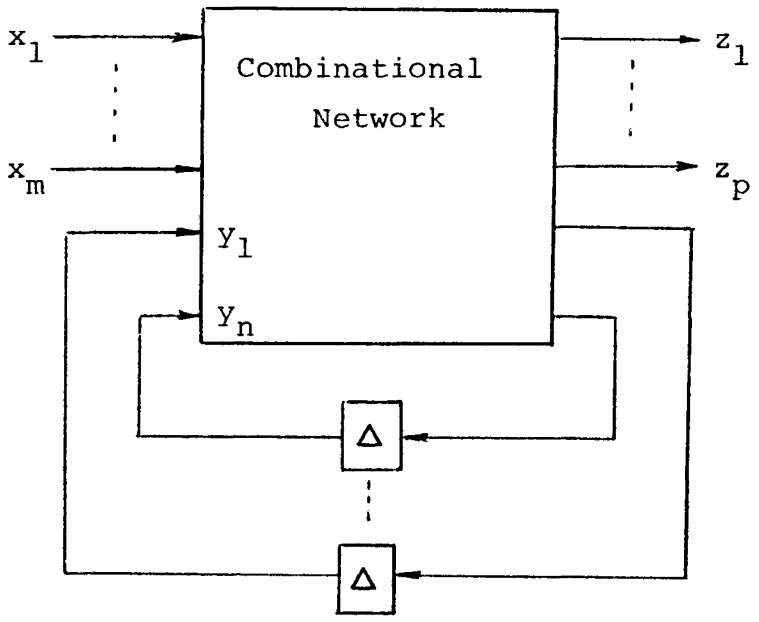

Fig. 2. Huffman Sequential Network Model

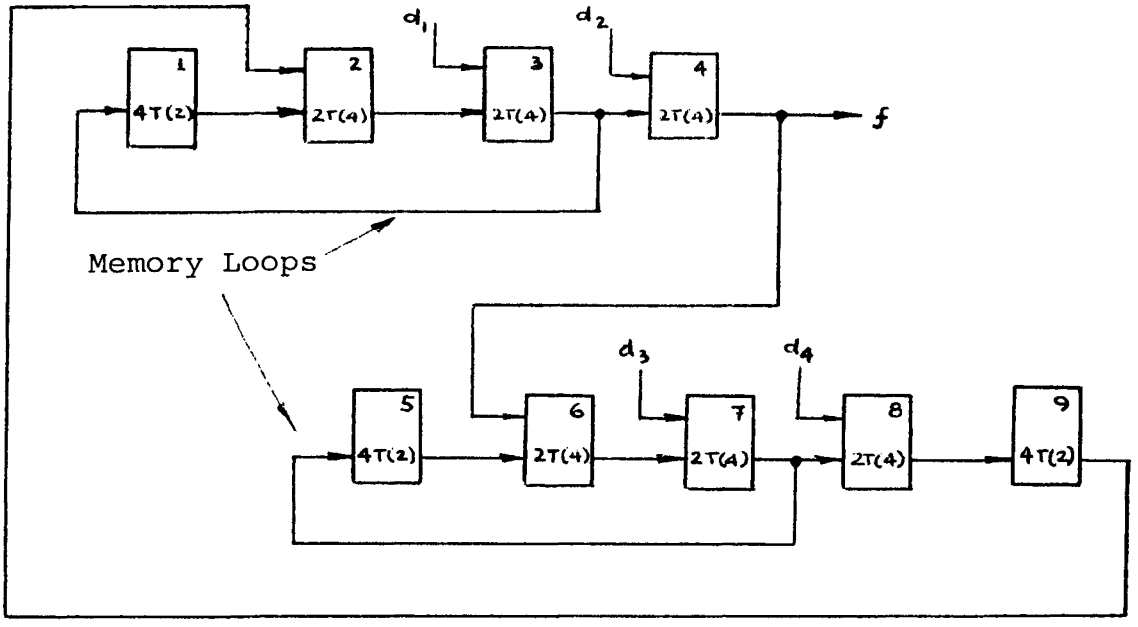

Fig. 3. Sequential Network satisfying Property 5

Now suppose it is assumed that control $\hat{k}_1$ of the 4T(2) gate shown in Table III is driven from the same network control input that drives the k_4 input of all 2T(4) gates in a DMS network. Then in relation to Table II and III,

$$a_4 = \hat{a}_1 .$$

Note that this sharing arrangement in no way limits control k_4 since there exists a normal mode for the 4T(2) gate both for $\hat{k}_1 = \hat{a}_1$ and $\bar{\hat{a}}_1$. Hence the logic value assumed by control k_4 in normal mode is determined only by the truth table of 2T(4) gate. However, selection of a suitable value for k_4 also determines the proper value that $\hat{k}_2$ must have in reference to Table III.

It is now shown that a DMS network built with 2T(4) gates and 4T(2) gates, where necessary as per Property 5, can be tested with a set of function-independent tests:

Theorem 3: Given any m-data input DMS network with five controls, k_1,k_2,k_3,k_4,k_5 where k_4,k_5 controls all 4T(2) gates and k_1,k_2,k_3,k_4 controls all 2T(4) gates, any single fault is detected by the following test set:

$\underset{\sim}{T}_e = (k_1,k_2,k_3,k_4=\hat{k}_1,k_5=\hat{k}_2,d_1,\ldots,d_m) = (a_1,a_2,\bar{a}_3,a_4=\hat{a}_1,\bar{\hat{a}}_2,0,\ldots,0)$

with expected output 1,

$\underset{\sim}{T}_i = (k_1,k_2,k_3,k_4=\hat{k}_1,k_5=\hat{k}_2,d_1,\ldots,d_m) = (a_1,a_2,a_3,a_4=\hat{a}_1,\bar{\hat{a}}_2,0,\ldots,0)$

with expected output 0,

$\underset{\sim}{T}_0 = (k_1,k_2,k_3,k_4=\hat{k}_1,k_5=k_2,d_1,\ldots,d_m) = (a_1,a_2,a_3,a_4=\hat{a}_1,\hat{a}_2,0,\ldots,0)$

with expected output 0,

$\bar{T}_{e} = (k_1,k_2,k_3,k_4=\hat{k}_1,k_5=\hat{k}_2,d_1,\ldots,d_m) = (\bar{a}_1,\bar{a}_2,a_3,\bar{a}_4=\bar{\hat{a}}_1,\hat{a}_2,1,\ldots,1)$

with expected output 0,

$\bar{T}_{i} = (k_1,k_2,k_3,k_4=\hat{k}_1,k_5=\hat{k}_2,d_1,\ldots,d_m) = (\bar{a}_1,\bar{a}_2,\bar{a}_3,\bar{a}_4=\bar{\hat{a}}_1,\hat{a}_2,1,\ldots,1)$

with expected output 1 and

$\bar{T}_{0} = (k_1,k_2,k_3,k_4=\hat{k}_1,k_5=\hat{k}_2,d_1,\ldots,d_m) = (\bar{a}_1,\bar{a}_2,\bar{a}_3,\bar{a}_4=\bar{\hat{a}}_1,\bar{\hat{a}}_2,1,\ldots,1)$

with expected output 1, where subsets $\{T_e,T_i,T_0\}$ and $\{\bar{T}_e,\bar{T}_i,\bar{T}_0\}$ represent ordered sequences of tests.

Proof: Noting that the expected outputs for T_0 and $\bar{T}_0$ are complements of each other, all network outputs are testable by T_0 or $\bar{T}_0$ since a fault in an output would force it to have the same logic value when either test is applied. Also since a fault in the output of a gate internal to the network is indistinguishable from corresponding faults in the inputs of gates fed by the faulty output, only input faults will be considered in proving this theorem. Moreover, faults in control lines that affect more than one gate can be considered as distributed faults in the control inputs of the affected gates since there is no way to distinguish between the two as far as stuck-fault testing is concerned.

In proving this theorem, four different types of faults need to be considered:

(i) a single fault that affects only 2T(4) gates,

(ii) a single fault at the data input of a 4T(2) gate,

(iii) a single fault affecting the $\hat{k}_1$ input of one or more 4T(2) gates and

(iv) a single fault affecting the $\hat{k}_2$ input of one or more 4T(2) gates.

Case (i): Consider some single fault that affects 2T(4) gates only. Then there exists at least one 2T(4) gate such that all gates in the path from the faulty gate to a network output are fault-free. Then after $\underset{\sim}{T}_i$ or $\bar{\underset{\sim}{T}}_i$ has initialized the network, $\underset{\sim}{T}_0$ or $\bar{\underset{\sim}{T}}_0$, being tests for 2T(4) gates as in [1], will propagate the fault to the output of that faulty gate and from there to the network output via the fault-free path since both $\underset{\sim}{T}_0$ and $\bar{\underset{\sim}{T}}_0$ puts the entire network in test mode.

Case (ii): Once $\underset{\sim}{T}_i$ or $\bar{\underset{\sim}{T}}_i$ has initialized the network to 0 or 1 respectively, application of $\underset{\sim}{T}_0$ or $\bar{\underset{\sim}{T}}_0$ forces a single data input fault at a 4T(2) gate to appear at its output and hence as a fault at an input of the 2T(4) gate fed by the faulty 4T(2) gate. This condition is now identical to that in Case (i) and hence this fault is detectable by $\underset{\sim}{T}_0$ or $\bar{\underset{\sim}{T}}_0$.

Case (iii): A fault in $\hat{k}_1$ may affect either 2T(4) gates or 4T(2) gates or both since this control is shared with k_4 of 2T(4) gates. Also, a single fault may affect one or more gates. Once again then, there must be at least one faulty gate such that all gates in the path from the faulty gate to a network output are fault-free. Then after $\underset{\sim}{T}_i$ or $\bar{\underset{\sim}{T}}_i$ has initialized the network, $\underset{\sim}{T}_0$ or $\bar{\underset{\sim}{T}}_0$ will propogate the fault to the output of that gate, irrespective of the type of gate it is, and then to the network output via the fault-free path since fault-free 2T(4) and 4T(2) gates will propagate a fault when $\underset{\sim}{T}_0$ or $\bar{\underset{\sim}{T}}_0$ is applied.

Case (iv): Because of the unique characteristics of control $\hat{k}_2$ a fault in $\hat{k}_2$ is tested by first storing an erroneous value in all the loops by applying tests $\underset{\sim}{T}_e$ or $\bar{\underset{\sim}{T}}_e$ followed by $\underset{\sim}{T}_i$ or $\bar{\underset{\sim}{T}}_i$ respectively which initializes only those loops with fault-free 4T(2) gates to the desired logic value while leaving the erroneous value still stored in the loops with faulty 4T(2) gates. Then there exists at least one loop with a faulty 4T(2) gate which has a path to a network output such that all gates from that loop to the network output are fault-free. Now when $\underset{\sim}{T}_0$ or $\bar{\underset{\sim}{T}}_0$ respectively is applied, the erroneous value previously stored in the fault loop will propagate to the output via the fault-free path since both $\underset{\sim}{T}_0$ and $\bar{\underset{\sim}{T}}_0$ force all gates into test mode. Thus, all single faults are detectable.

Q.E.D.

Note that a fault in $\hat{k}_2$ may be detectable after $\underset{\sim}{T}_i$ or $\bar{\underset{\sim}{T}}_i$ is applied. That would happen when the path from a faulty loop to a network output does not traverse through a fault-free loop since then the fault-free 4T(2) gate in the latter loop would block the propagation of the erroneous value when $\underset{\sim}{T}_i$ or $\bar{\underset{\sim}{T}}_i$ is applied.

The concept of function-independent testing will now be extended to the problem of multiple fault detection. The following theorem shows that at least (r+2) controls are needed to detect any pattern of faults as long as no gate in the DMS network has more than t faults in its control inputs where r is the number of controls required by the 2T(r) gates and the relation between r and t, as defined before in [1], is,

$$t = \lfloor (r-2)/2 \rfloor \ .$$

Note that while the necessity for five controls has been proven

earlier for single faults, it differs from the bound mentioned above, and soon to be proven, in that for t=1, at least six controls are required. This is because in the former case, only single faults were considered while the latter case covers all patterns of faults provided no gate has more than one fault in its control inputs.

Theorem 4: A DMS network built with 2T(r) gates and 4T($\hat{r}$) gates, where necessary, requires at least (r+2) control inputs so that any pattern of faults, provided no gate has more than t faults in its control inputs, can be tested with a set of function-independent tests, where r and t satisfy the relation,

$$t = \lfloor (r-2)/2 \rfloor \ .$$

Proof: Using the same approach as Theorem 1, it can be shown that a DMS network needs at least (r+1) controls, where r controls are required, as shown in [1], by the 2T(r) gates while the 4T($\hat{r}$) gates use the extra control and share upto r controls with the 2T(r) gates. It will now be shown by contradiction that at least (r+2) controls are required to detect all the faults specified in the theorem. So assume that (r+1) controls are sufficient. Let one of the tests for the 2T(r) gates, without loss of generality, be,

$$\underset{\sim}{T}_0 = (k_1,\ldots,k_r,k_{r+1},d_1,\ldots,d_m) = (a_1,\ldots,a_r,0,0,\ldots,0)$$

with expected output 0, where the values specified for k_1 through k_r and d_1 through d_m are as specified by Table I in [1]. Then the related initializing pattern must be

$$\underset{\sim}{T}_i = (k_1,\ldots,k_r,k_{r+1},d_1,\ldots,d_m) = (a_1,\ldots,a_r,1,0,\ldots,0)$$

with expected output 0. Then there exists two choices for the pair of complement tests:

(i) $\bar{T}_{0} = (k_1,\ldots,k_r,k_{r+1},d_1,\ldots,d_m) = (\bar{a}_1,\ldots,\bar{a}_r,1,1,\ldots,1)$

is the test with expected output 1 and

$$\bar{T}_{i} = (k_1,\ldots,k_r,k_{r+1},d_1,\ldots,d_m) = (\bar{a}_1,\ldots,\bar{a}_r,0,1,\ldots,1)$$

is the initializing pattern with expected output 1 or

(ii) $\bar{T}_{0} = (k_1,\ldots,k_r,k_{r+1},d_1,\ldots,d_m) = (\bar{a}_1,\ldots,\bar{a}_r,0,1,\ldots,1)$

is the test with expected output 1 and

$$\bar{T}_{i} = (k_1,\ldots,k_r,k_{r+1},d_1,\ldots,d_m) = (\bar{a}_1,\ldots,\bar{a}_r,1,1,\ldots,1)$$

is the initializing pattern with expected output 1.

Suppose the pair of complement tests is as defined in Case (i). Then consider one of the $4T(\hat{r})$ gates in which control k_{r+1} and the data inputs are s-a-0. One way to test for this would be to have a pattern whose output depends on the presence or absence of the fault pattern. Evidently, neither $\bar{T}_{i}$ nor $\bar{T}_{0}$ tests for this since with k_{r+1} s-a-0 and k_1 through k_r at values prescribed by $\bar{T}_{i}$ or $\bar{T}_{0}$, the gate will initialize the loop to 1 independent of the faults in its data inputs. Also, since the values for k_1, through k_r are fixed by Table I as required values to keep all 2T(r) gates in test mode, no other pattern exists whose output would depend on the presence or absence of the fault. An alternative approach to test this fault, as used in Theorem 3, is to store an erroneous value and then test the effectiveness of the initializing control, k_{r+1}. Even with this approach, this fault pattern is not tested by the sequence $\{\bar{T}_{i},\bar{T}_{0}\}$ since with the data inputs s-a-0, an erroneous 1 cannot be stored in

the loop. Thus, the output for both $\underset{\sim}{T}_i$ and $\underset{\sim}{T}_0$ would be 0 implying no fault exists. Also, the sequence $\{\bar{\underset{\sim}{T}}_i, \bar{\underset{\sim}{T}}_0\}$ as defined in Case (i) does not detect the fault since with k_{r+1} s-a-0, the gate will initialize the loop to 1 at all times, prevent storing erroneous data, and thus the output for both $\bar{\underset{\sim}{T}}_i$ and $\bar{\underset{\sim}{T}}_0$ will not indicate a fault. For Case (ii), a s-a-1 fault in k_{r+1} along with data inputs of the $4T(\hat{r})$ gate s-a-0 will not be detected. Hence, the contradiction.

Q.E.D.

Theorem 2 has already shown that we need a minimum of six function-independent tests. We will present a 4T(2) gate that allows us to meet this bound as well as the bound set by Theorem 4. We will also show that, independent of the value of r, any pattern of faults in this 4T(2) gate is detectable with the specified set of function-independent tests. So consider the truth table of the 4T(2) gate shown in Table IV. The properties of the gate are similar to the 4T(2) gate in Table III except that the individual segments of the table as defined by the control inputs have been changed. Also, Property 1 must still hold. We will now define a set of tests and prove that it satisfies Theorem 4.

Theorem 5: In any DMS network built with 2T(r) and 4T(2) gates, the latter as defined in Table IV, any pattern of s-a-faults, provided that faults in the control lines affect at most $t = \left\lfloor \frac{r-2}{2} \right\rfloor$ control inputs of any 2T(r) gate, will be detected by the test set $T = \{\underset{\sim}{T}_e, \underset{\sim}{T}_i, \underset{\sim}{T}_0, \hat{\underset{\sim}{T}}_i, \hat{\underset{\sim}{T}}_0, \hat{T}_e\}$ defined below:

$$(k_1,\ldots,k_{r-1},k_r,k_{r+1}=\hat{k}_1,k_{r+2}=\hat{k}_2,d_1,\ldots\ d_m) =$$

$$(a_1,\ldots,a_{r-1},\bar{a}_r,\bar{\bar{a}}_1,a_2,0,\ldots,0) = \underset{\sim}{T}_e$$

TABLE IV

$\hat{k}_1$	$\hat{k}_2$	d	f
$\hat{a}_1$	$\hat{a}_2$	0	0
$\hat{a}_1$	$\hat{a}_2$	1	1
$\hat{a}_1$	$\bar{\hat{a}}_2$	0	0
$\hat{a}_1$	$\bar{\hat{a}}_2$	1	1
$\bar{\hat{a}}_1$	$\hat{a}_2$	x	0
$\bar{\hat{a}}_1$	$\bar{\hat{a}}_2$	x	1

with expected output 1,

$$(k_1,\ldots,k_{r-1},k_r,k_{r+1}=\hat{k}_1,k_{r+2}=\hat{k}_2,d_1,\ldots,d_m) =$$
$$(a_1,\ldots,a_{r-1},a_r,\bar{\hat{a}}_1,\hat{a}_2,0,\ldots,0) = \underset{\sim}{T}_i$$

with expected output 0,

$$(k_1,\ldots,k_{r-1},k_r,k_{r+1}=\hat{k}_1,k_{r+2}=\hat{k}_2,d_1,\ldots,d_m) =$$
$$(a_1,\ldots,a_{r-1},a_r,\hat{a}_1,\hat{a}_2,0,\ldots,0) = \underset{\sim}{T}_0$$

with expected output 0,

$$(k_1,\ldots,k_{r-1},k_r,k_{r+1}=\hat{k}_1,k_{r+2}=\hat{k}_2,d_1,\ldots,d_m) =$$
$$(\bar{a}_1,\ldots,\bar{a}_{r-1},\bar{a}_r,\bar{\hat{a}}_1,\bar{\hat{a}}_2,1,\ldots,1) = \hat{\underset{\sim}{T}}_i$$

with expected output 1,

$$(k_1,\ldots,k_{r-1},k_r,k_{r+1}=\hat{k}_1,k_{r+2}=\hat{k}_2,d_1,\ldots,d_m) =$$
$$(\bar{a}_1,\ldots,\bar{a}_{r-1},\bar{a}_r,\hat{a}_1,\bar{\hat{a}}_2,1,\ldots,1) = \hat{\underset{\sim}{T}}_0$$

with expected output 1 and

$$(k_1,\ldots,k_{r-1},k_r,k_{r+1}=\hat{k}_1,k_{r+2}=\hat{k}_2,d_1,\ldots,d_m) =$$
$$(\bar{a}_1,\ldots,\bar{a}_{r-1},\bar{a}_r,\hat{a}_1,\hat{a}_2,1,\ldots,1) = \hat{\underset{\sim}{T}}_e$$

with expected output 1 where the subsets $\{\underset{\sim}{T}_e,\underset{\sim}{T}_i,\underset{\sim}{T}_0\}$ and $\{\hat{\underset{\sim}{T}}_i,\hat{\underset{\sim}{T}}_0,\hat{\underset{\sim}{T}}_e\}$ are ordered sequences of tests and where controls, k_r and k_{r+1} drive 4T(2) gates only while controls, $k_1,\ldots,k_r$, drive 2T(r) gates only.

Proof: Since test $\underset{\sim}{T}_0$ and $\hat{\underset{\sim}{T}}_e$ have opposite outputs, all network outputs will be tested. Also, since a fault in a gate output is indistinguishable from identical faults in the data inputs fed by that output, we will consider input faults only in this proof. Finally, faults in control lines that affect more than one gate will be considered as distributed faults in the control inputs of the same gates.

Now consider any pattern of faults provided no 2T(r) gate has more than t faults in its control inputs. Then there exists at least one gate which has a path to a network output such that all gates in the path from the faulty gate to that output are fault-free. Then we need to consider three different conditions:

(a) the faulty gate is a 2T(r) gate

(b) the faulty gate is a 4T(2) gate in a loop containing no faulty 2T(r) gates

(c) the faulty gate is a 4T(2) gate in a loop with at least one 2T(r) gate with a fault pattern at its inputs which can prevent propagation of some desired logic value around that loop.

Consider Case (a). In a fault free network, sequences $\underset{\sim}{T}_i, \underset{\sim}{T}_0$ and $\hat{\underset{\sim}{T}}_i, \hat{\underset{\sim}{T}}_0$ will force a logic value of 0 and 1 respectively at all network outputs. In a faulty network, however, fitting the description of Case (a), $\underset{\sim}{T}_0$ or $\hat{\underset{\sim}{T}}_0$ will force the fault to the output of the faulty 2T(r) gate since $\underset{\sim}{T}_0$ and $\hat{\underset{\sim}{T}}_0$ are tests for 2T(r) gates. Propagation of the fault to the network output via the fault-free path is guaranteed because $\underset{\sim}{T}_0$ and $\hat{\underset{\sim}{T}}_0$ will put all the gates in that path in test mode.

Next consider Case (b). In a fault-free network, all network outputs will be as defined before when sequences $\underset{\sim}{T}_e, \underset{\sim}{T}_i, \underset{\sim}{T}_0$ and $\hat{\underset{\sim}{T}}_i, \hat{\underset{\sim}{T}}_0, \hat{\underset{\sim}{T}}_e$ are applied. In a faulty network fitting Case (b), any fault in the 4T(2) gate is detectable at least at the outputs of the loop, as shown in Table V. Also, since every test sequence in Table V ends

TABLE V

	k_1 s-a-	k_2 s-a-	d s-a-	TEST SEQUENCE
*	$\hat{a}_1$			T_e, T_i, T_0
	a_1			$\hat{T}_i, \hat{T}_0, \hat{T}_e$
*		$\hat{a}_2$		$\hat{T}_i, \hat{T}_0$
*		a_2		T_e, T_i, T_0
			0	$\hat{T}_i, \hat{T}_0$
			1	T_e, T_i, T_0
*	$\hat{a}_1$	$\hat{a}_2$		T_e, T_i, T_0
*	a_1	a_2		T_e, T_i, T_0
	a_1	$\hat{a}_2$		$\hat{T}_i, \hat{T}_0$
	$\hat{a}_1$	$\hat{a}_2$		T_e, T_i, T_0
	$\hat{a}_1$		0	$\hat{T}_i, \hat{T}_0$
	$\hat{a}_1$		1	T_e, T_i, T_0
	a_1		0	$\hat{T}_i, \hat{T}_0, \hat{T}_e$
	a_1		1	$\hat{T}_i, \hat{T}_0, T_e$
		$\hat{a}_2$	0	T_i, T_0
		a_2	1	T_e, T_i, T_0
		$\hat{a}_2$	0	$T_i, \hat{T}_0$
		a_2	1	T_e, T_i, T_0
	$\hat{a}_1$	$\hat{a}_2$	0	T_i, T_0
	$\hat{a}_1$	a_2	1	T_e, T_i, T_0
	a_1	$\hat{a}_2$	0	T_i, T_0
	$\hat{a}_1$	a_2	1	T_e, T_i, T_0
	a_1	$\hat{a}_2$	0	T_i, T_0
	$\hat{a}_1$	$\hat{a}_2$	1	T_i, T_0
	a_1	a_2	0	T_e, T_i, T_0
	a_1	a_2	1	T_e, T_i, T_0

*faults requiring storage of erroneous value.

with either $\underset{\sim}{T}_0$, $\hat{\underset{\sim}{T}}_0$ or $\hat{\underset{\sim}{T}}_e$, all of which put the network in test mode, the faulty logic value will propagate to the network output via the fault free path.

Case (c) requires the most attention. Since the particular 4T(2) gate in this case still has a fault-free path to an output, according to assumption for this case, the other faults in the same loop has to be beyond the loop output that feeds the fault-free path. To prove that all faults described by Case (c) are detectable, note that, of all faults considered in Table V, only those marked with an asterisk require storage of an erroneous value for detection of the fault. Those that do not depend on storage are patterns with either $\hat{k}_1$ s-a-$\bar{\hat{a}}_1$ or the data input is at fault or both. In all those cases, the output of the 4T(2) gate is not affected by other faults in the loop and, as Table V shows, these faults propagate to the output of the 4T(2) gate when $\underset{\sim}{T}_0$, $\hat{\underset{\sim}{T}}_0$ or $\hat{\underset{\sim}{T}}_e$ is applied, each of which puts all gates in test mode and thus guarantees propagation to the network output via the fault-free path. Those faults that do depend on storage are patterns with the data input of the 4T(2) gate fault-free and the $\hat{k}_1$ input either fault-free or s-a-$\hat{a}_1$. In both cases, the 4T(2) gate will be in test mode when $\underset{\sim}{T}_0$, $\hat{\underset{\sim}{T}}_0$ or $\hat{\underset{\sim}{T}}_e$ is applied, thus allowing the nearest fault in the loop to propagate to its output and then to the network output via the fault-free path. This is identical to Case (a). Hence all faults are detectable provided no 2T(r) gate has more than t faults in its control inputs.

Q.E.D.

Using the same approach as in the above theorem we can show that the test set $\hat{T} = \{T_e, T_i, T_0, T_1, \hat{T}_i, \hat{T}_0\}$ will also detect the same faults as defined in above theorem where tests $T_e, T_i, T_0, \hat{T}_i$ and $\hat{T}_0$ are as defined in the above theorem while T_1 is as defined below:

$$(k_1,\ldots,k_r,k_{r+1}=\hat{k}_1,k_{r+2}=\hat{k}_2,d_1,\ldots,d_m) = (a_1,\ldots,a_r,\hat{a}_1,\bar{\hat{a}}_2,0,\ldots,0)$$

with expected output 0.

As a consequence of Theorem 5, we have the following corollary:

Corollary 1: In any DMS network with (r+2) controls, built with 2T(r) gates and 4T(2) gates shown in Table IV, any pattern of $t = \left\lfloor \frac{r-2}{2} \right\rfloor$ or fewer faults will be detected by both the test sets T and $\hat{T}$ defined before.

Theorem 4 and the existence of at least one 4T(2) gate, as shown in Table IV, that satisfies the bound of that theorem implies the following theorem:

Theorem 6: The necessary and sufficient condition for a DMS network so that all patterns of faults so that no 2T(r) gate has more than t faults in its control inputs can be tested with a function - independent test set is that the network have at least (r+2) = (2t+4) control inputs, provided that the distance between normal mode and any test mode in a 2T(r) gate, as defined by the control values, is at least $\lfloor r/2 \rfloor$.

An interesting special case arises when the output of the gate

that drives the 4T(2) gate in each loop also serves as the only loop output. Then under the assumption that a fault in the data input of the 4T(2) gate also affects the output of the 2T(4) gate driving it and hence the loop output, it is possible to show that there exists at least one 4T(2) gate such that the DMS network built with it needs only (2t+2) controls and four function-independent tests. Consider the 4T(2) gate of Table VI. If controls $\hat{k}_1$ and $\hat{k}_2$ are shared with controls k_1 and k_2 of 2T(r) gates respectively, then it can be shown that the test set $T_s = \{\underset{\sim}{T}_{s_1}, \underset{\sim}{T}_{s_2}, \bar{\underset{\sim}{T}}_{s_1}, \bar{\underset{\sim}{T}}_{s_2}\}$ will detect any pattern of faults provided that no 2T(r) gate has more than $t = \left\lfloor \frac{r-2}{2} \right\rfloor$ faults in its control inputs, where tests $\underset{\sim}{T}_{s_1}$, $\underset{\sim}{T}_{s_2}$, $\bar{\underset{\sim}{T}}_{s_1}$ and $\bar{\underset{\sim}{T}}_{s_2}$ are as defined below:

$(\hat{k}_1=k_1, \hat{k}_2=k_2, k_3, k_4, \ldots k_r, d_1 \ldots d_m) = (\hat{a}_1=a_1, \hat{a}_2=a_2, \bar{a}_3, a_4, \ldots a_r, 0, \ldots 0)$

$= \underset{\sim}{T}_{s_1}$ with expected output 1,

$(\hat{k}_1=k_1, \hat{k}_2=k_2, k_3, k_4, \ldots k_r, d_1 \ldots d_m) = (\hat{a}_1=a_1, \hat{a}_2=a_2, a_3, a_4, \ldots a_r, 0, \ldots 0)$

$= \underset{\sim}{T}_{s_2}$ with expected output 0,

$(\hat{k}_1=k_1, \hat{k}_2=k_2, k_3, k_4, \ldots k_r, d_1, \ldots d_m) = (\bar{\hat{a}}_1=\bar{a}_1, \bar{\hat{a}}_2, \bar{a}_2, a_3, \bar{a}_4, \ldots \bar{a}_r, 1, \ldots 1)$

$= \bar{\underset{\sim}{T}}_{s_1}$ with expected output 0 and

$(\hat{k}_1=k_1, \hat{k}_2=k_2, k_3, k_4, \ldots k_r, d_1 \ldots d_m) = (\bar{\hat{a}}_1=\bar{a}_1, \bar{\hat{a}}_2, \bar{a}_2, \bar{a}_3, \bar{a}_4, \ldots \bar{a}_r, 1, \ldots 1)$

$= \bar{\underset{\sim}{T}}_{s_2}$ with expected output 1.

As Table VII shows, the above test set detects all control input faults in the 4T(2) gate of Table VI. Date input faults in 4T(2) are ignored since by assumption they are analogous to faults occurring at

TABLE VI

k_1	k_2	d	f
$\hat{a}_1$	$\hat{a}$	x	0
$\hat{a}_1$	$\bar{\hat{a}}$	0	0
$\hat{a}_1$	$\bar{\hat{a}}$	1	1
$\bar{\hat{a}}_1$	$\hat{a}$	0	0
$\bar{\hat{a}}_1$	$\hat{a}$	1	1
$\bar{\hat{a}}_1$	$\bar{\hat{a}}$	x	1

x ≡ don't care

TABLE VII

k_1 s-a	k_2 s-a	TEST SEQUENCE
$\hat{a}_1=a_1$		$\bar{\underset{\sim}{T}}s_1, \bar{\underset{\sim}{T}}s_2$
$\bar{\hat{a}}_1=\bar{a}_1$		$\underset{\sim}{T}s_1, \underset{\sim}{T}s_2$
	$\hat{a}_2=a_2$	$\bar{\underset{\sim}{T}}s_1, \bar{\underset{\sim}{T}}s_2$
	$\bar{\hat{a}}_2=\bar{a}_2$	$\underset{\sim}{T}s_1, \underset{\sim}{T}s_2$
$\hat{a}_1=a_1$	$\hat{a}_2=a_2$	$\bar{\underset{\sim}{T}}s_1, \bar{\underset{\sim}{T}}s_2$
$\hat{a}_1=a_1$	$\bar{\hat{a}}_2=\bar{a}_2$	$\underset{\sim}{T}s_1, \underset{\sim}{T}s_2$
$\bar{\hat{a}}_1=\bar{a}_1$	$\hat{a}_2=a_2$	$\bar{\underset{\sim}{T}}s_1, \bar{\underset{\sim}{T}}s_2$
$\bar{\hat{a}}_1=\bar{a}_1$	$\bar{\hat{a}}_2=\bar{a}_2$	$\underset{\sim}{T}s_1, \underset{\sim}{T}s_2$

the outputs of 2T(r) gates. Test set T_s can be shown to detect all other faults provided that no 2T(r) gate has more than t faults in its control inputs. Note that subsets $\{\underset{\sim}{T}_{s_1}, \underset{\sim}{T}_{s_2}\}$ and $\{\bar{\underset{\sim}{T}}_{s_1}, \bar{\underset{\sim}{T}}_{s_2}\}$ represent ordered sequences of tests.

III. Conclusions and Discussions

In this paper, we have first briefly covered the unique properties of the 2T(r) gate that were developed in [1], emphasizing the fact that the 2T(r) gate does not have the capability to initialize loops in sequential networks. Having shown the need for a second gate, referred to as $4T(\hat{r})$ gates, we first proved that at least five controls are required for any DMS network so that all single faults can be detected with a function-independent test set. We then showed that the function-independent test set must contain at least six tests and finally presented a particular 4T(2) gate which satisfies both the above bounds.

Next, we addressed the problem of multiple faults. We first showed that a DMS network needs at least (r+2) controls to detect any kind of multiple faults. We then presented a specific 4T(2) gate that satisfies the above bound and having two alternative sets of six function-independent tests both of which detect all possible fault combinations in the 4T(2) gate. Total fault coverage then becomes a function of the number of control inputs that the 2T(r) gates have, i.e. (r+2) controls are necessary for a function-independent test set to detect any pattern of faults provided no 2T(r) gate has more than $t = \left\lfloor \frac{r-2}{2} \right\rfloor$ faults in its control inputs. Moreover, we showed an interesting extension of the above idea where under certain restrictive assumptions, there does exist a 4T(2) gate whose use in a network requires the network to have just r controls to detect any pattern of faults with four tests, provided no 2T(r) gate has more

than $t = \left\lfloor \frac{r-2}{2} \right\rfloor$ faults in its control inputs.

The above results are important for two reasons - first, function-independence of the test sets makes the test sets immune to the function and implementation of the network and secondly, the fixed size of the test set overcomes the problem of handling test sets whose sizes are related to the complexity of the network. In the case of multiple faults, it is interesting to note that as in the case of DMC networks in [1], given a certain number of control inputs, enlarging the test set will not increase the test coverage. This conclusion, reached earlier in [1], applies here also since increase in test coverage is strictly a function of the number of controls in 2T(r) gates because all faults in the particular 4T(2) gate defined for multiple fault detection are detectable.

Note that as in the case of DMC networks, logical redundancies in DMS networks become invisible in test mode and hence do not affect testing of DMS networks for all specified faults. Also the treatment of sequential networks here does not make any distinction between the different data inputs which have either an all-0 or all-1 pattern in test mode. Hence, this approach applies just as well to asynchronous designs as to the synchronous ones where the inputs referred to here as data inputs represent both logic and clock inputs in normal mode.

One must note that, as in [1], we have considered only faults that occur at the inputs or output of a gate. We have not investigated whether there is any fault internal to the 4T(2) gates or the 2T(r)

gates which does not manifest itself as a s-a-fault at an input or output of the gate since the effect of such a fault is dependent on the specific realization of the gate on an LSI chip. In defense of our approach, it can be said that if any fault does exist that does not appear outside the gate, it may be possible either to interconnect the internal elements of the gates in such a way that if an undetectable fault occurs, it also causes a detectable fault to occur, or to use redundancy in interconnections to drastically reduce the probability of such an undetectable failure.

References

[1] S. DasGupta, C.R.P. Hartmann and L.D. Rudolph, "Dual-Mode Combinational Logic for Function-Independent Fault Testing," Computer and Information Science Technical Report #2-77, Syracuse University, submitted for publication to the IEEE Transactions on Computers.

[2] A.D. Friedman and P.R. Menon, Fault Detection in Digital Circuits, Prentice Hall, 1971.

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

[4] H.Y. Chang, E.G. Manning and G. Metz, Fault Diagnosis in Digital Systems, Wiley Interscience, 1970.

[5] W.W. Peterson and E.J. Weldon, Error Correcting Codes, 2nd ed., Cambridge, Mass: M.I.T. Press, 1972.