Center for Reliable Computing

SEL 76-038

CASCADE STRUCTURE IN TOTALLY SELF-CHECKING NETWORKS

Stephen Kolupaev

Technical Report No. 108

April 1976

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

This work was supported by the National Science Foundation under Grant GJ-27527.

CASCADE STRUCTURE IN TOTALLY SELF-CHECKING NETWORKS

Stephen Kolupaev

Technical Report No. 108

April 1976

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

ABSTRACT

In the well-known totally self-checking (TSC) network, a failure must not change one output codework into another. Called the fault-secure property, this permits a receiver of the net's output to assume that any codework it receives is correct. Further, the self-testing property requires that each possible failure in the net must produce at least one non-code output. Thus a receiver can monitor the health of the network by watching for non-code outputs.

In this paper we propose modifications of these two properties. The self-testing property is made more stringent. Each possible failure in the net is required to produce an output which is in a distinguished subset of the non-code outputs. The fault-secure requirement is modified to permit a fault to interchange certain output codewords. In particular, all outputs not in the distinguished subset are partitioned into equivalent classes, and a fault is permitted to change the output from one codeword to another codeword in the same class. However, a fault is not permitted to change the output from a codeword to any member of a different equivalence class (one not containing the correct output).

These modified properties define a generalization of the TSC network. A network which meets the modified properties is called a generalized self-checking (GSC) network. Self-checking and self-testing (Morphic) networks and TSC networks are special cases of the GSC network.

Examining TSC networks, we find a further connection with the GSC network. It has been known for some time that not every subnetwork of a

TSC network need be TSC. We show that every subnetwork of a TSC network is GSC, and every TSC network is a cascade of GSC networks. This establishes the GSC network as the basic building block from which every TSC network is constructed.

We explore a brute-force method for constructing a desired TSC network by cascading GSC subnetworks. The method resorts to enumeration at many points of decision and thus is not a practical design tool. However, it does yield a very nice alternate realization of the Morphic OR, and suggests specializations which merit further study.

INDEX TERM: Self-checking networks
Fault-tolerant computer
Logic design

1. INTRODUCTION

A totally self-checking (TSC) network [Anderson, 1973] is a fault-tolerant network. When its inputs are held within a special set of input combinations, called its input code, it exhibits a systematic fault-tolerance. For a set of faults, called the fault set, the errors permitted at the network output are limited. In normal use, the network's inputs are kept within the input code. The set of output combinations which then occur with no faults is called the output code. All other output combinations are called non-code outputs.

When the inputs are not within the input code, the TSC network's behavior under faults is unrestricted. But while the inputs are held within its input code, the output behavior of a TSC network satisfies the following two rules:

1) Self-testing (ST) property: For each fault, f, there must be some input codework i, such that the network output, for input i with f present, is not an output codeword.

2) Fault-secure (FS) property: For each fault f, and each input codeword i, if the network output for input i with f present is an output codeword, it must be the correct output codeword.

The FS property requires that no fault may change the output from one codeword to another. The ST property requires each fault to cause at least one non-code output. These are useful properties

in fault-tolerant computing. A failing TSC network will soon give a non-code output, which can trigger an error-alarm. Until that happens, the outputs are codewords which may be believed correct.

We shall generalize the ST and FS properties because the following considerations often apply. There are applications where output codewords elicit identical actions in subsequent circuits. An example is the TSC parity checker reported by [Carter, 1968], checking for even parity on a bus. In normal operation, various even parity signals pass by on the bus, and the checker output alternates between two output codewords. The particular output codeword which appears indicates which of 2 groups of even-parity signals is present on the bus. The checker gives a non-code output for every odd parity signal on the bus.

Subsequent circuitry does not care which even parity signal may be passing on the bus. It only cares to know whether the bus signal has bad parity (checker output non-code), or the checker has an internal fault (checker output non-code). Subsequent circuitry takes the same action for both output codewords, so both of them can be thought to have the same meaning. In this application, there is no reason to prohibit faults from changing one output codeword to another, for both have the same meaning. Subsequent circuitry might see an incorrect output codeword, but would take the same action as for the correct output.

In the general case it may be difficult to build an error-alarm circuit which triggers on all non-code outputs. So why not permit a non-code output to go unrecognized? That output could be interpreted as one of the output codewords, provided that the meaning attributed to the non-code output is the correct one. Subsequent circuits would have to take the same action for the unrecognized non-code output as they would for the correct output. And to assure eventual triggering of the error-alarm, each fault would still have to cause a recognized non-code output.

If the rules for a TSC network are relaxed in this way, the network output response under faults is less highly constrained, the network is just as useful, and perhaps easier to build. We shall call a network which follows the relaxed rules generalized self-checking (GSC). While the inputs are held within the input code, the output behavior of a GSC network satisfies the following:

1) Generalized Self-Testing (GST) Property: For each fault f, there must be some input codeword i, such that the output, for input i and fault f, is a recognized non-code output.

2) Generalized Fault-Secure (GFS) Property: For each fault f and each input codeword i, if the output for input i when f is present is not a recognized non-code output, then it must have the same meaning as the correct output.

A GSC network can be viewed as a TSC network where sets of output vectors replace individual output vectors. Where the TSC network has an output codeword, the GSC network has a set of outputs which contains one or more output codewords, plus perhaps some non-code outputs. All the members of the set have the same meaning. Under faults, the GSC network preserves these sets (GFS property), whereas the TSC network preserves the output codewords (FS property). Where the TSC network has the set of all non-code outputs, the GSC network has the set of recognized non-code outputs. Under faults, the TSC network produces non-code outputs (ST property), while the GSC network produces recognized non-code outputs (GST property).

A TSC network is clearly a special case of the GSC network. If all non-code outputs are recognized as non-code, and every output codeword has a different meaning, the GST and GFS properties reduce to the original ST and FS properties of the TSC network.

The oldest type of self-checking network is the self-checking and self-testing (SCST) network proposed by [Carter, 1968]. The most well-known networks of this type are the "Morphic" networks [Carter, 1971]. Carter's SCST network is a special case of the GSC network. The rules it follows are similar to the GST and GFS properties, but are stated differently. These rules are:

1) "Every single gate failure is detected during normal operation,"

2) "No erroneous result induced by a single gate-failure goes undetected."

These correspond respectively to the GST and GFS properties for a fault set containing all single faults. The generalization we have made here shows a conceptual similarity of the SCST and TSC networks. In this paper we will show another connection.

An acyclic network is one with no feedback loops. For such a network we can define a complete subnetwork. A complete subnetwork of a parent network is some portion which interrupts every path between the inputs and outputs. Every such path must enter the subnetwork exactly once, and exit from the subnetwork exactly once. Thus a complete subnetwork is a contiguous subnetwork which stands astride all paths from inputs to outputs.

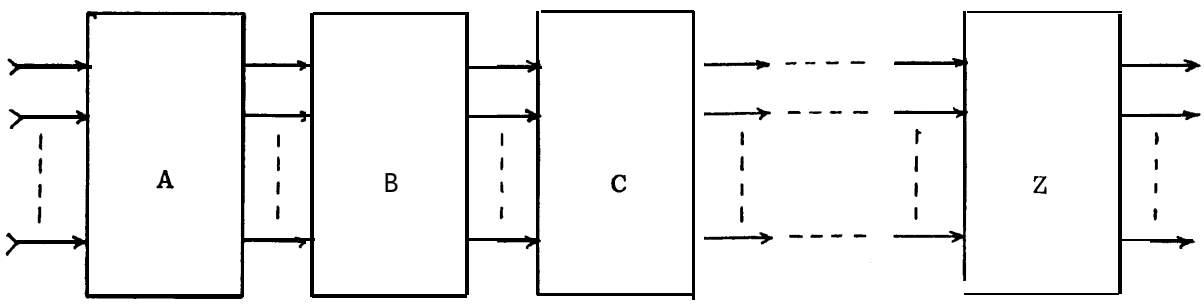

Fig. 1.1 Cascade of complete subnetworks

Any cycle-free TSC network can be cut into a cascade of 2 or more complete subnetworks, as in Fig. 1.1. As the major result of this paper, we

1) Show that every complete subnetwork of a TSC parent network is GSC. Therefore every acyclic TSC network is a cascade of GSC networks.

2) Develop a technique for joining together 2 GSC networks to form a GSC cascade.

2. GSC NETWORK FORMALIZED

To make use of a GSC network, one must know which outputs should be recognized as non-code. For the remaining outputs, one must know which should share the same meaning. These features of a GSC network shall be described by an output partition. An output partition identifies a subset of the outputs which should be recognized as non-code, called the alarm set. In application, these outputs should trigger an error-alarm. All the remaining outputs are partitioned into equivalence classes, called M-classes, where every M-class must contain at least one output codeword. In application the same meaning should be attributed to all members of an M-class.

Definition:

The output partition Q of a network is a proper subset Z of the set U of all possible output vectors, and a partitioning of the remainder U-Z into several equivalence classes $E_1, E_2, \ldots, E_n$.

Z is called the alarm set of Q, and may contain no output codewords. $E_1, E_2, \ldots, E_n$ are called the M-classes of Q, and each must contain at least one output codeword. Q may be written:

$$Z||E_1|E_2|\ldots|E_n.$$

An output partition serves as a set of rules for the interpretation of the output vectors of a network. Outputs in the alarm set are to cause error-alarms to be triggered. Outputs in an M-glass are all to be interpreted identically by subsequent circuitry. A network then is GSC for an input code, fault set, and an output partition.

Definition:

A network is generalized self-checking (GSC) for input code I, fault set $\mathcal{F}$ and output partition $Q = Z||E_1|E_2|...|E_n$ iff its behavior under faults in $\mathcal{F}$ satisfies

1) Generalized Self-Testing (GST) Property:

For every $f \in \mathcal{F}$, there exists some $i \in I$ for which $C(i,f) \in Z$, and

2) Generalized Fault-Secure (GFS) Property:

For every $f \in \mathcal{F}$ and every $i \in I$, whenever $C(i,f) \in E_k$, $1 \leq k \leq n$, the correct output $C(i,\lambda) \in E_k$ also,

where $C(i,f)$ gives the output for input i under fault f, and λ is the null fault.

Example: Determine if the network in Fig. 2.1 below is GSC for the following output partition. The input code is the 1-out-of-4 code. The output code is shown in Table 2.1, a partial truth table for the network. Table 2.1 shows the output for each input

codeword, under each fault. The set of faults to be considered is single stuck faults on gate inputs and outputs. The output partition to be considered is $Z||E_1|E_2$, where

E_1 = (1000) (logic vectors $z_3z_2z_1z_0$),

E_2 = (0001, 0010, 0111},

Z = {all other combinations).

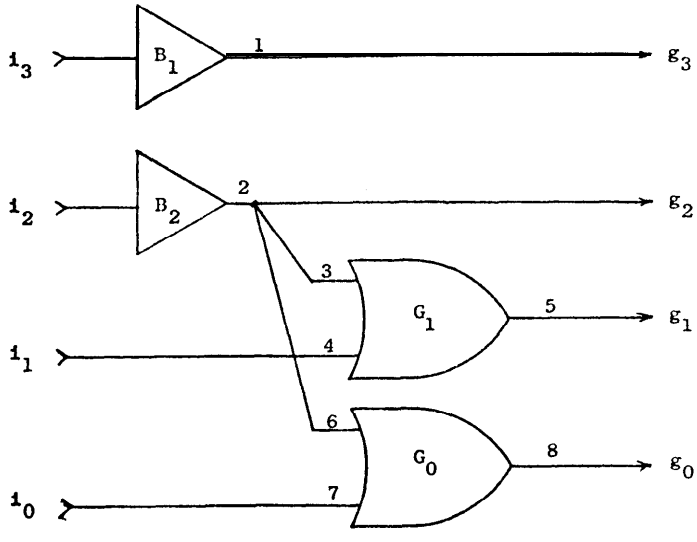

Fig. 2.1 Sample Network

Table 2.1

Sample Network Output under Various Faults

		fault											
$i_3 i_2 i_1 i_0$	no fault	1/0	1/1	2/0	2/1	3/0	4/0	5/0	5/1	6/0	7/0	8/0	8/1
E_1 {1000	1000	0000	1000	1000	1111	1000	1000	1000	1010	1000	1000	1000	1001
E_2 { 0100	0111	0111	1111	0000	0111	0101	0111	0101	0111	0110	0111	0110	0111
E_2 { 0010	0010	0010	1010	0010	0111	0010	0000	0000	0010	0010	0010	0010	0011
E_2 { 0001	0001	0001	1001	0001	0111	0001	0001	0001	0011	0001	0000	0000	0001

note 1: Output vector shown is $g_3 g_2 g_1 g_0$

note 2: Incorrect outputs are underlined.

note 3: Fault heading j/k means wire j stuck at k.

The table represents the output of the network under all faults in the fault set. OR gate inputs stuck at 1 have been omitted from table because response is identical when OR gate output is stuck at 1.

The GST property is satisfied because the alarm set contains at least one underlined output from each column. We verify the GFS property as follows. When the correct output is 1000, incorrect outputs (found underlined in first row of Table 2.1) are all in alarm set. None of them are in E_2. When correct output is in E_2, incorrect outputs (found underlined in last 3 rows of Table 2.1) are all in alarm set or E_2. None of them are in E_1. We conclude that the network is GSC for the input code, fault set, and output partition given.

End of Example

Now since we are using an output partition to describe meanings of the output vectors of a network, we might as well use the same formalism to describe meanings of its input vectors. For a given network and output partition, we will use an input partition to describe the network response when fault-free with reference to a given output partition. It specifies which inputs give output in the alarm set of the output partition, and which cause outputs in the various M-classes.

Definition:

Let the output partition of a network be $Q = Z||E_1|E_2|....|E_n$. Let Y be the set of all input vectors which cause output in Z. Let V_1 V_2, ...,V_n be the sets of input vectors which cause output in E_1,E_2, ...,E_n respectively.

The input partition P of the network is the proper subset Y of the set U of all possible input vectors, and the partitioning of the remainder U-Y into equivalence classes $V_1,V_2,....,V_n$. Y is called the alarm set of P, and $V_1,V_2,....,V_n$ are called M-classes of P. P may be written:

$$P = Y||V_1|V_2|....|V_n.$$

3. DETERMINING OUTPUT PARTITIONS FOR WHICH A GIVEN NETWORK IS GSC

To make use of a GSC network, one must be sure that each fault causes some recognized non-code output. For each fault there is in general a set of non-code outputs which it causes. We shall call these outputs alarm outputs. Each fault then has a set of alarm outputs. To satisfy the GST property, at least one alarm output per fault must be recognized as non-code.

Alarm outputs should not be confused with the alarm set of an output partition. The alarm set is a set of non-code outputs specified in the output partition. An alarm output is not necessarily a member of the alarm set of any output partition. It is however, a candidate for membership in the alarm set.

For each output w, under a particular fault set there is in general a set of output codewords in whose place w may appear. We shall call this set the home class of w. The home class of an output codeword j always contains j, since output j appears under no-fault. It contains a different output codeword x if some fault can change the output from x to w. The home class of a non-code output t never contains t, since t is not an output codeword. It contains an output codeword y if there is some fault which changes y to t. If there is no such output codeword, the home class for t is empty. To satisfy the GFS property, an output w and every member of its home class must have the same meaning. This is required of all w except recognized non-code outputs.

If we know the alarm outputs for each fault, and the home class for every output vector, we can figure out all the assignments of meaning to the output vectors for which the GFS and GST properties are satisfied. There is generally a large number of such assignments.

Example: Determine all output partitions for which the earlier sample network in Fig. 2.1 is GSC. The input code and fault set are the same: l-out-of-4 code and single stuck faults. Table 2.1 gives output of network under the various faults.

The alarm outputs for a fault are the outputs in its column which are underlined and are non-code. At least one alarm output per fault must be included in the alarm set of an output partition, if the network is to be GSC for that output partition. The alarm outputs for the various faults are shown in Table 3.1.

Table 3.1

Alarm Outputs for Various Faults in Sample Network

			f a ult							
1/0	1/1	2/0	2/1	3/0	4/0	5/0	5/1	6/0	7/0	8/0	8/1
	1001		1111								
	1010				0101	1010				0110	1001
0000	1111	0000		0101	0000	0000	0011	0110	0000	0000	0011

The home class for an output vector is composed of the no fault outputs from all rows in Table 2.1 where the output vector appears. The home classes for all output combinations are extracted from Table 2.1 and shown in Table 3.2.

Table 3.2

Home Class for Various Output Vectors in Sample Network

output vector $\langle g_3 g_2 g_1 g_0 \rangle$															
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0001, 0010, 0111; 1000,	0001	0010	0001, 0010	∅	0111	0111	0001, 0010, 0111	00	0001, 1000	0010, 1000	∅	∅	∅	∅	0111, 1000

note 1: ∅ means home class is empty set

To satisfy the GST property, at least one alarm output per fault must be recognized as a non-code output. We must choose from Table 3.1 a set of alarm outputs which covers all columns. 0000, 1111, 0101, and 0110 must be chosen since each is the sole entry in some columns. The remaining columns can be covered by choosing 0011 alone, or 1010 and 1001. Thus we must include one of the following two sets in the alarm set of the output partition:

A1 = {0000, 1111, 0101, 0110, 0011).

A2 = (0000, 1111, 0101, 0110, 1010, 1001).

Turning to the GFS property, the home class members for any output vector which is not in the alarm set must have the same meaning. Output vectors 0000, 1111, 0101, 0110 must be in the alarm set so they need not be considered. output vectors with empty home class need not be considered. Output codewords with only themselves in their home class

need not be considered. The meanings of the remaining output vectors must be consistent with the GFS requirement. After removing columns from Table 3.2 which need no longer be considered, the columns shown in Table 3.3 remain.

Table 3.3

Home Class for Output Vectors which Remain in Question

output vector			
0111	0011	1001	1010
	0001		
0010 0001	0010	1000 0001	1000 0010
0111			

Since 0111 is a codeword, the requirement that members of its home class have the same meaning cannot be avoided by placing it in the alarm set. If that were done, error alarms would be triggered almost immediately when the network is put in operation! Then 0001, 0010, 0011, and 0111 must have the same meaning. That completes consideration of home class for outputs 0111 and 0011. There remain only the last two columns in Table 3.4 to be considered.

If A2 is not included in the alarm set, either or both of the outputs 1001 or 1010 is not recognized as a non-code output, and so output 1000 must have the same meaning as the other 3 output codewords. Then there can be only

a single M-class in the output partition. Only if A2 is included can 1000 have a different meaning. Thus there are 2 groups of output partitions for which the network is GSC:

1. Two M-classes: 0001, 0010, 0111 contained in M-class E_1. 1000 contained in M-class E_2. A2 contained in alarm set. Remaining output vectors can be put in alarm set or either M-class.

   $Q = Z||E_1|E_2$: 3^6 choices

2. One M-class: 0001, 0010, 0111, 1000 contained in single M-class E_1. Either Al or A2 is contained in alarm set. Remaining output vectors can be put in alarm set or the M-class.

   $Q = Z||E_1$: 125 choices

In conclusion, consideration of alarm outputs and home classes determines all output partitions for which a network is GSC, in a straight-forward fashion. The result can be very complex to state, and often involves a large number of partitions. However in a later section, we will derive GFS and GST preserving transformations which allow simpler representation for large numbers of partitions.

4. SUBNETWORKS OF A TSC PARENT NETWORK

The TSC network has been generalized to a GSC network. The GSC network includes TSC and SCST networks as special cases. One might suspect that some structural regularity in TSC networks could be found by considering the GSC network. Indeed, we will show in this section that an acyclic TSC network is always a cascade of GSC subnetworks.

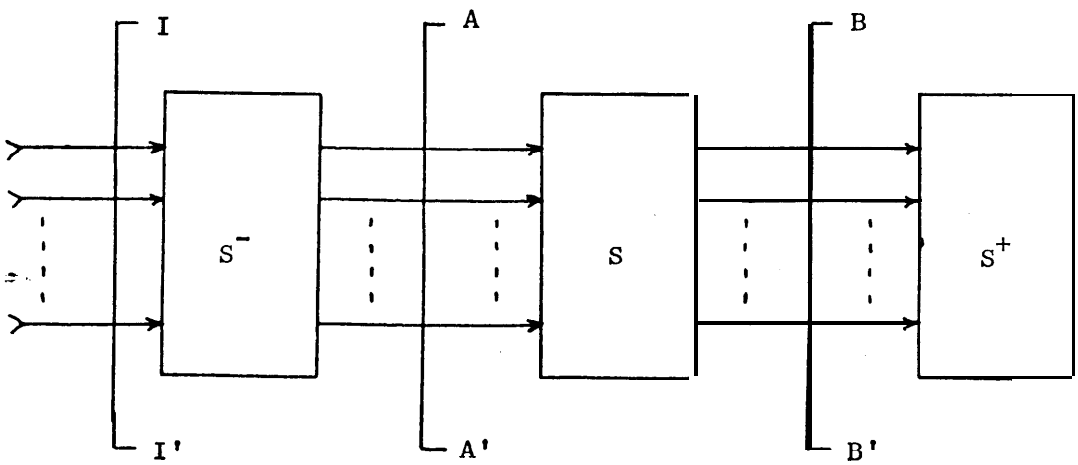

Fig. 4.1 TSC network showing a typical complete subnetwork.

Consider an acyclic TSC network and any complete subnetwork S, as in Fig. 4.1. Consider faults which are located entirely within S, faults which are local to S. S^- is fault-free. When the input codewords are applied to the parent network at I-I', a set of logic vectors appears at A-A' which we will call the local input code of S. While the inputs of S are held within the local input code, the ST and FS properties must be satisfied for the faults local to S.

The ST property holds, so each local fault causes a non-code output at G-G'. Then each fault causes an output at B-B' which means that a non-code output will appear at G-G'. The FS property holds, so no local fault causes an incorrect output codeword to appear at G-G'. Then no local fault changes the output at B-B' to a vector which means that an incorrect output codeword will appear at G-G'. Consider the B-B' vectors which cause non-code output at G-G' to be recognized non-code outputs of s. Let the remaining vectors have meanings according to the output codewords they cause at G-G'. Then S clearly satisfies the GFS and GST properties.

Subnetworks of a GSC parent network behave much the same. Consider a GSC network with output partition Q_b and input code I_a cut into a cascade of 2 subnetworks A and B as in Fig. 4.2.

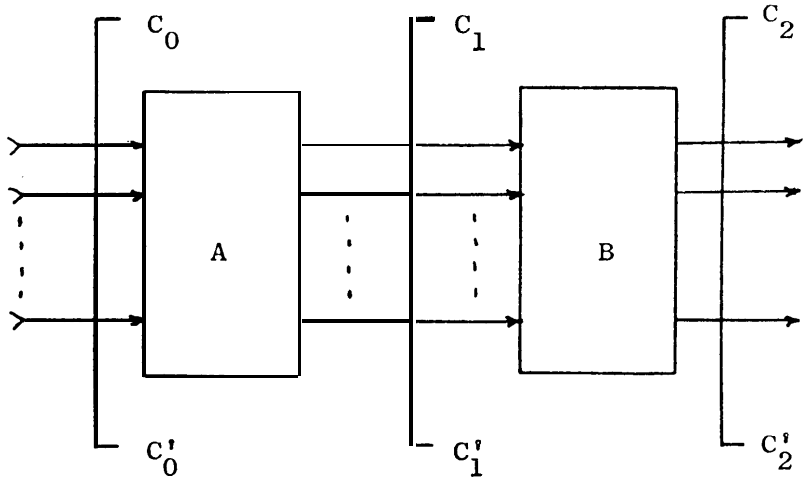

Fig. 4.2 GSC network cut into a cascade of 2.

From I_a derive the output code Z_a of A and the local input code I_b of B. Clearly, $Z_a = I_b$. From Q_b derive the input partition P_b of B. P_b describes how the output at C_2-C_2' responds to output vectors from A.

Since the GST property holds, there must be an output in the alarm set of Q_b for each fault in A. Then there is an output of A in the alarm set of P_b. Since the GFS property holds, no fault in A can cause the output to move from one M-class of P_b to another. Then no fault in A can cause the output of A to move between M-classes of P_b. Then A is GSC for input code I_a, its local faults, and an output partition equal to P_b. B is trivially GSC for the input code I_b, its local faults, and the output partition Q_b. Thus both subnetworks of GSC parent network are themselves GSC.

Now suppose A is GSC for input code I_a and output partition Q_a. Suppose the output code of A is Z_a. Suppose G is GSC for input code I_b and output partition Q_b. Let the input partition of B be P_b. Suppose $Z_a = I_b$ and $Q_a = P_b$, thus re-creating the exact conditions found in the network above. Is the cascade A:B GSC?

Since A has the GST property, each fault in A causes an output in the alarm set of Q_a. Since $Q_a = P_b$, that in turn causes an ouput in the alarm set of Q_b. Since A has the GFS property, no fault moves the output of A between M-classes of Q_a. Then the input of B stays within an M-class of P_b so the output of C_2-C_2 stays

within an M-class of Q_b. Thus both GST and GFS properties hold for the cascade, for faults in A. Since B is given to be GSC and its proper input code is supplied by A, the GST and GFS properties hold for the cascade for faults in B. Than A:B must be GSC for the input code I_a and output partition Q_b. This proves Theorem 1.

Theorem 1:

Given 2 GSC networks A and B. A is GSC for the input code I_a fault set $\mathcal{F}_a$ and output partition Q_a. B is GSC for input code I_b fault set $\mathcal{F}_b$ and output partition Q_b.

The cascade A:B is GSC for the input code I_a fault set $\mathcal{F}_a \cup \mathcal{F}_b$ and output partition Q_b if and only if

1) The output code of A is equal to the input code of B (I_b), and
2) The input partition of B is equal to the output partition of A (Q_a).

This is quite a powerful result. By recursive application, it applies to cascades of any number of subnetworks. If a GSC network is cut into n subnetworks, all n are GSC. If a network is a cascade of n subnetworks, it is GSC if all n subnetworks are GSC and they match input and output codes, and input and output partitions.

This result can be extended to apply to TSC networks. In the introduction we showed that a network is TSC if it is GSC with all non-code outputs being recognized as non-code, and a different meaning for each output codeword. These conditions are described by an output partition with each M-class containing a single member which is an output codeword. We shall call this kind of output partition a singleton output partition. For a given output code, there is only one singleton output partition. Then a network is TSC if it is GSC for the singleton output partition. The reverse is clearly true.

Theorem 2:

A network is TSC for input code I and fault set $\mathcal{F}$ if and only if it is GSC for input code I, fault set $\mathcal{F}$ and the singleton output partition.

Thus a network is TSC if and only if it is a cascade of GSC subnetworks matching codes and partitions per Theorem 1, and the output partition of the last subnetwork is a singleton output partition. We shall exploit these results in the following sections.

5. FAMILIES OF OUTPUT PARTITIONS

We have seen that a network can be GSC for a single input code, a single fault set, and many, many output partitions. Given a network, input code, and fault set, we will call the output partitions, for which the network is GSC, good output partitions. In this section we will show some useful structure in the set of all good output partitions.

We will show that good output partitions naturally fall into families. Each family has at its head a unique ancestor partition. A simple test shows if a given output partition Q is in the family of a given ancestor. If the test is made over all ancestors and Q is in some family, Q is good. Otherwise Q is not good. Since the set of all ancestors is usually much smaller than the set of all good output partitions, it is a highly compact way to represent the set of all good output partitions.

We begin by defining 3 operations on an output partition, which preserve the GST and GFS properties. Using the operations merging and reduction we will derive all ancestor partitions for a given network. Using the operation expansion, we will generate all good output partitions from the set of all ancestor partitions. Merging, as we shall presently see, applies to M-classes, while expansion and reduction apply to the alarm set.

Merging

Suppose we have a network T which is GSC for the input code I, fault set $\mathcal{F}$, and output partition $Q = X||E_1|E_2|....|E_n$. Suppose

we-merge together (i.e. form the set union of) two M-classes E_i and E_j, $i \neq j$, forming an output partition $Q' = X||V_1|V_2|....|V_n$ 1. Clearly the GST property is satisfied for output partition Q', since X is unchanged. The GFS property is preserved also; since no output codeword can be moved to E_k from E_i or E_j individually by faults in $(k \neq j \neq i)$, no output codeword can be moved out of $E_i \; E_j$ (set union).

Definition:

An output partition $Q' = X||V_1|V_2|....|V_n$ 1 is a merging of an output partition $Q = X||E_1|E_2|...|E_n$ iffthereexists some onto mapping R from the set $\{1,2,...,n\}$ onto the set $\{1,2,...,n-1\}$ such that $E_i \subseteq V_{R(i)}$ for all $1 \leq i \leq n$.

Lemma 1:

Given that a network is GSC for the input code I, fault set $\mathcal{F}$ and output partition Q. Then T is GSC for the input code I, fault set $\mathcal{F}$, and any output partition Q' which is a merging of Q.

Expansion:

Now suppose we have a network T which is GSC for an output partition $Q = X||E_1|E_2|...|E_m$. Suppose we remove an element v from one of the M-classes E_i and place it in the alarm set. If v is not an output codeword of T, the result is also an output partition. We will call this operation expansion, referring

to the expansion of the alarm set.

Definition:

An output partition $Q' = X||E'_1|E'_2|....|E'_m$ of a network T with output code Z is an immediate expansion of an output partition $Q = X||E_1|E_2|....|E_m$ iff for some $k, k \in \{1,2,...,m\}$, there exists $v \in E_{k'}$ $v \notin Z$, where

$X' = X + v$ (addition of element to set),

$E'_k = E_k - v$ (removal of element from set), and

$E'_j = E_j$ for all $j \neq k$

v is called the exchange element of the expansion, and E_k is called the exchange class.

Definition:

An output partition Q_2 is an expansion of an output partition Q_1 iff Q_2 can be obtained from Ql by a sequence of immediate expansions.

Expansion preserves the GSC nature of a network. Consider a network which is GSC for an output partition Q. Consider Q', an immediate expansion of Q. Because the alarm set of Q' contains the alarm set of Q, the GST property is satisfied for Q' also. Further, because every M-class of Q' is an improper subset of some M-class of Q, the GFS property of Q implies the GFS property for Q'. Then the network is GSC for Q'. By repeated application we have:

Lemma 2:

Given that a network T is GSC for the input code I, fault set $\mathcal{F}$, and output partition Q. Then T is GSC for the input code I, fault set $\mathcal{F}$, and any output partition Q' which is an expansion of Q.

Reduction:

Next we define the reverse of the expansion operation, called reduction. Suppose we have a network which is GSC for an output partition Q. Let us select an element v of the alarm set, where the home class of v is contained in one M-class M. If v is not the sole alarm output for a fault, we can move v into M without violating the GST or GFS property. GST preservation is trivial because v is not the only alarm output for any fault. GFS property is preserved because no 2 members of v's home class are in different M-classes - all are within M. This argument should seem familiar, since it is the criterion we used in section 3 to determine all good output partitions.

Definition:

An output partition $Q' = X'||E_1'|E_2'|....|E_m'$ is an immediate reduction of an output partition $Q = X||E_1|E_2|....|E_m$ iff for some $v \in X$,

$X' = X - v$ (removal of element from set),

$E_k' = E_k + v$ (Addition of element to set), and

$E_j' = E_j$ for all $j \neq k$.

v is called the exchange element of the reduction and E_k is called the exchange class.

Definition:

An output partition Q_2 is a reduction of an output partition Ql iff Q_2 can be obtained from Q_1 by a sequence of immediate reductions.

Clearly if an output partition Q' is an immediate reduction of Q, then Q is an immediate expansion of Q'. Whatever is done to an output partition by reduction can be undone by expansion, and vice-versa.

Lemma 3:

Given that a network T is GSC for the input code I, fault set $\mathcal{F}$, and output partition Q.

If Q' is an immediate reduction of Q, for which v is the exchange element, and M is the exchange class, and

1) v is not the only alarm output for any fault in $\mathcal{F}$,

2) the home class of v is contained in M,

then T is GSC for the input code I, fault set $\mathcal{F}$, and output partition Q'.

Immediate reduction can be applied repeatedly to the output partition of a GSC network, generating output partitions with smaller and smaller alarm set. Unlike the expansion operation, reduction

cannot in general be carried out to its extreme, an empty alarm set, while preserving the GFS and GST properties. At some point short of that, it will be found that each of the alarm set members is either the sole alarm output for a fault or has a home class overlapping more than one M-class. No further reduction preserves the GSC properties.

The set of all ancestor partitions for a given network, input code, and fault set can be found as follows. Generate the singleton output partition and all mergings of it. Discard the output partitions which are not good. For each remaining partition, apply immediate reduction in all possible ways. Where each chain of reductions ends, the last output partition is an ancestor partition.

From the set of ancestors, the set of all good output partitions can be found by forming all possible expansions of the ancestors. Once the set of ancestor partitions is known, one can determine whether an output partition is good by a simple test. It is good iff it is an expansion of one of the ancestor partitions. An output partition $Q_1 = A||B_1|B_2|....|B_n$ is an expansion of $Q_2 = X||Y_1|Y_2|...|Y_n$ iff X is contained in A, and each B_k is contained in some Y_j, $1<k\leq n$, $1<j\leq n$. This simple test will be very useful in the method of synthesis which follows in the next section.

6. CASCADING GSC NETWORKS

Theorem 1 and Lemmas 1–3 are a sufficiently powerful set of formal tools for us to build GSC networks by cascading. In this section, we will show how they are used, The situation in cascading is shown in Fig. 6.1. We have a network A which is GSC for a particular input code I_a, fault set $\mathcal{F}_a$, (composed entirely of faults within A), and a set of ancestor partitions. We have a network B which is GSC for a particular input code I_b, fault set $\mathcal{F}_b$ (composed entirely of faults within B), and an output partition Q_b. We now have sufficient tools to determine if A:B is GSC.

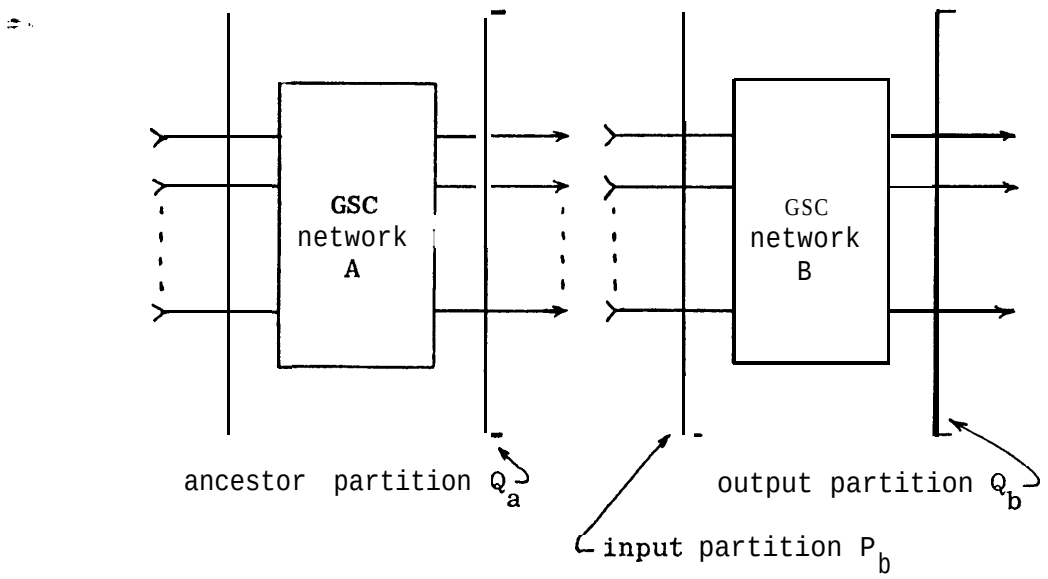

Fig. 6.1 Cascading GSC networks

First, we check that the output code of A is identical to the input code of B. If so, A:B may be GSC, and we proceed further. If not, B must be re-examined with the output code of A as its input code, and any good output partitions Q_b found.

Second, we form the input partition P_b of B, from Q_b and the

fault-free truth table of B. If P_b is an expansion of one of the ancestor partitions Q_a of A, A:B is GSC. Q_a can be expanded until it is equal to P_b, and A will still be GSC. Then the output partition of A is equal to the input partition of B, and the conditions of Theorem 1 are fulfilled.

Using this cascading step, it is possible to build a desired GSC network by repetition. One can keep on suffixing networks, as we suffixed B to A, until the desired network is formed. If the initial network A is correct, and the proper suffix networks are added in the proper order, any GSC network can be built up in this way. It may therefore be possible to develop a synthesis algorithm for GSC networks, based on the cascading tools given here.

An example where luck and persistence have allowed us to construct a new GSC network by cascading, is given next. The example is intended to show use of the cascading rules, and to motivate further research on a synthesis algorithm. But first we must make a small elaboration of the cascading rules, to make the example easier.

In cascading, it may be specified that a certain subset L of the non-code inputs to A (Fig. 6.1), shall cause alarm set output from the cascade. This happens, for example, when the cascade is to be a GSC checker. Each member of L causes some output v from A, which must be in the alarm set of the particular output partition of A, found in the finished cascade. The response of a network to non-code inputs, however, has nothing to do with its GSC properties. Constraining a GSC network's response to non-code

inputs simply creates a special case of the GSC network.

We will extend the definition of a good output partition to cover this constraint. A good output partition becomes one for which the network is GSC, and an alarm set output is given for each of the members of L. The procedure for finding ancestor partitions for A is modified accordingly. For each member of L, the output of A (when fault-free) should be in the alarm set of every good output partition. In the reduction process leading to the ancestor partitions, such outputs must not be removed from the alarm set. Then the ancestors describe the set of all output partitions for which A is GSC, and for which R-gives alarm set output for the specified non-code inputs.

Example: We wish to design a GSC 2 input Morphic OR network. An n-input Morphic OR [Carter, 1971] network has $2n$ inputs, divided into n groups of 2 wires. Each group is called a pair. The input code is all combinations of complementary signals on the pairs. Since there are 2 ways a pair can carry complementary signals, the input code as 2^n members. Here with $n=2$, the input code has 4 members.

The n-input Morphic OR has 2 outputs. The output code is the two complementary combinations 01 and 10. The function of the Morphic OR when fault-free is to give a non-code output when all n input pairs are non-complementary. When one or more pairs are complementary,

the output is complementary. Thus the network we are to design is a GSC network with a heavily constrained response to non-code inputs. The application of the 2 input Morphic OR is to observe 2 self-testing checkers which normally give complementary outputs. The Morphic OR senses if both checkers simultaneously give the abnormal, non-complementary output.

Table 6.1

Morphic OR Truth Table

	input $i_3 i_2 i_1 i_0$	encoded output	
input code	0101	01 or 10	both 01 and 10 must appear
	0110	01 or 10	
	1001	01 or 10	
	1010	01 or 10	
	0100	01 or 10	
	0111	01 or 10	
	1000	01 or 10	
	1011	01 or 10	
	0001	01 or 10	
	1101	01 or 10	
	0010	01 or 10	
	1110	01 or 10	
	0000	00 or 11	
	0011	00 or 11	
	1100	00 or 11	
	1111	00 or 11	

In designing a GSC Morphic OR we must realize the GST property and the GFS property. In the function of the Morphic OR, both output codewords bear the same information, namely that the 2 checkers being observed are not both emitting non-code outputs, and no internal fault is currently in evidence within the Morphic OR. Because both bear the same message, it does no harm if the Morphic OR confuses the two output codewords under internal faults. We shall permit faults to change one output codeword to the other, making the design easier and no less useful. To do this we will require the following output partition on the Morphic OR output: {00, 11}||{01, 10} (single M-class containing 01 and 10, alarm set containing all other combinations). Placement of both non-code outputs in the null alarm set is automatic because this is a partition at the network output, and all observers by convention interpret non-code outputs as cause for error alarm. The network is much easier to design in this case, because of the single M-class; the GFS property cannot possibly be violated.

We will design the network for the fault set of all single stuck-faults on gate inputs and outputs. This is the choice made in [Carter, 1968], and is reasonable to deal with.

The input-output function desired is highly constrained, as can be seen in Table 6.1. The Morphic OR gives non-code output for the 4 combinations at the bottom of Table 6.1. These 4 combinations must cause alarm set output in our realization. The remaining 12 inputs should cause output (codewords) in the M-class. These constraints can be expressed as an input partition for the realization. Then the GSC network we desire has the following properties:

1) Input code: {0101, 0110, 1001, 1010)

2) Output code: {01, 10}

3) Fault set: all single stuck faults

4) Output partition: {00, 11}||{01, 10}

5) Input partition:

Alarm set: {0000, 0011, 1100, 1111)

M-class: $\left\{\begin{matrix} 0001, & 0010, & 0100, & 0101, & 0110, & 1111, \\ 1000, & 1001, & 1010, & 1011, & 1101, & 1110 \end{matrix}\right\}$

We begin the design with the hypothesis that we can transform the 2 rail input code into the l-out-of-4 code with a single level of AND gates, which is GSC and has the desired input partition. The single level AND network is particularly easy to work with, because it can be represented by a "set of implicants (one for each AND gate output) on a single Karnaugh map. A map showing the required response

is shown in Fig. 6.1. The realization, if it exists, will be called "subnetwork A."

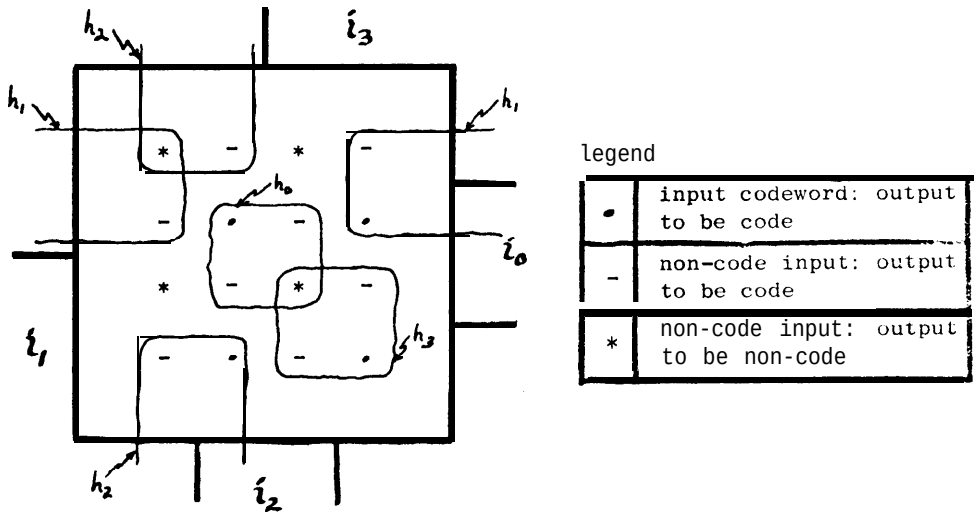

Fig. 6.2 Karnaugh map of subnetwork A.

Each cell on the map is marked with a dot ".", a star "*", or a dash "-". A cell marked with a dot is one of the 4 input codewords. A cell marked with a star is one of the 4 inputs for which the Morphic OR must give a non-code output. Each cell marked with a dash is a non-code input for which the Morphic OR must give a code output.

The single level network hypothesized must give a 1-out-of-4 output for each cell marked with a dot or dash, and give a non l-out-of-4 output for each cell marked with a star. We have assumed that the network can be realized

with 4 AND gates, one for each output. Each gate can then be represented by an implicant on a Karnaugh map. A realization of the desired function will have 4 implicants on the map, one for each gate, and every cell marked with a dot or dash will be included in exactly one implicant, and every cell marked with a star will be included in zero or more than 1 implicants.

A few trials yield the 4 implicants already drawn on Fig. 6.2, $h_3 = i_3 i_1$, $h_2 = \bar{i}_3 \bar{i}_0$, $h_1 = \bar{i}_2 \bar{i}_1$, $h_0 = i_2 i_0$. Two of the cells marked with a star are included in 2 implicants, and the others in no implicants. All other cells on the map are included in one implicant. From the map we can check if the realization is tested for all single faults by the 4 input codewords, by considering growth and existence tests [Paige, 1969]. Each implicant contains one input codeword, so each AND gate receives the 11 input (existence test). Each growth of each implicant includes a second input codeword, so each input stuck-at-1 fault on an AND gate input causes a 2-out-of-4 output for that second input (growth test). Then the network which realizes these implicants, shown in Fig. 6.3, satisfies the GST property and is GSC.

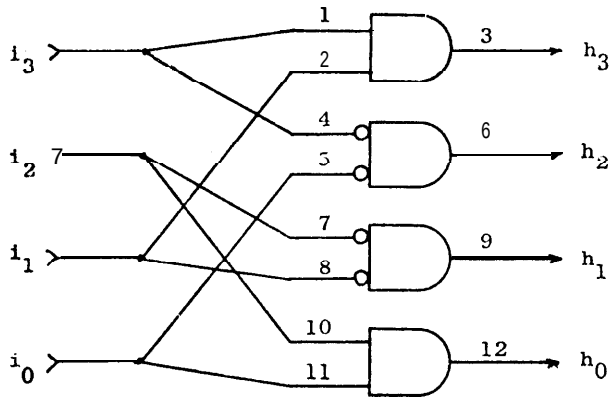

Fig. 6.3 Realization of subnetwork A.

Next we determine the set of all good output partitions for this network. First we consider the input-output function of the Morphic OR. The last 4 input combinations in Table 6.1 give outputs $h_3h_2h_1h_0$= 0000, 0110, and 1001. The Morphic OR output is to be non-code for these input combinations, so 0000, 0110, and 1001 must be in the alarm set of-the network (Fig. 6.3). Code output for the first 12 input combinations is already guaranteed because they give l-out-of-4 output at h_3-h_0. The final Morphic OR will have only one M-class, so we can restrict to good output partitions with one M-class. Ancestor partitions subject to these restrictions must now be found.

Having a single M-class simplifies getting ancestor partitions, because the GFS property is trivial to satisfy.

An output partition is a (good) ancestor partition if its alarm set contains 0000, 0110, and 1001, and no other member of its alarm set can be removed without violating the GST property. The set of alarm outputs for each fault is shown in Table 6.2

Table 6.2

Alarm Outputs for Various Faults in Subnetwork A

fault															
1/1	2/1	3/0	3/1	4/0	5/0	6/0	6/1	7/0	8/0	9/0	9/1	10/1	11/1	12/0	12/1
1100	1010	0000	1001	1100	0101	0000	0101	0011	1010	0000	0011	0011	0101	0000	0101
			1010				0110				0110				0011
			1100				1100				1010				1001

In Table 6.2 note that we have omitted gate input stuck faults which have the same effect as one of the gate output stuck faults. The outputs 0000, 0110, and 1001 must be included in the alarm set to satisfy the constraint on response to non-code inputs. The outputs 1100, 1010, 0101, and 0011 must be included because each is the sole alarm output for one or more faults. This appears as a column in Table 6.2 with only a single entry. We can remove from Table 6.2 all columns covered by these outputs. If any columns them remain in the table, any minimal cover, when added to the alarm set, forms an ancestor partition.

Remove the columns containing 0000, 0110, or 1001, and all columns with only a single entry. No columns remain. For compactness, let us denote output vectors $h_3h_2h_1h_0$ by their decimal equivalents,

$$\sum_{0\leq k\leq 3} h_k * 2^k .$$

Outputs 0, 6, and 9 are required in the alarm set to give the required input-output response. Additional outputs 3,5, 10 and 12 complete the GST property. Any other outputs may be removed from the alarm set. Then the partition {0, 3, 5, 6, 9, 10, 12}||{1, 2, 4, 7, 8, 11, 13, 14, 15} is the only ancestor partition.

Table 6.3

Encodings for Subnetwork A

codes		alarm sets	
input code	5,6,9,10	to detect faults	0,3,5,10,12
output code	1,2,4,8	to catch bad inputs . . .	0,6,9
		to detect faults and bad inputs	0,3,5,6,9,10,12

At this point, we have successfully transformed the 2 rail input code to the l-out-of-4 code. The realization in Fig. 6.3 is GSC for the input code {5,6,9,10}, the set of

all single stuck faults, and the ancestor partition $\{0,3,5,6,9,10,12\}||\{1,2,4,7,8,11,13,14,15\}$.

We now hypothesize another l-level AND network with the $\{1,2,4,8\}$ input code, whose input partition has an alarm set which contains $\{0,3,5,6,9,10,12\}$.

A Karnaugh map showing these requirements is shown in Fig. 6.4. If a realization exists we will call it "subnetwork B."

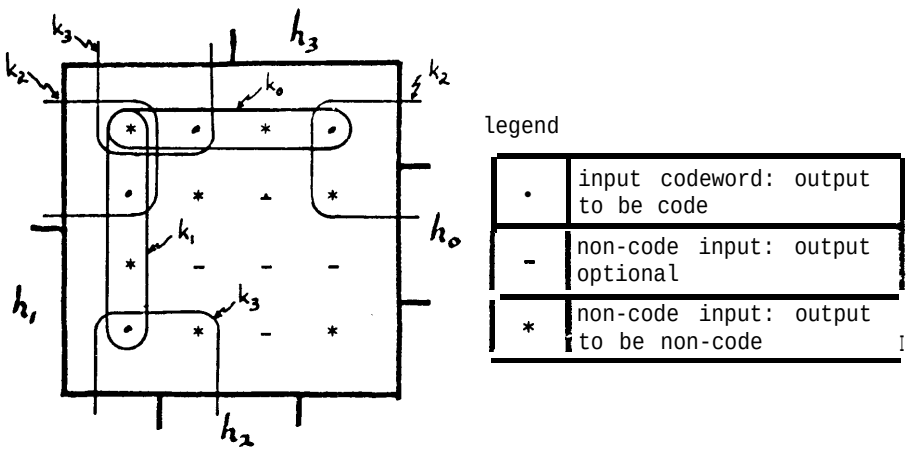

Fig. 6.4 Karnaugh map of subnetwork B

A few trials yield the realization in Fig. 6.5 whose implicants have been drawn already on be Karnaugh map. The output code of this network is a subset of the 2-out-of-4 code. The Karnaugh map again shows growth and existence tests for every implicant, each test giving

a non-code output. We again solve a covering problem on the alarm outputs (Table 6.4) to find the ancestor partitions at the output. As can be seen the solution to this covering problem is unique, and is shown in Table 6.5; there is only one ancestor partition.

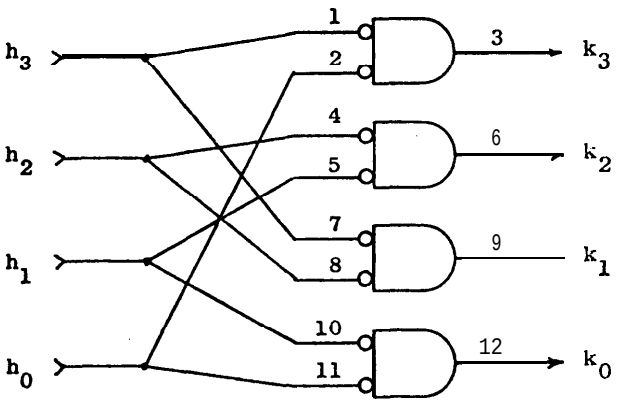

Fig. 6.5 Realization of subnetwork B

Table 6.4

Alarm Outputs For Various Faults in Subnetwork B

fault															
1/0	2/0	3/0	3/1	4/0	5/0	6/0	6/1	7/0	8/0	9/0	9/1	10/1	11/0	12/0	12/1
1101	1110	0001 0010	1101 1110	1101	1110	0001 0010	1101 1110	0111	1011	1000 0100	0111 1011.	1011	0111	0100 1000	1011 0111

Table 6.5

Encodings of l-out-of-4 to 2-out-of-4 Converter

codes	alarm sets
input code . . .{1,2,4,8}	to detect faults . . .{1,4,7,11,13,14}, {1,7,8,11,13,14}, {2,4,7,11,13,14}, {2,7,8,11,13,14}.
output code . . .{5,6,9,10}	to catch bad inputs . . . {0,1,2,4,8,15}
	to detect faults and bad inputs . . {0,1,2,4,7,8,11,13,14,15}

Thus subnetwork B in Fig. 6.5 is GSC for the input code {1,2,4,8}, all single faults, and the ancestor partition {0,1,2,4,7,8,11,13,14,15}|| {3,5,6,9,10,12}. This ancestor partition is formed directly from the covering problem solution shown in Table 6.5. We will attempt to suffix subnetwork B to subnetwork A. Because B is to be suffixed to A, we want B to offer the largest possible alarm set at its input. So we expand ancestor output partition of B to the utmost, {0,1,2,3,4,7,8,11,12,13,14,15}|| {5,6,9,10}, moving all non-code outputs into the alarm set. We shall attempt to cascade B, with this expanded output partition, to A. The input partition of B then is

P_b = {0,3,5,6,7,9,10,11,12,13,14,15}||{1,2,4,8}.

Happily, P_b is an expansion of A's ancestor partition $\{0,3,5,6,9,10,12\}||\{1,2,4,7,8,11,13,14,15\}$. Then we can expand the ancestor partition of A to make it match the input partition offered by B. A is still GSC. Then, since both the codes and partitions match, the cascade is GSC by Theorem 1. It is GSC for the input code $\{5,6,9,10\}$, all single stuck faults, and the output partition $\{0,1,2,3,4,7,8,11,12,13,14,15\}||\{5,6,9,10\}$.

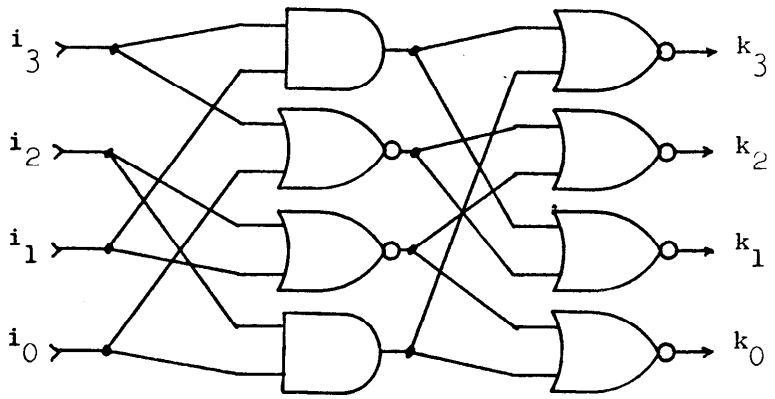

Fig. 6.6 Partially complete cascade.

The cascade A:B is shown in Fig. 6.6, where we have changed the NOT-AND gates in both subnetworks into OR-NOT gates, following DeMorgan's Law. This change does not affect any of the properties derived for these networks, because the fault model we use supposes stuck-faults on the output wire and input wires of any gate. The faults can be thought to be

on the wires, rather than within the gates. Then we are free to replace a gate with an equivalent one, so long as the number of input wires is unchanged.

Now the next step in the cascading procedure is to find the ancestor partitions for the cascade, and then seek a new subnetwork to put on its right. But we don't need to go to that trouble, because an answer already exists. The output code from the cascade, {5,6,9,10}, is the same 2 rail code we began with. The remainder of the network must accept this as its input code, and may simply offer an alarm set equal to all non-code inputs, namely {0,1,2,3,4,7,8,11,12,13,14,15}.

The Morphic AND, or "MAND", described in [3] has just these features. It has the 2 rail output code {1,2} which we wish to reach. It is GSC for the input code {5,6,9,10}, all single stuck faults and the output partition {0,3}||{1,2}. For this output partition, its input partition is {0,1,2,3,4,7,8,11,12,13,14,15}||{5,6,9,10}. Then using Theorem 1 we suffix the MAND to the cascade, and obtain the desired GSC Morphic OR. The complete cascade is shown in Fig. 6.7. The last 2 logic levels, an AND-OK combination, constitute the MAND which was suffixed. Some internal signals for this cascade are shown in Table 6.6.

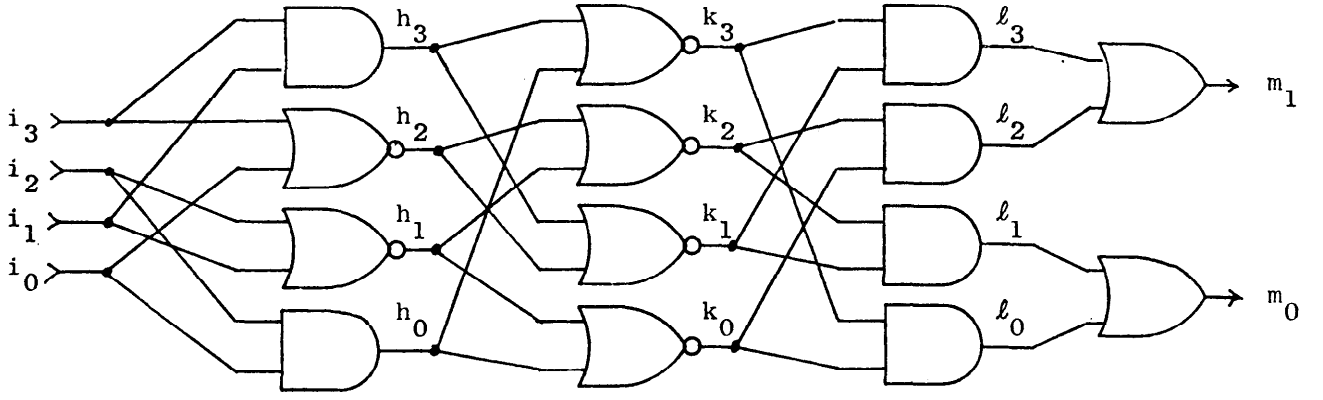

Fig. 6.7 Morphic OR obtained by cascading.

Table 6.5

Internal Signals of Morphic OR When Fault-Free

$i_3i_2i_1i_0$	$h_3h_2h_1h_0$	$k_3k_2k_1k_0$	$\ell_3\ell_2\ell_1\ell_0$	m_1m_0
0101	0001	0110	0010	01
0110	0100	1001	0001	01
1001	0010	1010	1000	10
1010	1000	0101	0100	10
0100	0100	1001	0001	01
0111	0001	0110	0010	01
1000	0010	1010	1000	10
1011	1000	0101	0100	10
0001	0010	1010	1000	10
0010	0100	1001	0001	01
1101	0001	0110	0010	01
1110	1000	0101	0100	10
0000	0110	1000	0000	00
0011	0000	1111	l-111	11
1100	0000	1111	1111	11
1111	1001	0100	0000	00

The Morphic OR reported in [3] is shown for comparison in Fig. 6.8. The two realizations visibly differ in fanouts, level count, and gate count.

End of Example

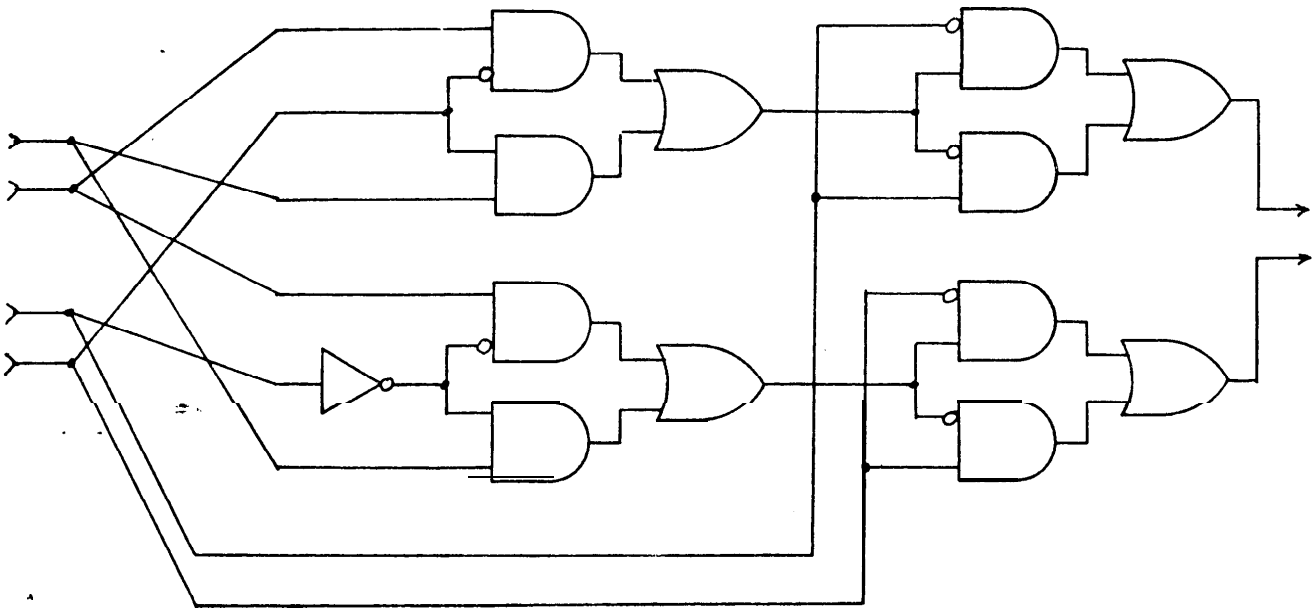

Fig. 6.8 Morphic OR given by Carter.

The preceding example has shown the synthesis of a GSC network. The same method can be used to get a TSC network, since TSC is a special case of GSC. Referring to Theorem 2, a TSC network is a GSC network whose output partition is composed of singleton M-classes. Then to synthesize a TSC network, one should stipulate an output partition with as many M-classes as there are output codewords. When the design is finished, expansion (Lemma 2) can be used to remove all non-code outputs from the

various M-classes, leaving a single codeword in each. Then the output partition would become a singleton output partition, and so the network would be TSC.

It follows from Theorems 1 and 2 that any TSC network which exists has a realization which is a cascade of GSC networks. Those networks will successfully cascade together by the method given here and reconstitute the original network. We have here a cascading technique with perfect generality, but we need a constructive synthesis algorithm.

In the example, the first 2 subnetworks were obtained by a combination of intuition and trial-and-error persistence. These efforts started always with a working hypothesis of a single-level GSC network with a particular output code. The hypothesis may have been false. There exist input code-output code pairs, for which no l-level GSC AND network realization exists. Had one of these pairs been hypothesized, no solution would have been found and the example would have failed.

To have a synthesis algorithm based on cascading, we need an algorithm to produce the GSC subnetworks used. Perhaps it may be possible to break down a desired input-output function into a canonical sequence of code transformations, each of which is realizable by a GSC network. It should certainly be possible to do this for some well-chosen class of transformations between input and output codes.

An attempt at this has been made in [5]. Input and output codes were restricted to have Hamming distance 2 or greater, and only 4 or fewer lines. Single level AND networks were chosen, and all code transformations having single level AND realizations were found by exhaustive analysis. Only 48 non-trivial realizations existed. By cascading them, quite a few code transformations are possible, but the number was judged to small to warrant further work.

This is a complex problem, and a good topic for further investigation. There are a great many other ways to approach it. It is hoped that the cascading technique given here will provide the encouragement for further work, and that ultimately, a synthesis algorithm for TSC and GSC networks will be found.

7. SUMMARY

In this paper we have seen 2 major results. First, practical considerations led us to define the GSC network, a generalization of the TSC network. We found that the self-checking and self-testing (SCST) network as well as the TSC network is a special case of the GSC network.

Second, we found very pronounced cascade structure in combinational acyclic TSC, SCST, and GSC networks. When such a network is divided into a series cascade, every piece of the cascade is GSC. This establishes a much stronger relation between the TSC and SCST networks than was ever suspected. We found conditions permitting a GSC network to be built up by cascading smaller GSC networks.

With this cascading technique it is possible to build up a desired TSC, SCST, or GSC network with specific input-output response by cascading, through a risky process of trial-and-error. In an example we used this approach and derived an alternate realization of the Morphic OR network [3]. The technique we used strongly suggests that a canonical code transformation could be found, which would eliminate most of the trial-and-error risk from synthesis by cascading. If such a transformation is found, cascading synthesis of TSC networks will become practical, and TSC networks will become a much more significant part of fault-tolerant computing.

8. BIBLIOGRAPHY

[1] D. A. Anderson, "Design of totally self-checking circuits for m-out-of-n codes," IEEE Trans. Comp., Vol. C-22, pp. 263-269, March 1973.

[2] W. C. Carter, P. R. Schneider, "Design of dynamically checked computers," IFIPS 68.

[3] W. C. Carter, A. B. Wadia, D. C. Jessup, "Implementation of checkable acyclic automata by morphic boolean functions," IBM Report RC-3324, April 1971.

[4] M. R. Paige, "Generation of diagnostic tests using prime implicants," University of Illinois, Co-ordinated Science Laboratory, Report R-414, May 1969.

[5] S. G. Kolupaev, "Cutting planes and self-checking networks," Ph.D. thesis, Stanford University, Department of Electrical Engineering, June 1976.