REPORT R-693 OCTOBER,1975 UILU-ENG 75-2228

COORDINATED SCIENCE LABORATORY

GENERAL DESIGN RULES FOR THE CONSTRUCTION OF m-OUT-OF-n TOTALLY SELF-CHECKING CHECKERS

JAMES E. SMITH
GERNOT METZE

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) GENERAL DESIGN RULES FOR THE CONSTRUCTION OF m-OUT-OF-n TOTALLY SELF-CHECKING CHECKERS		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER R-693; UILU-ENG 75-2228
7. AUTHOR(s) James E. Smith and Gernot Metze		8. CONTRACT OR GRANT NUMBER(s) DAAB-07-72-C-0259
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program		12. REPORT DATE October, 1975
		13. NUMBER OF PAGES 27
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Totally Self-Checking Checkers
m-out-of-n Codes
Unidirectional Faults

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report presents a set of conditions sufficient to characterize a class of realizations of m-out-of-n code totally self-checking checkers. The same conditions are shown to be necessary for the construction of two-level checkers and a design method for minimal two-level checkers is given. Using a well known theorem from combinatorics, a relationship which must exist between m and n in order for the above mentioned conditions to be satisfied is shown.

UILU-ENG 75-2228

GENERAL DESIGN RULES FOR THE CONSTRUCTION OF m-OUT-OF-n TOTALLY SELF-CHECKING CHECKERS

by

James E. Smith and Gernot Metze

This work was supported by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract Number DAAB-07-72-C-0259.

General Design Rules for the Construction of m-out-of-n Totally Self-Checking Checkers

James E. Smith and Gernot Metze

Abstract

This report presents a set of conditions sufficient to characterize a class of realizations of m-out-of-n code totally self-checking checkers. The same conditions are shown to be necessary for the construction of two-level checkers and a design method for minimal two-level checkers is given. Using a well known theorem from combinatorics, a relationship which must exist between m and n in order for the above mentioned conditions to be satisfied is shown.

1. Introduction

Achieving fault tolerant operation in digital systems through self-checking has received a great deal of attention recently [1,3,5,6]. In self-checking systems, inputs and outputs of logic function blocks, e.g. adders, are encoded in such a way that a fault within the block either does not affect the particular output of the block or transforms it into an invalid or noncode word. The validity of the output words is checked by checker circuits which produce "valid"/"invalid" indications and, of course, must check themselves during normal operation. That is, if a fault occurs within the check circuit itself, the same "invalid" indication is produced as for noncode inputs.

Some logic function blocks, e.g. memories, are particularly susceptible to unidirectional faults, i.e. faults whose components have all failed to the same value. Unidirectional faults are easily detected by m-out-of-n codes in which code words have precisely m 1's and n-m 0's and the change of any number of 1's to 0's, or vice versa, always results in a noncode word.

This report presents a set of conditions which are sufficient to characterize a class of realizations of m-out-of-n code checkers that are self-checking for unidirectional faults. The conditions given greatly simplify the design of m-out-of-n code self-checking checkers. These same conditions are then shown to be necessary for the realization of m-out-of-n totally self-checking checkers in two levels.

For arbitrary m and n, the above mentioned conditions cannot always be met. Using a well known theorem from combinatorics, a relationship which m and n must satisfy in order for the conditions to be met is shown.

Using the theory developed earlier, specific checker designs will be discussed. Designs proposed previously [2,5] will be shown to meet the given conditions and designs for minimal two-level checkers will be given.

1.1. Representation and Modeling of Faults

The _fault model_ used here is the usual stuck-at model [4]. That is, hardware failures in a logic circuit are modeled as some gate input or gate output lines stuck-at-1 (s-a-1) or stuck-at-0 (s-a-0). While the stuck-at fault model does not model all possible hardware failures, it models most of the more common failure modes and is easy to work with.

A _fault_ occurs when some lines in a logic circuit become s-a-1 or s-a-0. When a single line is stuck, we have a _single fault_, and a circuit has a _unidirectional fault_ if one or more lines are stuck at the same logic value.

The faults which are of primary interest in this report are unidirectional faults. The assumption is made that since the checker is being checked during normal operation, a unidirectional fault will be detected before a non-unidirectional fault occurs.

1.2. Properties of Totally Self-Checking Checkers

In his Ph.D. thesis [1], D. A. Anderson defined the concept of totally self-checking circuits and the properties which a totally self-checking checker must have. Similar definitions are given here.

Definition 1.1: A circuit is self-testing if, for every fault from a prescribed set, the circuit produces a noncode space output for at least one code space input.

Definition 1.2: A circuit is fault-secure if, for every fault from a prescribed set, the circuit never produces an incorrect code space output for code space inputs.

Definition 1.3: A circuit is totally self-checking for a prescribed fault set if it is both fault-secure for that set and self-testing for that set.

Definition 1.4: A circuit is code disjoint if the fault-free circuit always maps noncode space inputs to noncode space outputs.

Definition 1.5: A circuit is a totally self-checking checker if it is both totally self-checking and code disjoint.

For our checkers, the code space inputs are all m-out-of-n words, and our prescribed fault set is all unidirectional faults. Although the definition of totally self-checking checkers allows any number of outputs, it is customary to use the fewest necessary. It is easy to see that at least two are necessary [3], and in most cases two are sufficient. We will restrict ourselves to two outputs, as in [1], and let (0,1) and (1,0) be in the code space, and (1,1) and (0,0) in the noncode space. Based on the definition of totally self-checking checker, we now give the properties our checkers must have.

1) (Code disjoint property). The checker must map the $\binom{n}{m}$ code vertices onto the output pairs (0,1) or (1,0). The remaining (noncode) vertices are mapped onto either (0,0) or (1,1).

2) (Self-testing property). If a unidirectional fault occurs within the checker, at least one code word must be mapped onto (0,0) or (1,1).

3) (Fault-secure property). If a unidirectional fault occurs within the checker, no code word will be mapped onto the incorrect code output.

1.3. Notation

We consider an n input logic network to have the 2^n vertices of the n-cube as possible input combinations. We write a particular vertex as

$$x = (x_1, x_2, \ldots, x_n) \qquad \text{where} \quad x_i \in \{0,1\}.$$

A vertex with exactly k ones will be called a k-vertex. Thus, the set of m-out-of-n code inputs will be the set of m-vertices of the n-cube. This set will also be called the set of code vertices at times.

We define the usual partial ordering on the vertices:

$$x \geq y \quad \text{iff} \quad x_i \geq y_i \quad \text{for} \quad i = 1,2,\ldots,n.$$

For example, $(1,1,1,1) \geq (1,1,0,1)$

$(1,1,0,1) \geq (1,0,0,1)$

but, $(1,0,0,1) \not\geq (0,1,0,1)$.

We say that x covers y if $x \geq y$.

The complement of a vertex is formed by complementing all components. For example, the complement of (1,1,0,0) is (0,0,1,1).

Let the two logic functions which the two-output checker realizes be called f and g. This output pair will be written (f,g). For totally self-checking checkers, $(f,g) = (0,1)$ or $(1,0)$ for code input vertices and $(f,g) = (0,0)$ or $(1,1)$ for noncode input vertices.

2. Theorems Concerning the Constructions of Inverter-Free Totally Self-Checking Checkers

This report considers the class of totally self-checking checker networks which may be constructed using no inverters. Such inverter-free circuits have many desirable properties when unidirectional faults are being considered. All of the totally self-checking checker designs given previously [2,5] have used inverter-free networks.

2.1. Properties of Inverter-Free m-out-of-n Totally Self-Checking Checkers

Two well known properties of inverter-free circuits will be given below as lemmas without proofs. The proofs follow directly from results presented by Betancourt [7].

Lemma 2.1: Let f be the function realized by an inverter-free logic network. Then, for any two input vertices x and y such that $x \geq y$, $f(x) \geq f(y)$.

If this lemma is applied to our checker output pair, we see that if $x \geq y$ then

$$(f(x),g(x)) \geq (f(y),g(y)).$$

Lemma 2.2: Let f be the function realized by an inverter-free logic network and let $f_o(f_1)$ be any function the network realizes under a unidirectional s-a-0 (s-a-1) fault. Then, for any input vertex x, $f_1(x) \geq f(x) \geq f_o(x)$.

If we let $(f,g)_0$ be the checker output pair under a s-a-0 unidirectional fault and let $(f,g)_1$ be the output pair under a s-a-1 unidirectional fault, then we see that for any input x, $(f(x),g(x))_1 \geq (f(x),g(x)) \geq (f(x),g(x))_0$.

Consider now an inverter-free but otherwise unconstrained realization of a totally self-checking m-out-of-n checker with outputs (f,g). Lemma 2.1 can be used to characterize the distribution of the noncode vertices into F and G, the sets of true vertices for f and g, as follows.

Theorem 2.1: In an inverter-free realization of a totally self-checking m-out-of-n checker with outputs f and g, all vertices with more than m 1's must be in both F and G, and no vertex with fewer than m 1's may be in either F or G.

Proof: Let y denote an input vertex with fewer than m 1's (a noncode vertex). Then there exists at least one vertex x with exactly m 1's (a code vertex) such that $x > y$ and therefore, by Lemma 2.1, $(f(x),g(x)) \geq (f(y),g(y))$. But by Property 1 we must have $(f(x),g(x)) = (0,1)$ or $(1,0)$ and $(f(y),g(y)) = (0,0)$ or $(1,1)$. Since $(0,1) \ngeq (1,1)$ and $(1,0) \ngeq (1,1)$, any input vertex with fewer than m 1's must produce $(f,g) = (0,0)$, i.e. it cannot belong to either F or G. A similar argument shows that any input vertex with more than m 1's must produce $(f,g) = (1,1)$, i.e. it must belong to both F and G. Q.E.D.

Thus, the use of an inverter-free but otherwise unconstrained realization for totally self-checking m-out-of-n checkers completely determines which noncode vertices appear in F and G. Since Property 1 also requires that each code vertex belong either to F or to G but not to both, all that remains is the question of how to partition the code vertices into F and G.

We approach this question by constraining the realization of the checkers to be not only inverter-free but also irredundant and state two conditions on the partition of the code vertices into F and G.

Condition 1: Every (m+1)-vertex covers at least one m-vertex in F and at least one m-vertex in G.

Condition 2: Every (m-1)-vertex is covered by at least one m-vertex in F and at least one m-vertex in G.

These conditions are met by the partitions used in [1] as well as by the partition used in Example 2.1 below, although this is by no means easy to check. The conditions will be proved to be sufficient for any irredundant, inverter-free realization to be totally self-checking, and necessary for such realizations in two levels.

2.2. Sufficiency of Conditions 1 and 2 for a Totally Self-Checking Checker

Theorem 2.2: If Conditions 1 and 2 are met by the partition of the m-vertices of the n-cube into F and G, then any irredundant inverter-free network realizing f and g will be a totally self-checking m-out-of-n checker.

Proof: Assume a particular partition of the m-vertices into F and G is given. We will prove that the three properties of a totally self-checking checker given earlier are present.

Property 1: Since the m-vertices are partitioned into F and G, each of the m-vertices is in one and only one of F and G. Therefore, for any input code vertex, we must get an output of (0,1) or (1,0).

By Condition 1 and Lemma 2.1, the output (f,g) associated with an $(m+1)$-vertex must satisfy $(f,g) \geq (1,0)$ and $(f,g) \geq (0,1)$, thus $(f,g) = (1,1)$. Furthermore, the output (f',g') associated with an $(m+k)$-vertex, $k \geq 1$, must satisfy $(f',g') \geq (f,g) \geq (1,1)$, thus $(f',g') = (1,1)$. Similarly, by Condition 2 and Lemma 2.1, the output (f,g) associated with an $(m-k)$-vertex, $k \geq 1$, must satisfy $(1,0) \geq (f,g)$ and $(0,1) \geq (f,g)$, thus $(f,g) = (0,0)$. Hence, the outputs produced for noncode vertices are $(1,1)$ or $(0,0)$.

<u>Property 2</u>: Say we have a unidirectional s-a-0 fault which causes our network to realize $(f,g)_o$. Assume the network is not self-testing, i.e. code vertices produce only code outputs under this fault. Then, since the network is irredundant, a noncode vertex x must be a test for this fault.

By Condition 1 there exists a code vertex in F, say y, such that $x > y$. Then $(f(y),g(y)) = (1,0) \geq (f(y),g(y))_o$ by Lemma 2.2. But y is assumed to be a code vertex that produces only code outputs under the fault. Therefore $(f(y),g(y))_o = (0,1)$ or $(1,0)$ which combines with the previous result to yield the unique solution

$$(f(y),g(y)) = (1,0) = (f(y),g(y))_o.$$

Since a unidirectional fault in an inverter-free network yields another inverter-free network, we have by Lemma 2.1,

$$\begin{aligned}(f(x),g(x))_o &\geq (f(y),g(y))_o \\ &= (1,0) \text{ by the above argument.}\end{aligned}$$

By Condition 1 there also exists a code vertex in G, say z, such that $x > z$. By similar reasoning we obtain

$$(f(x),g(x))_o \geq (f(z),g(z))_o = (f(z),g(z)) = (0,1).$$

In combination, we must therefore have

$$(f(x),g(x))_o = (1,1) \geq (1,0) \text{ or } (0,1).$$

Since x is a test for the fault we have $(f(x),g(x)) \neq (f(x),g(x))_o$ which combines with Lemma 2.2 to yield $(f(x),g(x)) > (f(x),g(x))_o$. Since x is a noncode vertex we must have $(f(x),g(x)) = (1,1)$ or $(0,0)$. Thus we must satisfy

$$(1,1) \text{ or } (0,0) = (f(x),g(x)) > (f(x),g(x))_o = (1,1)$$

from the arguments concerning vertices y and z, which is impossible. Thus, a code vertex must exist which tests for the unidirectional s-a-0 fault. Furthermore, the output of the network with the s-a-0 fault under this test must be (0,0) due to Lemma 2.2.

By a similar argument using Condition 2, it can be shown that a code vertex must test for any unidirectional s-a-1 fault and the output of the faulty network under the test is (1,1).

<u>Property 3</u>: $(f(x),g(x)) \geq (f(x),g(x))_o$ for any x by Lemma 2.2. Since $(f(x),g(x)) = (1,0)$ or $(0,1)$ for a code vertex x, $(f(x),g(x))_o = (0,0)$ or $(f(x),g(x))$. By a similar argument, $(f(x),g(x))_1 = (1,1)$ or $(f(x),g(x))$ for any code word x. Thus, the fault either produces a noncode output or leaves the correct code output unaffected. Q.E.D.

We note that this theorem was derived independently of Theorem 2.I. Thus, the partition of the code vertices into F and G according to Conditions 1 and 2 is sufficient to determine the totally self-checking nature of the checker, and the distribution of the noncode vertices into F and G is implicitly determined.

Theorem 2.2 states that all of the code vertices are sufficient to test any realization of f and g. However, all of the code vertices are not always necessary as can be seen in Example 2.1.

Example 2.1: For a 3-out-of-6 checker, we may use as F and G:

F = {all vertices with more than m 1's and the following m-vertices:
(1,1,1,0,0,0),(1,1,0,1,0,0),(0,0,1,1,1,0),(0,0,1,1,0,1),
(1,0,0,0,1,1),(0,1,0,0,1,1)}

G = {all vertices with more than m 1's and all m-vertices not in F}.

The above partition of code vertices into F and G is one of many which meets Conditions 1 and 2. One of the many realizations of f and g is shown in Figure 2.1. It can be verified that a complete test set for unidirectional faults in that circuit is {(0,1,1,1,0,0),(1,0,0,0,1,1),(1,1,0,1,0,0), (0,0,1,0,1,1),(1,1,0,0,0,1),(0,0,1,1,1,0)}, so that the set of code vertices is certainly sufficient.

2.3. Necessity of Conditions 1 and 2 for a Two-level Totally Self-Checking Checker

Since Conditions 1 and 2 are sufficient for any inverter-free realization to be totally self-checking, they are obviously sufficient for a two-level realization. As we will see in this section, they are also necessary for a two-level realization to be totally self-checking.

Lemma 2.3: In an irredundant two-level m-out-of-n totally self-checking checker, the only fanout which can occur is at the primary inputs.

Proof: By exhausting all possible forms of fanout (except primary input fanout) it can be easily shown that each form leads to a circuit which either cannot possibly be a checker or which is not self-testing.

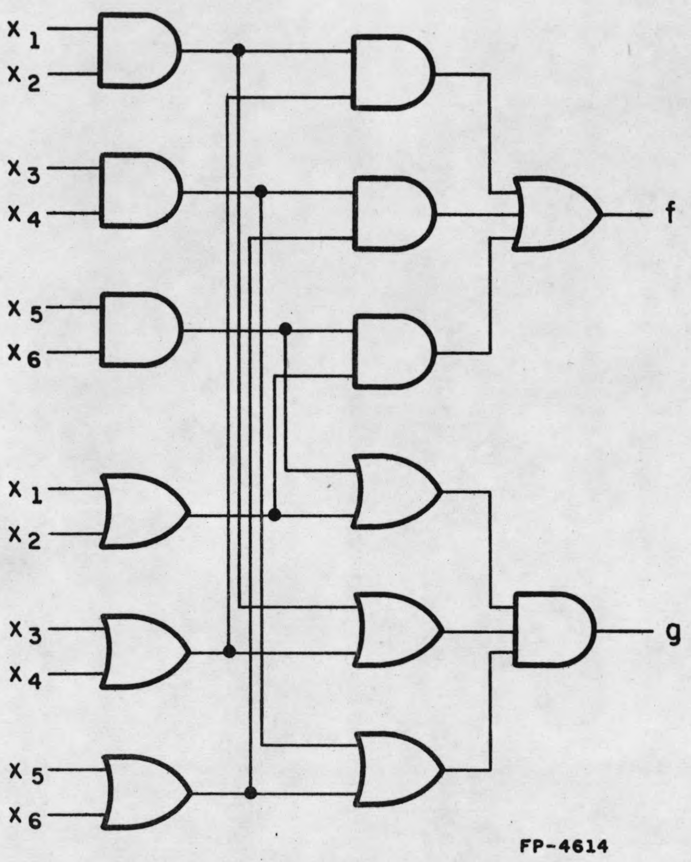

Figure 2.1 A Totally Self-checking
3-out-of-6 Checker

The only form of fanout which even merits consideration is the fanout from the AND (OR) gate output of an AND-OR (OR-AND) realization of f into the output gate which forms g. In this case we cannot test a s-a-0 (s-a-1) fault where the fanout line enters the output gate of g with a code input. Q.E.D.

As a consequence of Lemma 2.3, f and g are each either in AND-OR or OR-AND form with only primary inputs shared by the logic forming f and g.

<u>Lemma 2.4</u>: In an AND-OR realization of f or g, each AND gate must have exactly m inputs. In an OR-AND realization of f or g, each OR gate must have exactly n-m inputs.

<u>Proof</u>: In an AND-OR realization, if an AND gate has fewer than m inputs, it will produce the value 1 for a vertex with fewer than m 1's. Therefore, there must be a true vertex of the function it realizes with fewer than m 1's; this contradicts Theorem 2.1.

If the AND gate has more than m inputs, it will require a vertex with more than m 1's to check for the output of the AND s-a-0. Therefore, the circuit is not self-testing.

Thus, we find that every AND gate must have exactly m inputs in an AND-OR realization.

By a dual argument, we get the result that every OR gate must have exactly n-m inputs in an OR-AND realization. Q.E.D.

From the above lemma, we see that each AND gate in an AND-OR realization corresponds to one code vertex and has inputs corresponding to the coordinates of the vertex which are 1's. The AND gate produces the value 1 when the code vertex it corresponds to is present at the inputs and the value 0 when any other code vertex is present.

Each OR gate in an OR-AND realization also corresponds to one code vertex, and has inputs which correspond to the coordinates of the vertex which are 0's. The OR gate takes the value 0 when the code vertex it corresponds to is present at its inputs and the value 1 when any other code vertex is present.

Theorem 2.3: Conditions 1 and 2 are necessary for any two-level irredundant inverter-free m-out-of-n checker to be totally self-checking.

Proof: Assume we fail Condition 1 when we partition m-vertices into F and G. Then without loss of generality, say F has no m-vertex covered by a particular (m+1)-vertex in F.

Case 1: Let f be realized by an AND-OR circuit. Then, due to Theorem 2.1, there must be an AND gate which goes to 1 for the (m+1)-vertex but which is 0 for any code vertex. This requires an AND gate with more than m inputs which contradicts Lemma 2.4.

Case 2: Let f be realized by an OR-AND circuit. Then, f cannot have as a true vertex any of the m-vertices covered by the (m+1)-vertex. Then, all of the true vertices of f must have at least one 1 from the n-(m+1) positions which are not covered by the (m+1)-vertex. This implies that f goes to 1 for the (m-1)-vertex formed by complementing the (m+1)-vertex. This contradicts Theorem 2.1.

So, if we fail Condition 1, then the checker cannot be realized by a two-level inverter-free circuit.

Now, assume we fail Condition 2 when we partition the m-vertices into F and G. Without loss of generality, say all m-vertices covering a particular (m-1)-vertex appear in F.

<u>Case 1</u>: Let f be realized by an AND-OR circuit. Consider an AND gate which corresponds to an m-vertex covering the (m-1)-vertex. To test the input which does not correspond to one of the m-1 coordinates of the (m-1)-vertex for a s-a-1 fault, we must set the m-1 inputs to 1 and the line being tested to 0. However, to be self-testing we must have one more input variable which is set to 1. Whichever variable we choose, we must apply a true vertex of F and thus desensitize the path from the line being tested. Therefore, in this case we must have a circuit which is not self-testing.

<u>Case 2</u>: Let f be realized by an OR-AND circuit. In this case, by Theorem 2.1, f must go to 0 for the particular (m-1)-vertex, but goes to 1 for all the m-vertices covering it. Then, there must be an OR gate which goes to 0 for the (m-1)-vertex, but which goes to 1 for all the code vertices. This requires an OR gate with more than n-m inputs. This is a contradiction of Lemma 2.4.

Therefore, if we fail Condition 2, the checker cannot be realized by a two-level inverter-free circuit. Q.E.D.

3. The Existence of Satisfactory Partitions for an Arbitrary m and n

In Section 2, we proved two theorems concerning partitions of the m-vertices of the n-cube which meet Conditions 1 and 2. However, for some m and n, such partitions cannot be made. In this section we will present a relationship between m and n whose satisfaction is a necessary and sufficient condition for the existence of a partition which meets Conditions 1 and 2.

3.1. Ramsey's Theorem and Ramsey Numbers

Ramsey's theorem [8] is a well known theorem in combinatorics. The theorem is usually presented in terms of subsets of a set, and the Ramsey numbers $N(p,q;r)$ are defined; a good proof and examples can be found in [9]. We present here a version in terms of vertices of an n-cube that is specialized for our purposes and let $R(m) = N(m+1,m+1;m)$.

Theorem 3.1 (Ramsey): For any given integer $m \geq 1$ there exists a finite number $R(m)$ depending solely on m such that for any n-cube C, $n \geq R(m)$, and for any arbitrary partition of all the m-vertices of C into two blocks, there exists an (m+1)-vertex of C which covers no m-vertex in one of the blocks.

The Ramsey numbers where $m=1$ are easily computed, however if $m \geq 2$ their determination is much more difficult and very few are known [9].

Example 3.1: It is known that $R(2)=6$. This means that if we partition the 2-vertices of the 6-cube (or larger cube) into two blocks in any manner, there must be a 3-vertex which covers no 2-vertex in one block of the partition. Conversely since R(2) is the least number having these properties, at least one 2-block partition of the 2-vertices of a 5-cube (or smaller cube) exists such that every 3-vertex covers at least one 2-vertex in each block, as in the partition

$$A = \{12,14,25,34,35\},\ B = \{13,15,23,24,45\}.$$

We see that Conditions 1 and 2 are in some sense duals. That is, if we can get a partition for an m-out-of-n code, we can also get a partition for an (n-m)-out-of-n code by simply complementing the partitioned m-vertices.

For this reason, without loss of generality, we will consider only the case where $m \leq \left\lfloor \frac{n}{2} \right\rfloor$.

We will show that we can partition the m-out-of-n vertices, $m \leq \left\lfloor \frac{n}{2} \right\rfloor$, such that the partition meets Conditions 1 and 2 if and only if $R(m) > n$.

3.2. Main Theorem

Let $P^+(m,n)$ be the collection of partitions of m-vertices of the n-cube which satisfy Condition 1. Let $P^-(m,n)$ be the collection of partitions of m-vertices of the n-cube which satisfy Condition 2.

Let $A|B$ denote a partition of the m-vertices where A and B are the two blocks of the partition. If $P^-(m,n)$ is not empty an arbitrary partition $A|B$ can obviously be converted into a member of $P^-(m,n)$ by moving some subset of block A into block B and some subset of block B into block A. We will perform such moves, but only one vertex at a time, and only if a certain condition is met.

Definition 3.1: A move of an m-vertex from A to B (B to A) is a proper move only if it covers an (m-1)-vertex which was not previously covered by any member of B (A).

Lemma 3.1: Starting with an arbitrary initial partition $A|B$ of the m-vertices, some series of proper moves will lead to a member of $P^-(m,n)$ if $P^-(m,n) \neq \emptyset$.

Proof: Say $A'|B' \in P^-(m,n)$ and $A|B$ can be converted to $A'|B'$ by moving $A^* \subseteq A$ into B and $B^* \subseteq B$ into A. We see that $A' = (A-A^*) \cup B^*$ and $B' = (B-B^*) \cup A^*$. Now, we make instead a series of proper moves from A into B and from B into A. However, we only move members of A^* into B and members of B^* into A. Say we reach a partition $A''|B''$ such that we can no longer make a proper move.

If $A''|B'' \in P^-(m,n)$ we are done. If not, then there is some (m-1)-vertex not covered by any member of A'' or some (m-1)-vertex not covered by any member of B''. Without loss of generality, say s is an (m-1)-vertex not covered by any member of A''. Since we can no longer make a proper move, no member of B^* covers s. But, we know that $A' = (A-A^*) \cup B^*$ covers all (m-1)-vertices. Hence, at least one member of B^* must cover the (m-1)-vertex s, and we have a contradiction. Therefore, $A''|B''$ must be a member of $P^-(m,n)$. Q.E.D.

<u>Lemma 3.2</u>: Given any member of $P^+(m,n)$, if we make any proper move, the result will also be a member of $P^+(m,n)$.

<u>Proof</u>: Let $A|B$ be a member of $P^+(m,n)$. Without loss of generality, say we are making a move from A to B which results in the partition $A'|B'$ and that the moved m-vertex is intended to cover the (m-1)-vertex $(\underbrace{1,1,1,\ldots,1}_{(m-1)\ 1\text{'s}},0,0,\ldots,0)$. Then, before the move, the m-vertices

$$(\overbrace{1,1,1,\ldots,1}^{(m-1)\ 1\text{'s}},1,0,0,\ldots,0,0)$$
$$(1,1,1,\ldots,1,0,1,0,\ldots,0,0)$$
$$(1,1,1,\ldots,1,0,0,1,\ldots,0,0)$$
$$\vdots$$
$$(1,1,1,\ldots,1,0,0,0,\ldots,0,1)$$

must all have been in A. We see that each (m+1)-vertex which covers one of these m-vertices also covers another. Thus, if we move any one of these m-vertices as our proper move, all the (m+1)-vertices which cover it still cover other members of A. Therefore, if $A|B \in P^+(m,n)$ before the proper move, $A'|B' \in P^+(m,n)$ after the proper move. Q.E.D.

Lemma 3.3: If $P^{+}(m,n) \neq \phi$ and $P^{-}(m,n) \neq \phi$ then $P^{+}(m,n) \cup P^{-}(m,n) \neq \phi$.

Proof: If we are given a member of $P^{+}(m,n)$ initially, by Lemma 3.1 some set of proper moves will lead to a member of $P^{-}(m,n)$ since $P^{-}(m,n) \neq \phi$. Since each move was proper, by Lemma 3.2, the resulting member of $P^{-}(m,n)$ must also be a member of $P^{+}(m,n)$. Q.E.D.

Theorem 3.2: For $m \leq \left\lfloor \frac{n}{2} \right\rfloor$, $R(m) > n$ iff the m-vertices of the n-cube can be partitioned to meet Conditions 1 and 2.

Proof: If $R(m) > n$ then we can partition the m-vertices of the n-cube into two blocks such that all m-vertices covered by an (m+1)-vertex are not in the same block. This implies $P^{+}(m,n) \neq \phi$.

Since $n-m \geq m$ for $m \leq \left\lfloor \frac{n}{2} \right\rfloor$, $R(n-m) > n$. Therefore, $P^{+}(n-m,n) \neq \phi$. But we see that we can complement the (n-m)-vertices in a partition in $P^{+}(n-m,n)$ to get a partition in $P^{-}(m,n)$. Therefore, $P^{-}(m,n) \neq \phi$.

Now $P^{+}(m,n) \cup P^{-}(m,n) \neq \phi$ by Lemma 3.3, which means that we can select a partition of m-vertices such that Conditions 1 and 2 are satisfied. Satisfying Condition 1 implies that $P^{+}(m,n) \neq \phi$ which implies that $R(m) > n$. Q.E.D.

We now present what is known about Ramsey numbers of the type $R(m) = N(m+1,m+1;m)$. It is easy to show that $R(1) = 3$ and relatively easy to show that $R(2) = 6$ [9]. These are the only known Ramsey numbers of this form. Bounds for higher m do exist, however. For $m = 3$ it has been shown [10,11] that $13 \leq R(3) \leq 17$. Bounds for $m > 4$ are rather loose [9], but may still be useful in some instances.

From the preceding paragraph and our previous theorems, several conclusions can be drawn concerning the construction of totally self-checking checkers. For example, the only 1-out-of-n and 2-out-of-n two-level totally self-checking checkers that can be constructed are the trivial 1-out-of-2, the 2-out-of-4, and the 2-out-of-5 checkers. From the bounds on R(3) we can conclude that 3-out-of-n totally self-checking checkers can definitely be constructed in two levels for $6 \leq n \leq 12$, and definitely not for $n \geq 17$. Furthermore, we can reach similar conclusions for (n-1)-out-of-n, (n-2)-out-of-n and (n-3)-out-of-n checkers. From a consideration of coverings we see that it is easy to partition the m-out-of-n codewords for $m = \left\lfloor \frac{n}{2} \right\rfloor$ so that we meet Conditions 1 and 2, and a relatively large number of such partitions exists. As m becomes smaller (or larger) for a fixed n, fewer and fewer satisfactory partitions exist and it becomes increasingly difficult to find one. Finally, when $R(m) \leq n$ it becomes impossible to find a satisfactory partition.

An m-out-of-n code has an optimal or near optimal number of code words when $m \simeq \left\lfloor \frac{n}{2} \right\rfloor$ [1]. For this reason, m-out-of-n codes with $m \simeq \left\lfloor \frac{n}{2} \right\rfloor$ are of the greatest practical interest. For these particular codes, the designer of totally self-checking checkers is given a great deal of latitude in choosing a design which fits his particular constraints. Any of a large number of partitions of m-vertices into F and G may be used, and given the partition, any inverter-free circuit may be used to realize f and g. Thus, a designer may find that one partition and a particular circuit give him a design with minimal gates, minimal lines, or whatever property he may be interested in. Therefore, although the problem of finding a satisfactory partition of the code vertices into F and G is in general difficult, it is relatively easy for codes of practical interest.

4. Some Specific Checker Designs

Of the previous checker designs, the ones by Bouricius, Carter, Duke, Roth, and Schneider [3,12] do not consider faults on gate input lines, and therefore all of the results given here do not apply to them.

The first designs for totally self-checking m-out-of-n checkers where faults may occur on input lines were given by D. A. Anderson [1]. He used majority functions to realize k-out-of-2k checkers and simple code translators to realize arbitrary m-out-of-n checkers except for 1-out-of-3 and 1-out-of-7 checkers. The checkers proposed by Reddy [5] differ from Anderson's in the way the majority functions are formed but use the same partitions. Reddy also presented a design for a 1-out-of-7 checker.

It will be shown that the partition of code words which Anderson uses for his k-out-of-2k checker meets Conditions 1 and 2, and therefore any inverter-free realization is totally self-checking.

The design of two level totally self-checking m-out-of-n checkers will be discussed including minimal two-level designs.

4.1. Majority Function Checkers for k-out-of-2k Codes

In the k-out-of-2k checkers proposed by Anderson, the 2k-bit input word is divided into two halves α and β each having k bits. $T(n_\alpha \geq i)$ is the majority function which takes the value 1 if the number of 1's in the α half (n_α) is greater than or equal to i.

The functions f and g are defined as follows:

$$f = \sum_{i=0}^{k} T(n_\alpha \geq i)\cdot T(n_\beta \geq k-i) \quad i \text{ odd}$$

$$g = \sum_{i=0}^{k} T(n_\alpha \geq i)\cdot T(n_\beta \geq k-i) \quad i \text{ even.}$$

Any (k-1)-vertex y must have at least one 0 in each of α and β. By placing a 1 in a 0 position in the α half of y, we form a k-vertex y'. By placing a 1 in a 0 position in the β-half of y, we form a k-vertex y". If k is odd, y' will be in F and y" will be in G. If k is even, the opposite will be true. In either case, for an arbitrary (k-1)-vertex, at least one k-vertex covering it will be true for F and at least one covering it will be true for G. Therefore, Condition 2 is met by the partition into F and G.

Any (k+1)-vertex z must have at least one 1 in the α half and at least one 1 in the β half. We form z' and z" by changing a 1 in the α half to a 0 and by changing a 1 in the β half to 0. By an argument similar to the one above, one of z' and z" goes into F and the other into G. Thus, Condition 1 is met by the partition into F and G.

The partition of the k-vertices into F and G imposed by Anderson's majority function method must meet both Conditions 1 and 2. Therefore, any inverter-free realization of Anderson's f and g functions must be totally self-checking.

4.2. Minimal Two Level m-out-of-n Checkers

A totally self-checking system is composed of modules with information passing between the modules in encoded form. Each time information is passed, it must be checked before more information can be transferred. For this reason, it is important that the checking process be as fast as possible.

Since the speed of a combinational circuit depends largely upon the number of logic levels used, it is easy to justify the desirability of two-level checkers. In this section, we will present a procedure for designing two-level totally self-checking checkers for m-out-of-n codes which use a minimum number of gates.

As we have shown, it is not always possible to partition the m-vertices such that Conditions 1 and 2 are met for arbitrary m and n. Therefore, it is not always possible to construct a two-level totally self-checking checker for arbitrary m and n. For example a 2-out-of-6 checker cannot be realized in two levels since $R(2) \not> 6$.

If a satisfactory partition of the m-vertices of an n-cube can be made, we let $H(m,n)$ be the minimal number of code vertices that must be in block F. Then an AND-OR realization of f must have at least $H(m,n)+1$ gates since there is one AND gate for every m-vertex in F.

Similarly, an OR-AND realization of g must have at least $H(m,n)+1$ gates since one OR gate corresponds to each false m-vertex of g. Any other realizations of f or g under this partition can only require more gates. The same is true if a partition is used where $|\text{m-vertices in F}| > H(m,n)$ and $|\text{m-vertices in G}| > H(m,n)$. Therefore, a minimal two-level m-out-of-n checker must have at least $2H(m,n)+2$ gates.

We can achieve this minimum by following the procedure just discussed. We should first partition the m-vertices according to Conditions 1 and 2 such that the number of m-vertices in F is minimal, then we should realize f using an AND-OR circuit and g using an OR-AND circuit. The inputs to the AND gates which realize f correspond to the 1 positions in the vertices in F, and the inputs to the OR gates which realize g correspond to 0 positions of the vertices in F. It is easy to see that such a realization performs the mapping specified by F and G and must be a totally self-checking checker.

Example 4.1: Minimal two-level 3-out-of-6 totally self-checking checker. Let F and G be the same sets as in Example 2.1. (For this partition $|\text{m-vertices in F}|$ is minimum.) The checker is shown in Figure 4.1.

4.3. A Lower Bound on H(m,n)

A nonexhaustive procedure for generating a partition such that $|\text{m-vertices in F}| = H(m,n)$ has not been found, and this partition is believed to be quite difficult in the general case. In fact the numbers H(m,n) are not known in general, although we can derive a lower bound on H(m,n) which appears to be rather tight for $m \simeq \left\lfloor \frac{n}{2} \right\rfloor$.

Let $H^-(m,n)$ be the minimum number of m-vertices of the n-cube which are required to cover all (m-1)-vertices, and let $H^+(m,n)$ be the minimum number of m vertices we must have in order for all (m+1)-vertices to cover at least one m-vertex. We observe that $H(m,n) \geq \max(H^-(m,n), H^+(m,n))$.

The covering of (m-1)-tuples by m-tuples has been treated rather extensively in combinatorics [13,14]. From [13] we get a bound on $H^-(m,n)$; in particular,

$$H^-(m,n) \geq \left\lceil \frac{n}{m} \left\lceil \frac{n-1}{m-1} \left\lceil \cdots \left\lceil \frac{n-m+3}{3} \left\lceil \frac{n-m+2}{2} \right\rceil \right\rceil \cdots \right\rceil \right\rceil \right\rceil .$$

Due to the same type of 'duality' we observed between Conditions 1 and 2 we see that

$$H^+(m,n) = H^-(n-m,n).$$

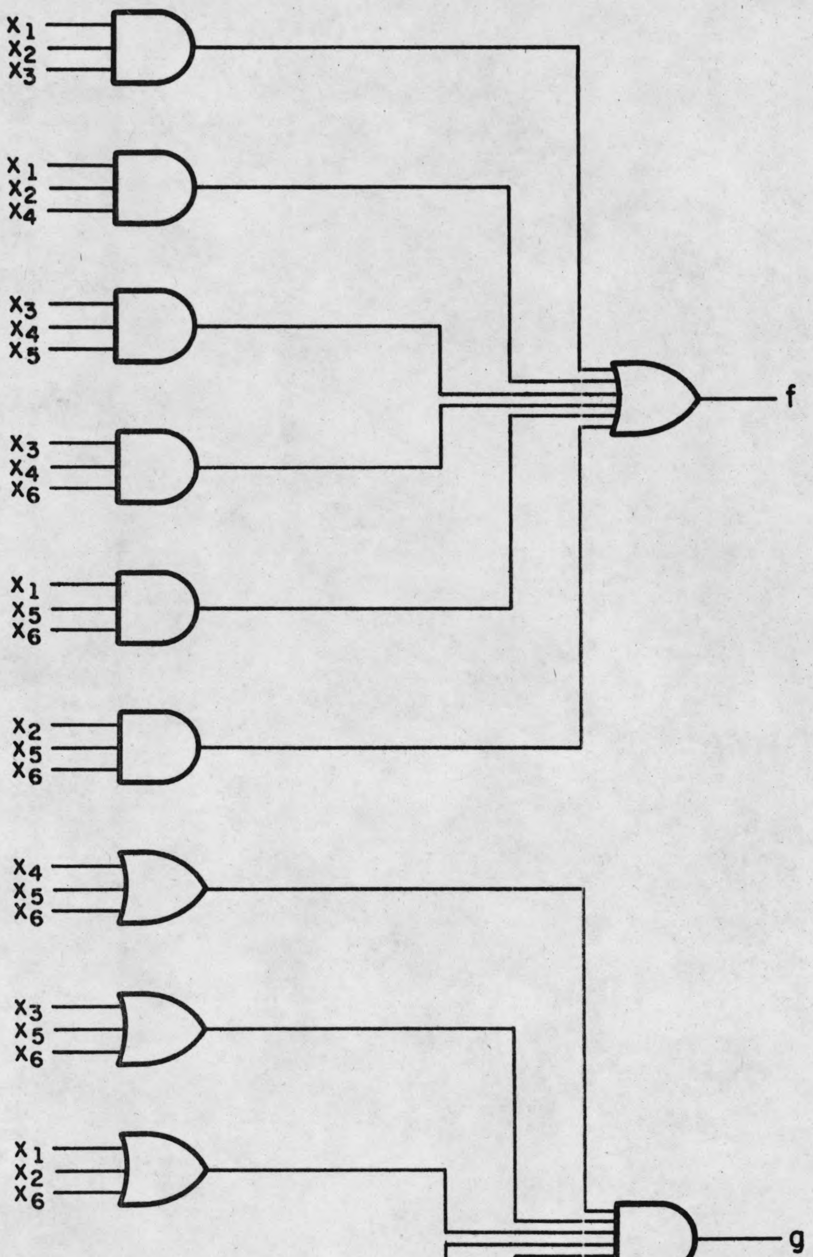

Figure 4.1. A Minimal Two-Level 3-out-of-6 Totally Self-Checking Checker

Therefore, $H^{+}(m,n) \geq \left\lceil \frac{n}{n-m} \left\lceil \frac{n-1}{n-m-1} \left\lceil \cdots \left\lceil \frac{m+3}{3} \left\lceil \frac{m+2}{2} \right\rceil \right\rceil \cdots \right\rceil \right\rceil \right\rceil$, and

$$H(m,n) \geq \max\left(\left\lceil \frac{n}{m} \left\lceil \frac{n-1}{m-1} \left\lceil \cdots \left\lceil \frac{n-m+3}{3} \left\lceil \frac{n-m+2}{2} \right\rceil \right\rceil \cdots \right\rceil \right\rceil \right\rceil, \right.$$

$$\left. \left\lceil \frac{n}{n-m} \left\lceil \frac{n-1}{n-m-1} \left\lceil \cdots \left\lceil \frac{m+3}{3} \left\lceil \frac{m+2}{2} \right\rceil \right\rceil \cdots \right\rceil \right\rceil \right\rceil \right).$$

For H(2,4), H(3,6), H(4,8) this bound is 2, 6, and 14 respectively. Actual partitions have been found which meet these bounds exactly in these cases. Thus, for H(k,2k) it appears that our bound is very close to being exact. We further observe that for k-out-of-2k codes we can expect H(k,2k) to be considerably less than $\binom{2k}{k}\Big/2$ which would result from a partition with an equal number of code vertices in F and G.

For H(2,5) our bound is 4; however, it can be shown that H(2,5) = 5. This is evidence that as m moves further away from $\left\lfloor \frac{n}{2} \right\rfloor$ our bound becomes less exact, and other examples confirm this trend.

5. Summary and Conclusions

Sufficient conditions on the mapping of m-out-of-n code words such that any inverter-free realization of the mapping is a totally self-checking checker are given. Previously given checker designs fall into the class of checkers which meet these conditions.

The subclass of two-level checkers is then considered, and the sufficient conditions are proved to be necessary for two-level realizations. From this we can deduce that our sufficient conditions are as loose as they can be. That is, if a mapping of m-out-of-n words does not meet the sufficient conditions, at least one realization of that mapping will <u>not</u> be a totally self-checking checker.

The sufficient conditions cannot always be met by a particular m and n, and a relationship between m and n using Ramsey numbers is given from which it can be determined whether the conditions can or cannot be met.

Finally, specific checker designs are discussed, and minimal two-level checker designs are given.

The difficulty of showing the presence of the sufficient conditions for an arbitrary m and n and the difficulty of finding minimal two-level designs for arbitrary m and n point out the level of complexity inherent in the structure of m-out-of-n codes. However, this complexity is not prohibitive if we desire m-out-of-n checkers with $m \simeq \frac{n}{2}$, or two-level checkers which are only near-minimal.

For future research, we suggest that the structure of other unordered codes, such as the Berger code, be studied more closely to determine if similar results can be found for them. Since the structure of m-out-of-n codes has been shown to be rather complex, perhaps some other unordered codes can be found which have a simpler structure which would lead to simpler check circuits, and possibly simpler totally self-checking circuits in general.

The use of covering concepts and Ramsey numbers proved to be very helpful in our analysis; we suspect that other results exist in combinatorics which will provide further insight into the structure of unordered codes.

References

1. Anderson, D. A., "Design of Self-Checking Digital Networks Using Coding Techniques," Coordinated Science Laboratory Report R-527, Universoty of Illinois, September 1971.

2. Anderson, D. A. and G. Metze, "Design of Totally Self-Checking Check Circuits for m-out-of-n Codes," IEEE Trans. Comp., Vol. C-22, pp. 263-269, March 1973.

3. Carter, W. C. and P. R. Schneider, "Design of Dynamically Checked Computers," IFIP 68, Vol. 2, Edinburg, Scotland, pp. 878-883, August 1968.

4. Chang, H. Y., E. Manning, and G. Metze, Fault Diagnosis of Digital Systems, Wiley-Interscience, New York, 1970.

5. Reddy, S. M., "A Note on Self-Checking Checkers," IEEE Trans. Comp., Vol. C-23, pp. 1100-1102, October 1974.

6. Ho, D., "The Study of a Totally Self-Checking Adder," Coordinated Science Laboratory Report R-582, University of Illinois, August 1972.

7. Betancourt, R., "Derivation of Minimum Test Sets for Unate Logical Circuits," IEEE Trans. Comp., Vol. C-20, pp. 1264-1269, November 1971.

8. Ramsey, F. P., "On a Problem of Formal Logic," Proc. London Math. Soc., 2nd Ser., Vol. 30, pp. 264-286, 1930.

9. Liu, C. L., Topics in Combinatorial Mathematics, notes on lectures given at the 1972 MAA Summer Seminar, Williams College, Williamstown, Mass., Math. Assoc. of America, 1972.

10. Isbell, J. R., "$N(4,4;3) \geq 13$," Journal of Comb. Theory, Vol. 6, p. 210, 1969.

11. Kalbfleisch, J. G., "On the Ramsey Number $N(4,4;3)$," Recent Progress in Combinatorics, W. J. Tutte, Ed., Academic Press, New York, pp. 273-282, 1969.

12. Bouricius, W. G., W. C. Carter, K. A. Duke, J. P. Roth, and P. R. Schneider, "Interactive Design of Self-Testing Circuitry," Purdue Centennial Year Symp. on Information Processing, pp. 73-80, April 1969.

13. Schönheim, J. "On Coverings," Pacific Journal of Math., Vol. 14, pp. 1405-1411, 1964.

14. Kalbfleisch, J. G. and R. G. Stanton, "Maximal and Minimal Coverings of (k-1)-tuples by k-tuples," Pacific Journal of Math., Vol. 26, No. 1, pp. 131-140, 1968.