



Fault location in cellular arrays
by Kenneth James Thurber

A thesis submitted to the Graduate Faculty in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY in Electrical Engineering
Montana State University
© Copyright by Kenneth James Thurber (1969)

Abstract:

The subject of this thesis is the development of concepts and techniques to be used in the testing of cellular logic arrays. In particular, the subject is to develop techniques to locate, detect, and isolate failures in cellular logic arrays.

The primary purpose of the thesis is to develop the most general fault location theory for two-dimensional cellular arrays consisting of two-input, one-output cells. The theory presented utilizes a physical basis to assume a maximum set of allowable errors and thus realistically accounts for the appearance of cell failures.

The content of the thesis is summarized as follows: First, a physical basis for the assumption of the maximum error set is given.

The assumption of the maximum error set assures that the most general solution will be obtained. Second, the problem of testing a two-dimensional cellular array is expressed in terms of testing a Maitra cascade. Third, a necessary and sufficient condition for the location of a single error in a Maitra cascade is given and proven. Also, a least upper bound on the number of tests needed to test the array is derived. Fourth, an algorithm is given, based on the necessary and sufficient condition, for error location, which allows testing of arrays for either location or detection of errors. Fifth, error detection and error location methods are given for certain very important proper subsets of the maximum error subset. Sixth, examples are given that 1) illustrate the various error location and detection methods that were proposed, and 2) illustrate that location of some multiple errors is possible utilizing single error location theories.

FAULT LOCATION IN CELLULAR ARRAYS

by

KENNETH JAMES THURBER

A thesis submitted to the Graduate Faculty in partial
fulfillment of the requirements for the degree


of

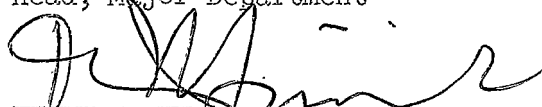
DOCTOR OF PHILOSOPHY

in

Electrical Engineering

Approved:


Head, Major Department


Chairman, Examining Committee


Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

June, 1969

D378
7424
cop. 2

iii

ACKNOWLEDGMENT

The author wishes to express his appreciation to Professor R. C. Minnick for his many suggestions, frequent encouragement, and untiring guidance during the course of this graduate work and thesis research.

The financial support of the graduate work furnished by the Electrical Engineering Department, National Defense Education Act Title IV Fellowship No. 67-06596, and National Science Foundation Grant No. GJ-158 is gratefully acknowledged, as is the thesis support of National Defense Education Act Title IV Fellowship No. 67-06596 and U. S. Air Force Cambridge Research Laboratories Contract F 19628-67-C-0293.

TABLE OF CONTENTS

	Page
Chapter 1: INTRODUCTION TO THE TESTING OF INTEGRATED CELLULAR LOGIC ARRAYS	1
1.1 Introduction	2
1.2 Problem Definition	3
1.3 Practical Considerations	6
1.4 Testing	8
1.5 Generation of Tests and Test Equipment	9
1.6 Conclusion	11
Chapter 2: SURVEY OF PUBLISHED WORK IN THE FIELD OF FAULT DETECTION	12
2.1 Introduction	13
2.2 General Strategy	13
2.3 Some Diagnostic Techniques	14
2.3.1 Boolean Graphs	14
2.3.2 Armstrong's Method	20
2.3.3 Redundant Circuits	24
2.3.4 Kautz's Method	25
2.3.5 An Adaptive Solution	26
2.3.6 Programmed Algorithms	28
2.4 Testing of Cellular Arrays	30
2.5 Digital Simulation of Failures in Digital Systems	31
2.6 Conclusion	32
Chapter 3: FAULT LOCATION IN CASCADES	33
3.1 Introduction	34
3.2 Assumptions	35
3.3 Objectives	36
3.4 General Concepts	36
3.5 Definitions	39

TABLE OF CONTENTS (continued)

	Page
Chapter 3 (continued)	
3.6 A Necessary and Sufficient Condition for Location of Faults in a Cascade and a Least Upper Bound for the Number of Tests	40
3.7 Test Algorithm	59
3.8 Conclusion	59
Chapter 4: TRIBUTARY CASCADES	63
4.1 Introduction	64
4.2 Classification of Tributary Cascades	64
4.3 Subcascades	66
4.4 Location and Isolation of Faults in Tributary Cascades	68
4.5 Conclusion	74
Chapter 5: EXAMPLES	75
5.1 Introduction and Description of Examples	76
5.2 Examples	79
Chapter 6: CONCLUSION	109
6.1 Summary	110
6.2 Suggestions for Further Study	112
LITERATURE CITED	116

LIST OF FIGURES

	Page
Figure 1.1	Interconnection Structure of Cascades 4
Figure 1.2	Construction of a Testable Cellular Array 5
Figure 2.1	Circuit in Which a Gate s-a-l Cannot Be Located 15
Figure 2.2	A Boolean Graph, Its Injective Word, and Its Analysis 17
Figure 3.1	Determination of Y_{i-1} 38
Figure 3.2	Test Decision Map for f_{14} 42
Figure 3.3	Test Decision Map for f_{11} 43
Figure 3.4	Test Decision Map for f_8 44
Figure 3.5	Test Decision Map for f_2 45
Figure 3.6	Test Decision Map for f_6 46
Figure 3.7	Test Decision Map for f_9 47
Figure 3.8	Test Decision Map for f_{13} 48
Figure 3.9	Test Decision Map for f_7 49
Figure 3.10	Test Decision Map for f_4 50
Figure 3.11	Test Decision Map for f_1 51
Figure 3.12	Test Decision Map for f_{10} 52
Figure 3.13	Test Decision Map for f_5 53
Figure 3.14	Flow Chart for the Test Algorithm 60

LIST OF FIGURES (continued)

		Page
Figure 5.1	Test Example	107
Figure 5.2	Test Example	107
Figure 5.3	Test Example	107
Figure 5.4	Test Example	107
Figure 5.5	Test Example	108
Figure 5.6	Test Example	108
Figure 5.7	Test Example	108
Figure 5.8	Test Example	108

ABSTRACT

The subject of this thesis is the development of concepts and techniques to be used in the testing of cellular logic arrays. In particular, the subject is to develop techniques to locate, detect, and isolate failures in cellular logic arrays.

The primary purpose of the thesis is to develop the most general fault location theory for two-dimensional cellular arrays consisting of two-input, one-output cells. The theory presented utilizes a physical basis to assume a maximum set of allowable errors and thus realistically accounts for the appearance of cell failures.

The content of the thesis is summarized as follows: First, a physical basis for the assumption of the maximum error set is given. The assumption of the maximum error set assures that the most general solution will be obtained. Second, the problem of testing a two-dimensional cellular array is expressed in terms of testing a Maitra cascade. Third, a necessary and sufficient condition for the location of a single error in a Maitra cascade is given and proven. Also, a least upper bound on the number of tests needed to test the array is derived. Fourth, an algorithm is given, based on the necessary and sufficient condition, for error location, which allows testing of arrays for either location or detection of errors. Fifth, error detection and error location methods are given for certain very important proper subsets of the maximum error subset. Sixth, examples are given that 1) illustrate the various error location and detection methods that were proposed, and 2) illustrate that location of some multiple errors is possible utilizing single error location theories.

Chapter 1

INTRODUCTION TO THE TESTING OF INTEGRATED
CELLULAR LOGIC ARRAYS

1.1 Introduction

Testing of complex integrated cellular logic circuits fabricated using LSI techniques has become a source of concern to users and manufacturers. Since no economically feasible solution to testing problems is visible for the complex arrays contemplated for the near future, manufacturers have acknowledged the seriousness of this problem. Currently some observers believe that LSI cannot be tested because general procedures for testing and diagnosing digital circuits are applicable to rather small networks of approximately 30 gates, while cellular arrays are contemplated as containing hundreds or thousands of gates on one chip. However, if arrays are constrained to be in a cellular form, then testing problems can be simplified and test schedules can be produced which utilize the interconnection structure of cellular arrays.

In some cases, the iterative interconnection structure of cellular arrays enables derivation of test schedules which also exhibit an iterative nature, thus reducing the complexity of the testing problem in comparison to testing problems encountered in testing a non-iterative structure containing an equal number of gates. It will be shown that the structure of single-rail cascades can be utilized to great advantage in derivation of test algorithms for Maitra and general function cascades {10} and that this testing can be accomplished from the edge of the cascade. These results are extendable to a large class of arrays. However, Kautz {8, 9} has shown that there exist cellular arrays which cannot be tested from their edge terminals.

1.2 Problem Definition

The iterative interconnection structure of cellular arrays allows decomposition of testing problems for LSI cellular arrays into several sub-problems. One sub-problem is testing of single-rail cascades such as shown in Figure 1.1. These cascades can be utilized in the production of more complex cellular arrays and techniques can be derived such that if a single-rail cascade can be tested then certain complex arrays can be tested. Examination of problems encountered during solution of the problem of testing single-rail cascades utilizing only input and output terminals of cascades produces methods that can be utilized to test more complex arrays. Specifically, solution of problems involved in testing single-rail cascades lends insight to methods useful in testing cellular arrays from their edge terminals by computers utilizing only an average of 2 or 3 tests per cell contained in the array.

Figure 1.2 indicates the manner of construction of an important class of cellular arrays. An example of an important class of arrays that has this interconnection structure is a cutpoint array { 12 } ; however, cutpoint arrays do not have a buss running lengthwise through each vertical cascade. This array consists of collector rows and vertical cascades. There is a buss running the length of each vertical cascade and all busses extend across all collector rows, while busses distribute every variable to every vertical cascade. This construction reduces testing this array to testing a single-rail cascade since each collector row can be tested as a cascade and each vertical cascade can

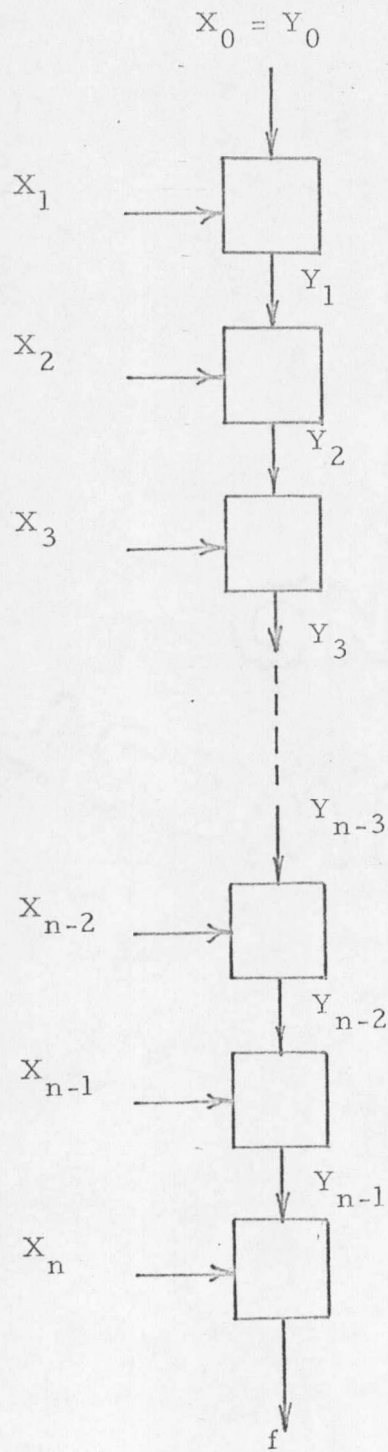


Figure 1.1. Interconnection Structure of Cascades.

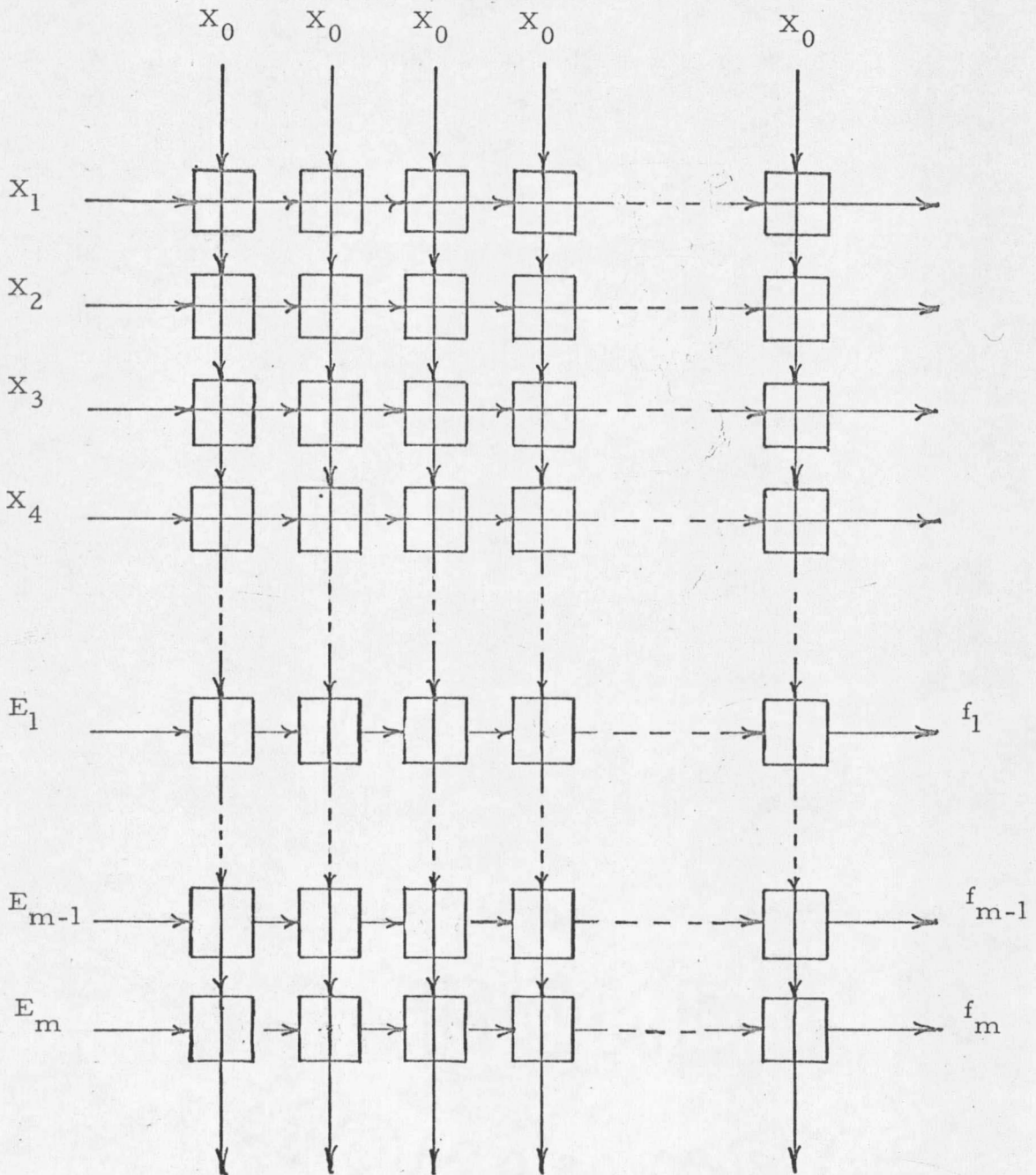


Figure 1. 2. Construction of a Testable Cellular Array.

be likewise tested. One problem in this approach is that the vertical busses must be guaranteed to be error-free or alternatively the ability to place both a 0 and a 1 on the input into the busses that extend across the collector rows must be guaranteed. Output values of vertical cascades are measured at the bottom of the array; whereas, collector row output values are measured on the right-hand side of the array.

1.3 Practical Considerations

Testing LSI circuits is a potentially difficult task; however, circuits of today are tested and they presented complex problems a few years ago. A consideration of testing problems currently solved leads to a conclusion that possibly many problems inherent in testing of LSI arrays have been solved while reaching for solutions to test problems caused by printed circuit boards and today's integrated circuits. Today's complex printed circuit boards may become tomorrow's elementary LSI chips. Consideration of testing problems produced by LSI chips may help develop test algorithms that could be used to test today's complex printed circuit boards. However, complex cellular arrays in practice will be harder to test than printed circuit boards. Consider that not only must exact error locations be indicated, but a decision must be made based on the number of errors and their locations as to what can be done with imperfect arrays. Are imperfect arrays discarded or can they be salvaged in some manner? Minnick {11} and Spandorfer {16} have suggested that at predetermined intervals in arrays such as in Figure 1.2,

extra vertical cascades and collector rows be installed. If a vertical cascade or collector row has an error, then the extra cascade or row could be used to produce the correct function.

Before any test procedures can be established an error or circuit failure criterion must be established which allows definition of possible error types that may appear in LSI construction. In Chapter 3, a logical method of deciding on an allowable set of errors for certain types of cellular arrays will be presented.

Placing an accessible test pad on an interconnection between cells reduces the effective area usable for the cells. For this reason attempts should be made to accomplish all testing and location of faulty cells from the terminals of the array without any test pads being included in the array. Actual testing of arrays is to be accomplished utilizing a computer. A test schedule could verify the complete truth table, transfer function, or state table for any given device; however, this procedure would require too much computer time and add greatly to the expense of the array. Instead of a complete verification procedure, another solution could be to test certain input conditions on a probabilistic or expected utilization basis; however, this method is still very unsatisfactory. A feasible approach is to decide on a dominant failure mode from which a set of allowable errors can be derived for each cell type used in arrays under consideration. With this knowledge, manufacturers could construct arrays utilizing certain interconnection structures and design cells with redundant properties causing an increase in the proba-

bility that if a failure occurs which was one of the dominant failure types, then the cell error that occurs is a cell error which is contained in the set of allowable errors.

1.4 Testing

Redundant design, failure modes, allowable errors and required confidence level contribute to determination of the number of tests required, but the array's structure can determine the number of tests almost independently of these factors. To test arrays of the type shown in Figure 1.2, the most complex array to be tested is a single-rail cascade. Admittedly, it would be desirable to test all collector rows (vertical cascades) simultaneously; however, in order to accomplish this restrictions on the array structure must be made that restrict the class of testable arrays until the procedure becomes practically useless.

In Figure 1.1, cell n is tested first, then cell $n-1$, etc. If an error appears in cell $n-j$, its propagation may be stopped by one of cells $n-1, n-2, \dots, n-j+1$, thereby enabling cell n to be tested. Once cell n is tested, it may be set such that it transmits the output of cell $n-1$ to the output terminal of the cascade. In this manner (under certain error assumptions) the cells may be tested in the following order until error location results: $n, n-1, \dots, 1$. It is shown in Chapter 3 that the maximum number of tests needed to locate errors under certain assumptions is a linear function of n , namely $2(n+1)$, where n is the number of cells comprising the single-rail cascade. Hence, assuming that all

vertical cascades are tested implies the number of tests does not exceed $p(2(n+1)) + 2(p+1)$ (where p equals number of vertical cascades and n equals number of cells per vertical cascade) to test an array with one collector row. If there are m collector rows, i.e., array produces m function of $n+1$ variables, then verifying complete truth tables of m functions of $n+1$ variables would require not more than $p(2(n+1)) + m(2(p+1))$ tests. A considerable savings in the number of tests is noted due to construction and structural interconnections of the array of Figure 1.2 if n is large. It is noted that the number of tests was proportional to the number of input pins.

1.5 Generation of Tests and Test Equipment

Test schedules are constructed to verify whether each cell is producing its specified function. This method of testing was chosen in preference to verifying an array's truth table because in general the number of tests needed is less than $m(2^{n+1})$, where m functions of $n+1$ variables are produced. Under certain assumptions choosing test schedules capable of accomplishing the task of locating every error in arrays such as shown in Figure 1.2 is plausible (see Chapter 3) and these test schedules can be programmed for testing utilizing digital computers. Iterative structures of cellular arrays simplifies problems connected with detection of faults. Certain forms of exhaustive test schedules are practical and can be implemented on computers for a small number of variables.

Since test schedules can be programmed in the case of single-rail cascades, computers will be able to test many types of arrays with very minor software input changes. In particular, for the single-rail cascade under the assumptions of Chapter 3, a general fault location program can be written. In order to test a cascade, the only needed input information would be the cell types and their location in the cascade. With this information the general program is able to test all cascades of one type. When the type of cascade changes this information can be given the computer as input data and all cascades of the new type can then be tested. Due to the structural interconnection of arrays shown in Figure 1.2, no reprogramming of the test computer is needed when a new type of array appears. Structure of the test computers is determined by structural complexity of arrays.

Computer testing of arrays could be very costly if a large number of tests are required. If the number of required tests is large enough, it is conceivable that computer time could be the largest cost factor in production of cellular arrays. A procedure of exhaustive testing such as verifying complete truth tables of every function in an array will be very costly if the number of variables is large; however, a high level of confidence can be placed on arrays so tested. Alternately, if required tests were few in number, then computer time could be kept from becoming such a dominant cost factor. Use of computer testing could allow test time to be less than handling time and thereby reduce prices.

Conceptually, the actual test system contains several testers, one of which is an initial tester to determine if an array has an error. If it does have an error, then it is tested by a computer specifically designed for error location; otherwise it is prepared for shipment. Once the computer has found all errors, the array is placed in a computer which attempts to correct and reroute the logic. If the array is correctable it is corrected and prepared for shipment. All other arrays are checked by a fourth computer which determines if the array is salvagable or not. An array will be salvagable if it can be used to produce non-trivial functions. If there is a large enough number of errors in an array and these errors have considerable effect on performance, then it might be advisable to consider using the array to produce functions other than those originally specified for the array.

1.6 Conclusions

Although prior work on actually testing cellular arrays is restricted to very few papers, noticeable progress has been made on the theoretical aspect of fault detection and location in cellular arrays. It is seen that test schedules for cellular arrays can test complete arrays with a number of tests proportional to the number of input terminals to the array. Detailed consideration of the structure of arrays can yield significant concepts enabling derivation of very short test schedules for cellular arrays. Utilizing computers and testing from the array's pins seems to be a solution to the testing problems of cellular arrays worth considerable detailed study.

Chapter 2

SURVEY OF PUBLISHED WORK IN THE FIELD OF
FAULT DETECTION

2.1 Introduction

As the title suggests, the purpose of this chapter is to survey work done in the field of fault detection. However, it is to be noted that the work to be reported differs from previous work in that previous investigators were interested in fault detection and not in fault location.

The relation of the previous problems studied to the problem considered here is that one must ascertain that a failure exists in order to locate it. Most previous investigations have only considered determining whether a circuit contains a fault and not where the fault is located. Most of the work surveyed allows only two error types; i.e., s-a-1 (stuck at 1) and s-a-0 (stuck at 0); whereas, this work allows more than two error types.

2.2 General Strategy

The general strategy employed in most of the papers studied is to assume a certain set of allowable failures and to allow only one element to fail. The system in question is then analyzed utilizing a fault table. A test is said to detect a fault if and only if the output of the system differs from the correct theoretical output of the system.

A test is any combinational input. Analysis of the circuit allows the fault table to be constructed as follows: Rows of the table will correspond to a possible test whereas columns of the table will correspond to a possible fault. A table with every possible fault and test is

constructed such that a 1(0) at the intersection of a row and column corresponds to detection (failure to detect the given fault by the corresponding test) of the given fault by the corresponding test. After the table is constructed a minimal set of rows is selected such that this set of rows has a 1 in every column. One notes that this set of tests generally gives some indication of where the error is located and in some cases could lead to error location.

2.3 Some Diagnostic Techniques

2.3.1 Boolean Graphs

Galey et al, { 5 } consider the Boolean graph method. In general, a Boolean graph is a set of nodes together with several input and output lines. Each input line corresponds to a variable whereas each output line corresponds to a function. These building blocks are connected together to form a system with no feedback. The functions in the system have defined for them on and off arrays, a procedure is established which allows one to construct the injective word of the graph, and derive a set of tests for the graph.

Galey et al. consider the problems of finding whether a circuit had an element s-a-0 or s-a-1 and then of isolating this element if possible. (Isolation to a single element is fault location; however, there exist circuits in which location is impossible in certain cases. For example, if any gate in Figure 2.1 is s-a-1, then the error cannot be located.)

In general, the procedure is to express the set of all possible tests for any element in terms of the primary inputs of the circuit. This is accomplished by means of starting at the output of the circuit and working back to the inputs of the circuit. The totality of inputs that are capable of distinguishing between a good and bad machine are then calculated. Some of the tests may also give fault isolation or location depending on the circuit under consideration.

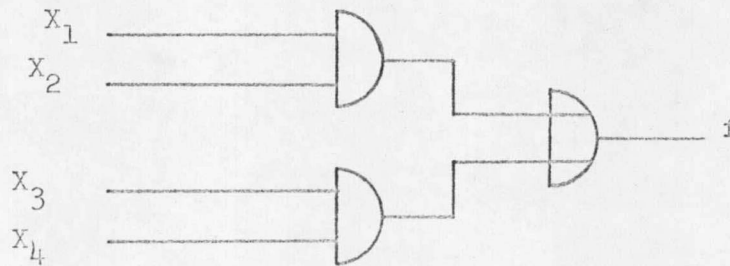


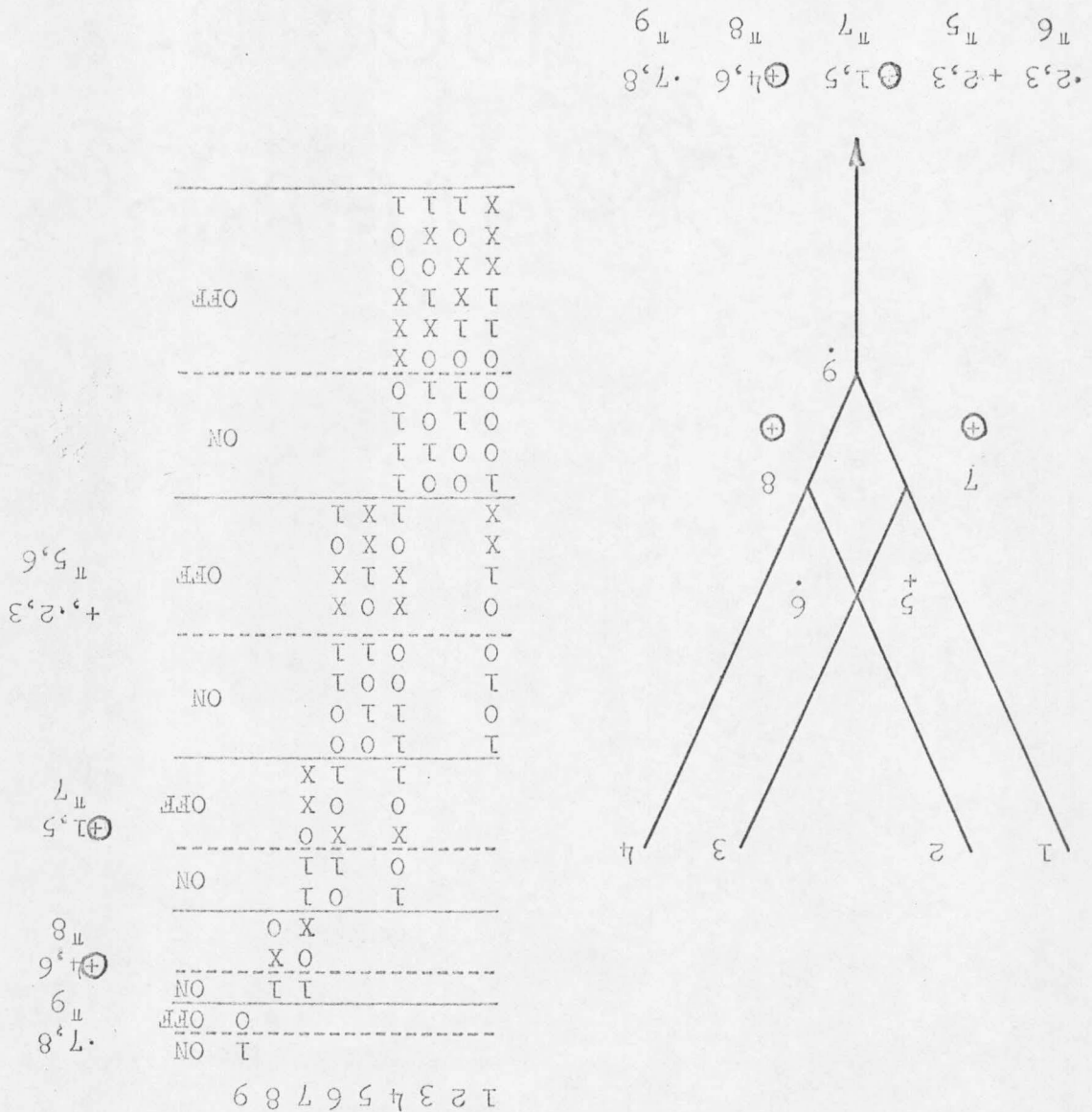
Figure 2.1. Circuit in Which a Gate s-a-l Cannot Be Located.

The following is an outline of the general strategy of Galey et al.:

A Boolean graph is a set of nodes along with input lines and output lines. Each input (output) line has an argument (Boolean function) attached. The output lines of some nodes could be the input lines of other nodes.

First the injective word, on, and off arrays of the graph are constructed. The on array for a node is the values for which the function is 1; i.e., 1 AND 2 has the on array 11 and off arrays OX and XO, where X stands for a don't care condition. The injective word is just a list of the injection operators; i.e., a list of the functions in the graph (read from right to left) where the rightmost (leftmost) entry is the function of the node nearest the output (inputs). Figure 2.2 is an example of the construction of the on and off arrays from a Boolean graph and its injective word taken from { 5 }.

Figure 2.2. A Boolean Graph, Its Inflective Word, and Its Analysis.



⊕ 7, 8 ⊕ 4, 6 ⊕ 1, 5 ⊕ 7, 8
 " 6 " 5 " 7 " 8 " 9

		X	X	X	X	X	X	X	X
		O	O	O	O	O	O	O	O
		1	1	1	1	1	1	1	1
		0	0	0	0	0	0	0	0
ON	OFF	1	1	1	1	1	1	1	1
		X	X	X	X	X	X	X	X
		O	O	O	O	O	O	O	O
		1	1	1	1	1	1	1	1
		0	0	0	0	0	0	0	0
ON	OFF	1	1	1	1	1	1	1	1
		X	X	X	X	X	X	X	X
		O	O	O	O	O	O	O	O
		1	1	1	1	1	1	1	1
		0	0	0	0	0	0	0	0
ON	OFF	1	1	1	1	1	1	1	1
		X	X	X	X	X	X	X	X
		O	O	O	O	O	O	O	O
		1	1	1	1	1	1	1	1
		0	0	0	0	0	0	0	0
ON	OFF	1	1	1	1	1	1	1	1

⊕ 7, 8 ⊕ 4, 6 ⊕ 1, 5 ⊕ 7, 8
 " 6 " 5 " 7 " 8 " 9

Next, the set of tests that distinguish a machine that is good from a machine in which the i^{th} branch is s-a-1 is computed. (A similar procedure holds for s-a-0 in the i^{th} branch.) Consider the i^{th} node as an input and ignore the subgraph feeding it. This forms a new graph with a new injective word. Compute the on and off arrays for this new graph deleting all on (off) arrays with a don't care condition in position i . For every cube in the on array with a 1 in position i , change the 1 to a 0 and intersect this cube with all cubes in the off array. This set (denoted $P(0,1)$) represents the totality of inputs for which the good machine gives a 0 output and a bad machine (node i s-a-1) gives a 1 output. This set must now be translated into a primary input representation. To do this, one considers the subgraph feeding node i and finds the set of all inputs that produce a 0 at node i .

Galey et al. then give the "Algorithm for Computing Totality of All Tests." This algorithm may be simply explained as follows:

A matrix is constructed in which a column is labeled for each possible s-a-l failure and a row is labeled for each test. A 1 is placed in row i column j if and only if failure j is detected by test i . Two matrices are constructed in this manner; i.e., one for s-a-l faults and one for s-a-0 faults. The remaining problem is to select a minimal test set which covers both matrices.

An example problem, which consists of testing an eight-bit parity-check circuit for s-a-l and s-a-0 failures, is worked {5} by Galey et al. The s-a-l and s-a-0 matrices each had four rows and 102 columns. Four tests were needed to detect any failure in the circuit.

This algorithm could be applied to cellular cascades with only two faults allowed; however, since the Corollary to Theorem 4.1 of Chapter 4 of this thesis shows that at most two tests are needed to detect a cascade which has faults f_0 (s-a-0), f_{15} (s-a-l), and f_{15-p} (where f_p is the cell function) the

application of the above algorithm would not seem logical considering its complexity.

2.3.2 Armstrong's Method

Armstrong [1] describes a procedure for detection of s-a-1 and s-a-0 faults in gate networks with the following gate functions: AND, OR, NAND, NOR, and NOT. Armstrong's method is best suited for large circuits in which the fault table methods are unwieldy due to the size of the fault table.

Armstrong calculates the equivalent normal form for the circuit. Since the equivalent normal form is a sum of products form, the circuit now corresponds to a hypothetical circuit of AND gates feeding an OR gate. Utilizing the concept of path sensitizing, an algorithm is described which will allow one to obtain a set of tests which detects faults in the equivalent normal form. It is shown that this set also detects all faults in the original circuit.

The following is an outline of Armstrong's method:

The equivalent normal form (enf) of the circuit is derived. This is done utilizing two criteria: (1) The identity of every path from the input to the output is maintained, and (2) no redundant terms are discarded. In the enf

if input X goes through gates 1,6,12,19, 20, and 21 in one path from input to output, a term will appear in the enf with $X_{1,6,12,19,20,21}$ as a literal involved.

Next, two tables are formed: one for the enf and one for the complement of the enf. These two tables are used to derive s-a-1 and s-a-0 tests. One notes that a s-a-0(1) test for a literal in the enf is a s-a-1(0) test for the corresponding literal in the complement of the enf. Armstrong's method selects s-a-1 tests for both the enf and its complement. Both tables are constructed as follows: Row 1 gives the enf, row 2 gives the score from the s-a-1 scoring function, row 3 gives the order of all literal appearances based on the highest (second highest) score being number 1(2), and the remaining rows contain the binary values assigned the literals by the s-a-1 tests that are generated.

To generate the s-a-1 tests: (1)
The literal appearance with the highest

score is selected (from either table) and assigned the value 0. (2) All other literals in the same term of the enf with the preceding literal are assigned the value 1. (3) Every other appearance of these literals are assigned the appropriate value of 0 or 1. (4) One now checks to see if any values must be "forced" to stop the test from failing. (A test fails if the previous assignment causes another term to have all 1's assigned to its variables). (5) Once all literals have been assigned the test is complete. (6) All literals tested by a test are labeled. (7) The highest scoring literal (that is not labeled) is selected and a second test is constructed. (8) This process continues until all literals are tested.

The scoring function is given for reference. It is derived on the basis of the following two properties: A term has the fewest number of literals (property A) and a literal in that term that has the largest number of complimented forms

appearing in other terms (property B).

This scoring function seems to give about the appropriate weights to properties A and B to enable derivation of near minimal test schedules with much less effort than needed by the fault table method.

$$(S_{c_1})_k = (1 - \frac{V_j}{V}) + \frac{\lambda_k}{L}$$

where

$(S_{c_1})_k$ = the s-a-l score for the k^{th}

literal of the enf,

V_j = the number of variables in the

j^{th} term of the enf, where the

k^{th} literal is in the j^{th} term,

V = total number of literal appear-

ance in the enf,

$$\lambda_k = \begin{cases} A_i, & \text{if the } k^{\text{th}} \text{ literal is un-} \\ & \text{primed} \\ A_i, & \text{if the } k^{\text{th}} \text{ literal is} \\ & \text{primed} \end{cases}$$

A_i = number of unprimed appearances

of the i^{th} variable in the enf, and
 A_i = number of primed appearances
of the i^{th} variable in the enf.

One notes that if literal k is
being considered $A_i (A_i')$ for literal
 $i=k$ is used. The scoring function
would probably be clearer if A_k were
used in its statement instead of A_i .

Armstrong works an example and compares it to the same
example worked utilizing fault table methods. The minimal
test set has six tests; whereas, Armstrong's method calculates
eight tests.

This algorithm could be applied to cascades with cell
functions of AND, OR, NAND, NOR, and NOT; however, utilizing
the Corollary to Theorem 4.1 of Chapter 4 of this thesis one
could produce the minimal test schedule with much less effort.

2.3.3 Redundant Circuits

Friedman [4] considers fault detection in redundant
circuits of the same type as those considered by Armstrong.
Several examples are presented which illustrate the problems
encountered in redundant circuits; however, no algorithm for
testing redundant circuits is given. Friedman does state that
"With redundancy present, it is necessary to verify that every

test remains valid if preceded by any sequence of undetectable faults," where "test" means a detection test derived by single-fault analysis techniques.

2.3.4 Kautz's Method

Kautz [7] considers the general problem of fault testing. He presents several solutions to the problem of fault testing. Fixed test schedule procedures and serial test procedures are both considered.

For the fixed test schedule problem, a specific set of errors is not assumed; instead, it is assumed that the network can produce certain erroneous functions. Since no errors have been specified, Kautz should assume that any of the 2^{2^n} functions of n variables could be produced, but with this assumption the test schedule would have 2^n tests. Kautz does not say what the total number of functions are, but that they can be reduced to m distinguishable erroneous functions. A fault matrix F consisting of the correct function in column 1 and the m possible erroneous functions in columns 2, 3, ..., $m+1$ is constructed. Next, as many rows as possible are deleted from the fault matrix subject to the condition that every column is distinguishable. This gives a minimal test schedule for detecting all the erroneous functions.

Kautz shows how serial testing may be generalized from fault testing with fixed test schedule procedures. Actually, the tests are always applied serially, but these are derived serially. The algorithm is as follows: Given the fault matrix F (described in the preceding paragraph), delete the row p where the number of $(0,1)$ pairs are maximum; split F into F_0 and F_1 , where $F_0(F_1)$ is the set of columns having 0's (1's) in row p ; repeat the algorithm utilizing the previous results.

In order to effectively utilize Kautz's method to test cellular arrays, one would have to translate the maximum error set into a maximum set of possible erroneous functions; however, this would be an exceedingly difficult task. Since Theorems 3.1 and 3.2 give the minimal test schedule for any cellular array considered in this thesis, it would not seem logical to utilize Kautz's method to test for faults in these cellular arrays.

2.3.5 An Adaptive Solution

Cohn and Ott {3} explain an adaptive approach to fault detection. The problem they solve is design of minimum-expected-cost testing procedures for both detection and isolation of single failures. In certain special cases the problem is related to the design of a minimum redundancy binary code for a source whose messages are statistically independent (Huffman {6}) and to the problem of deciding which test to

omit and the sequence in which the remaining tests are performed in a tree structure with one limb where the number of nodes is less than the number of elements in the system. The problem formulation is given here because it is a very interesting way in which to consider fault testing. The system is to consist of parts which may be single elements or modules. The elements are defined so that each element has associated with it a probability of failure with the probability of multiple failures assumed to be negligible, while each test has associated with it the cost to perform the test. The problem is to find a minimum cost test schedule. This is a very complex approach to fault detection and location; however, it would seem to be the most unique serial type test concept available at present. The algorithm is very good if one is interested in test procedures in which there are probabilistic failure assumptions. The algorithm proceeds as follows:

The condition no fault is replaced by an element S_0 with all 0's in every test vector in the 0^{th} component. The elements are arranged in rows with all components in a row having the same number of elements in each set. For example:

	(0 1 2)	
(0 1)	(1 2)	(0 2)
(0)	(1)	(2)

The evaluation E_D of a subset D is

$$\min (p_D C_{A,B} + E_A + E_B)$$

where $D=A \cup B$, $A \cap B = \phi$, p_D is the sum of probabilities of elements in D, and $C_{A,B}$ is the cost of the least expensive test partitioning D into non-trivial subsets A and B. The subsets in the lowest row have $E_D=0$ since they represent single elements and thus have no ambiguity. From the fact that for every element in the lowest row $E_D=0$ and the expression E_D , the array can be evaluated and a test schedule found.

2.3.6 Programmed Algorithms

Roth et al. [15] have programmed two algorithms for fault detection in combinational logic circuits. Roth [5,13-15] has written several papers on fault detection in combinational logic circuits. The two algorithms programmed in [15] are based on the other papers Roth published on fault detection problems.

The first algorithm is DALG-II. DALG-II is a test generation procedure which guarantees finding a test for a failure (s-a-1 and s-a-0 are the allowable failures) if such a test exists. The second algorithm is called TEST-DETECT. TEST-DETECT is an algorithm to ascertain all failures detected by a given test. TEST-DETECT and DALG-II are used in conjunction with each other to produce test schedules for combinational logic circuits; however, strategies for their use together need to be worked out so that a small number of tests can be quickly generated which detect all failures in the circuit and after all faults have been detected, a larger set of tests can be generated which isolate the error to within the smallest replaceable module. Roth et al. discuss three such strategies.

The algorithms are given in the form of APL (A Programming Language, Iverson's notation) programs. DALG-II has thirteen subroutines whereas TEST-DETECT has one subroutine. Since the detection problem for three errors (s-a-0, s-a-1, complementation) is solved in a minimal manner by Corollary 1 to Theorem 4.1 of this thesis, Roth's algorithms are not reproduced.

2.4 Testing of Cellular Arrays

Kautz {8,9} and Yau and Orsic {17} have considered testing cellular arrays. Kautz looks at general arrays whereas Yau and Orsic look at cutpoint cellular arrays. The results of Kautz tend to be necessary and sufficient conditions for an array to be testable. These results are of a very general nature and no algorithm is given to determine the test schedules which have been shown to exist.

Yau and Orsic consider cutpoint arrays; however, in testing cutpoint arrays there are problems because s-a-0 (s-a-1) appear to look like cell functions 0 (1) which are allowed in cutpoint arrays. The purpose of Yau and Orsic is to locate faulty columns and collector rows. Their approach is to construct two fault matrices somewhat analogous to fault tables previously discussed.

The algorithm of Yau and Orsic is not given in this thesis for the following two reasons: (1) Since the purpose of Yau and Orsic is to locate faulty rows and columns with two allowable faults (s-a-1 and s-a-0), an application of Corollary 1 to Theorem 4.1 of this thesis gives location of the faulty rows and columns in $2(m+N)$ tests; however, Yau and Orsic's algorithm needs $2m(n+N+1)$ tests (where m, n , and N are the number of columns, input variables, and rows of the array). (2) Two $(n+N) \times m$ matrices are utilized in the algorithm and the construction of these two matrices is just one out of eight steps in the algorithm.

2.5 Digital Simulation of Failures in Digital Systems

This section is concerned with a paper by Chang {2}. In his paper Chang gives a method for simulating shorted input diode failures other than the s-a-0 and s-a-1 faults. The method is extendable to simulate all modes of failure that are describable by truth tables and Boolean algebra.

The importance of Chang's work is that while experience has shown that s-a-1 and s-a-0 are common faults, there are many types of failure modes that may occur in integrated circuits that did not occur in current discrete logic circuit technology. Also, in gate networks it is possible for shorted diode failures to occur and for that reason the problem of fault detection and location should not be restricted to consideration of only s-a-0 and s-a-1 failure models.

The method Chang utilizes is to change the theoretical truth table. For example, if the function $f_{14}(X,Y)$ can have as a possible error a condition described by shorting diode Y to 0, then this error can be described by the truth table function $f_{12}(X,Y)$.

The implication of Chang's work {2} is that as testing procedures become more sophisticated, more faults will be added to the allowable error sets; however, this trend will have no effect on this thesis since it utilizes the maximum possible error set.

2.6 Conclusion

The usual problem considered in the field of fault detection and location is the detection in an arbitrary digital network of two failure modes, i.e., s-a-0 and s-a-1. A fault table is a common solution to the problem; however, there exist other methods of solving the problem.

That most researchers concern themselves with detection of faults and allow only two failure modes to exist points out the complexities of the problems involved in this field. Location of a fault after it has been detected is in most cases an impossible problem; however, some results can be used to isolate the failure to a certain part of the system. It is interesting to note that work has been done on simulating faults other than s-a-0 and s-a-1 [2] .

In the future, more work should be done in systems in which more than two possible errors are allowed to exist, since these systems can be simulated if only by hand calculation. It is the purpose of the following sections of this thesis to attempt a solution of the problem of fault location in cellular arrays with an expanded error set and develop an algorithm that will allow one to test cellular arrays of certain interconnection structures.

Chapter 3

FAULT LOCATION IN CASCADES

3.1 Introduction

This section is a review of Chapters 1 and 2. Production of practical, reliable, and economical batch-fabricated circuits is an important trend in the field of digital circuits. However, use of these complex integrated circuit modules increases the chance that some of the components in the circuit will be faulty. Construction of these circuits in the form of a cellular array has the following advantages: uniform interconnection structure makes the circuits easy to manufacture, reduces the number of pins, and increases reliability. Since there are no test pads in the array, all testing must be accomplished using only the input and output terminals; therefore, fault location is a very important problem if cellular arrays are to be utilized in practical circuits.

The current trends in fault detection in arbitrary digital circuits are due to Armstrong {1} and Roth {5,13-15} . Both Roth and Armstrong allow only two types of errors; i.e., s-a-1 (stuck at one) or s-a-0 (stuck at zero). Kautz {8,9} has studied fault detection in many different types of cellular arrays.

This research studied a specific cellular array called the cascade. In 1962, Maitra {10} studied a very fundamental cellular array which may be described as a single-rail cascade. Cells in this array are connected as shown in Figure 1.1. Each cell in the cascades under consideration produces a function selected from the following 12 func-

tions: Y^* , $X^* + Y^*$, X^*Y^* or $X^* \oplus Y$. (See Definition 3.6, page 40.)

One notes that current work is based on fault detection on a very limited set of allowable errors whereas this research is concerned with fault location on an allowable set of errors which will contain more than two failure types.

In general, the relationship between current work and this research is that the fault must be detected before it can be located. For arbitrary digital circuits it is hard to decide on an allowable set of errors; however, in cascades there exists a natural manner in which to determine the set of allowed faults.

3.2 Assumptions

The physical structure of the cells of the cascade furnishes a basis to obtain a maximum allowable set of errors. Since every cell has two input leads it is logical to assume that if a cell fails, it produces one of the remaining 15 functions of two variables; therefore, it is assumed that every cell has 15 possible modes of failure. In accordance with assumptions implicit in previous work done in fault detection, it is assumed that the interconnections between cells do not fail, that the failure is time independent, i.e., if cell m is in error at time t_1 then cell m is still in error at time $t_2 > t_1$, the error in cell m has not changed, and that the input and output leads of the system do not fail.

It is assumed that every cell is a two-input, one-output cell and cascades are constructed as indicated in Figure 1.1, if a cascade has a

faulty cell, then only one cell may produce an erroneous function and whether a cascade is redundant or irredundant, the variables are numbered as shown in Figure 1.1.

3.3 Objectives

The objective of the research is to obtain and prove a necessary and sufficient condition (Theorem 3.1) for single failure location in cascades. A least upper bound on the number of tests needed to verify whether a cascade has a failure will be obtained (Theorem 3.2). The proof of Theorem 3.1 will be constructive and an algorithm will be obtained from this proof which will allow location of a fault in a cascade subject to the assumptions of Theorem 3.1.

3.4 General Concepts

As explained in Section 1.4, the cells will be tested in order $n, n-1, \dots, 2, 1$ as indicated on Figure 1.1. Utilizing this testing procedure, the cascade may be thought of in the following manner: a tested section (cells $n, n-1, \dots, i+1$), a cell being tested (cell i), and an untested section (cells $i-1, \dots, 1$). A set of constants $C_n, C_{n-1}, \dots, C_{i+1}$ may be chosen and applied to the cascade shown in Figure 1.1 (where C_j is applied to input X_j) such that the output of the cascade will be equal to the output of cell i (or its complement). In order to test cell i , Y_{i-1} (where Y_{i-1} is defined to be the value on the interconnection between cells i and $i-1$) must be determined. Cell 1 is a

special case because the tester has access to both $X_0 = Y_0$ and X_1 and therefore there is no uncertainty in testing cell 1.

In order to completely test cell i , the two-variable truth table for cell i must be verified. Since X_i is accessible, both a 0 and a 1 may be placed on the input lead X_i ; however, no direct access to Y_{i-1} is available.

By careful testing procedures Y_{i-1} can be consistently produced and determined (under appropriate assumptions). If C_0, C_1, \dots, C_{i-1} are applied to the inputs X_0, X_1, \dots, X_{i-1} in the untested portion of the cascade and if they theoretically set Y_{i-1} to a particular value, C , then the actual Y_{i-1} is either correct or not correct; however, if $Y_{i-1} = C$ is needed for another test in the test schedule, then C_0, C_1, \dots, C_{i-1} will be used (since the error has been assumed to be time independent, it is noted that if C_0, C_1, \dots, C_{i-1} produce $Y_{i-1} = F$ at time t_1 , then C_0, C_1, \dots, C_{i-1} produce $Y_{i-1} = F$ at any time $t_2 > t_1$) for this test. Although Y_{i-1} may not equal C , at least it has the same value as it had previously.

Assume that the same constants C_0, C_1, \dots, C_{i-1} (D_0, D_1, \dots, D_{i-1}) are used to produce every $Y_{i-1} = 0(1)$, then there will be only four possible receivable Y_{i-1} sequences, as shown in Figure 3.1.

X_i	Y_{i-1}	Desired Y_{i-1}	Possible Y_{i-1}
0	0	0	0 1 0 1
0	1	1	1 0 0 1
1	0	0	0 1 0 1
1	1	1	1 0 0 1

Figure 3.1. Determination of Y_{i-1} .

One notes from Figure 3.1. that there are only four possible Y_{i-1} sequences that can be received. These are: 0, 1, 0, 1; 1, 0, 1, 0; 0, 0, 0, 0; 1, 1, 1, 1. These four sequences come about because the $Y_{i-1} = 0$ (1) setting is either right or wrong, but if one 0 (1) is wrong, then so is the other 0 (1) because the same set of constants was used to produce both 0's (1's) and the system is time independent. Once the test scheme is understood, the problem reduces to somehow being able to account for the three possible bad sequences that are capable of being received.

The effect of this testing procedure will be illustrated. If the Y_{i-1} sequences are not produced utilizing the same settings for both 0's (1's) then since either 0 (1) could be correct or incorrect, but the 0's (1's) are not necessarily both correct or incorrect, there are 2^4 possible receivable sequences and this means instead of three incorrect sequences to be accounted for, there are fifteen. In some special cases this number is only seven incorrect sequences; i.e., the same constants

are used to produce either the 0 or the 1 value but not both, thereby giving 2^3 possible sequences. Theorem 3.1 utilizes the previously discussed test procedures and illustrates under what assumptions faults can be located.

3.5 Definitions

The following definitions are standard for the rest of the discussion on fault location.

Definition 3.1 An error occurs in a cell whenever the cell produces a function that is not the same as the function specified for that particular cell.

Definition 3.2 The set of sixteen functions of two variables is denoted G .

Definition 3.3 I_p denotes 1, 2, 3, 4, ..., p.

Definition 3.4 The error function E is a mapping from $G \times I_n$ to G , where $E(f_i, j) = f_k$ denotes that cell j was theoretically to produce $f_i \in G$, but instead it produced $f_k \in G$. Clearly, $E(f_i, j) = f_i$ indicates that cell j does not have an error occurring in it.

Definition 3.5 The cell functions are numbered as follows:

X_i	Y_{i-1}	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Definition 3.6 X^* means either X or X' but not both.

3.6 A Necessary and Sufficient Condition for Location of Faults in a Cascade and a Least Upper Bound for the Number of Tests.

It is to be noted that the allowable cell functions for a cascade are $X^* + Y^*$, $X^* Y^*$, Y^* , and $X^* \oplus Y$.

Theorem 3.1

Given a cascade (12 possible cell functions) of length n , then the error can be located if and only if for every $i \in I_n - \{1\}$

(1) $E(f_{14}, i) \neq f_{15}, f_{12}, f_{13}$

(2) $E(f_{11}, i) \neq f_{15}, f_7, f_3$

(3) $E(f_8, i) \neq f_0, f_{12}, f_4$

(4) $E(f_2, i) \neq f_0, f_3, f_1$

(5) $E(f_6, i) \neq f_9, f_{12}, f_3$

(6) $E(f_9, i) \neq f_6, f_{12}, f_3$

$$(7) \quad E(f_{13}, i) \neq f_{12}, f_{14}, f_{15}$$

$$(8) \quad E(f_7, i) \neq f_3, f_{11}, f_{15}$$

$$(9) \quad E(f_4, i) \neq f_0, f_8, f_{12}$$

$$(10) \quad E(f_1, i) \neq f_0, f_2, f_3$$

$$(11) \quad E(f_{10}, i) \neq f_0, f_{15}, f_5$$

$$(12) \quad E(f_5, i) \neq f_{10}, f_0, f_{15}$$

Proof of Theorem 3.1

The proof is an induction proof. Clearly, the theorem is true for the case $n = 1$. Assume the theorem is true for a positive integer k and consider a cascade with $k + 1$ cells. Given the cell function for cell $k + 1$, if it can be shown that the error can be located in cell $k + 1$ if and only if assumptions (1) - (12) are valid for cell $k + 1$, then the proof is complete.

Assume conditions (1) - (12). This part of the proof is now completed in Figures 3.2 - 3.13.

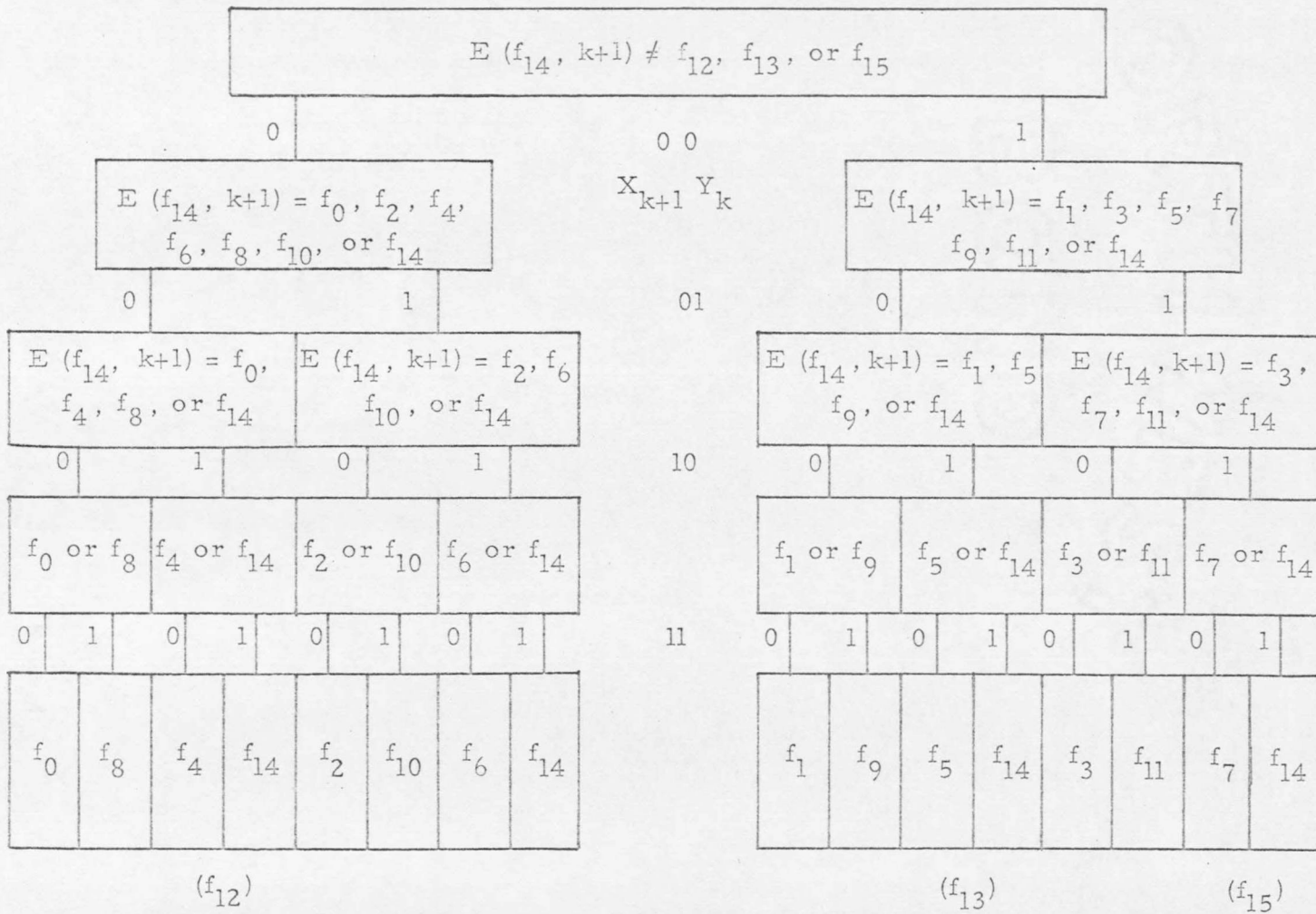
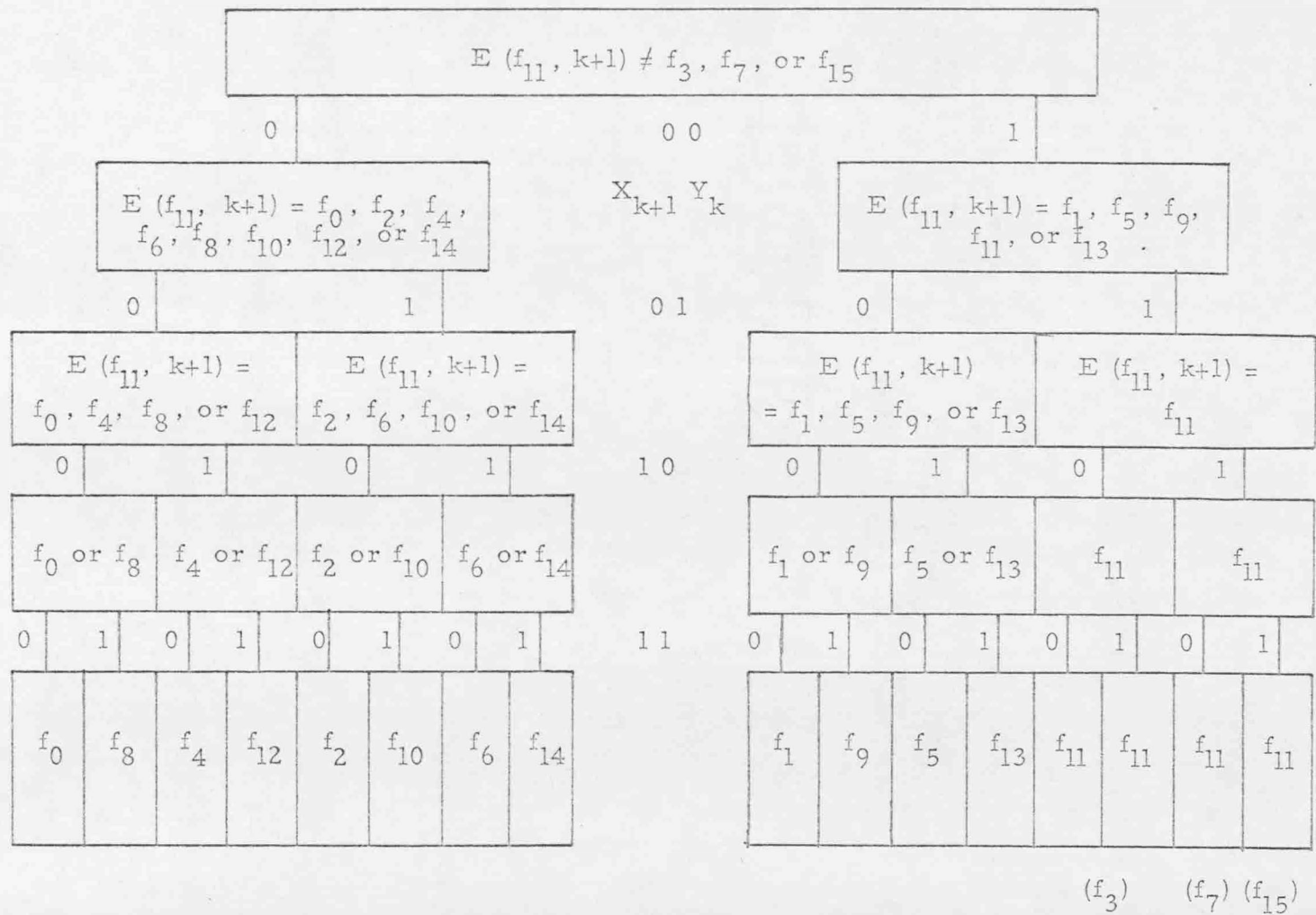


Figure 3.2. Test Decision Map for f_{14} .



-43-

Figure 3.3. Test Decision Map for f_{11} .

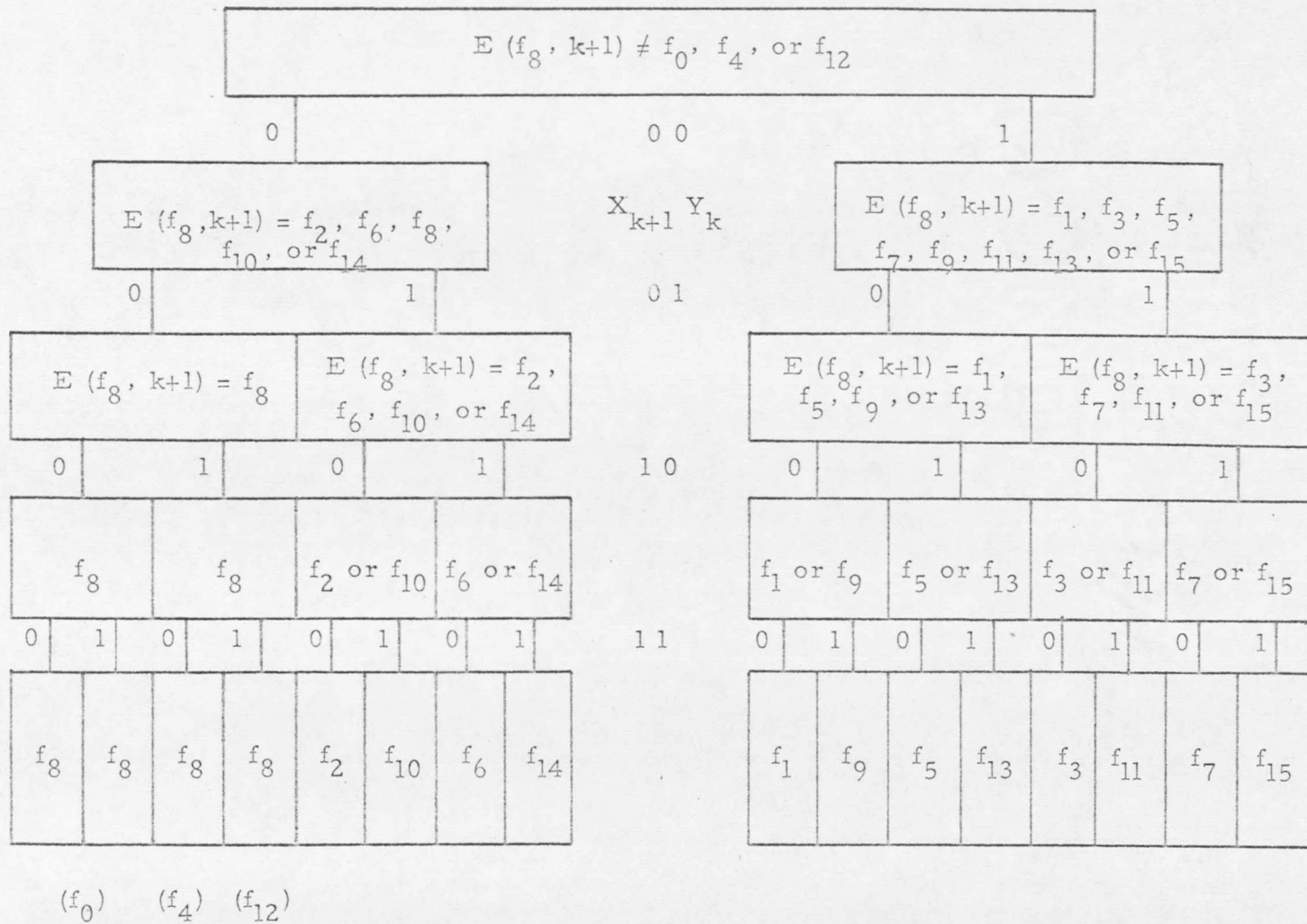
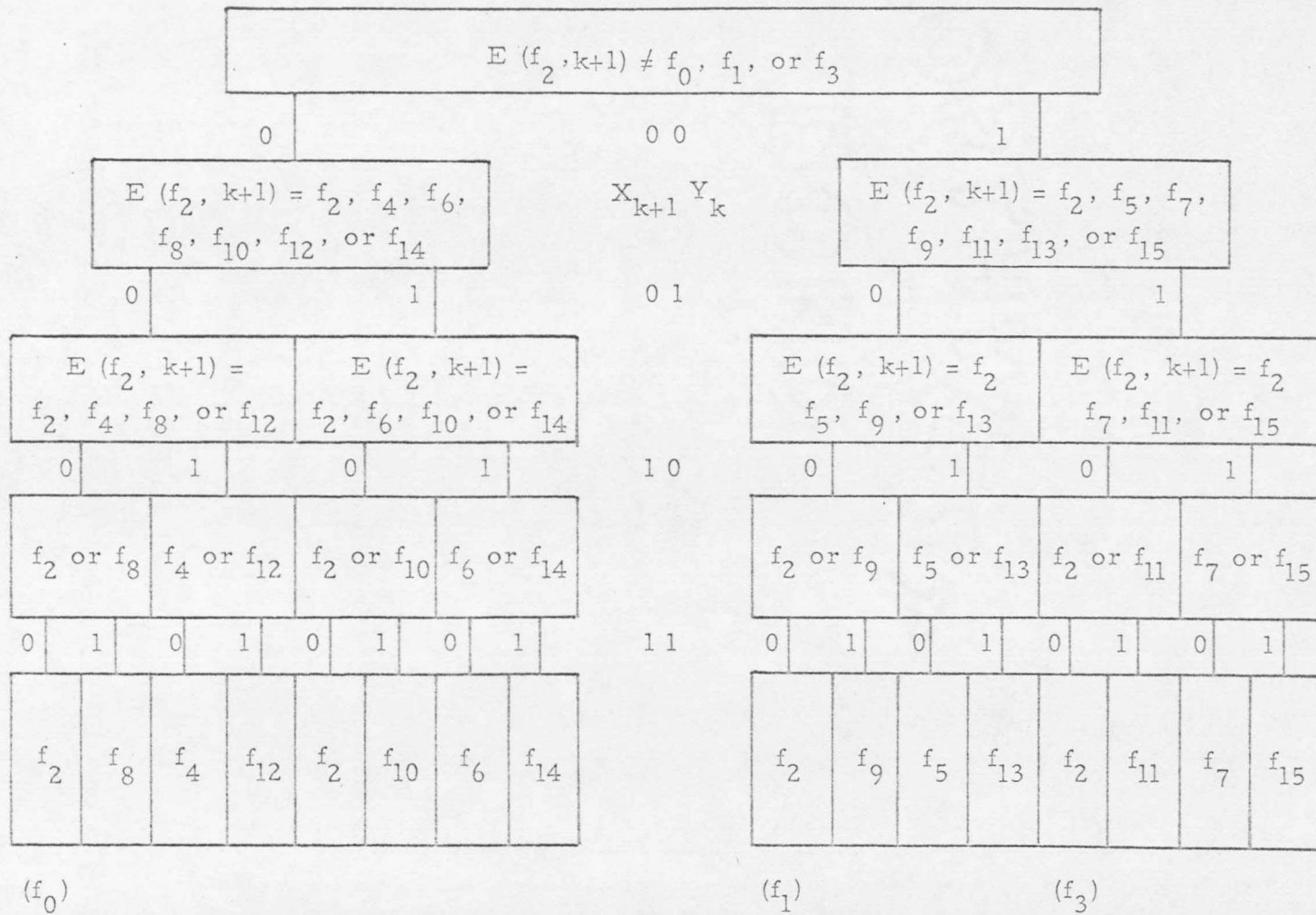


Figure 3.4. Test Decision Map for f_8 .



-45-

Figure 3.5. Test Decision Map for f_2 .

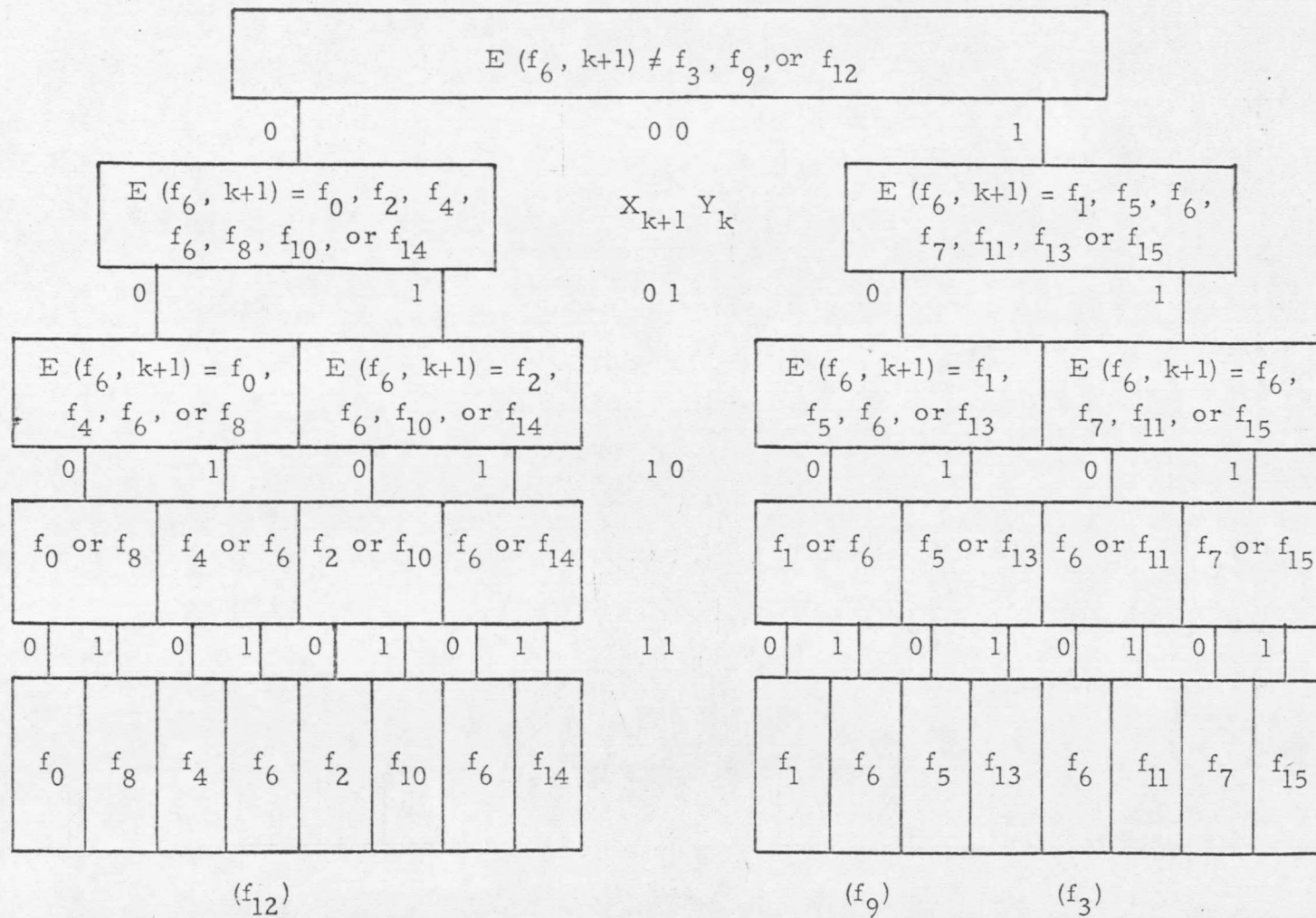
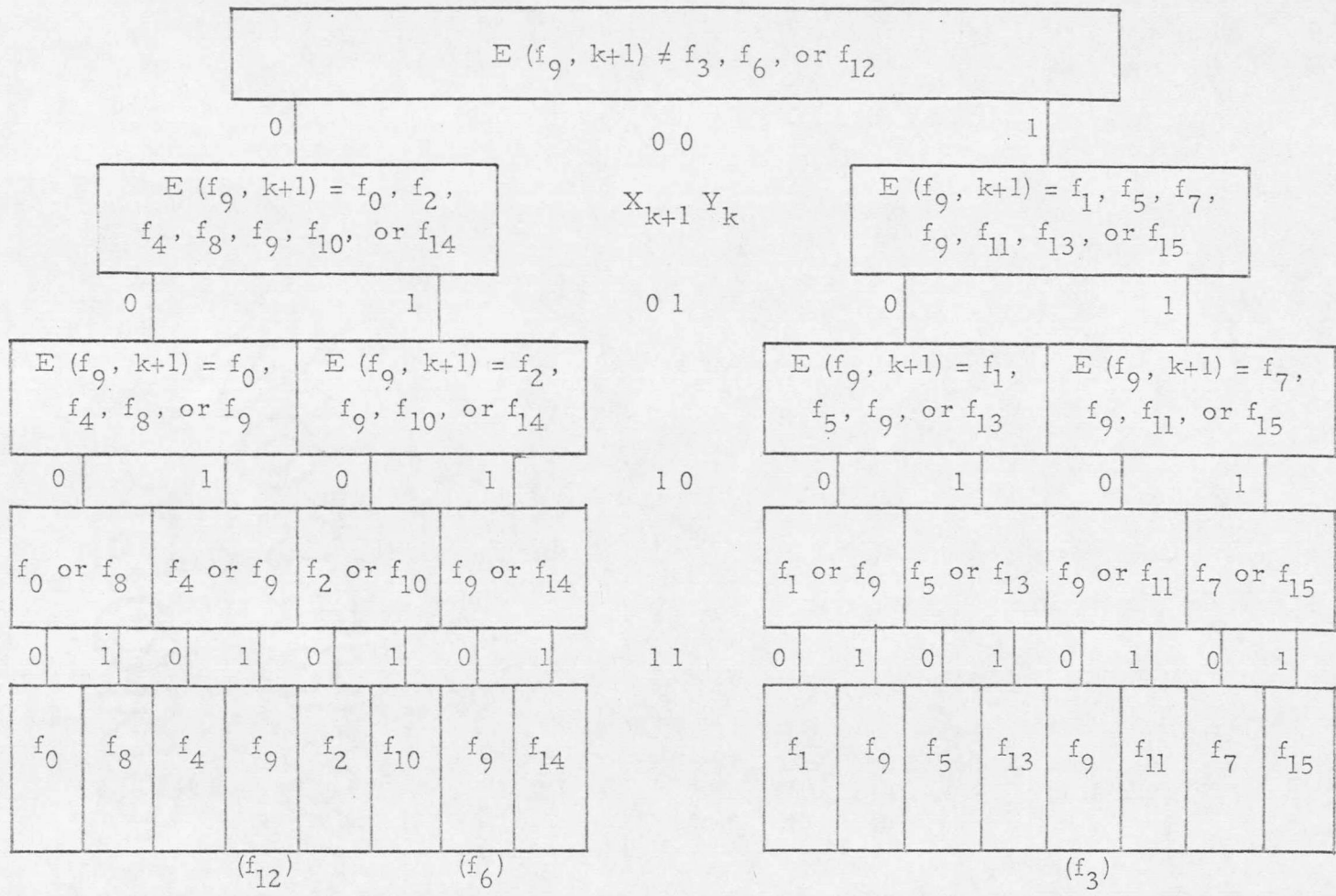


Figure 3.6. Test Decision Map for f_6 .



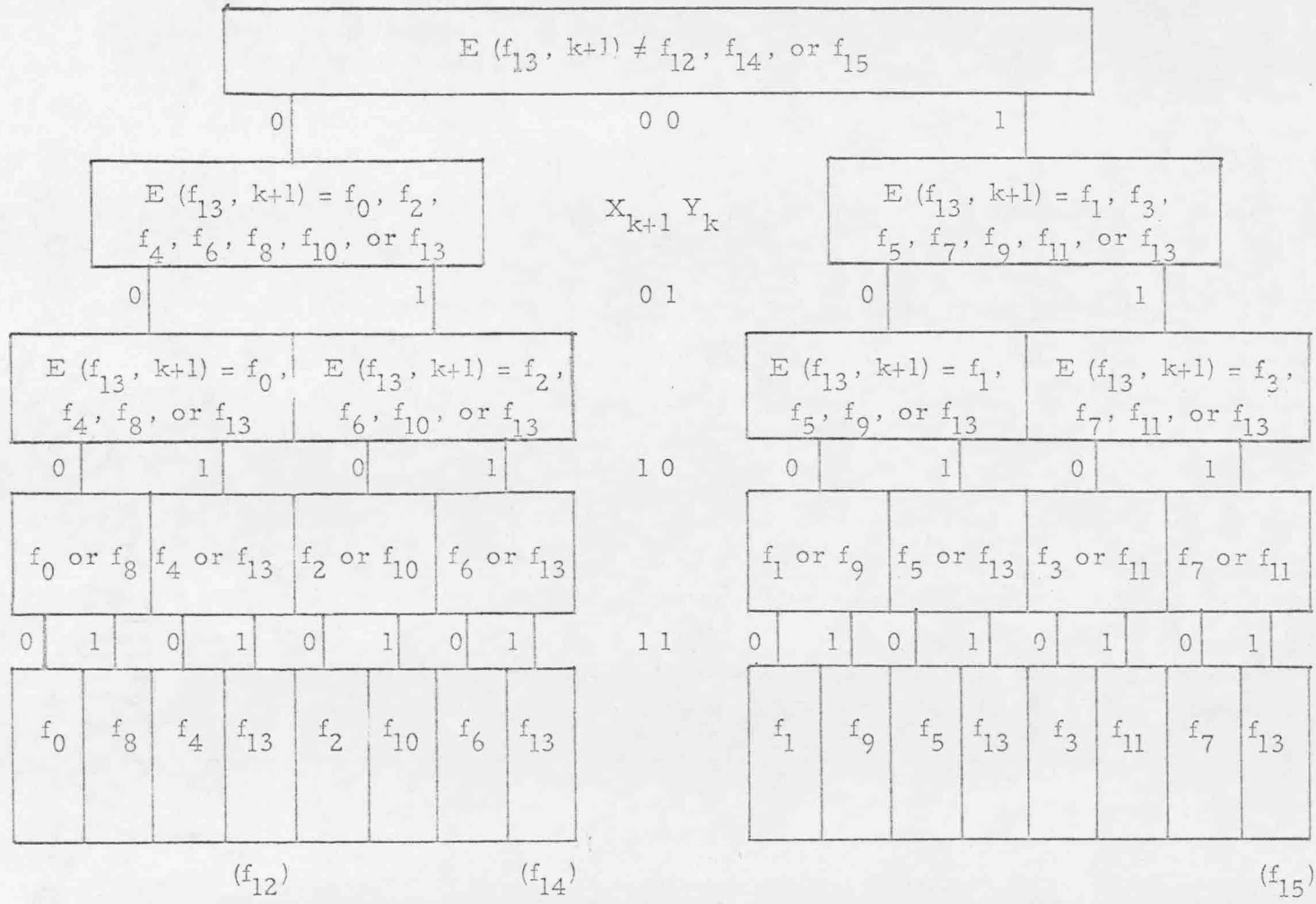


Figure 3.8. Test Decision Map for f_{13} .

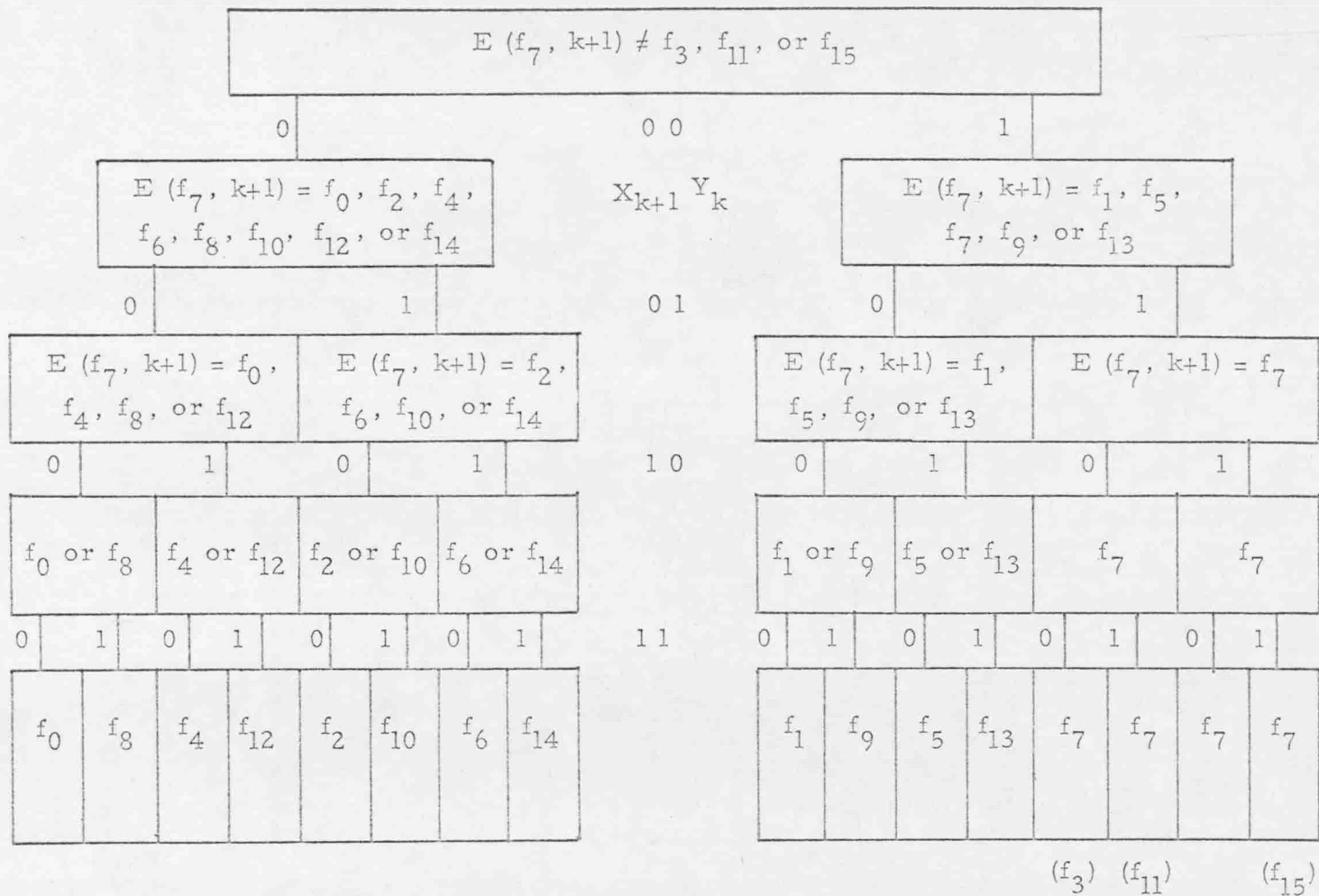


Figure 3.9. Test Decision Map for f_7 .

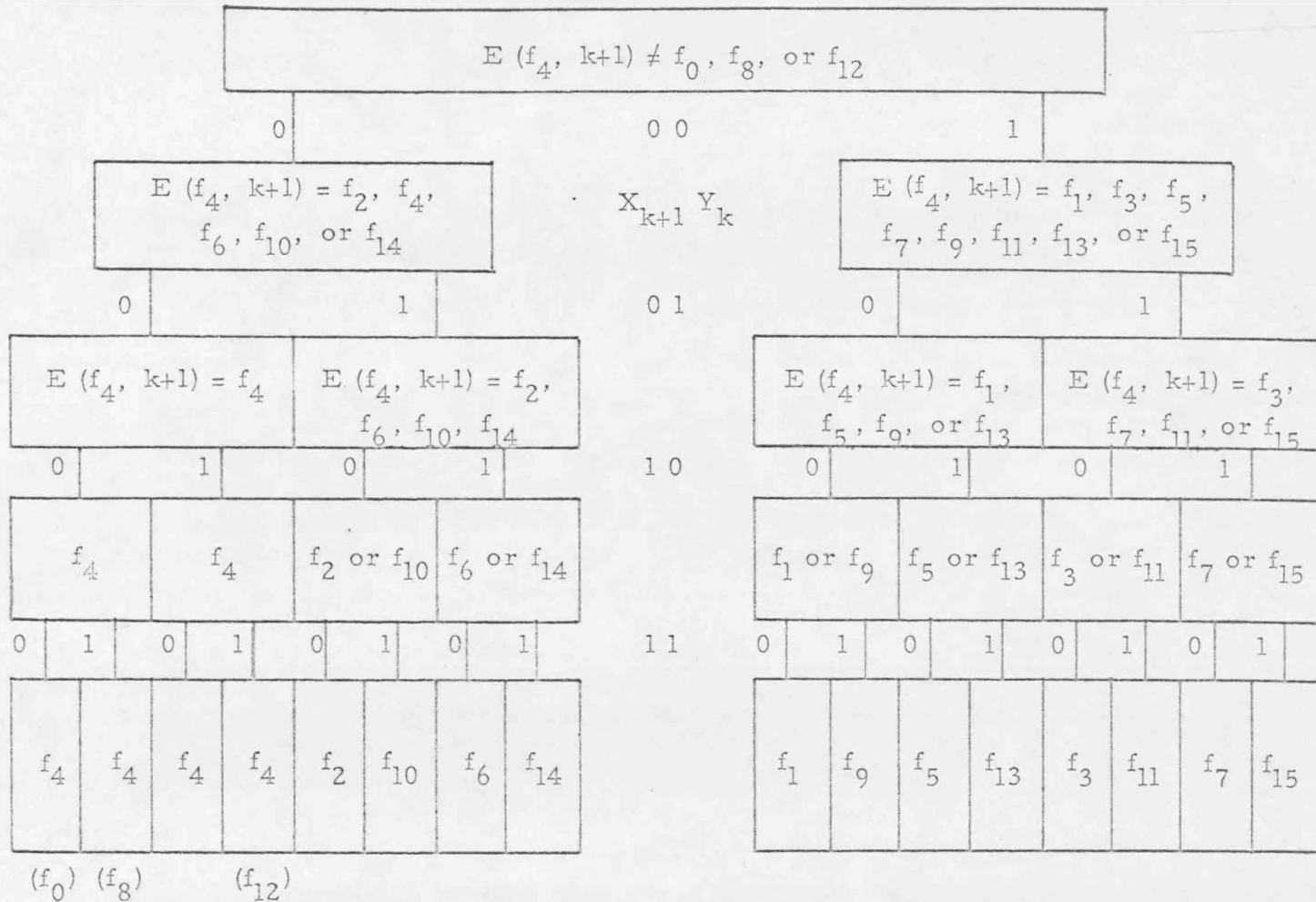


Figure 3.10. Test Decision Map for f_4 .

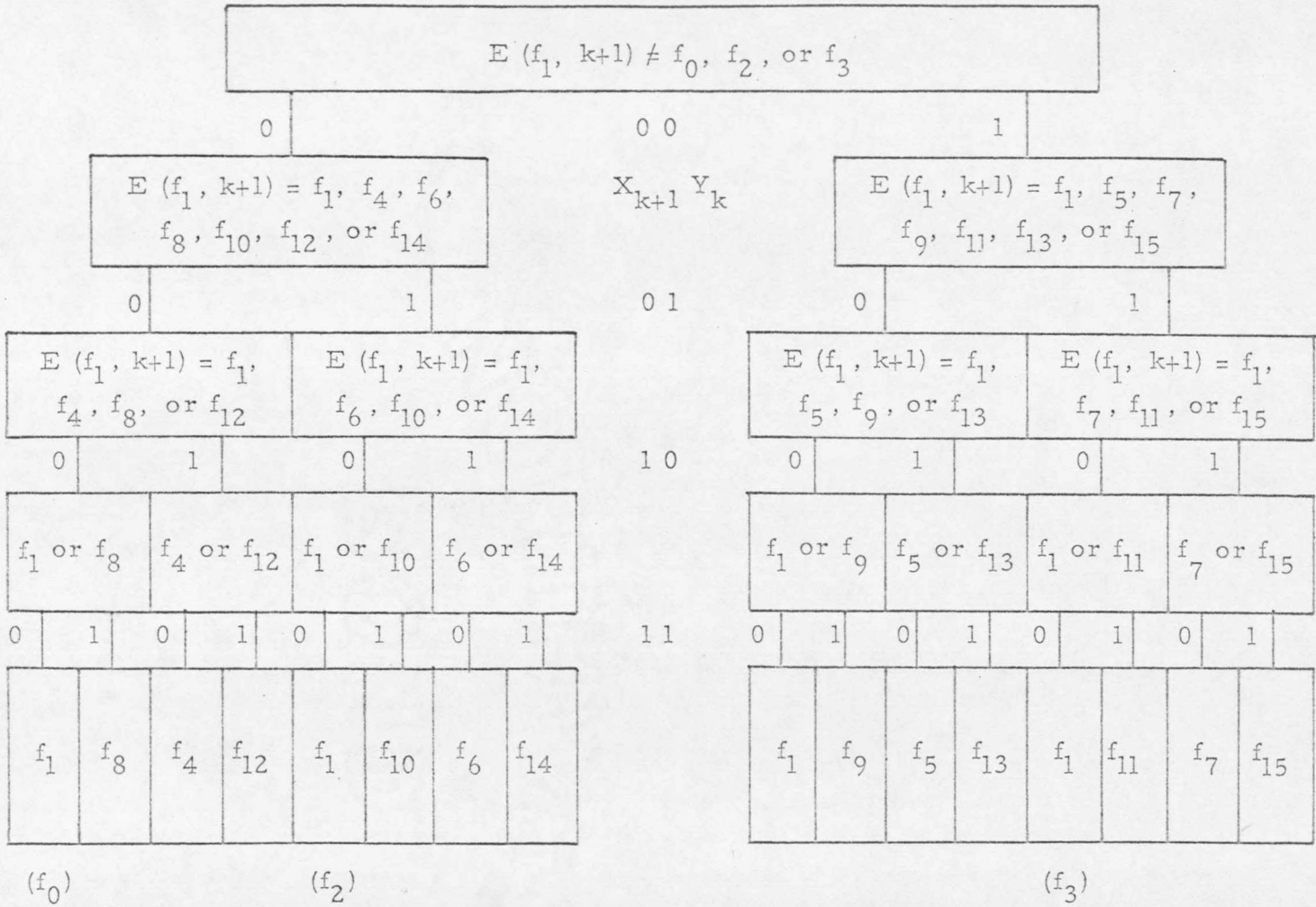


Figure 3.11. Test Decision Map for f_1 .

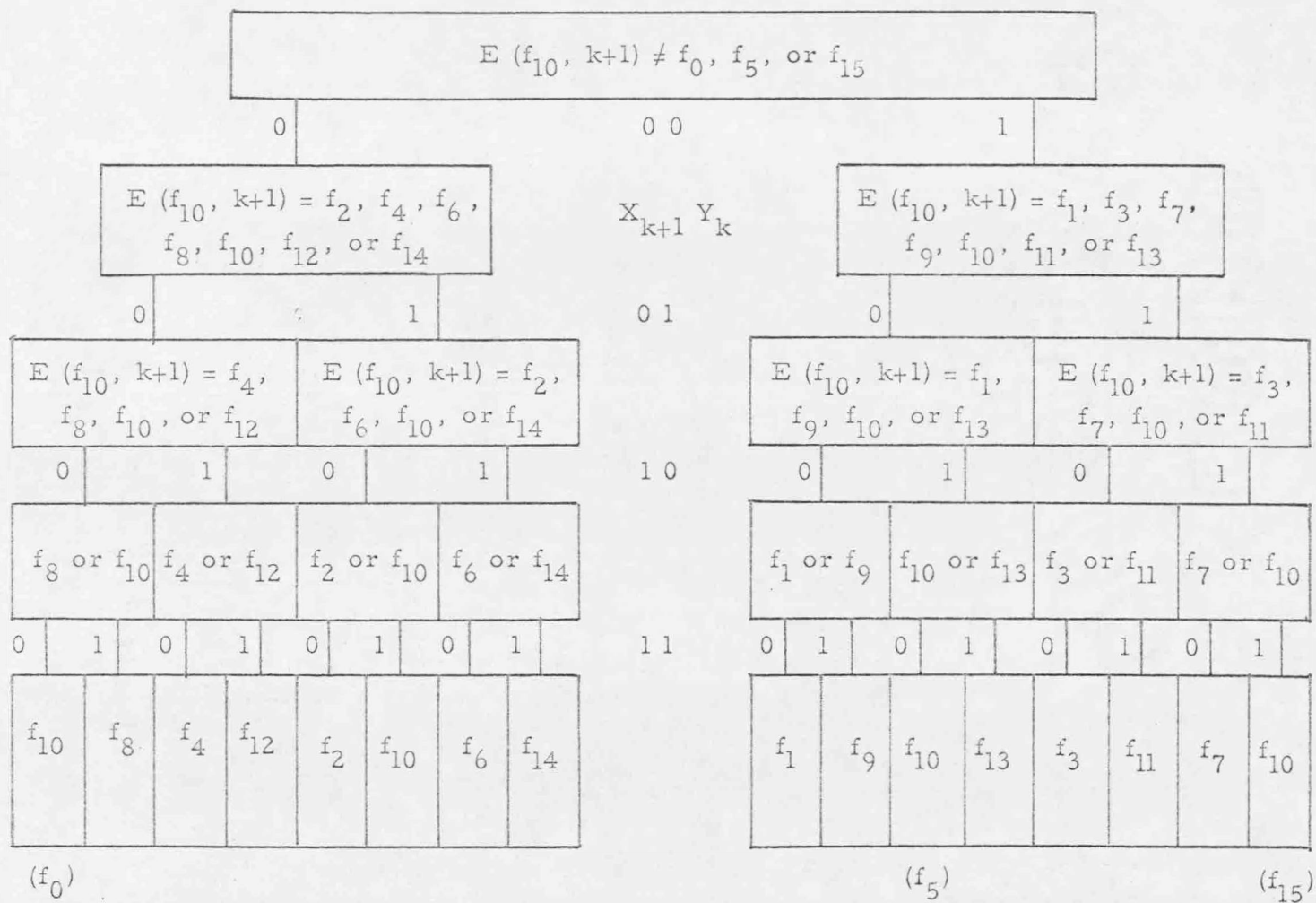
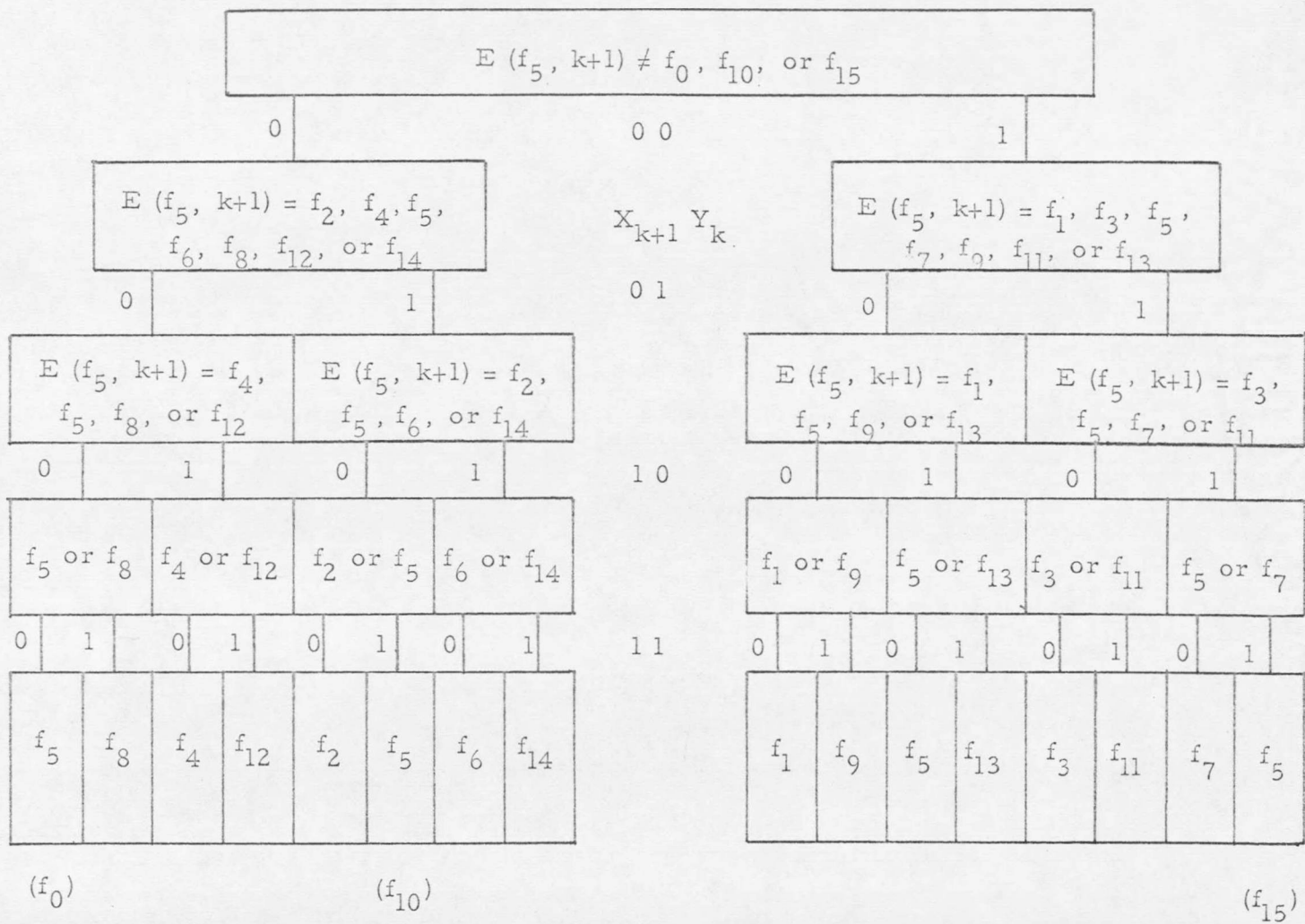


Figure 3.12. Test Decision Map for f_{10} .



X_{k+1} Y_k

0 0
0 1
1 0
1 1

Figure 3.13. Test Decision Map for f₅.

The proof of the other half of the theorem will be by contradiction. Assume the error can be located, but that the restrictions (1) - (12) are not needed. Then it can be verified that the following pairs of conditions give the same output at the terminals. Since the two conditions give the same outputs, the error cannot be located, which is a contradiction of the assumption; therefore, the assumption that the restrictions are not needed is incorrect and the proof is completed. After (1) an abbreviated notation is used.

(1) $Y_k = 1, 1, 1, 1$ and $E(f_{14}, k+1) = f_{14}$ is equivalent to $Y_k = 0, 1, 0, 1$ and $E(f_{14}, k+1) = f_{15}$ at the cascade's output terminal.

$Y_k = 0, 0, 0, 0$ and $E(f_{14}, k+1) = f_{14}$ is equivalent to $Y_k = 0, 1, 0, 1$ and $E(f_{14}, k+1) = f_{12}$ at the cascade's output terminal.

$Y_k = 1, 0, 1, 0$ and $E(f_{14}, k+1) = f_{14}$ is equivalent to $Y_k = 0, 1, 0, 1$ and $E(f_{14}, k+1) = f_{13}$ at the cascade's output terminal.

(2) $Y_k = 0, 0, 0, 0$ and $E(f_{11}, k+1) = f_{11}$; $Y_k = 0, 1,$

$$0, 1 \text{ and } E(f_{14}, k+1) = f_3.$$

$$Y_k = 1, 1, 1, 1 \text{ and } E(f_{11}, k+1) = f_{11}; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_{14}, k+1) = f_{15}.$$

$$Y_k = 1, 0, 1, 0 \text{ and } E(f_{11}, k+1) = f_{11}; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_{14}, k+1) = f_7.$$

$$(3) \quad Y_k = 1, 1, 1, 1 \text{ and } E(f_8, k+1) = f_8; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_8, k+1) = f_{12}.$$

$$Y_k = 0, 0, 0, 0 \text{ and } E(f_8, k+1) = f_8; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_8, k+1) = f_0.$$

$$Y_k = 1, 0, 1, 0 \text{ and } E(f_8, k+1) = f_8; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_8, k+1) = f_4.$$

$$(4) \quad Y_k = 1, 1, 1, 1 \text{ and } E(f_2, k+1) = f_2; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_2, k+1) = f_3.$$

$$Y_k = 0, 0, 0, 0 \text{ and } E(f_2, k+1) = f_2; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_2, k+1) = f_0.$$

$$Y_k = 1, 0, 1, 0 \text{ and } E(f_2, k+1) = f_2; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_2, k+1) = f_1.$$

$$(5) \quad Y_k = 1, 1, 1, 1 \text{ and } E(f_6, k+1) = f_6; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_6, k+1) = f_3.$$

$$Y_k = 0, 0, 0, 0 \text{ and } E(f_6, k+1) = f_6; Y_k = 0, 1,$$

$$0, 1 \text{ and } E(f_6, k+1) = f_{12}.$$

$Y_k = 1, 0, 1, 0$ and $E(f_{6,k+1}) = f_6$; $Y_k = 0, 1, 0, 1$ and $E(f_{6,k+1}) = f_9$.

(6) $Y_k = 1, 1, 1, 1$ and $E(f_{9,k+1}) = f_9$; $Y_k = 0, 1, 0, 1$ and $E(f_{9,k+1}) = f_{12}$.

$Y_k = 0, 0, 0, 0$ and $E(f_{9,k+1}) = f_9$; $Y_k = 0, 1, 0, 1$ and $E(f_{9,k+1}) = f_3$.

$Y_k = 1, 0, 1, 0$ and $E(f_{9,k+1}) = f_9$; $Y_k = 0, 1, 0, 1$ and $E(f_{9,k+1}) = f_6$.

(7) $Y_k = 1, 1, 1, 1$ and $E(f_{13,k+1}) = f_{13}$; $Y_k = 0, 1, 0, 1$ and $E(f_{13,k+1}) = f_{12}$.

$Y_k = 0, 0, 0, 0$ and $E(f_{13,k+1}) = f_{13}$; $Y_k = 0, 1, 0, 1$ and $E(f_{13,k+1}) = f_{15}$.

$Y_k = 1, 0, 1, 0$ and $E(f_{13,k+1}) = f_{13}$; $Y_k = 0, 1, 0, 1$ and $E(f_{13,k+1}) = f_{14}$.

(8) $Y_k = 1, 1, 1, 1$ and $E(f_7,k+1) = f_7$; $Y_k = 0, 1, 0, 1$ and $E(f_7,k+1) = f_3$.

$Y_k = 0, 0, 0, 0$ and $E(f_7,k+1) = f_7$; $Y_k = 0, 1, 0, 1$ and $E(f_7,k+1) = f_{15}$.

$Y_k = 1, 0, 1, 0$ and $E(f_7,k+1) = f_7$; $Y_k = 0, 1, 0, 1$ and $E(f_7,k+1) = f_{11}$.

(9) $Y_k = 1, 1, 1, 1$ and $E(f_4,k+1) = f_4$; $Y_k = 0, 1, 0, 1$ and $E(f_4,k+1) = f_0$.

$Y_k = 0, 0, 0, 0$ and $E(f_{4,k+1}) = f_4$; $Y_k = 0, 1, 0, 1$ and $E(f_{4,k+1}) = f_{12}$.

$Y_k = 1, 0, 1, 0$ and $E(f_{4,k+1}) = f_4$; $Y_k = 0, 1, 0, 1$ and $E(f_{4,k+1}) = f_8$.

(10) $Y_k = 1, 1, 1, 1$ and $E(f_{1,k+1}) = f_1$; $Y_k = 0, 1, 0, 1$ and $E(f_{1,k+1}) = f_0$.

$Y_k = 0, 0, 0, 0$ and $E(f_{1,k+1}) = f_1$; $Y_k = 0, 1, 0, 1$ and $E(f_{1,k+1}) = f_3$.

$Y_k = 1, 0, 1, 0$ and $E(f_{1,k+1}) = f_1$; $Y_k = 0, 1, 0, 1$ and $E(f_{1,k+1}) = f_2$.

(11) $Y_k = 1, 1, 1, 1$ and $E(f_{10,k+1}) = f_{10}$; $Y_k = 0, 1, 0, 1$ and $E(f_{10,k+1}) = f_{15}$.

$Y_k = 0, 0, 0, 0$ and $E(f_{10,k+1}) = f_{10}$; $Y_k = 0, 1, 0, 1$ and $E(f_{10,k+1}) = f_0$.

$Y_k = 1, 0, 1, 0$ and $E(f_{10,k+1}) = f_{10}$; $Y_k = 0, 1, 0, 1$ and $E(f_{10,k+1}) = f_5$.

(12) $Y_k = 1, 1, 1, 1$ and $E(f_{5,k+1}) = f_5$; $Y_k = 0, 1, 0, 1$ and $E(f_{5,k+1}) = f_0$.

$Y_k = 0, 0, 0, 0$ and $E(f_{5,k+1}) = f_5$; $Y_k = 0, 1, 0, 1$ and $E(f_{5,k+1}) = f_{15}$.

$Y_k = 1, 0, 1, 0$ and $E(f_{5,k+1}) = f_5$; $Y_k = 0, 1, 0, 1$ and $E(f_{5,k+1}) = f_{10}$.

Before Theorem 3.2 is stated and proved, it is interesting to note that by utilizing Figures 3.2 - 3.13 and the contradiction half of the proof it is obvious what the received values of Y_k were even though they could not be measured.

Theorem 3.2

Given a cascade with n cells, then the error can be located in 2^{n+1} tests or less if and only if it can be located in $2(n+1)$ tests or less.

Proof of Theorem 3.2

Clearly, if the error can be located in $2(n+1)$ tests or less, it can be located in 2^{n+1} tests or less.

Assume the error can be located in 2^{n+1} tests or less and consider cell n . Four tests are required to characterize cell n . If cell n is functioning properly, then cell $n-1$ is tested, and so on. For any cell i (where $i \neq n$) only two additional tests are required since two distinct settings of cell i were utilized to test cell $i + 1$ and the assumption that the error can be located allows Theorem 3.1 to be utilized so that the values of Y_i are known for the two distinct tests. Therefore, $4 + (n-1) 2 = 2(n+1)$ tests (or less) are needed to locate the error.

3.7 Test Algorithm

The test algorithm in Figure 3.14 is a result of Theorems 3.1 and 3.2 and is applicable to cascades satisfying their assumptions. It should be noted that even if conditions (1) - (12) of Theorem 3.1 can not be met, fault detection of any single error is possible and the test algorithm of Figure 3.14 is still valid. Also, Theorem 3.2 is valid for detection of faults utilizing the theory presented in Section 3.4.

3.8 Conclusion

A new approach to fault testing in digital systems is presented. A specialized digital system (cascades and certain arrays) is considered in order to obtain a condition that will enable the location of a single fault in the system (each cascade). Since elements of the system are all two-input, one-output cells, a physical basis exists to choose a maximum set of possible errors. All testing is accomplished utilizing only the input and output terminals of the system under consideration because no test pads have been placed between cells.

Consideration of the interconnection of the systems to be tested yields a highly efficient test algorithm which under certain assumptions yields location of any one of twelve possible cell errors; however, one particularly notes from the proof of Theorem 3.1 that if the assumptions restricting the possible errors from fifteen per cell to twelve per cell are relaxed, fault detection can still be accomplished. However, in general the fault could only be isolated. According to the algorithm

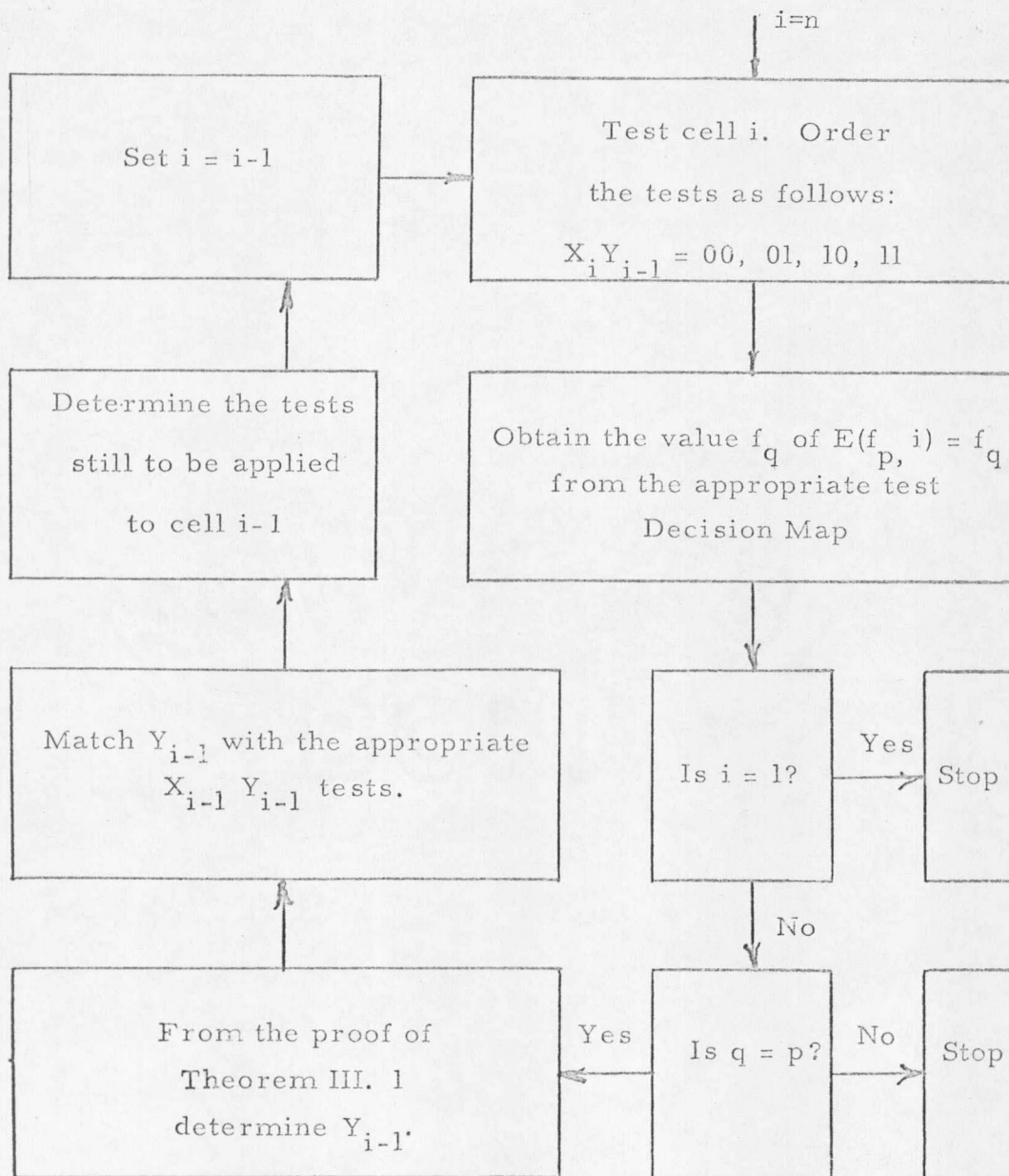


Figure 3.14.
Flow Chart for the Test Algorithm.

the cells are to be tested in the following order: $n, n-1, \dots, 2, 1$. This order is chosen on the basis that if cells $n, n-1, \dots, i+1$ are known to be fault-free, then the output of cell i can be measured at the output terminal whereas the only uncertainty in testing cell i is in the actual value of Y_{i-1} . Uncertainties in Y_{i-1} are minimized utilizing the fact that the error is time independent, thereby allowing the fault to be located. If certain minor assumptions of Theorem 3.1 cannot be met, then all errors are detectable; however, the error may be only isolated to within a group of cells and possible cell errors.

Careful testing procedures and the interconnection structure under study has considerable effect on the number of tests needed to locate a large number of possible single faults. For a n -cell cascade and n large $2(n+1) \ll 2^{n+1}$, and therefore computerized testing of cascades can be done quickly and efficiently.

Finally, the simplicity of the actual test algorithm makes this method of testing especially suitable for computerized testing, since a master program could be written so that the cell functions and their location can be put in as data and the cascades tested without altering the master program.

If the remaining four functions are desired in a cascade, a good understanding of the method discussed in Chapter 3 will allow one to extend the results of Theorem 3.1; however, when these functions are allowed one must be careful not to allow trivial cascades to be tested.

If one allows certain trivial cascades to be tested many unusual problems may occur.

Since one error can be located in a cascade under assumptions of Theorem 3.1, one error can be located in each vertical cascade and collector row of an array. The theories presented are applicable to use in any array that can be decomposed into a series of cascade structures; therefore, even cobweb arrays can be considered, because to obtain a cascade doesn't imply that physically each cell lies directly below the previous cell, but that a structure such as shown in Figure 1.1 can be obtained.

Chapter 4

TRIBUTARY CASCADES

4.1 Introduction

An important set of cascades can be built utilizing the following three functions: AND (f_8), OR (f_{14}), and EXOR (f_6). Also, if the set of possible errors is restricted, some interesting properties and results can be derived.

In this chapter, tributary cascades will be considered from the viewpoint of obtaining interesting results that may be extended to more general cases. However, the results will tend to be most interesting in the context that they are derived.

The allowable set of errors for the function f_p will be f_{15-p} , f_0 , f_{15} , f_{12} , f_3 , f_{10} , and f_5 . The number of tests needed to test these cascades will not be calculated since Theorem 3.2 gives a very good bound on the number of tests. The proofs of theorems in this section will be argumentative since the detailed proof of most of these theorems can be obtained from Theorem 3.1 under suitable modifications.

4.2 Classification of Tributary Cascades

The purpose of this section is to utilize two tests to classify tributary cascades. Theorem 4.1 and its corollary are the same whether a tributary cascade or a cascade (see Chapter 3) are under consideration. In order that the special case of cell 1 need not be considered the theorem is stated in terms of X_0 , but it could be easily modified to utilize X_1 .

Theorem 4.1.

Let the cascade have n cells. If $C_1, C_2, C_3, \dots, C_n$ are such that $f(X_0, C_1, C_2, \dots, C_n) = X_0^*$

then:

(1) $f(1, C_1, \dots, C_n) = 1^*$ and $f(0, C_1, \dots, C_n) = 0^*$

imply there is no error in the cascade or there exists one cell i such that $E(f_p, i) = f_{10}$.

(2) $f(1, C_1, \dots, C_n) = (1^*)$ and $f(0, C_1, \dots, C_n) = (0^*)$

imply there exists a cell i such that $E(f_p, i) = f_5$ or f_{15-p} .

(3) $f(1, C_1, \dots, C_n) = f(0, C_1, \dots, C_n)$ implies there

exists a cell j such that $E(f_p, j) = f_0, f_{15}, f_{12},$ or f_3 .

Theorem 4.1 will not be proven; however, the Corollary to Theorem 4.1 will be proven. The corollary has an error set that is restricted to f_0, f_{15}, f_{15-p} where f_p is the cell function.

Corollary to Theorem 4.1

Let the cascade have n cells. If C_1, \dots, C_n are such that $f(X_0, C_1, \dots, C_n) = X_0^*$ then:

(1) $f(1, C_1, \dots, C_n) = f(0, C_1, \dots, C_n)$ implies there

exists a cell i such that $E(f_p, i) = f_0$ or f_{15} .

(2) $f(1, C_1, \dots, C_n) = (1^*)$ and $f(0, C_1, \dots, C_n) = (0^*)$
imply there exists a cell i such that $E(f_p, i) = f_{15-p}$.

(3) $f(1, C_1, \dots, C_n) = 1^*$ and $f(0, C_1, \dots, C_n) = 0^*$
imply there is no error in the cascade.

Proof of the Corollary to Theorem 4.1

In part (1) f does not depend on X_0 ; therefore, there is a cell i such that $E(f_p, i) = f_0$ or f_{15} . In part (2) f depends on (X_0^*) ; therefore, there is a cell i such that $E(f_p, i) = f_{15-p}$; whereas, the proof of part (3) should now be obvious.

4.3 Subcascades

Definition 4.1 A subcascade of a cascade is any subset of adjacent cells.

Theorem 4.2 A cascade can be tested and the error located if and only if any subcascade can be tested and the error located.

Proof of Theorem 4.2

If any subcascade can be tested and the error located, then the cascade can be tested and the error

located since the cascade is a subcascade of itself.

Assume the cascade can be tested and the fault located. This assumption implies that every cell in the cascade can be tested, which implies any subset of cells of the cascade can be tested. Therefore, any subcascade can be tested and since the fault can be located, if it is in the subcascade it can be located. However, it should be noted that if the subcascade doesn't start with cell n , more than just the subcascade under consideration may have to be tested.

Assume that a cascade is given in which the subcascade consisting of cells 1, 2, and 3 is to be tested, but that cell 4 is producing $E(f_j, 4) = f_0$. In order to test the subcascade, cell 4 is tested and it is determined that cell 4 is producing $E(f_j, 4) = f_0$, is this a contradiction of Corollary 2? No, because there is only one error in the cascade and it is not in the subcascade; therefore, it may be concluded that the subcascade is error-free without ever actually testing the subcascade.

Theorem 4.2 is true for any cascade, even multi-rail cascades, since no actual mention was made of the cell functions or structure other than that it was a cascade interconnection structure.

4.4 Location and Isolation of Faults in Tributary Cascades

Three theorems will be presented in this section pertaining to tributary cascades. Theorem 4.3 is concerned with location of the seven possible errors in tributary cascades; whereas, Theorem 4.4 is concerned with the location of three possible errors in tributary cascades. Theorem 4.4 leads to Theorem 4.5 which is concerned with isolation of the errors allowed in Theorem 4.4 when the hypothesis of Theorem 4.4 are relaxed. Theorems 4.3 and 4.4 can be proven utilizing Theorem 3.1 since they are special cases of Theorem 3.1 or they can be proven directly. A discussion of Theorem 4.3 is provided which will allow one to construct the proof directly. Theorems 4.3 and 4.4 can be easily extended to encompass cascades more complex than tributary cascades; however, although Theorem 4.5 can be extended to encompass cascades, if more possible errors are allowed in Theorem 4.5, the theorem becomes too complex to warrant further attention.

Theorem 4.3

Given a tributary cascade with n cells, then the error can be located if and only if for every $i \in$

$$I_n = \{1\}$$

$$(1) \quad E(f_{14}, i) \neq f_{15}, f_{12}$$

$$(2) \quad E(f_8, i) \neq f_0, f_{12}$$

$$(3) \quad E(f_0, i) \neq f_9, f_{12}, f_3$$

Discussion of Theorem 4.3

Consider the EXOR (OR; AND) cell. If a sequence of all 0's is received, then this would appear the same as if the function f_{12} ($f_{12}; f_0$) was produced when the correct test sequence was received. If a sequence of all 1's is received, then this would appear the same as if the function f_3 ($f_{15}; f_{12}$) was produced when the correct test sequence was received. If the actually received test sequence was the compliment of the desired test sequence, then this would appear the same as if the function f_9 , ($f_{13}; f_4$) was produced when the correct test sequence was received; however, by assumption f_{13} and f_4 were not possible errors. Therefore, it is clear how the conditions for this theorem can be verified.

For the purpose of Theorem 4.4 it will be assumed that there are only three possible errors for a cell function f_p ; i.e., f_{15-p} , f_{15} , and f_0 . It should be noted that this is the standard allowable set of other investigators {1,4,5,7-9,13-15,17} augmented by f_{15-p} .

Theorem 4.4

Given a tributary cascade with n cells, then the error can be located if and only if for every $i \in$

$$I_n = \{1\}$$

$$(1) \quad E(f_{14}, i) \neq f_{15}$$

$$(2) \quad E(f_8, i) \neq f_0$$

$$(3) \quad E(f_6, i) \neq f_9$$

In order to consider isolation of errors when the hypotheses of Theorem 4.4 are relaxed, the following definitions are helpful.

Definition 4.2 An error is said to be isolated within k cells if and only if there exists a set of exactly $m > 1$ error functions with the property that $E(f_p, j) = f_e$ (where $p \neq e$) capable of producing the results obtained from the test schedule and k is the number of the distinct values in I_n that j is assigned in this set of error functions.

Since it is impossible to distinguish between certain error functions from the output terminal if the hypotheses of Theorem 4.4 are relaxed, the following definition explains the concept of a cell appearing to have a certain error.

Definition 4.3 Cell i appears to have the following error function
 $(E(f_{14}, i) = f_{15} \quad (E(f_8, i) = f_0, E(f_6, i) = f_9))$ if
and only if to the observer at the output terminal of the cascade, the test schedule indicates that $E(f_{14}, i)$

$= f_{15}$ ($E(f_8, i) = f_0$, $E(f_6, i) = f_9$) is a possible error function to be included in the set of error functions producing the results of the test schedule. (It should be noted, however, that the test schedule used may have to be supplemented with more tests in some cases in order to establish that cell i does appear to behave as if $E(f_{14}, i) = f_{15}$. ($E(f_8, i) = f_0$, $E(f_6, i) = f_9$) is a possible error function. Examples of this are given in the proof of Theorem 4.5.)

Theorem 4.5

Given a tributary cascade of n cells, then cell i appears to have the error function $E(f_{14}, i) = f_{15}$ ($E(f_8, i) = f_0$, $E(f_6, i) = f_9$) if and only if the error can be isolated to within the set of $m + 1$ cells in $\{i, i-1, \dots, i-m\}$ where cells in $\{i, i-1, \dots, i-m+1\}$ all have the cell function f_{14} (f_8, f_6) and cell $i-m$ is the first cell such that it has the cell function f_8 or f_6 (f_{14} or f_6 , f_{14} or f_8) and $m > 0$.

Proof of Theorem 4.5

The proof consists of considering three cases.

Case I. Assume cells in $\{i, i-1, \dots, i-m+1\}$ have the cell function f_8 , cell $i-m$ has the cell function f_{14} (f_6) and it appears that $E(f_8, i) = f_0$.

If $X_{i-m} = 0$ was used in the test, then set $X_{i-m} = 1$ and run the test again. If $E(f_8, i) = f_0$ still appears as a possible error, then the error lies in one of the cells in $\{i, i-1, \dots, i-m\}$ because if there exists a cell j such that $j < i-m$ and $E(f_p, j) = f_0$ or f_{15-p} is causing a 0 to be received by cell $i-m$, then cell $i-m$ is producing a 1 because its cell function is f_{14} (f_6) and in this case $E(f_8, i) \neq f_0$ is the conclusion. Therefore, since cells in $\{n, \dots, i+1\}$ have already been tested, the error is isolated to the $m+1$ cells in $\{i, \dots, i-m\}$.

Case II

If cell i is a cell with cell function f_{14} , the conclusion can be reached in a dual manner in the case where cell $i-m$ has the cell function f_8 ; however, the case in which cell $i-m$ has the cell function f_6 is not obvious. In the test schedule for cell $i-m$ having a cell function f_6 , the cells must be set as follows: $X_i = X_{i-1} = \dots X_{i-m+1} = X_{i-m} = 0$ and the constant produced by cell $i-m-1$ is set such that it is 0. If it appears that $E(f_{14}, i) = f_{15}$ (i.e., when X_i was set to 1 without anything else being changed, the output from cell i remained at 1) then set $X_{i-m} = 1$ in the original test (instead of 0) and test again. If $f = 1$ for both

of these tests, then the error lies in the cells $\{i, i-1, \dots, i-m\}$ since if there exists a cell j such that $j < i-m$ and $E(f_p, j) = f_{15}$ or f_{15-p} is causing cell $i-m$ to receive a 1, then setting $X_{i-m} = 1$ causes cell $i-m$ to produce a 0 output, since in the previous case it had to be producing a 1 output, otherwise cell i would not have appeared as if $E(f_{14}, i) = f_{15}$ and allowed the conclusion that $E(f_{14}, i) = f_{14}$.

Case III The case for cell i having the cell function f_6 and cell $i-m$ having the cell function f_8 is trivial; however, the case for cell $i-m$ having the cell function f_{14} is not obvious but can be obtained from the proof of Case II with suitable modifications.

To prove the other half of the theorem, assume the error is isolated within the cells in $\{i, \dots, i-m\}$, then the error could not be located. Therefore, one of the cells $p \in \{i, \dots, i-m\}$ with cell function f_d must have the error function $E(f_d, p) = f_{15}$ ($E(f_d, p) = f_0, E(f_d, p) = f_9$) and when any one of those cells has this error function it appears as if $E(f_{14}, i) = f_{15}$ ($E(f_8, i) = f_0, E(f_6, i) = f_9$).

4.5 Conclusion

Several concepts were presented in this chapter which allow certain facts to be ascertained about faults in cellular cascades. A reduced set of possible errors is assumed; although the theorems are concerned with tributary cascades, they can be extended to encompass cascades with more possible errors.

It is interesting to note the results of cascade classification and error isolation. Depending on the types of cascades and error sets under consideration in a specific problem it may be useful to utilize these sections in the testing procedure; however, these results are too complex in the general cases to be useful.

Chapter 5

EXAMPLES

5.1 Introduction and Description of Examples

The purpose of this chapter is to present thirty-one examples of the use of the previously presented theorems.

For all examples f_T denotes the theoretical value of the function of the cascade under consideration; whereas, f_A denotes the actually measured value. The specification of the error that is in the cascade has been placed immediately following the example number. The specification of the error is given in the notation of this paper.

Examples 5.1 - 5.14 are direct applications of Theorems 3.1 and 3.2. Examples 5.1 - 5.14 are worked on the basis of the hypothesis of Theorem 3.1 concerning the restriction of errors. The details of the application of the test algorithm are not given, the conclusions of the tests are given in the notation previously developed, and the tests are grouped as they would be in actual application of the test algorithm; i.e., four tests are applied to cell n and a conclusion is then made, the two remaining tests are applied to cell $n-1$ and a conclusion is then made, etc.

Examples 5.15 and 5.16 were included to illustrate that there do exist test schedules based on theories of single-fault location that can locate multiple faults in a system. The multiple faults were located by assuming that there was only one fault and when this fault was located the tests were continued as if looking for another single fault, etc. The notation for Examples 5.15 and 5.16 is the same as

the notation for Examples 5.1 - 5.14. Examples 5.15 and 5.16 utilize the hypothesis of Theorem 3.1 except that they have multiple faults. The tests schedules for Examples 5.15 and 5.16 are taken from Theorem 3.1 just as if only one fault was being searched for.

Examples 5.17 - 5.20 are examples of the use of Theorem 4.4. Only three errors are allowed in Theorem 4.4 and the cascade under consideration satisfies the hypothesis of Theorem 4.4 on the restriction of errors. In utilizing Theorem 4.4 it has been found that the following tests will usually be sufficient: OR cell (00, 10), AND cell (11, 01), and EXOR cell (00, 10), where CD means $X_i = C$ and $Y_{i-1} = D$. Usually these are the tests one wants to utilize if possible when using Theorem 4.4. It is also noted that with suitable care the test schedule length should be $n+1$ (or less) for this situation.

Examples 5.21 - 5.24 are examples of the use of Theorem 4.4 on a cascade other than a tributary cascade. Suitable modifications in the assumptions of Theorem 4.4 have been made and with this example and Theorem 4.4 the reader should be able to construct the general case of Theorem 4.4.

Examples 5.25 - 5.27 are examples of the general case of Theorem 4.5. With Theorem 4.5 and Examples 5.25 - 5.27 the interested reader should be able to construct the general case of Theorem 4.5.

Examples 5.28 - 5.31 are examples of the use of Theorem 4.1.

Some of the theorems given may easily be extended. For example, if $X_i + Y_{i-1}$ is a cell function in a cascade, then one can construct the proper result utilizing $X_i + Y_{i-1}$ and then compliment the values of X_i . Examples of this procedure can be seen by comparing Examples 5.5 and 5.9 or Examples 5.21 and 5.25. This procedure makes it useful for some considerations to list the cascade functions as follows:

$X^* + Y$, $X^* + Y'$, $X^* Y$, $X^* Y'$, $X^* \oplus Y$, and Y^* .

It should be noted that Theorem 3.1 allows serial, hybrid, or tabular testing. If one wants, the testing may be accomplished in the serial mode (Examples 5.17 - 5.27), in the hybrid mode (Examples 5.1 - 5.16, or in a tabular mode (no examples).¹

¹ The figures for this chapter appear at the end of the chapter beginning with page 107.

5.2 Examples

Example 5.1: Assume there is no fault in the cascade shown in Figure 5.1.

Test	X_0	X_1	X_2	X_3	f_T	f_A	Conclusion
	0	0	0	1	0	0	
	0	0	0	0	0	0	
	0	0	1	0	0	0	
	0	0	1	1	1	1	
							$E(f_8, 3) = f_8$
	1	1	1	1	0	0	
	1	1	0	1	1	1	
							$E(f_6, 2) = f_6$
	0	1	0	1	1	1	
	1	0	0	1	1	1	
							$E(f_{14}, 1) = f_{14}$

Example 5.2: Let $E(f_6, 2) = f_0$ in the cascade in Figure 5.1.

Test

X_0	X_1	X_2	X_3	f_T	f_A	Conclusion
0	0	0	0	0	0	
0	0	0	1	0	0	
0	0	1	1	1	0	
0	0	1	0	0	0	
						$E(f_8, 3) = f_8$
0	1	0	1	1	0	
0	1	1	1	0	0	
						$E(f_6, 2) = f_0$

Example 5.3: Let $E(f_8, 3) = f_2$ in the cascade shown in Figure 5.1.

Test

X_0	X_1	X_2	X_3	f_T	f_A	Conclusion
0	0	0	1	0	0	
0	0	0	0	0	0	
0	0	1	0	0	1	
0	0	1	1	1	0	

$$E(f_8, 3) = f_2$$

Example 5.4: Let $E(f_{14}, 1) = f_{15}$ in the cascade shown in Figure 5.1.

Test

X_0	X_1	X_2	X_3	f_T	f_A	Conclusion
-------	-------	-------	-------	-------	-------	------------

0	0	0	1	0	1	
---	---	---	---	---	---	--

0	0	0	0	0	0	
---	---	---	---	---	---	--

0	0	1	0	0	0	
---	---	---	---	---	---	--

0	0	1	1	1	0	
---	---	---	---	---	---	--

$E(f_8, 3) = f_8$

1	1	1	1	0	0	
---	---	---	---	---	---	--

1	1	0	1	1	1	
---	---	---	---	---	---	--

$E(f_6, 2) = f_6$

0	1	0	1	1	1	
---	---	---	---	---	---	--

1	0	0	1	1	1	
---	---	---	---	---	---	--

$E(f_{14}, 1) = f_{15}$

Example 5.5: Assume there is no fault in the cascade shown in Figure 5.2.

Test								Conclusion
X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	
0	0	0	0	0	0	0	0	
0	0	0	0	0	1	1	1	
0	0	0	0	1	1	0	0	
0	0	0	0	1	0	1	1	$E(f_6, 5) = f_6$
0	1	0	1	0	0	1	1	
0	1	0	1	1	0	1	1	$E(f_{14}, 4) = f_{14}$
0	0	0	1	0	0	0	0	
0	1	0	0	0	0	0	0	
								$E(f_8, 3) = f_8$
0	0	1	1	0	0	0	0	
0	1	1	1	0	0	1	1	
								$E(f_{10}, 2) = f_{10}$
1	0	1	1	0	0	1	1	
1	1	1	1	0	0	0	0	$E(f_6, 1) = f_6$

Example 5.6: Let $E(f_{14}, 4) = f_1$ in the cascade shown in Figure 5.2.

Test

X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusion
0	0	0	0	0	0	0	1	
0	0	0	0	0	1	1	0	
0	0	0	0	1	1	0	1	
0	0	0	0	1	0	1	0	
0	1	0	1	0	0	1	0	
0	1	0	1	1	0	1	0	

$$E(f_6, 5) = f_6$$

$$E(f_{14}, 4) = f_1$$

Example 5.7: Let $E(f_6, 1) = f_{14}$ in the cascade shown in Figure 5.2.

Test

X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusion
0	0	0	0	0	0	0	0	
0	0	0	0	0	1	1	1	
0	0	0	0	1	1	0	0	
0	0	0	0	1	0	1	1	
								$E(f_6, 5) = f_6$
0	1	0	1	0	0	1	1	
0	1	0	1	1	0	1	1	
								$E(f_{14}, 4) = f_{14}$
0	0	0	1	0	0	0	0	
0	1	0	0	0	0	0	0	
								$E(f_8, 3) = f_8$
0	0	1	1	0	0	0	0	
0	1	1	1	0	0	1	1	
								$E(f_{10}, 2) = f_{10}$
1	0	1	1	0	0	1	1	
1	1	1	1	0	0	0	1	
								$E(f_6, 1) = f_{14}$

Example 5.8: Let $E(f_8, 3) = f_{15}$ in the cascade shown in Figure 5.2.

Test

X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusion
0	0	0	0	0	0	0	1	
0	0	0	0	0	1	1	0	
0	0	0	0	1	1	0	0	
0	0	0	0	1	0	1	1	
								$E(f_6, 5) = f_6$
0	1	0	1	0	0	1	1	
0	1	0	1	1	0	1	1	
								$E(f_{14}, 4) = f_{14}$
0	0	0	1	0	0	0	1	
0	1	0	0	0	0	0	1	
								$E(f_8, 3) = f_{15}$

Example 5.9: Assume there is no fault in the cascade shown in Figure 5.3.

Test

X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusion
0	1	0	1	0	1	0	0	
0	1	0	1	0	0	1	1	
0	1	0	1	1	0	0	0	
0	1	0	1	1	1	1	1	$E(f_9, 5) = f_9$
0	0	0	0	0	1	1	1	
0	0	0	0	1	1	1	1	$E(f_{14}, 4) = f_{14}$
0	1	0	0	0	1	0	0	
0	0	0	1	0	1	0	0	$E(f_2, 3) = f_2$
0	1	1	0	0	1	0	0	
0	0	1	0	0	1	1	1	$E(f_{10}, 2) = f_{10}$
1	1	1	0	0	1	1	1	
1	0	1	0	0	1	0	0	$E(f_9, 1) = f_9$

Example 5.10: Let $E(f_{14}, 4) = f_6$ in the cascade shown in Figure 5.3.

Test								Conclusion
X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	
0	1	0	1	0	1	0	0	
0	1	0	1	0	0	1	1	
0	1	0	1	1	0	0	0	
0	1	0	1	1	1	1	1	
								$E(f_9, 5) = f_9$
0	0	0	0	0	1	1	1	
0	0	0	0	1	1	1	0	
								$E(f_{14}, 4) = f_6$

Example 5.11: Let $E(f_9, 5) = f_0$ in the cascade shown in Figure 5.3.

Test

X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusion
0	1	0	1	0	1	0	0	
0	1	0	1	0	0	1	0	
0	1	0	1	1	0	0	0	
0	1	0	1	1	1	1	0	

$$E(f_9, 5) = f_0$$

Example 5.12: Let $E(f_2, 3) = f_6$ in the cascade shown in Figure 5.3.

Test

X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusion
0	1	0	1	0	1	0	1	
0	1	0	1	0	0	1	0	
0	1	0	1	1	0	0	0	
0	1	0	1	1	1	1	1	
								$E(f_9, 5) = f_9$
0	0	0	0	0	1	1	1	
0	0	0	0	1	1	1	1	
								$E(f_{14}, 4) = f_{14}$
0	1	0	0	0	1	0	0	
0	0	0	1	0	1	0	0	
								$E(f_2, 3) = f_6$

Example 5.13: Assume there is no fault in the cascade shown in Figure 5.4.

Test

X_0	X_1	X_2	X_3	f_T	f_A	Conclusion
-------	-------	-------	-------	-------	-------	------------

0	0	1	0	0	0	
---	---	---	---	---	---	--

0	0	1	1	1	1	
---	---	---	---	---	---	--

0	0	0	1	1	1	
---	---	---	---	---	---	--

0	0	0	0	1	1	
---	---	---	---	---	---	--

$$E(f_{14}, 3) = f_{14}$$

1	0	0	0	0	0	
---	---	---	---	---	---	--

1	0	1	0	1	1	
---	---	---	---	---	---	--

$$E(f_9, 2) = f_9$$

1	1	1	0	0	0	
---	---	---	---	---	---	--

0	1	1	0	0	0	
---	---	---	---	---	---	--

$$E(f_2, 1) = f_2$$

Example 5.14: Let $E(f_2, 1) = f_0$ in the cascade shown in Figure 5.4.

Test

X_0	X_1	X_2	X_3	f_T	f_A	Conclusion
-------	-------	-------	-------	-------	-------	------------

0	0	1	0	0	0	
---	---	---	---	---	---	--

0	0	1	1	1	1	
---	---	---	---	---	---	--

0	0	0	1	1	1	
---	---	---	---	---	---	--

0	0	0	0	1	1	
---	---	---	---	---	---	--

$E(f_{14}, 3) = f_{14}$

1	0	0	0	0	1	
---	---	---	---	---	---	--

1	0	1	0	1	0	
---	---	---	---	---	---	--

$E(f_9, 2) = f_9$

1	1	1	0	0	0	
---	---	---	---	---	---	--

0	1	1	0	0	0	
---	---	---	---	---	---	--

$E(f_2, 1) = f_0$

Example 5.15: Let $E(f_{14}, 3) = f_6$ and $E(f_2, 1) = f_{10}$ in the cascade shown in Figure 5.4.

Test

X_0	X_1	X_2	X_3	f_T	f_A	Conclusion
0	0	1	0	0	0	
0	0	1	1	1	1	
0	0	0	1	1	0	
0	0	0	0	1	1	$E(f_{14}, 3) = f_6$
1	0	0	0	0	0	
1	0	1	0	1	1	$E(f_9, 2) = f_9$
1	1	1	0	0	1	
0	1	1	0	0	0	$E(f_2, 1) = f_{10}$

Example 5.16: Let $E(f_{14}, 3) = f_9$, $E(f_9, 2) = f_1$, and $E(f_2, 1) = f_6$
in the cascade shown in Figure 5.4.

Test

X_0	X_1	X_2	X_3	f_T	f_A	Conclusion
0	0	1	0	0	1	
0	0	1	1	1	0	
0	0	0	1	1	1	
0	0	0	0	1	0	
						$E(f_{14}, 3) = f_9$
1	0	0	1	0	0	
1	0	1	1	1	0	
						$E(f_9, 2) = f_1$
1	1	0	1	1	1	
0	1	0	1	1	0	
						$E(f_2, 1) = f_6$

Example 5.17: Assume there is no error in the cascade shown in Figure 5.5.

Test

X_0	X_1	X_2	X_3	X_4	f_T	f_A	Conclusions
0	0	1	0	0	0	0	$E(f_6, 4) \neq f_{15}$ $E(f_8, 3) \neq f_{15}$ or f_7
0	0	1	0	1	1	1	$E(f_6, 4) \neq f_0$
0	0	1	1	0	1	1	$E(f_{14}, 2) \neq f_1$ or f_0
0	0	0	1	0	0	0	$E(f_{14}, 1) \neq f_{15}$ or f_1
0	1	0	1	1	1	1	$E(f_{14}, 1) \neq f_0$

Example 5.18: Assume $E(f_8, 3) = f_{15}$ for the cascade shown in Figure 5.5.

Test							Conclusions
X_0	X_1	X_2	X_3	X_4	f_T	f_A	
0	0	1	0	0	0	1	$E(f_6, 4) = f_{15}$ or $E(f_8, 3) = f_7$ or f_{15}
0	0	1	0	1	1	0	$E(f_6, 4) \neq f_{15}$
0	0	1	1	0	1	1	$E(f_8, 3) \neq f_7$

Example 5.19: Assume $E(f_{14}, 1) = f_{15}$ for the cascade shown in Figure 5.5.

Test							Conclusions
X_0	X_1	X_2	X_3	X_4	f_T	f_A	
0	0	1	0	0	0	0	$E(f_6, 4) \neq f_{15}$ $E(f_8, 3) \neq f_7$ or f_{15}
0	0	1	0	1	1	1	$E(f_8, 4) \neq f_0$
0	0	1	1	0	1	1	$E(f_{14}, 2) \neq f_0$ or f_1
0	0	0	1	0	0	1	$E(f_{14}, 1) = f_{15}$ or f_1
0	1	0	1	0	1	1	$E(f_{14}, 1) \neq f_1$

Example 5.20: Assume $E(f_{14}, 2) = f_0$ for the cascade shown in Figure 5.5.

Test							Conclusions
X_0	X_1	X_2	X_3	X_4	f_T	f_A	
0	0	1	0	0	0	0	$E(f_6, 4) \neq f_{15}$ $E(f_8, 3) \neq f_7$ or f_{15}
0	0	1	0	1	1	1	$E(f_6, 4) \neq f_{15}$
0	0	1	1	0	1	0	$E(f_{14}, 2) = f_0$ or f_1
0	0	0	1	0	0	0	$E(f_{14}, 2) \neq f_1$

Example 5.21: Assume there is no error in the cascade shown in Figure 5.6.

Test								Conclusions
X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	
0	0	1	0	0	0	0	0	$E(f_6, 5) \neq f_{15}$ $E(f_4, 4) \neq f_{11}$ or f_{15}
0	0	1	0	0	1	1	1	$E(f_6, 5) \neq f_0$
0	0	1	0	1	0	1	1	$E(f_{13}, 3) \neq f_2$ $E(f_{14}, 2) \neq f_0$ or f_1
0	0	1	1	1	0	0	0	$E(f_{13}, 3) \neq f_0$
0	0	0	0	1	0	0	0	$E(f_8, 1) \neq f_1$ or f_{15}
0	1	0	0	1	0	1	1	$E(f_8, 1) \neq f_0$

Example 5.22: Assume $E(f_{14}, 2) = f_1$ for the cascade shown in Figure 5.6.

Test								Conclusions
X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	
0	0	1	0	0	0	0	0	$E(f_6, 5) \neq f_{15}$ $E(f_4, 4) \neq f_{11}$ or f_{15}
0	0	1	0	0	1	1	1	$E(f_6, 5) \neq f_0$
0	0	1	0	1	0	1	0	$E(f_{13}, 3) = f_2$ or $E(f_{14}, 2) = f_1$ or $E(f_8, 1) = f_7$ or f_{15}
0	0	1	1	1	0	0	0	$E(f_{13}, 3) \neq f_2$
0	0	0	0	1	0	0	1	$E(f_8, 1) \neq f_7$ or f_{15}

Example 5.23: Assume $E(f_6, 5) = f_0$ for the cascade shown in Figure 5.6.

Test								Conclusions
X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	
0	0	1	0	0	0	0	0	$E(f_6, 5) \neq f_{15}$ $E(f_4, 4) \neq f_{11}$ or f_{15}
0	0	1	0	0	1	1	0	$E(f_6, 5) = f_0$

Example 5.24: Assume $E(f_8, 1) = f_7$ for the cascade shown in Figure 5.6.

Test								
X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusions
0	0	1	0	0	0	0	0	$E(f_6, 5) \neq f_{15}$ $E(f_4, 4) \neq f_{11}$ f_{15}
0	0	1	0	0	1	1	1	$E(f_6, 5) \neq f_0$
0	0	1	0	1	0	1	0	$E(f_{13}, 3) = f_2$ or $E(f_{14}, 2) = f_1$ or $E(f_8, 1) = f_7$ or f_{15}
0	0	1	1	1	0	0	0	$E(f_{13}, 3) \neq f_2$
0	0	0	0	1	0	1	0	$E(f_{14}, 2) \neq f_1$
0	1	0	0	1	0	1	0	$E(f_8, 1) \neq f_{15}$

Example 5.25: Assume there is no error in the cascade shown in Figure 5.7.

Test								
X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusions
0	0	1	1	0	0	0	0	$E(f_6, 5) \neq f_{15}$ $E(f_4, 4) \neq f_{11}$ or f_{15}
0	0	1	1	0	1	1	1	$E(f_6, 5) \neq f_0$
0	0	1	1	1	0	1	1	$E(f_{13}, 3) \neq f_2$ $E(f_{14}, 2) \neq f_0$ or f_1
0	0	1	0	1	0	0	0	$E(f_{13}, 3) \neq f_0$
0	0	0	1	1	0	0	0	$E(f_{14}, 1) \neq f_1$ or f_{15}
0	1	0	1	1	0	1	1	$E(f_8, 1) \neq f_0$

Example 5.26: Assume $E(f_7, 3) = f_{15}$ for the cascade shown in Figure 5.7.

Test								
X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusions
0	0	1	1	0	0	0	0	$E(f_6, 5) \neq f_{15}$ $E(f_4, 4) \neq f_{11}$ or f_{15}
0	0	1	1	0	1	1	1	$E(f_6, 5) \neq f_0$ or f_9
0	0	1	1	1	0	1	0	$E(f_4, 4) = f_0$ or $E(f_7, 3) = f_{15}$ or $E(f_{14}, 2) = f_0$

Example 5.27: Assume $E(f_6, 5) = f_9$ for the cascade shown in Figure 5.7.

Test

X_0	X_1	X_2	X_3	X_4	X_5	f_T	f_A	Conclusions
0	0	1	1	0	0	0	1	$E(f_4, 4) = f_{15}$ or f_{11} $E(f_6, 5) = f_9$ or f_{15}
0	0	1	1	0	1	1	0	$E(f_4, 4) = f_{11}$ or $E(f_6, 5) = f_9$

For Examples 5.28 - 5.31, the cascade shown in Figure 5.8 is considered. The function for this cascade is $f_T = X_5 + X_4 (X_3 (X_2 + X_1 X_0))$ and the chosen constants are $C_1 = 1, C_2 = 0, C_3 = 0, C_4 = 1,$ and $C_5 = 0$. This yields $f_T (X_0, C_1, C_2, C_3, C_4, C_5) = X_0$.

Example 5.28: Assume there is no error in the cascade shown in Figure 5.8.

$f_A(1, C_1, C_2, C_3, C_4, C_5) = 1$ and $f_A(0, C_1, C_2, C_3, C_4, C_5) = 0$
imply that there is no error in the cascade.

Example 5.29: Assume $E(f_{13}, 3) = f_0$ for the cascade shown in Figure 5.8.

$f_A(1, C_1, C_2, C_3, C_4, C_5) = 1$ and $f_A(0, C_1, C_2, C_3, C_4, C_5) = 1$
imply there exists an $i \in I_5$ such that $E(f_p, i) = f_0$ or f_{15} .

Example 5.30: Assume $E(f_{13}, 3) = f_{15}$ for the cascade shown in Figure 5.8.

$f_A(1, C_1, C_2, C_3, C_4, C_5) = 0$ and $f_A(0, C_1, C_2, C_3, C_4, C_5) = 0$
imply there exists an $i \in I_5$ such that $E(f_p, i) = f_0$ or f_{15} .

Example 5.31: Assume $E(f_{14}, 2) = f_1$ for the cascade shown in Figure 5.8.

$f_A(1, C_1, C_2, C_3, C_4, C_5) = 0$ and $f_A(0, C_1, C_2, C_3, C_4, C_5) = 1$
imply there exists an $i \in I_5$ such that $E(f_p, i) = f_{15-p}$.

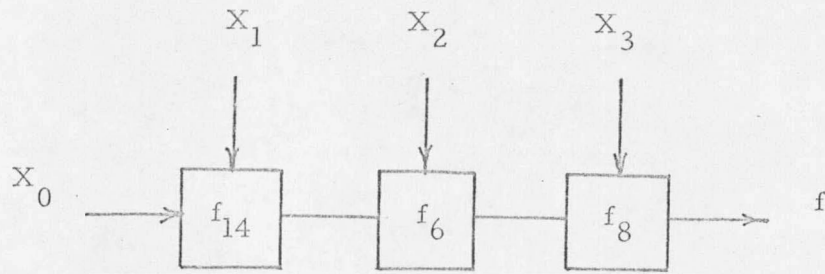


Figure 5.1. Test Example.

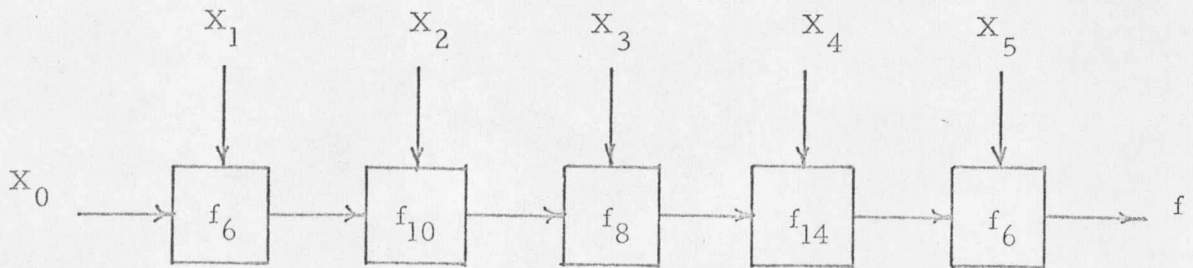


Figure 5.2. Test Example.

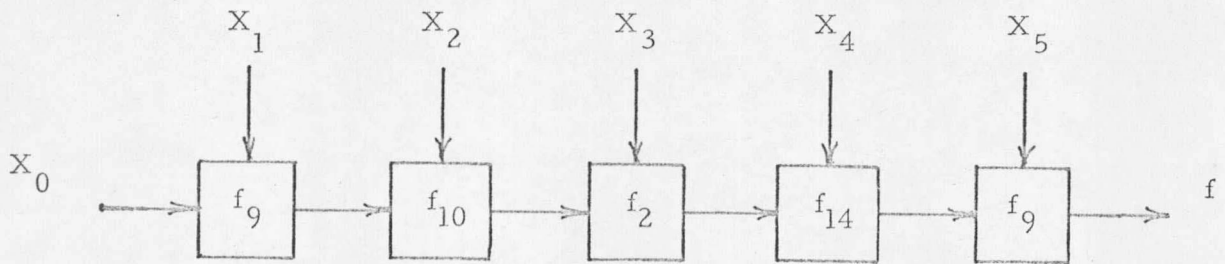


Figure 5.3. Test Example.

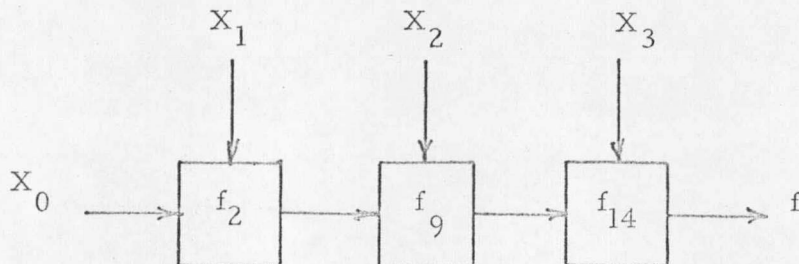


Figure 5.4. Test Example.

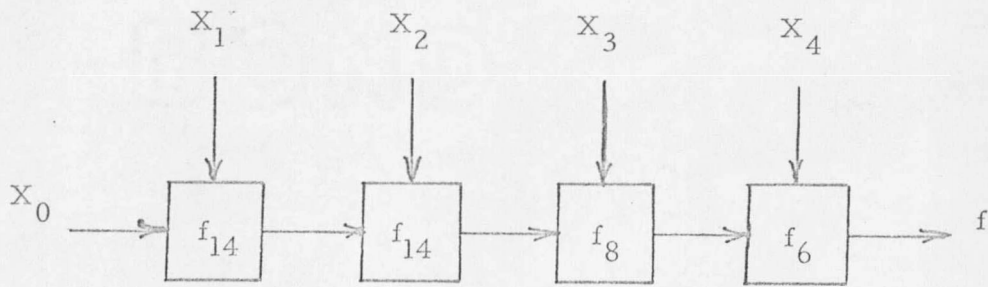


Figure 5.5. Test Example.

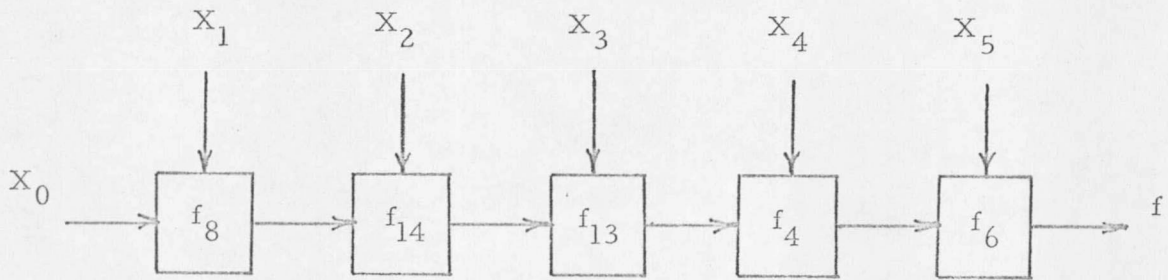


Figure 5.6. Test Example.

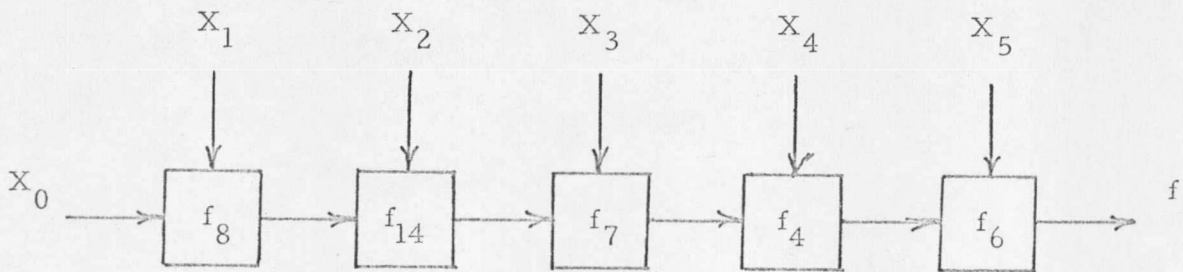


Figure 5.7. Test Example.

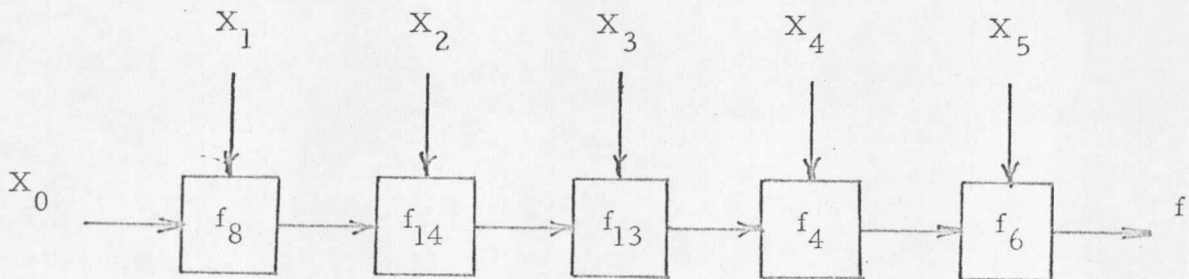


Figure 5.8. Test Example.

Chapter 6

CONCLUSION

6.1 Summary

Fault location in cellular arrays composed of two-input, one output cells is examined in this research. It is shown that arrays of this type can be decomposed so that the actual problem to be solved is fault location in a single-rail cascade. A physical basis for the assumption of a maximum allowable set of possible errors is given and it is seen that each cell could have fifteen possible errors.

An algorithm is given for testing a cellular cascade. If the algorithm is followed it is seen that three of the possible fifteen errors cannot be allowed to occur if fault location is to result.

This information is invaluable to the designer of the circuits to be used in cellular arrays. Since if he can design the circuits using redundancy techniques such that the probability that these three errors can occur is decreased, then the testing problems for cellular arrays can be simplified. Also, once the circuits are designed this way all errors will be locatable, thus providing more reliable circuits than would otherwise be possible.

If fault detection and isolation is to result, it can be easily seen that any of the fifteen errors can be detected. A necessary and sufficient condition for the location of faults (Theorem 3.1) is given. From the proof of Theorem 3.1 one can easily deduce that any of the fifteen possible errors can be detected. Theorem 3.2 gives a firm least upper bound on the number of tests needed to locate a fault in a cas-

cade. One notes that fault detection is a much simpler problem to solve than fault location especially if only two failure types (s-a-1 and s-a-0) are allowed.

Chapter 4 examines classification (detection) of cascades, location of faults, and isolation of faults on restricted sets of possible errors. The Corollary to Theorem 4.1 is especially interesting because it points out a method that will allow detection of s-a-0 and s-a-1 faults in cellular arrays in fewer tests than any method examined by other researchers. Although the corollary is not written for the set of cutpoint functions, it can be easily altered to apply to cutpoint arrays if certain trivial cascades are allowed. Location of faults in sub-cascades is considered in Theorem 4.2. Theorems 4.3 and 4.5 deal with the problem of fault location on restricted error sets. Theorem 4.6 concerns isolation of errors when the hypotheses of Theorem 4.5 are relaxed. It is shown that with the addition of a few extra tests the fault can be isolated to within a set of cells that can be specified analytically.

A test algorithm is given; however, the given algorithm is only one of several possible algorithms. If the tester desires, a complete test schedule could be derived or the testing could be done in a serial mode. Thirty-one examples of various test procedures are worked with emphasis given to examples of the utilization of Theorems 3.1 and 3.2 and the given test algorithm.

Since cells are tested individually, the given test algorithm is the most natural because it gives a conclusion as soon as a cell has been tested.

6.2 Suggestions for Further Study

Five topics worthy of further consideration are evident from this research.

1. The method of Chapter 3 should be extended to n-input one-output cascades in which $X_{i,1}$, $X_{i,2}, \dots, X_{i,n-1}$ would take the place of X_i in the single-rail cascades. This extension should be easy since all that needs to be done is to try all combinations of $X_{i,1}$, $X_{i,2}, \dots, X_{i,n-1}$ for $Y_{i-1} = 0$ and 1.
2. Extension of single fault location procedures should be considered. Examples 5.15 and 5.16 show that some multiple faults are locatable utilizing single fault location techniques.
3. The test algorithm given in Figure 3.14 should be programmed to see how fast computer testing could be.
4. Fault location in multiple-rail cascades

should be considered. In order to test a multiple-rail cell one should apply the same method as used for single-rail cascades. For a n-rail cascade there are $(2^n)^{2^n} - 1$ possible bad sequences because one would like to test all 2^n possible n-rail combinations any of these combinations could be incorrect, and there is a possible 2^n ways for the sequence to be incorrect.

However, some of these sequences are equivalent; i.e., if the desired test sequence for a two-rail cascade cell $Y_{i,1} = X_i + Y_{i-1,1} + Y_{i-1,2}$ ($Y_{i,2}$ is not being considered) is $Y_{i-1,1} Y_{i-1,2} = 00, 01, 10, 11$, then the following bad sequences give the same output $Y_{i,1}$;

10, 00, 00, 01

10, 00, 00, 10

10, 00, 00, 11

01, 00, 00, 01

01, 00, 00, 10

01, 00, 00, 11

11, 00, 00, 01

11, 00, 00, 10

11, 00, 00, 11

All of the above sequences give the $Y_{i,1}$ sequence $1, X_i, X_i, 1$ so that they are all terminally indistinguishable with respect to $Y_{i,1}$. There is an added complication; however, due to the large number of functions available in the multiple-rail cascades it is suggested that the research utilize equivalence classes.

5. In all types of cascades isolation of errors should be considered because as the complexity of cells increases the location problem may become insolvable due to notational complexities; whereas, these difficulties may not appear in fault-isolation considerations.

LITERATURE CITED

1. Armstrong, D. B., "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets," IEEE Transactions on Electronics Computers, Vol. EC-15, Feb. 1966, pp. 66-73.
2. Chang, H. Y., "A Method for Digitally Simulating Shorted Input Diode Failures," Repository of IEEE Transactions on Electronic Computers.
3. Cohn, M. and Ott, G., The Design of Adaptive Procedures for Fault Detection and Isolation, Sperry Rand Research Report SRRR-RR-66-64, Program 13-551-70, Oct. 66.
4. Friedman, A.D., "Fault Detection in Redundant Circuits," IEEE Transactions on Electronic Computers, Vol. EC-16, Feb. 1967, pp. 99-100.
5. Galey, J.M., Norby, R.E., and Roth, J.P., "Techniques for the Diagnosis of Switching Circuit Failures," IEEE Transactions on Communications and Electronics, Vol. 83, Sept. 1964, pp. 509-514.
6. Huffman, D.A., "A Method for the Construction of Minimum Redundancy Codes," Proceedings of IRE, Vol. 40, Sept. 1952, pp. 1098-1101.
7. Kautz, W. H., "Fault Testing and Diagnosis in Combinational Digital Circuits," IEEE Transactions on Electronic Computers, Vol. C-17, April 1968, pp. 352-366.

8. Kautz, W. H., "Diagnosis and Testing of Cellular Arrays," Properties of Cellular Arrays for Logic and Storage, SRI Project 5876, Scientific Report 3, July 1967, pp. 119-145.
9. Kautz, W. H., "Testing for Faults in Combinational Cellular Logic Arrays," 1967 Switching and Automata Theory Symposium.
10. Maitra, K. K., "Cascaded Switching Networks of Two-input Flexible Cells," IRE Transactions on Electronic Computers, Vol. EC-11, April 1962.
11. Minnick, R. C., "A Survey of Microcellular Research," Journal of the Association for Computing Machinery, Vol. 14, April 1967, pp. 203-241.
12. Minnick, R. C., "Cutpoint Cellular Logic," IEEE Transactions on Electronic Computers, Vol. EC-13, Dec. 1964, pp. 685-698.
13. Roth, J. P., "Algorithms for Diagnosis of Automation Failures," IBM J. Res. and Develop., Vol. 10, July 1966, pp. 278-291.
14. Roth, J. P., "Diagnosis of Automata Failures: A Calculus and a Method," IBM J. Res. and Develop., Vol. 10, July, 1966, pp. 278-291.
15. Roth, J. P., Bouricins, W. G., and Schneider, P. R., "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," IEEE Transactions on Electronic Computers, Vol. EC-16, Oct. 1967, pp. 567-580.

16. Spandorfer, L. M., and Murphy, J. V., Synthesis of Logic Functions on an Array of Integrated Circuits, Scientific Rep. No. 1 for UNIVAC Project 4645, AFCRL-63-528, Contract AF 19(628)2907, Sperry Rand Corp., UNIVAC Engineering Center, Blueke 11, Pa., Oct. 1963.
17. Yau, S. S., and Orsic, M., "Fault-Diagnosis and Repair of Cutpoint Cellular Arrays," personal communication.

MONTANA STATE UNIVERSITY LIBRARIES



3 1762 10011562 3

D378
T424
cop.2

Thurber, Kenneth James
Fault location in
cellular arrays

NAME AND ADDRESS

Thurber, Kenneth James	111 201
Patel 3215 6	
R	
R	
K	

D378
T424
cop.2