1970

Fault detection in asynchronous sequential circuits

Jeng-Chuan Kau

Department:

FAULT DETECTION IN ASYNCHRONOUS SEQUENTIAL CIRCUITS

BY

JENG-CHUAN KAU, 1944-

A

THESIS

submitted to the faculty of

UNIVERSITY OF MISSOURI - ROLLA

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Rolla, Missouri

1970

Approved by

James A. Tracey (advisor) Jairus M. Taylor

Charles Hatfield

ABSTRACT

As the asynchronous sequential circuit has become more and more important to digital systems in recent years high reliability and simple maintenance of the circuit is stressed. This paper presents a fault-detection algorithm which will be applicable to most of the practical asynchronous sequential circuits. The asynchronous sequential circuit is treated from the combinatoric point of view. First the minimal set of states, both stable states and unstable states, sufficient to detect all possible faults of the circuit is found from the fault table. Then a test sequence is generated to go through these states. It is assumed that testing outputs can be added. Simple and systematic techniques are also presented for the construction of fault table and the generation of test sequence. The usefulness of this algorithm increases as the density of the stable states associated with the circuit increases.

ACKNOWLEDGEMENT

The author wishes to express his sincere appreciation and gratitude to Dr. J.H. Tracey, for his guidance and assistance during the entire course of this investigation.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. The Testing Problem

As the range of problems to which digital computing systems have been applied has widened, the task of ensuring that a computing system is operating correctly has become steadily more important. Incorrect computer operation in some applications such as the control of chemical process units and nuclear reactors, and military command and control, can be potentially disastrous. In many practical situations the synchronizing clock pulses are not available and asynchronous circuits must be designed. Moreover, within large synchronous systems it is often desirable to allow certain subsystems to operate asynchronously, thereby increasing the overall speed of operation. In this paper the methods to diagnose the asynchronous circuits are investigated.

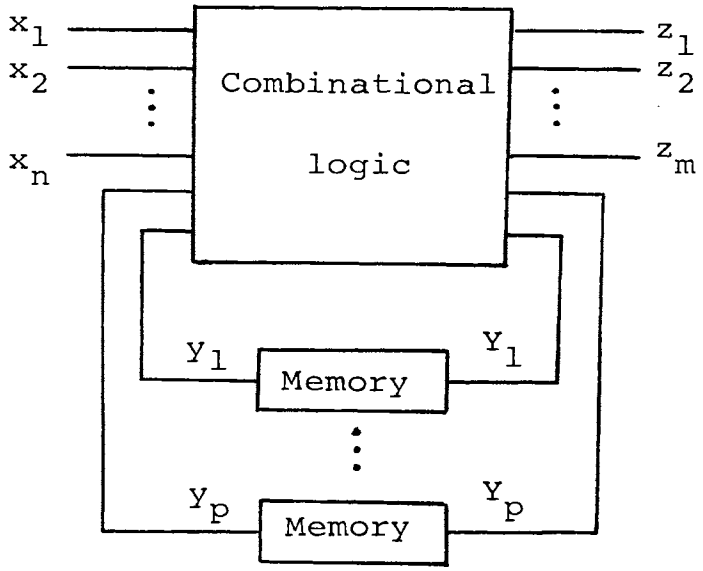
The testing problem for sequential circuits may be stated as follows: given a circuit, find a testing procedure which determines whether the circuit is performing correctly by applying signals to and measuring signals on the terminals of the circuit. There are two kinds of experiments: 1) simple experiments, which are performed on a single copy of the machine, and 2) multiple experiments, which are performed on two or more identical copies of the

machine. In practice, most machines are available in just a single copy, and therefore simple experiments are preferable to multiple ones.

B. Circuit Models and Definitions

There are two basically different kinds of sequential circuits: synchronous sequential circuits and asynchronous sequential circuits. A generalized model for the synchronous circuit and its state tables are shown in Figure 1. Each of the combinations of the values of the present state variables $(y_1, y_2, \ldots, y_p)$ on the feedback lines is called a "state" and corresponds to a row in the table. Each of the columns in the table corresponds to a combination of the values of the primary input variables $(x_1, x_2, \ldots, x_n)$. In table (b), the first entry of each cell is called the "next state" and the second entry is the "output state". The Mealy circuit is characterized by the property that the output is a function of both the state and the input. But the Moore circuit specifies the output as a function of the internal state only.

The block diagram shown in Figure 2 is the basic model for fundamental-mode asynchronous sequential circuits. The delay elements represent a "lumping" of the distributed delays in the combinational elements into single delay elements, one for each feedback variable.



(a) Block diagram

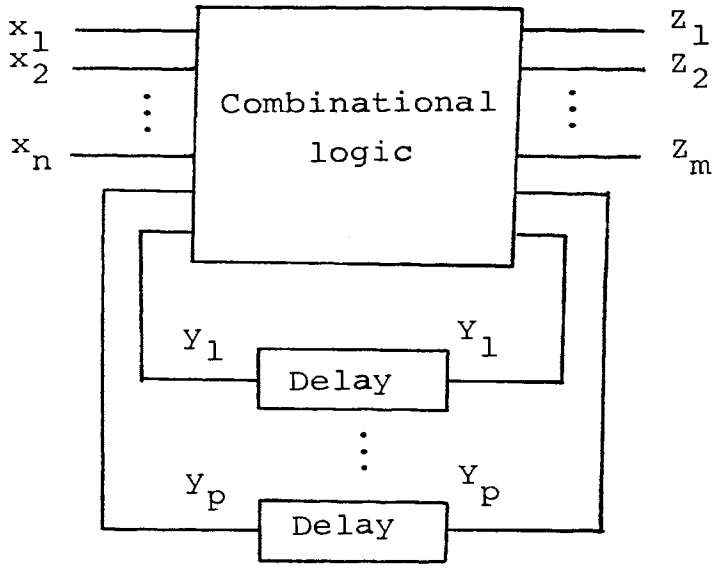
y_1y_2	X / S	x_1x_2 00	01	11	10
00	1	3.0	1.1	2.0	3.1
01	2	2.1	3.0	1.0	1.0
11	3	1.0	3.1	4.1	2.0
10	4	4.0	1.1	3.1	2.1

(b) State table of Mealy circuit

y_1y_2	X / S	x_1x_2 00	01	11	10	z
00	1	3	4	2	3	0
01	2	2	3	1	1	1
11	3	1	3	4	2	1
10	4	4	1	3	2	0

(c) State table of Moore circuit

Figure 1. The synchronous sequential circuit



(a) Block diagram

Y_1Y_2 \ x_1x_2	00	01	11	10
00	(00)	01	(00)	01
01	11	(01)	00	(01)
11	(11)	(11)	10	10
10	00	11	(10)	(10)

Y_1Y_2

(b) Transition table

Y_1Y_2 \ x_1x_2	00	01	11	10
00	00	01	10	11
01	00	01	10	11
11	10	00	10	01
10	00	00	10	01

Z_1Z_2

(c) Specified output table

Figure 2. The fundamental-mode asynchronous sequential circuit

In table (b) of Figure 2, the rows are defined by the signals at the output of the delay elements and the columns are defined by the combinations of the values of the primary input variables. A state of the circuit is a combination of the values of the variables $(x_1,x_2,\ldots,x_n, y_1,y_2,\ldots,y_p)$ and denotes a cell of the table. A row state consists of the values of the variables $(y_1,y_2,\ldots,y_p)$ only. If the next row state is the same as the present row state for a given input combination, then the entry is circled and is said to be stable. If the next row state is different from the present one, then the entry is not circled and is said to be unstable. The table (c) is same as table (b) except that the cells in this table represent the primary outputs of the circuit.

Before proceeding to discuss the literature a number of definitions are presented first.

A failure (fault) - In a logic circuit, any transformation of hardware that changes the logical character of the function realized by the hardware.

A primary input - In a logical circuit, a line that is not fed by any other line in that circuit.

A primary output - In a logical circuit, a line whose signal is accessible to the exterior of the circuit.

A test (for a failure) - A pattern of signals on primary inputs such that the value of the signal on some primary output will differ according to the presence or absence of that failure.

Experiment - The application of input sequences to the input terminals of a sequential machine and the observation of the corresponding output sequences in order to conclude something about the machine.

Distinguishing sequence - An input sequence which when applied to a sequential machine allows the initial state of the machine to be determined by observation of the corresponding output sequence.

Homing sequence - An input sequence which allows the final state of the given machine to be determined by observation of the corresponding output sequence.

Preset experiment - An experiment such that the input sequence to be applied is completely determined in advance.

Adaptive experiment - An experiment such that the input sequence consists of subsequences which are selected by observing the response to previously applied subsequences.

C. Discussion of Past Work

For a combinational network a test is just an input combination. A test detects a fault if the network output differs from the correct output when the test is applied and the fault is present. The simplest way of constructing the fault-detection tests is the use of fault tables. A fault table is a table in which there is a row for every possible test (i.e., input combination) and a column for each possible fault. A 1 is entered at the intersection of a row and column if the corresponding test detects the corresponding fault; otherwise a 0 is entered. The problem of finding a minimal set of tests then reduces to the problem of finding a minimal set of rows in the table such that they include at least one 1 in every column. In order to handle networks with larger number of variables a more systematic method of deriving the minimal test set is developed by Kautz (1).

For large networks the previously described procedure becomes prohibitive because of the size of the fault table. Armstrong (2) then provides a short-cut procedure based on the "path-sensitizing" technique. By sensitizing all the paths of the network under test, all faults in the entire network will be detected. This procedure also appears to ensure finding a sufficient set of tests to detect all detectable faults. But it may become cumbersome when the number of paths through out the network is large. Another well-known procedure of fault detection is the D-algorithm (3). If a test exists for a given failure the algorithm will find such a test. The method is to find the D-cube chains, which extends from the primary inputs to the outputs, necessarily to detect all possible faults.

The fault detection of sequential networks is more difficult than that of combinational networks. Hennie introduced the transition checking approach by use of the distinguishing sequences. Hennie's approach yields good results for machines that possess distinguishing sequences, and when the actual circuit has no more states than the correctly operating circuit. For machines which do not have a distinguishing sequence, Hennie's approach yields very long experiments, which are impractical. Further development of this approach was done by Kime (4). When the machine has no distinguishing sequence, Kime makes two modifications: the addition of testing points and the

addition of logic. Nevertheless, the length of experiments is still very long and the experiments are extremely hard to apply in any practical situation. Kohavi and Lavallee (5) then present a method for designing sequential circuits in such a way that they will be made to possess distinguishing sequences with repeated symbols by use of the additional output logic if the circuits do not have such a distinguishing sequence. The circuits thus modified have shorter fault-detection experiments than those of the original circuits.

The distinguishing sequences just discussed are the preset fixed-length distinguishing sequences (FLDS). A variable-length distinguishing sequence (VLDS) is a preset distinguishing sequence X_o such that, if the machine is started is an unknown state, the output response of the machine to some prefix of X_o (i.e., some subsequence of X_o) will identify the initial state. The length of the required prefix is a function of the initial state. I. Kohavi and Z. Kohavi (6) apply VLDS to the construction of fault-detection experiments. Consequently, the machine will have shorter and more efficient experiments when VLDS is used, provided the average length of the VLDS of the machine is shorter than that of the FLDS, and provided that a larger number of states possess the shorter prefixes of X_o.

The above fault-detection approaches for sequential circuits generally lead to long testing sequence. One reason these sequences are so long is that these approaches are actually doing machine identification rather than simply fault detection. Based on Armstrong's theory, Kohavi (7) finds the minimal fault-detection tests set of the combinational networks from their Karnaugh maps. Kohavi then applies his approach to the generation of testing sequence for the sequential networks.

Let the experiment for sequential networks be divided into two parts: the first part identifies that the given n-state machine has n states and the second part distinguishes between the given n-state machine and those n-state machines which the given machine can be transformed into as a result of some fault. Kohavi's method generates a much shorter test sequence for the second part of the experiment than previously discussed methods. The class of n-state machines that the given machine can be transformed into as a result of some malfunction form a subset of all n-state machines. In order to distinguish between the given machine and the above subset of machines, shorter experiments are required than in the case of distinguishing between the given machine and the entire set of n-state machines. In other words, one need not identify the n distinct states of the machine and check all the transitions according to the given state table.

Based on the above idea and Hughes' method (8), a fault-detection algorithm for asynchronous sequential circuits is presented in the following chapter.

II. THE ALGORITHM

A. Assumptions

The asynchronous sequential circuit presents a somewhat more difficult testing problem. In general, an asynchronous circuit is not as well behaved as the corresponding synchronous circuit. Before going into the details, it is necessary to make some assumptions.

1) The circuits considered here are fundamental-mode asynchronous sequential circuits. The inputs are levels, and are never changed unless the circuit is internally stable. It is required that only one input is changed at a time.

2) The machines are strongly connected. This provides the capability of always establishing the feedback variables to be zero before beginning the test.

3) The circuitry must be irredundant. A circuit is irredundant if its Boolean expressions are in minimal forms. The redundant circuit consists of redundant literals,where by literal we mean an appearance of a variable or its complement. Without this restriction, the function realized

with a fault in the redundant circuitry is equal to the function realized without the fault.

4) The flow table is free from critical races and oscillations.

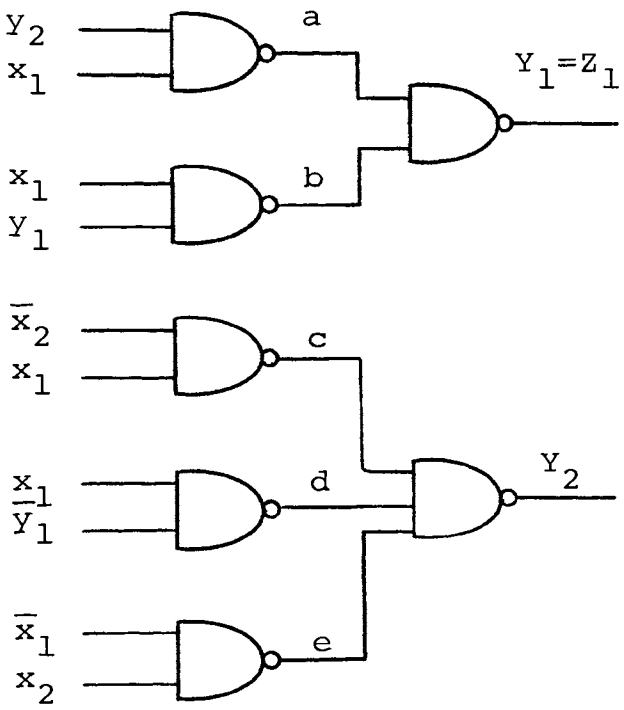
5) Only single faults are considered.

6) The faults are permanent faults due to component failures and manifest themselves as stuck-at-one (s-a-1) or stuck-at-zero (s-a-o).

In treating the single fault case one implies that fault-detection test will be run frequently enough so that, in general, multiple faults will not occur. From a practical point of view this is not a severe limitation, since most circuits are reliable enough so that the probability of occurrence of multiple faults is very small. Other assumptions will be given as the discussion progresses.

B. Algorithm

The asynchronous sequential circuit shown in Figure 3, with x_1x_2 as primary inputs and Z_1Z_2 as primary outputs, will be used to illustrate the algorithm. For convenience, the output portion (without feedbacks) of the circuit will be considered as the output network although it is regarded as part of the single block of the combinational logic as shown in Figure 2. Thus, the lower part of the circuit in Figure 3 will be called the output network.



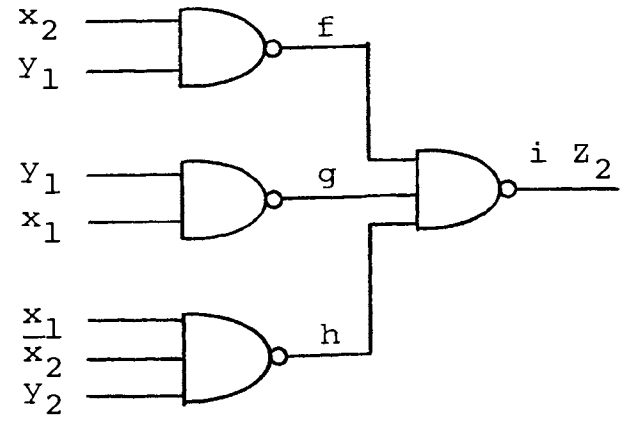
Y_1Y_2 \ x_1x_2	00	01	11	10
00	(00)	01	01	01
01	00	(01)	11	11
11	00	01	10	(11)
10	00	01	(10)	11

Y_1Y_2

Transition table

$$Y_1 = x_1y_2 + x_1y_1 = \bar{a} + \bar{b}$$

$$Y_2 = x_1\bar{x}_2 + x_1\bar{y}_1 + \bar{x}_1x_2 = \bar{c} + \bar{d} + \bar{e}$$



Z_1Z_2 \ x_1x_2	00	01	11	10
00	00	00	00	00
01	00	00	10	11
11	00	01	11	11
10	00	01	11	11

Z_1Z_2

Output table

$$Z_1 = x_1y_2 + x_1y_1$$

$$z_2 = x_2y_1 + x_1y_1 + x_1\bar{x}_2y_2 = \bar{f} + \bar{g} + \bar{h}$$

Figure 3. Circuit for illustrating algorithm

In the remainder of this paper when input and output are spoken of, they will mean primary input and primary output respectively unless otherwise specified. Since the delay is not a physical element, they are not shown in the circuit and will not be shown on subsequent figures.

Figure 4 is the fault table of Figure 3 under stable state conditions. The meaning of each column in the table is as follows:

S	:	all stable state conditions; and, under stable state conditions;
$(N.S.)_c$	:	the correct values of next state variables;
$(N.S.)_f$	:	the values of next state variables for the s-a-1 and s-a-o faults designated by the subscript;
Z_c	:	the correct outputs;
Z_f	:	the values of the outputs under s-a-1 and s-a-o faults of those lines related to the outputs.

In constructing the fault table, all possible faults of the lines that are related to the next state variables should be included in column $(N.S.)_f$. Only those faults associated with the output network and not included in column $(N.S.)_f$ are contained in column Z_f. The values of the next state variables under $(N.S.)_f$ are circled whenever they are different from the correct values under $(N.S.)_c$ of the same row.

S				$(N.S.)_c$		$(N.S.)_f$									
x_1	x_2	y_1	y_2	Y_1	Y_2	$(x_1)_1$	$(x_1)_0$	$(\bar{x}_2)_1$	$(\bar{x}_2)_0$	$(\bar{y}_1)_1$	$(\bar{y}_1)_0$	$(\bar{x}_1)_1$	$(\bar{x}_1)_0$	$(x_2)_1$	$(x_2)_0$
0	0	0	0	0	0	11	00	00	00	00	00	00	00	01	00
0	1	0	1	0	1	10	01	01	01	01	01	01	00	01	00
1	0	1	1	1	1	11	00	11	10	11	11	11	11	10	11
1	1	1	0	1	0	10	01	11	10	11	10	11	10	10	11

$(N.S.)_f$													
$(a)_1$	$(a)_0$	$(b)_1$	$(b)_0$	$(c)_1$	$(c)_0$	$(d)_1$	$(d)_0$	$(e)_1$	$(e)_0$	$(Y_1)_1$	$(Y_1)_0$	$(Y_2)_1$	$(Y_2)_0$
00	10	00	10	00	01	00	01	00	01	10	00	01	00
01	11	01	11	01	01	01	01	00	01	11	01	01	00
11	11	11	11	10	11	11	11	11	11	11	01	11	10
10	10	O.S.	10	10	11	10	11	10	11	10	01	11	10

Figure 4. Fault table for Figure 3

Z_c	Z_f							
Z_2	$(f)_1$	$(f)_o$	$(g)_1$	$(g)_o$	$(h)_1$	$(h)_o$	$(i)_1$	$(i)_o$
0	0	①	0	①	0	①	①	0
0	0	①	0	①	0	①	①	0
1	1	1	1	1	1	1	1	⓪
1	1	1	1	1	1	1	1	⓪

Figure 4. Fault table for Figure 3 (cont.)

The same is true for Z_f. It should be emphasized that the values in the table are all in steady states, except the entry "O.S." under column $(b)_1$. The symbol "O.S." indicates that the circuit will oscillate if one tries to put it in stable state $x_1x_2Y_1Y_2 = 1110$ under s-a-1 fault on b. A suggested short-cut technique for deriving the entries under $(N.S.)_f$ consists of writing the state equation as a function of the inputs and internal state variables for the fault specified, then forming the transition table. By comparing this faulty transition table with the original table, one can determine the proper entry. For example, the column $(x_1)_1$ (x_1 s-a-1) is found from the equations:

$$Y_1 = 1y_2 + 1y_1 = y_2 + y_1$$

$$Y_2 = 1\bar{x}_2 + 1\bar{y}_1 + 0x_2 = \bar{x}_2 + \bar{y}_1.$$

The transition table becomes

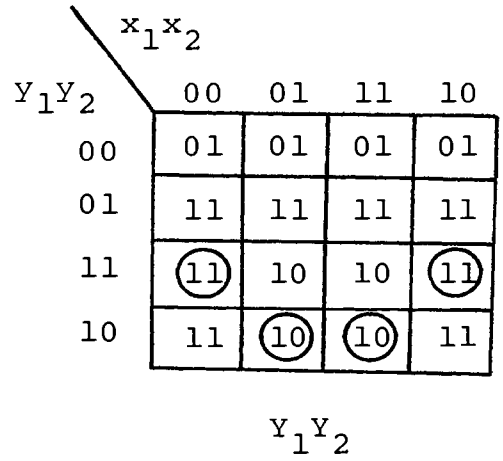


Figure 5. The transition table for x_1 s-a-1

Therefore, the stable state $x_1x_2y_1y_2$ = 0000 is changed to the faulty stable state 0011 and state 0101 to 0110 owing to the fault $(x_1)_1$. In other words, in trying to get to stable state 0000 and 0101 the next state variables Y_1Y_2 become 11 and 10 respectively instead of 00 and 01 in the presence of this fault. Thus, these two entries are circled, while the other states remain unchanged.

It is seen that the s-a-1 and s-a-o faults of Y_1 and Y_2 are not considered. The reason is that the feedback is a direct connection. The signals labeled y_1 (or y_2) and Y_1 (or Y_2) in Figure 3 will really be the same signal. Therefore, if one is stuck, it is assumed that the other is stuck at the same value.

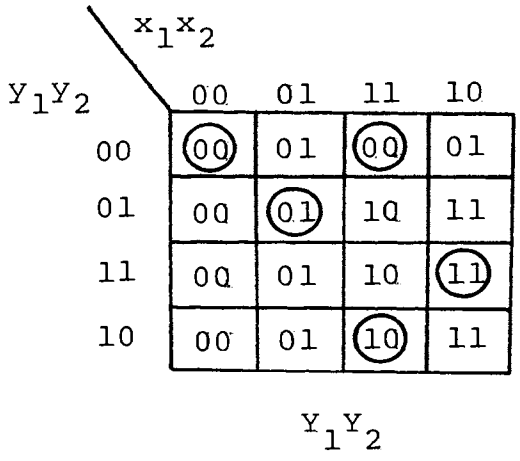
It is apparent from the fault table that stable states $x_1x_2y_1y_2$ = 0101, 1011, and 1110 are sufficient to detect

all faults except $(\bar{y}_1)_o$, $(a)_1$, $(d)_1$, $(f)_1$, $(g)_1$, and $(h)_1$. If the fault table is much larger, Kautz's method (1) should be applied to simplify the fault table for obtaining the minimal set of test states.

When a circuit has a fault some of the following conditions may occur.

1) Some stable states become unstable.
2) Some unstable states become stable.
3) The circuit is oscillating after some input sequence is applied.

Now, in order to detect the faults undetectable by stable states, it is necessary to make use of the second property to see if some unstable state becomes stable owing to these six faults: $(\bar{y}_1)_o$, $(a)_1$, $(d)_1$, $(f)_1$, $(g)_1$, and $(h)_1$. Take $(\bar{y}_1)_o$ for instance; the transition table is changed to that shown in Figure 6. The previously unstable state 1100 becomes stable. Similarly unstable states 1100, 1101, and 1001 are stable in the presence of $(a)_1$ fault, while $(d)_1$ produces the same result as $(\bar{y}_1)_o$. However $(f)_1$, $(g)_1$, and $(h)_1$ do not cause any unstable state to become stable. This is because lines f, g, and h are not related to the next state variables. They belong to the combinational output network.



$$Y_1 = x_1y_2 + x_1y_1$$
$$Y_2 = x_1\bar{x}_2 + \bar{x}_1x_2$$

Figure 6. The transition table for $\bar{y}_1$ s-a-o

Attention is now given to the equations:

$$f = \bar{x}_2 + y_1$$
$$g = \bar{x}_1 + \bar{y}_1$$
$$h = \bar{x}_1 + x_2 + \bar{y}_2.$$

Under stable state conditions f, g, and h assume values of 0 and 1 as shown in Figure 7. To illustrate how one can detect these faults, the variable f will be examined. When f is s-a-1 and the stable state 1110 is reached, the value on f differs according to the presence or absence of the fault. If f is made a testing output, the undetectable fault $(f)_1$ can now be detected. The same is true for $(g)_1$ and $(h)_1$ faults. As a result, the two stable states 1011

and 1110 should be used to detect these three faults: $(f)_1$, $(g)_1$, and $(h)_1$. Consequently, the set of states $x_1x_2y_1y_2$ = {0101, 1100, 1110, 1011} completely detects all faults when testing outputs are added.

x_1	x_2	y_1	y_2		f	g	h	i
0	0	0	0		1	1	1	0
0	1	0	1		1	1	1	0
1	0	1	1		1	0	0	1
1	1	1	0		0	0	1	1

Figure 7. Stable state conditions for the output network of Figure 3

After finding the minimal set of test states it is necessary to generate a test sequence to go through all these states. It should be emphasized that the unstable state is not treated as a stable state in the preparation of this experiment. Unstable states are used to detect some particular faults, but the unstable states become stable only when particular faults are present. To provide a systematic method for deriving a short test sequence the tree approach (9) is used. The transition table of Figure 3 is given again in Figure 8 for illustration. Every stable state and unstable state necessary to be tested are labeled with different numbers. And the states contained in the

test set are marked with "*". The state-transition tree is shown in Figure 9. The circuit is first reset to the

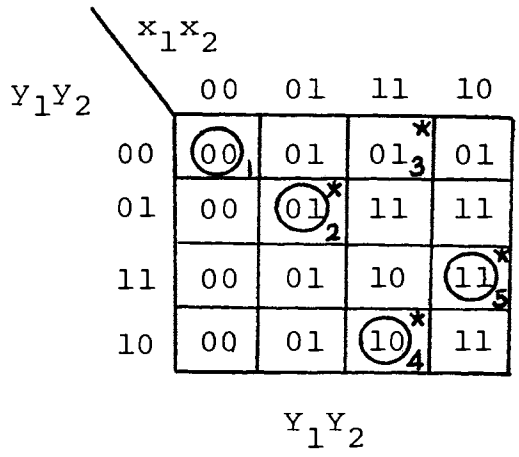


Figure 8. The transition table for Figure 3

starting state $x_1x_2y_1y_2$ = 0000 by x_1x_2 = 00. If the circuit can not be reset to a starting state by a homing sequence, either preset sequence or adaptive sequence, we may conclude that there exists a fault and thus complete the experiment without any further test. The state at a particular level, say J, will be a terminal state whenever it appeared at the level less than J. Starting from state 1, the circuit goes to state 2 after inputs x_1x_2 = 01 are applied. Then it goes to state 1 if next inputs are 00. Now this branch is terminated since state 1 had appeared at level 1. In the same manner the whole tree will be generated.

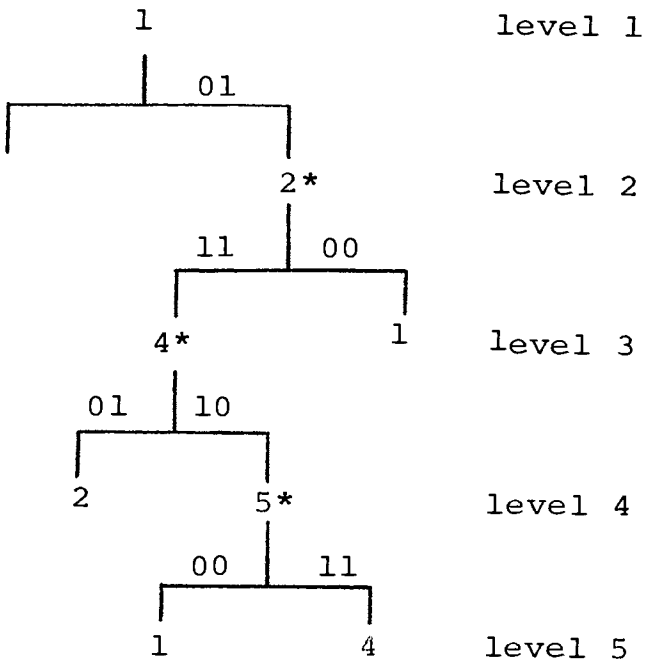


Figure 9. State-transition tree for Figure 8

As shown in the tree every state has only two possible next states since there are two inputs associated with the circuit and one input change is allowed at a time. The unstable state, if any, will be put directly above its stable state and bracketed in order to distinguish it from stable state. By inspection, it is quite straightforward to generate a test sequence to go through the desired states. Unfortunately, state 3 cannot be reached because of the limitation that multiple input changes are not allowed. As stated before, state 3 is used to detect either $(\bar{y}_1)_0$, $(d)_1$ or $(a)_1$. It is, therefore, necessary

to pick up unstable state 1101 for the detection of $(a)_1$. Note that $(\bar{y}_1)_o$ can be detected under stable state 0101 (in the set of test states) if testing output is added. However, $(d)_1$ fault is undetectable in this particular example. The experiment is as follows:

Input	x_1x_2	00	01	11	10
Output	$Y_2Z_1Z_2$	000	100	011	111
	$\bar{y}_1fgh$	xxxx	1xxx	x0xx	xx00

where Y_2, $\bar{y}_1$, f, g, and h are used as testing outputs and the symbol "x" represents "don't care".

Since the worst case is considered and only one input change at a time is allowed, the $(d)_1$ fault is undetectable by this algorithm. It is interesting to note that the above experiment might detect this fault. From Figure 6 when the machine is in stable state 0101 and inputs 11 are applied it might go to stable state 1100 owing to the critical race. Thus, the $(d)_1$ fault could be detected if it exists.

In general, the density of stable states in a practical circuit is much higher than the above example where only 1/4 of the states are stable. Therefore, the probability of the existence of practical circuits with faults which are undetectable by this algorithm seems to be small. It should be pointed out here that the tree approach may become very tedious for large networks. In this way, one may choose

the "trial and error" procedure to derive the test sequence.

The testing procedure can be listed now as follows. It is assumed that a circuit and its transition table and output table are given, and that testing outputs can be added.

1) Construct the fault table under stable state conditions, simulating s-a-o and s-a-1 faults on each line.

2) Under column $(N.S.)_f$ of the fault table if not all faults are detected under stable state conditions, find the unstable states which will become stable in the presence of the undetectable faults.

3) If not all faults in columns $(N.S.)_f$ and Z_f are detected, find the stable states which will indicate the faults when testing outputs are added to those lines.

4) Find the minimal set of test states sufficient to detect all faults.

5) Generate the state-transition tree to derive a shortest test sequence to go through the states found in 4).

There exists a subset of the former class of asynchronous sequential circuits. They are the circuits with the outputs

identical to the next state variables. In this case, step 3) of the above procedure is unnecessary. An example in the next chapter is presented to illustrate this case.

C. Evaluation of the Shortest Test Sequence

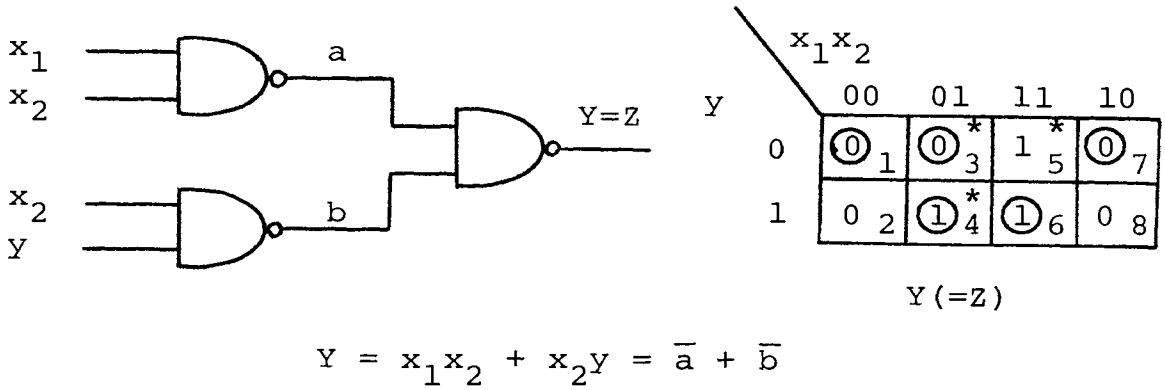
In this section the minimization of test sequences is investigated. In a circuit which has some particular fault some stable states become unstable and some unstable states become stable. If all unstable states are included in the fault table and optimun use is made of the fault table, it is possible to find a shorter test sequence for some circuit. The algorithm discussed previously is used to construct the fault table under stable state conditions only; if all stable states or some subset of them are sufficient to detect all faults, then the test sequence is generated; otherwise the necessary unstable states are added to the test sequence to detect the faults undetectable by stable states, or testing outputs are added.

To illustrate the idea presented above consider the following example taken from Hughes (8). Each decimal number in the transition table identifies a state.

The fault table is shown in Figure 11. The minimal set of test states for this circuit is

$$x_1x_2Y = \left\{ \begin{array}{cccc} 010^* & 011^* & 110^* & \begin{array}{c}100\\001\\101\end{array} \end{array} \right\} .$$

The meaning of the above notation is that the states marked with "*" are essential and that one state out of 100, 001,



$$Y = x_1x_2 + x_2y = \bar{a} + \bar{b}$$

Figure 10. Circuit for evaluating the shortest test sequence

and 101 can be selected. The state-transition tree is shown in Figure 12. From it the shortest test sequence x_1x_2 = 00 01 11 01 00 is found. When the algorithm described in last section is applied, the state 100 will be selected. Thus, the test sequence becomes x_1x_2 = 00 01 11 01 00 10. Note that the length of this test sequence is longer than the former. However, it must be pointed out that not all circuits have this property (i.e., their test sequence can be further minimized) and that the length can not be made much shorter by this method although it causes the fault table to be more cumbersome. For example, this method is not useful to the two examples in the next chapter.

S			Z_c	Z_f						
x_1	x_2	y	Z	$(x_1)_1$	$(x_1)_0$	$(x_2)_1$	$(x_2)_0$	$(a)_1$	$(a)_0$	$(b)_1$
0	0	0	0	0	0	0	0	0	①	0
0	1	0	0	①	0	0	0	0	①	0
1	0	0	0	0	0	①	0	0	①	0
0	1	1	1	1	1	1	⓪	1	1	⓪
1	1	1	1	1	1	1	⓪	1	1	1
1	1	0	1	1	⓪	1	⓪	⓪	1	1
0	0	1	0	0	0	①	0	0	①	0
1	0	1	0	0	0	①	0	0	①	0

Z_f		
$(b)_0$	$(Y)_1$	$(Y)_0$
①	①	0
①	①	0
①	①	0
1	1	⓪
1	1	⓪
1	1	⓪
①	①	0
①	①	0

Figure 11. Fault table for Figure 10

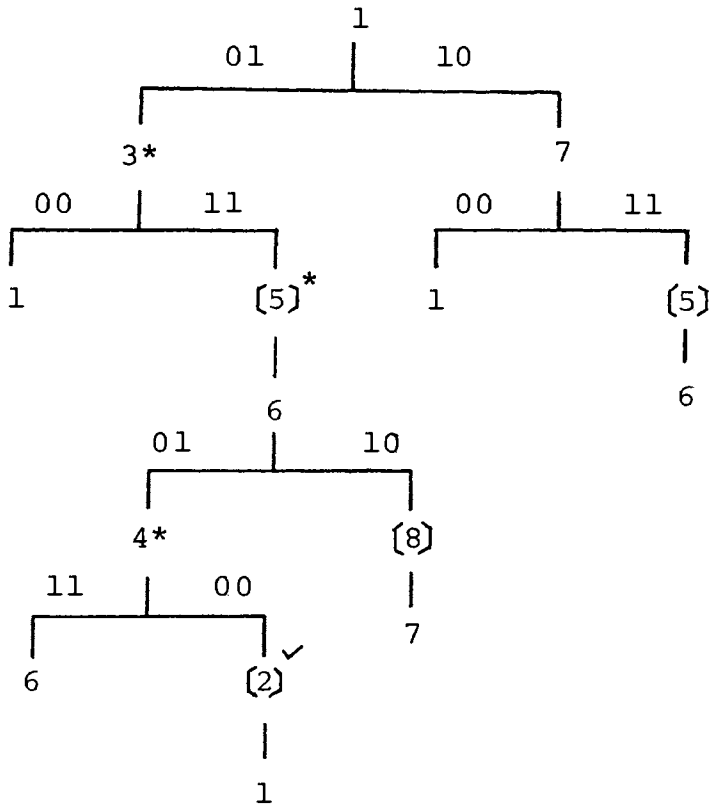


Figure 12. State-transition tree for Figure 10

III. EXAMPLES

In this chapter two examples are presented to illustrate the algorithm in more detail.

Example 1

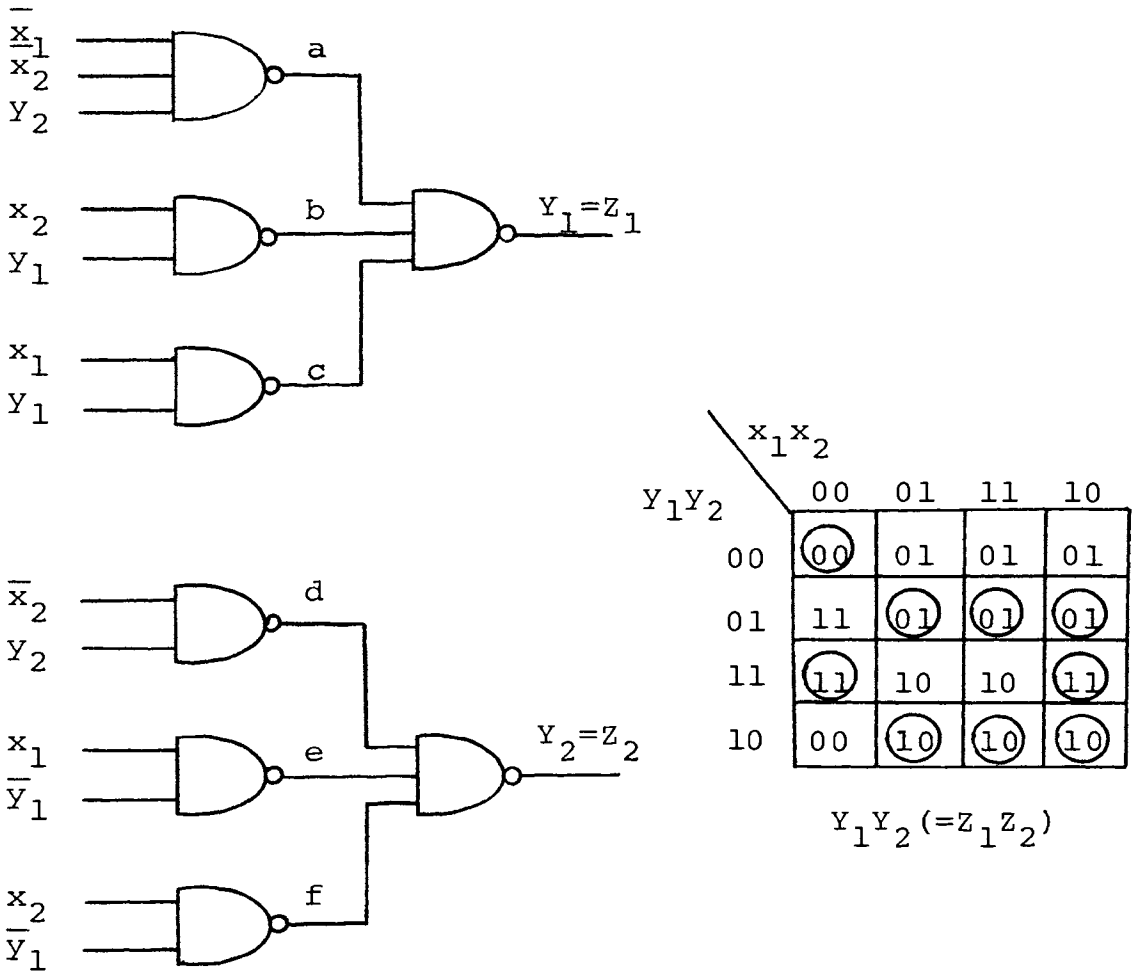
The circuit in Figure 14 (taken from Kime (4)) has the outputs identical to next state variables. Step 3) of the testing procedure is to be omitted. The fault table is shown in Figure 15. It is seen from the fault table that the $(e)_1$ fault can not be detected under stable state conditions. But when line e is stuck at one the transition table becomes that shown in Figure 13.

Y_1Y_2 \ x_1x_2	00	01	11	10
00	(00)	01	01	(00)
01	11	(01)	(01)	(01)
11	(11)	10	10	(11)
10	10	(10)	(10)	(10)

Y_1Y_2 $(=Z_1Z_2)$

$$Y_1 = \bar{x}_1\bar{x}_2Y_2 + x_2Y_1 + x_1Y_1$$
$$Y_2 = \bar{x}_2Y_2 + x_2\bar{Y}_1$$

Figure 13. The transition table for e s-a-1



$$Y_1 = \bar{x}_1\bar{x}_2y_2 + x_2y_1 + x_1y_1 = \bar{a} + \bar{b} + \bar{c}$$

$$Y_2 = \bar{x}_2y_2 + x_1\bar{y}_1 + x_2\bar{y}_1 = \bar{d} + \bar{e} + \bar{f}$$

Figure 14. Circuit for example 1

S				Z_c		Z_f									
x_1	x_2	y_1	y_2	Z_1	Z_2	$(\bar{x}_1)_1$	$(\bar{x}_1)_0$	$(\bar{x}_2)_1$	$(\bar{x}_2)_0$	$(\bar{y}_1)_1$	$(\bar{y}_1)_0$	$(x_2)_1$	$(x_2)_0$	$(x_1)_1$	$(x_1)_0$
0	0	0	0	0	0	00	00	00	00	00	00	01	00	01	00
0	0	1	1	1	1	11	01	11	00	11	11	10	11	11	11
0	1	0	1	0	1	01	01	11	01	01	00	01	11	01	01
0	1	1	0	1	0	10	10	10	10	11	10	10	00	10	10
1	1	0	1	0	1	01	01	01	01	01	00	01	01	01	01
1	1	1	0	1	0	10	10	10	10	11	10	10	11	10	10
1	0	0	1	0	1	11	01	01	01	01	01	01	01	01	11
1	0	1	1	1	1	11	11	11	10	11	11	10	11	11	11
1	0	1	0	1	0	10	10	10	10	11	10	10	11	10	00

Figure 15. Fault table for Figure 14

Z_f															
$(a)_1$	$(a)_0$	$(b)_1$	$(b)_0$	$(c)_1$	$(c)_0$	$(d)_1$	$(d)_0$	$(e)_1$	$(e)_0$	$(f)_1$	$(f)_0$	$(Y_1)_1$	$(Y_1)_0$	$(Y_2)_1$	$(Y_2)_0$
00	(1) 0	00	(1) 0	00	(1) 0	00	0 (1)	00	0 (1)	00	0 (1)	(1) 0	00	(11)	00
(0) 1	11	11	11	11	11	(00)	11	11	11	11	11	11	(0) 1	11	(00)
01	(1) 1	01	(1) 1	01	(1) 1	01	01	01	01	0 (0)	01	(10)	01	01	0 (0)
10	10	(01)	10	10	10	10	1 (1)	10	1 (1)	10	1 (1)	10	(01)	1 (1)	10
01	(1) 1	01	(1) 1	01	(1) 1	01	01	01	01	01	01	(10)	01	01	0 (0)
10	10	10	10	10	10	10	1 (1)	10	1 (1)	10	1 (1)	10	(01)	1 (1)	10
01	(1) 1	01	(1) 1	01	(1) 1	01	01	01	01	01	01	(1) 1	01	01	0 (0)
11	11	11	11	(0) 1	11	1 (0)	11	11	11	11	11	11	(0) 1	11	1 (0)
10	10	10	10	(01)	10	10	1 (1)	10	1 (1)	10	1 (1)	10	(01)	1 (1)	10

Figure 15. Fault table for Figure 14 (cont.)

Therefore, unstable state $x_1x_2y_1y_2$ = 1000 is included in the set of test states $x_1x_2y_1y_2$ = {0000, 0011, 0101, 0110, 1001, 1011, 1000}. Figure 16 shows the transition table with desired states marked and the state-transition tree.

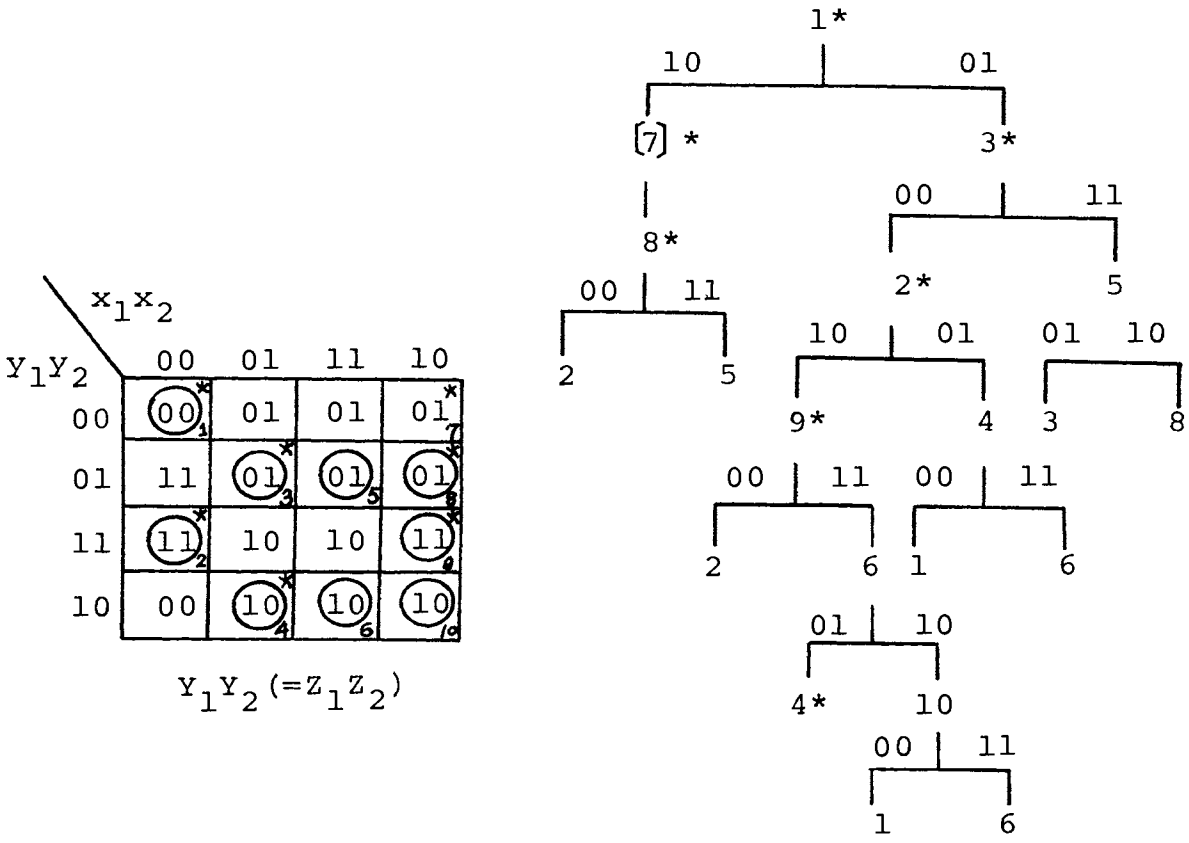


Figure 16. Transition table and state-transition tree for Figure 14

The shortest test sequence is

Input x_1x_2	00	10	11	01	00	10	11	01
Output z_1z_2	00	01	01	01	11	11	10	10.

In order to place the circuit in initial state $x_1x_2y_1y_2$ = 0000 an adaptive homing sequence should be used. First,

apply 00. If the output is 00, we then start the test sequence; otherwise another input sequence, 01 00, must be applied before beginning the test.

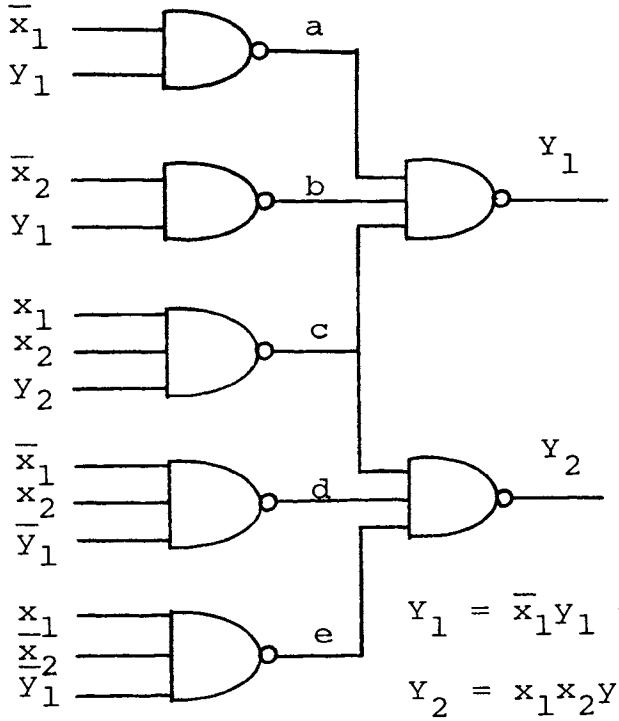
In his paper Kime obtained a experiment for this circuit. It required 31 input symbols in spite of the fact that numerous short cuts and tricks were used in its construction. This is almost 4 times longer than the above result which requires 8 symbols.

Example 2

Since the circuit shown in Figure 17 is a conventional asynchronous sequential circuit, all steps of the procedure should be applied. Figure 18 is its fault table. From it the set of test states $x_1x_2y_1y_2$ = {0010, 0101, 0110, 1100, 1111, 1001, 1010} is found. Note that all stable states except 0000 are included in this set. Therefore it is unnecessary to construct the state-transition tree. The test sequence is the following:

Input	x_1x_2	00	10	11	10	00	01	11	01
Output	$Y_1Y_2Z_1Z_2$	0000	0110	1101	1011	1010	1000	0011	0101

where Y_1 and Y_2 are testing outputs. Again, an adaptive homing sequence (x_1x_2 = 00 or 00 01 11 01 00) is used to put the circuit in the starting state $x_1x_2y_1y_2$ = 0000.

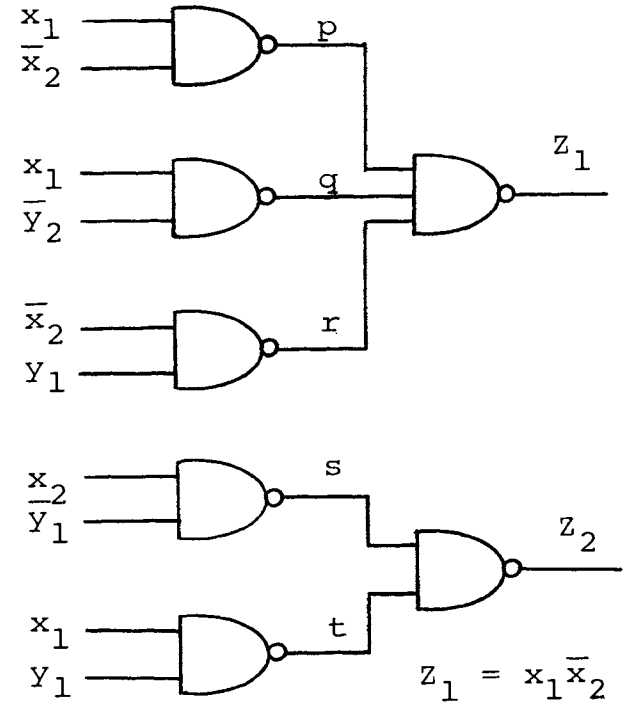


Y_1Y_2 \ x_1x_2	00	01	11	10
00	(00)	01	(00)	01
01	00	(01)	11	(01)
11	10	10	(11)	10
10	(10)	(10)	00	(10)

Y_1Y_2

$$Y_1 = \bar{x}_1y_1 + \bar{x}_2y_1 + x_1x_2y_2 = \bar{a} + \bar{b} + \bar{c}$$

$$Y_2 = x_1x_2y_2 + \bar{x}_1x_2\bar{y}_1 + x_1\bar{x}_2\bar{y}_1$$
$$= \bar{c} + \bar{d} + \bar{e}$$



Z_1Z_2 \ x_1x_2	00	01	11	10
00	00	01	11	10
01	00	01	01	10
11	10	00	01	11
10	10	00	11	11

Z_1Z_2

$$Z_1 = x_1\bar{x}_2 + x_1\bar{y}_2 + \bar{x}_2y_1 = \bar{p} + \bar{q} + \bar{r}$$

$$Z_2 = x_2\bar{y}_1 + x_1y_1 = \bar{s} + \bar{t}$$

Figure 17. Circuit for example 2

S				$(N.S.)_o$		$(N.S.)_f$									
x_1	x_2	y_1	y_2	Y_1	Y_2	$(\bar{x}_1)_1$	$(\bar{x}_1)_0$	$(\bar{x}_2)_1$	$(\bar{x}_2)_0$	$(x_1)_1$	$(x_1)_0$	$(x_2)_1$	$(x_2)_0$	$(\bar{Y}_1)_1$	$(\bar{Y}_1)_0$
0	0	0	0	0	0	00	00	00	00	0(1)	00	0(1)	00	00	00
0	0	1	0	1	0	10	10	10	10	10	10	10	10	10	10
0	1	0	1	0	1	01	0(0)	01	01	(1)1	01	01	0(0)	01	0(0)
0	1	1	0	1	0	10	(0)0	10	10	(0)0	10	10	10	1(1)	10
1	1	0	0	0	0	(11)	00	(11)	00	00	0(1)	00	0(1)	00	00
1	1	1	1	1	1	11	11	11	11	11	1(0)	11	1(0)	11	11
1	0	0	1	0	1	01	01	01	0(0)	01	0(0)	(1)1	01	01	0(0)
1	0	1	0	1	0	10	10	10	(0)0	10	10	(0)0	10	1(1)	10

Figure 18. Fault table for Figure 17

$(N.S.)_f$														
$(a)_1$	$(a)_0$	$(b)_1$	$(b)_0$	$(c)_1$	$(c)_0$	$(d)_1$	$(d)_0$	$(e)_1$	$(e)_0$	$(Y_1)_1$	$(Y_1)_0$	$(Y_2)_1$	$(Y_2)_0$
00	(1) 0	00	(1) 0	00	(11)	00	0 (1)	00	0 (1)	(1) 0	00	0 (1)	00
10	10	10	10	10	1 (1)	10	1 (1)	10	1 (1)	10	(0) 0	1 (1)	10
01	(1) 1	01	(1) 1	01	(1) 1	0 (0)	01	01	01	(10)	01	01	0 (0)
(01)	10	10	10	10	1 (1)	10	1 (1)	10	1 (1)	10	(01)	1 (1)	10
00	(1) 0	00	(1) 0	00	(11)	00	0 (1)	00	0 (1)	(1) 0	00	(11)	00
11	11	11	11	(00)	11	11	11	11	11	11	(0) 1	11	(00)
01	(1) 1	01	(1) 1	01	(1) 1	01	01	0 (0)	01	(10)	01	01	0 (0)
10	10	(01)	10	10	1 (1)	10	1 (1)	10	1 (1)	10	(01)	1 (1)	10

Figure 18. Fault table for Figure 17 (cont.)

Z_c		Z_f															
Z_1	Z_2	$(\bar{y}_2)_1$	$(\bar{y}_2)_0$	$(P)_1$	$(P)_0$	$(q)_1$	$(q)_0$	$(r)_1$	$(r)_0$	$(s)_1$	$(s)_0$	$(t)_1$	$(t)_0$	$(Z_1)_1$	$(Z_1)_0$	$(Z_2)_1$	$(Z_2)_0$
0	0	00	00	00	(1) 0	00	(1) 0	00	(1) 0	00	0 (1)	00	0 (1)	(1) 0	00	0 (1)	00
1	0	10	10	10	10	10	10	(0) 0	10	10	1 (1)	10	1 (1)	10	(0) 0	1 (1)	10
0	1	01	01	01	(1) 1	01	(1) 1	01	(1) 1	0 (0)	01	01	01	(1) 1	01	01	0 (0)
0	0	00	00	00	1 0	00	(1) 0	00	(1) 0	00	0 (1)	00	0 (1)	(1) 0	00	0 (1)	00
1	1	11	(0) 1	11	11	(0) 1	11	11	11	1 (0)	11	11	11	11	(0) 1	11	1 (0)
0	1	(1) 1	01	01	(1) 1	01	(1) 1	01	(1) 1	01	01	0 (0)	01	(1) 1	01	01	0 (0)
1	0	10	10	(0) 0	10	10	10	10	10	10	1 (1)	10	1 (1)	10	(0) 0	1 (1)	10
1	1	11	11	11	11	11	11	11	11	1 1	11	1 (0)	11	11	(0) 1	11	1 (0)

Figure 18. Fault table for Figure 17 (cont.)

IV. CONCLUSIONS

The algorithm developed in this paper presents a method for detecting a single failure in an asynchronous sequential circuit which is treated from the combinatoric point of view. Since the states sufficient to detect all possible faults of the circuit form a subset of the total states, the experiments are to check only this subset of states rather than all the transitions of the states. In this way, a very short test sequence can be generated. The effectiveness of the algorithm depends on the density of stable states. Attempts at finding a practical circuit with faults which are undetectable by this algorithm have been unsuccessful. The only disadvantage, and one which is common to most of the fault-detection methods, is that when the circuit has a large number of inputs and state variables the procedures of the algorithm may become quite lengthy.

V. BIBLIOGRAPHY

1. Kautz, W.H. "Fault Testing and Diagnosis in Combinational Digital Circuits", IEEE Trans. on Computers, Vol. C-17, No. 4, pp. 352-366, April 1968.

2. Armstrong, D.B., "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets", IEEE Trans. on Electronic Computers, Vol. EC-15, pp. 66-73, February 1966.

3. Roth, P.J., "Diagnosis of Automata Failures: A Calculus and a Method", IBM Journal, Vol. 10, No. 4, pp. 278-291, July 1966.

4. Kime, C.R., "A Failure Detection Method for Sequential Circuits", Dept. Elec. Engrg., Univ. of Iowa, Iowa City, Tech. Report 66-13, January 1966.

5. Kohavi, Z. and Lavallee, P., "Design of Sequential Machine with Fault Detection Capabilities", IEEE Trans. on Computers, Vol. EC-16, No. 4, pp. 473-484, August 1967.

6. Kohavi, I. and Kohavi, Z., "Variable-Length Distinguishing Sequences and Their Application to the Design of Fault Detection Experiments", IEEE Trans. on Computers, Vol. C-17, pp. 792-795, August 1968.

7. Kohave, I., "Fault Diagnosis of Logical Circuits", IEEE Conference Record of 1969 10th Annual Symposium on Switching and Automata Theory, pp. 166-173.

8. Hughes, V.W., Jr., "Fault Diagnosis of Sequential Circuits", M.S. Thesis, University of Missouri-Rolla, T2272, 1969.

9. Booth, T.L., Sequential Machines and Automata Theory, New York: John Wiley and Sons, Inc., 1968.

VI. VITA

Jeng-Chuan Kau was born on December 20, 1944, in Taipei, Taiwan, Republic of China. He received his primary and secondary education there and obtained his Bechelor of Science Degree in Electrical Engineering in 1967 from Tatung Institute of Technology in Taiwan, Republic of China.

He has been enrolled in the Graduate School of the University of Missouri-Rolla since September, 1969.