AFOSR-TR-71-2625

FAULT DETECTION AND LOCATION IN SEQUENTIAL CELLULAR ARRAYS

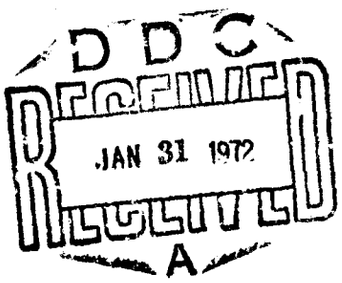
by

Chia-Hsiaing Sung and C. L. Coates

Department of Electrical Engineering

Technical Report No. 106

July 5, 1971



INFORMATION SYSTEMS RESEARCH LABORATORY

ELECTRONICS RESEARCH CENTER

THE UNIVERSITY OF TEXAS AT AUSTIN

Austin, Texas 78712

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)	2a REPORT SECURITY CLASSIFICATION
The University of Texas at Austin Electronics Research Center Austin, Texas 78712	UNCLASSIFIED
	2b GROUP

3 REPORT TITLE

FAULT DETECTION AND LOCATION IN SEQUENTIAL CELLULAR ARRAYS

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

Scientific Interim

5 AUTHOR(S) (Last name, first name, initial)

Chia-Hsiang Sung
C. L. Coates

6 REPORT DATE	7a TOTAL NO OF PAGES	7b NO OF REFS
5 July 1971	80	19

8a CONTRACT OR GRANT NO	9a ORIGINATOR'S REPORT NUMBER(S)
AFOSR Contract F44620-71-C-0091 b PROJECT NO 4751 c 61102F d 681305	JSEP, Technical Report No. 106 9b OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AFOSR-TR-71-2625

10 AVAILABILITY/LIMITATION NOTICES

1. This document has been approved for public release and sale; its distribution is unlimited.

11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY
TECH, OTHER	JSEP through AF Office of Scientific Research (NE) 1400 Wilson Boulevard Arlington, Virginia 22209

13 ABSTRACT

The Covering Condition (enabling the application of necessary tests on all cells in an array) and the Existence of Sensitized Path Condition (enabling the propagation of the effect of a faulty cell to some boundary output) for the testability of combinational cellular arrays of rectangularly and unilaterally interconnected cells are analyzed separately in this paper. Some properties on the existence of a rectangular tessera covering some CIO-statuses are uncovered. A necessary condition, namely the full-balance condition, for the existence of a set of rectangular-tesserae covering all CIO-statuses on the CIO-table specifying a cell, suggests that the result of a simple calculation will enable one to determine the need for finding some prime tessellation with respect to a CIO-status. A procedure for finding a prime tessellation with respect to a given CIO-status is presented. It is seen, for a class of combinational cellular arrays, that some arrays can be very efficiently tested independent of the size of the array where only nonprime tessellations are to be found and the number of tests depends on the size of the testable array where a prime tessellation with respect to some CIO-status is to be found. This property is conceivably true for the general class of combination cellular arrays. Two necessary conditions for the combinational cellular array testing, which are also sufficient for the detection of the presence of a single faulty cell, are analogously carried over to the situation for the testing of sequential cellular arrays. The results regarding the testability of combinational cellular arrays are immediately extendible to fault detection and location in the first category sequential cellular arrays where the state of each memory element in every cell can be set or reset through the external control. An immediate implication is that if the logic circuit in each cell under some state can facilitate an efficient testing on a first category sequential cellular array of cells each under the state, then the array is testable; furthermore, the faultlessness of this "hardcore" enables the location of a faulty cell with some uncertainty. Some convenient assumptions enable one to establish some sufficient conditions for the efficient fault detection and location in second category sequential cellular arrays, of two dimensions as well as of one dimension, where each cell is some finite-state machine. Information losslessness (of a finite-state machine) is a sufficient, but not necessary, condition for reconstructing the cell response to some test sequence from some sequence of boundary outputs. A considerably lower upper bound on the finite order of the information losslessness of the finite-state machine, where the cardinality of the input set equals that of the output set, is conjectured. In a special case where the assumption that each cell can be initialized to its initial state at any time upon a command is not made, one is able to determine the testability.

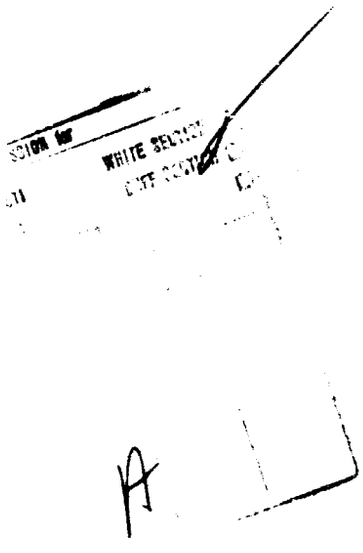
DD FORM 1473 JAN 64

UNCLASSIFIED
Security Classification

The Electronics Research Center at The University of Texas at Austin constitutes interdisciplinary laboratories in which graduate faculty members and graduate candidates from numerous academic disciplines conduct research.

Research conducted for this technical report was supported in part by the Department of Defense's JOINT SERVICES ELECTRONICS PROGRAM (U.S. Army, U.S. Navy, and the U.S. Air Force) through the Research Contract AFOSR F44620-71-C-0091. This program is monitored by the Department of Defense's JSEP Technical Advisory Committee consisting of representatives from the U.S. Army Electronics Command, U.S. Army Research Office, Office of Naval Research, and the U.S. Air Force Office of Scientific Research.

Additional support of specific projects by other Federal Agencies, Foundations, and The University of Texas at Austin is acknowledged in footnotes to the appropriate sections.



14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
FAULT DETECTION						
LOCATION OF FAULT						
SEQUENTIAL CELLULAR ARRAYS						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

GPO 886-551

FAULT DETECTION AND LOCATION IN SEQUENTIAL CELLULAR ARRAYS*

by

Chia-Hsiaing Sung and C. L. Coates
Department of Electrical Engineering

Technical Report No. 106
July 5, 1971

INFORMATION SYSTEMS RESEARCH LABORATORY

ELECTRONICS RESEARCH CENTER
THE UNIVERSITY OF TEXAS AT AUSTIN
Austin, Texas 78712

* Research sponsored in part by the Joint Services Electronics Program under Contract AFOSR F44620-71-C-0091 and NSF Grant GK-1146X.

ABSTRACT

The Covering Condition (enabling the application of necessary tests on all cells in an array) and the Existence of Sensitized Path Condition (enabling the propagation of the effect of a faulty cell to some boundary output) for the testability of combinational cellular arrays of rectangularly and unilaterally interconnected cells are analyzed separately in this paper. Some properties on the existence of a rectangular-tessera covering some CIO-statuses are uncovered. A necessary condition, namely the full-balance condition, for the existence of a set of rectangular-tesserae covering all CIO-statuses on the CIO-table specifying a cell, suggests that the result of a simple calculation will enable one to determine the need for finding some prime tessellation with respect to a CIO-status. A procedure for finding a prime tessellation with respect to a given CIO-status is presented. It is seen, for a class of combinational cellular arrays, that some arrays can be very efficiently tested independent of the size of the array where only nonprime tessellations are to be found and the number of tests depends on the size of the testable array where a prime tessellation with respect to some CIO-status is to be found. This property is conceivably true for the general class of combinational cellular arrays. Two necessary conditions for the combinational cellular array testing, which are also sufficient for the detection of the presence of a single faulty cell, are analogously carried over to the situation for the testing of sequential cellular arrays. The results regarding the testability of combinational cellular arrays are immediately extendible to fault detection and location in the first category sequential cellular arrays where the state of each memory element in every cell can be set or reset through the external control. An immediate implication is that if the logic circuit in each cell under some state can facilitate an efficient testing on a first category sequential cellular array of cells each under that state, then the array is

testable; furthermore, the faultlessness of this "hardcore" enables the location of a faulty cell with some uncertainty. Some convenient assumptions enable one to establish some sufficient conditions for the efficient fault detection and location in second category sequential cellular arrays, of two dimensions as well as of one dimension, where each cell is some finite-state machine. Information losslessness (of a finite-state machine) is a sufficient, but not necessary, condition for reconstructing the cell response to some test sequence from some sequence of boundary outputs. A considerably lower upper bound on the finite order of the information losslessness of the finite-state machine, where the cardinality of the input set equals that of the output set, is conjectured. In a special case where the assumption that each cell can be initialized to its initial state at any time upon a command is not made, one is able to determine the testability.

TABLE OF CONTENTS

Chapter I

INTRODUCTION

The essential part of the analysis of the fault detection and location in cellular arrays is to develop a procedure for determining where or not a cellular array of arbitrary size, with a given specification in terms of the external behavior of the typical cell without any implication with regard to the physical implementation of each cell, is testable. By testable it means that the presence of a single faulty cell in the cellular array can be detected by observing the boundary output sequence in responding to some boundary input sequence. An essential interest is to find the properties of the cell, in terms of its external behavior, of the cellular array of arbitrary size such that the cellular array with these properties is testable. Another interest is to find the boundary input sequence to be applied to the cellular array of arbitrary but finite size such that all cells in the cellular array can be tested completely. Only after the testability of a cellular array is affirmed, can one study additional constraints on the cellular array for the location of the single faulty cell within some Number U of cells, where U is called the uncertainty in location, or even to the location of multiple faults.

Some work has been done with regard to the fault detection [11], [12],[17],[18] and location [11], [18] in combinational cellular arrays. It is known that there are combinational cellular arrays where at least one CIO-status cannot be covered by any nonprime tessellation; but no procedure was given for finding some tessellation, specifically, some prime tessellation with respect to a given CIO-status, to cover that CIO-status. Breuer [3] has investigated the fault detection in a linear cascade of identical machines, where each machine is finite state, reduced, strongly-connected, and can be reset to its initial state at any time upon a command.

In this paper, a cellular array is a uniformly, rectangularly, and unilaterally interconnected array of identical cells. This does not necessarily constrain one from extending the study in this paper to the situation where diagonal interconnections are allowed. Throughout this paper, the assumption of the presence of single faulty cell in a cellular array will be maintained, the fault in a cell in a cellular array is assumed to be nontransient and may affect the cell output in any arbitrary way, and all logical functions are understood to be completely-specified.

Some properties on the combinational cellular array testing are developed and a procedure for finding some prime tessellation with respect to a given CIO-status is presented in Chapter II. The results for the combinational cellular array testing can be easily extended to the fault detection and location in second category sequential cellular arrays (both one-dimensional and two-dimensional) are investigated in Chapter IV. Some convenient assumptions enable one to establish some sufficient conditions for the second category sequential cellular array to be testable. In a special case where the assumption that each cell can be initialized to its initial state at any time upon a command is not made, some properties are found to be necessary and sufficient for the cellular array of that case to be testable.

Chapter II

FAULT DETECTION IN COMBINATIONAL CELLULAR ARRAYS

A. Introduction

Since the specification of the typical cell in the combinational cellular array is given in terms of the external behavior of the cell without any implication of its physical implementation (at gate level) and the fault within a cell may affect the cell output in any arbitrary way, it will be assumed that all input combinations are to be applied to a cell to test the cell completely in this chapter. At the end of this chapter, remarks will be made in connection with the situation where the physical implementation of the cell in the combinational cellular array is known. Within the scope of this chapter, an array will be understood to be a combinational cellular array unless otherwise stated.

Two necessary conditions for the array testing will be analyzed separately for two-dimensional arrays. Some properties of the cell in the array are found to be sufficient for obtaining efficient boundary input combinations to test an array completely independent of the array size. There are cases where at least one input combination is not applicable to the typical cell in the array. There are cases where some procedures are to be followed such that some boundary input combinations can be determined which enable all cells in the array to have some input combination applied. The implication of a CIO-statuses-compatible mapping on the array is that with the application of some boundary input combination to the array, each cell in the array has some input combination applied as described by the CIO-statuses-compatible mapping on the array.

B. General Conditions for Testability

Under the assumption that all input combinations are necessary to test a cell, these are two, well-known, necessary conditions that must be satisfied for array testing for the detection of the presence of a faulty cell in an array. They are the Covering Condition and Sensitized Path Condition as follows.

> Covering Condition: Every input combination must be applicable to every cell in the array.
>
> Sensitized Path Condition: For each input combination to a cell and each possible cell output change due to a fault, there must exist at least one sensitized path[1] from that cell to one of boundary outputs.

These two conditions together are sufficient for the Detection of the presence of single faulty cell in an array.

C. Two-Dimensional Arrays

Let S_I be the set of all positive integers. Let each element in the set $S_I \times S_I$ identify the location of some cell in the doubly-infinite array of cells. Thus, C_{ij}, where $i,j \in S_I$, denotes the cell at the location (i,j) in an array of cells. The general configuration of a two-dimensional array of finite size is shown in Figure 1. For a cell, the finite horizontal input set, the finite vertical input set, the finite horizontal output set, and the

[1] Consider cells in an array as nodes, boundary input terminals as sources, boundary output terminals as sinks, interconnections and the connections to boundary terminals as edges in an oriented graph, then a sensitized path from a cell, denoted by node n_f, to a boundary output terminal, denoted by sink n_k, is a path from n_f to n_k such that when the leading edge of the path (coming out from n_f) is perturbed from its nominal value, the remaining edges on the path are perturbed from their respective nominal values.

finite vertical output set are denoted by $X, Z, \hat{X}$, and $\hat{Z}$, respectively. x, z, $\hat{x}$, and $\hat{z}$ are arbitrary elements of $X, Z, \hat{X}$, and $\hat{Z}$, respectively. $\hat{X} = X$ and $\hat{Z} = Z$ are implicit here.

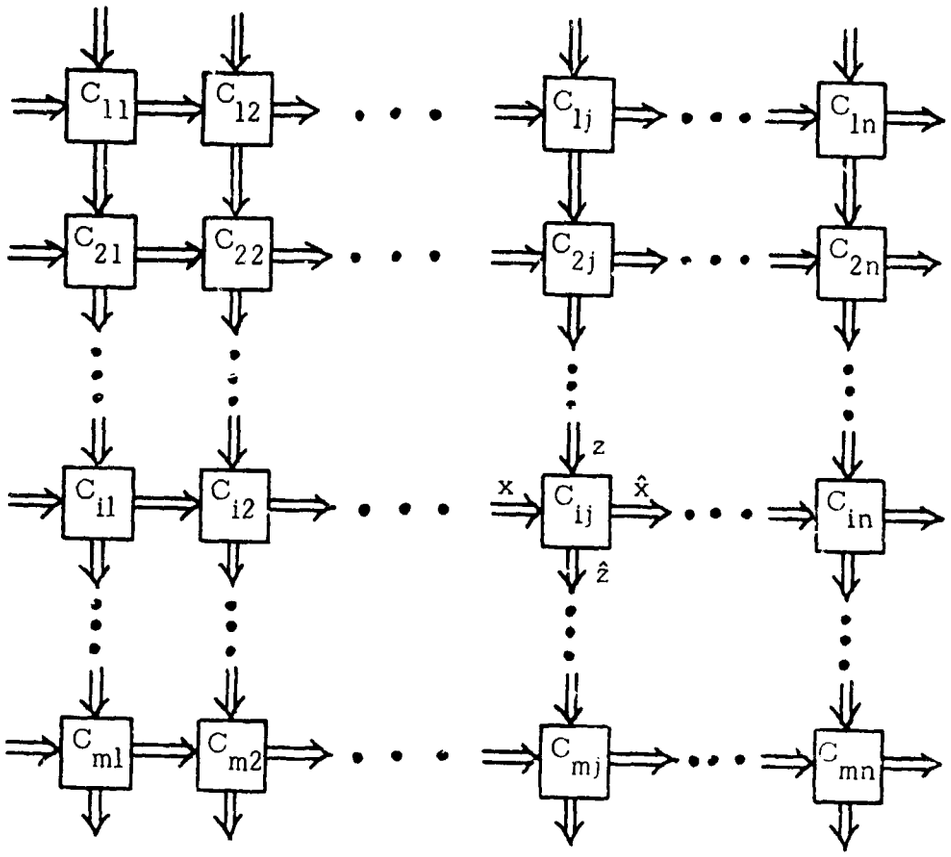


Figure 1. A Two-Dimensional Array of Size mxn.

Let the input-output behavior of the cell in an array be described in terms of a mapping $F: X \times Z \rightarrow \hat{X} \times \hat{Z}$. This mapping can be specified in a form of the cell-input-output-table, abbreviated as CIO-table, listing all possible cell input combinations and the corresponding cell outputs. A CIO-status is an ordered pair of a cell input combination and its corresponding cell output, in other words, a CIO-status is an element in $\{(x, z, \hat{x}, \hat{z}) \mid x \in X, z \in Z, \hat{x} \in \hat{X}, \hat{z} \in \hat{Z}$, and $(\hat{x}, \hat{z}) = F(x, z)\}$.

The Covering Condition and the Sensitized Path Condition for the testing of two-dimensional arrays are analyzed separately.

1. Covering Each Cell in an Array with Every Possible Input Combination

The problem of determing the constraints on the cell external behavior under which every possible input combination can be applied to any typical cell in a two-dimensional array is not easy because a cell input combination to a typical cell in the array is provided by an entire subarray of cells neither to the right nor below that cell with that cell excluded where the matching of the cell input and the cell output on adjacent cells must be observed.

By a CIO-status on a given cell is compatible with the CIO-statuses on all adjacent cells it means that the horizontal input part of the CIO-status on that cell is the same as the horizontal output part of the CIO-status on the cell to the left of that cell and the vertical input part of the CIO-status on that cell is the same as the vertical output part of the CIO-status on the cell above that cell.

Given the input-output behavior of the cell in an array of size MxN in terms of the mapping $F:XxZ\rightarrow\hat{X}x\hat{Z}$, define a CIO-statuses-compatible mapping on the array to be a mapping $G:C\rightarrow S_F$ where
$C = \{C_{ij} \mid i\in\{1,2,3,\ldots,M\}$ and $j\in\{1,2,3,\ldots,N\}\}$,
$S_F = \{(x,z,\hat{x},\hat{z}) \mid x\in X, z\in Z, \hat{x}\in\hat{X}, \hat{z}\in\hat{Z}$, and $(\hat{x},\hat{z}) = F(x,z)\}$, and for every $C_{ij}\in C$, the CIO-status on C_{ij}, i.e. $G(C_{ij})$, is compatible with the CIO-statuses on all adjacent cells in C. A CIO-statuses-compatible mapping on the doubly-infinite array, i.e. the array of size MxN where $M\rightarrow\infty$ and $N\rightarrow\infty$, is said to be a tessellation.

The following two theorems are obvious.

Theorem 1: The Covering Condition for testing an array of size MxN is satisfied if and only if there exists a nonempty set S of CIO-statuses-compatible mappings on the array such that each CIO-status on the CIO-table specifying the cell occurs on C_{ij} in at least one mapping in the set S

for $\forall i \in \{1,2,3,\ldots,M\}$ and $\forall j \in \{1,2,3,\ldots,N\}$.

Theorem 2: The Covering Condition for testing an array of arbitrary size, where the CIO-table specifying the cell is given, is satisfied if there exists a nonempty set S_t of tessellations such that each CIO-status on the CIO-table occurs on C_{ij} in at least one tessellation in the set S_t for $\forall i \in S_I$ and $\forall j \in S_I$.

In recalling that the existence of a nonempty set of tessellations such that each CIO-status on the CIO-table occurs on C_{ij} in some tessellation in the set $\forall i \in S_I$ and $\forall j \in S_I$ is also necessary for satisfying the Covering Condition for the double-infinite array, one can conceive that this property is also necessary for satisfying the Covering Condition for very large (in both dimensions) finite arrays.

The tessellation problem here is actually a special case of the Domino Problem. A domino is a square plate with edges colored, one color on each edge but different edges may have the same color. The type of a domino is identified by the colors on its edges. Hence, a domino can be thought of as a quadruple (a,b,c,d) where a,b,c and d represents the color on 1st, 2nd, 3rd, and 4th edge, respectively. The domino game is to take a finite set of domino types with infinitely many pieces of every type and try to cover the infinite plane with dominoes of these types such that any two adjoining edges have the same color and none of the dominoes is rotated or reflected in the covering. A finite set of domino types is said to be solvable if and only if the infinite plane can be so covered with dominoes of these types. The Domino Problem is to find an algorithm to decide, for any given finite set of domino types, whether it is solvable. Kahr et al. [10], Wang [19], and Berger [2] have proven that the Domino Problem is undecidable meaning that there does not exist a general algorithm which, given the specifications of an arbitrary (finite) domino set, will decide whether or not the set is solvable. Tammaru [18] carried the same conclusion of the undecidability of the Domino Problem over to the tessellation problem for two-dimensional arrays. In essence, Seth [17] conjectured that in general the condition on the existence of at least one tessellation for

every CIO-status on the CIO-table may not be a solvable problem. Notice that the Domino Problem deals with the class of all domino sets. In identifying the special features of the tessellation problem with respect to the Domino Problem, Kautz [11] noted that whether or not the special features of the tessellation problem permit the existence of a procedure for solving the tessellation problem remains to be determined.

a. Existence of Rectangular-Tesserae and Tessellations

One can also consider a CIO-status $\vec{q}$ as a quadruple (a,b,c,d), where a,b,c, and d are x-component, z-component, $\hat{x}$-component, and $\hat{z}$-component of $\vec{q}$, respectively. A superscript on a CIO-status can be used to designate its component, for instance, $\vec{q}^{x}$ is the x-component of $\vec{q}$. In the remainder of this chapter, these notations on the CIO-status will be adopted.

For some finite positive integer n, an n-tuple of CIO-statuses $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$, where $\vec{q}_i \in S_F = \{(x,z,\hat{x},\hat{z}) \mid x\in X,\ z\in Z,\ \hat{x}\in\hat{X},\ \hat{z}\in\hat{Z}$, and $(\hat{x},\hat{z}) = F(x,z)\}$ for $\forall i\in\{1,2,3,\ldots,n\}$, is said to be a row-chain if and only if $\vec{q}_1^{x} = \vec{q}_n^{\hat{x}}$ and $\vec{q}_i^{x} = \vec{q}_{i-1}^{\hat{x}}$ for $\forall i\in\{2,3,4,\ldots,n\}$. Similarly, for some finite positive integer m, an m-tuple of CIO-statuses $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_m)$, where $\vec{q}_i \in S_F$ for $\forall i\in\{1,2,3,\ldots,m\}$, is said to be a column-chain if and only if $\vec{q}_1^{z} = \vec{q}_m^{\hat{z}}$ and $\vec{q}_i^{z} = \vec{q}_{i-1}^{\hat{z}}$ for $\forall i\in\{2,3,4,\ldots,m\}$. For some finite positive integers m and n, an m·n-tuple of CIO-statuses $((\vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, q_{1n}), (\vec{q}_{21}, \vec{q}_{22}, \vec{q}_{23}, \ldots, \vec{q}_{2n}), \ldots, (\vec{q}_{m1}, \vec{q}_{m2}, \vec{q}_{m3}, \ldots, \vec{q}_{mn}))$, where $\vec{q}_{ij} \in S_F$ for $\forall i\in\{1,2,3,\ldots,m\}$ and $\forall j\in\{1,2,3,\ldots,n\}$ is said to be a rectangular-tessera if and only if $(\vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, \vec{q}_{1n})$ is a row-chain for $\forall i\in\{1,2,3, \ldots,m\}$ and $(\vec{q}_{1j}, \vec{q}_{2j}, \vec{q}_{3j}, \ldots, \vec{q}_{mj})$ is a column-chain for $\forall j\in\{1,2,3,\ldots,n\}$. This rectangular-tessera is said to be of order m x n.

Another way of viewing a rectanular-tessera (pl. rectangular-tesserae) is as follows. For some finite positive integers m and n, an m · n-tuple of CIO-statuses $((\vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, \vec{q}_{1n}), (\vec{q}_{21}, \vec{q}_{22}, \vec{q}_{23}, \ldots, \vec{q}_{2n}), \ldots\ldots,$

$(\vec{q}_{m1}, \vec{q}_{m2}, \vec{q}_{m3}, \ldots, \vec{q}_{mn}))$, where $\vec{q}_{ij} \in S_F$ for $\forall i \in \{1,2,3, \ldots, m\}$ and $\forall j \in \{1,2,3,\ldots,n\}$ is said to be a rectangular-tessera if and only if there exists a mapping $H: D \rightarrow S_F$, where $D = \{C_{rk}, C_{r(k+1)}, C_{r(k+2)}, \ldots, C_{r(k+n-1)}, C_{(r+1)k}, C_{(r+1)(k+1)}, C_{(r+1)(k+2)}, \ldots, C_{(r+1)(k+n-1)}, \ldots, C_{(r+m-1)k}, C_{(r+m-1)(k+1)}, C_{(r+m-1)(k+2)}, \ldots, C_{(r+m-1)(k+n-1)}\}$ for some $r,k \in S_I$, satisfying (1) $H(C_{ij}) = \vec{q}_{(i-r+1)(j-k+1)}$ for $\forall i \in \{r, r+1, r+2, \ldots, r+m-1\}$ and $\forall j \in \{k, k+1, k+2, \ldots, k+n-1\}$, (2) for every $C_{ij} \in D$, the CIO-status on C_{ij} is compatible with the CIO-statuses on all adjacent cells in D, and (3) $H(C_{i(k+n-1)})^{\hat{x}} = H(C_{ik})^{x}$ and $H(C_{(r+m-1)j})^{\hat{z}} = H(C_{rj})^{z}$ for $\forall i \in \{r, r+1, r+2, \ldots, r+m-1\}$ and $\forall j \in \{k, k+1, k+2, \ldots, k+n-1\}$.

A rectangular-tessera T is said to cover all CIO-statuses in a set S if every CIO-status in the set S appears in the rectangular-tessera T.

Theorem 3: Given an n-tuple of CIO-statuses $(\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n)$, where $\vec{q}_i \in S_F = \{(x,z,\hat{x},\hat{z}) \mid x \in X, z \in Z, \hat{x} \in \hat{X}, \hat{z} \in \hat{Z}$, and $(\hat{x},\hat{z}) = F(x,z)\}$ for $\forall i \in \{1,2,\ldots,n\}$ for some finite positive integer n, if (1) $(\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n)$ is both a row-chain and a column-chain, (2) $(\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n)$ is a row-chain and $(\vec{q}_n, \vec{q}_{n-1}, \ldots, \vec{q}_2, \vec{q}_1)$ is a column-chain, or (3) $(\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n)$ is a column-chain and $(\vec{q}_n, \vec{q}_{n-1}, \ldots, \vec{q}_2, \vec{q}_1)$ is a row-chain, then there exists a rectangular-tessera of order $n \times n$ covering all CIO-statuses $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}$, and $\vec{q}_n$.

Proof: Suppose the hypothesis holds.

(1) If $(\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n)$ is both a row-chain and a column-chain, then $(\vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n, \vec{q}_1)$ is both a row-chain and a column-chain, $(\vec{q}_3, \ldots, \vec{q}_{n-1}, \vec{q}_n, \vec{q}_1, \vec{q}_2)$ is both a row-chain and a column-chain, ..., and $(\vec{q}_n, \vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1})$ is both a row-chain and a column-chain. Thus, $((\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n), (\vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n, \vec{q}_1), (\vec{q}_3, \ldots, \vec{q}_n, \vec{q}_1, \vec{q}_2), \ldots,$ $(\vec{q}_n, \vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}))$ is a rectangular-tessera of order $n \times n$ covering all $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}$, and $\vec{q}_n$.

(2) If $(\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n)$ is a row-chain and $(\vec{q}_n, \vec{q}_{n-1}, \ldots, \vec{q}_2, \vec{q}_1)$ is a column-chain, then $(\vec{q}_n, \vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-2}, \vec{q}_{n-1})$ is a row-chain, ... $(\vec{q}_3, \vec{q}_4, \ldots, \vec{q}_n, \vec{q}_1, \vec{q}_2)$ is a row-chain, $(\vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n, \vec{q}_1)$ is a row-chain,

$(\vec{q}_1, \vec{q}_n, \vec{q}_{n-1}, \ldots, \vec{q}_3, \vec{q}_2)$ is a column-chain, $(\vec{q}_2, \vec{q}_1, \vec{q}_n, \ldots, \vec{q}_4, \vec{q}_3)$ is a column-chain, ..., and $(\vec{q}_{n-1}, \vec{q}_{n-2}, \ldots, \vec{q}_2, \vec{q}_1, \vec{q}_n)$ is a column-chain. Thus, $((\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n), (\vec{q}_n, \vec{q}_1, \ldots, \vec{q}_{n-2}, \vec{q}_{n-1}), \ldots, (\vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n, \vec{q}_1))$ is a rectangular-tessera of order nxn covering all $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}$, and $\vec{q}_n$.

(3) If $(\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}, \vec{q}_n)$ is a column-chain and $(\vec{q}_n, \vec{q}_{n-1}, \ldots, \vec{q}_2, \vec{q}_1)$ is a row-chain, then by symmetrical arguments to that in (2) $((\vec{q}_1, \vec{q}_n, \vec{q}_{n-1}, \ldots, \vec{q}_3, \vec{q}_2), (\vec{q}_2, \vec{q}_1, \vec{q}_n, \ldots, \vec{q}_4, \vec{q}_3), \ldots, (\vec{q}_{n-1}, \vec{q}_{n-2}, \ldots, \vec{q}_1, \vec{q}_n), (\vec{q}_n, \vec{q}_{n-1}, \ldots, \vec{q}_2, \vec{q}_1))$ is a rectangular-tessera of order nxn covering all $\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_{n-1}$, and $\vec{q}_n$. ◢

Given that $T = ((\vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, \vec{q}_{1n}), (\vec{q}_{21}, \vec{q}_{22}, \vec{q}_{23}, \ldots, \vec{q}_{2n}), \ldots, (\vec{q}_{m1}, \vec{q}_{m2}, \vec{q}_{m3}, \ldots, \vec{q}_{mn}))$, where $\vec{q}_{ij} \in S_\Gamma$ for $\forall i \in \{1,2,3,\ldots,m\}$ and $\forall j \in \{1,2,3,\ldots,n\}$ for some finite positive integers m and n, is a rectangular-tessera, then, $((\vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, \vec{q}_{1n}, \vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, \vec{q}_{1n}), \vec{q}_{21}, \vec{q}_{22}, \vec{q}_{23}, \ldots, \vec{q}_{2n}, \vec{q}_{21}, \vec{q}_{22}, \vec{q}_{23}, \ldots, \vec{q}_{2n}), \ldots, (\vec{q}_{m1}, \vec{q}_{m2}, \vec{q}_{m3}, \ldots, \vec{q}_{mn}, \vec{q}_{m1}, \vec{q}_{m2}, \vec{q}_{m3}, \ldots, \vec{q}_{mn}))$ is also a rectangular-tessera. The latter is said to be obtained from the former by <u>iterating the pattern</u> (of the former) <u>once in the horizontal dimension</u>. Similarly, the rectangular-tessera $((\vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, \vec{q}_{1n}), (\vec{q}_{21}, \vec{q}_{22}, \vec{q}_{23}, \ldots, \vec{q}_{2n}), \ldots, (\vec{q}_{m1}, \vec{q}_{m2}, \vec{q}_{m3}, \ldots, \vec{q}_{mn}), (\vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, \vec{q}_{1n}), (\vec{q}_{21}, \vec{q}_{22}, \vec{q}_{23}, \ldots, \vec{q}_{2n}), \ldots, (\vec{q}_{m1}, \vec{q}_{m2}, \vec{q}_{m3}, \ldots, \vec{q}_{mn}))$ is said to be obtained from the rectangular-tessera T by <u>iterating the pattern</u> (of the rectangular-tessera T) <u>once in the vertical dimension</u>.

Given any two rectangular-tesserae T_1 and T_2, if T_2 can be obtained from T_1 by iterating the pattern of T_1 at least once in the horizontal dimension or the vertical dimension, then T_2 is said to be <u>implied</u> by T_1 or T_1 is said to <u>imply</u> T_2. A rectangular-tessera T is said to be <u>minimal</u> if and only if T is implied by no rectangular-tessera. In the remainder of this paper, all rectangular-tesserae are understood to be minimal.

Let k be a fixed positive integer and a and b be any two integers. We define <u>$a \equiv b \bmod k$</u> if k divides (a-b). For any integer c, we define <u>$c_{\bmod k}$</u> to be the positive integer d where $d \equiv c \bmod k$ and $0 < d \leq k$.

Let T_q and T_p be two rectangular-tesserae of the same order, say $T_q = ((\vec{q}_{11}, \vec{q}_{12}, \vec{q}_{13}, \ldots, \vec{q}_{1n}), (\vec{q}_{21}, \vec{q}_{22}, \vec{q}_{23}, \ldots, \vec{q}_{2n}), \ldots, (\vec{q}_{m1}, \vec{q}_{m2}, \vec{q}_{m3}, \ldots, \vec{q}_{mn}))$ and $T_p = ((\vec{p}_{11}, \vec{p}_{12}, \vec{p}_{13}, \ldots, \vec{p}_{1n}), (\vec{p}_{21}, \vec{p}_{22}, \vec{p}_{23}, \ldots, \vec{p}_{2n}), \ldots, (\vec{p}_{m1}, \vec{p}_{m2}, \vec{p}_{m3}, \ldots, \vec{p}_{mn}))$ are two rectangular-tesserae of finite order mxn. T_p is said to be obtainable from T_q with horizontal-wise rotation of k slots, for some integer k, if $\vec{p}_{ij} = \vec{q}_{i(j-1)_{\text{mod } n}}$ for $\forall i \in \{1,2,3,\ldots,m\}$ and $\forall j \in \{1,2,3,\ldots,n\}$. Similarly, T_p is said to be obtainable from T_q with vertical-wise rotation of k slots, for some integer k, if $\vec{p}_{ij} = \vec{q}_{(i-k)_{\text{mod } m}j}$ for $\forall i \in \{1,2,3,\ldots,m\}$ and $\forall j \in \{1,2,3,\ldots,n\}$.

Let S be a set of all rectangular-tesserae of same order. Define the binary relation R on the set S as follows: for all $s_1, s_2 \in S$, $s_1 R s_2$ if and only if s_2 is obtainable from s_1 with horizontal-noise rotation of k slots and with vertical-wise rotation of r slots for some integers k and r. One can verify that the binary relation R on the set S is reflective, transitive, and symmetric. Hence, the binary relation R is an equivalence relation on the set S. For any two rectangular-tesserae T_1 and T_2 of same order, T_1 is said to be equivalent to T_2 or vice versa if and only if $T_1 R T_2$.

Given a rectangular-tessera of order mxn, one can mark the sub-array of cells $C_{11}, C_{12}, \ldots, C_{1n}, C_{21}, C_{22}, \ldots, C_{2n}, \ldots, C_{m1}, C_{m2}, \ldots,$ and C_{mn} in the doubly-infinite array with CIO-statuses according to the pattern of the given rectangular-tessera and then mark the remaining cells in the doubly-infinite array with CIO-statuses according to the resultant pattern of iterating the pattern of the given rectangular-tessera infinitely many times in the horizontal dimension and then iterating the intermediate result infinitely many times in the vertical dimension to obtain a tessellation. The given rectangular-tessera is said to induce the tessellation. A tessellation which can be obtained from the resultant pattern of such an iteration on a pattern of some rectangular-tessera is said to be nonprime. A tessellation which is not nonprime is said to be prime.

For any two nonprime tessellations $\mathcal{T}_1$ and $\mathcal{T}_2$ which are induced by the rectangular-tesserae T_1 and T_2, respectively, if T_1 and T_2 are of same order and equivalent, then $\mathcal{T}_1$ is said to be equivalent to $\mathcal{T}_2$ or $\mathcal{T}_1$ and $\mathcal{T}_2$ are said to be equivalent.

Given a rectangular-tessera T, then, the equivalence class of rectangular-tesserae represented by T is $cl(T) = \{T_i \mid T_i$ is a rectangular-tessera of the order same as T and $_T R_{T_i}\}$.

Theorem 4: Given an array of arbitrary size where the CIO-table specifying the cell is known, if there exists a rectangular-tessera T covering all CIO-statuses in a subset S of the set of all CIO-statuses on the CIO-table, then each CIO-status in S can occur on C_{ij} in at least one tessellation in the set $S_{\mathcal{J}}$ of all nonprime tessellations induced by all rectangular-tesserae in cl(T) for $\forall i \in S_I$ and $\forall j \in S_I$. Moreover, the number of steps required to enable all cells in the array to experience all CIO-statuses in S can be not greater than $|cl(T)|$, where $|cl(T)$ denotes the number of elements in the set cl(T) and is bounded from above by m·n where mxn is the order of T.

Proof: Suppose the hypothesis holds. The systemic way of generating cl(T) from T, where the order of T is mxn, is obtaining an element in cl(T) from T with horizontal-wise rotation of k slots and with vertical-wise rotation of r slots first for k = 0 and r = 0, second for k = 0 and r = 1, third for k = 0 and r = 2,..., for k = 1 and r = n-1,..., for k = m-1 and r = 0, for k = m-1 and r = 1,..., and finally for k = m-1 and r = n-1. Thus, $|cl(T)| \leq m \cdot n$. Let $S_{\mathcal{J}}$ be the set of all nonprime tessellations induced by all rectangular-tesserae in cl(T). It is clear that $|S_{\mathcal{J}}| = |cl(T)|$. It is also clear that each CIO-status in S can occur on C_{ij} in at least one nonprime tessellation in $S_{\mathcal{J}}$ for $\forall i \in \{1,2,3,\ldots,m\}$ and $\forall j \in \{1,2,3,\ldots,n\}$. Therefore, each CIO-status in S can occur on C_{ij} in at least one nonprime tessellation in $S_{\mathcal{J}}$ for $\forall i \in S_I$ and $\forall j \in S_I$. Each nonprime tessellation in $S_{\mathcal{J}}$ facilitates a step to enable each cell in the array to experience some CIO-status in S. All nonprime tessellations in $S_{\mathcal{J}}$ provide sufficient steps to enable all cells in the array to experience all CIO-statuses in S. Hence, the number of steps required to enable all cells in the array to experience all CIO-statuses in S needs not be greater than $|S_{\mathcal{J}}|$. ∎

The following theorem is then obvious.

Theorem 5: Given an array of arbitrary size where the CIO-table specifying the cell is known, if there exists a rectangular-tessera T covering all CIO-statuses in a subset $S = \{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_t\}$ of the set of all CIO-statuses on the CIO-table, then the number of steps required to enable all cells in the array to experience all CIO-statuses in S in utilizing the information provided by the set of all nonprime tessellations induced by all rectangular-tesserae in cl(T) is bounded below by $\left\lceil \frac{m \cdot n}{\min.(\eta_1, \eta_2, \ldots, \eta_t)} \right\rceil$, where mxn is the order of T, η_i is the number of appearances of $\vec{q}_i$ in T for $\forall i \in \{1, 2, \ldots, t\}$, $\min.(\eta_1, \eta_2, \ldots, \eta_t)$ is the smallest one of $\eta_1, \eta_2, \ldots, \eta_t$ and $\lceil r \rceil$ is the smallest integer not less than r.

Given an n-tuple of CIO-statuses $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$, where for every $i \in \{1, 2, 3, \ldots, n\}$, $\vec{q}_i \in S_q \{(x, z, \hat{x}, \hat{z}) \mid x \in X, z \in Z, \hat{x} \in \hat{X}, \hat{z} \in \hat{Z}$, and $(\hat{x}, \hat{z}) = F(x, z)\}$, $X = \hat{X} = \{x_1, x_2, x_3, \ldots, x_j\}$, and $Z = \hat{Z} = \{z_1, z_2, z_3, \ldots, z_k\}$. Then, $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$ is said to be in x/$\hat{x}$-balance if and only if the frequency of occurrences of x_i at the x-input component over $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$ equals that of x_i at the $\hat{x}$-output component over $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$ for all $i \in \{1, 2, 3, \ldots, j\}$. $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$ is said to be in z/$\hat{z}$-balance if and only if the frequency of occurrences of z_i at the z-input component over $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$ equals that of z_i at the $\hat{z}$-output component over $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$ for all $i \in \{1, 2, 3, \ldots, k\}$. $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$ is said to be in full-balance if and only if it is both in x/$\hat{x}$-balance and in z/$\hat{z}$-balance.

Given an n-tuple of CIO-statuses $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$, for some positive integer m such that $m < n$, an m-tuple of CIO-statuses $(\vec{p}_1, \vec{p}_2, \vec{p}_3, \ldots, \vec{p}_m)$ is said to be a sub-n-tuple of $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n)$ if $\vec{p}_1$ is identical to exactly one of $\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_n$, $\vec{p}_2$ is identical to exactly one of the remainder of $\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots$, and $\vec{q}_n$ after the one identified with $\vec{p}_1$ is deleted, and $\vec{p}_1$ is identical to exactly one of the remainder of

$\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots,$ and $\vec{q}_n$ after those identified with $\vec{p}_1, \vec{p}_2, \ldots,$ and $\vec{p}_{i-1}$ are deleted for every $i \in \{3,4,\ldots,m\}$. For an example, $(\vec{q}_1, \vec{q}_1, \vec{q}_2)$ is a sub-5-tuple $(\vec{q}_1, \vec{q}_2, \vec{q}_1, \vec{q}_3, \vec{q}_2)$.

Theorem 6 (Property 1): Given a set of 3 (distinct) CIO-statuses $S = \{\vec{q}_1, \vec{q}_2, \vec{q}_3\}$, if $(\vec{q}_1, \vec{q}_1, \vec{q}_2, \vec{q}_3)$ is in full-balance but no sub-4-tuple of the 4-tuple $(\vec{q}_1, \vec{q}_1, \vec{q}_2, \vec{q}_3)$ is in full-balance, then there exists a rectangular-tessera covering all CIO-statuses in the set S.

Proof: Suppose the hypothesis holds and $\vec{q}_1 = (a_1, b_1, c_1, d_1)$, $\vec{q}_2 = (a_2, b_2, c_2, d_2)$, and $\vec{q}_3 = (a_3, b_3, c_3, d_3)$. Then, two possible cases are considered.

Case 1, $a_1 \neq c_1$: Since $((a_1, b_1, c_1, d_1), (a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2), (a_3, b_3, c_3, d_3))$ is in full-balance, $a_1 \neq c_1$ implies that $c_2 = c_3 = a_1$ and $a_2 = a_3 = c_1$. Thus, each of $((a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2))$ and $((a_1, b_1, c_1, d_1), (a_3, b_3, c_3, d_3))$ is a row-chain.

Subcase 1a, $b_1 \neq d_1$: Under this subcase, $d_2 = d_3 = b_1$ and $b_2 = b_3 = d_1$, then $(a_2, b_2, c_2, d_2))$ and (a_3, b_3, c_3, d_3) are not distinct, a contradiction to the hypothesis. Hence, subcase 1a is impossible.

Subcase 1b, $b_1 = d_1$: That $((a_1, b_1, c_1, d_1), (a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2), (a_3, b_3, c_3, d_3))$ is in full-balance and $b_1 = d_1$ implies that either (a) $b_2 = d_2$ and $b_3 = d_3$ or (b) $b_2 = d_3$ and $b_3 = d_2$. Under (a), then each of $((a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2))$ and $((a_1, b_1, c_1, d_1), (a_3, b_3, c_3, d_3))$ is in full-balance, a contradiction to the hypothesis. Under (b), then each of $((a_1, b_1, c_1, d_1), (a_1, b_1, c_1, d_1))$ and $((a_2, b_2, c_2, d_2), (a_3, b_3, c_3, d_3))$ is a column-chain. Hence, there exists a rectangular-tessera $(((a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2)), ((a_1, b_1, c_1, d_1), (a_3, b_3, c_3, d_3)))$ of order 2x2 covering all CIO-statuses in the set S.

Case 2, $a_1 = c_1$: If $a_1 = c_1$, then $((a_1, b_1, c_1, d_1), (a_1, b_1, c_1, d_1))$ is a row-chain and $b_1 \neq d_1$. Thus, $d_2 = d_3 = b_1$ and $b_2 = b_3 = d_1$. Hence, each of $((a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2))$ and $((a_1, b_1, c_1, d_1), (a_3, b_3, c_3, d_3))$ is a column-chain.

Subcase 2a, $a_2 = c_2$: Under this subcase, $((a_1,b_1,c_1,d_1), (a_2,b_2,c_2,d_2))$ is in full-balance, a contradiction to the hypothesis. Hence, subcase 2a is impossible.

Subcase 2b, $a_2 \neq c_2$: Under this subcase, that $((a_1,b_1,c_1,d_1), (a_1,b_1,c_1,d_1), (a_2,b_2,c_2,d_2), (a_3,b_3,c_3,d_3))$ is in full-balance forces that $c_3 = a_2$ and $a_3 = c_2$. Then, $((a_2,b_2,c_2,d_2), (a_3,b_3,c_3,d_3))$ is a row-chain. Hence, there exists a rectangular-tessera $(((a_1,b_1,c_1,d_1), (a_1,b_1,c_1,d_1)), ((a_2,b_2,c_2,d_2), (a_3,b_3,c_3,d_3)))$ of order 2x2 covering all CIO-statuses in the set S.

Theorem 7 (Property 2)[2]: If the cell in an array of arbitrary size is chosen such that the mapping $F:X\times Z \rightarrow \hat{X}\times\hat{Z}$ describing the external behavior of the cell is one-to-one, then, there exists a set of rectangular-tesserae covering all CIO-statuses in the set S_F of CIO-statuses describing the cell, moreover, the number of steps required to enable all cells in the array to experience all CIO-statuses in S_F is $|X| \cdot |Z|$.

Proof: Suppose the hypothesis holds. Let $\mathcal{N} = \{(x,z) \mid x\in X \text{ and } z\in Z\}$. Let each member in the set $\mathcal{N}$ identifies a node in an oriented graph $\mathcal{G} = (\mathcal{N}, \rho)$, where ρ is a set of edges obtained as following: an edge e from the node $(x_1,z_1)\in\mathcal{N}$ to the node $(x_2,z_2)\in\mathcal{N}$ exists and is in the set ρ if and only if $F(x_1,z_1) = (x_2,z_2)$. Since $F:X\times Z\rightarrow\hat{X}\times\hat{Z}$ is one-to-one (also onto because $X\times Z = \hat{X}\times\hat{Z}$ and is finite), each node in the graph $\mathcal{G}$ has precisely one incoming edge and one outgoing edge. Then the graph $\mathcal{G}$ is actually a graph consisting of separated subgraphs where each subgraph is a loop. (A loop can be a self-loop or a loop containing two or more nodes.) Let $L = \{L_1, L_2, \ldots, L_k\}$ be a set of all loops in the graph $\mathcal{G}$ and $\ell(L_i)$ be the number of nodes in the loop L_i for $\forall i\in\{1,2,\ldots,k\}$. One can claim that there

---

[2] A special case in Kautz [11] (Theorem 12). The "proof" given by Kautz does not appear to be adequate.

exists a set of rectangular-tesserae $\{T_1, T_2, \ldots, T_k\}$, where T_i is a rectangular-tessera obtained from the information revealed by the loop L_i and is of order $\zeta(L_i) \times \zeta(L_i)$ for $\forall i \in \{1,2,\ldots,k\}$, covering all CIO-statuses in the set S_F. Indeed, if a loop $L_i \in L$ is a self-loop containing a node $(x_i, z_i) \in \mathcal{N}$, then, clearly (x_i, z_i, x_i, z_i) is a CIO-status in the set S_F, thus, there exists a rectangular-tessera, say T_i, consisting of the CIO-statuses (x_i, z_i, x_i, z_i). Suppose, in general, $L_i \in L$ is a loop containing $\zeta(L_i)$ nodes $(x_1, z_1), (x_2, z_2), (x_3, x_3), \ldots,$ and $(x_{\zeta(L_i)}, z_{\zeta(L_i)})$, where $F(x_1, z_1) = (x_2, z_2)$, $F(x_2, z_2) = (x_3, z_3), \ldots,$ and $F(x_{\zeta(L_i)}, z_{\zeta(L_i)}) = (x_1, z_1)$. Let $\vec{q}_1 = (x_1, z_1, x_2, z_2)$, $\vec{q}_2 = (x_2, z_2, x_3, z_3)$, $\vec{q}_3 = (x_3, z_3, x_4, z_4), \ldots,$ and $\vec{q}_{\zeta(L_i)} = (x_{\zeta(L_i)}, z_{\zeta(L_i)}, x_1, z_1)$. Then, noticed that $\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_{\zeta(L_i)} \in S_F$ also the $\zeta(L_i)$-tuple of CIO-statuses $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots, \vec{q}_{\zeta(L_i)})$ is both a row-chain and a column-chain. Then, by Theorem 3, there exists a rectangular-tessera, say T_i, of order $\zeta(L_i) \times \zeta(L_i)$ covering all the CIO-statuses $\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots,$ and $\vec{q}_{\zeta(L_i)}$. Furthermore, the number of steps required to enable all cells in the array to experience all distinct CIO-statuses $\vec{q}_1, \vec{q}_2, \vec{q}_3, \ldots,$ and $\vec{q}_{\zeta(L_i)}$ is $\zeta(L_i)$ by Theorems 4 and 5. In fact, there is an obvious one-to-one correspondence between the set $\mathcal{N}$ of nodes in the graph $\mathcal{G}$ and the set S_F. Each loop in the graph $\mathcal{G}$ identifies a rectangular-tessera in the set $\{T_1, T_2, \ldots, T_k\}$. Thus, there exists a set of rectangular-tesserae covering all CIO-statuses in the set S_F and the number of steps required to

enable all cells in the array to experience all CIO-statuses in the set S_F is the number of CIO-statuses in the set S_F which is $|X| \cdot |Z|$. ∎

Given a mapping $F: X \times Z \to \hat{X} \times \hat{Z}$, a reduced mapping $F]_{z_1}^{Z}$ for some $z_1 \in Z$ is the mapping $F]_{z_1}^{Z}: X \to \hat{X} \times \hat{Z}$ where $F]_{z_1}^{Z}(x) = F(x, z_1)$ for $\forall x \in X$.

Theorem 8 (Property 3)[31]: Let the cell in an array of arbitrary size be described by a mapping $F: X \times Z \to \hat{X} \times \hat{Z}$ and the two mappings $F_{\hat{X}}: X \times Z \to \hat{X}$ and $F_{\hat{Z}}: X \times Z \to \hat{Z}$ be defined by $(F_{\hat{X}}(x,z), F_{\hat{Z}}(x,z)) = F(x,z)$ for $\forall x \in X$ and $\forall z \in Z$. If the cell in the array is chosen such that the reduced mapping $F_{\hat{X}}]_{z}^{Z}: X \to \hat{X}$ is one-to-one for $\forall z \in Z$ and the reduced mapping $F_{\hat{Z}}]_{x}^{X}: Z \to \hat{Z}$ is one-to-one for $\forall x \in X$, then, there exists a set of rectangular-tesserae covering all CIO-statuses in the set S_F of CIO-statuses describing the cell, moreover, the number of steps required to enable all cells in the array to experience all CIO-statuses in the set S_F is $|X| \cdot |Z|$.

Proof: Suppose the hypothesis holds. Let $\mathcal{N} = \{(x, \hat{z}) \mid x \in X \text{ and } \hat{z} \in Z\}$. Then, there is an one-to-one correspondence between the set $\mathcal{N}$ and the set S_F because $F_{\hat{Z}}]_{x}^{X}: Z \to \hat{Z}$ is one-to-one also onto (note: $F_{\hat{Z}}]_{x}^{X}: Z \to \hat{Z}$ is one-to-one if and only if $F_{\hat{Z}}]_{x}^{X}: Z \to \hat{Z}$ is onto since $Z = \hat{Z}$ and is finite) for $\forall x \in X$. Let each number in the set $\mathcal{N}$ identifies a node in an oriented graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{E}$ is a set of edges obtained as following: an edge e from the node $(x_1, \hat{z}_1) \in \mathcal{N}$ to the node $(x_2, \hat{z}_2) \in \mathcal{N}$ exists and is in the set $\mathcal{E}$ if and only if there exists a CIO-status $\vec{q} \in S_F$ such that $\vec{q}^{X} = x_1$, $\vec{q}^{\hat{Z}} = \hat{z}_1$, $\vec{q}^{\hat{X}} = x_2$, and $\vec{q}^{Z} = \hat{z}_2$. Then, that each node in the graph $\mathcal{G}$ has precisely one outgoing and one incoming edge follows the facts that there is an one-to-one correspondence between the set $\mathcal{N}$ and the set S_F and the reduced mapping $F_{\hat{X}}]_{z}^{Z}: X \to \hat{X}$ is onto also one-to-one for $\forall z \in Z$. Thus, the graph $\mathcal{G}$ is actually a graph consisting of separated subgraphs where each subgraph is a loop. Let $L = \{L_1, L_2, \ldots, L_k\}$ be the set of all loops in the graph $\mathcal{G}$ and $\ell(L_i)$ be the number of nodes in the loop L_i for $\forall i \in \{1, 2, \ldots, k\}$. One can claim that there exists a set of rectangular-tesserae $\{T_1, T_2, \ldots, T_k\}$, where

[3] A special case in Kautz [11] (Theorem 13). The "proof" given by Kautz does not appear to be adequate.

T_i is a rectangular-tessera obtained from the information revealed by the loop L_i and is of order $\zeta(L_i)x\,\zeta(L_i)$ for $i\epsilon\{1,2,\ldots,k\}$, covering all CIO-statuses in the set S_F. Indeed, if a loop $L_i\epsilon L$ is a self-loop containing a node $(x_1,\hat{z}_1)\epsilon\mathcal{N}$, then, clearly $(x_1,\hat{z}_1,x_1,\hat{z}_1)$ is a CIO-status in the set S_F, thus, there exists a rectangular-tessera, say T_1, consisting of the CIO-status $(x_1,\hat{z}_1,x_1,\hat{z}_1)$. Suppose, in general, $L_i\epsilon L$ is a loop containing $\zeta(L_i)$ nodes $(x_1,\hat{z}_1),(x_2,\hat{z}_2),(x_3,\hat{z}_3),\ldots,$ and $(x_{\zeta(L_i)},\hat{z}_{\zeta(L_i)})$, where there exists an edge from the node $(x_i,\hat{z}_i)$ to the node $(x_{i+1},\hat{z}_{i+1})$ for $\forall i\epsilon\{1,2,3,\ldots,\zeta(L_i)-1\}$ and there exists an edge from the node $(x_{\zeta(L_i)},\hat{z}_{\zeta(L_i)})$ to the node $(x_1,\hat{z}_1)$. Let $\vec{q}_1=(x_1,\hat{z}_2,x_2,\hat{z}_1)$, $\vec{q}_2=(x_2,\hat{z}_3,x_3,\hat{z}_2)$, $\vec{q}_3=(x_3,\hat{z}_4,x_4,\hat{z}_3),\ldots,$ and $\vec{q}_{\zeta(L_i)}=(x_{\zeta(L_i)},\hat{z}_1,x_1,\hat{z}_{\zeta(L_i)})$. Then, one can see that $\vec{q}_1,\vec{q}_2,\vec{q}_3,\ldots,\vec{q}_{\zeta(L_i)}\epsilon S_F$ also $(\vec{q}_1,\vec{q}_2,\vec{q}_3,\ldots,\vec{q}_{\zeta(L_i)})$ is a row-chain and $(\vec{q}_{\zeta(L_i)},\ldots,\vec{q}_3,\vec{q}_2,\vec{q}_1)$ is a column-chain. By Theorem 3, there exists a rectangular-tessera, say T_i, of order $\zeta(L_i)x\,\zeta(L_i)$ covering all the CIO-statuses $\vec{q}_1,\vec{q}_2,\vec{q}_3,\ldots,$ and $\vec{q}_{\zeta(L_i)}$. What's more, the number of steps required to enable all cells in the array to experience all distinct CIO-statuses $\vec{q}_1,\vec{q}_2,\vec{q}_3,\ldots,$ and $\vec{q}_{\zeta(L_i)}$ is $\zeta(L_i)$ by Theorems 4 and 5. Indeed, each loop in the graph $\mathcal{G}$ identifies a rectangular-tessera covering some CIO-statuses in the set S_F. The fact that the graph $\mathcal{G}$ is a graph of isolated loops $L_1,L_2,\ldots,$ and L_k reveals the partition on the set S_F such that each CIO-status in the set S_F is in exactly one rectangular-tessera in the set $\{T_1,T_2,\ldots,T_k\}$. Thus, there exists a set of rectangular-tesserae covering all CIO-statuses in the set S_F and the number of steps required to enable all cells in the array to experience all CIO-statuses in the set S_F is the number of CIO-statuses in the set S_F which is $|X|\cdot|Z|$.

Theorem 9: There exists a rectangular-tessera consisting of t (not necessarily distinct) CIO-statuses $\vec{q}_1,\vec{q}_2,\ldots,$ and $\vec{q}_t$ only if the t-tuple of CIO-statuses $(\vec{q}_1,\vec{q}_2,\ldots,\vec{q}_t)$ is in full-balance.

Proof: This theorem is obvious in light of the fact that every row-chain is in the $x/\hat{x}$-balance and every column-chain is in $z/\hat{z}$-balance. ∎

The converse statement to Theorem 9 is not true. As a counter-example, a 3-tuple of CIO-statuses $((1,a,2,b),(2,b,2,a),(2,c,1,c))$ is in full-balance but there exists no rectangular-tessera consisting of the three CIO-statuses $(1,a,2,b)$, $(2,b,2,a)$, and $(2,c,1,c)$.

Given a set S of CIO-statuses, if there exists a set S_t of rectangular-tesserae $\{T_1, T_2, \ldots, T_t\}$, where T_i consists of K_i CIO-statuses for $\forall i \in \{1,2,\ldots,t\}$, covering all CIO-statuses in the set S, then, any $(\sum_{i=1}^{t} K_i)$-tuple of CIO-statuses consisting of all CIO-statuses in all rectangular-tesserae in the set S_t is in full-balance.

Theorem 10: Given a set S of j CIO-statuses $\{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_j\}$, there exists a nonempty set S_t of rectangular-tesserae covering all CIO-statuses in the set S only if there exists a j-tuple of positive integers $(m_1, m_2, \ldots, m_j)$ such that any $(\sum_{i=1}^{j} m_i)$-tuple of CIO-statuses consisting of m_1 copies of $\vec{q}_1$, m_2 copies of $\vec{q}_2, \ldots,$ and m_j copies of $\vec{q}_j$ is in full-balance.

Following Theorem 10, one has the following result.

Theorem 11: Given a set S_F of k CIO-statuses $\{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_k\}$ on the CIO-table, if all the possible nonnegative integer solutions to

(1) $m_1\vec{q}_1^{\vec{x}} + m_2\vec{q}_2^{\vec{x}} + \ldots + m_k\vec{q}_k^{\vec{x}} = m_1\vec{q}_1^{\vec{\hat{x}}} + m_2\vec{q}_2^{\vec{\hat{x}}} + \ldots + m_k\vec{q}_k^{\vec{\hat{x}}}$ and

(2) $m_1\vec{q}_1^{\vec{z}} + m_2\vec{q}_2^{\vec{z}} + \ldots + m_k\vec{q}_k^{\vec{z}} = m_1\vec{q}_1^{\vec{\hat{z}}} + m^2\vec{q}_2^{\vec{\hat{z}}} + \ldots + m_k\vec{q}_k^{\vec{\hat{z}}}$

for $m_1, m_2, \ldots,$ and m_k render some $m_i = 0$, $i \in \{1,2,\ldots,k\}$, then $\vec{q}_i$ can not be covered by any rectangular-tessera.

Given a set S_F of k CIO-statuses $\{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_k\}$ on the CIO-table specifying the cell in an array of arbitrary size, the nonnegative integer solution to (1) $\sum_{i=1}^{k} m_i\vec{q}_i^{\vec{x}} = \sum_{i=1}^{k} m_i\vec{q}_i^{\vec{\hat{x}}}$ and (2) $\sum_{i=1}^{k} m_i\vec{q}_i^{\vec{z}} = \sum_{i=1}^{k} m_i\vec{q}_i^{\vec{\hat{z}}}$ for $m_1, m_2, \ldots,$ and m_k is not unique. The existence of a solution where m_i's are all positive integers reveals the possibility that one might find a set of rectangular-tesserae covering all CIO-statuses in the set S_F. If all the

possible nonnegative integer solutions to (1) and (2) for m_i's render at least one $m_j = 0$, $j \in \{1,2,\ldots,k\}$, then at least $\vec{q}_j$ can not be covered by any rectangular-tessera, consequently, at least the existence of a set S_t of prime tessellations such that $\vec{q}_j$ occurs on C_{ip} in at least one tessellation in the set S_t for $\forall i \in S_I$ and $\forall p \in S_I$ is necessary for the Covering Condition for testing the array to be satisfied.

For a CIO-status $\vec{q}_j$ in the set S_F of all CIO-statuses on the CIO-table specifying the cell in an array of arbitrary size where $\vec{q}_j$ can not be covered by any rectangular-tessera, a <u>prime tessellation with respect to the CIO-status $\vec{q}_j$</u> is a mapping $G: C \to S_F$ where $C = \{C_{ip} \mid i \in \{1,2,3,\ldots,M\}$ and $p \in \{1,2,3,\ldots,N\}\}$, $G(C_{MN}) = \vec{q}_j$, where $M \to \infty$ and $N \to \infty$, and for every $C_{ip} \in C$, the CIO-status on C_{ip}, i.e. $G(C_{ip})$, is compatible with the CIO-statuses on all adjacent cells in C. The existence of a prime tessellation with respect to the CIO-status $\vec{q}_j$ implies the existence of a set S_t of prime tessellations such that $\vec{q}_j$ occurs on C_{ip} in at least one tessellation in the set S_t for $\forall i \in S_I$ and $\forall p \in S_I$ and thus enables all cells in the array to experience the CIO-status $\vec{q}_j$.

b. <u>Tessellations for Arrays of Cells with Binary Horizontal Input and Binary Vertical Input</u>

It is to be determined in this section whether or not the Covering Condition can be satisifed for each one in a class of arrays, say of size MxN where $M \geq 2$ and $N \geq 2$, with each cell having a binary horizontal input and a binary vertical input. (That it also has a binary horizontal output and a binary vertical output is understood.) Specifically, for an array in this class with its cell described in terms of a set of four CIO-statuses $\{\vec{q}_1, \vec{q}_2, \vec{q}_3, \vec{q}_4\}$ on the CIO-table, can the Covering Condition be satisfied? If yes, what is the number of steps required to apply all input combinations to all cells in the array?

The Procedure P to be followed is following.

S1. Solve $m_1\vec{q}_1^{x} + m_2\vec{q}_2^{x} + m_3\vec{q}_3^{x} + m_4\vec{q}_4^{x} - m_1\vec{q}_1^{\hat{x}} + m_2\vec{q}_2^{\hat{x}} + m_3\vec{q}_3^{\hat{x}} + m_4\vec{q}_4^{\hat{x}}$ and $m_1\vec{q}_1^{z} + m_2\vec{q}_2^{z} + m_3\vec{q}_3^{z} + m_4\vec{q}_4^{z} - m_1\vec{q}_1^{\hat{z}} + m_2\vec{q}_2^{\hat{z}} + m_3\vec{q}_3^{\hat{z}} + m_4\vec{q}_4^{\hat{z}}$ for a nontrivial nonnegative integer solution for m_1, m_2, m_3, and m_4. (A minimal solution is preferable.)

S2. If all the possible nontrivial nonnegative integer solutions for m_1, m_2, m_3, and m_4 in step S1 render some $m_i = 0$, where $i \in A = \{j \mid m_j = 0\}$, go to step S6. Otherwise, proceed to the next step.

S3. For a solution for m_1, m_2, m_3, and m_4 obtained in step S1, let $B = \{j \mid m_j > 0\}$. Form a $(\sum_{j\in B} m_j)$-tuple of CIO-statuses with m_j copies of $\vec{q}_j$ for all $j \in B$. Let it be $\mathcal{F}$.

S4. Partition $\mathcal{F}$ into several sub-$(\sum_{j\in B} m_j)$-tuples $\mathcal{F}_1, \mathcal{F}_2, \ldots$, and $\mathcal{F}_k$ such that $\mathcal{F}_i$ is a (minimal) sub-$(\sum_{j\in B} m_j)$-tuple of $\mathcal{F}$ satisfying full-balance condition for each $i \in \{1, 2, \ldots, k\}$ and $\vec{q}_i$ is in at least one of $\mathcal{F}_1, \mathcal{F}_2, \ldots$, and $\mathcal{F}_k$ for every $j \in B$.

S5. Obtain rectangular-tesserae $T_1, T_2, \ldots$, and T_k from $\mathcal{F}_1, \mathcal{F}_2, \ldots$, and $\mathcal{F}_k$. Go to step S8.

S6. For each $\vec{q}_i \in \{\vec{q}_j \mid j \in A\}$, determine a prime tessellation with respect to $\vec{q}_i$ using Procedure P_p.

S7. If all prime tessellations stated in step S6 are found, go to step S3. Otherwise, return with an indication that the Covering Condition is not satisfied.

S8. (Note that the Covering Condition is satisfied.) From the necessary prime tessellations obtained, determine the number of steps required for enabling all cells in the array to experience all $\vec{q}_i \in \{\vec{q}_j \mid j \in A\}$.

S9. Determine the number of steps required for enabling all cells in the array to experience all CIO-statuses in the set $\{\vec{q}_j \mid j \in B\}$ from the nonprime tessellations induced by the rectangular-tesserae obtained in step S5, take into account the possibility where the number of

steps for enabling all cells in the array to experience some CIO-status in the set $\{\vec{q}_j | j \epsilon B\}$ might have been accounted for in step S8, however.

S10. Take the sum of results in step S8 and in step S9. Return with the result.

Procedure P_p for determining a prime tessellation with respect to a given $\vec{q}_i \epsilon S = \{\vec{q}_1, \vec{q}_2, \vec{q}_3, \vec{q}_4\}$ in an array with a binary x-input, a binary z-input, a binary $\hat{x}$-output, and a binary $\hat{z}$-output:

I.1. For the convenience, denote the four CIO-statuses $\vec{q}_1, \vec{q}_2, \vec{q}_3$, and $\vec{q}_4$ in a given CIO-table by (1), (2), (3), and (4), respectively.

I.2. Consider finding a prime tessellation with respect to a $\vec{q}_i \epsilon S$ as marking unit squares each with some $\vec{q} \epsilon S$ on the second quadrant of the infinite plane such that the most lower-right square is marked with $\vec{q}_i$ and for each square the compatibility of the CIO-status with that of its neighboring squares is satisfied.

I.3. An ordered pair (r,c) is used to identify a square, where r and c identify, respectively, the row and the column the square is in. The enumeration on columns is from right to left and that on rows is from bottom to top. The distance d of (r,c) square is defined as $d(r,c) \triangleq r+c$.

I.4. For some positive integers M and N, the Procedure P_f for marking the array of MxN squares each with some $\vec{q} \epsilon S$ such that (1,1) square is marked with $\vec{q}_i$ and for each square the compatibility of the CIO-status with that of its neighboring squares is satisfied, is following:

I.4.a. Set COPY=A and FLAG=G.

I.4.b. Proceed the following Temporary Marking Procedure for each of unmarked squares in the array starting from one square with the smallest distance and in the order of monotonically nondecreasing

in the distance. After going through the Temporary Marking Procedure for all unmarked squares, if the number of unmarked squares remains the same, go to step I.4.g., otherwise, proceed to step I.4.c.

Temporary Marking Procedure:

(1) For a typical square identified by (r,c), if (r,c-1) square has been marked and each CIO-status has the same x-component, say α, then mark (r,c) square with all $\vec{q}_\epsilon \{\vec{q}_\epsilon S | \vec{q}^{\hat{x}} = \alpha\}$ separated by a slash "/" and make the " ← " indication at the common boundary of (r,c) and (r,c-1) squares, go to step (5). Otherwise, proceed to the next step.

(2) If (r-1,c) square has been marked and each CIO-status has the same z-component, say β, then mark (r,c) square with all $\vec{q}_\epsilon \{\vec{q}_\epsilon S | \vec{q}^{\hat{z}} = \beta\}$ separated by a slash "/" and make the " ↑ " indication at the common boundary of (r,c) and (r-1,c) squares, go to step (5). Otherwise, proceed to the next step.

(3) If (r,c+1) square has been marked and each CIO-status has the same $\hat{x}$-component, say δ, then mark (r,c) square with all $\vec{q}_\epsilon \{\vec{q}_\epsilon S | \vec{q}^{\hat{x}} = \delta\}$ separated by a slash "/" and make the " → " indication at the common boundary of (r,c) and (r,c+1) squares, go to step (5). Otherwise, proceed to the next step.

(4) If (r+1,c) square has been marked and each CIO-status has the same $\hat{z}$-component, say θ, then mark (r,c) square with all $\vec{q}_\epsilon \{\vec{q}_\epsilon S | \vec{q}^{\hat{z}} = \theta\}$ separated by a slash "/" and make the " ↓ " indication at the common boundary of (r,c) and (r+1,c) squares, go to step (5). Otherwise, make no marking on (r,c) square, proceed to the next step.

(5) Return (to the calling routine).

I.4.c. Proceed the following Compatibility Marking Procedure for each of all pairs of marked neighboring squares (r_1,c_1) and (r_2,c_2), where $d(r_2,c_2) < d(r_1,c_1)$ and no indication appears at their common boundary.

Compatibility Marking Procedure:

(1) If under no circumstance can a CIO-status on (r_1, c_1) square be compatible with a CIO-status on (r_2, c_2) square, return (to the calling routine) with FLAG=R. Otherwise, proceed to the next step.

(2) If $c_1 = c_2$, go to step (3). Otherwise, $r_1 = r_2$ here, if all CIO-statuses on (r_2, c_2) square have the same x-component, say α, delete those CIO-statuses on (r_1, c_1) having some $\hat{x}$-component different from α, go to step (4), else if all CIO-statuses on (r_1, c_1) square have the same $\hat{x}$-component, say β, delete those CIO-statuses on (r_2, c_2) square having some x-component different from β, go to step (4), otherwise, go to step (5).

(3) If all CIO-statuses on (r_2, c_2) square have the same z-component, say δ, delete those CIO-status on (r_1, c_1) square having some $\hat{z}$-component different from δ, go to step (4), else if all CIO-statuses on (r_1, c_1) square have the same $\hat{z}$-component, say θ, delete those CIO-statuses on (r_2, c_2) square having some z-component different from θ, proceed to step (4), otherwise, go to step (5).

(4) Make the "Δ" indication at the common boundary of (r_1, c_1) and (r_2, c_2) squares.

(5) Return (to the calling routine).

I.4.d. If FLAG=R and COPY=A, return (to the calling routine) with an indication that it is impossible to mark the array of MxN squares as specified. If FLAG=R and COPY=R, go to step I.4.h. Otherwise, FLAG=G here, proceed to the next step.

I.4.e. If there exists any pair of marked neighboring squares where their common boundary is without any indication, go to step I.4.g. Otherwise, proceed to the next step.

I.4.f. If by heuristic observation over the pattern of marked squares developed so far, one can (correctly) obtain the marked array of MxN squares satisfying all the constraints on it, return to the calling routine with the result. Otherwise, go to step I.4.b.

I.4.g. Let RECORD be a copy of the array of marked squares with the indications on common boundaries of neighboring squares developed so far. Among those squares marked with multiple CIO-statuses, select the one containing $\vec{q}_i$, provided there exists one, otherwise drop this criterion for selection also with the smallest distance, break a tie by selecting the one closest to a square marked with single CIO-status during the temporary marking. Make a choice among the marked CIO-statuses on that chosen square by choosing $\vec{q}_i$ if possible, otherwise choose some other one. Set COPY=R. Let RUNNING-COPY be a copy of RECORD with the exception that the chosen square is marked with the chosen CIO-status instead of all CIO-statuses. Record the choices on RECORD. Use RUNNING-COPY as the marking on the arrray of squares developed so far, go to step I.4.c.

I.4.h. From RECORD, construct a current RUNNING-COPY which is the same as RECORD except the chosen square is marked with an alternate CIO-status. Record this alternate choice on RECORD. If this is a last alternate CIO-status on the chosen square, set COPY=A, otherwise, COPY=R stands. Use the current RUNNING-COPY as the marking on the array of squares developed so far, go to step I.4.c.

P1. Execute the Procedure P_f with M=N=2. If the result is an indication that it is impossible to mark the array of 2x2 squares as specified, proceed to the next step. Otherwise, go to step P3.

P2. Return (to the calling routine) with an indication that there exists no prime tessellation with respect to the given $\vec{q}_i$.

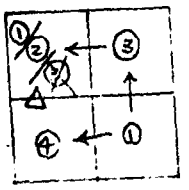
P3. Execute the procedure P_f with some larger integer for M and N, say M=N=6. In doing so, the result obtained so far can, of course, be utilized. If the result is an indication that it is impossible to mark the array of MxN squares as specified, go to step P2. Else if by heuristic observation over the pattern of the result so far, one can (correctly) obtain a prime tessellation with respect to the given CIO-status, return to the calling routine with the completed result, otherwise, proceed to the next step.

P4. Execute the Procedure P_f with some larger integer for M and N, say M=N=12. The handling of the result is same as that stated in step P3.

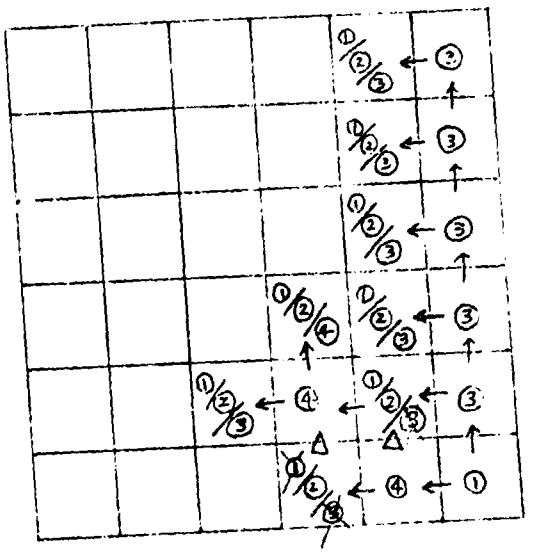
P5.

.
.
.

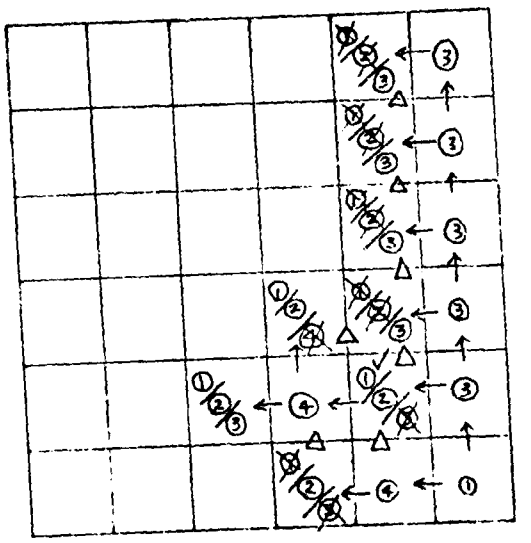
Example 1: Consider an MxN array with a specification of the cell as $F_{\hat{x}}(x,z) = \bar{x} + \bar{z}$ and $F_{\hat{z}}(x,z) = \bar{x} + z$. Let $\vec{q}_1 = (0,0,1,1)$, $\vec{q}_2 = (0,1,1,1)$, $\vec{q}_3 = (1,0,1,0)$, and $\vec{q}_4 = (1,1,0,1)$. All nontrivial nonnegative integer solutions to $m_1 \cdot 0 + m_2 \cdot 0 + m_3 \cdot 1 + m_4 \cdot 1 = m_1 \cdot 1 + m_2 \cdot 1 + m_3 \cdot 1 + m_4 \cdot 0$ and $m_1 \cdot 0 + m_2 \cdot 1 + m_3 \cdot 0 + m_4 \cdot 1 \equiv m_1 \cdot 1 + m_2 \cdot 1 + m_3 \cdot 0 + m_4 \cdot 1$ for m_1, m_2, m_3, and m_4 render $m_1 = 0$. This reveals the need to find a prime tessellation with respect to $\vec{q}_1$ as far as enabling all cells in the array to experience the CIC-status $\vec{q}_1$ is concerned. Some intermediate steps in following the Procedure P_p for determining a prime tessellation with respect to $\vec{q}_1$ are as follows.



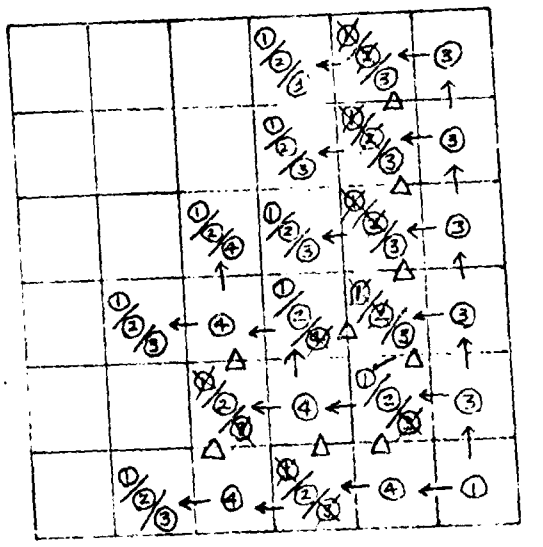
Step a



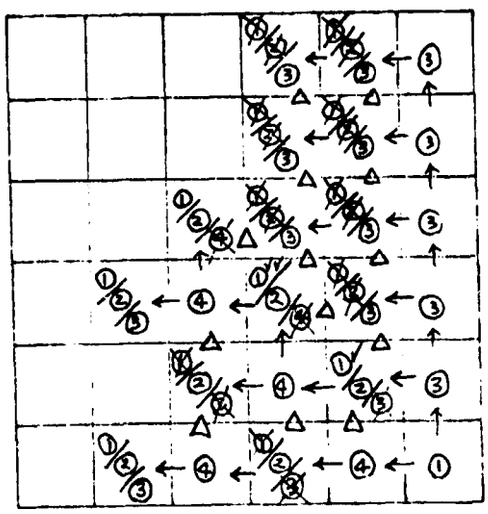
Step b



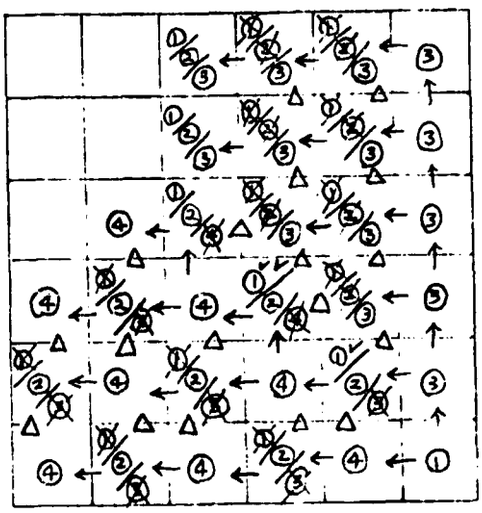
Step c The choice is indicated by ✓



Step d

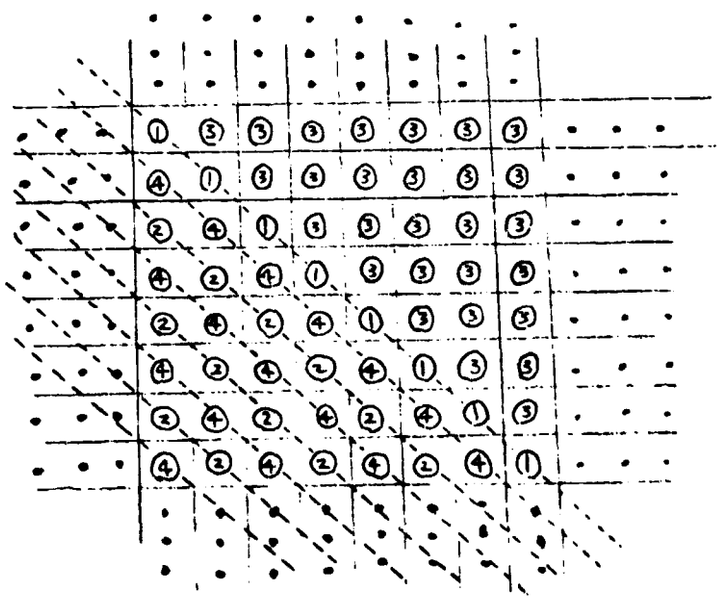


Step e

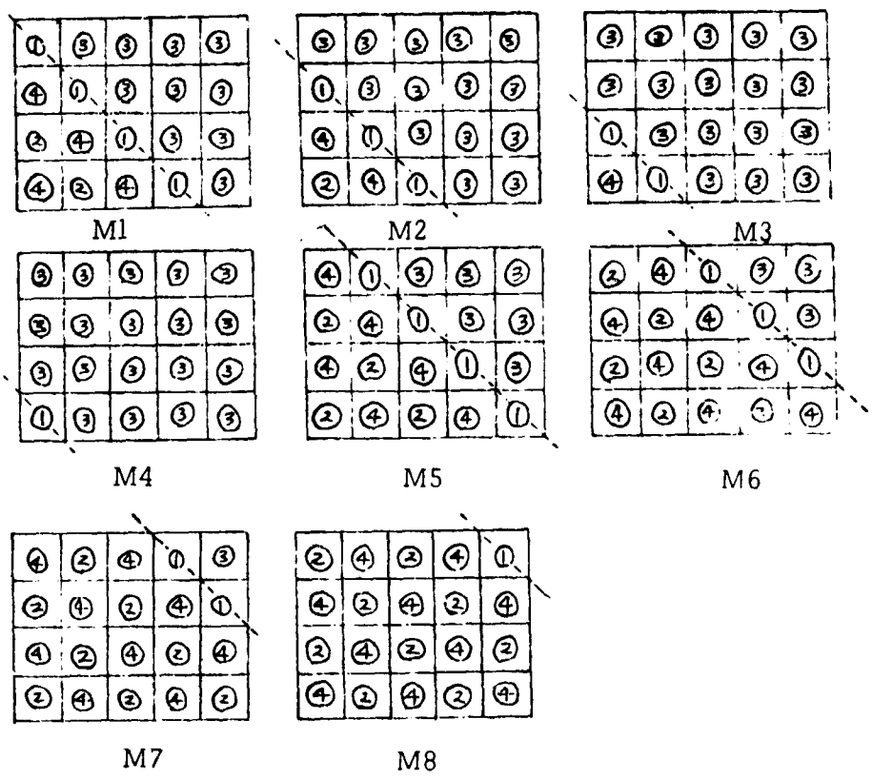


Step f

By heuristic observation over the pattern of marked squares as shown in step f, one can obtain a prime tessellation with respect to $\vec{q}_1$ as follows.



This prime tessellation with respect to the CIO-status $\vec{q}_1$ implies that M+N-1 steps are required to enable all cells in the array to experience the CIO-status $\vec{q}_1$. To see this point, one can observe a specific case, say for M = 4 and N = 5, where 4+5-1 = 8 CIO-statuses-compatible mappings on the 4x5 array can be obtained from the prime tessellation with respect to $\vec{q}_1$ as follows.



With the application of 00000 to the vertical boundary inputs, in the order from left to right, and 0101 to the horizontal boundary inputs, in the order from top to bottom, of the 4x5 array, all cells on the dotted line in M1 have the input combination corresponding to $\vec{q}_1$ applied. Similarly, with the application of 11000 to the vertical boundary inputs and 0101 to the horizontal boundary inputs of the 4x5 array, all cells on the dotted line in

M6 have the input combination corresponding to $\vec{q}_1$ applied. These 8 CIO-statuses-compatible mappings on the 4x5 array imply that 8 steps are required to enable all cells in the 4x5 array to experience the CIO-status $\vec{q}_1$. A nontrivial nonnegative integer solution to $m_1 \cdot 0 + m_2 \cdot 0 + m_3 \cdot 1 + m_4 \cdot 1 \equiv m_1 \cdot 1 + m_2 \cdot 1 + m_3 \cdot 0 + m_4 \cdot 0$ and $m_1 \cdot 0 + m_2 \cdot 1 + m_3 \cdot 0\ m_4 \cdot 1 \equiv m_1 \cdot 1 + m_2 \cdot 1 + m_3 \cdot 0 + m_4 \cdot 1$ for m_1, m_2, m_3, m_4 is $m_1 = 0$, $m_2 = 1$, $m_3 = 1$, and $m_4 = 1$. Let $\xi = (\vec{q}_2, \vec{q}_3, \vec{q}_4)$. Notice that $\xi_1 = (\vec{q}_3)$ and $\xi_2 = (\vec{q}_2, \vec{q}_4)$ are both in full-balance, furthermore, $\vec{q}_3$ alone is a rectangular-tessera and $(\vec{q}_2, \vec{q}_4)$ is a rectangular-tessera of order 1x2. Thus, it follows from Theorem 4 and Theorem 5 that one step is required to enable all cells in the array to experience the CIO-status $\vec{q}_3$ independent of the size of the array and two steps are required to enable all cells in the array to experience the CIO-statuses $\vec{q}_2$ and q_4 independent of the size of the array. Therefore, 1+2+(M+N-1) steps are required to enable all cells in the MxN array to experience all CIO-statuses $\vec{q}_1, \vec{q}_2, \vec{q}_3$, and $\vec{q}_4$.

The results of this subsection are summarized in Table 1. Of all the possible $2^{2^2} \times 2^{2^2} = 256$ cases, when a prime tessellation with respect to some CIO-status is required, most cases require at most reaching step P3 in Procedure P_p. There are 96 cases corresponding to those entries in Table 1 with *'s where in each of these cases at least one input combination is not applicable to the lower-right cell of a 2x2 array and thus not applicable to the typical cell in any MxN array, where $M \geq 2$ and $N \geq 2$.

An exhaustive search on the result on Table 1 reveals one fact about the class of arrays with each cell having a binary horizontal input and a binary vertical input related to the minimum number of array tests which is the number of tests required for testing a single cell, i.e., four in this class of arrays. This is stated in Theorem 12.

$F_{\bar{x}}$ \ $F_{\bar{z}}$	0	1	x	$\bar{x}$	z	$\bar{z}$	$x\bar{z}+\bar{x}z$	$xz+\bar{x}\bar{z}$	xz	$\bar{x}+\bar{z}$	$x\bar{z}$	$\bar{x}+z$	$\bar{x}z$	$x+\bar{z}$	$\bar{x}\bar{z}$	$x+z$
0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
x	*	*	1+1+2 (n)	1+1+2 (n)	1+1+1 +1 (n)	2+2 (n)	1+1+2 (n)	1+1+2 (n)	1+1+1 +M (n&p)	1+2+2 (n)	1+2+2 (n)	1+1+1 +M (n&p)	1+1+1 +M (n&P)	1+2+2 (n)	1+2+2 (n)	1+1+1 +M (n&p)
$\bar{x}$	*	*	2+2 (n)	2+2 (n)	2+2 (n)	2+2 (n)	4 (n)	4 (n)	2+2M (n&p)	2+4 (n)	2+4 (n)	2+2+ 2M (n&p)	2+2+ 2M (n&p)	2+4 (n)	2+4 (n)	2+2+ 2M (n&p)
z	*	*	1+1+2 (n)	2+2 (n)	1+1+2 (n)	2+2 (n)	1+3 (n)	1+3 (n)	1+1+N+ (M+N-1)· U(M-N) (n&p)	2+3 (n)	*	*	*	*	2+3 (n)	1+1+N+ (M+N-1)· U(M-N) (n&p)
$\bar{z}$	*	*	2+2 (n)	1+1+2 (n)	1+1+2 (n)	2+2 (n)	1+3 (n)	1+3 (n)	*	*	2+3 (n)	1+1+N+ (M+N-1)· U(M-N) (n&p)	1+1+N+ (M+N-1)· U(M-N) (n&p)	2+3 (n)	*	*
$x\bar{z}+\bar{x}z$	*	*	1+3 (n)	1+3 (n)	1+3 (n)	4 (n)	1+3 (n)	1+3 (n)	1+1+ 2^M (n&p)	3+3 (n)	1+4 (n)	1+2+M (n&p)	1+1+ N+M·N (n&p)	3+3 (n)	1+4 (n)	1+2+M (n&p)
$xz+\bar{x}\bar{z}$	*	*	1+3 (n)	1+3 (n)	1+1+2 (n)	4 (n)	1+3 (n)	1+3 (n)	1+2+M (n&p)	1+4 (n)	3+3 (n)	1+1+ N+M·N (n&p)	1+2+M (n&p)	1+4 (n)	3+3 (n)	1+1+ N+M·N (n&p)
xz	*	*	1+1+M+ (M+N-1)· U(N-M) (n&p)	*	1+1+1 +N (n&p)	2+2N (n&p)	1+1+ 2^N (n&p)	1+2+N (n&p)	1+1+ M+N (n&p)	*	*	1+1+N +M·N (n&p)	1+1+M +M·N (n&p)	1+2+ (M+N-1) (n&p)	2+2+ M+ 2(N-1) (n&p)	1+1+1+ (M+N-1) (n&p)
$\bar{x}+\bar{z}$	*	*	2+3 (n)	*	2+3 (n)	2+4 (n)	3+3 (n)	1+4 (n)	*	2+3 (n)	3+4 (n)	1+2+ (M+N -1) (n&p)	*	3+4 (n)	*	2+2+ N+ 2(M-1) (n&p)
$x\bar{z}$	*	*	*	1+1+M+ (M+N-1)· U(N-M) (n&p)	1+1+ 1+N (n&p)	2+2+ 2N (n&p)	1+1+ M+M·N (n&p)	1+2+N (n&p)	1+1+ N+M·N (n&p)	*	*	1+1+ M+N (n&p)	1+1+1 +(M+ N-1) (n&p)	2+2+ M+ 2(N-1) (n&p)	3+(M +N-1) (n&p)	1+1+ M+M·N (n&p)
$\bar{x}+z$	*	*	*	2+3 (n)	1+2+2 (n)	2+4 (n)	3+3 (n)	1+4 (n)	1+2+ (M+N -1) (n&p)	3+4 (n)	2+3 (n)	*	2+2+ N+ 2(M-1) (n&p)	*	3+4 (n)	*
$\bar{x}z$	*	*	*	2+3 (n)	1+2+2 (n)	2+4 (n)	1+4 (n)	3+3 (n)	*	3+4 (n)	*	2+2+ N+ 2(M-1) (n&p)	*	2+3 (n)	3+4 (n)	1+2+ (M+N -1) (n&p)
$x+\bar{z}$	*	*	*	1+1+M +(M+N -1)· U(N-M) (n&p)	1+1+1 +N (n&p)	2+2+ 2N (n&p)	1+2+ N (n&p)	1+1+M +M·N (n&p)	1+1+M +M·N (n&p)	1+2+ (M+N -1) (n&p)	2+2+ M+ 2(N-1) (n&p)	1+1+1 (M+N -1) (n&p)	1+1+ M+N (n&p)	*	*	1+1+N +M·N (n&p)
$\bar{x}\bar{z}$	*	*	2+3 (n)	*	2+3 (n)	2+4 (n)	1+4 (n)	3+3 (n)	2+2+ N+ 2(M-1) (n&p)	*	3+4 (n)	*	1+2+ (M+N -1) (n&p)	3+4 (n)	2+3 (n)	*
$x+z$	*	*	1+1+M +(M+N -1)· U(N-M) (n&p)	*	1+1+1 +N (n&p)	1+1+1 +N (n&p)	1+2+N (n&p)	1+1+M +M·N (n&p)	1+1+1 +(M+N -1) (n&p)	2+2+ M+ 2(N-1) (n&p)	1+2+ (M+N -1) (n&p)	1+1+M +M·N (n&p)	1+1+N +M·N (n&p)	*	*	1+1+ M+N (n&p)

Symbols: (n): only nonprime tessellations required
(n&p): nonprime and prime tessellations required
*: at least one CIO-status can not be experienced by a typical cell in the array

Table 1. The number of steps required to apply all cell input combinations to all cells in an array of size MxN in the class of arrays with each cell having a binary horizontal input, a binary vertical input, a binary horizontal output, and a binary vertical output.

Theorem 12: Given a 2-dimensional array of arbitrary size with each cell having a binary horizontal input, a binary vertical input and a set of CIO-statuses $\{\vec{q}_1, \vec{q}_2, \vec{q}_3, \vec{q}_4\}$ describing the cell in the array, the number of steps required to apply all possible input combinations to all cells in the array is equal to the minimum number of array tests, which is four, if and only if $(\vec{q}_1, \vec{q}_2, \vec{q}_3, \vec{q}_4)$ is in full-balance.

c. Tessellations for Arrays of Cells with Multiple Horizontal Input Terminals and Multiple Vertical Input Terminals

For the general class of arrays of cells each with multiple horizontal input terminals and multiple vertical input terminals, when a prime tessellation with respect to a given CIO-status in a set of CIO-statuses specifying a cell is to be found, the Procedure P_p in the preceding subsection can be extended to adapt to the general class of arrays here. One thing one must be aware of is that the condition that n-tuple of CIO-statuses C is in full-balance is, in general, not sufficient for the existence of a rectangular-tessera or rectangular-tesserae covering all distinct CIO-statuses in the n-tuple C.

2. Existence of a Sensitized Path

Let the cell in a two-dimensional array be described in terms of two mappings $F_{\hat{x}}: X \times Z \to \hat{X}$ and $F_{\hat{z}}: X \times Z \to \hat{Z}$, then, one has the following theorem regarding the existence of a sensitized path.

Theorem 13: If the logic of the cell in a two-dimensional array is chosen such that

(1) the mapping $F: X \times Z \to \hat{X} \times \hat{Z}$, defined by $F(x,z) = (F_{\hat{x}}(x,z), F_{\hat{z}}(x,z))$ for $\forall x \in X$ and $\forall z \in Z$, is one-to-one (and/or onto),

(2) $F_{\hat{x}}: X \times Z \to \hat{X}$ is as such that any change in x while z is constant or any change in z while x is constant induces some change in $\hat{x}$ and $F_{\hat{z}}: X \times Z \to \hat{z}$ is any (well-defined) mapping,

(3) $F_{\hat{z}}: X \times Z \to \hat{Z}$ is as such that any change in x while z is constant or any change in z while x is constant induces some change in $\hat{z}$ and $F_{\hat{x}}: X \times Z \to \hat{X}$ is any (well defined) mapping, or

(4) $F_{\hat{x}}: XxZ \rightarrow \hat{X}$ is as such that any change in x while z is constant induces some change in $\hat{x}$ and $F_{\hat{z}}: XxZ \rightarrow \hat{Z}$ is as such that any change in z while x is constant induces some change in $\hat{z}$,

then, for all possible input combinations to a cell and each possible cell output change due to a fault, there exists at least one sensitized path from that cell to one of the boundary outputs.

Proof: Part (1)

Suppose a cell C_1 on the diagonal line D_i is faulty as indicated in Figure 2. Then, at least one of the outputs of the cell C_1 is perturbed

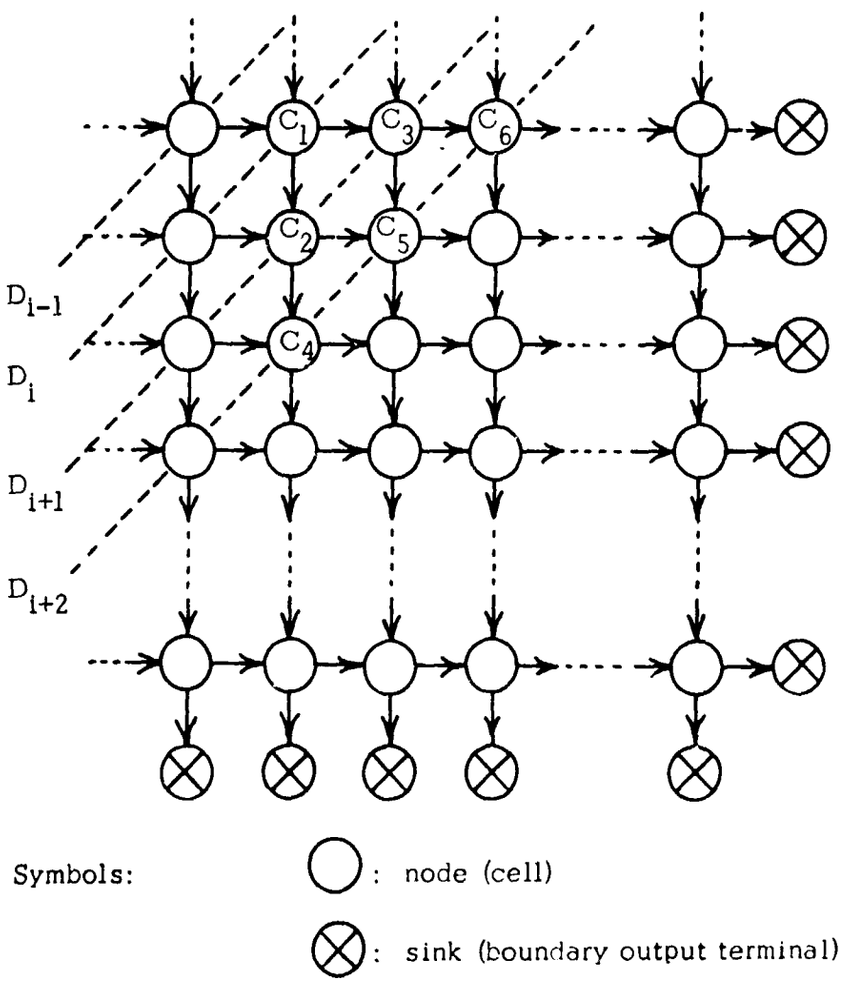


Figure 2. Graph for the Proof of Part (1) of Theorem 13.

from its nominal value. This causes the input change to at least one of two cells C_2 and C_3 on the diagonal line D_{i+1} receiving an input from the cell C_1. Thus, at least one of the outputs of cells C_2 and C_3 is perturbed from its nominal value. Eventually, the output change of cell C_1 is propagated through diagonal lines D_{i+1}, D_{i+2}, etc. until one or more of boundary output terminals is reached.

Part (2)

An equivalent statement for that $F_{\hat{x}}: X \times Z \rightarrow \hat{X}$ is as such that any change in x while z is constant or any change in z while x is constant induces some change in $\hat{x}$ is that $F_{\hat{x}}: X \times Z \rightarrow \hat{X}$ is a mapping where $|\hat{X}| \geq |Z|$, ($\hat{X} = X$ of course), $F_{\hat{x}}\left]\begin{matrix} Z \\ z \end{matrix}\right.$ is one-to-one for $\forall z \in Z$, and $F_{\hat{x}}\left]\begin{matrix} X \\ x \end{matrix}\right.$ is one-to-one $\forall x \in X$. Now, suppose there is a $\hat{x}$-output change due to a fault in a cell C_f. This change in $\hat{x}$ of the cell C_f, or equivalently, the change in x for the cell C_g immediately to the right of the cell C_f, causes at least the change in $\hat{x}$ of the cell C_g. Thus, the $\hat{x}$-output change due to a fault is at least propagated along a row until the boundary output terminal is reached. Suppose the fault in cell C_f causes the change in $\hat{z}$-output of cell C_f. This change in $\hat{z}$ of the cell C_f, or equivalently, the change in z for the cell C_h immediately below the cell C_f, causes at least the change in $\hat{x}$ of the cell C_h. And, this change in $\hat{x}$ of the cell C_h is then at least propagated along a row until the boundary output terminal is reached. Of course, the fault in cell C_f causing changes of both $\hat{x}$-output and $\hat{z}$-output will be detected at least at the boundary output terminal in the same row as cell C_f.

Part (3)

This is symmetrical to Part (2).

Part (4)

Here, $F_{\hat{x}}: X \times Z \rightarrow \hat{X}$ is a mapping where $F_{\hat{x}}\Big]_{z}^{Z}$ is one-to-one for $\forall z \in Z$ and $F_{\hat{z}}: X \times Z \rightarrow \hat{Z}$ is a mapping where $F_{\hat{z}}\Big]_{x}^{X}$ is one-to-one for $\forall x \in X$. Any $\hat{x}$-output change due to a fault in a cell C_f, or equivalently, a change in x for the cell C_g immediately to the right of the cell C_f, causes the change in $\hat{x}$ of the cell C_g. This $\hat{x}$-output change due to a fault is propagated along a row until the boundary output terminal is reached. Any $\hat{z}$-output change due to a fault in a cell C_f, or equivalently, a change in z for the cell C_h immediately below the cell C_f, causes the change in $\hat{z}$ of the cell C_h. This $\hat{z}$ output change due to a fault is propagated along a column until the boundary output terminal is reached. Certainly, the fault in cell C_f causing changes of both $\hat{x}$-output and $\hat{z}$-output will be detected at the boundary output terminal in the same row as cell C_f as well as at that in the same column as cell C_f. ∎

Recall that the presence of a single faulty cell in an array can be detected if and only if both the Covering Condition and the Sensitized Path Condition are satisfied.

D. Remarks

If the realization (at gate level) of the cell in an array is known or the erroneous effects on the cell input-output behavior due to all possible faults are known, then, a set of essential input combinations for testing a cell completely is obtainable [1], [8], [13], [16], [18], which is some subset of a set of all cell input combinations. Chances are the set of essential input combinations for testing a cell completely is a proper subset of the set of all cell input combinations. During the process of finding a non-empty set of tessellations to cover all CIO-statuses in a given CIO-table in a manner described in this chapter, one can find out, for each CIO-status $\vec{q}$ in the given CIO-table, whether or not $\vec{q}$ can appear in some tessellation (be it nonprime or prime). As long as each of the essential input

combinations is applicable to every cell in the array, or each of the corresponding CIO-statuses appears in some tessellation, the Covering Condition is satisfied for the given array even though there exists some input combination which is not applicable to some cell in the array.

On the other hand, there is a possibility that the Covering Condition is not satisfied when considering only the set of essential CIO-statuses corresponding to the essential input combinations but the Covering Condition would be satisfied if the set of all CIO-statuses in the CIO-table or some alternative realization of the cell were considered.

Chapter III

FAULT DETECTION AND LOCATION IN FIRST CATEGORY SEQUENTIAL CELLULAR ARRAYS

A. Introduction

A first category sequential array is a sequential array of cells each having some memory element as well as some logic circuit, where the state of each memory element in each cell can be set or reset through the external control. The permutation array and the accumulator array in Kautz [12] and the class of cutpoint cellular logic arrays in Minnick [14], where a cutpoint being "open" or "closed" can be though of the state of a memory element being "0" or "1", are examples of the first category sequential arrays. Under the assumption that all input-state combinations are necessary to test a cell completely, the two necessary conditions that must be satisfied for array testing for the detection of the presence of a faulty cell in a first category sequential array are same as the Covering Condition and the Sensitized Path Condition in Chapter II except that "input combination" is replaced by "input-state combination". They are:

> Condition A: Every input-state combination must be applicable to every cell in the array.
>
> Condition B: For each input-state combination to a cell and each possible cell output change due to a fault, there must exist at least one sensitized path from that cell to one of boundary outputs.

Again, these two necessary conditions together are sufficient for the detection of the presence of single faulty cell in a first category sequential array. The results with regard to the fault detection of combinational

arrays can be easily extended to the case here. In this chapter, an array shall be understood to be a first category sequential cellular array.

For a cell in an array, the finite horizontal input set, the finite vertical input set, the finite nonempty state set, the finite horizontal output set, and the finite vertical output set are denoted by X, Z, S, $\hat{X}$, and $\hat{Z}$, respectively. x, z, s, $\hat{x}$, and $\hat{z}$ are arbitrary elements of X, Z, S, $\hat{X}$, and $\hat{Z}$, respectively.

B. One-Dimensional Arrays

1. Arrays With Autonomous Cells, where $Z=\hat{Z}=\emptyset$ and $X=\hat{X}\neq\emptyset$

Let the cell be described in terms of a mapping $F:SxS\rightarrow\hat{X}$, then one has the following.

Theorem 14: A one-dimensional autonomous array is testable if and only if there exists some $s\epsilon S$ such that the reduced mapping $F\Big|_s^{S}:X\rightarrow\hat{X}$ is one-to-one (and/or onto); moreover, the number of tests required to test the array is in the range from $|X|\cdot|S|$ to $|X|\cdot(n\cdot|S|-n+1)$ where n is the number of cells in the array.

Theorem 15: Any single faulty cell in a one-dimensional autonomous array can be located with an uncertainty of 1 if and only if there exists some $s\epsilon S$ such that (1) the reduced mapping $F\Big|_s^{S}:X\rightarrow\hat{X}$ is one-to-one and (2) the response at the boundary output with respect to the $|X|$ test steps, where all cells are set to the state s and each of $|X|$ possible cell input combinations is applied to the first cell step by step, is nominal.

2. Arrays With $Z=\emptyset$, $\hat{Z}\neq\emptyset$, and $X=\hat{X}\neq\emptyset$

Let the cell be described in terms of two mappings $F_{\hat{x}}:SxX\rightarrow\hat{X}$ and $F_{\hat{z}}:SxX\rightarrow\hat{Z}$, then one has the following.

Theorem 16: A one-dimensional array with $Z=\emptyset$, $\hat{Z}\neq\emptyset$, and $X=\hat{X}\neq\emptyset$ is testable if and only if there exists some $s\epsilon S$ such that the reduced mapping $F_{\hat{x}}\rfloor_{s}^{S}:X\rightarrow\hat{X}$ is one-to-one (while $F_{\hat{z}}:SxS\rightarrow\hat{Z}$ is any well-defined mapping); moreover, the number of tests required to test the array is in the range from $|X|\cdot|S|$ to $|X|\cdot(n\cdot|S|-n+1)$ where n is the number of cells in the array.

Theorem 17: Any single faulty cell in a one-dimensional array with the typical cell having no external cell input can be located with an uncertainty of 1 if and only if there exists some $s\epsilon S$ such that (1) the reduced mapping $F_{\hat{x}}\rfloor_{s}^{S}:X\rightarrow\hat{X}$ is one-to-one and (2) the response at the $\hat{x}$-output of the last cell with respect to the $|X|$ test steps, where all cells are set to the state s and each of $|X|$ possible cell input combinations is applied to the first cell step by step, is nominal; however, any single faulty cell in the array can be located with an uncertainty of 2 if there exists some $s_1\epsilon S$ such that $F_{\hat{x}}\rfloor_{s_1}^{S}$ is one-to-one and there exists some $s_2\epsilon S$ such that $F_{\hat{z}}\rfloor_{s_2}^{S}$ is one-to-one provided $|\hat{Z}|\geq|X|$.

3. Arrays With $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, and $\hat{Z}=\emptyset$

Theorem 18: Given a one-dimensional array with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, $\hat{Z}=\emptyset$, and the cell is described in terms of a mapping $F:SxXxZ\rightarrow\hat{X}$, then it is testable if and only if there exist some $s\epsilon S$ and $z\epsilon Z$ such that the reduced mapping $F\rfloor_{s}^{S}\rfloor_{z}^{Z}:X\rightarrow\hat{X}$ is one-to-one; moreover, the number of tests required to test the array is in the range from $|X|\cdot|Z|\cdot|S|$ to $|X|\cdot(n\cdot|S|\cdot|Z|-n+1)$ where n is the number of cells in the array.

Theorem 19: Given a one-dimensional array with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, $\hat{Z}=\emptyset$ and the cell is described in terms of a mapping $\Gamma:SxXxZ\rightarrow\hat{X}$, then any single faulty cell can be located with an uncertainty of 1 if and only if there exist some $s_1\epsilon S$ and $z_1\epsilon Z$ such that (1) $F\rfloor_{s_1}^{S}\rfloor_{z_1}^{Z}:X\rightarrow\hat{X}$ is one-to-one and (2) the response at the boundary output with respect to the $|X|$ test steps,

where all cells are set to the state s_1, the z-input to every cell is z_1, and each of $|X|$ possible x-input combinations is applied to the first cell step by step, is nominal.

4. Arrays With $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, and $\hat{Z}\neq\emptyset$

Theorem 20: Given a one-dimensional array with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, $\hat{Z}\neq\emptyset$, and the cell is described in terms of two mappings $F_{\hat{x}}:SxXxZ\rightarrow\hat{X}$ and $F_{\hat{z}}:SxXxZ\rightarrow\hat{Z}$, then it is testable if and only if either (1) there exist some $s_1\epsilon S$ and $z_1\epsilon Z$ such that the reduced mapping $F_{\hat{x}}\left]{}^{S}_{s_1}\right]^{Z}_{z_1}:X\rightarrow\hat{X}$ is one-to-one or (2) for any $\hat{x}\epsilon\hat{X}$, there exist some $x\epsilon X$, $s_2\epsilon S$, and $z_2\epsilon Z$ such that $F_{\hat{x}}(s_2,x,z_2)=\hat{x}$ and there exist some $s_3\epsilon S$ and $z_3\epsilon Z$, where s_3 is not necessarily distinct from s_2 and z_3 is not necessarily distinct from z_2, such that the reduced mapping $F_{\hat{z}}\left]{}^{S}_{s_3}\right]^{Z}_{z_3}:X\rightarrow\hat{Z}$ is one-to-one provided $|\hat{Z}|\geq|X|$.

Theorem 21: Given a one-dimensional array with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, $\hat{Z}\neq\emptyset$, and the cell is described in terms of two mappings $F_{\hat{x}}:SxXxZ\rightarrow\hat{X}$ and $F_{\hat{z}}:SxXxZ\rightarrow\hat{Z}$, then any single faulty cell can be located with an uncertainty of 2 if for any $\hat{x}\epsilon\hat{X}$, there exist some $s_1\epsilon S$, $x\epsilon X$, and $z_1\epsilon Z$ such that $F_{\hat{x}}(s_1,x,z_1)=\hat{x}$, $|\hat{Z}|\geq|X|$, and there exist some $s_2\epsilon S$ and $z_2\epsilon Z$, where s_2 is not necessarily distinct from s_1 and z_2 is not necessarily distinct from z_1, such that the reduced mapping $F_{\hat{z}}\left]{}^{S}_{s_2}\right]^{Z}_{z_2}:X\rightarrow\hat{Z}$ is one-to-one; however, any single faulty cell can be located with an uncertainty of 1 if and only if there exist some $s_3\epsilon S$ and $z_3\epsilon Z$ such that the reduced mapping $F_{\hat{x}}\left]{}^{S}_{s_3}\right]^{Z}_{z_3}:X\rightarrow\hat{X}$ is one-to-one and the response to the $|X|$ test steps, where all cells are set to the state s_3, the z-input to every cell is z_3, and each of $|X|$ possible x-input combinations is applied to the first cell step by step, is nominal.

C. Two-Dimensional Arrays

One way to specify the cell in an array is by the cell-input-state-output-table, abbreviated as CISO-table, listing all possible cell

input-state combinations and the corresponding cell outputs. A CISO-status is a triplet (i,s,o) where i, s, and o are the cell input, state, and the corresponding output, respectively.

1. Covering Each Cell in an Array With Every Possible Input-State Combination

Let S_f be a set of all CISO-statuses in the CISO-table T_{CISO} describing the cell in an array. Let T_{CIO} be a CIO-table carrying incomplete information about the table T_{CISO} in that the state information in the table T_{CISO} are discarded in the table T_{CIO}. Let S_I be a set of all (distinct) CIO-statuses in the table T_{CIO}. Clearly, one has the following.

Theorem 22: The Condition A is satisfied if there exists a nonempty set S_t of tessellations such that each CIO-status in the set S_I occurs on C_{ij} in at least one tessellation in the set S_t for $\forall i \epsilon S_I$ and $\forall j \epsilon S_I$.

Notice that the existence of a nonempty set S_t of tessellations such that each CIO-status in the set S_I occurs on C_{ij} in some tessellation in the set S_t for $\forall i \epsilon S_I$ and $\forall j \epsilon S_I$ is also necessary for satisfying the Condition A for the doubly-infinite array, one can conceive that this property is also necessary for satisfying the Condition A for very large (in both dimensions) finite arrays.

The procedures in obtaining tessellations in section II.C.1. are applicable to the situation here except that in calculating the number of steps required for applying all possible input-state combinations to all cells in the array, one needs to refer back to the table T_{CISO} since a CIO-status in the table T_{CIO} might represent two or more distinct CISO-statuses in the set S_f.

2. Existence of a Sensitized Path -- A Sufficiency for Satisfying the Condition B

Let the cell in a two-dimensional array be described in terms of two mapping $F_{\hat{X}}:SxXxZ\rightarrow\hat{X}$ and $F_{\hat{Z}}:SxXxZ\rightarrow\hat{Z}$, then one has the following theorem regarding the existence of a sensitized path.

Theorem 23: If the cell in a two-dimensional array is chosen such that

(1) there exists some $s_1\epsilon S$ such that the reduced mapping $F\rfloor^{S}_{s_1}:XxZ\rightarrow\hat{X}x\hat{Z}$, where $F:SxXxZ\rightarrow\hat{X}x\hat{Z}$ is defined by $F(s,x,z)=(F_{\hat{X}}(s,x,z),F_{\hat{Z}}(s,x,z))$ for $\forall s\epsilon S$, $\forall x\epsilon X$, and $\forall z\epsilon Z$, is one-to-one,

(2) there exists some $s\epsilon S$ such that the reduced mapping $F_{\hat{X}}\rfloor^{S}_{s}\rfloor^{Z}_{z}:X\rightarrow\hat{X}$ is one-to-one for $\forall z\epsilon Z$, the reduced mapping $F_{\hat{X}}\rfloor^{S}_{s}\rfloor^{X}_{x}:Z\rightarrow\hat{X}$ is one-to-one for $\forall x\epsilon X$ provided $|\hat{X}|\geq|Z|$, and $F_{\hat{Z}}:SxXxZ\rightarrow\hat{Z}$ is any (well-defined) mapping,

(3) there exists some $s\epsilon S$ such that the reduced mapping $F_{\hat{Z}}\rfloor^{S}_{s}\rfloor^{X}_{x}:Z\rightarrow\hat{Z}$ is one-to-one for $\forall x\epsilon X$, the reduced mapping $F_{\hat{Z}}\rfloor^{S}_{s}\rfloor^{Z}_{z}:X\rightarrow\hat{Z}$ is one-to-one for $\forall z\epsilon Z$ provided $|\hat{Z}|\geq|X|$, and $F_{\hat{X}}:SxXxZ\rightarrow\hat{X}$ is any (well-defined) mapping, or

(4) there exists some $s\epsilon S$ such that the reduced mapping $F_{\hat{X}}\rfloor^{S}_{s}\rfloor^{Z}_{z}:X\rightarrow\hat{X}$ is one-to-one for $\forall z\epsilon Z$ and the reduced mapping $F_{\hat{Z}}\rfloor^{S}_{s}\rfloor^{X}_{x}:Z\rightarrow\hat{Z}$ is one-to-one for $\forall x\epsilon X$,

then, the Condition B is satisfied.

Proof: Part (1)

By setting all cells in an array satisfying (1) to the state s_1, the logic of all cells behaves in a manner described in (1) of Theorem 13. The proof here, then, follows that for Part (1) of Theorem 13.

Similarly, Part (2), Part (3), and Part (4) here follow the implication of the proofs for Part (2), Part (3), and Part (4) of Theorem 13, respectively. ∎

Theorem 24: Let the cell in an array be described in terms of two mappings $F_{\hat{X}}:SxXxZ\rightarrow\hat{X}$ and $F_{\hat{Z}}:SxXxZ\rightarrow\hat{Z}$. If the cell in an array is chosen such that there exists some $s_1\epsilon S$ such that

(1) the reduced mapping $F\rceil_{s_1}^{S}:XxZ\rightarrow\hat{X}x\hat{Z}$ is one-to-one, where $F:SxXxZ\rightarrow\hat{X}x\hat{Z}$ is defined by $F(s,x,z)=(F_{\hat{X}}(s,x,z),F_{\hat{Z}}(s,x,z))$ for $\forall s\epsilon S$, $\forall x\epsilon X$, and $\forall z\epsilon Z$, or

(2) the reduced mapping $F_{\hat{X}}\rceil_{s_1}^{S}\rceil_{z}^{Z}:X\rightarrow\hat{X}$ is one-to-one for $\forall z\epsilon Z$ and the reduced mapping $F_{\hat{Z}}\rceil_{s_1}^{S}\rceil_{x}^{X}:Z\rightarrow\hat{Z}$ is one-to-one for $\forall x\epsilon X$,

then, the array is testable; moreover, the minimum number of tests required to test all possible input-state combinations on all cells in the array is $|X|\cdot|Z|\cdot|S|$.

Proof: Suppose the cell in an array is chosen such that the hypothesis is satisfied. Then, that the Condition B is satisfied follows Part (1) or Part (4) of Theorem 23. That the Condition A is satisfied follows the implication of Theorem 7 or Theorem 8 in Chapter II. Thus, the array is testable meaning that the presence of a single faulty cell in the array can be detected.

Suppose the cell in an array is chosen such that (a) the reduced mapping $F\rceil_{s}^{S}:XxZ\rightarrow\hat{X}x\hat{Z}$ is one-to-one for $\forall s\epsilon S$ or (b) the reduced mapping $F_{\hat{X}}\rceil_{s}^{S}\rceil_{z}^{Z}:X\rightarrow\hat{X}$ is one-to-one for $\forall s\epsilon S$, $\forall z\epsilon Z$ and the reduced mapping $F_{\hat{Z}}\rceil_{s}^{S}\rceil_{x}^{X}:Z\rightarrow\hat{Z}$ is one-to-one for $\forall s\epsilon S$, $\forall x\epsilon X$, then, the number of tests required to test all possible input-state combinations on all cells in the array is $|X|\cdot|Z|\cdot|S|$ which is the minimum number of tests in the array testing. ∎

Theorem 25: Let the cell in an array be described in terms of two mappings $F_{\hat{X}}: S \times X \times Z \to \hat{X}$ and $F_{\hat{Z}}: S \times X \times Z \to \hat{Z}$. If the cell in an array is chosen such that there exists some $s_1 \in S$ such that

(1) the reduced mapping $F\rceil_{s_1}^{S}: X \times Z \to \hat{X} \times \hat{Z}$ is one-to-one, where $F: S \times X \times Z \to \hat{X} \times \hat{Z}$ is defined by $F(s,x,z) = (F_{\hat{X}}(s,x,z), F_{\hat{Z}}(s,x,z))$ for $\forall s \in S$, $\forall x \in X$, and $\forall z \in Z$, or

(2) the reduced mapping $F_{\hat{X}}\rceil_{s_1}^{S}\rceil_{z}^{Z}: X \to \hat{X}$ is one-to-one for $\forall z \in Z$ and the reduced mapping $F_{\hat{Z}}\rceil_{s_1}^{S}\rceil_{x}^{X}: Z \to \hat{Z}$ is one-to-one for $\forall x \in X$

and the response to the $|X| \cdot |Z|$ tests, where every cell in the array is set to the state s_1, enabling the checking for all those CISO-statuses each having s_1 as its state-component in the set S_f on all cells in the array, is nominal, then, any faulty cell in the array can be located with an uncertainty of 1.

In fact, for any array satisfying the hypothesis in Theorem 25, all possible faults in any cell in the array are effectively those faults each introducing some erroneous cell output while the cell is in some state other than the state s_1, hence, each of multiple faulty cells in the array can be located with an uncertainty of 1.

Chapter IV

FAULT DETECTION AND LOCATION IN SECOND CATEGORY SEQUENTIAL CELLULAR ARRAYS

A. Introduction

A second category sequential array is a sequential array of cells each having some memory element as well as some logic circuit where the state of each memory element in each cell is controlled by some logic within the cell and each cell behaves as a deterministic, strongly-connected, completely-specified, finite-state, and synchronous sequential machine. In this chapter, an array shall be understood to be a second category sequential cellular array.

Again, for a cell in an array, the finite horizontal input set, the finite vertical input set, the finite nonempty state set, the finite horizontal output set, and the finite vertical output set are denoted by X, Z, S, $\hat{X}$, and $\hat{Z}$, respectively. x, z, s, $\hat{x}$, and $\hat{z}$ are arbitrary elements of X, Z, S, $\hat{X}$, and $\hat{Z}$, respectively.

B. A Special Case

In this section, the investigation is on the detection and location of a faulty cell in a synchronous sequential array of cells each with a trigger flip-flop as its memory element as shown in Figure 3. C_1 and C_2 are timing signals (pulses) enabling the cell to behave in the synchronous manner. The combinational logic within a cell, $F:SxXxZ\rightarrow\hat{X}$ where $X=Z=S=\hat{X}=\{0,1\}$, can be specified in the form of a L-table, which is actually a Karnaugh map with 8 entries in two rows each corresponding to

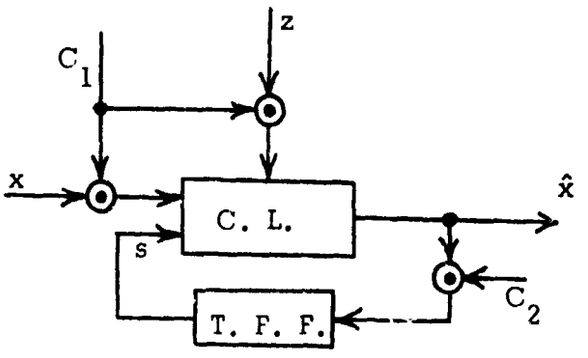


Figure 3. A cell in a synchronous sequential array of cells each with a trigger flip-flop as its memory element.

a distinct state of the memory element and four columns each corresponding to a distinct cell input combination.

Given a Boolean function $F(x_1, x_2, \ldots, x_k, \ldots, x_n): \{0,1\}^n \rightarrow \{0,1\}$, the subcomplement of $F(x_1, x_2, \ldots, x_k, \ldots, x_n)$ with respect to x_1, x_2, ..., and x_k, denoted by $F^*_{x_1, x_2, \ldots, x_k}$, is a mapping $F^*_{x_1, x_2, \ldots, x_k}(x_1, x_2, \ldots, x_k, \ldots, x_n): \{0,1\}^n \rightarrow \{0,1\}$ defined as following:

Let V_0 be a collection of 2^{n-k} vertices where $x_1=0$, $x_2=0$, ..., $x_k=0$, V_1 be a collection of 2^{n-k} vertices where $x_1=0, x_2=0, \ldots, x_{k-1}=0$, $x_k=1$, V_2 be a collection of 2^{n-k} vertices where $x_1=0, x_2=0, \ldots, x_{k-1}=1$, $x_k=0, \ldots\ldots$, and V_{2^k-1} be a collection of 2^{n-k} vertices where $x_1=1$, $x_2=1, \ldots, x_k=1$. Let $S=\{V_0, V_1, V_2, \ldots\ldots, V_{2^k-1}\}$. Partition S into two blocks S_c and S_d, i.e., $S_c \cup S_d = S$ and $S_c \cap S_d = \emptyset$, according to the criterion that $V_i \epsilon S_c$ if and only if F(v) are identical for all $v \epsilon V_i$. Then,

(1) for all i such that $V_i \epsilon S_c$ and for all $v \epsilon V_i$, $F^*_{x_1, x_2, \ldots, x_k}(v) = F(v)$, and

(2) for all j such that $V_j \epsilon S_d$ and for all $v \epsilon V_j$, $F^*_{x_1, x_2, \ldots, x_k}(v) = \bar{F}(v)$.

Theorem 26: The presence of a single faulty cell in a one-dimensional array (a horizontal linear cascade) of identical cells of the type as shown in Figure 3 can be detected if and only if

(1) for any state a cell is in, there exists some $z_1 \epsilon Z$ such that any change in x-input induces some change in the cell output ($\hat{x}$-output),

(2) the fault of the combinational logic in the cell is not the one, in effect, causing the logic to realize the subcomplement of F with respect to x and z, where F is the Boolean function specified for the combinational logic, and

(3) F is not independent of s.

Proof: Necessity

In order to provide all possible horizontal input combinations, i.e., x=0 and x=1, to a typical cell, both 0 and 1 must appear as entries in the L-table specifying the combinational logic in a cell. It is impossible to keep every cell in the array in a particular state at all time during entire test schedule, what's more, any cell output change due to a fault must be detectable at the boundary output ($\hat{x}$-output of the last cell), thus, for any $s \epsilon S$, there must exist some $z_1 \epsilon Z$ such that $F(s, x_1, z_1) \neq F(s, x_2, z_1)$ for any pair $x_1, x_2, \epsilon X$ with $x_1 \neq x_2$. Therefore, the condition (1) is necessary.

That the condition (2) is necessary can be shown by contradiction. Suppose that there are two separate cells A and B of the type as shown in Figure 3 both satisfying the conditions (1) and (3) but the logic of cell A is realizing F and the logic of cell B is realizing $F^*_{x,z}$. Suppose the input combination of $x=a \wedge z=b$ for some a $b \epsilon \{0,1\}$ is applied to the two separate cells A and B simultaneously, if the two entries under the column $x=a \wedge z=b$ in the L-table specifying F are same, say each is a 0, then the corresponding two entries in the L-table specifying $F^*_{x,z}$ are same, namely

each is a 0, thus the output of cell A is same as the output of cell B in responding to the input combination of $x=a\Lambda z=b$ regardless of the states of these two cells, if the two entries under the column $x=a\Lambda z=b$ in the L-table specifying F are different, say the entry corresponding to $s=0\Lambda x=a\Lambda z=b$ is a 1 and the entry corresponding to $s=1\Lambda x=a\Lambda z=b$ is a 0, then the corresponding two entries in the L-table specifying $F^*_{x,z}$ are different, namely the entry corresponding to $s=0\Lambda x=a\Lambda z=b$ is a 0 and the entry corresponding to $s=1\Lambda x=a\Lambda z=b$ is a 1, thus the output of cell A at state $c\epsilon S=\{0,1\}$ is same as the output of cell B at state $\bar{c}$ in responding to the same input condition; furthermore, if the next state of cell A remains same as its present state, then the next state of cell B remains same as its present state, if cell A changes its state, then cell B changes its state. Therefore, the sequence of the output of cell A initially at state c is same as that of cell B initially at state $\bar{c}$ with response to the same input sequence. Hence, if the conditions (1) and (3) are satisfied but the condition (2) is not met, then the occurrence of the fault in a cell of the array, in effect, causing the logic in the cell to realize $F^*_{x,z}$ instead of the specified F, can not be detected. This shows that the condition (2) is necessary.

That the condition (3) is necessary is obvious: if F is independent of the state of the cell, then whether the memory element is functioning properly or not has no effect on the input-output behavior of a cell, thus the fault in the memory element of a cell can not be detected.

Sufficiency

The proof is by construction, namely, it is to be shown that if a one-dimensional array of identical cells of the type as shown in Figure 3 satisfied the conditions (1), (2), and (3), then a test schedule can be derived which enables the detection of the presence of a faulty cell in the array. Three cases are considered.

Case 1. The conditions (1), (2), and (3) are satisfied where there exists some $z_1 \epsilon Z$ such that any change in x-input induces some change in the cell output and the two entries in the L-table under each of the two columns corresponding to $z=z_1$ are same, i.e., $F(0,0,z_1)=F(1,0,z_1)$, $F(0,1,z_1)=F(1,1,z_1)$, and $F(0,0,z_1) \neq F(0,1,z_1)$ for some $z_1 \epsilon Z$.

Without loss of generality, suppose the two entries under the column $x=x_1 \Lambda z=z_1$ are 0's for some $x_1 \epsilon \{0,1\}$, then the two entries under the column $x=\bar{x}_1 \Lambda z=z_1$ are 1's. Then, the essential part of the test schedule is applying z_1 to each z-input of all cells in the cascade and the input sequence of $x_1\text{-}\bar{x}_1\text{-}x_1\text{-}\bar{x}_1$ to the x-input of the first cell in the cascade. It is easy to verify that if none of cells in the cascade is stuck at any state and the sequence of the boundary output in responding to the essential part of the test schedule is correct, then the four entries in the L-table under the two columns corresponding to $z=z_1$ are known to have been checked for every cell in the cascade. The fact that the boundary output sequence in responding to the essential part of the test schedule is correct ensures the existence of a sensitizing path by virtue of applying z_1 to each z-input of the second to the last cell while testing the first cell and thus the subsequent part of the test schedule can be subsequently proceeded. If the boundary output sequence in responding to the essential part of the test schedule is not correct, then, the presence of a faulty cell is detected and there is no need to proceed to the subsequent part of the test schedule. The subsequent part of the test schedule begins with applying z_1 to each z-input of the second to the last cell in the cascade and appropriate sequence of input combination to the first cell to complete the test on the first cell since the output of the first can be determined by observing the boundary output and tracing backward along the sensitized

path to the output of the first cell. The fault, if any, in the first cell can thus be detected. Depending upon the actual specification of the combinational logic within a cell, it requires four to six test steps to check the four entries under the two columns corresponding to $z=\bar{z}_1$ for a cell. For instance, suppose that the two entries under the column $x=x_2 \Lambda z=\bar{z}_1$, for some $x_2 \epsilon \{0,1\}$, are distinct and the 0-entry corresponds to $s=s_1 \Lambda x=x_2 \Lambda z=z_1$ for some $s_1 \epsilon \{0,1\}$, furthermore, the two entries under the column $x=\bar{x}_2 \Lambda z=\bar{z}_1$ are both 1's. Then, if the first cell is fault-free and is in state $\bar{s}_1$, then the output sequence of the first cell should be 1-0-1-1 in responding to the input sequence of $(x=\bar{x}_2 \Lambda z=\bar{z}_1)-(x=x_2 \Lambda z=\bar{z}_1)-(x=\bar{x}_2 \Lambda z=\bar{z}_1)-(x=x_2 \Lambda z=\bar{z}_1)$. If the first cell is fault-free and is in state s_1, then its output sequence should be 1-1-0-1-1 in responding to the input sequence of $(x=\bar{x}_2 \Lambda z=\bar{z}_1)-(x=x_2 \Lambda z=\bar{z}_1)-(x=x_2 \Lambda z=\bar{z}_1)-(x=\bar{x}_1 \Lambda z=\bar{z}_1)-(x=\bar{x}_2 \Lambda z=\bar{z}_1)$. With this in mind, then if the output of the first cell is 0 in responding to the test step of $(x=x_2 \Lambda z=\bar{z}_1)$, then the first cell is faulty, otherwise, proceed the second test step of $(x=x_2 \Lambda z=\bar{z}_1)$ and the third test step onward should be depending upon its output in responding to the second test step. Note that any fault in the combinational logic excluded by the condition (2) or any fault in the memory element of the first cell should result an output sequence at the first cell other than 1-0-1-1 in responding to the input sequence of $(x=\bar{x}_2 \Lambda z=\bar{z}_1)-(x=x_2 \Lambda z=\bar{z}_1)-(x=\bar{x}_2 \Lambda z=\bar{z}_1)-(x=x_2 \Lambda z=\bar{z}_1)$ and 1-1-0-1-1 in responding to $(x=\bar{x}_2 \Lambda z=\bar{z}_1)-(x=x_2 \Lambda z=\bar{z}_1)-(x=x_2 \Lambda z=\bar{z}_1)-(x=\bar{x}_1 \Lambda z=z_1)-(x=\bar{x}_2 \Lambda z=\bar{z}_1)$. After the first cell has been checked to be fault-free, the test on the second cell can be proceeded, similarly, by applying z_1 to the z-inputs of all cells but the second cell in the cascade and appropriate sequence of x-input to the first cell and z-input to the second cell. A similar test procedure is then used to test the remaining cells in the cascade if necessary. Therefore, with a test schedule of at most 6n+4 test steps, where n is the number of cells in

the cascade, the presence or absence of a faulty cell in the cascade can be detected.

Case 2. The conditions (1), (2), and (3) are satisfied, the additional specification stated in Case 1 is not met, and there exists some $z_1 \in Z$ such that any change in x-input induces some change in the cell output and the two entries in the L-table under each of the two columns corresponding to $z=z_1$ are distinct, i.e., $F(0,0,z_1) \neq F(1,0,z_1)$, $F(0,1,z_1) \neq F(1,1,z_1)$, $F(0,0,z_1) \neq F(0,1,z_1)$, and $F(1,0,z_1) \neq F(1,1,z_1)$ for some $z_1 \in \{0,1\}$.

Depending on the actual specification in terms of the L-table, the next state of a cell can be set to the state 1 or reset to the state 0 with an input combination of $x=0 \wedge z=z_1$ (or $x=1 \wedge z=z_1$) regardless the present state of the cell. Thus, a cascade of n cells under this case can be initialized to some state with some sequence of n steps of inputs.

The essential part of the test schedule is applying z_1 to each z-input of all n cells in the cascade and appropriate sequence of n steps of x-input to the first cell to initialize the cascade to some state followed by appropriate sequence of x-input to the first cell to check, for each of all cells in the cascade, all four entries in the L-table corresponding to $z=z_1$. If the response (the sequence of $\hat{x}$-output at the last cell) to this essential part of the test schedule is nominal, i.e., unperturbed, then by virtue of applying z_1 to each z-input of the sensitized path, while completing the test on the first cell, is assured. To see this, one can visualize that if a fault within a cell results in some change in its input-output behavior, it might also result in some false state initialization on that cell and the following cells. Now, suppose that the specification for the combinational logic within a cell in this case is F and that the response to some test sequence for checking the four entries corresponding to $z=z_1$,

say the horizontal input sequence of $I_{nominal}$ with z-input fixed at z_1, while the state of the cell is initially at s_0, is $O_{nominal}$. With z-input fixed at z_1, the fault in the logic that can produce the response of $O_{nominal}$ in responding to the horizontal input sequence of $I_{nominal}$ is the one, in effect, causing the logic to realize some Boolean function F_f which is consistent with $F^*_{x,z}$ at least at the four entries corresponding to $z=z_1$, while the state of the cell is incorrectly initialized at $\bar{s}_0$. This situation does not prevent the cell from correctly responding to any change in x-input as long as the z-input to this cell is z_1. If the logic of the cell is correct but its state is incorrectly initialized at $\bar{s}_0$ due to a fault in one of the preceeding cells, then, with z-input fixed at z_1, the only horizontal input sequence I which can produce the response of $O_{nominal}$ is $I=\bar{I}_{nominal}$, where $\bar{I}_{nominal}$ is obtained by complementing each step of the $I_{nominal}$. One can see that no fault in the combinational logic within the cell can produce the response of $\bar{O}_{nominal}$ in responding to the horizontal input sequence $I_{nominal}$ with z-input fixed at z_1. Without loss of generality, suppose the entry corresponding to $s=0\Lambda x=x_1\Lambda z=z_1$ in the L-table describing F is a 0 for some $x_1\epsilon\{0,1\}$, then, the entry corresponding to $s=1\Lambda x=x_1\Lambda z=z_1$ is a 1, the entry corresponding to $s=0\Lambda x=\bar{x}_1\Lambda z=z_1$ is a 1, and the entry corresponding to $s=1\Lambda x=\bar{x}_1\Lambda z=z_1$ is a 0. In any test sequence to the cell to check the four entries corresponding to $z=z_1$, there must exist two consecutive steps $(s=1\Lambda x=x_1\Lambda z=z_1)$-$(s=0\Lambda x=x_1\Lambda z=z_1)$ or $(s=0\Lambda x=\bar{x}_1\Lambda z=z_1)$-$(s=1\Lambda x=\bar{x}_1\Lambda z=z_1)$. With z-input fixed at z_1, if there were some fault in the logic of the cell causing the response of $\bar{O}_{nominal}$, instead of $O_{nominal}$ should the cell be fault-free, in responding to the horizontal input sequence of $I_{nominal}$ to the cell, then the fault either would have been rendering an output of 0 while the state is at $s_1\epsilon\{0,1\}$ with the input combination at $x=x_1\Lambda z=z_1$ and rendering an output of 1 while the state is also at s_1 with the input combination also at $x=x_1\Lambda z=z_1$,

which is a contradiction, or would have been rendering an output of 0 while the state is at $s_1 \epsilon \{0,1\}$ with the input combination at $x=\bar{\bar{x}}_1 \Lambda z=z_1$, which is also a contradiction.

If the response to the essential part of the test schedule is incorrect, then the presence of a faulty cell is detected and there is no need to use the subsequent part of the test schedule.

The fact that the response to the essential part of the test schedule is nominal enables one to use the subsequent part of the test schedule which is the part to test the four entries corresponding to $z=\bar{z}_1$ for the first to the last cell in the cascade one by one. By applying z_1 to each z-input of the second to the last cell and appropriate sequence of input combination to the first cell, the first cell can thus be tested completely. The only fault in the first cell which might produce the nominal response at its output is the one, in effect, causing the cell to realize $F^*_{x,z}$ instead of F. But this is excluded by the condition (2). After the first cell has been checked to be fault-free by applying z_1 to all z-inputs except the z-input of the second cell, $\bar{z}_1$ to the z-input of the second cell, and appropriate horizontal input sequence to the first cell, the test on the second cell can thus be completed. In a similar manner, all cells in the cascade can be tested in succession.

Case 3. The conditions (1), (2), and (3) are satisfied but neither the constraint in case 1 nor the constraint in case 2 is satisfied.

Under this case, there exists precisely one $z_1 \epsilon \{0,1\}$ such that $F(0,0,z_1) \neq F(0,1,z_1)$, $F(1,0,z_1)=F(1,1,z_1)$, $F(1,0,\bar{z}_1) \neq F(1,1,\bar{z}_1)$, and $F(0,0,\bar{z}_1)=F(0,1,\bar{z}_1)$ and there exists some column, say corresponding to $x=x_1 \Lambda z=z_2$ where $x_1, z_2 \epsilon \{0,1\}$, in the L-table such that the two entries under that column are distinct.

Since there exists some column in the L-table such that the two

entries under the column are distinct, a cascade of n cells under this case can be initialized to some state with some sequence of n steps of input combination.

The essential part of the test schedule is applying an appropriate sequence of n steps of input combination to all n cells in the cascade to initialize the cascade to some state, say $s_0^1 s_0^2 s_0^3 \ldots . s_0^n$, where s_0^1, s_0^2, s_0^3, $\ldots . s_0^n \epsilon \{0,1\}$, followed by appropriate sequence of input combinations to check, for each of all cells in the cascade, the four entries in the L-table corresponding to $s=0 \Lambda x=0 \Lambda z=z_1$, $s=0 \Lambda x=1 \Lambda z=z_1$, $s=1 \Lambda x=0 \Lambda z=\bar{z}_1$, and $s=1 \Lambda x=1 \Lambda z=\bar{z}_1$. If the response to the essential part of the test schedule is nominal, then, by virtue of applying appropriate z-input to each of the second to the last cell, the existence of a sensitized path, while completing the test on the first cell, is assured. By applying appropriate z-input to a cell here, it is meant that z_1 is applied to z-input of the cell while its state is supposedly at 0 and $\bar{z}_1$ is applied to the z-input of the cell while its state is supposedly at 1. In fact, the fault in the logic within the cell that can produce the nominal response with respect to the input sequence $I_{nominal}$ supposed to check the four entries the essential part of test schedule is designed to check is the one, in effect, causing the logic to realize some Boolean function F_f which is consistent with $F^*_{x,z}$ at least at the four entries corresponding to $s=0 \Lambda x=0 \Lambda z=z_1$, $s=0 \Lambda x=1 \Lambda z=z_1$, $s=1 \Lambda x=0 \Lambda z=\bar{z}_1$, and $s=1 \Lambda x=1 \Lambda z=\bar{z}_1$, while the state of the cell is incorrectly initialized at $\bar{s}_0$ where s_0 is the supposed initial state of the cell. This situation does not prevent the cell from correctly responding to any change in x-input as long as the appropriate z-input is applied to this cell. As to the situation where the logic of the cell is correct but its state is incorrectly initialized at $\bar{s}_0$ due to a fault in one of the preceeding cells, as long as appropriate z-input is applied to the cell at each test step, no horizontal input sequence can produce the

response of $O_{nominal}$, where $O_{nominal}$ is the nominal response with respect to $I_{nominal}$ while the initial state is s_0.

If the response to the essential part of the test schedule is incorrect, then, the presence of a faulty cell is detected and there is no need to proceed with the subsequent part of the test schedule.

The fact that the response to the essential part of the test schedule is nominal enables one to proceed to the subsequent part of the test schedule which is the part to test the other four entries corresponding to $s=0\Lambda x=0\Lambda z=\bar{z}_1$, $s=0\Lambda x=1\Lambda z=\bar{z}_1$, $s=1\Lambda x=0\Lambda z=z_1$ and $s=1\Lambda x=1\Lambda z=z_1$ for the first to the last cell in the cascade in succession. The only fault in any cell in the cascade which might produce the nominal response at its output is the one, in effect, causing the cell to realize $F^*_{x,z}$ instead of F. But this is excluded by the condition (2). Therefore, the presence or absence of a faulty cell in cascade can be detected. ∎

Theorem 27: A faulty cell in a one-dimensional array of identical cells of the type as shown in Figure 3 can be located with an uncertainty of 1 if and only if

(1) the array is testable, i.e., the conditions (1), (2), and (3) stated in Theorem 26 are satisfied, and

(2) the response to the essential part of the test schedule described in the proof for Theorem 26 is nominal.

Proof: Necessity

That an array is testable is essential to the location of a faulty cell, hence, the condition (1) here is necessary.

If the array is testable but the response to the essential part of the test schedule described in the proof for Theorem 26 is not nominal for

each of all possible alternatives[4], then, as one can see while following the proof for Theorem 26, the presence of a faulty cell in the cascade can thus be detected but there exists no proper sensitized path, the faulty cell could be at anywhere from the first to the last cell position. Thus, the location of the faulty cell with an uncertainty of 1 is impossible.

Sufficiency

Following the Sufficiency part of the Proof for Theorem 26, it is easy to see that the addition of the condition (2) here to the conditions (1), (2) and (3) stated in Theorem 26 essentially enables the complete test on the first cell to the last cell in the cascade in sucession. Therefore, a faulty cell in the cascade can be located with an uncertainty of 1.

C. General Conditions for Testability

Two necessary conditions that must be satisfied for array testing for the detection of the presence of a faulty cell in a second category sequential array are the Condition A* and the Condition B* as follows.

Condition A*: Some test sequence to test a cell completely must be applicable to every cell in the array.

Condition B*: The sequence of outputs of any cell in the array with response to any test sequence must be (correctly) reconstructable from some sequence of boundary outputs, i.e., it can be uniquely determined by observing a sequence of some boundary outputs of some length.

---

[4] For an example, it is possible that $F(0,0,z_1)=F(1,0,z_1)$, $F(0,1,z_1)=F(1,1,z_1)$, and $F(0,0,z_1)\neq F(0,1,z_1)$ for $z_1 \epsilon Z$ also $F(0,0,z_2)=F(1,0,z_2)$, $F(0,1,z_2)=F(1,1,z_2)$ and $F(0,0,z_2)\neq F(0,1,z_2)$ for $z_2 \epsilon Z$ where $z_2 \neq z_1$.

These two conditions together are sufficient for the detection of the presence of single faulty cell in an array.

It is further assumed in the remainder of this Chapter that some test sequence for a cell, with which it can be determined whether or not the cell is faulty, is known [5], [6], [9], [15]. Furthermore, it is assumed that each cell in an array is a reduced machine and despite the presence of a faulty cell in an array, all cells except the faulty one can be correctly initialized to a unique initial state.

A machine is said to be information lossless (IL) if the initial state, the final state, and the response to an unknown input sequence are sufficient to uniquely determine the unknown input sequence. A machine M is defined to be IL of finite order r if M is IL and the initial state, the current and last r output symbols are sufficient to uniquely determine the rth past input symbol. r is the smallest integer which satisfies this definition. Procedures for testing whether or not a machine is IL or IL of finite order are given in Huffman [7] and Even [4].

If a finite-state machine M with finite nonempty state set S is IL of finite order r, then the obvious upper bound for r is $|S|C_2$. Given a deterministic, reduced, strongly-connected, completely-specified, and finite-state machine M with finite nonempty state set S, finite nonempty input set I, finite nonempty output set O, and $|I|=|O|$, if M is IL of finite order r, then a lower upper bound for r can be given in the form of a conjecture as follows: $r \leq \lfloor \log_2 |O| \rfloor$ where $\lfloor a \rfloor$ denotes the largest integer not greater than a.

For a machine $M=(I,O,S,\delta,\beta)$ where I, O, S, δ, and β are finite nonempty input set, finite nonempty output set, finite nonempty state set, next-state mapping, and output mapping, respectively, specifically,

$\delta:SxI\to S$ and $\beta:SxI\to\mathcal{C}$, $\underline{I}$ is used to denote a sequence of symbols over the set I of some length, e.g., $\underline{I}=i_1 i_2 i_3 \ldots i_k$, where $i_1, i_2, i_3, \ldots, i_k \epsilon I$, is an input sequence of length k, the length of the sequence $\underline{I}$ is denoted by $lg(\underline{I})$. The extension of δ over input sequences is denoted by $\hat{\delta}$, e.g., $\hat{\delta}(s,\underline{I})$ is the state M enters when starting in state s and applying the input sequence $\underline{I}$. The state sequence formed is denoted by $\underline{\delta}$, e.g., $\underline{\delta}(s,\underline{X})$ denotes $s_1 s_2 s_3 \ldots s_k$ where $\underline{X}=x_1 x_2 x_3 \ldots x_k$, $s_0=s$, $s_k=\hat{\delta}(s,\underline{X})$, and $s_i=\delta(s_{i-1},x_i)$ for $\forall i\epsilon\{1,2,3,\ldots,k\}$. The similar notation is used for $\underline{O}$ and $\underline{\beta}$.

D. One-Dimensional Arrays

1. Arrays with Autonomous Cells, Where $Z=\hat{Z}=\emptyset$ and $X=\hat{X}\neq\emptyset$

Given the next-state mapping and the output mapping for a cell as $\delta:SxX\to S$ and $\beta:SxX\to\hat{X}$, respectively, the cell is said to be x/$\hat{x}$-information lossless (x/$\hat{x}$-IL) if and only if for each pair $(s,\underline{X})$ there does not exist a $\underline{X}^d\neq\underline{X}$ such that $\hat{\delta}(s,\underline{X}^d)=\hat{\delta}(s,\underline{X})$ and $\underline{\beta}(s,\underline{X}^d)=\underline{\beta}(s,\underline{X})$. A procedure given in Breuer [3] can be used for testing whether or not the cell is x/$\hat{x}$-IL or x/$\hat{x}$-IL of finite order. An oriented graph G, called a testing graph, which consists of a set of nodes and a set of branches, is to be constructed. Each node identifies a pair of states (s_i,s_j) called x-compatible state pair. States s_i and s_j form a x-compatible state pair (s_i,s_j) if either

P1: there exist some $s\epsilon S$, $x,x^d\epsilon X$, where $x^d\neq x$, such that $\delta(s,x)=s_i$, $\delta(s,x^d)=s_j$, and $\beta(s,x)=\beta(s,x^d)$, or

P2: (s_k,s_l) is a x-compatible state pair and there exist some $x,x^d\epsilon X$, where x^d is not necessarily distinct from x, such that $\delta(s_k,x)=s_i$, $\delta(s_l,x^d)=s_j$, and $\beta(s_k,x)=\beta(s_l,x^d)$.

There exists a branch from node (s_k, s_l) to node (s_i, s_j) in G if and only if these two nodes satisfy P2. Then,

R1: the cell is $x/\hat{x}$-IL if and only if G contains no node of the form (s_i, s_i) and

R2: if the cell is $x/\hat{x}$-IL, then it is $x/\hat{x}$-IL of finite order r if and only if G is loop free and the length of the longest path in G has r nodes.

Theorem 28: A one-dimensional autonomous array is testable if the cell is chosen such that the reduced mapping $\beta\rceil_s^S: X \to \hat{X}$ is one-to-one (and/or onto) for $\forall s \in S$.

Proof: If the cell of a one-dimensional autonomous array is chosen such that the reduced mapping $\beta\rceil_s^S: X \to \hat{X}$ is one-to-one for $\forall s \in S$, then $\beta\rceil_s^S: X \to \hat{X}$ is also onto for $\forall s \in S$, what's more, the cell is $x/\hat{x}$-IL of zero order, or equivalently, any change in the cell input induces some change in the cell output at any instance (meaning for $\forall s \in S$). The procedure of testing the array, say of N cells C_1, C_2, C_3, ..., and C_N chosen as above, is as follows, first for i=1, then for i=2, i=3, ..., and finally i=N.

(1) Initialize C_1, C_2, C_3, ..., and C_N to the unique initial state.

(2) Apply input sequence $\underline{X}_1$ to C_1 such that the input sequence to C_i is $\underline{X}^t$ which is a test sequence to test a cell completely.

(3) Reconstruct the response of C_i in responding to the test sequence $\underline{X}^t$ by observing the boundary output sequence (i.e., the output sequence of C_N here).

Suppose C_i is faulty by the assumption of the presence of single faulty cell, C_j must be operating correctly for all $j \in \{1, 2, \ldots, i-1, i+1, \ldots, N\}$. Hence, any x-input is correctly applicable to C_i at any instance meaning that any test sequence is correctly applicable to C_i; furthermore, the

output sequence of C_i is correctly reconstructable from the boundary output sequence. Then, the perturbed output sequence of C_i in responding to the test sequence to C_i reveals that C_i is in error. ∎

Note that it is impossible to locate a faulty cell for any array here.

2. Arrays With the Typical Cell Having No External Cell Input, Where $Z=\emptyset$, $\hat{Z}\neq\emptyset$, and $X=\hat{X}\neq\emptyset$

Given the next-state mapping and the output mapping for a cell as $\delta:SxX\rightarrow S$ and $\beta:SxX\rightarrow\hat{Z}x\hat{X}$, respectively, let $\beta_{\hat{z}}:SxX\rightarrow\hat{Z}$ and $\beta_{\hat{x}}:SxX\rightarrow\hat{X}$ such that $\beta=\beta_{\hat{z}}x\beta_{\hat{x}}$, then, the cell is said to be $x/\hat{z}\hat{x}$-information lossless ($x/\hat{z}\hat{x}$-IL) if and only if for each pair $(s,\underline{X})$ there does not exist a $\underline{X}^d\neq\underline{X}$ such that $\hat{\delta}(s,\underline{X}^d)=\hat{\delta}(s,\underline{X})$ and $\underline{\beta}(s,\underline{X}^d)=\underline{\beta}(s,\underline{X})$. The cell is said to be $x/\hat{z}$-information lossless ($x/\hat{z}$-IL) if and only if for each pair $(s,\underline{X})$ there does not exist a $\underline{X}^d\neq\underline{X}$ such that $\hat{\delta}(s,\underline{X}^d)=\hat{\delta}(s,\underline{X})$ and $\underline{\beta}_{\hat{z}}(s,\underline{X}^d)=\underline{\beta}_{\hat{z}}(s,\underline{X})$. To test for $x/\hat{z}\hat{x}$-IL or $x/\hat{z}\hat{x}$-IL of finite order, the procedure in Section IV.D.1. is applied, and the appearances of $x/\hat{x}$-IL in R1 and R2 are replaced by $x/\hat{z}\hat{x}$-IL. To test for $x/\hat{z}$-IL or $x/\hat{z}$-IL of finite order, the same procedure is applied except all appearances of β are replaced by $\beta_{\hat{z}}$, and the appearances of $x/\hat{x}$-IL in R1 and R2 are replaced by $x/\hat{z}$-IL.

Theorem 29: A one-dimensional array with $Z=\emptyset$, $\hat{Z}\neq\emptyset$, and $X=\hat{X}\neq\emptyset$ is testable if the cell is chosen such that the reduced mapping $\beta_{\hat{x}}\rfloor_s^{\neg S}:X\rightarrow\hat{X}$ is one-to-one for $\forall s\in S$.

Proof: The proof is similar to that for Theorem 28. Note that part of the response of a cell, namely the output sequence at $\hat{z}$-output, in responding to a test sequence is directly observable. ∎

Theorem 30: If some test sequence is applicable to every cell in a horizontal cascade of identical cells with $Z=\emptyset$, $\hat{Z}\neq\emptyset$, and $X=\hat{X}\neq\emptyset$, then the

cascade of cells is testable if the cell is either $x/\hat{z}\hat{x}$-IL of finite order or $x/\hat{z}$-IL of finite order.

Proof: The procedure of testing the cascade of N cells C_1, C_2, C_3, ..., and C_N chosen as stated in the hypothesis is as follows, first for i=1, then for i=2, i=3, ..., and finally for i=N.

(1) Initialize C_1, C_2, C_3, ..., and C_N to the unique initial state.

(2) Apply input sequence $\underline{X}_i$ to C_1 such that the input sequence to C_i is $\underline{X}^t$ which is a test sequence to test a cell.

(3) Next apply to C_1 any known additional input sequence $\underline{X}_i^a$ of length L=(N-i)r if the cell is $x/\hat{z}\hat{x}$-IL of finite order r, or of length L=r if the cell is $x/\hat{z}$-IL of finite order r.

(4) Reconstruct the part of response, namely, sequence of $\hat{x}$-output, of C_i in responding to $\underline{X}^t$ by observing the boundary output sequence of C_{i+1} (at $\hat{z}$-output) if the cell is $x/\hat{z}$-IL of finite order or by observing the boundary output sequences of C_{i+1}, C_{i+2}, ..., and C_N if the cell is $x/\hat{z}\hat{x}$-IL of finite order in responding to the input sequence of $\underline{X}_i^t\underline{X}_i^a$ to C_1. The output sequence at $\hat{z}$-output of C_i in responding to $\underline{X}^t$ is directly observable.

The presence of a single faulty cell in the cascade can thus be detected. ∎

Following the Proof for Theorem 30, one has the following.

Theorem 31: If some test sequence is applicable to every cell in a horizontal cascade of identical cells with $Z=\emptyset$, $\hat{Z}\neq\emptyset$, and $X=\hat{X}\neq\emptyset$, then, a single faulty cell in the cascade can be located with an uncertainty of 2 if the cell is $x/\hat{z}$-IL of finite order.

3. Arrays With $X=\hat{X}\neq\emptyset$ $Z\neq\emptyset$, and $\hat{Z}=\emptyset$

Given the next-state mapping and the output mapping for a cell as

δ:SxXxZ→S and β:SxXxZ→$\hat{X}$, respectively, in a horizontal cascade where the z-input to each cell is a boundary input, then, the cell is said to be x/$\hat{x}$-IL if and only if for each triplet $(s,\underline{X},\underline{Z})$ there does not exist a $\underline{X}^d \neq \underline{X}$ such that $\hat{\delta}(s,\underline{X}^d,\underline{Z})=\hat{\delta}(s,\underline{X},\underline{Z})$ and $\underline{\beta}(s,\underline{X}^d,\underline{Z})=\underline{\beta}(s,\underline{X},\underline{Z})$. The procedure for testing whether or not the cell is x/$\hat{x}$-IL or x/$\hat{x}$-IL of finite order is same as that in Section IV.D.1. except the arguments for δ and β are adjusted accordingly.

Theorem 32: If the cell of a one-dimensional array with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, and $\hat{Z}=\emptyset$ is chosen such that the reduced mapping $\beta\rceil_s^S$:XxZ→$\hat{X}$ is onto for $\forall s\in S$, then, any test sequence is applicable to every cell in the array.

Theorem 33: If some test sequence is applicable to every cell in a horizontal cascade of identical cells with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, $\hat{Z}=\emptyset$ and the cell is x/$\hat{x}$-IL of finite order, then, the cascade is testable.

Proof: The proof is similar to that for Theorem 30 under the situation the cell there is x/$\hat{z}\hat{x}$-IL of finite order except (2), (3), and (4) should be as follows.

(2) Apply boundary input sequences $(\underline{X}_i,\underline{Z}_i^1)$ to C_1, $\underline{Z}_i^2$ to C_2, $\underline{Z}_i^3$ to C_3, ..., and $\underline{Z}_i^N$ to C_N such that the input sequence to C_i is $(\underline{X}^t,\underline{Z}^t)$ which is a test sequence to test a cell completely.

(3) Next apply any known additional boundary input sequences $(\underline{X}_i^a,\underline{Z}_i^{1a})$ to C_1, $\underline{Z}_i^{2a}$ to C_2, ..., and $\underline{Z}_i^{Na}$ to C_N each of length L=(N-i)r if the cell is x/$\hat{x}$-IL of finite order r.

(4) Reconstruct the response of C_i in responding to the test sequence $(\underline{X}^t,\underline{Z}^t)$ by observing the boundary output sequence at C_N of length $lg(\underline{X}^t)+(N-i)r$ in responding to the boundary input sequences $(\underline{\lambda}\underline{X}_i^a,\underline{Z}_i^1\underline{Z}_i^{1a})$ to C_1, $\underline{Z}_i^2\underline{Z}_i^{2a}$ to C_2, ..., and $\underline{Z}_i^N\underline{Z}_i^{Na}$ to C_N. ∎

The status of a cell at one particular instance can be denoted by a triplet (s_1, x_1, z_1) where s_1, x_1, z_1 are the state, the input at the x-input, the input at the z-input, respectively, of the cell at that particular instance. Define the set of potentially information lossy statuses[5] as $S_L = \{(s,x,z) \mid s \in S, x \in X, z \in Z \text{ and } \exists\, x^d \in X \text{ with } x^d \neq x \text{ satisfying } \beta(s,x,z) = \beta(s,x^d,z)\}$. The following Theorem is then obvious.

Theorem 34: If some test sequence is applicable to every cell in a horizontal cascade of identical cells C_1, C_2, ..., and C_N with $X = \hat{X} \neq \emptyset$, $Z \neq \emptyset$ and $\hat{Z} = \emptyset$ but the cell is not $x/\hat{x}$-IL, the presence of an error in C_i is still detectable by observing the boundary output sequence if none of C_{i+1}, C_{i+2}, ..., and C_N is in a status in the set S_L during the testing for C_i.

4. Arrays With $X = \hat{X} \neq \emptyset$, $Z \neq \emptyset$, and $\hat{Z} \neq \emptyset$

Given the next-state mapping and the output mapping for a cell as $\delta: S \times X \times Z \to S$ and $\beta: S \times X \times Z \to \hat{Z} \times \hat{X}$, respectively, in a horizontal cascade where the z-input to each cell in the cascade is a boundary input, let $\beta_{\hat{x}}: S \times X \times Z \to \hat{Z}$ and $\beta_{\hat{z}}: S \times X \times Z \to \hat{X}$ such that $\beta = \beta_{\hat{z}} \times \beta_{\hat{x}}$, then, the cell is said to be $\underline{x/\hat{z}\hat{x}\text{-IL}}$ if and only if for each triplet $(s, \underline{X}, \underline{Z})$ there does not exist a $\underline{X}^d \neq \underline{X}$ such that $\hat{\delta}(s, \underline{X}^d, \underline{Z}) = \hat{\delta}(s, \underline{X}, \underline{Z})$ and $\underline{\beta}(s, \underline{X}^d, \underline{Z}) = \underline{\beta}(s, \underline{X}, \underline{Z})$. The definition for $x/\hat{z}$-IL is similar to that for $x/\hat{z}\hat{x}$-IL except the appearances of $\underline{\beta}$ are replaced by $\underline{\beta}_{\hat{z}}$. The procedure for testing whether or not the cell is $x/\hat{z}\hat{x}$-IL, $x/\hat{z}\hat{x}$-IL of finite order, $x/\hat{z}$-IL, or $x/\hat{z}$-IL of finite order is similar to that in Section IV.D.1. where the adjustment on the arguments

---

[5] A remark with regard to the special case in Section IV.B. can be made here. It is not assumed there that all cells can be correctly initialized to a unique initial state. The cell in a horizontal cascade of cells satisfying the hypothesis in Theorem 26 might not be $x/\hat{x}$-IL, but a particular input to the z-input of a cell could prevent the cell from entering into any potentially information lossy status, thus, render the detection of a faulty cell in the cascade possible.

for δ, β, and $\beta_{\hat{z}}$ is required.

Theorem 35: If the cell of a one-dimensional array with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, and $\hat{Z}\neq\emptyset$ is chosen such that the reduced mapping $\beta_{\hat{x}}\rceil_{s}^{S}:XxZ\rightarrow\hat{X}$ is one-to for $\forall s\in S$, then, any test sequence is applicable to every cell in the array.

A good understanding of the proof for Theorem 30 and that for Theorem 33 enables one to justify the following theorem.

Theorem 36: If some test sequence is applicable to every cell in a horizontal cascade of identical cells with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, $\hat{Z}\neq\emptyset$, and the cell is either $x/\hat{z}$-IL of finite order or $x/\hat{z}\hat{x}$-IL of finite order, then, the cascade is testable.

Define $T_L=\{(s,x,z)\mid s\in S, x\in X, z\in Z$ and $\exists\, x^d\in X$ with $x^d\neq x$ satisfying $\beta_{\hat{z}}(s,x,z)=\beta_{\hat{z}}(s,x^d,z)\}$.

The following theorem is obvious.

Theorem 37: If some test sequence is applicable to every cell in a horizontal cascade of identical cells $C_1, C_2, \ldots, C_N$ with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, and $\hat{Z}\neq\emptyset$ but the cell is neither $x/\hat{z}$-IL nor $x/\hat{z}\hat{x}$-IL, the presence of an error in C_i is still detectable by observing the boundary output sequences at C_i, $C_{i+1}, \ldots, C_N$ if none of $C_{i+1}, C_{i+2}, \ldots,$ and C_N is in a status in the set S_L or the set T_L during the testing for C_i.

Theorem 38: If some test sequence is applicable to every cell in a horizontal cascade of identical cells with $X=\hat{X}\neq\emptyset$, $Z\neq\emptyset$, and $\hat{Z}\neq\emptyset$, then a single faulty cell in the cascade can be located with an uncertainty of 2 if the cell is $x/\hat{z}$-IL of finite order.

E. Two-Dimensional Arrays

Given the next-state mapping and the output mapping for a cell as $\delta:SxXxZ\rightarrow S$ and $\beta:SxXxZ\rightarrow\hat{Z}x\hat{X}$, respectively, let $\beta_{\hat{z}}:SxXxZ\rightarrow\hat{Z}$ and $\beta_{\hat{x}}:SxXxZ\rightarrow\hat{X}$

such that $\beta=\beta_{\hat{Z}} x \beta_{\hat{X}}$, then, one has the following theorem.

Theorem 39: If the cell of a two-dimensional array is chosen such that

(1) the reduced mapping $\beta_{\hat{X}}\rceil_{s}\rceil_{z}^{S}\rceil_{z}^{Z}: X \to \hat{X}$ is onto for $\forall s \in S$ and $\forall z \in Z$ and the reduced mapping $\beta_{\hat{Z}}\rceil_{s}\rceil_{x}^{S}\rceil_{x}^{X}: Z \to \hat{Z}$ is onto for $\forall s \in S$ and $\forall x \in X$,

(2) $|X| \geq |Z|$ and the two reduced mappings $\beta_{\hat{X}}\rceil_{s}\rceil_{z}^{S}\rceil_{z}^{Z}: X \to \hat{X}$ and $\beta_{\hat{Z}}\rceil_{s}\rceil_{z}^{S}\rceil_{z}^{Z}: X \to \hat{Z}$ are both onto for $\forall s \in S$ and $\forall z \in Z$, or

(3) $|Z| \geq |X|$ and the two reduced mappings $\beta_{\hat{X}}\rceil_{s}\rceil_{x}^{S}\rceil_{x}^{X}: Z \to \hat{X}$ and $\beta_{\hat{Z}}\rceil_{s}\rceil_{x}^{S}\rceil_{x}^{X}: Z \to \hat{Z}$ are both onto for $\forall s \in S$ and $\forall x \in X$,

then, any test sequence is applicable to every cell in the array.

Proof: Suppose the hypothesis holds, then, at any instance, i.e., no matter the states all cells are in, any input combination to a cell is applicable to any cell in the array with an application of some boundary inputs. Thus, any test sequence is applicable to every cell in the array. ∎

If the hypothesis in Theorem 39 is not satisfied, then, in order that a test sequence, say $(\underline{X}^t, \underline{Z}^t)$, where $\underline{X}^t = x_1 x_2 x_3 \ldots x_p$ and $\underline{Z}^t = z_1 z_2 z_3 \ldots z_p$, may be applicable to the cell (i,j) in ith row and jth column of a MxN array, at every particular instance, say at time t_k, $1 \leq k \leq p$, all the states in the cells in the first i rows and the first j columns except the cell (i,j) must be known and the marking of CISO-statuses (tessellation problem) on these cells must be solved where the input combination to the cell (i,j) is to be (x_k, z_k) and the CISO-status compatibility of a cell with respect to its neighboring cells is to be met.

A cell is said to be information lossless (IL) if and only if for each triplet $(s, \underline{X}, \underline{Z})$ there does not exist $\underline{X}^d$, $\underline{Z}^d$, where either $\underline{X}^d \neq \underline{X}$ or $\underline{Z}^d \neq \underline{Z}$ or both, such that $\hat{\delta}(s, \underline{X}^d, \underline{Z}^d) = \hat{\delta}(s, \underline{X}, \underline{Z})$ and $\underline{\beta}(s, \underline{X}^d, \underline{Z}^d) = \underline{\beta}(s, \underline{X}, \underline{Z})$. In constructing a testing graph G for testing whether or not a cell is IL or IL of

finite order, each node represents a compatible state pair (s_i, s_j), where states s_i and s_j form a compatible state pair (s_i, s_j) if either

P3: there exist some $s \epsilon S$, $x, x^d \epsilon X$, $z, z^d \epsilon Z$, where either $x^d \neq x$ or $z^d \neq z$ or both, such that $\delta(s, x, z) = s_i$, $\delta(s, x^d, z^d) = s_j$, and $\beta(s, x, z) = \beta(s, x^d, z^d)$, or

P4: (s_k, s_l) is a compatible state pair and there exist some $x, x^d \epsilon X$, $z, z^d \epsilon Z$, where x^d is not necessarily distinct from x and z^d is not necessarily distinct from z, such that $\delta(s_k, s, z) = s_i$, $\delta(s_l, x^d, z^d) = s_j$, and $\beta(s_k, x, z) = \beta(s_l, x^d, z^d)$.

There exists a branch from node (s_k, s_l) to node (s_i, s_j) in G if and only if these two nodes satisfy P4. Then, one has two results same as R1 and R2 in Section IV.D.1. except all appearances of $x/\hat{x}$-IL are replaced by IL.

Theorem 40: If some test sequence is applicable to every cell in a two-dimensional array and the cell is IL of finite order, then the array is testable.

Proof: Suppose a MxN array of cells $C_{11}, C_{12}, \ldots, C_{1N}, C_{21}, C_{22}, \ldots, C_{2N}, \ldots\ldots, C_{M1}, C_{M2}, \ldots,$ and C_{MN} satisfies the hypothesis and the cell is IL of finite order r, then, the testing procedure is testing M·N cells one by one starting at the cell C_{11}, then $C_{12}, C_{13}, \ldots, C_{21}, C_{22}, \ldots\ldots,$ and finally C_{MN}, the testing on the typical cell C_{ij} is as follows.

(1) Initialize all cells to the unique initial state.

(2) Apply some boundary input sequences to boundary cells such that the input sequence to C_{ij} is $(\underline{X}^t, \underline{Z}^t)$ which is a test sequence to test a cell completely.

(3) Next apply to boundary cells any known additional boundary input sequences of length L=(M-i+N-j)r.

(4) Reconstruct the response of C_{ij} in responding to the test sequence $(\underline{X}^t, \underline{Z}^t)$ by observing the boundary output sequences.

Suppose C_{ij} is faulty, by the assumption of the presence of single faulty cell, all other cells in the array must be operating correctly. Hence, the test sequence is correctly applicable to C_{ij}, what's more, the output sequence of C_{ij} is correctly reconstructable from the boundary output sequences. Then, the perturbed output sequence of C_{ij} in responding to the test sequence reveals that C_{ij} is in error. ∎

Define $U_L=\{(s,x,z) \mid s\epsilon S, x\epsilon X, z\epsilon Z$ and $\exists x^d\epsilon X, z^d\epsilon Z$, where either $x^d\neq x$ or $z^d\neq z$ or both, such that $\beta(s,x,z)=\beta(s,x^d,z^d)\}$. Then the following theorem is then obvious.

Theorem 41: If some test sequence is applicable to every cell in a MxN array of cells $C_{11}, C_{12}, \ldots, C_{1N}, C_{21}, C_{22}, \ldots, C_{2N}, \ldots, C_{M1}, C_{M2}, \ldots$, and C_{MN} but the cell is not IL, the presence of an error in C_{ij} is still detectable by observing the boundary output sequences if none of $C_{i(j+1)}, C_{i(j+2)}, \ldots, C_{iN}, C_{(i+1)j}, C_{(i+1)(j+1)}, \ldots, C_{(i+1)N}, C_{(i+2)j}, C_{(i+2)(j+1)}, \ldots, C_{(i+2)N}, \ldots\ldots\ldots, C_{Mj}, C_{M(j+1)}, C_{M(j+2)}, \ldots$, and C_{MN} is in a status in the set U_L during the testing for C_{ij}.

Theorem 42: If the cell of a two-dimensional array is chosen such that the reduced mapping $\beta_{\hat{x}}\left]{}^{S}_{s}\right]^{Z}_{z}: X\rightarrow\hat{X}$ is onto for $\forall s\epsilon S$ and $\forall z\epsilon Z$ and the reduced mapping $\beta_{\hat{z}}\left]{}^{S}_{s}\right]^{X}_{x}: Z\rightarrow\hat{Z}$ is onto for $\forall s\epsilon S$ and $\forall x\epsilon X$, then, a single faulty cell in the array can be located with an uncertainty of at most M+N-1, where M and N are the numbers of rows and columns, respectively, of the array.

Proof: Suppose the cell of a MxN array of cells $C_{11}, C_{12}, \ldots, C_{1N}, C_{21}, C_{22}, \ldots, C_{2N}, \ldots\ldots\ldots, C_{M1}, C_{M2}, \ldots$, and C_{MN} is chosen such that $\beta_{\hat{x}}\left]{}^{S}_{s}\right]^{Z}_{z}: X\rightarrow\hat{X}$ is onto for $\forall s\epsilon S$ and $\forall z\epsilon Z$ and $\beta_{\hat{z}}\left]{}^{S}_{s}\right]^{X}_{x}: Z\rightarrow\hat{Z}$ is onto for $\forall s\epsilon S$ and $\forall x\epsilon X$. Then, by Theorem 39, any test sequence is applicable

to every cell in the array. Since $X=\hat{X}$, $Z=\hat{Z}$, and are finite, $\beta_{\hat{x}\rfloor s\rfloor z}^{\neg S\neg Z}: X \rightarrow \hat{X}$ is onto for $\forall s \in S$ and $\forall z \in Z$ implies that $\beta_{\hat{x}\rfloor s\rfloor z}^{\neg S\neg Z}: X \rightarrow \hat{X}$ is one-to-one for $\forall s \in S$ and $\forall z \in Z$, and $\beta_{\hat{z}\rfloor s\rfloor x}^{\neg S\neg X}: Z \rightarrow \hat{Z}$ is onto for $\forall s \in S$ and $\forall x \in X$ implies that $\beta_{\hat{z}\rfloor s\rfloor x}^{\neg S\neg X}: Z \rightarrow \hat{Z}$ is one-to-one for $\forall s \in S$ and $\forall x \in X$. Now, suppose cell C_{ij} is faulty, then, by the assumption of the presence of single faulty cell, all other cells in the array must be fault-free. During the array testing, if the $\hat{x}$-output of C_{ij} is perturbed due to the fault, then, the $\hat{x}$-output of $C_{i(j+1)}$ is perturbed, eventually, the $\hat{x}$-output of C_{iN}, which is a boundary output, is perturbed, while the boundary outputs at C_{1N}, C_{2N}, C_{3N}, ..., and $C_{(i-1)N}$ are nominal, if the $\hat{z}$-output of C_{ij} is perturbed due to the fault, then, the $\hat{z}$-output of $C_{(i+1)j}$ is perturbed, eventually, the $\hat{z}$-output of C_{Mj}, which is a boundary output, is perturbed, while the boundary outputs at C_{M1}, C_{M2}, C_{M3}, ..., and $C_{M(j-1)}$ are nominal. Certainly, the presence of the fault in C_{ij} is detectable at boundary outputs.

Now, suppose in responding to the array testing, the boundary output sequences at C_{1N}, C_{2N}, C_{3N}, ..., $C_{(p-1)N}$, C_{M1}, C_{M2}, C_{M3}, ..., and $C_{M(q-1)}$ are nominal but that at C_{pN} and C_{Mq} are perturbed for some $1 \leq p \leq M$ and $1 \leq q \leq N$, and that at the remaining boundary cells each might be nominal or perturbed, then, the faulty cell could be any one of C_{1q}, C_{2q}, C_{3q}, ..., $C_{(p-1)q}$, C_{p1}, C_{p2}, C_{p3}, ..., $C_{p(q-1)}$, and C_{pq}, i.e., the faulty cell could be located with an uncertainty of $(p-1)+(q-1)+1=p+q-1$. In general, the range of uncertainty is from 1 to M+N-1. Thus, the Theorem follows. ∎

BIBLIOGRAPHY

[1]: Armstrong, D. B., "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets," IEEE Trans. EC-15, February 1966, pp. 66-73.

[2]: Berger, R., "The Undecidability of the Domino Problem," Memoirs of the American Math. Soc., No. 66, Providence, R. I., 1966.

[3]: Breuer, M. A., "Fault Detection in a Linear Cascade of Identical Machines," Pro. of Ninth Annual Symposium on Switching and Automata Theory, Schenectady, N. Y., October 1968.

[4]: Even, S., "On Information Lossless Automata of Finite Order," IEEE Trans. on Electronic Computers, EC-14, 1965, pp. 561-569.

[5]: Farmer, D. E., "A Strategy for Detecting Faults in Sequential Machines Not Possessing Distinguishing Sequences," Pro. of 1970 Fall Joint Computer Conference, pp. 493-501.

[6]: Hennie, F. C., "Fault Detecting Experiments for Sequential Circuits," Pro. of Fifth Ann. Sym. on Switching Theory and Logical Design, Princeton, N. J., November 1964, pp. 95-110.

[7]: Huffman, D. A., "Cannonical Forms for Information-Lossless Finite-State Logical Machines," IRE Trans. Circuit Theory Special Supplement CT-6, 1959, pp. 41-59.

[8]: Kohavi, I., "Fault Diagnosis of Logical Circuits," Pro. of Tenth Ann. Sym. on Switching and Automata Theory, Ontario, Canada, October 1969.

[9]: Kahavi, Z., and P. Lavalle, "Design of Sequential Machines with Fault-Detecting Capabilities," IEEE Trans. on Electronic Computers, EC-16, 1967, pp. 473-485.

[10]: Kahr, A. S., E. F. Moore and H. Wang, "Entscheidungsproblem Reduced to the VEV Case," Pro. Nat. Acad. of Sci., 48, 1962, pp. 365-377.

[11]: Kautz, W. H., "Testing for Faults in Combinational Cellular Logic Arrays," Pro. of Eighth Ann. Sym. on Switching and Automata Theory, Austin, Texas, October 1967, pp. 161-174.

[12]: _____, "Properties of Cellular Arrays for Logic and Storage," SRI Scientific Report 3, Contract AF19(628)-5828, Stanford Research Institute, Menlo Park, Calif., July 1967, Chapter VI.

[13]: _____, "Fault Testing and Diagnosis in Combinational Digital Circuits," IEEE Trans. on Computers, C-17, April 1968, pp. 352-366.

[14]: Minnick, R. C., "Cutpoint Cellular Logic," IEEE Trans. on Electronic Computers, EC-13, December 1964, pp. 685-698.

[15]: Poage, J. F., and E. J. McCluskey, Jr., "Derivation of Optimum Test Sequences for Sequential Machines," Pro. of Fifth Ann. Sym. on Switching Theory and Logical Design, Princeton, N. J., November 1964, pp. 121-132.

[16]: Roth, J. P., "Diagnosis of Automata Failures: A Calculus and a Method," IBM J. Res. and Dev. 10, July 1966, pp. 278-291.

[17]: Seth, S. C., "Fault Diagnosis of Combinational Cellular Arrays," Pro. of Seventh Ann. Allerton Conference on Circuit and System Theory, Urbana, Illinois, October 1969, pp. 272-283.

[18]: Tammaru, E., "Efficient Testing of Combinational Logic Cells in Large-Scale Arrays," Stanford Electronics Laboratories Technical Report No. 4601-1 (SU-SEL-68-040), May 1968, also the IEEE Computer Group Repository R-69-115, IEEE, New York, N. Y., 1969.

[19]: Wang, H., "Dominoes and the AEA Case of the Decision Problem," Proc. of the Sym. on Math. Theory of Automata, Polytechnic Press of the Polytechnic Institute of Technology, Vol. 12, New York, April 1962, pp. 23-55.