COMPUTER SYSTEMS LABORATORY

STANFORD UNIVERSITY . STANFORD, CA 943054055

Fault Equivalence in Sequential Machines

by

R. Boute
E. J. McCluskey

June 1971

Technical Report No. 15

This work was supported by the National Science Foundation under Grant GJ 27527, by the Joint Services Electronics Program under Contract N-00014-67-A-0112-0044, and while Mr. Boute was partially supported by the Nationaal Fonds voor Wetenschappelijk Onderzoek of Belgium.

Technical Report no. 15 June 1971

SEL-71-038

FAULT EQUIVALENCE IN SEQUENTIAL MACHINES

R. Boute and E. J. McCluskey

DIGITAL SYSTEMS LABORATORY

Stanford University, Stanford, California

ERRATA SHEET

P. 9 Line 24 Fig. 2 should be Fig. lb

P. 11 Lines 3,4 . ..faults should be . ..faults. with $|F_Q| = (|\sigma| + 1)^m$. It will...

P. 14 Line 6 Figure 3 should be Figure 2

P. 22 Line 21 $q' = \bar{\delta}(q,\bar{x})$ should be $q' = \delta(q,\bar{x})$

SEL-71-038

FAULT EQUIVALENCE IN SEQUENTIAL MACHINES

by

R. Boute and E. J. McCluskey

June 1971

Technical Report no. 15

DIGITAL SYSTEMS LABORATORY

Stanford Electronics Laboratories

Stanford University

Stanford, California

This work was supported by the National Science Foundation under grant GJ 27527, by the Joint Services Electronics Program under contract N-00014-67-A-0112-0044, and while Mr. Boute was partially supported by the Nationaal Fonds voor Wetenschappelijk Onderzoek of Belgium.

ABSTRACT

This paper is concerned with the relationships among faults as they affect sequential machine behavior. Of particular interest are equivalence and dominance relations.

It is shown that for output faults (i.e., faults that do not affect state behavior), fault equivalence is related to the existence of an automorphism of the state table. For the same class of faults, the relation between dominance and equivalence is considered and some properties are pointed out. Another class of possible faults is also considered, namely, memory faults (i.e., faults in the logic feedback lines). These clearly affect the state behavior of the machine, and their influence on machine properties, such as being strongly connected, is discussed. It is proven that there exist classes of machines for which this property of being strongly connected is destroyed by every possible single fault. Further results on both memory and output faults are also presented.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

The increasing complexity of digital systems has necessitated a more systematic approach in designing adequate test procedures than the heuristic or intuitive methods that prevailed in the early days of computer technology.

For sequential machines, the first "testing' was done by Moore [1] although his primary interest was in analyzing the state behavior of a machine rather than in looking for possible faults. Seshu and Freeman [2, 13] designed test sequences to detect faults, while looking at the possible ways in which a given circuit can be affected by a fault. Hennie [3] also developed fault detecting methods, but the test sequences were designed without the extra information obtainable from an investigation of the influence of faults from a given fault set on the machine: the sequences verify whether or not the machine performs its given task. Thus this method links up with Moore's approach.

A systematic procedure, guaranteeing optimal test sequences for sequential machines, is given by Poage and McCluskey [4]. In this paper, all the information concerning the faults that is available is used, as in the Seshu and Freeman method, but the procedure used does not suffer from the disadvantages of local optimization. It is also in the work of Poage and McCluskey that the idea of "dominance" for faults in sequential circuits is introduced, in analogy with dominance of rows and columns in prime implicant tables [5].

While studying the behavior of combinational circuits, it was soon realized that relations existed between faults. Investigations on equivalence relations have been done by Clegg and McCluskey [6,7] and Schertz [8,9]. Roughly speaking, two faults are said to be equivalent if and only if they have the same effect on the circuit according to some suitable criterion, different criteria giving rise to different equivalence relations. Perhaps the most basic type of equivalence is functional equivalence: two faults are functionally equivalent if they give rise to the same set of circuit output functions. For most of the definitions used, equivalence always implies functional equivalence. This means that the corresponding partitions are refinements of the partition induced by functional equivalence, and thus all faults in the same equivalence class yield the same input/output behavior. The importance of this observation becomes clear from the fact that fault detecting sets have to be designed only for as many faults as there are equivalence classes, instead of for each element of the specified fault set separately.

Another kind of relation exists in combinational circuits, namely dominance. A fault F_1 dominates a fault F_2 if and only if it can be detected by every test for F_2. This also will reduce the number of faults that have to be examined separately. Mei [10] has shown that dominance gives rise to drastic reductions in the number of faults which must be considered when the circuit satisfies some general conditions.

Similar phenomena also exist in sequential machines, and it is the main objective of this paper to point out those similarities.

Further, some specific aspects of faults in sequential machines that have no counterpart in combinational circuits are also examined, such as the properties of output faults and the influence of memory faults on the strong-connectedness of the machine. This last aspect might seem rather loosely related to equivalence and dominance, but the results derived will show the desirability of having reset circuitry in case we want to compare faulty machines with each other.

The considerations on dominance and equivalence in this paper have a dual purpose: first, the simplification of the task of designing test sequences by reducing the number of faults that need to be considered separately, and second, the gaining of a better understanding of the phenomena that affect the behavior of a machine in the presence of faults.

In what follows, we first introduce the notation used throughout this paper. Subsequently, the concepts of equivalence and dominance in sequential and combinational circuits are clearly defined and their significance explained. Also, the concept of "reset" is introduced in order to pin down these ideas more closely. Then we consider a particular class of faults, namely, output faults (i.e., those that do not affect the state behavior) and the existence of an equivalence relation is shown to depend on the existence of a state-table automosphism.

The relationships between dominance and equivalence are explored for combinational circuits and then for sequential circuits, thus enabling us to point out some similarities between the two cases. It will become clear at this point that the presence of reset circuitry makes the setting up of a basis for comparing sequential machines much easier.

Finally, the problem of memory faults will be considered and we will show that there exist classes of strongly connected machines for which every single fault destroys this property. The main result of this section implies in fact that reset circuitry is Important if we want to compare two faulty machines for all possible Initial states. We conclude then with some suggestions on future research.

II. NOTATION

A. Combinational Circuits

Let C be a combinational circuit realizing a mapping $\Phi: \sigma^n \to \sigma^m$, where σ is a finite nonempty set of symbols. In most applications, $\sigma = \{0, 1\}$ and Φ is then a Boolean function.

The number of input leads is n and the number of output leads, m. Input leads are denoted by x_i; output leads by y_i; see for example Fig. la.

$$\Phi(x_1, x_2, \ldots, x_n) = \langle y_1, y_2, \ldots y_n \rangle .$$

Here we write ( ) instead of (< >) for simplicity, but otherwise we will always use angle brackets < > to denote ordered n-tuples.

B. Sequential Machines

A sequential machine M is a 5-tuple:

$$M = \langle I, O, Q, \delta, \lambda \rangle \quad (1)$$

where I = a finite nonempty set of inputs

O = a finite nonempty set of outputs

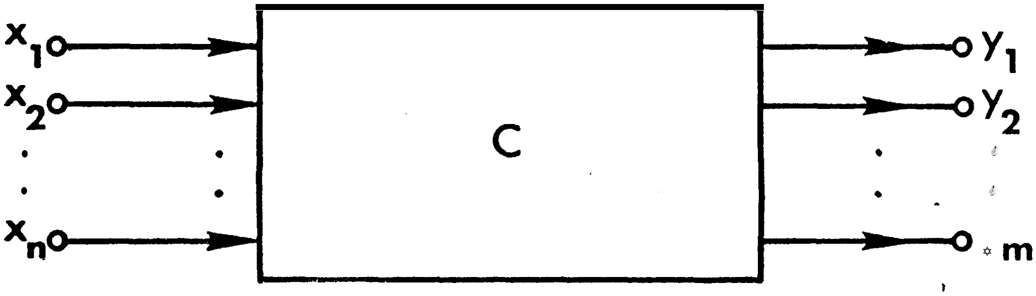
Q = a finite nonempty set of states

δ = the next-state function: δ: Q x I → Q; defined by $\delta: \langle q_t, i_t \rangle \mapsto q_{t+1}$; the subscript t standing for "at time t ".

λ = the output function:

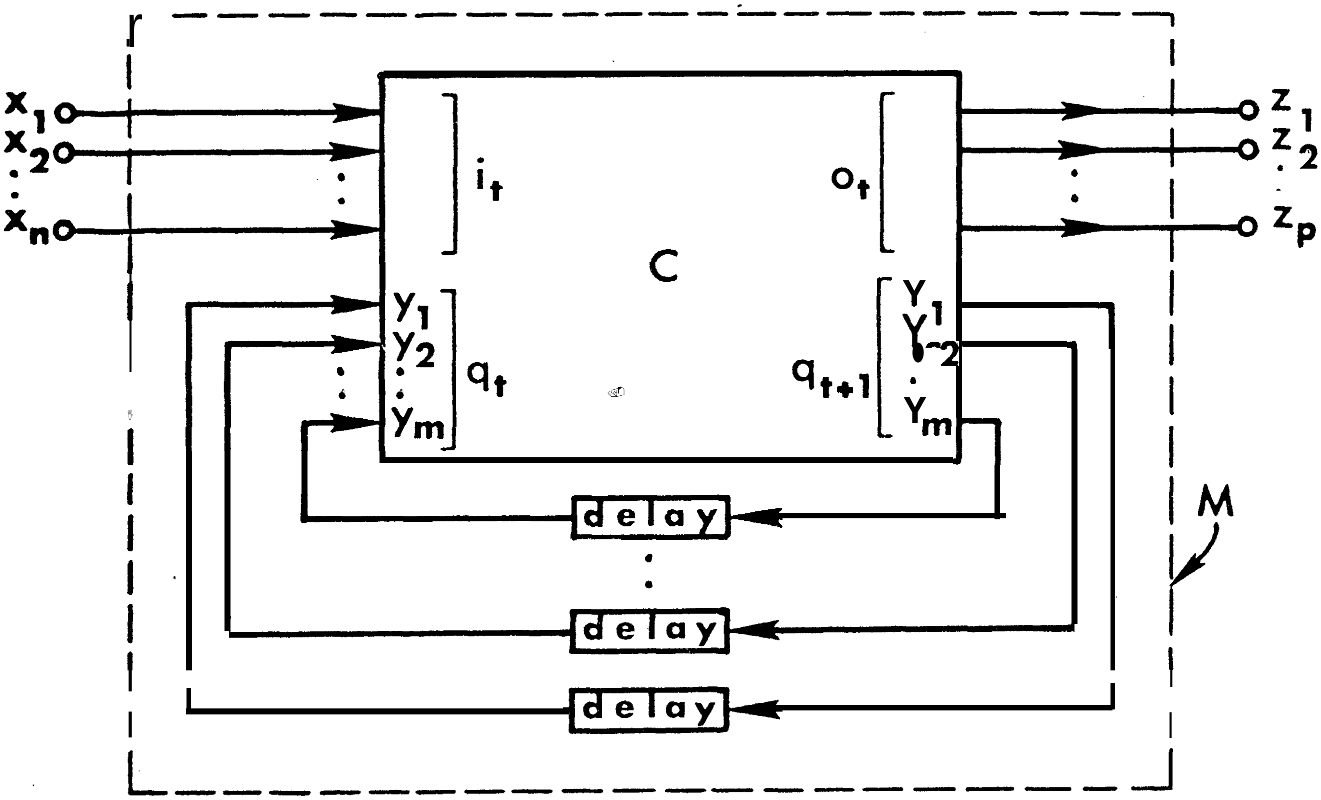
(a) for a Moore machine, $\lambda: Q \to O$ defined by, $\lambda: q_t \mapsto o_t$

(b) for a Mealy machine, $\lambda: Q \times I \to O$ defined by, $\lambda: \langle q_t, i_t \rangle \mapsto o_t$



(a)

Combinational



(b)

Sequential

Fig. 1: Representation of a combinational circuit C and a sequential machine M

Following Hartmanis and Stearns [11], we define now

I^+: the set of all finite input sequences of nonzero length consisting of symbols from I;

$I^* = I^+ \cup \{\Lambda\}$ where A is the empty sequence.

Elements of I^+ or I^* will be denoted by $\bar{x}$. We extend δ to $Q \times I^*$ in the following recursive way:

(1) $\delta(q, \Lambda) = q \ \forall \ q \in Q$

(2) $\delta(q, \bar{x}x) = \delta[\delta(q, \bar{x}), x]$,
where $q \in Q$, $x \in I$, $\bar{x} \in I^*$

We also define $\bar{\lambda}$ as follows:

let $q \in Q$, $x \in I$, $\bar{x} \in I^*$ (or I^+); then,

a) for a Moore machine, $\bar{\lambda}: Q \times I^* \rightarrow I+$,
defined by $\bar{\lambda}(q,\Lambda) = h(q)$
and $\bar{\lambda}(q,\bar{x}x) = \bar{\lambda}(q,\bar{x}) . \lambda[\delta(q,\bar{x}x)]$

b) for a Mealy machine, $\bar{\lambda}$: Q x I+' I+,
defined by $\bar{\lambda}(q,x) = \lambda(q,x)$
and $\bar{\lambda}(q,\bar{x}x) = \bar{\lambda}(q,\bar{x}) . \lambda[\delta(q,\bar{x}), x]$.

One of the possible realizations is shown in Fig. lb:

$$Q = \sigma^m = \{q \mid q = \langle y_1, y_2, \ldots, y_m \rangle, y_i \in \sigma\} \tag{2a}$$

$$I = \sigma_I^n = \{i \mid i = \langle x_1, x_2, \ldots, x_n \rangle, x_j \in \sigma_I\} \tag{2b}$$

$$O = \sigma_O^p = \{o \mid o = \langle z_1, z_2, \ldots, z_p \rangle, z_k \in \sigma_O\} \tag{2c}$$

Usually one has: $\sigma_I = \sigma = \sigma_O = \{0, 1\}$, but in general $\sigma_{..}$ can be any set of symbols (finite and nonempty). Except when explicitly mentioned, such as will be the case for the section on memory faults, we will think about the machine as the abstract device specified by (1), without referring to the representation of Fig. lb.

C. Faults

In what follows we will only consider permanent faults. By this we mean that their effect on machine behavior does not change during or in between tests. It will also be assumed that a fault in a combinational network does not transform the network into a sequential circuit. No other restrictions on the faults are assumed unless they are explicitly stated.

We will use the notation F(C) for the combinational circuit C affected by a fault F, and similarly F(M) for a sequential machine M with fault F. The use of the notation for F as if it were a function can be justified as follows. If we consider interconnections between circuit elements also as 'components", then the domain of F is the set of all machines that have a specified subset of their components in common. The mapping F is then defined by describing the way in which F affects the components of that common subset. Note that this applies to a wide variety of faults, including bridging faults. However, we will only use this approach for the case of memory faults, where the subset mentioned above will be clearly specified. Another point of view can be that the domain contains only a single machine (C or M).

The absence of a fault is denoted by **e**, called the "empty fault". So, for example, e(C) = C and e(M) = M.

If Φ is the function realized by C, then Φ_F is the one realized by F(C). The same convention is used for sequential machines, i.e.,

$$\text{if } M = \langle I, 0, Q, 6, \lambda \rangle$$

$$\text{then } F(M) = \langle I, 0, Q_F, \delta_F, \lambda_F \rangle .$$

There is no advantage gained by letting F affect I and 0, but for Q it will become conceptually useful.

Finally, the set of all faults that will be considered for a given machine or class of machines (circuits) will be denoted by $\mathfrak{F}$. $\mathfrak{F}$ includes e, by convention.

D. Classes of Faults

In general, a fault F in a sequential machine can affect the next-state function δ as well as the output function λ. For many realizations, however, it is reasonable to consider subclasses of faults that affect either δ or λ, but not both. This is especially true for cases in which δ and λ are realized by separate circuits. Thus, we can consider as special subclasses:

Output faults

A fault F is said to be an "output fault" for a machine M iff $Q_F = Q$ and $\delta_F = 6$. The set of all output faults for a machine M will be denoted by $\mathfrak{F}_\lambda$.

Next-state faults

A fault F is a next-state fault for a machine M iff $\delta_F \neq \delta$, while the output circuitry is unaffected. The set of all next-state faults for a given machine is denoted by $\mathfrak{F}_\delta$. A further subclass of next-state faults are the memory faults described next.

Memory faults

We consider the set $\mathfrak{M}_m$ of all machines with m delay elements and a given alphabet σ (see Fig. 2). For the set of feedback lines $L_m = \{ \ell_i | i = 1, 2, .. m\}$ we define a stuck-at fault F as a map

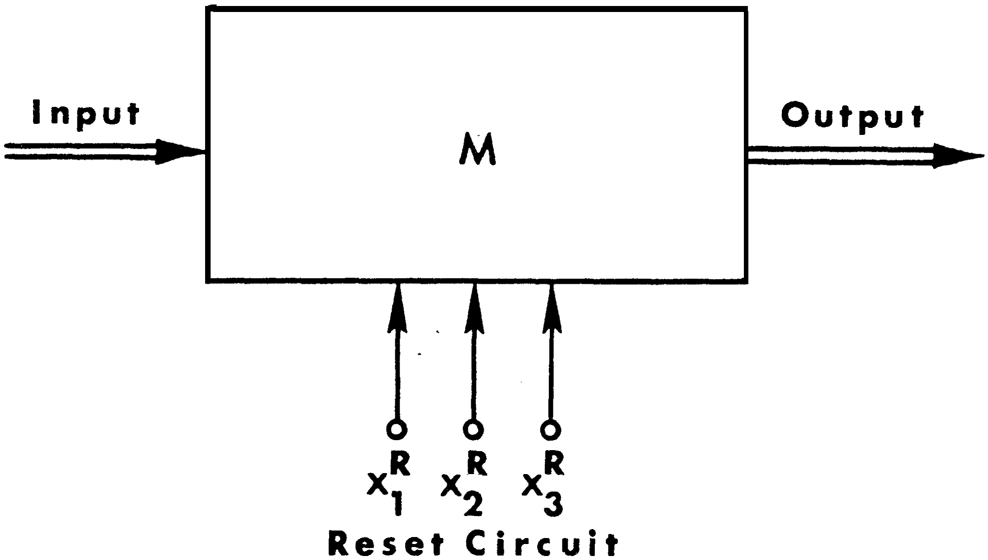


Fig. 2: A sequential machine with a reset circuit

$$F: \quad L_m \rightarrow \sigma \cup \{N\} \tag{3}$$

assigning to each line a stuck-at value or the status N ("normal").

We denote by $\mathcal{F}_Q$ the set of all such faults.

[$|\mathcal{F}_Q| = (|\sigma| + 1)^m$]. It will become clear at a later stage how this definition is made compatible with the notation $F(M)$.

We will also define $T_F = \{k \mid F(\ell_k) \neq N\}$.

III. DEFINITIONS

A. Combinational Circuits

A detection set for a fault F is a subset of the set of inputs defined by

$$S_F = \{\langle x_1, x_2, \ldots, x_n\rangle \mid \Phi_F(x_1, \ldots, x_n) \neq \Phi_e(x_1, \ldots, x_n)\}$$

$$(S_F \subseteq \{0, 1\}^n)$$

A combinational circuit is nonredundant iff $S_F \neq \emptyset \ \forall F \in \mathfrak{F} - \{e\}$. This refers to a well-specified class of faults $\mathfrak{F}$.

A fault F_1 dominates a fault F_2 iff $S_{F_1} \supseteq S_{F_2}$. Thus every test detecting F_2 also detects F_1. Notation used: $F_1 \geq F_2$.

Two faults F_1 and F_2 are functionally equivalent, written $F_1 \sim F_2$, or briefly equivalent if and only if

$$\Phi_{F_1}(\bar{x}) = \Phi_{F_2}(\bar{x}) \quad \forall \bar{x} \in \{0, 1\}^n .$$

Two faults F_1 and F_2 are detection equivalent, written $F_1 \simeq F_2$, if and only if $S_{F_1} = S_{F_2}$.

B. Sequential Machines

Two machines M_1 and M_2 are equivalent iff

$$\forall q_1 \in Q_1 \text{ (state set of } M_1) \ \exists \ q_2 \in Q_2 \ni$$

$$\bar{\lambda}_1 (q_1, \bar{x}) = \bar{\lambda}_2 (q_2, \bar{x}) \ \forall \ \bar{x} \in I^* \text{ or } I^+$$

and vice versa.

Two faults F_1 and F_2 in a sequential machine M are equivalent iff $F_1(M)$ and $F_2(M)$ are equivalent.

c. Resets

Let $M = \langle I, 0, Q, 6, \lambda\rangle$ be a sequential machine with a reset circuit. By this we mean an extra set of input terminals on which we can apply inputs from a given set J. The circuit is constructed in such a way that for some subset of J, denoted by IR (the "reset inputs"), there exists a map $p: I_R \to Q$ with the following property: if any $i \in I_R$ is applied to the reset terminals, the machine goes to state p(i), no matter what the state was before. For practical reasons we require that I_R be a proper subset of J and that all $i \in J - I_R$ remain without effect on the machine. Indeed, the machine should be able to distinguish resets from normal operation mode. One can also use the elements of $J - I_R$ to put the machine in other modes ("partial resets", for example, where only some of the y_i in $q = \langle y_i, y_2, \ldots, y_m\rangle$ are reset), but we will not exploit this possibility at this point, although it can resultin more flexible test sequence design.

Note that this approach, where we consider resets as an extra feature, is only taken for convenience. Indeed, these extra circuits can be incorporated in the usual model for a sequential machine by redefining the machine $M = \langle I, 0, Q, 6, \lambda\rangle$ as $M = \langle I', 0, Q, \delta', \lambda'\rangle$ where $I' = I \times J$ (Cartesian product)

$$\delta'(q, \langle i,j\rangle) \begin{cases} = \delta(q, i) & \text{if } j \notin IR \\ = \rho(j) & \text{if } j \in I_R \end{cases}$$

$$\lambda'(q, \langle i,j\rangle) = \lambda(q, i) \quad \text{if } j \notin I_R$$
$$= \text{undefined if } j \in I_R .$$

Conversely, if a machine M has already inputs i such that $\delta(q, i) = \delta(q', i)\ \forall\ q, q' \in Q$, then we can define the set I_R as being the set of elements with this property. In this case $I - I_R$ is the set of "normal" machine inputs.

This way the two points of view can be considered as equivalent.

EXAMPLE. Figure 3 represents a sequential machine with a reset circuit consisting of three extra inputs: x_1^R, x_2^R, x_3^R, where $x_i^R \in \{0, 1\}$. Thus $J = \{0, 1\}^3$. We define I_R arbitrarily as the set

$$I_R = \left\{\langle x_1^R, x_2^R, 1\rangle \mid x_i^R \in \{0, 1\}\right\} \subseteq J .$$

Physically, this means that we consider x_3^R as the enable/disable line for the reset circuitry.

We now can specify ρ, for example, as follows:

$$\forall\ \langle x_1^R, x_2^R, 1\rangle = i \in I_R$$

[illegible] $\langle x_1^R + x_2^R, x_2^R, x_1^R\rangle \in$ [illegible] .

This results in the table:

reset input	reset state
0 0 1	0 0 0
0 1 1	1 1 0
1 0 1	1 0 1
1 1 1	1 1 1

If the elements of I are of the form $\langle x_1, x_2, \ldots x_n\rangle$, we can denote those of I' by $\langle x_1, x_2, \ldots, x_n, x_1^R, x_2^R, x_3^R\rangle$ and define

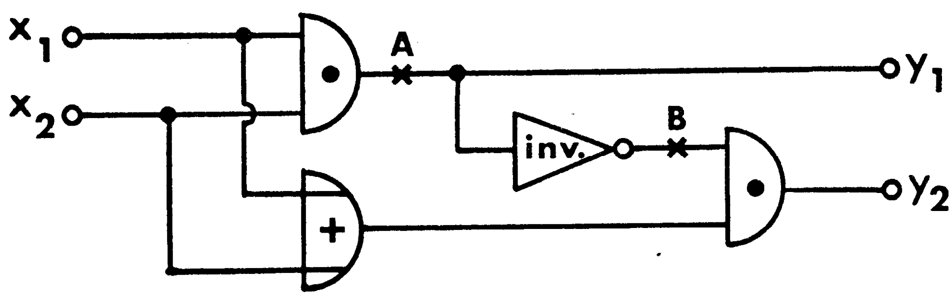


Fig. 3: To prove lemma 3 in section VI

$$\delta'(q, <x_1, \ldots, x_n, x_1^R, x_2^R, x_3^R>)$$

$$= \begin{cases} \delta(q, <x_1, \ldots, x_n>) & \text{if } x_3^R = 0 \\ \rho(x_1^R, x_2^R, x_3^R) & \text{if } x_3^R = 1 \end{cases}$$

D. Faults and the Reset Circuitry

Suppose we have a machine with reset as described above, but with a fault F.

This fault F may act in such a way that upon application of $i \in I_R$, the new state also depends on the previous one. For simplicity, we will assume such faults do not occur;* i.e., in the faulty machine M_F we still have a map $\rho_F: I_R \rightarrow Q_F$ that determines $\rho_F(i)$ as the state after the application of an input $i \in I_R$.

Although possibly ρ_F differs from ρ, we will say that, in such cases, the reset is unaffected by the fault F. This simply means that, if we know the fault, we also know the state of the machine after applying any $i \in I_R$.

This becomes important in the following definition: a reset test sequence for a fault F is a sequence $\bar{x} \in I^+$ or I^* preceded by a reset input $i \in I_R$, such that

$$\bar{\lambda}[\rho(i), \bar{x}] \neq \bar{\lambda}_F[\rho_F(i), \bar{x}] .$$

A fault F1 dominates a fault F_2 iff every reset test sequence for F_2 is also a reset test sequence for F1 (i.e., $S_{F_1} \supseteq S_{F_2}$ where $S_F = \{\bar{x} \mid \bar{x}$ is a reset test sequence for $F\}$).

------

* In the case of stuck-at faults, this assumption is very reasonable, except for such cases as x_3R stuck at 0 in the example just discussed.

Single reset machines are machines where $|I_R| = 1$.

A machine M_1 is reset equivalent to M_2 (both same type) iff

(1) $I_{R_1} = I_{R_2}$, $I_1 = I_2$

(2) for all $i \in I_R$ and $\bar{x} \in I^+$ or I^*, we have

$$\lambda_1 [\rho_1(i), \bar{x}] = \lambda_2 [\rho_2(i), \bar{x}] .$$

A fault F_1 is reset equivalent to F_2 $(F_1 \sim F_2)$ iff M_{F_1} is reset equivalent to M_{F_2}.

A fault F_1 is detection equivalent to F_2 iff $S_{F_1} = S_{F_2}$.

IV. EQUIVALENCE OF OUTPUT FAULTS IN SEQUENTIAL MACHINES

DEFINITION. An output function λ is nontrivial iff $\exists\ q_1, q_2 \in Q$ and $i_1, i_2 \in I \ni \lambda(q_1, i_1) \neq \lambda(q_2, i_2)$, or if $|Q \times I| = 1$ (i.e., don't blame λ if M is trivial).

LEMMA. Given λ_1 and λ_2, both defined on Q x I, and $\lambda_1 \neq \lambda_2$. Then, in case $\exists\ q_1, i_1$ and $q_2, i_2 \ni \lambda_1(q_1, i_1) = \lambda_2(q_2, i_2)$, at least one of them is nontrivial.

PROOF. Suppose $\lambda_1(q_1, i_1) = \lambda_1(q, i)\ \forall q, i$ (i.e., λ_1 trivial). Since $\lambda_1 \neq \lambda_2\ \exists\ q_3, i_3 \ni \lambda_1(q_3, i_3) \neq \lambda_2(q_3, i_3)$.
So $\lambda_2(q_2, i_2) = \lambda_1(q_1, i_1) = \lambda_1(q_3, i_3) \neq \lambda_2(q_3, i_3)$; i.e., $\lambda_2(q_2, i_2) \neq \lambda_2(q_3, i_3)$, and λ_2 is nontrivial.

For the remainder of this section, let F_1 and F_2 be output faults for a machine M, and in the interest of simplicity, let λ_{F_1} and λ_{F_2} be denoted by λ_1 and λ_2, respectively.

THEOREM. If $F_1(M)$ and $F_2(M)$ are equivalent and $\lambda_{F_1} \neq \lambda_{F_2}$, then both λ_{F_1} and λ_{F_2} are nontrivial.

PROOF. Let $q_1, q_2, q_3, q_4 \in Q$. Since $F_1(M)$ and $F_2(M)$ are equivalent, $\forall q_1 \exists q_2$ (and $\forall q_2 \exists q_1$), such that $\bar{\lambda}_1(q_1, \bar{x}) = \bar{\lambda}_2(q_2, \bar{x})$ $\forall\ \bar{x} \in I^*$ (or I^+). Thus, $\exists\ q_1$ and $q_2 \ni \lambda_1(q_1, i) = \lambda_2(q_2, i)\ \forall i \in I$ and $\exists\ q_1$ and q_2, and $\exists\ i_1, i_2 \in I$ such that $\lambda_1(q_1, i_1) = \lambda_2(q_2, i_2)$ (take simply $i_1 = i_2 = i$).

According to the previous lemma, one or both output functions is nontrivial, say, $\lambda_1(q_3, i_3) \neq \lambda_1(q_4, i_4)$. But since $F_1(M)$ and $F_2(M)$ are equivalent: $\exists\, q_3', q_4' \ni \lambda_2(q_3', i_3) = \lambda_1(q_3, i_3)$ and $\lambda_2(q_4', i_4) = \lambda_1(q_4, i_4)$. So $\lambda_2(q_3', i_3) \neq \lambda_2(q_4', i_4)$ and λ_2 is also nontrivial. This proves the theorem.

Trivial λ_i are uninteresting since they make all states of $F_i(M)$ equivalent and don't reveal anything about the state transitions in the machine. Therefore, the above theorem is interesting because it assures us that no trivial λ_i exist in the conditions specified.

State Table Automorphisms and Output Faults

DEFINITION. A state table automorphism for a machine M is a bijective map $\varphi: Q \to Q$ such that $\delta[\varphi(q), i] = \varphi[\delta(q, i)]$ for all $q \in Q, i \in I$. (More general definitions, such as $\Phi: Q \times I \to Q \times I$, etc. are possible, but we will not consider them here.) See also [14, 15, 16, 17, 18].

The trivial automorphism ($\varphi = 1$) always exists.

THEOREM. Let F_1 and F_2 be equivalent. Then $F_1(M)$ and $F_2(M)$ are equivalent (by definition), and if $F_1(M)$ is reduced then

1) $F_2(M)$ is reduced
2) $\exists$ is a uniquely defined state table automorphism φ
3) φ is nontrivial if $\lambda_1 \neq \lambda_2$.

Remark. By "uniquely defined" we mean "uniquely specified" by λ_1 and λ_2, not necessarily that the automorphism itself is unique for the given M.

PROOF. Define a relation qRq' iff $[\bar{\lambda}_1(q, \bar{x}) = \lambda_2(q', \bar{x})$ $\forall \bar{x} \in I+$ or $I^*]$ (4)

Now define $S(q)$ as the set $\{q' \in Q \mid qRq'\}$.

(a) Because of machine equivalence: $\forall q \in Q \; \exists \; q' \in Q \ni qRq'$, i.e., $S(q) \neq \emptyset \; \forall q$

(b) Further, suppose $q' \in S(q_i) \cap S(q_j)$, i.e., q_iRq', q_jRq', then $\bar{\lambda}_1(q_i, \bar{x}) = \lambda_2(q', \bar{x}) = \bar{\lambda}_1(q_j, \bar{x}) \; \forall \bar{x} \in I+$ or I^*. But this implies, since $F_1(M)$ is reduced, $q_i = q_j$. So $q_1 \neq q_j \Rightarrow S(q_i) \cap S(q_j) = \emptyset$.

This implies, for an N-state machine $\left| \bigcup_{i=1}^{N} S(q_i) \right| = \sum_{i=1}^{N} \left| S(q_i) \right|$ with $\left| S(q_i) \right| \geq 1$ because of (a). Since $\bigcup_{i=1}^{N} S(q_1) \subseteq Q$ we obtain $\sum_{i=1}^{N} \left| S(q_1) \right| \leq N$ and conclude $\left| S(q_i) \right| = 1$. Therefore R is a function which we denote by $\varphi: Q \to Q$. Further $\left| \bigcup_{i=1}^{N} S(q_i) \right| = N$ implies that φ is surjective, and since the range and the domain are the same set, bijective. Thus $\forall q \in Q$ one can write $qR\varphi(q)$.

1) Take $q_1' \neq q_2'$ then $\varphi^{-1}(q_1') \neq \varphi^{-1}(q_2')$ (bijective φ) and $\bar{\lambda}_1[\varphi^{-1}(q_1'), \bar{x}] \neq \bar{\lambda}_1[\varphi^{-1}(q_2'), \bar{x}]$ for some $\bar{x}$ because $F_1(M)$ is reduced. This implies also $\bar{\lambda}_2(q_1', \bar{x}) \neq \bar{\lambda}_2(q_2', \bar{x})$ by definition of cp. So $F_2(M)$ is reduced.

2) The relation R and thus φ is uniquely defined. We have only to prove now φ is an automorphism. Take any $q \in Q$, then the definition of φ implies

$\bar{\lambda}_1(q, x\bar{x}) = \bar{\lambda}_2[\varphi(q), x\bar{x}] \; \forall x \in I$ and $\bar{x} \in I^+$ or I^*

$\bar{\lambda}_1[\delta(q, x), \bar{x}] = \bar{\lambda}_2\{\delta[\varphi(q), x], \bar{x}\} \; \forall x \in I$ and $\bar{x} \in I^+$ or I^*,

i.e., $\delta(q, x) \; R \; \delta[\varphi(q), x] \quad \forall x \in I$

and since R defines φ uniquely (i.e., $\forall$ q $\in$ Q

$\exists$ only one q' $\in$ Q $\ni$ qRq', namely q' = $\varphi(q)$), we get

$\delta[\varphi(q), x] = \varphi[\delta(q, x)] = \varphi[\delta(q, x)]$ $\forall$ x $\in$ I

and since q was arbitrarily chosen, also $\forall$ q $\in$ Q .

3) If $\lambda_1 \neq \lambda_2$, then $\exists$ q, i $\in$ Q, I

$\ni \lambda_1(q, i) \neq \lambda_2(q, i)$

i.e., $\varphi(q) \neq q$. φ is nontrivial.

Note that $\lambda_1 \neq \lambda_2 \Rightarrow$ λ_1 and λ_2 nontrivial
I φ nontrivial .

Application of the Theorem

As will become clear from the considerations below, most machines do not have a nontrivial state table automorphism. For such a machine M without nontrivial state table automorphisms, every fault $F \in \mathfrak{F}_\lambda$ with the property that F(M) is reduced will fall in a separate equivalence class of its own. In most cases, this will necessitate separate testing for each of these faults, especially when no dominance relations exist.

B. The Automorphisms of a State Table

The general problem of finding all automorphisms of a given state table is as yet unsolved. The same holds for the equivalent task of finding the automorphisms of a graph, which is an important problem in organic chemistry and many other areas.

Nevertheless, several heuristic approaches are possible. In particular, for state tables that are not too large, inspection "by eye" usually reveals very quickly the nonexistence of a nontrivial automorphism, if such is the case.

A study of the properties of state table automorphisms may simplify the heuristic approach considerably, since they show what one should look for first, while searching for automorphisms.

Weeg, Fleck and Barnes have done a considerable amount of work in the area of automorphisms of machines [14, 15, 16, 17].

Let us point out some of the facts that are relevant to this problem.

Since a state-table automorphism φ implies a bijective mapping from the state set Q onto itself, it is in fact a permutation on Q.

LEMMA. The set A_M of all state-table automorphisms for a machine M with N states forms a subgroup of S_N (under composition).

PROOF. Closure, identity, inverses are easily verified, and the remark above implies that $A \subseteq S_N$.

LEMMA. If M is strongly connected, then A_M is a group of regular permutations. By this we mean: the cycles of every permutation have equal length.

PROOF. Let q and q' be two arbitrary elements of Q, and φ a state table automorphism. Then we can find an input sequence $\bar{x} \in I^*$ such that $q' = \bar{\delta}(q, \bar{x})$. Suppose k is the length of the cycle of φ that contains q where φ is considered as a permutation on Q written in cycle notation. Then

$\varphi^k(q) = q$ and $\varphi^k(q') = \varphi^k[\bar{\delta}(q, \bar{x})] = \bar{\delta}[\varphi^k(q), \bar{x}] = \bar{\delta}(q, \bar{x}) = q'$, as can be easily seen from the definition of $\bar{\delta}$. Therefore, the length k' of the cycle containing q' satisfies $k' < k$. Similarly, one shows $k < k'$ and thus $k' = k$.

LEMMA. If M is strongly connected and $\tau \in A_M$, then giving $\tau(q_o)$ for any $q_o \in Q$ specifies τ completely.

PROOF. By assumption, any $q \in Q$ can be written as $q = \delta(q_o, \bar{x})$ for some $\bar{x} \in \bar{I}^*$. Therefore $\tau(q) = \tau[\delta(q_o, \bar{x})] = \bar{\delta}[\tau(q_o), \bar{x}]$.

The above lemmas can be found, under a slightly different form, in the references mentioned. Let us now introduce some lemmas that can be used in the heuristic approach for finding state table automorphisms.

LEMMA. Consider a state machine $M = \langle I, Q, \delta \rangle$ and let $I = I_1 \cup I_2 \cup \ldots \cup I_k$. If we define $M_j = \langle I_j, Q, \delta \mid Q \times I_j \rangle$, then the automorphism group A_M is given by $A_M = A_{M_1} \cap A_{M_2} \cap \ldots \cap A_{M_k}$.

PROOF. Obvious.

This last lemma is very important, since it allows us to consider separate columns of the state table ($I_j = \{i_j\}$) and thus reduces the problem of finding the automorphism group of the state table to finding the group for each single column separately, and then taking the intersection.

LEMMA. If $i \in I$ is a permutation input, i.e., the map $\pi_i : Q \to Q$, defined by $\pi_i(q) = \delta(q, i)$ is bijective, then π is an automorphism for the ith column of the state table which represents the state machine $<\{i\}, Q, \delta|_{Q \times \{i\}}>$.

PROOF. $\delta(q, i) = \pi_i(q)$ for all q in Q. Thus, $\delta[\pi_i(q), i] = \pi_i[\pi_i(q)] = \pi_i[\delta(q, i)]$ for all q in Q.

COROLLARY. If the permutation π_i in the above lemma has a single cycle, then the automorphism group for the ith column is the cyclic group generated by π.

PROOF. For any q_o, $q \in Q$ we can write $q = \pi^k(q_o)$ for some $k \in N$ (nonnegative numbers). Thus, the machine is strongly connected by $q = \bar{\delta}(q_o, i^k)$, and any arbitrary τ of the state table is then completely specified by $\tau(q_o)$. Now the powers π^k of I-r assign to q_0 successively all elements q of Q if k ranges over 0, 1, ... |Q I-1. Therefore τ must be one of these powers.

<u>Remark</u>. In case π is not a single cycle, the situation is more complicated. Let us only note here that elements of a given cycle can be mapped to elements of another cycle only in case the cycles have the same length.

LEMMA. If, in a given column of the state table, an element q_k occurs with multiplicity m_k, and there exists an automorphism that maps q_k into q_ℓ, we must have $m_k = m_\ell$.

PROOF. Obvious.

Application to heuristics

The two main cases considered are mutually exclusive: either an input i is a permutation input, or in the corresponding column some q_k must occur with multiplicity $m_k > 1$. Note further the duality between length of a cycle and multiplicity of a state.

The lemmas above aid very much in a fast visual check for automorphisms. Needless to say, they can also be incorporated in algorithms,

Finally, it is clear that state tables with automorphisms are the exception rather than the rule.

C. Examples of Output Equivalent Faults

1) Consider a machine M with output faults F_1 and F_2 yielding the machines $F_1(M)$ and $F_2(M)$ shown below.

$F_1(M)$

q	0	1	
A	D,0	J,0	
B	D,0	K,1	
C	A,1	L,1	
D	L,1	H,0	
E	C,0	D,0	
F	E,0	B,1	
G	E,1	A,1	$\delta(q,i),\lambda_1(q,i)$
H	I,1	E,0	
I	G,1	C,0	
J	H,0	G,0	
K	H,1	F,0	
L	J,0	I,1	

$F_2(M)$

q	0	1	
A	D,0	J,0	
B	D,1	K,0	
C	A,0	L,1	
D	L,1	H,0	
E	C,1	D,0	
F	E,0	B,1	
G	E,0	A,0	$\delta(q,i)\lambda_2(q,i)$
H	I,0	E,0	
I	G,1	C,1	
J	H,1	G,1	
K	H,0	F,1	
L	J,1	I,0	

$F_1(M)$ is reduced as can be seen as follows. Consider the equivalence relation defined by:

$$q_1 \equiv q_2(\pi'), \text{ if and only if } \lambda_1(q_1, x) = \lambda_1(q_2, x) \forall x \in I = \{0,1\}$$

then $\pi' = \overline{AJE}, \overline{FLB}, \overline{CG}, \overline{HKDI}$

under input 0 $\overline{DHC}, \overline{EJD}, \overline{AE}, \overline{IHLG}$

1 $\overline{JGD}, \overline{BIK}, \overline{LA}, \overline{EFHC}$ } shows next states, grouped as parts of blocks of π'

$F_1(M)$ and $F_2(M)$ are equivalent, as can be seen from direct product (Hennie [19], p. 25): denote states of $F_2(M)$ with a prime and consider a 24-state machine?

$\pi = \overline{AG'JA'EH'}, \overline{FK'LC'BF'}, \overline{CI'GJ'}, \overline{HD'KB'DE'IL'}$

input 0 $\overline{DE'HD'CI'}, \overline{EH'JA'DE'}, \overline{AG'EH'}, \overline{IL'HD'LC'GJ'}$

input 1 $\overline{JA'GJ'DE'}, \overline{BF'IL'KB'}, \overline{LC'AG'}, \overline{EH'FK'HD'GI'}$

We obtain: $\{\overline{AG'}\ \overline{JA'}\ \overline{EH'}\ \overline{FK'}\ \overline{LC'}\ \overline{BF'}\ \overline{CI'}\ \overline{GJ'}\ \overline{HD'}\ \overline{KB'}\ \overline{DE'}\ \overline{IL'}\} = \pi_R$ †

This represents the relation R, giving directly the automorphism φ

q	A	B	C	D	E	F	G	H	I	J	K	L
$\varphi(q)$	G	F	I	E	H	K	J	D	L	A	B	C

This is, of course, not the only one, since

$$\delta[\varphi^n(q), i] = \varphi\ \delta[\varphi^{n-1}(q), i] = \ldots = \varphi^n\delta(q, i) \quad \forall\ q \in Q,\ i \in I,\ n \in Z+$$

but it is uniquely specified by λ_1 and λ_2 as obtained by the above procedure

----

* The states are ordered in such a way that the partitions with substitution property can be easily recognized

† π_R: see Hartmanis and Stearns [12], pp. 55-56

2) Consider the machine M such that

M_q =

--	0	1
A	A	B
B t	A	B

Clearly, there exists no nontrivial automorphisms. According to the theorem, for no reduced $F_i(M)$ does there exist on $F_j(M)$ such that $F_i(M) \equiv F_j(M)$. Consider all reduced $F_i(M)$ possible (write only λ_i):

λ_1:

	0	1
A	0	0
B	0	1

λ_4:

	0	1
A	0	1
B	0	0

λ_7:

	0	1
A	1	1
B	0	0

λ_{10}:

	0	1
A	1	0
B	0	0

λ_2:

	0	1
A	0	0
B	1	0

λ_5:

	0	1
A	0	1
B	1	0

λ_8:

	0	1
A	1	1
B	0	1

λ_{11}:

	0	1
A	1	0
B	0	1

λ_3:

	0	1
A	0	0
B	1	1

λ_5:

	0	1
A	0	1
B	1	1

λ_9:

	0	1
A	1	1
B	1	0

λ_{12}:

	0	1
A	1	0
B	1	1

We then obtain for the product machine:

$$\pi = \overline{A_1A_2A_3B_4B_7B_{10}},\ \overline{A_4A_5A_6B_1B_8B_{11}},\ \overline{A_7A_8A_9B_3B_6B_{12}},\ \overline{A_{10}A_{11}A_{12}B_2B_5B_9}$$

input 0: $\overline{A_1A_2A_3}\,\overline{A_4}\,\overline{A_7}\,\overline{A_{10}},\ \overline{A_4A_5A_6}\,\overline{A_1}\,\overline{A_8}\,\overline{A_{11}},\ \overline{A_7A_8A_9}\,\overline{A_3}\,\overline{A_6}\,\overline{A_{12}},\ \overline{A_{10}A_{11}A_{12}}\,\overline{A_2}\,\overline{A_5}\,\overline{A_9}$

input 1: $\overline{B_1}\,\overline{B_2}\,\overline{B_3}\,\overline{B_4B_7B_{10}},\ \overline{B_4}\,\overline{B_5}\,\overline{B_6}\,\overline{B_1B_8B_{11}},\ \overline{B_7}\,\overline{B_8}\,\overline{B_9}\,\overline{B_3B_6B_{12}},\ \overline{B_{10}}\,\overline{B_{11}}\,\overline{B_{12}}\,\overline{B_2B_5B_9}$

$\pi_R = \{\overline{A_1}, \overline{A_2}, \ldots, \overline{B_{12}}\}$. So no two machines are equivalent; there exist not even two equivalent states in product machine.

This last remark illustrates the following.

THEOREM. (Gill [20]) If Ml and M_2 are strongly connected and non-equivalent, then no state in Ml is equivalent to any state in M_2.

The proof is very simple, so we omit it.

Observe the fact that M is strongly connected in this example.

D. Case where the Faulty Machines Are Not Reduced

ANTITHEOREM. The theorem does not hold in case $F_1(M)$ is not reduced.

PROOF. The theorem breaks down in part 3). (It is not interesting to consider $\lambda_1 = \lambda_2$ = trivial case, since then state transitions do not influence output.) A nontrivial case is $(\lambda_1 \neq \lambda_2)$

	0	1
A	C,0	A,1
B	C,0	A,1
C	A,1	C,1

$F_1(M)$

	0	1
A	C,1	A,1
B	C,1	A,1
C	A,0	C,1

$F_2(M)$

The machines $F_1(M)$ and $F_2(M)$ are equivalent. This can be seen from the equivalences described by the following pairs of states denoted by $< q_1, q_2 >$, where ql is a state of $F_1(M)$ and q_2 is a state of $F_2(M)$: $< A,C >$, $< B,C >$, $< C,A >$, and $< C,B >$.

There is no nontrivial automorphism for this machine M.

Before we investigate what happens in the case where $F_1(M)$ is not reduced (therefore, $F_2(M)$ is not reduced either), we recall [12]: a partition π on Q has substitution property if and only if $q_1 \equiv q_2(\pi)$ implies that $\delta(q_1, i) \equiv \delta(q_2, i)$ for all i in I. We denote the set of partition blocks by Q_π.

If π is a partition with substitution property, then the <u>π-image</u> of M is the state machine $M_\pi = \langle I, Q_\pi, \delta_\pi \rangle$, where, for all $B_\pi \in Q_\pi$,

$$\delta_\pi(B_\pi, i) = B'_\pi \text{ if and only if}$$

$$\delta(B_\pi, i) \subseteq B'_\pi .$$

THEOREM. In case a machine M has no two partitions π_1 and π_2 with substitution property such that $\pi_1 \neq \pi_2$ but M_{π_1} is isomorphic to M_{π_2}, then the equivalence of two faults F_1 and F_2 implies the existence of a unique (in the same sense as before) automorphism for the state table of the reduced machine of $F_1(M)$. The automorphism is nontrivial in case $\lambda_{F_1} \neq \lambda_{F_2}$.

PROOF. The equivalence between $F_1(M)$ and $F_2(M)$ implies the equivalence of the corresponding reduced machines, denoted by M_1 and M_2, which are, therefore, isomorphic.

The states of the reduced machines are blocks of some partitions π_1 and π_2, both with substitution property. From the assumption, it follows that $\pi_1 = \pi_2 = \pi$ and therefore, Ml and M_2 must have the same state set Q_π.

Further, $\lambda_{F_1} \neq A_{F_2}$ implies also that the reduced machines have different output functions since they have the same state set Q_π and, for all $q_1, q_2 \in B_\pi \in Q_\pi$, $\delta(q_1, i) . \delta(q_2, i)$.

Thus we are reduced to the situation of the previous theorems.

Strongly Connected Machines

1) For a long time we conjectured that there exists no strongly connected machine with two different substitution-property partitions, π_1 and π_2, such that $M_{\pi_1} \cong M_{\pi_2}$. In case M_{π_1} (thus also M_{π_2}) has only two states, this conjecture holds, according to a private communication from J. Ullman. In general, however, it is false, as can be illustrated by the example of Table I.

Table I.

M	0	1
1	6	3
2	8	4
3	4	8
4	5	7
5	4	8
6	1'	2
7	1	2
8	2	1

$$\pi_1 = \{ \overline{12},\ 3457\ \overline{678} \} = \{ B_1, B_2, B_3 \}$$

$$\pi_2 = \{ \overline{3567}, \overline{28}, \overline{14} \} = \{ B_1', B_2', B_3' \}$$

M_{π_i}	0	1
B_1^*	B_3^*	B_2^*
B_2^*	B_2^*	B_3^*
B_3^*	B_1^*	Bl*

2) Note that M_{π_1} does not have a nontrivial automorphism in the above example. However, it seems to be true that the existence of two different substitution-property partitions π_1 and π_2 with $M_{\pi_1} \cong M_{\pi_2}$ implies the existence of a nontrivial automorphism for M_{π} (where $\pi = \pi_1 \cdot \pi_2$) that can be found according to the following procedure:

Assume, without loss of generality, that the blocks $\{B_i\}$ of π_1 and $\{B_i'\}$ of π_2 are indexed in such a manner that B_i' corresponds to B_i in the isomorphism $M_{\pi_1} \cong M_{\pi_2}$. Since $\pi_1 \neq \pi_2$, there exists at least on pair of blocks B_j and B_j' such that $B_j \neq B_j'$. Therefore one of these two blocks contains a state q_o that does not belong to the other one*; without loss of generality we can assume $q_o \in B_j'$ but $q_o \notin B_j$. Further, $q_o \in B_k$ for some $k \neq j$.

Now define a map φ on the state set Q_π of M_π as follows: let D be a block of Q_π such that $D \subseteq B_j$ and define $\varphi(D) = B_\pi(q_o)$. Since M, and thus M_π, is strongly connected, for every $D' \in Q_\pi$ there exists a sequence $\bar{x} \in I^*$ such that $D' = \delta_\pi(D, \bar{x})$. Thus we define:

$$\varphi(D') = \delta_\pi[\varphi(D), \bar{x}] .$$

Our conjecture is now that φ is well defined, i.e., if $\bar{x}_1$ and $\bar{x}_2$ are two sequences satisfying $\delta_\pi(D, \bar{x}_1) = \delta_\pi(D, \bar{x}_2)$, then we have $\delta_\pi[B_\pi(q_o), \bar{x}_1) = \delta_\pi[B_\pi(q_o), \bar{x}_2)$. Once this fact is established, it is easily shown that φ is an automorphism; let indeed D' be any element of Q_π; then we have for all $\bar{x}' \in I^*$:

- - - - - -

* In fact, one can show that $B_j \cap B_j' \neq \emptyset$ implies $\pi_1 = \pi_2$

$$\varphi[\delta_\pi(D', \bar{x}')] = \varphi[\delta_\pi(D, \overline{xx}')]$$

$$= \delta_\pi[\varphi(D), \overline{xx}'] \quad \text{(definition of cp and conjecture)}$$

$$= \delta_\pi\{\delta_\pi[\varphi(D), \bar{x}], \bar{x}'\}$$

$$= \delta_\pi[\varphi(D'), \bar{x}']$$

where $\bar{x}$ is the sequence leading from D to D'. Note here that cp is necessarily nontrivial because $q_o \notin D$ and thus $\varphi(D) = B_\pi(q_0) \neq D$. Further, φ is not always unique, since it depends on the choice of D and q_o.

3) Example.

Using the machine described by Table I, where:

$$\pi = \pi_1 \cdot \pi_2 = \{ \bar{1}, \bar{2}, \overline{35}, \bar{4}, \overline{67}, \bar{8} \} = \{ D_1, \ldots, D_6\}$$

we obtain the following automorphisms for M_π:

B_j	D	q_o	$\varphi(D_1)$	$\varphi(D_2)$	$\varphi(D_3)$	$\varphi(D_4)$	$\varphi(D_5)$	$\varphi(D_6)$
B_1	D_1	3	D_3	D_5	D_6	D_2	D_4	D_1
B_1	D_1	6	D_5	D_3	D_2	D_6	D_1	D_4
B_1'	D_3	1	D_6	D_4	D_1	D_5	D_2	D_3

There are, in fact, two more nontrivial automorphisms for M_π, namely $\varphi(D_1) = D_2$ and $\varphi(D_1) = D_4$, but these cannot be obtained from the above procedure since 1 and 2 appear in the same block of B_1' and 1 and 4 in the same block of B_3'.

$\pi = \pi_1 \cdot \pi_2$

M_π	0	1
D_1	D_5	D_3
D_2	D_6	D_4
D_3	D_4	D_6
D_4	D_3	D_5
D_5	D_1	D_2
D_6	D_2	D_1

4) Conclusion

If the conjecture stated above holds, then we can conclude that, for strongly connected machines, equivalence of two output faults, Fl and F_2, with $A_{F_1} \neq \lambda_{F_2}$ always implies a nontrivial automorphism for the state table of M_π for some **substitution-property** partition π. Here π may be the trivial partition $\pi = 0$ where the blocks are the states of M.

V. EQUIVALENCE AND DOMINANCE

A. Combinational Circuits

LEMMA 1. Dominance induces partial ordering on the equivalence classes with respect to detection equivalence.

PROOF. The proof follows from the correspondence between faults and detection sets:

$$F_1 \geq F_2 \Leftrightarrow S_{F_1} \supseteq S_{F_2}$$

LEMMA 2. For single-output, irredundant networks, $F_1 \sim F_2$ iff $F_1 \simeq F_2$.

PROOF. Assume $F_1 \sim F_2$ and let $x \in S_{F_1}$; then, $\Phi_{F_1}(x) = \Phi_{F_2}(x) \neq \Phi_\lambda(x)$ so $x \in S_{F_2}$ and $S_{F_1} \subseteq S_{F_2}$. Similarly, $S_{F_2} \subseteq S_{F_1}$, so $S_{F_1} = S_{F_2}$ and $F_1 \simeq F_2$. Next assume $F_1 \simeq F_2$ and consider $x \in \{0,1\}^n$ (any x)

either $x \notin S_{F_1} = S_{F_2}$ then $\Phi_{F_1}(x) = \Phi_\lambda(x) = \Phi_{F_2}(x)$

or $x \in S_{F_1} = S_{F_2}$ then $\Phi_{F_1}(x) \neq \Phi_\lambda(x)$

$$\Phi_{F_2}(x) \neq \Phi_\lambda(x)$$

But, for single output, $\Phi_F(x) \in \{0,1\}$ which forces $\Phi_{F_1}(x) = \Phi_{F_2}(x)$.

LEMMA 3. The above does not hold, in general, for multiple-output networks.

PROOF. The circuit in Fig. 3 realizes

$$y_1 = x_1 \cdot x_2$$
$$y_2 = x_1 \oplus x_2$$

Referring to this figure, let

F_1 = "A stuck at 0"

F_2 = "B stuck at 1" .

Then the only test for F_1, as well as for F_2, is $x_1 = x_2 = 1$.

In the case of F_1 we get $\begin{cases} y_1 = 0 \\ y_2 = x_1 + x_2 \end{cases}$

F_2 we get $\begin{cases} y_1 = x_1 \cdot x_2 \\ y_2 = x_1 + x_2 \end{cases}$

i.e., $\Phi_{F_1} \neq \Phi_{F_2}$.

So in this case $\begin{cases} S_{F_1} = S_{F_2} = \{\langle 1,1 \rangle\} \Rightarrow F_1 \simeq F_2 \\ \Phi_{F_1} \neq \Phi_{F_2} \Rightarrow F_1 \not\sim F_2 \end{cases}$.

Remarks.

1) Clearly $F_1 \sim F_2 \Rightarrow F_1 \simeq F_2$ (in proving sufficiency of lemma 2, a single-fault assumption was not used)

2) Conjectures: (multiple output)

---if reconvergent fanout, every circuit has faults exhibiting lemma 3

---if no reconvergent fanout, lemma 2 holds always.

B. Sequential Circuits

LEMMA 1'. Dominance induces partial ordering on the set of equivalence classes with respect to detection equivalence.

LEMMA 2'. If two faults are reset equivalent, they are also detection equivalent.

Proofs are analogous to the combinational case.

LEMMA 3'. Detection equivalence does not always imply reset equivalence.

There are necessary and sufficient conditions for detection equivalence to imply reset equivalence, but they will not be discussed here.

PROOF. Consider the machine M and its fault versions as described below.

Let $|I_R| = 1$ and $\rho(i) = A \qquad (i \in I_R)$

$$\rho_{F_1}(i) = B$$

$$\rho_{F_2}(i) = A$$

Consider:

M

	0	1
A	B,0	C,1
B	A,1	C,0
C	D,1	E,1
D	A,0	C,1
E	D,0	E,0

$F_1(M)$

	0	1
A	B,1	C,0
B	A,0	C,1
C	D,0	E,0
D	B,1	B,0
E	D,0	A,1

$F_2(M)$

	0	1
A	D,0	E,1
B	A,0	C,1
C	D,1	C,0
D	A,1	E,0
E	D,0	C,0

Using the method described by Poage and McCluskey [4], we obtain:

	0	1
A B A	B A D	C C E
	0 0 0	1 1 1
B A D	A B A	C C E
	1 1 1	0 0 0
C C E	D x x	E x x
	1 - -	1 - - s
D x x	A x x	C x x
	0 - -	1 - -
E x x	D x x	E x x
	0 - -	0 - -

From this table it is clear that $S_{F_1} = S_{F_2}$. Nevertheless, if we apply the input sequence:

$$< reset > , 0, 1, 1, 0, 1$$

we get as outputs: $F_1(M)$: 0 0 0 0 0
$F_2(M)$: 0 0 0 1 0

so clearly $F_1(M)$ and $F_2(M)$ are <u>not</u> reset equivalent. So $F_1 \simeq F_2$ but $F_1 \not\simeq F_2$.

VI. REMARKS ON MEMORY FAULTS

In this section we discuss the influence of stuck-at faults in the feedback lines of a sequential circuit on the behavior of the machine. In particular, it will be shown that, for some machines, the fundamental properties can be very drastically changed, thus adding extra complexity to the problem.

Consider now the fault F as defined in section III under 'memory faults". On the set $\mathfrak{M}_m$ the fault F defines a map which we denote by the same symbol as the map defined by (3).

$$F: \mathfrak{M}_m \rightarrow \mathfrak{M}_h \quad (h = m - |T_F|)$$

$$M = \langle I, \Theta, Q, \delta, \lambda \rangle \mapsto \langle I, \Theta, Q_F, \delta_F, \lambda_F \rangle = F(M)$$

where $F(M)$ is defined as the result of the following transformation on the model of M (Fig. lb):

a) delete all $\ell_i \ni F(\ell_i) \neq N$ and corresponding memory elements (delays)

b) for each of these ℓ_i, the corresponding input to the combinational logic is fixed at the value $F(\ell_i)$ and the combinational function is redefined on the remaining variables only.

The set Q_F is obtained from Q by deleting in each $q = \langle y_1, y_2, \ldots, y_m \rangle \in Q$ those y_i that correspond to $F(\ell_i) \neq N$. The corresponding map $c_F: Q \rightarrow Q_F$ will be called "fault projection function."

We also define a map $r_F: Q \rightarrow Q$

$$r_F: \langle y_1, y_2, \ldots, y_m \rangle \mapsto \langle y_1', y_2', \ldots, y_m' \rangle$$

as follows: $y'_i = y_i$ in case $F(\ell_i) = N$

$y'_i = F(\ell_i)$, otherwise.

From this it is possible to find a bijective map $b_F: Q_F \to r_F(Q) \subseteq Q$ (5)

such that the diagram of Fig. 4a commutes. The reader can consult Arbib [11] for the concept, "commuting diagram".

It is also easy to verify that b_F is unique. In fact, b_F does nothing else than insert the "defective" y_i with the corresponding values of $F(\ell_i) \neq N$ in the representation of $q = \langle \ldots \rangle \in Q_F$.

The maps b_F and c_F will be used throughout this section.

DEFINITION. A machine $M = \langle I, 0, Q, 6, \lambda \rangle$ <u>describes</u> a machine $M' = \langle I', O', Q', \delta', \lambda' \rangle$ iff there exist maps:

$h_1: Q' \to Q$
$h'_1: Q \to Q'$
$h_2: I' \to I$
$h'_3: 0 \to 0'$

such that $\delta'(q', i') = h'_1\{\delta[h_1(q'), h_2(i')]\}$

$\lambda'(q', i') = h_3\{\lambda[h_1(q'), h_2(i')]\}$,

i.e., the diagrams of Fig. 4b commute.

THEOREM 1. M describes $F(M)$

PROOF. If the machine F(M) is in state q', then the machine M in presence of the fault F is in the state q_B defined by $q_B = b_F(q')$ according to the definition (5)of b. The next state of M is then $\delta[b_F(q'), i]$ and for $F(M)$ this is thus $c_F\{\delta[b_F(q'), i]\}$.

We obtain $\delta_F(q', i) = c_F\{\delta[b_F(q'), i]\}$ (6a)

Similarly $\lambda_F(q', i) = \lambda[b_F(q'), i]$ (6b)

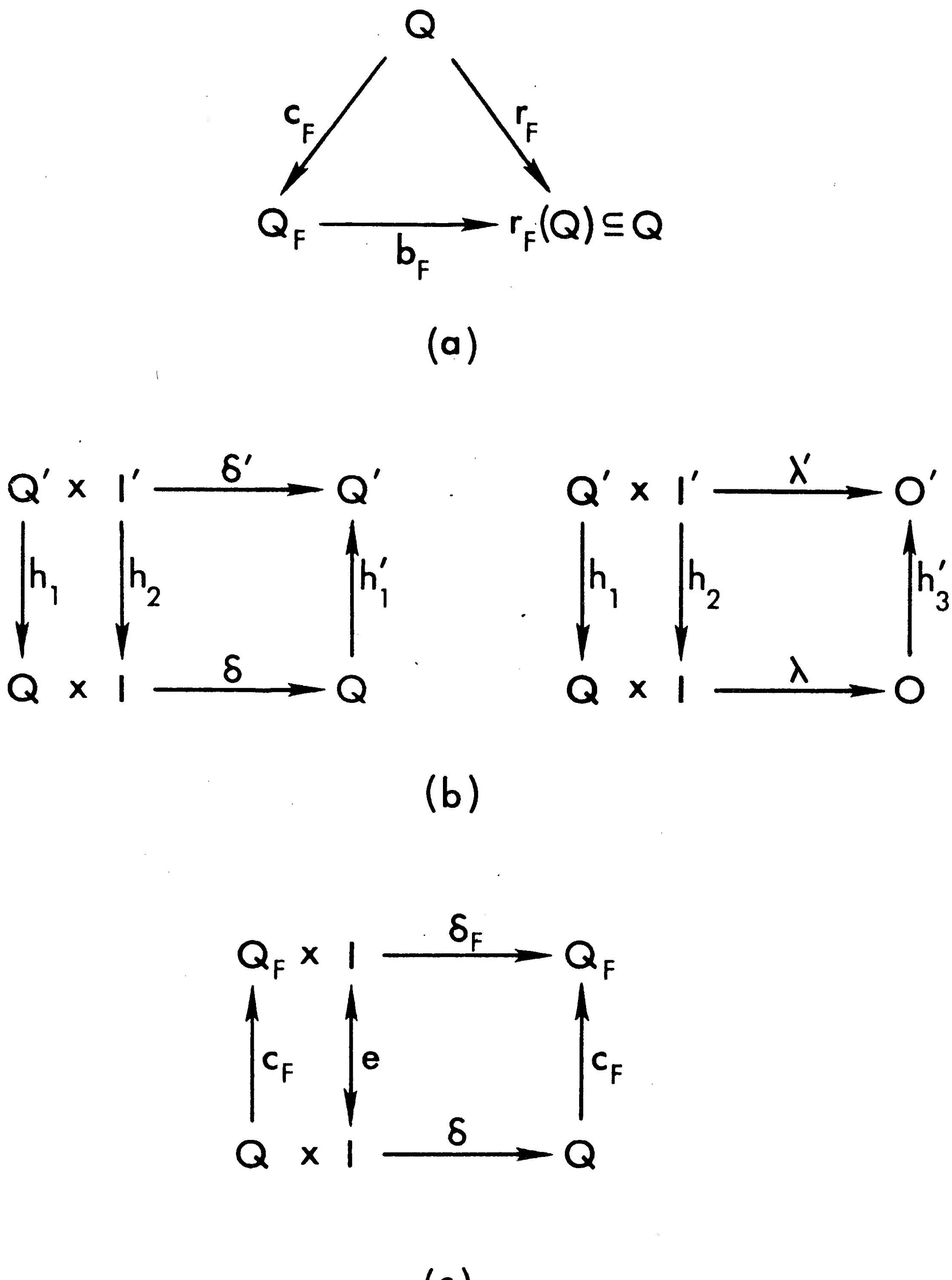


Fig. 4: (a) To define b_F; (b) A machine M describing M';
(c) To illustrate when $F(M)$ is the homomorphic image of M

Taking now
$$h_1 = b_F$$
$$h_1' = c_F$$
$$h_2 = \text{identity map on } I$$
$$h_3' = \text{identity map on } O ,$$

we see that M describes F(M) in the sense defined above.

Remark. The relationship between M and F(M) is not, in general, a homomorphism since the diagram of Fig. 4c does not, in general, commute. It can be shown that the diagram commutes in case the "fault partition" π_F, defined by $q_1 \equiv q_2(\pi_F)$ if and only if $c_F(q_1) = c_F(q_2)$, has substitution property.

EXAMPLE. $F(\ell_3) = 0$ $F(\ell_1) = F(\ell_2) = N$ $(a = \{0,1\},\ m = 3)$

q $\langle y_1\ y_2\ y_3\rangle$	$\delta(q, i_0)$ $\langle y_1, y_2, y_3\rangle$	$c_F(q)$ q'	$b_F(q')$ q_B
0 0 0	1 0 0	00	000
0 0 1	1 1 0		
0 1 1	1 0 1	01	010
0 1 0	0 1 0		
1 1 0	0 1 0	11	110
1 1 1	1 1 0		
1 0 1	0 0 1	10	100
1 0 0	1 1 1		

This table yields, using (6a), the state table for F(M):

q'	$\delta'(q', i_0)$
00	10
01	01
11	01
10	11

Let us now consider the influence of memory faults on the strongly connectedness of machines. The following point will be useful in proving a theorem on this subject.

For all m and all finite sets of the form $\sigma = (0, 1, \ldots, |\sigma| -1\}$ there exists a single cycle permutation ψ on $Q = \sigma^m$, with the property that q and $\psi(q)$ differ only in a single component y_i for any $q \in Q$.

We will not provide a proof for this. It is, in fact, a consequence of a stronger result, where ψ has to satisfy the extra requirement that the difference between the y_i that are unequal in q and $\psi(q)$ is always 1 modulo $|\sigma|$. See also the related material about "unit distance codes" in ref. [5].

The following example shows, for m = 3, one of the permutations ψ that satisfy this stronger requirement.

EXAMPLE. We describe ψ by arbitrarily taking $q_o = < 0\ 0\ 0 >$ and listing all $\psi^j(q_o)$ as j ranges over $0, 1, \ldots, |Q| -1$

$|\sigma|$ = 2 yields the table

j =	0	1	2	.	.	.		7
y_3	0	0	0	0	1	1	1	1
y_2	0	0	1	1	1	1	0	0
y_1	0	1	1	0	0	1	1	0

while $|\sigma|$ = 3 yields

j =	0	1	2	.	.	.	.	.	.	.																	26
y_3	0	1	2	2	1	0	0	1	2	2	1	0	0	0	1	2	2	1	1	1	2	2	2	1	0	0	0
y_2	0	0	0	1	1	1	2	2	2	2	2	2	1	0	0	0	1	1	1	0	0	1	2	2	2	1	0
y_1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2

It is useful to look at the paths $q_o, \psi(q_o), \ldots$ in 3-dimensional space, since it shows how these sequences are formed and how the fact stated above can be proven in general.

THEOREM 2.

1) For all m there exist a large class of strongly-connected machines such that every single fault $F \in \mathfrak{F}_Q$ destroys the strongly connectedness.

2) For all m there exist strongly connected machines M such that F(M) is strongly connected for all $F \in \mathfrak{F}_Q$.

PROOF.

Part 1 Let δ be defined as follows:

$$\delta(q, i) = \begin{cases} \psi(q) & \text{for some } i_q \in I \\ q & \text{for all other } i \neq i_q . \end{cases}$$

This corresponds to a very wide class of strongly-connected (ψ has a single cycle) machines. Let F be an arbitrary single fault defined by $F(\ell_k) = \alpha \in \sigma$.

The definition of ψ implies the existence of a state

$q = \langle y_1, \ldots, y_k = \alpha, \ldots, y_n \rangle$ such that

$\psi(q) = \langle y_1, \ldots, y_k' \neq \alpha, \ldots, y_n \rangle$. For F(M) this implies that

$c_F(q) = c_F\psi(q) = q' \in Q_F$ and $b_F(q') = q$.

The next-state function for the faulty machine gives, for q',

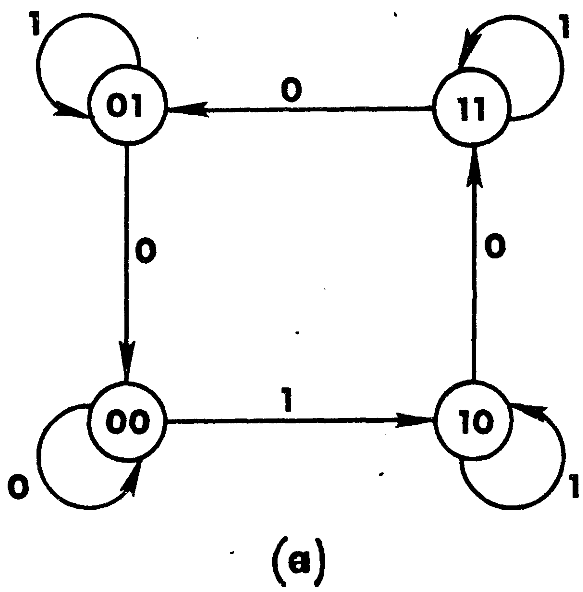
$$\begin{aligned} \delta_F(q', i) &= c_F\delta[b_F(q'), i] && \{\text{from (6a)}\} \\ &= c_F\delta(q, i) \\ &= \begin{cases} c_F(q) & \text{in case } i \neq i_q \\ c_F\psi(q) & \text{in case } i = i_q . \end{cases} \end{aligned}$$

Therefore, the only state reachable from $q' \in Q_F$ is q' itself, and therefore, F(M) is not strongly connected. This process is illustrated in Fig. 5 for $|\sigma| = 2$, $m = 2$, and y_2 stuck at zero.

Part 2. Consider a machine M with Q = I and $\delta(q, i) = i$ for all q and i in I. It is easily verified that M is strongly connected and also that, in F(M), any $q' \in Q_F$ can be reached from any other state of Q_F by applying an input satisfying $c,(i) = q'$.

Of course, many less trivial machines with this property can be found easily.

Consequences for the study of equivalence and dominance. This theorem clearly illustrates the need for having several reset states available in order to have a sound basis for comparing faulty machines. It would be inaccurate to say that two faulty machines are (reset) equivalent based on only one single reset state from which only a small portion of the actual states can be reached.



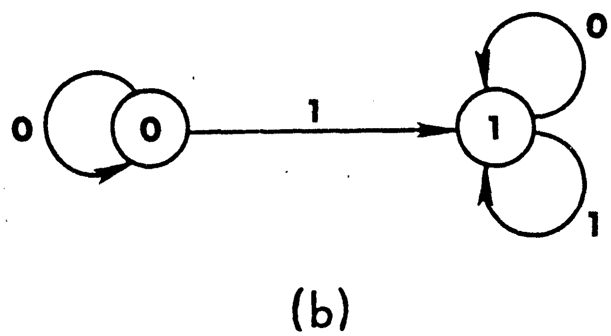


Fig. 5: To illustrate Theorem 2 of Section VI.

(a) A strongly connected machine M

(b) The machine F(M) after the fault F = (line ℓ_2 stuck at 0) occurred.

Note that the input labeling in M can be changed arbitrarily as long as the graph remains deterministic.

VII. CONCLUSION

From this paper, three main ideas have become clear. First, the possibility that output faults are equivalent, without being functionally equivalent for the output function, can be investigated directly from the structure of the state table only. This gives certain information on the equivalence classes.

Second, it has been shown for both combinational (mainly to demonstrate the parallel) and sequential networks that reset equivalence and detection equivalence are not always the same thing.

Finally, the invalidity of the assumption that a strongly connected machine is still strongly connected after a fault occurs has been demonstrated by exhibiting some large class of machines for which it is clearly false. Another consequence is the need to have a reset circuitry in order to be able to compare such machines in a meaningful way.

It has also become apparent that extreme caution is needed when comparing machines, i.e., the basis for comparison must be specified very precisely, and as insensitive to faults as possible.

Further research is being done to find conditions under which faulty machines are equivalent under faults of a certain class. Also, the relationship between reset and detection equivalence is being further investigated.

REFERENCES

[1] E. F. Moore, "Gedanken Experiments on Sequential Machines," Automata Studies, No. 34, pp. 129-133, Princeton University Press, Princeton, N. J. (1956).

[2] S. Seshu and D. N. Freeman, "The Diagnosis of Asynchronous Sequential Switching Systems," I.R.E. Transactions on Electronic Computers, Vol. EC 11, No. 4, pp. 459-465 (Aug. 1962).

[3] F. C. Hennie, "Fault Detecting Experiments for Sequential Circuits," Proc. 5th Ann. Sympos. on Switching Theory and Logical Design, pp. 95-110, Princeton, N. J. (Nov. 1964).

[4] J. F. Poage and E. J. McCluskey, "Derivation of Optimum Test Sequences for Sequential Machines," Proc. 5th Ann. Sympos. on Switching Theory and Logical Design, pp. 121-132 (1964).

[5] E. J. McCluskey, Introduction to the Theory of Switching Circuits, McGraw-Hill, N. Y. (1965).

[6] F. W. Clegg and E. J. McCluskey, "Algebraic Properties of Faults in Logic Networks," Technical Report No. 4, Digital Systems Laboratory, Stanford Electronics Labs, Stanford, California (March 1970).

[7] F. W. Clegg and E. J. McCluskey, "The Algebraic Approach to Faulty Logic Networks," IEEE 1971 International Symposium on Fault-Tolerant Computing, pp. 44-45, Pasadena, California (March 1971).

[8] D. R. Schertz, "On the Representation of Faults," Report R-418, Coordinated Science Laboratory, University of Illinois (1969).

[9] D. R. Schertz and G. Metze, "On the Indistinguishability of Faults in Digital Systems," Proc. 6th Ann. Allerton Confer. on Circuit and Syst. Thy., pp. 572-760 (1968).

[10] K. C. Y. Mei, "Fault Dominance in Combinational Circuits," Technical Note No. 2, Digital Systems Laboratory, Stanford Electronics Labs, Stanford University, California (August 1970).

[11] M. A. Arbib, Theories of Abstract Automata, Prentice-Hall (1969).

[12] J. Hartmanis, R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice-Hall (1966).

[13] H. X. Chang, E. G. Manning, G. Metze, Fault Diagnosis of Digital Systems, Wiley (1970).

[14] G. P. Weeg, "The Structure of an Automaton and its Operation-Preserving Transformation Group," J.A.C.M., 9, No. 3, p. 345 (1962).

[15] A. C. Fleck, "Isomorphism Groups of Automata," J.A.C.M., 9, No. 4, p. 469 (1962).

[16] A. C. Fleck, "On the Automorphism Group of an Automaton," J.A.C.M., 12, No. 4, p. 566 (1965).

[17] B. Barnes, "Groups of Automorphisms and Sets of Equivalence Classes of Input for Automata,' J.A.C.M., 12, No. 4, p. 561 (1965).

[18] H. O. Rabin and D. Scott, "Finite Automata and their Decision Problems," IBM Journal on Research and Development, No. 3, pp. 114-125 (1959).

[19] F. C. Hennie, Finite-State Models for Logical Machines, Wiley (1968).

[20] A Gill, Introduction to the Theory of Finite-State Machines, McGraw-Hill, N. Y. (1962).