An Algebra of Multiple Faults in RAMs[1]

J. A. Brzozowski,[2] H. Jürgensen[3]

Abstract: Cell array faults in random-access memories (RAMs) are usually represented by Mealy automata. In such a model, multiple faults should also be representable by automata; in fact, it should be possible to compute the automaton representing a multiple fault from the automata representing the single faults that make up the multiple fault. In this paper we study properties of binary composition operations on automata that are appropriate for the representation of multiple faults in RAMs. First, we derive a set of generic conditions that every composition operation must satisfy. Second, we develop a set of physical conditions that the composition must satisfy in order to apply to stuck-at, transition and coupling faults in RAMs. Third, we represent the transition table rules used by van de Goor and Smit by a composition operation and prove that this operation satisfies both the generic and physical conditions. Fourth, we point out that in some circumstances, it is appropriate to use a different composition operation (defined by us in a previous paper) to permit a different handling of coupling faults in the presence of stuck-at or transition faults. We compare and relate the properties of the two algebras.

Index Terms: Automaton, composition, coupling fault, multiple fault, pattern-sensitive fault, RAM, semilattice, stuck-at fault, testing, transition fault.

1. Introduction

We study multiple faults in random-access memories (RAMs). Most test generation algorithms target only single faults, because the detection of multiple faults is too complex, expensive, and time consuming. Nevertheless, it is important to understand multiple faults and their effects. A test designed to detect a given fault may fail to detect that fault in the presence of another fault. Even if we cannot afford to design tests for multiple faults, it would be useful to know which multiple faults

---

[1] This work was supported by the Natural Sciences and Engineering Research Council of Canada, Grants OGP0000871 and OGP0000243, and by a grant from the Information Technology Research Centre of Ontario. An extended summary of this paper appears in [5].

[2] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1.

[3] Department of Computer Science, The University of Western Ontario, London, Ontario, Canada, N6A 5B7.

are detected by a given test, so that the test with the highest overall fault coverage could be selected. To be able to evaluate and generate RAM tests for multiple faults automatically, we must make explicit and unambiguous the unstated assumptions regarding how multiple faults combine.

In this paper we consider memory cell array faults. In particular, we are concerned mainly, but not exclusively, with the classical faults in the Thatte-Abraham model [19], that is, with stuck-at, transition, and coupling faults. The usual model for such faults is the Mealy automaton model; for recent examples of the usefulness of this model in the representation of RAM faults see [1], [2], [4], [8], [9], [10], [12], [14], [15], [17], [18].

We address here the problem of characterizing multiple faults in RAMs. For simplicity, we assume that a RAM consists of n one-bit cells. Suppose that a physical defect D—for example, a short from the output of cell i to ground—exists in a RAM. Such a defect is usually represented at the logic level by an abstract fault F, in this case a stuck-at-0 fault on output i. Formally, fault F is represented by a Mealy automaton M, in which cell i can only have 0 as output, but the remaining cells operate correctly. The use of the automaton model permits us to be mathematically precise about the definition of a fault.

Suppose now that two defects D and D' are present in a RAM. For example, D may be a short to ground in cell i, whereas D' may be a coupling fault from cell j to cell k—say, a fault that causes a 1 to be written in both cells j and k when they store the value 0 initially and a 1 is written to cell j. Suppose further that automata M and M' represent defects D and D', respectively. It should be possible to represent the simultaneous presence of both defects by another automaton. What is needed is an *operation on automata,* say $\diamond$, such that $M \diamond M'$ would correctly represent the double fault. Although there are cases, such as the example above, where it is quite obvious how to construct the automaton for the double fault composed of M and M', in general this problem is rather complex.

A composition operation on RAM faults was first introduced in [1]. That definition was made in terms of a general class of so-called delta faults (defined also in this paper), but turned out to be applicable only to coupling faults in RAMs. The deficiency of this definition was noted in [8], when it was observed that this composition produced incorrect results for transition faults. A different operation was then defined in [8] for transition faults, and yet another one for stuck-at faults. The need to have three different operations—two of them rather awkward—is clearly undesirable. In [3] and [4] we introduced a single operation applicable to all three cases; we discuss this definition later. Another attempt to address part of this problem was made in [16] and [20]. A somewhat different approach was used in [14], [15], and [18], where a set of rules was defined for combining state tables of Mealy automata. That work describes a method for automatic verification of RAM tests. In particular, the method determines whether a test is complete and irredundant for a fault model.

In this paper we formulate abstract conditions that need to be satisfied by *any* composition operation on automata in order to model fault composition correctly. We then formalize the work in [14] and [15] by defining a composition operation that corresponds to the set of rules used by van de Goor and Smit. We show that this operation has the necessary properties to represent multiple faults, and we study the resulting algebra of multiple faults. We discuss two possible physical models for handling coupling faults in conjunction with stuck-at and transition faults, and we compare the model of [15] and the present paper with that of [4]. Additional mathematical properties of fault composition operations are discussed in [6].

The paper is structured as follows. In Section 2 we derive generic conditions that should be satisfied by any composition operation on faults. In Section 3 we state the assumptions we make concerning stuck-at, transition, and coupling faults. The basic Mealy automaton model, along with a related "component automaton" model, are defined in Section 4. The formal model of RAM faults is presented in Section 5. A composition operation on RAM faults is next defined in Section 6, and it is verified that the operation satisfies the generic conditions on fault composition. In Section 7 we show that our operation satisfies the physical conditions for the class of multiple faults generated by single stuck-at, transition, and coupling faults; we also study the algebra of these faults. In Section 8 a comparison is made of the algebra developed in this paper with that of [3], [4]. Section 9 concludes the main body of the paper. Some technical proofs are included in an appendix.

2. Generic Conditions on Fault Composition

The definition of a composition operation $\diamond$ has two quite different aspects, which we will call "generic" and "physical." In this section we motivate the generic conditions by some simple examples.

Suppose we have two defects represented by automata and we wish to model the combined effect of both defects being present in one circuit. In many cases the double defect can still be modelled by an automaton. Consider, however, the case in which the two defects are a short to ground from line i and a short to V_{DD} from line i. Then line i would be stuck-at-0 as well as stuck-at-1; this is an inconsistent situation at the logic level. There is no such conceptual difficulty at the physical level, because shorts are not ideal but resistive; consequently, in the presence of shorts from i to both ground and V_{DD}, the voltage at i would assume some intermediate value between V_{DD} and ground. In summary, two such defects can co-exist, but the resulting circuit can no longer be represented as a logic fault. One possible solution to this problem would be to develop a much more general multivalued, or perhaps even continuous, model for the circuits being considered. Such a model would be significantly more complex and is beyond the scope of our work. To handle this case we adopt a much simpler solution:

Hypothesis 1 *Whenever the presence of two defects, represented by automata M and M', leads to a circuit outside the automaton model, we consider these defects to be* incompatible *and we represent the resulting circuit by a special automaton* $\mathbf{0}$*, called* zero *(defined formally later); thus,*

$$M \diamond M' = \mathbf{0}.$$

If two defects together lead to a circuit outside our model, what happens when a third defect is added? While it may be conceivable that the presence of the three defects results in a circuit that is again inside our model, we consider it more realistic and safer to assume that the circuit is outside the model. Thus, we make the following assumption about $\mathbf{0}$:

Hypothesis 2 *The composition of any fault with a fault outside the automaton model is outside the model, that is,*

$$\mathbf{0} \diamond M = M \diamond \mathbf{0} = \mathbf{0}.$$

Under this hypothesis the composition of two automata representing any two faults is again an automaton, possibly the automaton $\mathbf{0}$ that represents the class of all faults that fall outside the model. Moreover, the automaton $\mathbf{0}$ behaves like a zero if $\diamond$ is viewed as a multiplicative operation.

The composition operation needs to satisfy further algebraic laws as justified by the following explanations.

First, suppose D is a short to ground from line i and D' is another short to ground from line i. If both D and D' are present in a circuit, the net effect is the same as if only one short to ground existed. Since D and D' are represented by the same automaton, it follows that the composite automaton should be exactly the same as the components. Thus, the composition operation should be *idempotent*, that is, it should satisfy the law

$$M \diamond M = M,$$

for all automata M.

Second, suppose D is a short to ground from line i and D' is a break in line j. Then the order in which we consider these defects when they are both present in the same circuit should be immaterial. Thus, the composition operation should be *commutative*, that is, it should satisfy the law

$$M \diamond M' = M' \diamond M,$$

for all automata M and M'.

Third, suppose D and D' are as in the paragraph above and D'' is a short to $V_{\rm DD}$ from line k. Then it should not matter how we compose these defects; the final result should be a circuit with all three defects. Thus, the composition operation should be *associative*, that is, it should satisfy the law

$$(M \diamond M') \diamond M'' = M \diamond (M' \diamond M''),$$

for all automata M, M', and M''.

An algebraic system $(S, \diamond)$, where S is any set and $\diamond$ is an associative binary operation on S is a *semigroup.* An idempotent and commutative semigroup is called a *semilattice* [7]. In algebraic terms, the considerations above lead to the following:

Hypothesis 3 *The set of all faults of a circuit (within a given fault model) together with a composition operation $\diamond$, is a semilattice, possibly with the zero element $\mathbf{0}$.*

In summary, any operation modelling the composition of faults at the logic level should have at least the properties of a semilattice operation. These properties provide a generic criterion for the acceptability of a proposed operation on a given class of faults.

3. Physical Conditions on RAM Fault Composition

In addition to the generic properties of fault composition, there are also conditions derived from the physical nature of defects. We state such conditions for the Thatte-Abraham fault model in this section. The same assumptions are made, though not explicitly, by van de Goor and Smit [15].

Our first condition states that stuck-at-0 and stuck-at-1 faults in the same cell are incompatible. A RAM with cell i stuck-at-a, where a is either 0 or 1, is represented by automaton $M^{i=a}$ (read "i stuck-at-a"). Let $\overline{a}$ denote the complement of a. In view of our handling of incompatible faults we have

Hypothesis 4

$$M^{i=a} \diamond M^{i=\overline{a}} = \mathbf{0}.$$

It turns out that all of our remaining conditions have the same form, namely

$$M \diamond M' = M.$$

In a semilattice, one can define a partial ordering corresponding to this type of equation. We say that fault type M is *stronger than or equal to* M', written $M \rhd M'$, if $M \diamond M' = M$. Therefore, the relation $\rhd$ is reflexive ($M \rhd M$), antisymmetric ($M \rhd M'$ and $M' \rhd M$ implies $M = M'$), and transitive ($M \rhd M'$ and $M' \rhd M''$ implies $M \rhd M''$).

If $M \triangleright M'$, this means that the effects of M' cannot manifest themselves when M is present as well.

As has been mentioned above, our main interest in this paper is in stuck-at, transition, and coupling faults in RAMs. We use the following notation, which is explained more fully later. Let a and b be binary values from the set $\{0, 1\}$, and let i and j be integers, $1 \leq i, j \leq n$, where n is the number of cells in the RAM. The notation for stuck-at faults was already introduced. A RAM unable to undergo a transition from $\overline{a}$ to a is represented by $M^{i\neg a}$ (read "i not to a"). A RAM with a coupling fault from *coupling cell* i started in state $\overline{a}$ to *coupled cell* j started in state $\overline{b}$ behaves as follows. When the value a is written in cell i, then cell j gets the value b. Such a fault is represented by $M^{ia \Rightarrow jb}$ (read "if i to a then j to b"). We make certain assumptions about the relative strengths of these three faults.

First, notice that a cell stuck-at-a can have only one state, namely a. On the other hand, transition fault $M^{i\neg b}$ can be in either state at power-up; after a $\overline{b}$ is written into cell i, subsequent writing of b into this cell fails. When both a transition fault and a stuck-at fault are present in one cell, we make the following assumption:

Hypothesis 5 *A stuck-at fault in a cell is stronger than a transition fault in that cell, that is,*

$$M^{i=a} \diamond M^{i\neg b} = M^{i=a}.$$

Next consider the situation where the first fault is $M^{i=0}$, and the second fault is $M^{j1 \Rightarrow i1}$. In the cell with both defects, the first defect tries to keep cell i at 0, whereas the second defect tries to change the state of cell i to 1 when a 1 is being written into cell j. We make the assumption that the stuck-at fault is stronger. In case the first fault is $M^{i=1}$ and the second is $M^{j1 \Rightarrow i1}$, cell i can never be in state 0, and so the coupling fault can never occur. These two cases can be summarized in general form as follows:

Hypothesis 6 *A stuck-at fault in a cell is stronger than any coupling fault in which that cell is the coupled cell, that is,*

$$M^{i=a} \diamond M^{jb \Rightarrow ic} = M^{i=a}.$$

Note, however, that a transition fault in a coupled cell does not prevent the coupled cell from changing [13], [15], that is coupling of the form $M^{ia \Rightarrow jb}$ *does* occur in the presence of transition fault $M^{j\neg b}$.

The next two hypotheses deal with coupling cells. Consider a coupling fault $M^{i1 \Rightarrow j1}$ and suppose that cell i cannot undergo a transition from 0 to 1. We assume that when writing into cell i is not successful, coupling into cell j does not take place. We thus obtain

Hypothesis 7 *A transition fault in a cell is stronger than any coupling fault in which that cell is the coupling cell, that is,*

$$M^{i\neg a} \diamond M^{ia \Rightarrow jb} = M^{i\neg a}.$$

By a similar reasoning we conclude that the coupling $M^{i1\Rightarrow j1}$ does not take place when cell i is stuck-at-0. In the case of the coupling fault $M^{i0\Rightarrow j0}$, cell i must start in state 1 for coupling to take place. This is impossible when cell i is stuck-at-0. Summarizing these two types of cases we get

Hypothesis 8 *A stuck-at fault in a cell is stronger than any coupling fault in which that cell is the coupling cell, that is,*

$$M^{i=a} \diamond M^{ib \Rightarrow jc} = M^{i=a}.$$

Here we note that there are physical defects for which Hypotheses 7 and 8 do not hold; this is discussed in detail in Section 8. Our objective in this paper is to define a binary operation on the set of RAM faults that satisfies Hypotheses 1–8. In [3], [4] we have defined an operation $\circ$ that obeys Hypotheses 1–6 but not 7 and 8. A comparison of the fault algebras resulting from these two operations is made in Section 8.

4. Component Automata

In this section, we introduce some basic definitions concerning Mealy automata, which constitute our basic formal model of RAM faults. Since the state of a RAM is always a vector with the cell states as components, it is appropriate to use a special type of automata called component automata. These automata are also defined here.

A *finite deterministic Mealy automaton*, called simply a *Mealy automaton* or *automaton* in the sequel, is a quintuple $M = (Q, X, Y, \delta, \lambda)$, where Q is a finite nonempty set called the *state set*, X is a finite nonempty set called the *input alphabet*, δ is a function, called the *transition function*, from $Q \times X$ to Q, Y is a finite nonempty set called the *output alphabet*, and λ is a function, called the *output function*, from $Q \times X$ to Y.

A *component automaton* is a deterministic Mealy automaton $M = (Q, X, Y, \delta, \lambda)$, where Q has the following special property: There exists an integer $n > 0$ and n finite nonempty sets $Q_1, \ldots, Q_n$ such that $Q \subseteq Q_1 \times \ldots \times Q_n$. Thus each state $q \in Q$ is an n-tuple $q = (q_1, \ldots, q_n)$, where $q_i \in Q_i$ for $i = 1, \ldots, n$. The *empty component automaton* is denoted by $\mathbf{0}$ and has empty transition and output functions, that is,

$$\mathbf{0} = (\emptyset, X, Y, \emptyset, \emptyset).$$

A component automaton is called *binary* if $Q_i \subseteq \{0,1\}$ for all i. In this paper we consider only binary component automata[4].

An automaton or component automaton $M = (Q, X, Y, \delta, \lambda)$ together with an initial state $q^0 \in Q$ is called an *initialized automaton (initialized component automaton)*.

We assume that each RAM under test starts in some unknown initial state. A fault-free RAM is represented by an initialized component automaton M. A faulty RAM is called a *fault* and is also represented by an initialized component automaton. For the purposes of this paper, however, we do not require initialized automata, since we can handle all initialized versions at once. Consequently, we introduce the concept of "type." A RAM or fault without an initial state is called a *RAM type* or *fault type*.

Stuck-at, transition, and coupling fault types in RAMs all belong to a more general class of fault types, called "delta faults." For this reason we associate with the fault-free component automaton M a family of component automata $M' = (Q', X, Y, \delta', \lambda')$, called *delta fault types* of M, which can differ from M in the state set, the transition function, and, trivially, the output function[5]: The state set Q' must be a subset of the state set Q of the fault-free component automaton M. The transition function may be arbitrary. The output function λ' is the restriction of λ to $Q' \times X$. Note that $\mathbf{0}$ is a delta fault type of M.

5. RAM Fault Types

We now define the notation used for RAMs. The state of cell i is denoted by q_i, $q_i \in \{0,1\}$, and the state of the entire n-cell RAM is $q = (q_1, q_2, \ldots, q_n)$. The operations on a cell are r^i (read cell i), w_0^i (write 0 to cell i), and w_1^i (write 1 to cell i). When a read operation is applied to cell i, the value q_i is obtained. For mathematical convenience we associate with each write operation the dummy symbol \$, representing no output. Thus the output alphabet of a RAM is $\{0, 1, \$\}$.

Formally, we have the following definition: A *fault-free* RAM type is a binary component automaton $M = (Q, X, Y, \delta, \lambda)$ where $Q = \{0,1\}^n$, $X = \bigcup_{i=1}^n X_i$ with $X_i = \{r^i, w_0^i, w_1^i\}$, $Y = \{0, 1, \$\}$, and δ and λ are defined by

$$
\begin{aligned}
\delta((q_1, \ldots, q_i, \ldots, q_n), r^i) &= (q_1, \ldots, q_i, \ldots, q_n), & \lambda((q_1, \ldots, q_i, \ldots, q_n), r^i) &= q_i, \\
\delta((q_1, \ldots, q_i, \ldots, q_n), w_0^i) &= (q_1, \ldots, 0, \ldots, q_n), & \lambda((q_1, \ldots, q_i, \ldots, q_n), w_0^i) &= \$, \\
\delta((q_1, \ldots, q_i, \ldots, q_n), w_1^i) &= (q_1, \ldots, 1, \ldots, q_n), & \lambda((q_1, \ldots, q_i, \ldots, q_n), w_1^i) &= \$,
\end{aligned}
$$

where $q = (q_1, q_2, \ldots, q_n)$ is the present state of the memory.

---

[4] The definition differs from that used in [3] and [4]. The condition used there was that Q *is equal to* $Q_1 \times \ldots \times Q_n$. That definition turned out to be too restrictive, in general, but it reappears in a natural way in Section 7.

[5] In [3] and [4] a more restrictive definition was used for delta faults.

As mentioned before, we associate a delta fault type $M' = (Q', X, Y, \delta', \lambda')$ with each defective n-bit RAM type. We study delta fault types satisfying the following conditions:

Condition 1 *A read operation in a faulty RAM type does not change the state of the RAM type, that is, for all $q \in Q'$, and $i \in \{1, \ldots, n\}$,*

$$\delta'(q, r^i) = q.$$

RAM delta fault types satisfying Condition 1 are called *write-activated.* An error can occur in such a RAM fault only if a write operation is executed.

Condition 2 *Writing a value to a cell in a faulty RAM type does not change the state of the RAM type if that value is already stored in that cell, that is, for all $q \in Q'$, $a \in \{0, 1\}$, and $i \in \{1, \ldots, n\}$,*

$$\text{if} \quad q_i = a, \quad \text{then} \quad \delta'(q, w_a^i) = q.$$

RAM delta fault types satisfying Conditions 1 and 2 are called *new-write-activated.* Note that an error can occur in a new-write-activated RAM fault type only if a new value is being written to some cell.

Condition 3 *If a new-write operation to a cell is unsuccessful, then the state of the RAM type does not change, that is, for all $q \in Q'$, $i \in \{1, \ldots, n\}$, if $q_i = \overline{a}$ and $\delta_i'(q, w_a^i) = \overline{a}$, then $\delta'(q, x) = q$.*

RAM delta fault types satisfying Condition 3 are said to be *successful-write-activated,* in the sense that a side effect of a new-write operation on a cell can occur only if the operation succeeds. Let $\mathcal{N}_n$ and $\mathcal{S}_n$ be the classes of new-write-activated and successful-write-activated delta fault types of n-bit RAMs, respectively. The following are examples of fault types in $\mathcal{N}_n \cap \mathcal{S}_n$.

A RAM type with a *stuck-at* fault in cell i is a delta fault type

$$M^{i=a} = (Q^{i=a}, X, Y, \delta^{i=a}, \lambda^{i=a}),$$

where $a \in \{0, 1\}$ and

$$Q^{i=a} = \{q \mid q = (q_1, \ldots, q_{i-1}, a, q_{i+1}, \ldots, q_n)\},$$

X and Y are as in the fault-free memory, $\delta^{i=a}$ is the restriction of δ to $Q^{i=a} \times X$, *except* for the transitions under input $w^i_{\overline{a}}$. This input fails to write $\overline{a}$ in cell i, that is,

$$\delta^{i=a}((q_1,\ldots,q_{i-1},a,q_{i+1},\ldots,q_n),w^i_{\overline{a}}) = (q_1,\ldots,q_{i-1},a,q_{i+1},\ldots,q_n).$$

Finally, $\lambda^{i=a}$ is the restriction of λ to $Q^{i=a} \times X$.

Two stuck-at faults are possible for each cell i, one for each value of a. An example of this fault type is given in Figure 5.1, where cell i is stuck-at-0.

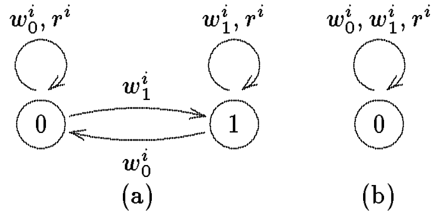

Figure 5.1. A memory cell i stuck-at-0: (a) the fault-free automaton; (b) the automaton $M^{i=0}$. Only the states of cell i are shown.

A RAM type with a *transition* fault in cell i is a delta fault type

$$M^{i\neg a} = (Q, X, Y, \delta^{i\neg a}, \lambda^{i\neg a}),$$

where $a \in \{0,1\}$ and $\delta^{i\neg a}$ behaves like δ *except* that, when the input w^i_a is applied to any state in which cell i is in state $\overline{a}$, the write operation is unsuccessful, that is,

$$\delta^{i\neg a}((q_1,\ldots,q_{i-1},\overline{a},q_{i+1},\ldots,q_n),w^i_a) = (q_1,\ldots,q_{i-1},\overline{a},q_{i+1},\ldots,q_n),$$

as in the stuck-at-$\overline{a}$ fault type. Two transition faults are possible for each cell i, one for each value of a. Figure 5.2 illustrates a transition fault type.

A *coupling* fault type involves two different memory cells, i and j. When the present states of cells i and j are $\overline{a}$ and $\overline{b}$, respectively, where $a, b \in \{0,1\}$ and a is written into cell i, then cell j, as well as cell i, changes state. All other transitions are correct. The fault type is denoted $M^{ia\Rightarrow jb}$. There are four coupling faults for each pair of cells, corresponding to the four possible binary values for a and b. Figure 5.3 illustrates the fault type $M^{i1\Rightarrow j1}$.

The single stuck-at, transition, and coupling fault types constitute the basic RAM fault model due to Thatte and Abraham [19]. Let $\mathcal{T}_n$ be the set of these fault types in n-bit RAMs.

A *stuck-equal* fault type involves a much stronger kind of coupling between two different memory cells, i and j. The states of the two cells are always equal. Whenever

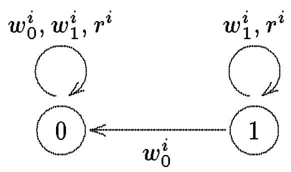

Figure 5.2. The transition fault type $M^{i\neg 1}$. Only the states of cell i are shown.

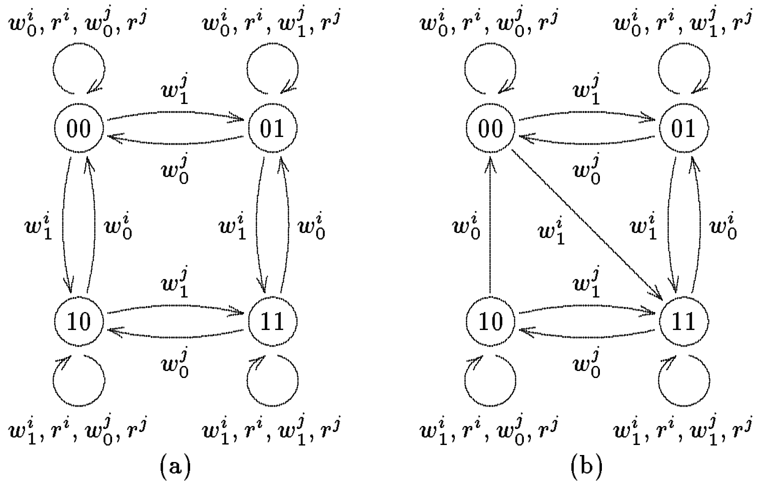

Figure 5.3. A coupling fault from cell i to cell j: (a) the fault-free automaton; (b) the automaton $M^{i1 \Rightarrow j1}$. Only the states of cells i and j are shown, the first and second components giving the states of cells i and j, respectively.

a value is written into one of the two cells, then writing succeeds and the same value is also written into the other cell. The fault type is denoted by $M^{i \Leftrightarrow j}$. Figure 5.4 illustrates this fault type [17].

All fault types listed so far are both new-write-activated and successful-write-activated, that is, in $\mathcal{N}_n \cap \mathcal{S}_n$. The fault type of Figure 5.5 is in $\mathcal{N}_n$, but not in $\mathcal{S}_n$. For two cells i and j containing the value 0, writing 1 into cell i fails, but changes cell j to 1 [3], [4]. Clearly, this fault violates Condition 3.

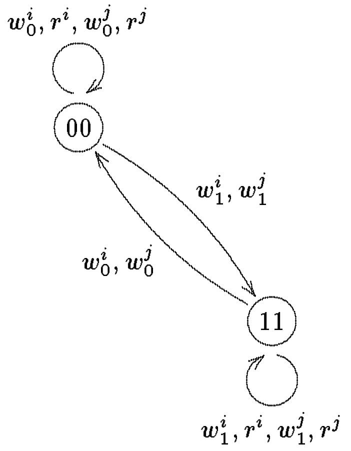

Figure 5.4. The stuck-equal fault type $M^{i \Leftrightarrow j}$. Only the states of cells i and j are shown.

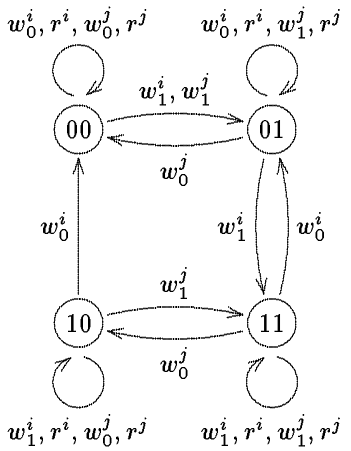

Figure 5.5. A coupling fault without successful writing. Only the states of cells i and j are shown.

6. Composition of Successful-Write-Activated RAM Fault Types

In practice more than one simple fault type may occur in a RAM. Thus it is important to study the behaviour of a memory when multiple fault types are present. In general, let $M^1 = (Q^1, X, Y, \delta^1, \lambda^1)$ and $M^2 = (Q^2, X, Y, \delta^2, \lambda^2)$ be two delta fault types of an n-bit RAM. We define the composition $M^1 \diamond M^2$ to be the component automaton

$$M^{1\diamond 2} = (Q^{1\diamond 2}, X, Y, \delta^{1\diamond 2}, \lambda^{1\diamond 2}),$$

where $Q^{1\diamond 2} = Q^1 \cap Q^2$, $\lambda^{1\diamond 2}$ is the restriction of λ to $Q^{1\diamond 2}$, and $\delta^{1\diamond 2}$ is defined as follows. Consider a state $q = (q_1, \ldots, q_n) \in Q^{1\diamond 2}$ and an input $x \in X$. Let

$$\begin{array}{ll} \delta(q,x) = (p_1, \ldots, p_n), & \delta^1(q,x) = (p_1^1, \ldots, p_n^1), \\ \delta^2(q,x) = (p_1^2, \ldots, p_n^2), & \delta^{1\diamond 2}(q,x) = (p_1^{1\diamond 2}, \ldots, p_n^{1\diamond 2}). \end{array}$$

To determine $\delta^{1\diamond 2}$ one applies the following rule.

Rule 1 *Suppose that $q_i = \overline{a} \in \{0,1\}$. The state of $M^{1\diamond 2}$ does not change unless x is a new-write operation, w_a^i, and that operation succeeds in both components. In that case, $p_i^{1\diamond 2} = a$ and, for all $j \neq i$,*

$$\textit{if } Q_j^{1\diamond 2} = \{b\} \textit{ then } p_j^{1\diamond 2} = b,$$

where $b \in \{0,1\}$; otherwise

$$p_j^{1\diamond 2} = p_j \oplus [(p_j \oplus p_j^1) + (p_j \oplus p_j^2)],$$

where $\oplus$ denotes the exclusive OR *operation.*

Note that Rule 1 preserves Conditions 1, 2, and 3. Thus, if M^1 and M^2 are in $\mathcal{N}_n$, $\mathcal{S}_n$, or $\mathcal{N}_n \cap \mathcal{S}_n$ then $M^1 \diamond M^2$ is again in $\mathcal{N}_n$, $\mathcal{S}_n$, or $\mathcal{N}_n \cap \mathcal{S}_n$, respectively.

The next result shows that the operation we have defined satisfies the generic conditions for the composition of successful-write-activated RAM fault types.

Theorem 6.1 *The algebras $(\mathcal{S}_n, \diamond)$ and $(\mathcal{N}_n \cap \mathcal{S}_n, \diamond)$ are semilattices with* **0** *as zero element.*

Proof: The proof is provided in the appendix. $\square$

Note that the algebra $(\mathcal{N}_n, \diamond)$ is not a semilatice because idempotence does not hold. For example, let M' be the automaton of Figure 5.5. One verifies that $M' \in \mathcal{N}_n$, but $M' \diamond M' \neq M'$. The laws of commutativity and associativity, however, do hold in $\mathcal{N}_n$ for $\diamond$.

7. Semilattice of Thatte-Abraham RAM Fault Types

We now investigate multiple fault types composed from single stuck-at, transition, and coupling fault types.

Proposition 7.1 *All RAM stuck-at, transition, and coupling fault types are in $\mathcal{N}_n \cap \mathcal{S}_n$.*

Proof: As has been pointed out in Section 5, all these fault types satisfy Conditions 1–3. $\square$

Let $\mathcal{T}_n^\diamond$ be the subsemilattice of $(\mathcal{N}_n \cap \mathcal{S}_n, \diamond)$ generated by $\mathcal{T}_n$, that is, let $\mathcal{T}_n^\diamond$ be the set of all multiple faults obtained by composing single faults from $\mathcal{T}_n$. We now show that the $\diamond$ operation satisfies the physical conditions for stuck-at, transition, and coupling fault types.

Theorem 7.1 *Hypotheses 4–8 hold for $\diamond$, that is, for all i, j with $1 \le i, j \le n$ and $i \neq j$ and for all $a, b, c \in \{0, 1\}$, we have*

(ϱ_1) $M^{i=a} \diamond M^{i=\overline{a}} = \mathbf{0}$,

(ϱ_2) $M^{i=a} \diamond M^{i\neg b} = M^{i=a}$,

(ϱ_3) $M^{i=a} \diamond M^{jb\Rightarrow ic} = M^{i=a}$,

(ϱ_4) $M^{i\neg a} \diamond M^{ia\Rightarrow jb} = M^{i\neg a}$,

(ϱ_5) $M^{i=a} \diamond M^{ib\Rightarrow jc} = M^{i=a}$.

One also verifies that, except for these cases, the composition of any two distinct fault types in $\mathcal{T}_n$ is not in $\mathcal{T}_n$.

Proof: To verify claims ϱ_1–ϱ_5 we need to apply Rule 1 to each composition above and verify that the result is the right-hand side. To verify the second claim we need to compute the composition of various pairs of fault types by using Rule 1 and verify that the result is not one of the fault types in $\mathcal{T}_n$. These verifications are straightforward. $\square$

Proposition 7.2 *Equation ϱ_5 is a consequence of Equations ϱ_1–ϱ_4.*

Proof: One obtains

$$M^{i=a} = M^{i=a} \diamond M^{i\neg b} = M^{i=a} \diamond (M^{i\neg b} \diamond M^{ib\Rightarrow jc}) = M^{i=a} \diamond M^{ib\Rightarrow jc}$$

using ϱ_2, ϱ_4, associativity, and again ϱ_2. $\square$

The equations in Theorem 7.1 permit us to simplify expressions for multiple faults. For example,

$$M^{i=0} \diamond M^{l=1} \diamond M^{k1\Rightarrow j1} \diamond M^{i\neg 0} \diamond M^{i1\Rightarrow j0} \diamond M^{k1\Rightarrow i0} = M^{i=0} \diamond M^{l=1} \diamond M^{k1\Rightarrow j1}.$$

Such simplifications should be valuable in the process of designing and verifying RAM tests. It can be shown [6] that all possible simplifications can be achieved using only ϱ_1–ϱ_4.

Next, we show that each fault type in semilattice $\mathcal{T}_n^\diamond$ satisfies a special condition on its state set. A set $Q' \subseteq \{0,1\}^n$ is called a *subproduct* if $Q' = Q'_1 \times \cdots \times Q'_n$ for some sets $Q'_1, \ldots, Q'_n \subseteq \{0,1\}$. A delta fault type is called a *subproduct* fault type if its state set is a subproduct. Note that $\mathbf{0}$ is a subproduct fault type, but the stuck-equal fault type is not. Thus, not all fault types in $\mathcal{S}_n$ are subproduct fault types.

Proposition 7.3 *Let M^1 and M^2 be subproduct fault types. Then $M^1 \diamond M^2$ is also a subproduct fault type. Thus, the set of subproduct fault types in $\mathcal{S}_n$ is a subsemilattice of $\mathcal{S}_n$ with $\mathbf{0}$ as zero element.*

Proof: The set of subproducts of $\{0,1\}^n$ is closed under intersection. □

Proposition 7.4 *Every $M' \in \mathcal{T}_n^\diamond$ is a subproduct fault type.*

Proof: The state set of every fault type in $\mathcal{T}_n$ is a subproduct. Thus, the claim follows by Proposition 7.3. □

Pattern-insensitivity is another property of the semilattice $\mathcal{T}_n^\diamond$. A new-write-activated RAM delta fault type $M' = (Q', X, Y, \delta', \lambda')$ has a *pattern-sensitive transition fault* in cell i if there exist $q, q' \in Q'$ such that $q'_i = q_i$ and $\delta'_i(q, w_a^i) \neq \delta'_i(q', w_a^i)$. In words, M' has a pattern-sensitive transition fault if there are states q and q', which agree in component i, and writing a into cell i is performed correctly when M' starts in q but incorrectly when it starts in q', or vice versa.

A new-write-activated RAM delta fault type $M' = (Q', X, Y, \delta', \lambda')$ has a *pattern-sensitive coupling fault* if there exist $q, q' \in Q'$ such that $q'_i = q_i$, $q'_j = q_j$, and writing a in cell i causes exactly one of q, q' to have an incorrect value in cell j, that is

$$\delta'_j(q, w_a^i) \neq \delta'_j(q', w_a^i).$$

A new-write-activated RAM fault type M' is *pattern-sensitive* if it has a pattern-sensitive transition or coupling fault. A new-write-activated RAM fault type is *pattern-insensitive* if it is not pattern-sensitive. Note that $\mathbf{0}$ is pattern-insensitive.

Proposition 7.5 *Let M^1 and M^2 be pattern-insensitive fault types in $\mathcal{N}_n \cap \mathcal{S}_n$. Then also $M^1 \diamond M^2$ is pattern-insensitive. Thus, the set of pattern-insensitive fault types in $\mathcal{N}_n \cap \mathcal{S}_n$ is a subsemilattice of $\mathcal{N}_n \cap \mathcal{S}_n$ with $\mathbf{0}$ as zero element.*

A proof of Proposition 7.5 is provided in the appendix. The proposition confirms the intuition that pattern-sensitivity cannot be introduced through the co-existence of fault types all of which are pattern-insensitive. On the other hand, the composition of a pattern-sensitive fault type with a pattern-insensitive fault type can be pattern-insensitive. For example, consider the pattern-sensitive fault type M' that has $n = 3$ and behaves like $M^{11 \Rightarrow 21}$ when the contents of cell 3 is 0 and like the fault-free RAM when it is 1. When also the pattern-insensitive fault type $M^{3=0}$ is present, then $M' \diamond M^{3=0}$ coincides with $M^{11 \Rightarrow 21} \diamond M^{3=0}$ which is pattern-insensitive.

The set of subproduct fault types and the set of pattern-insensitive fault types are not comparable, that is, neither is contained in the other. For example, the stuck-equal fault type is not a subproduct fault type but it is pattern-insensitive. On the other hand, the pattern-sensitive fault type M' above is a subproduct fault type.

Proposition 7.6 *Every* $M' \in \mathcal{T}_n^{\diamond}$ *is pattern-insensitive.*

Proof: Every fault type in $\mathcal{T}_n$ is pattern-insensitive. Thus, the claim follows by Proposition 7.5. $\square$

8. Composition of Fault Types in $\mathcal{N}_n$

Until now we have followed the commonly used assumptions that coupling faults do not occur in the presence of transition or stuck-at faults in the coupling cells [11],[13]. In [3] and [4] a different composition operation $\circ$ for RAM fault types was introduced. Coupling can also occur between bit lines and within address decoders. Thus coupling should still be expected, at least in some cases, even if the addressed cells have stuck-at or transition faults. In other words, coupling phenomena can be due to effects outside the immediate region of the affected cells. The composition operation for such a physical model satisfies the generic conditions, Hypotheses 1–3, but only some of the physical conditions for the Thatte-Abraham model, namely Hypotheses 4–6. The operation $\circ$ differs from $\diamond$ in the way it models the composition of coupling faults with transition or stuck-at faults. While $\diamond$ requires the write operation to be successful for the coupling to take place, the composition $\circ$ permits the effect of the coupling fault to take place in the coupled cell even if the write operation in the coupling cell is unsuccessful due to a transition or stuck-at fault.

In essence this means that we have a different type of coupling fault which, as a single fault, is functionally equivalent to the coupling fault considered before, but the physical characteristics of which make it behave differently as a component of a multiple fault.[6]

Here we provide only a very brief outline of the connection between $\circ$ and $\diamond$. More details are given in [6]. Let $M^1 = (Q^1, X, Y, \delta^1, \lambda^1)$ and $M^2 = (Q^2, X, Y, \delta^2, \lambda^2)$ be

[6] This problem was already recognized in [3] and [4], where different representations of these two kinds of coupling faults are proposed, using so-called two-step automata introduced there.

two delta fault types of a binary component automaton with n components. We define the composition $M^1 \circ M^2$ to be the component automaton

$$M^{1\circ 2} = (Q^{1\circ 2}, X, Y, \delta^{1\circ 2}, \lambda^{1\circ 2}),$$

where $Q^{1\circ 2} = Q^1 \cap Q^2$, $\lambda^{1\circ 2}$ is the restriction of λ to $Q^{1\circ 2}$, and $\delta^{1\circ 2}$ is defined as follows. Consider a state $q = (q_1, \ldots, q_n) \in Q^{1\circ 2}$ and an input $x \in X$. Using the same notation as in Section 6, to determine $\delta^{1\circ 2}$ one applies the following:

Rule 2 *If the next state of a cell is incorrect in either component fault type, then it is also incorrect in the composite fault type provided that state is in the state set. Formally,*

$$\textit{for all } j,\ p_j^{1\circ 2} = \begin{cases} a, & \textit{if } Q_j^{1\circ 2} = \{a\}, \\ p_j \oplus [(p_j \oplus p_j^1) + (p_j \oplus p_j^2)], & \textit{if } |Q_j^{1\circ 2}| = 2. \end{cases}$$

Note that $\circ$ is defined for any kind of delta fault type of a binary RAM. The definition in [3] and [4] is more restrictive as only subproduct fault types are considered there. This restriction, however, turns out to be unnecessary.

Theorem 8.1 ([3], [4]) *The family of delta fault types of a binary component automaton with $\circ$ as composition is a semilattice with* **0** *as zero element.*

Thus, $\circ$ satisfies the generic conditions, Hypotheses 1–3, for the composition of fault types.

Proposition 8.1 *The set $\mathcal{N}_n$ is closed under $\circ$, but the sets $\mathcal{S}_n$ and $\mathcal{N}_n \cap \mathcal{S}_n$ are not.*

Proof: One verifies that Rule 2 preserves Conditions 1 and 2. This shows that $\mathcal{N}_n$ is closed under $\circ$.

Now consider the two fault types $M^{i\neg 1}$ and $M^{i1\Rightarrow j1}$. The composite fault $M^{i\neg 1} \circ M^{i1\Rightarrow j1}$ shown in Figure 5.5 is not successful-write-activated. $\square$

Let $\mathcal{T}_n^\circ$ be the semilattice generated by $\mathcal{T}_n$, the Thatte-Abraham fault types, with $\circ$ as the composition operation. By Proposition 8.1, $\mathcal{T}_n^\circ$ is a subsemilattice of $(\mathcal{N}_n, \circ)$ which, by the example of Figure 5.5, is not contained in $\mathcal{S}_n$. The difference results from a different interpretation of the coupling fault types.

The following result is the analogue of Theorem 7.1 for the case of $\circ$ as composition operation.

Theorem 8.2 ([3], [4]) *Hypotheses 4–6 hold for $\circ$, that is, for all i, j with $1 \le i, j \le n$ and $i \ne j$ and for all $a, b, c \in \{0, 1\}$, we have*

(γ_1) $M^{i=a} \circ M^{i=\overline{a}} = \mathbf{0}$,
(γ_2) $M^{i=a} \circ M^{i\neg b} = M^{i=a}$,
(γ_3) $M^{i=a} \circ M^{jb\Rightarrow ic} = M^{i=a}$,

One also verifies that, except for these cases, the $\circ$-composition of any two distinct fault types in $\mathcal{T}_n$ is not in $\mathcal{T}_n$.

By Theorem 8.2, the operations $\circ$ and $\diamond$ coincide on the stuck-at, transition, and coupling fault types considered separately, and also on the stuck-at and transition fault types combined. Because of ϱ_4, they differ on the combination of transition and coupling fault types and also, as a consequence of Proposition 7.2, on the combination of stuck-at and coupling fault types. The relation among the families of fault types is illustrated in Figure 8.1. Note that $\mathcal{S}_n \cap \mathcal{T}_n^\circ = \mathcal{T}_n^\diamond$ by [6].

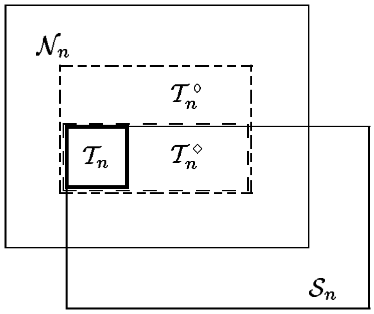

Figure 8.1. The relation among the families of new-write-activated and of successful-write-activated fault types.

9. Concluding Remarks

We have presented a unified framework for the study of multiple faults. In this framework we have investigated the special case of faults in RAMs. We have defined fault composition operations that satisfy both a set of generic conditions—which are independent of the nature of the faults—and a set of physical conditions—which are determined by the details of physical defects. In our model a number of previously unstated assumptions have been made explicit, and several issues about the interactions among RAM faults have been clarified. We have demonstrated that our basic framework remains applicable under different physical assumptions about defects. Our results lend themselves to automatic generation and verification of test sequences.

Acknowledgments: The authors wish to thank Bruce Cockburn for his assistance with the physical defects corresponding to coupling faults, Michael Gössel for his suggestion to separate the generic and physical properties of composition operations, and Kathleen Zhou for many useful discussions during the early stages of this work.

References

[1] J. A. Brzozowski, B. F. Cockburn: Detection of Coupling Faults in RAMs. *J. of Electronic Testing: Theory and Applications* **1** (1990), 151–162.

[2] J. A. Brzozowski, H. Jürgensen: A Model for Sequential Machine Testing and Diagnosis. *J. of Electronic Testing: Theory and Applications* **3** (1992), 219–234.

[3] J. A. Brzozowski, H. Jürgensen: Component Automata and RAM Faults. *Abstracts of the 2nd International Colloquium on Words, Languages, and Combinatorics*, Kyoto, Japan, 1992, 6–10.

[4] J. A. Brzozowski, H. Jürgensen: Component Automata and RAM Faults. *Proc. 2nd International Colloquium on Words, Languages, and Combinatorics, Kyoto, Japan, 1992*, edited by M. Ito and H. Jürgensen. World Scientific, Singapore, 1994, 49–67.

[5] J. A. Brzozowski, H. Jürgensen: Composition of Multiple Faults in RAMs. *Proceeedings, 1995 IEEE International Workshop on Memory Technology, Design and Testing, San Jose, CA, August 7–8, 1995*, IEEE Computer Society Press, Los Alamitos, CA.

[6] J. A. Brzozowski, H. Jürgensen: Semilattices of Multiple-Fault Automata. Technical report in preparation, 1995.

[7] A. H. Clifford, G. B. Preston: *The Algebraic Theory of Semigroups*, vol. 1. Amer. Math. Soc., Providence, 1961.

[8] B. F. Cockburn: *Fault Models and Tests for Coupling Faults in Random-Access Memories.* PhD Thesis, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1990; also Research Report CS-90-41, University of Waterloo.

[9] B. F. Cockburn: Deterministic Tests for Detecting Single V-Coupling Faults in RAMs. *J. of Electronic Testing: Theory and Applications* **5** (1994), 91–113.

[10] B. F. Cockburn, J. A. Brzozowski: Near-Optimal Tests for Classes of Write-Triggered Coupling Faults in RAMs. *J. of Electronic Testing: Theory and Applications* **3** (1992), 251–264.

[11] J. Cocking: RAM Test Patterns and Test Strategy. *Digest of 1975 Semiconductor Test Symposium*, Cherry Hill, NJ, IEEE Computer Society Press, Los Alamitos, CA, 1975, 1–8.

[12] R. David, J. A. Brzozowski, H. Jürgensen: Random Test Length for Bounded Faults in RAMs. *Proc. ETC'93*, IEEE Computer Society Press, Los Alamitos, CA, 1993, 149–158.

[13] A. J. van de Goor, *Testing Semiconductor Memories. Theory and Practice.* Wiley, Chichester, England, 1991.

[14] A. J. van de Goor, B. Smit: Automating the Verification of March Tests. *Records of the 1993 IEEE International Workshop on Memory Testing, San Jose, CA, August 9–10, 1993,* IEEE Computer Society Press, Los Alamitos, CA, 1993, 131–136.

[15] A. J. van de Goor, B. Smit: Automating the Verification of Memory Tests. *Proc. 12th VLSI Test Symposium, Cherry Hill, NJ, April 25–28, 1994,* IEEE Computer Society Press, Los Alamitos, CA, 1994, 312–318.

[16] H. Jürgensen, P. Wong: How to Prove Fault Coverage—or: Testing for the Most Difficult Fault. *Proc. 6th Workshop on New Directions for Testing,* Montréal, Québec, Canada, May 20–22, 1992, 285–294.

[17] L. Shen, B. F. Cockburn: An Optimal March Test for Locating Faults in DRAMs. *Records of the 1993 IEEE International Workshop on Memory Testing, San Jose, CA, August 9–10, 1993,* IEEE Computer Society Press, Los Alamitos, CA, 1993, 61–66.

[18] B. Smit: *Automatic Verification of March Tests.* Thesis Report 1-68340-28(1993)23, Department of Electrical Engineering, Delft University of Technology, Delft, The Netherlands, 1993.

[19] S. M. Thatte, J. A. Abraham: Testing of Semiconductor Random Access Memories. *Digest of Papers, 7th Int. Conf. on Fault-Tolerant Computing,* Los Angeles, June 28–30, 1977, 81–87.

[20] P. Wong: *A Language Theoretic Approach to Fault Coverage and Fault Testing.* MSc Thesis, Department of Computer Science, The University of Western Ontario, London, Ontario, Canada, 1992; also Research Report 314.

Appendix

Proof of Theorem 6.1: It is sufficient to prove the statement for $\mathcal{S}_n$. As $\mathcal{N}_n$ is closed under $\diamond$ and as $\mathbf{0} \in \mathcal{N}_n$ it follows that also $(\mathcal{N}_n \cap \mathcal{S}_n, \diamond)$ is a semilattice with $\mathbf{0}$ as zero element.

Consider $M^1, M^2, M^3 \in \mathcal{S}_n$. Rule 1 is symmetric in M^1 and M^2. Therefore, $\diamond$ is commutative. For the proof of idempotence, assume that $M^1 = M^2$. Then $Q^{1\diamond 2} = Q^1$ and $\lambda^{1\diamond 2} = \lambda^1$. If $M^1 = \mathbf{0}$ then also $M^{1\diamond 2} = \mathbf{0}$. Therefore, assume that $M^1 \neq \mathbf{0}$. Consider $q \in Q^{1\diamond 2}$, $i \in \{1, \ldots, n\}$ and $x \in X_i$. If x is not a new-write operation, or is an unsuccessful new-write operation, then $\delta^1(q, x) = q$ because

$M^1 \in \mathcal{S}_n$. By Rule 1, $p^{1\diamond 2} = q$ as well. If $x = w_a^i$ is a successful new-write operation, then $p_i^1 = a = p_i^2 = p_i$ and, by Rule 1, $p_i^{1\diamond 2} = a$. Now consider $j \neq i$. If $Q_j^{1\diamond 2} = \{b\}$ then $Q_j^1 = Q_j^2 = \{b\}$ and $p_j^1 = p_j^2 = p_j^{1\diamond 2} = b$. On the other hand, if $|Q_j^{1\diamond 2}| = 2$ then $|Q_j^1| = |Q_j^2| = 2$ and

$$p_j^{1\diamond 2} = p_j \oplus (p_j \oplus p_j^1) = p_j^1 = p_j^2.$$

This proves that $M^1 \diamond M^2 = M^1$.

We now turn to the proof of associativity. Let

$$M^{(1\diamond 2)\diamond 3} = (Q^{(1\diamond 2)\diamond 3}, X, Y, \delta^{(1\diamond 2)\diamond 3}, \lambda^{(1\diamond 2)\diamond 3}) = M^{1\diamond 2} \diamond M^3$$

and

$$M^{1\diamond(2\diamond 3)} = (Q^{1\diamond(2\diamond 3)}, X, Y, \delta^{1\diamond(2\diamond 3)}, \lambda^{1\diamond(2\diamond 3)}) = M^1 \diamond M^{2\diamond 3}.$$

The associativity of the intersection of sets implies $Q^{(1\diamond 2)\diamond 3} = Q^{1\diamond(2\diamond 3)}$. Let $Q^{1\diamond 2\diamond 3} = Q^{(1\diamond 2)\diamond 3}$. As $\lambda^{(1\diamond 2)\diamond 3}$ and $\lambda^{1\diamond(2\diamond 3)}$ are the restrictions of λ to $Q^{(1\diamond 2)\diamond 3}$ and $Q^{1\diamond(2\diamond 3)}$, respectively, one has $\lambda^{(1\diamond 2)\diamond 3} = \lambda^{1\diamond(2\diamond 3)}$. Now consider a state $q = (q_1, \ldots, q_n) \in Q^{1\diamond 2\diamond 3}$. Let

$$\begin{aligned}
\delta(q,x) &= (p_1, \ldots, p_n), & \delta^1(q,x) &= (p_1^1, \ldots, p_n^1),\\
\delta^2(q,x) &= (p_1^2, \ldots, p_n^2), & \delta^3(q,x) &= (p_1^3, \ldots, p_n^3),\\
\delta^{1\diamond 2}(q,x) &= (p_1^{1\diamond 2}, \ldots, p_n^{1\diamond 2}), & \delta^{2\diamond 3}(q,x) &= (p_1^{2\diamond 3}, \ldots, p_n^{2\diamond 3}),\\
\delta^{(1\diamond 2)\diamond 3}(q,x) &= (p_1^{(1\diamond 2)\diamond 3}, \ldots, p_n^{(1\diamond 2)\diamond 3}), & \delta^{1\diamond(2\diamond 3)}(q,x) &= (p_1^{1\diamond(2\diamond 3)}, \ldots, p_n^{1\diamond(2\diamond 3)}).
\end{aligned}$$

We distinguish two cases.

Case 1: Suppose that x is a read or old-write operation. Then $\delta(q,x) = q$. As the fault types are successful-write-activated one also has $\delta^1(q,x) = \delta^2(q,x) = \delta^3(q,x) = q$. By Rule 1, $\delta^{1\diamond 2}(q,x) = \delta^{2\diamond 3}(q,x) = q$; hence $\delta^{(1\diamond 2)\diamond 3}(q,x) = \delta^{1\diamond(2\diamond 3)}(q,x) = q$ as well.

Case 2: Suppose that $x = w_a^i$ is a new-write operation. Then $q_i = \overline{a}$. Suppose that $p_i^k = \overline{a}$ for some $k \in \{1,2,3\}$. If $k \leq 2$ then $\delta^{1\diamond 2}(q, w_a^i) = q$ by Rule 1, that is, $p_i^{1\diamond 2} = \overline{a}$. If $k \geq 2$ then $\delta^{2\diamond 3}(q, w_a^i) = q$, that is, $p_i^{2\diamond 3} = \overline{a}$. It follows that $\delta^{(1\diamond 2)\diamond 3}(q, w_a^i) = q = \delta^{1\diamond(2\diamond 3)}(q, w_a^i)$.

Therefore, assume that $p_i^k = a$ for all $k \in \{1,2,3\}$. Then also $p_i^{1\diamond 2} = a = p_i^{2\diamond 3}$ and $p_i^{(1\diamond 2)\diamond 3} = a = p_i^{1\diamond(2\diamond 3)}$. Consider j with $j \neq i$. If $Q_j^{1\diamond 2\diamond 3} = \{b\}$ then $p_j^{(1\diamond 2)\diamond 3} = p_j^{1\diamond(2\diamond 3)} = b$. On the other hand, if $Q_j^{1\diamond 2\diamond 3} = \{0,1\}$, then also $Q_j^{1\diamond 2} = Q_j^{2\diamond 3} = Q_j^1 =$

$Q_j^2 = Q_j^3 = \{0,1\}$ and

$$\begin{aligned} p_j^{1\diamond 2} &= p_j \oplus [(p_j \oplus p_j^1) + (p_j \oplus p_j^2)], & p_j^{2\diamond 3} &= p_j \oplus [(p_j \oplus p_j^2) + (p_j \oplus p_j^3)], \\ p_j^{(1\diamond 2)\diamond 3} &= p_j \oplus [(p_j \oplus p_j^{(1\diamond 2)}) + (p_j \oplus p_j^3)], & p_j^{1\diamond(2\diamond 3)} &= p_j \oplus [(p_j \oplus p_j^1) + (p_j \oplus p_j^{(2\diamond 3)})]. \end{aligned}$$

One verifies that $p_j^{(1\diamond 2)\diamond 3} = p_j^{1\diamond(2\diamond 3)}$ [4].

Finally, $M^1 \diamond \mathbf{0} = \mathbf{0}$ for any $M^1 \in \mathcal{S}_n$. This concludes the proof. $\square$

Proof of Proposition 7.5: Consider pattern-insensitive delta fault types $M^1 = (Q^1, X, Y, \delta^1, \lambda^1)$ and $M^2 = (Q^2, X, Y, \delta^2, \lambda^2)$ in $\mathcal{N}_n \cap \mathcal{S}_n$. If $M^{1\diamond 2} = \mathbf{0}$ then nothing needs to be proved. Hence, assume that $M^{1\diamond 2} \neq \mathbf{0}$ and suppose that $M^{1\diamond 2}$ is pattern-sensitive.

First, suppose that $M^{1\diamond 2}$ has a pattern-sensitive transition fault in cell i. Then there exist $q, q' \in Q^{1\diamond 2}$ with $q_i = q'_i$ and $\delta_i^{1\diamond 2}(q, w_a^i) \neq \delta_i^{1\diamond 2}(q', w_a^i)$ for some $a \in \{0,1\}$. By Theorem 6.1, $M^{1\diamond 2}$ satisfies Condition 2. Therefore, if $q_i = a = q'_i$ then $\delta_i^{1\diamond 2}(q, w_a^i) = \delta_i^{1\diamond 2}(q', w_a^i)$, a contradiction. Hence, $q_i = \overline{a} = q'_i$. Without loss of generality we assume that $\delta_i^{1\diamond 2}(q, w_a^i) = a$ and $\delta_i^{1\diamond 2}(q', w_a^i) = \overline{a}$.

By Rule 1 it follows that at least one of $\delta_i^1(q', w_a^i)$ and $\delta_i^2(q', w_a^i)$ is equal to $\overline{a}$. As M^1 and M^2 are pattern-insensitive, one has $\delta_i^1(q, w_a^i) = \delta_i^1(q', w_a^i)$ and $\delta_i^2(q, w_a^i) = \delta_i^2(q', w_a^i)$. Therefore, $\delta_i^{1\diamond 2}(q, w_a^i) = \overline{a}$ by Rule 1, a contradiction. Thus, $M^{1\diamond 2}$ does not contain a pattern-sensitive transition fault.

Hence, suppose that $M^{1\diamond 2}$ contains a pattern-sensitive coupling fault. Then there are distinct cells i and j and states $q, q' \in Q^{1\diamond 2}$ with $q_i = q'_i$ and $q_j = q'_j$ such that $\delta_j^{1\diamond 2}(q, w_a^i) \neq \delta_j^{1\diamond 2}(q', w_a^i)$. Again, if $q_i = a$ then, by Condition 2,

$$\delta_j^{1\diamond 2}(q, w_a^i) = q_j = q'_j = \delta_j^{1\diamond 2}(q', w_a^i).$$

Hence, assume that $q_i = \overline{a} = q'_i$. If $\delta_i^{1\diamond 2}(q, w_a^i) \neq \delta_i^{1\diamond 2}(q', w_a^i)$ then we also have a pattern-sensitive transition fault, and this case has already been excluded. Therefore, $\delta_i^{1\diamond 2}(q, w_a^i) = \delta_i^{1\diamond 2}(q', w_a^i)$.

If $\delta_i^{1\diamond 2}(q, w_a^i) = \overline{a} = \delta_i^{1\diamond 2}(q', w_a^i)$ then also $\delta_j^{1\diamond 2}(q, w_a^i) = \delta_j^{1\diamond 2}(q', w_a^i)$, a contradiction. Thus, $\delta_i^{1\diamond 2}(q, w_a^i) = a = \delta_i^{1\diamond 2}(q', w_a^i)$. As M^1 and M^2 are pattern-insensitive, one has

$$\delta_j^1(q, w_a^i) = \delta_j^1(q', w_a^i) \quad \text{and} \quad \delta_j^2(q, w_a^i) = \delta_j^2(q', w_a^i);$$

hence $\delta_j^{1\diamond 2}(q, w_a^i) = \delta_j^{1\diamond 2}(q', w_a^i)$. $\square$