6-1-1989

Detailed Modeling and Reliability Analysis of Fault-Tolerant Processor Arrays

Noe Lopez-Benitez
Purdue University

Jose A. B. Fortes
Purdue University

Detailed Modeling and Reliability Analysis of Fault-Tolerant Processor Arrays

Noe Lopez-Benitez
Jose A. B. Fortes

TR-EE 89-31
June, 1989

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

DETAILED MODELING AND RELIABILITY ANALYSIS OF FAULT-TOLERANT PROCESSOR ARRAYS

Noe Lopez-Benitez and Jose A. B. Fortes†

School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

ABSTRACT

Recent advances in VLSI/WSI technology have led to the design of processor arrays with a large number of processing elements confined in small areas. The use of redundancy to increase fault-tolerance has the effect of reducing the ratio of area dedicated to processing elements over the area occupied by other resources in the array. The assumption of fault-free hardware support (switches, buses, interconnection links, etc.,), leads at best to conservative reliability estimates. However, detailed modeling entails not only an explosive growth in the model state space but also a difficult model construction process. To address the latter problem, a systematic method to construct Markov models for the reliability evaluation of processor arrays is proposed. This method is based on the premise that the fault behavior of a processor array can be modeled by a Stochastic Petri Net (SPN). However, in order to obtain a more compact representation, a set of attributes is associated with each transition in the Petri net model. This representation is referred to as a Modified Stochastic Petri Net (MSPN) model. A MSPN allows the construction of the corresponding Markov model as the reachability graph is being generated. The Markov model generated can include the effect of failures of several different components of the array as well as the effect of a peculiar distribution of faults when the reconfiguration occurs. Specific reconfiguration schemes such as Successive Row Elimination (SRE), Alternate Row-Column Elimination (ARCE) and Direct Reconfiguration (DR), are analyzed

† This research was supported in part by the National Science Foundation under Grant DC1-8419745 and in part by the Innovative Science and Technology Office of the Strategic Defense Initiative Organization and was administered through the Office of Naval Research under contracts No. 00014-85-k-0588 and No. 00014-88-k-0723.

in detail. Randomization techniques are used to solve the inherently large models that can be generated via a MSPN representation. A model reduction technique based on the discrimination of states with low mean holding times is discussed. Finally, an analysis of hierarchical structures formed with variations of the schemes analyzed, is presented. The results reported in this work were obtained using MGRE (Model Generation and Reliability Evaluator) which is a software package designed to analyze fault-tolerant processor arrays for which a MSPN representation is given.

CHAPTER I INTRODUCTION

1.1. Objectives

Recent advances in VLSI/WSI technology have led to the design of processor arrays with a large number of processing elements (PE's) confined in small areas. At wafer level, the elimination of interchip connections and faster signal propagation due to shorter interconnections between PE's results in higher processing speed. However, large densities and hardware complexity increase the likelihood of faults during the fabrication process. In addition, a large number of active elements involved in a single computation cycle, increases the possibility of failures at any time during the operational life of the array. Increased likelihood of faults during the fabrication process results in low production yield whereas large number of active elements involved in the functional operation of the array, decreases reliability. Research efforts have been directed not only to yield improvement but also to increase fault-tolerance [KoB84,KoP87]. To increase both yield and reliability, several reconfiguration algorithms which use the available redundancy in the array have been proposed. In addition, fault detection and recovery upon the occurrence of faults is required during the fabrication process or the functional operation of the array. Because of their susceptibility to faults and the added hardware complexity on the overall array, components other than processing elements become very important in the analysis of fault-tolerant processor arrays. However, there is a tendency to limit the reliability analysis of processor arrays to failures of processing elements only [RaA84]. The assumption of fault-free support hardware in the form of switches, buses, interconnection links, etc., for the analysis of fault-tolerant processor arrays leads at best to conservative reliability estimations. The need to construct more general models is recognized in the mathematical framework derived by Koren et al [KoB84,KoP87,KoP86] to evaluate yield improvement and performance-related measures of different array architectures.

A number of analytic models and methods for the reliability analysis of fault tolerant computer systems currently exists [Tri85,GeT83,HiE83]. Combinatorial analysis and Markov models are the most common methods used to develop analytical models for the reliability analysis of fault tolerant systems. In the case of combinatorial methods, if the system can be divided into several modules, they are assumed independent (i.e. no faults or repair dependencies exist) and reliability estimations can be derived using fault trees and series-parallel structures for which definite mathematical tools exist [Tri82]. Markov models on the other hand offer the alternative of analyzing systems with interdependent components. A fault-tolerant processor array can be considered a system with interdependent components in the sense that the failure of one component may or may not affect other components of the array. The manner in which the components of the array are interdependent varies with the topology implemented and the reconfiguration algorithm that takes place when a faulty component is detected.

In this work the problem of evaluating the reliability of processor arrays using Markov models is addressed. However, a detailed modeling that takes into consideration failure interdependencies of several components, entails not only an explosive growth in the model state space but also a difficult model construction process. To address the latter problem, a systematic method to construct Markov models for the reliability evaluation of processor arrays is proposed. This method is based on the premise that the fault behavior of a processor array can be modeled by a Stochastic Petri Net (SPN). In order to achieve a complete mapping from places and transitions in an SPN-like representation of small size to states and transition rates in the Markov model, a modified version of SPN's is proposed. However a more compact representation is obtained by associating a set of attributes with each transition in the Petri net. This representation is referred to as a Modified Stochastic Petri Net (MSPN) model. The set of attributes includes a probability function such that the effect of faulty spares in the reconfiguration algorithm is captured each a time a configuration change occurs. This distribution includes the probabilities of survival given that a number of components required by the reconfiguration process is faulty. Depending on the type of component and the reconfiguration scheme, probabilities of survival can be determined using closed form expressions or via simulation. Reconfiguration schemes such as Successive Row Elimination (SRE) Alternate Row-Column Elimination (ARCE) [FoR85] and Direct Reconfiguration (DR) [SaS86a],[SaS86b] are analyzed in detail using MSPN's.

Once the Petri net model and the corresponding reachability graph have been obtained, all the information required to build the transition matrix of the corresponding Markov chain is available. At this stage, figures of merit such as reliability, performability, etc., can be readily addressed. Reliability evaluation tools such as ARIES [MaA82] and SHARPE [ShT86] can be used to evaluate the models developed here. However the use of these packages is limited to the evaluation of models with a relatively low number of states. To circumvent the problems involved in the numerical evaluation of large models, a software implementation based on the randomization procedure, has been used to generate the reliability results reported in this work. A software package (MGRE) has been developed [Lop89] to analyze fault-tolerant processor arrays based on the approach presented in this report. By the execution of appropriate commands, MGRE can generate Markov models, evaluate reliability and MTTF given the size of the processor array and a set of failure rates.

1.2. Overview

In the second chapter the main topic of this report is developed. A modified version of stochastic Petri nets is presented and its application is illustrated by the generation of a model corresponding to the SRE reconfiguration scheme. In the third chapter the ARCE and DR reconfiguration schemes are analyzed in detail. Also, expressions required to calculate probabilities of survival in the presence of faults, are derived. Results concerning the state space and reliability analysis of these schemes are also reported. The fourth chapter discusses software algorithms for the numerical evaluation of large Markov models. In chapter five, several hierarchical configurations are compared with respect to their MTTF and reliability performance.

CHAPTER II
MODIFIED STOCHASTIC PETRI NETS

2.1. Introduction

As is the case with many systems, Markov models can be used to evaluate the reliability of processor arrays. However, reliability estimations are mostly based only on the failures of processing elements [RaA84]. Components other than processing elements become very important in the analysis of fault-tolerant processor arrays because of their susceptibility to faults and the added hardware complexity of the overall array. This fact has played an important role in the derivation of a mathematical framework developed by Koren et al [KoB84,KoP86,KoP87] to evaluate yield improvement and performance-related measures of different array architectures. However, a detailed modeling of fault-tolerant processor arrays, which explicitly takes into consideration the failure statistics of each component as well as their possible interdependencies, entails not only an explosive growth in the model state space but also a difficult model construction process. Therefore, in this chapter the latter problem is emphasized and a systematic method to construct Markov models to evaluate the reliability of processor arrays is discussed. This method is based on the premise that the fault behavior of a processor array can be modeled by a Stochastic Petri Net (SPN) However, in order to obtain a more compact representation, a set of attributes is associated with each transition in the Petri net model. The resulting model is referred to as a Modified Stochastic Petri Net (MSPN) representation. An MSPN representation allows the construction of the corresponding Markov model as the generation of the reachability graph takes place. Included in the set of attributes associated with each transition in a MSPN, is a discrete probability distribution such that the effect of faulty spares in the reconfiguration algorithm is captured each time a configuration change occurs. This distribution includes the probabilities of survival given that a number of components required by the reconfiguration process are faulty. Depending on the type of component and the reconfiguration scheme, probabilities of survival are determined using simulation or closed form expressions.

Once the Petri net model and the corresponding reachability graph have been obtained, all the information required to build the transition matrix of the corresponding Markov chain is available. At this stage, figures of merit such as reliability, performability, etc., can be readily addressed. Reliability evaluation tools such as ARIES [MaA82] and SHARPE [ShT86] can be used to evaluate the models developed here.

The second section of this chapter discusses some basic notation and concepts related to array configurations. An SPN representation is derived using as an example the SRE reconfiguration scheme. The third section of this chapter discusses Modified Stochastic Petri Nets (MSPN's) and an MSPN representation for the SRE scheme is derived; also a procedure to construct MSPN's is outlined. In the fourth section, the correspondence between markings in an MSPN representation and the states in a Markov chain is described; a procedure to construct the reachability graph of a given MSPN is outlined.

2.2. Concepts and Notation

In this section a representation of array configurations using a SPN is discussed. For illustration, examples using the SRE reconfiguration scheme are presented.

2.2.1. Array Configurations

To analyze a fault-tolerant array architecture with k types of components, the configuration of an array is represented as a k-tuple:

$$C_i = (\eta_{1i},\ \eta_{2i},\ \cdots\ ,\eta_{ki}) \qquad i = 0,1,\ \cdots\ ,|C|$$

where η_{li} denotes the number of elements of component type l and C is the set of all possible configurations of the array. Examples of component types include processing elements, links, switches, spare links and spare processing elements. The occurrence of faults and the application of the reconfiguration algorithm define a sequence of configurations that begins with C_0 as the *initial configuration*; any other configuration can correspond to the failure state or an operational state of the array. The latter will be referred to as an *operational configuration.*

Upon detection of a faulty component, the reconfiguration algorithm may not send the array to an operational configuration if any of the following happens:

1) The reconfiguration circuitry failed. This possibility can be considered through a *coverage factor* (denoted by c) defined as the probability of successful reconfiguration given that a fault has occurred [BoC69]. This is a measure of the probability of successful operation of all circuitry related to fault detection, isolation and reconfiguration. The coverage factor is assumed constant and it will be associated with failures of active components only.

2) Redundancy is exhausted. This information can be inferred from C_i.

3) The presence of faults in non-active components (redundancy) hinders a successful reconfiguration. Redundant components are present in C_i as spare processing elements, spare switches, spare links, spare buses, etc. Some of these components become active in the new configuration.

In a given configuration with a number of faulty components, successful reconfiguration will depend not only on the type of faults but also on their distribution in the array. Thus, the probability of correct reconfiguration in the presence of faults is referred to as the *probability of survival* [SaS86b]. Because the reconfiguration algorithm may choose one of several new configurations (including a non-operational one), a probability is assigned to each possible new configuration. The probability of survival corresponds to the sum of probabilities assigned to new operational configurations.

2.2.2. SPN Representation

While Markov models provide an analytical basis to derive reliability estimations of complex systems, their inherently large state space is difficult to construct and cannot be directly described in a convenient manner. On the other hand, SPN's provide a succinct representation of the system and support a mechanized construction of the Markov model, because markings in an SPN correspond bijectively to states in the Markov chain [Mol82]. In the case of fault-tolerant processor arrays, an operational configuration corresponds to an operational state in the Markov chain; thus, to derive all possible operational configurations of the array, a marking in the SPN must correspond to an operational configuration of the array. Each place p_i identifies components of type i and, at a given marking M_q, the number of tokens m_{iq} corresponds to η_{iq} which is the number of components of type i. Two or more distinct component types may identify the same physical component; for example, a physical spare is a component of the type "active spare" when it is used to replace a faulty part and it is a component of the type "non-active spare" otherwise.

Consider for example an $n \times n$ array that supports the Successive-Row-Elimination (SRE) reconfiguration scheme with a layout as in Figure 2.1. The SRE scheme as presented in [FoR85], is based on the successive elimination of rows. Row elimination is done by using switches (S's) and redundant links to bypass all processing elements (PE's) of any row containing at least one faulty PE or at least one faulty horizontal link (HL) or at least one faulty input/output link (I0L); spare bypass links (SBL's) become active bypass links (BL's) which are used to bypass faulty rows; the array fails when rows are exhausted or if either one active bypass link or a switch fails. A marking q is described as:

$M_q = (\#PE, \#IOL, \#S, \#BL, \#HL, \#SBL) = (m_{1q}, m_{2q}, m_{3q}, \ldots, m_{6q})$

where the symbol "#" is used to denote "number of".

A possible SPN representation is given in Figure 2.2. The firing of $\hat{t}_1$, represents the occurrence of a fault in a PE, a fault in an IOL is represented by the firing of $\hat{t}_2$ and so on. In general, the firing of $\hat{t}_i$ represents a fault occurrence in a component of type i where $1 \leq i \leq k$ and k is the number of places and transitions. In SRE, component types 1 through 6 correspond to PE, IOL, S, BL, HL, SBL and $k=6$, respectively.

Input and output places with respect to $\hat{t}_i$, represent the effect of a fault on the corresponding components; i.e., each transition $\hat{t}_i$, involves a set of I/0 functions $A_i = \{I(\hat{t}_i),\ O(\hat{t}_i)\}$ and a set V_i of multiplicities μ such that when $\hat{t}_i$ fires in M_q, the number of tokens m_{lq} in each place $p_l \epsilon A_i$ is modified as follows:

$$m_{lj} = m_{lq} - \mu_l^I + \mu_l^O$$

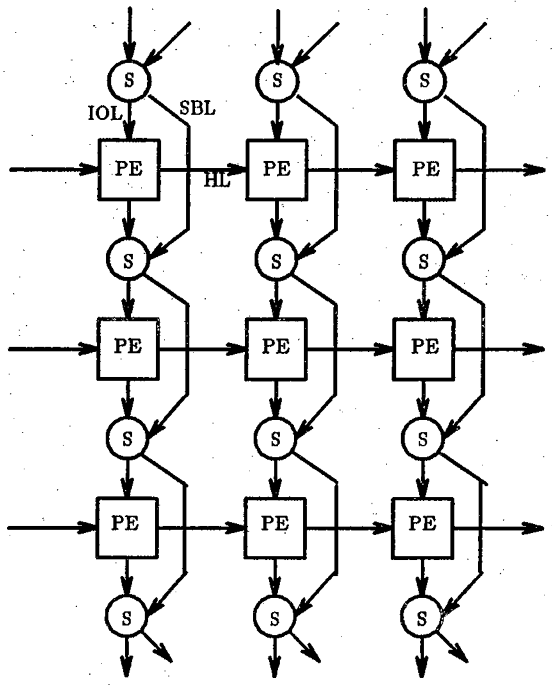


Figure 2.1. Schematic Layout of the SRE Structure

where m_{lj} is the modified number of tokens in p_l in a new marking M_j. If $\mu_l^O = 0$, p_l is only an input place and if $\mu_l^I = 0$, p_l is only an output place. For example in Fig. 2.2, the failure of a PE $(\hat{t}_1)$, affects the following components: PE's, IOL's, HL's, SBL's and BL's; therefore, a set of input and output places with respect to t_1 is given as:

$$A_1 = \{\{p_1,p_2,p_5,p_6\},\{p_4\}\}$$

Assuming an $n{\times}n$ processor array, the set of multiplicities associated with A_1 is given as:

$$V_1 = \{\mu_1^I = n;\ \mu_2^I = 2n;\ \mu_5^I = n-1;\ \mu_6^I = n;\ \mu_4^O = n\}.$$

If less than the number of spares required for reconfiguration are available, then the array fails to reconfigure and the resulting marking is characterized by the fact that $m_6 < 0$. The negative value of m_6 can be used to identify a failure marking reached due to exhaustion of spares. Some schemes require that specific transitions be enabled or disabled depending on the current configuration and the reconfiguration algorithm. Assume for example, that not only rows but also columns are eliminated during the reconfiguration process. In these cases, each μ_l becomes a function of the current marking; e.g., μ_1^I equals the number of columns in the current configuration. Hence, in order to include all possible cases, μ_l must be regarded as a *variable multiplicity* (this concept is an extension of the usual notion of "multiplicities" in Petri Net theory).

Although the SPN in Fig. 2.2, might provide the number of operational configurations required, it fails to consider the cases when enough spares are available but reconfiguration cannot take place (due for example to a peculiar distribution of faults). As a consequence, this approach might provide overly optimistic reliability estimates. Conceivably, a different SPN model can be used to accurately represent the dependency of successful reconfiguration on fault distributions. However, such an SPN would itself consist of a very large number of places and transitions which increases with the size of the array. One of the intents in this chapter is to provide an extension of the SPN concept so that dependence on fault-distribution can be accounted for in a model with a complexity comparable to that of Figure 2.2 regardless of the size of the array and reconfiguration scheme used.

2.3. MSPN Representation

The fact that several types of faults affect the array in the same manner, suggests the possibility of a more compact SPN-like representation, which is referred to as Modified SPN (MSPN). A MSPN takes into account different fault

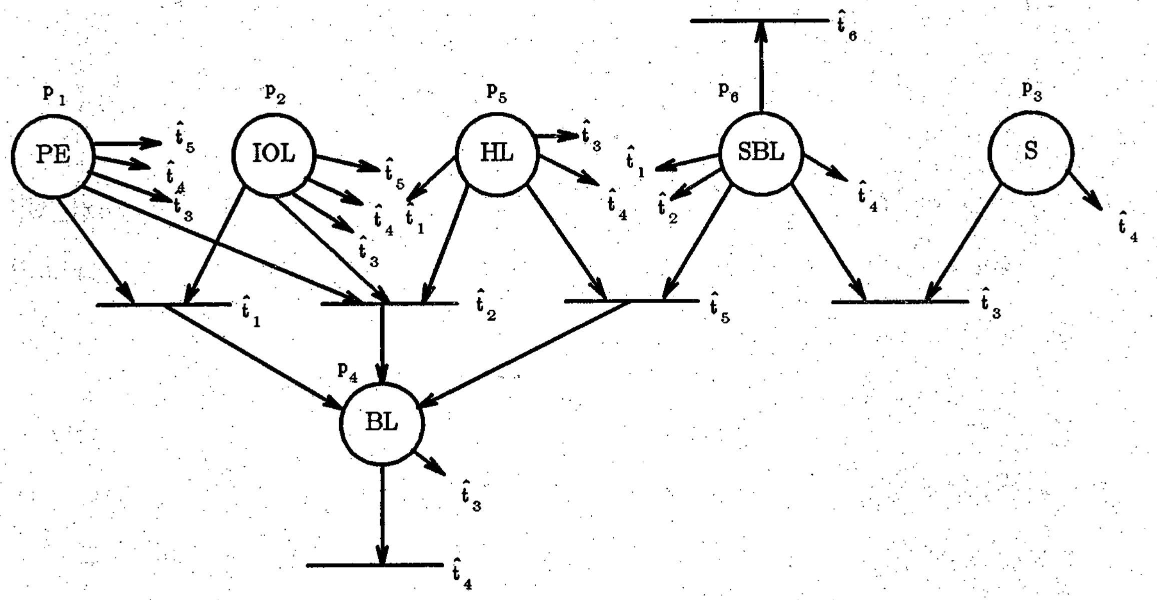


Figure 2.2. SPN of the SRE Scheme

distributions by associating with each transition t_i, a set of attributes described as follows:

$$t_i\text{: } <P(x \mid M_q, t_i), S_i(M_q), B_i, c_i>$$

where:

$P(x \mid M_q, t_i)$ defines a discrete probability function where x represents a random marking M_j in a set R directly reachable from a particular marking M_q; the notation Pr^i_{qj} is used to denote the *probability of reconfiguration* $P(x = M_j \mid M_q, t_i)$, i.e., the probability that the net is in marking M_j after t_i fires when the net is in marking M_q. The advantage of assigning a probability function to each transition t_i, is the ability to determine correct transition rates between states in the corresponding Markov model as the reachability graph is being generated. Thus, when t_i fires, one value of the probability function is assigned to the marking generated, another value is assigned to another marking that can be generated by the immediate firing of a possibly different transition and so on. This sequence of firings is dictated by the current marking and the reconfiguration scheme modeled. Notice that the markings generated could have been generated previously and the probabilities of reconfiguration are only used to modify the transition rates;

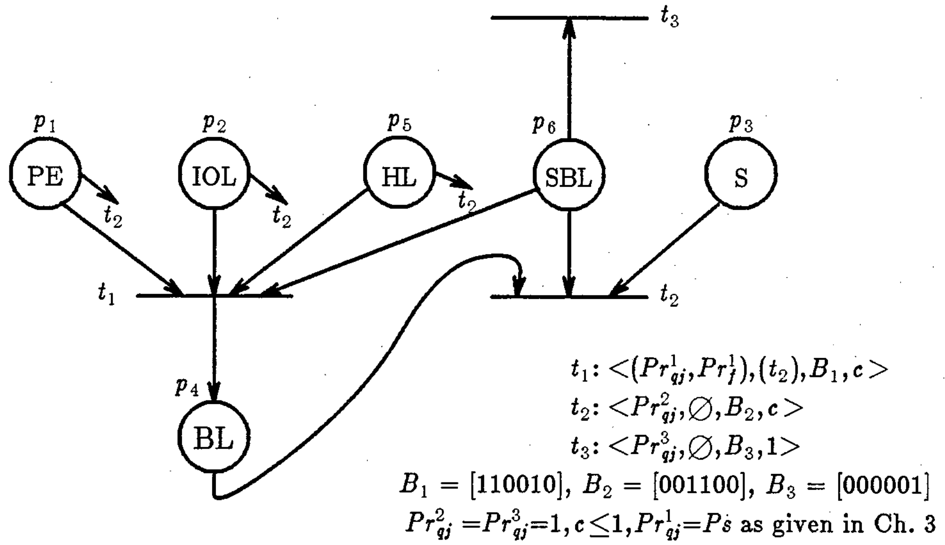
$S_i(M_q)$ is a sequence of transitions that will fire immediately after t_i fires. If no immediate firing is required then $S_i(M_q)$ is a null sequence. Depending on the reconfiguration scheme, S_i can be unique for all markings or is determined in terms of M_q;

B_i is a *Binary Transition Vector* with k elements b_l such that, $b_l = 1$ if the failure of the *lth* component triggers the transition t_i and $b_l = 0$ otherwise. This vector is used to identify those components that can trigger the corresponding transition t_i; it facilitates the merging of non-distinct markings, the derivation of probability transition vectors (defined in section 2.4) and the derivation of flags (i.e., failure rate conditions) that signal a possible non-occurrence of a transition;

The term c_i is a coverage factor associated with t_i, such that if t_i is triggered by the failure of a spare (inactive) component then $c_i = 1$; thus, the possibility of array failure at the time that one of these components fails is non-existent; if t_i is triggered by the failure of an active component then c_i corresponds to the probability of detection given that a fault occurs (i.e., $c_i \leq 1$).

To illustrate the set of transition attributes just discussed, consider the SRE scheme in Fig. 2.1 and the SPN of Fig. 2.2. Since $\hat{t}_1$, $\hat{t}_2$ and $\hat{t}_5$, have the same effect on the array, a single transition t_1 is defined in a MSPN with a vector $B_1 = [1\ 1\ 0\ 0\ 1\ 0]$ to indicate that either the failure of a PE, IOL or a HL, can cause t_1 to fire. Likewise, $\hat{t}_3$ and $\hat{t}_4$ become t_2 with a vector $B_2 = [0\ 0\ 1\ 1\ 0\ 0]$. The firing of t_2 represents the failure of a BL or a S either of which is fatal. Transition $\hat{t}_6$ becomes t_3 and represents the failure of a SBL with a vector $B_3 = [0\ 0\ 0\ 0\ 0\ 1]$.

Consider now the case when a given operational configuration contains faulty SBL's. The probability that none of them lies in the row that is eliminated when t_1 fires, corresponds to the probability of reconfiguration Pr^1_{qj}. If at least one SBL is faulty in the row deleted, the array fails to reconfigure with a probability $Pr^1_{qf} = 1 - Pr^1_{qj}$ (the failure marking is denoted as M_f). In general, probabilities of reconfiguration are complicated functions of the characteristics of each reconfiguration scheme (e.g., replacement rules, hardware requirements, dependencies on fault distributions, etc.) and the size and shape of the array. They must be derived for each scheme and in some cases extensive simulation is required due to the complexity of the combinatorial analysis involved. Examples where suitable expressions can be derived include SRE, ARCE and DR. Since different markings may correspond to different number of SBL's, Pr^1_{qj} is a function of the current marking (i.e., M_q). Let Ps denote the probability of survival, then for SRE, $Ps = Pr^1_{qj}$ and can be estimated via equation (3.8) or (3.9) which are derived in chapter 3 in terms of the number N of faulty SBL's, present in the current marking M_q. Some transitions in a MSPN will exhibit a dual nature: they can be fired exponentially or immediatedly; once a transition fires exponentially, a sequence of immediate firings (specified by S_i) of possibly different transitions may follow. Therefore, in SRE, when t_1 fires exponentially, t_2 will fire immediately to generate a failure marking and $S_1 = (t_2)$. This follows from the fact that when t_1 fires, the required SBL's become BL's and if one of them is faulty t_2 fires immediately. When t_2 or t_3 fire exponentially no immediate sequence is required and S_2 and S_3 are null sequences. It will be seen that in some applications such as ARCE, the sequence of immediate firings is not unique as it is selected depending on the current marking.



$$t_1: <(Pr^1_{qj}, Pr^1_f), (t_2), B_1, c>$$
$$t_2: <Pr^2_{qj}, \emptyset, B_2, c>$$
$$t_3: <Pr^3_{qj}, \emptyset, B_3, 1>$$
$$B_1 = [110010],\ B_2 = [001100],\ B_3 = [000001]$$
$$Pr^2_{qj} = Pr^3_{qj} = 1, c \leq 1, Pr^1_{qj} = P\dot{s} \text{ as given in Ch. 3}$$

Figure 2.3. MSPN of the SRE scheme

Considering the coverage factors, the sets of attributes associated with the transitions in the MSPN of the SRE scheme, are summarized in Figure 2.3.

2.3.1. I/O Places and Variable Multiplicities

Associated with each t_i in the MSPN there is a set of I/O places $A_i^* = \{I(t_i),\ O(t_i)\}$ and a set of multiplicities:

$$V_i^* = \{\mu_l^I, \mu_k^O \mid p_l \epsilon \{I(t_i)\}, p_k \epsilon \{O(t_i)\}\}$$

Two transitions in the SPN, can have the same set of I/O places and different set of multiplicities; i.e. two distinct faults may affect the same components in different ways. In order for distinct transitions in the SPN to be merged in a single transition in the MSPN, they must have the same set of I/O places and the same set of multiplicities. Because a coverage factor is associated with each transition in the MSPN, components whose failures fire a given transition must all correspond to either active or non-active components only (i.e., components of both types can not be present).

Example 2.1: Consider the SPN in Fig. 2.2. The failure of a PE ($\hat{t}_1$), the failure of a IOL ($\hat{t}_2$) or the failure of an HL ($\hat{t}_5$) affect the same components of the array and the set of multiplicities for each set of arcs is the same; therefore transitions $\hat{t}_1$, $\hat{t}_2$ and $\hat{t}_5$ can be merged into a single transition t_1 such that

$A_1^* = \{ I(t_1),\ O(t_1)\} = A_1 = A_2 = A_5 = \{\{p_1,p_2,p_5,p_6\},\ \{p_4\}\}$

and

$V_1^* = V_1 = V_2 = V_5 = \{\mu_1^I = n;\ \mu_2^I = 2n;\ \mu_5^I = n - 1;\ \mu_6^I = n;\ \mu_4^O = n\}.$

Likewise, $A_3 = A_4$; i.e., the failure of either a S or a BL ($\hat{t}_3$ or $\hat{t}_4$) causes the array to fail affecting all components in the same way; i.e., transitions $\hat{t}_3$ and $\hat{t}_4$ can be merged into a single transition t_2 such that

$A_2^* = \{I(t_2),\ O(t_2)\} = \{\{p_1,p_2,p_3,p_4,p_5,p_6\},\ \varnothing\}$

and a set of multiplicities

$V_2^* = V_3 = V_4 = \{\mu_1^I = \#PE_q;\ \mu_2^I = \#IOl_q;\ \mu_3^I = \#S_q;\ \mu_4^I = \#BL_q;\ \mu_5^I = \#HL_q;\ \mu_6^I = \#SBL_q\};$

Finally, we have that the failure of a SBL ($\hat{t}_6$) affects only SBL's; i.e.,

$A_3^* = \{I(t_3),\ O(t_3)\} = A_6 = \{\{p_6\},\ \{0\}\}$ with $V_3^* = V_6 = \{\mu_6 = 1\}$

where t_3 in the MSPN replaces $\hat{t}_6$. □.

In summary, an MSPN model is an extension of a SPN with the purpose of representing fault-tolerant processor arrays and can be defined as follows:

$$MSPN = (P,\ T,\ A,\ M,\ Pr,\ Sq,\ B,\ Cv)$$

where Pr is a set of probability functions $P(x \mid M_q, t_i)$ associated to transitions $t_i \epsilon T$; Sq is a set of sequences S_i of of transitions that fire immediately after an exponential firing of $t_i \epsilon T$; B is the set of binary transition vectors B_i defined for each $t_i \epsilon T$; Cv is the set of coverage constants c_i associated with each $t_i \epsilon T$.

2.3.2. MSPN Construction

Given a processor array and a reconfiguration scheme, an informal procedure to construct the corresponding MSPN is as follows:

Procedure MSPN

Inputs:

Array size;

Initial configuration $C_0 = (\eta_1\ \eta_2\ \cdots\ \eta_k)$;

Set of rules that determine the actions of the reconfiguration algorithm in response to faults; i.e., a rule r_{il} specifies the type and number of components added or subtracted if a component type i fails in an operational configuration C_l;

Coverage factors for the failures of both active and non-active components;

Tables of probabilities of reconfiguration for each non-active component type. (For each different non-active component type, a probability of successful reconfiguration is conditioned by the number of faults of each type; the probability of configuration is then the product of these conditional probabilities).

Output: MSPN representation: (P, T, A, M, Pr, Sq, B, Cv)

```
Begin
    Step 1:
    Assign a place $p_i$ and a transition $\hat{t}_i$ to component type $i$.
    Step 2:
    for each rule $r_{il}$ do
```

```
        Group the component types affected by the failure of component type $i$
        into a set $A_i$ as the set of I/O places with respect to transition $\hat{t}_i$.
        Determine the multiplicity of each $p \epsilon A_i$ and let $V_i$ be the set of
        multiplicities associated with $A_i$.
    **end** for
    *Step 3*:
    Group those transition $\hat{t}_i$ that: a) are fired by components which are all of
    type active or all of type non-active b) have the same set of input and output
    places c) have the same set of multiplicities and d) have same coverage
    factor.
    **For** each group i **do**
        Define transition $t_i$, sets $A_i^*$ and $V_i^*$.
        Specify $S_i(M_q)$.
        Form the binary transition vector $B_i$ such that a component $b_l$ of $B_i$ is
        one if the failure of a component type l can fire $t_i$ and zero otherwise.
    **end** for
**end** procedure
```

As pointed out previously a discrete probability distribution is associated with each transition such that a number of faulty spare components present in the current configuration is taken into account when the firing occurs. These probabilities are estimated using simulation or by closed form expressions.

2.4. Reachability Graph

In this section the generation of the reachability graph is discussed. A mapping from the transitions in the MSPN to the transitions between states in the corresponding Markov model is established. An implementation procedure to generate the reachability graph is also described.

2.4.1. Probability Transition Vectors

For each marking $M_{qj} \epsilon R$ generated when t_i fires (as stated previously R is the set of markings that can be directly generated from the same marking M_q), B_i and the distribution function $P(x \mid M_q, t_i)$ defined previously can be used to generate vectors of the form:

$$Pr^i_{qj} B_i = [pr_1 \cdots pr_k]$$

where $pr_l = Pr^i_{qj}$ if $b_l = 1$ and $pr_l = 0$ if $b_l = 0$. These vectors are referred to as the *Probability Transition Vectors (PTV's)*. The use of PTV's is illustrated in Fig. 2.4; assuming j=1,2,..., $|R|$ identifies the markings that can be directly generated from the same marking M_q, then $\sum_{j=1}^{|R|} Pr^i_{qj} = 1$. In the event that two or more non-distinct new markings are generated from the same marking, a merging to a single new marking is carried out by a vector addition of the corresponding probability transition vectors.

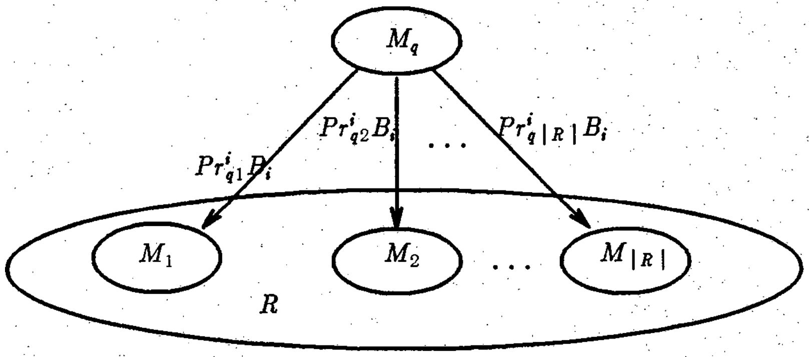


Figure 2.4.- Marking Generation with Pr^i_{qj}

Example 2.2: Analyzing a particular marking (in a 4×4 array with a MSPN as in Figure 2.3) say M_{18} = [12 24 20 4 9 11], then if t_1 fires (i.e. either a PE, a

IOL or a HL failed) the marking $M_{30} = [8\ 16\ 20\ 8\ 6\ 7]$ results with $Pr^1_{18,30} = 0.667$. Thus, a PTV is given by $Pr^1_{18,30}B_1 = [0.667\ 0.667\ 0\ 0\ 0.667\ 0]$. However the array may fail with $Pr^1_{30,f} = 0.333$ due to the existence of one faulty SBL (#PE's - #SBL's = 12 - 11 = 1). Thus, a PTV is given by $Pr^1_{18,f}B_1 = [0.333\ 0.333\ 0\ 0\ 0.333\ 0]$

When t_2 fires (i.e a S or a BL failed), the array fails with probability $Pr^2_{18,f} = 1$, the corresponding PTV is given by: $Pr^2_{18,f}B_2 = [0\ 0\ 1\ 1\ 0\ 0]$. Therefore, the overall PTV associated with the transition to the failure marking when t_1 or t_2 fires is obtained as follows: $Pr^1_{18,f}B_1 + Pr^2_{18,f}B_2 = [0.333\ 0.333\ 1\ 1\ 0.333\ 0]$. The probability of survival is obtained by applying equation (3.9) derived in chapter 3. $\square$.

2.4.2. Failure Rate Condition

Any transition in the MSPN can fire on every possible marking that represents an operational configuration. However, if all components whose failures fire a particular transition never fail, that transition will never fire. In the process of generating the reachability graph, a flag s_i signals this condition for each transition t_i. Let $\underline{\alpha}$ be a vector with components α_l where each α_l is the failure rate of the component type l and $\hat{\alpha}_i = B_i\underline{\alpha}^t$. The flag s_i is set to one if $\hat{\alpha}_i > 0$ and zero otherwise.

2.4.3. Derivation of Transition Rates

Let $\{X(t),\ t\geq 0\}$ denote a continuous time homogeneous Markov process on a finite state space $S = \{1,2,\ \cdots\ ,s\}$ and denote the state probability vector at time t by $P(t) = [P_1(t),\ P_2(t),\ \cdots\ ,\ P_s(t)]$, where $P_i(t) = P\{X(t){=}i\}$, $i\epsilon S$. The process is characterized by the following set of Chapman-Kolmogorov differential equations:

$$\frac{dP_j(t)}{dt} = \sum_i a_{ij} P_i(t) - a_{jj} P_j(t)$$

where a_{ij} are the transition rates from previous states i and $a_{jj} = \sum_k a_{jk}$, where a_{jk} are transitions rates from state j to states k. The solution for the set of differential equations is given in matrix form as follows:

$$P(t) = P(0)e^{At}$$

where A is the transition rate matrix with elements a_{ij} and $P(0) = (1,0, \cdots ,0)^t$ is the initial vector state. Assuming there is only one absorbing state (i.e., failure state) and indexing transient (operational) states by $1,...,s-1$, then the reliability of the system is given by: $R(t) = \sum_{i=1}^{s} P_i(t)$

The transition rates a_{ij} can be expressed in terms of the attributes associated with the transitions in the MSPN, i.e., $a_{qx} = f(P(x \mid M_q, t_i), B_i, c_i)$.

Denote by α a diagonal transition rate matrix with elements $\alpha_{ii} = \alpha_i$ and $\alpha_{ij} = 0$ for $i \neq j$ (α_i is the failure rate of the ith component). Let $\hat{M}_q = M_q \alpha$ where $M_q = [m_{1q}, m_{2q}, \cdots m_{kq}]$ denotes as before, a particular marking q, then

$$a_{ql} = [\sum_i c_i Pr_{ql}^i b_j;\ b_j \epsilon B_i, j=1,...,k]\hat{M}_q^T \qquad (2.1)$$

defines the transition rate from state q to an operational state l. The summation is defined over all transitions that fire exponentially and generate the same marking l. It is interesting to notice the relationship of equation (2.1) with the firing rate of a particular transition t_i given by the vector product $B_i \hat{M}_q^T$. A transition to the failure state occurs for lack of support (i.e., enough spares) or lack of coverage. In the first case, lack of support occurs if the reconfiguration algorithm failed due to exhaustion of spare components or the fact that the array fails to reconfigure if a given distribution of faults is not supported by the reconfiguration algorithm. Denote by λ_{qf} the transition to the failure state (f) for lack of support, then

$$\lambda_{qf} = [\sum_i Pr^i_{qf} b_j;\ b_j \epsilon B_i, j=1,...,k]\hat{M}^T_q$$

Let λ^c_{qf} be the transition to f for lack of coverage then:

$$\lambda^c_{qf} = \sum_l ([\sum_i (1-c_i) Pr^i_{ql} b_j;\ b_j \epsilon B_i, j=1,...,k]\hat{M}^T_q)$$

The overall transition rate to the failure state is:

$$\hat{\lambda}_{qf} = \lambda_{qf} + \lambda^c_{qf}$$

The diagonal term of the matrix A is calculated as follows:

$$a_{qq} = -(\sum_{l \neq q} a_{ql} + \hat{\lambda}_{qf}) = -(\sum_{l \neq q} \lambda_{ql} + \lambda_{qf})$$

where:

$$\lambda_{ql} = [\sum_i Pr^i_{ql} b_j;\ b_j \epsilon B_i, j=1,...,k]\hat{M}^T_q$$

Example 2.3: Consider a particular marking say M_{18} = [12 24 20 4 9 11], then with the following failure rates: $\alpha_0 = 1$, $\alpha_1 = \alpha_2 = \cdots \alpha_k = .01$, yield $\hat{M}_{18}$ = [12 .24 .20 .04 .09 .11]. As in example 2.2, the following probabilities are used: $Pr^1_{18,30} = 0.667$, $Pr^1_{18,f} = 0.333$, $Pr^2_{18,f} = 1$. Let $c_1 = c_2 = c$ then, when t_1 fires the following transition rates in the Markov chain are generated: $a_{18,f} = c[.667\ .667\ 0\ 0\ .667\ 0]\hat{M}^T_{18} = 8.2241c$. The transition to the failure state due to lack of support when t_1 and t_2 fire is $\lambda_{18,f} = ([.333\ .333\ 0\ 0\ .333\ 0]+[0\ 0\ 1\ 1\ 0\ 0])\hat{M}^T_{18} = 4.34589$. For lack of coverage $\lambda^c_{qf} = (1-c)[.667\ .667\ 0\ 0\ .667\ 0]\hat{M}^T_4 = 8.2241(1-c)$. The overall transition rate to the failure state is given as $\hat{\lambda}_{18,f} = 8.2241*(1-c) + 4.34589$. Considering that a failure of a SBL (t_0) yields a transition rate $a_{18,19} = [0\ 0\ 0\ 0\ 0\ 1]\hat{M}_{18}^T = 0.11$ then the diagonal term is calculated as: $a_{18,18} = 8.2241+.11+4.34589 = 12.68$. □

2.4.4. Implementation Procedure

The implementation procedure MODELGEN outlined below merges repeated markings as they are being generated and calculates or modifies the transition rates in the process. The new markings generated every iteration, are targets of the currently visited marking; they are inserted into a linked list of markings. As they are generated, markings are sorted with respect to the sum of those components indicated by the user through comparison flags previously set. An array of pointers to the newly created targets is updated. A pointer to the current marking is denoted by *cm*. A pointer to the next marking in the list is referred to by *next*. A systematic indexing of markings is carried out such that the resulting transition rate matrix is always upper triangular; a marking number is assigned to every next marking fetched from the sorted list; a pointer *nxtopr* points to markings which are candidates to be printed or saved in a file. If all its targets have been numbered, then a marking is saved or printed out; therefore, no significant amount of memory is required to generate large models. The procedure stops when all markings have been fetched from the sorted list. Notice that the linked list corresponds to the reachability graph and as it is formed, both the transition rate matrix representation of the Markov model and the reachability graph are printed out or saved as requested by the user.

Procedure MODELGEN

Inputs: Set of failure rates (α_i); File names (to save the matrix representation of the Markov model); An MSPN representation of the reconfiguration algorithm (Procedure MSPN outlined previously).

```
    Outputs: Reachability graph description; Matrix representation of the
    Markov model.
Begin
    Set a coverage flag for each transition that it is fired by non-active
    components. (this allows the evaluation of a symbolic matrix for different
    coverage values).
    Set comparison flags (to select those components by which the list of
    markings is sorted).
    Load initial marking
    for each $t_i$ let $s_i = B_i \underline{\alpha}^T$
    Let $cm$ point to initial marking
    $nxtopr = cm$
    while not end of list do
        fetch current marking
        assign a number to current marking
        for each $t_i$ and if $s_i > 0$ do
            get $P(x \mid M_q, t_i)$
            fire $t_i$ and those transitions $t_l \epsilon S_i$
            calculate transition rates
            store targets in temporary table
        end for
        merge repeated targets
        insert new targets in sorted list
        insert pointers to new targets in current marking
        while all targets of $nxtopr$ are numbered do
            output $nxtopr$ marking
            let $nxtopr = nxtopr \longrightarrow next$
        end while
        let $cm = cm \longrightarrow next$
    end while
end procedure
```

The execution time of MODELGEN is proportional to the number of states generated and therefore depends on the reconfiguration algorithm. For the cases given in Table 3.2 in chapter 3 the execution time is $O(n^3)$ for n$\times$n processor arrays.

2.5. Summary

A systematic procedure to construct Markov models was discussed in this chapter. Using an SPN-like representation, transition firing represents the occurrence of faults; A place contains a number of tokens tokens which correspond to the number of components of a single type; thus, the collection of different components is represented by a marking in an SPN-like representation and a marking corresponds to an operational configuration of the array. However an SPN-like representation fails to take into account the distribution of faulty spares that exist when reconfiguration takes place and may cause the array to fail. Also the fact that several component types causes the same effect in the array when it reconfigures, leads to a more compact representation of the array which is referred to as an MSPN representation.

A mapping from transitions and markings in an MSPN representation to transitions and states in the Markov model was derived. This mapping allows the construction of the corresponding Markov model as the generation of the reachability graph is taking place.

The more the detail (number of components or places in the MSPN representation) that is included in the model the larger is the state space in the resulting Markov model. In summary the application of this method entails two interrelated problems. In the first place a thorough analysis of the array is required such that all interdependencies of failures of different components are defined; this can be a difficult task for complex reconfiguration schemes; secondly, in some applications a detailed modeling may be limited to small size arrays because of the large number of states generated. Finally, considering the fact that reconfiguration algorithms are primarily designed to treat failures of PE's elements only, an MSPN representation for a given reconfiguration scheme, will depend on the assumptions made as to how the algorithm treats failures on component types other than PE's.

CHAPTER III
MSPN APPLICATIONS

3.1. Introduction

In this chapter three reconfiguration schemes are analyzed in detail. Algorithms to calculate probabilities of reconfiguration are derived. As stated in the previous chapter, probability functions $P(x \mid M_q, t_i)$ are associated with each transition t_i in the MSPN; probabilities of reconfiguration are used to determine Probability Transition Vectors which are in turn used to calculate transition rates in the Markov model. The probability of survival corresponds to the sum of probabilities of reconfiguration if the new markings generated are all operational; i.e. if the new markings correspond to operational configurations. In the second section of this chapter two algorithms are derived to calculate probabilities of reconfiguration for the SRE scheme. The third and fourth sections deal in detail with the ARCE and DR reconfiguration schemes, respectively. Expressions to determine the number of states for each reconfiguration scheme are given in section five. Also, in this section, results of the reliability analysis of each scheme are reported.

3.2. SRE Reconfiguration Scheme

In chapter 2, the basic concepts underlining MSPN's representations of fault tolerant processor arrays were thoroughly discussed. Concurrently and for the purpose of illustration, the SRE reconfiguration scheme was analyzed. Hence, in this section, expressions to estimate probabilities of reconfiguration for this scheme are derived.

The process of generating the reachability graph which describes the corresponding Markov model of an SRE scheme, starts with an initial marking given as follows:

p_1: $\#PE = n^2$

p_2: $\#IOL = 2n^2$

p_3: $\#S = n+n^2$

p_4: $\#BL = 0$

p_5: $\#HL = n(n-1)$

p_6: $\#SBL = n^2$

A failure marking corresponds to the case when $\#PE$=0 or $\#SBL$<0. Transition t_1 will take place if either α_1, α_2 or α_5 is greater than zero; t_2 will take place if either α_3 or α_4 is greater than zero. Likewise, t_3 takes place if α_5>0.

3.2.1. Probability Transition Vectors in SRE

To derive PTV's, estimations of probabilities of reconfiguration are required for each reconfiguration scheme. In the SRE case, its MSPN indicates (Fig. 3) that when t_1 fires, two transitions in the Markov model may occur (corresponding to a sequence of two firings in the MSPN: t_1 followed by t_2); one with probability

Pr^1_{qj} which corresponds to the probability of survival denoted by Ps. The other transition fires immediately and will lead the processor array to a failure state with probability Pr^1_{qf}. The array will survive with probability Ps if the failure of a PE or an IOL that triggers the reconfiguration algorithm occurs in a row that contains no faulty SBL. Let N denote the number of faulty SBL in a marking M_q then N is determined by:

$$N = \#PE_q - \#SBL_q = m_{1q} - m_{6q}$$

where $0 \leq N \leq n \times n$.

To estimate Ps, it is necessary to find all the possible ways in which N faulty SBL's can be mapped into a total of $r \times c$ SBL's in the array with a current configuration containing r rows and c columns. To simplify this problem we count all the possible partitions of N into r parts with a value not greater than c columns. Because each row contains c SBL's, up to c faulty SBL's per row are possible.

Each partition of N is a k-tuple $(a_1, a_2, \ldots, a_k)$, such that:

$$N = a_1 + a_2 + \ldots + a_k \tag{3.1}$$

where: $c \geq a_i \geq a_{i+1} \geq 1$ for $1 \leq i \leq k-1$ and $1 \leq k \leq r$.

In [Ber81] it is shown that the number of partitions of N with k or less parts and $N \geq a_i \geq a_{i+1} \geq 1$, is given by the recurrence relations:

$$F(N-k, k) = F(N, 1) + F(N, 2) + \ldots + F(N, k) \tag{3.2}$$

$$F(N, 1) = F(N, N) = 1$$

The function $F(N, k)$ is interpreted as the number of partitions of N with k parts.

Theorem 1: Let us denote by $F(N)$ the number of wanted partitions of N; i.e. those partitions with at least one part $a_i > c$ are excluded. Then $F(N)$ is given by:

$$F(N) = F(N,c,1) + \ \dots\ + F(N,c,r) \tag{3.3}$$

where:

$$F(N,c,k) = \begin{cases} 0 & \text{if } N > kc \\ \sum_{l=\lceil N/k \rceil}^{c} F(N-l,l,k-1) & \textit{otherwise} \end{cases} \tag{3.4}$$

and

$$F(N,c,1) = F(N,c,N) = 1 \text{ if } N \leq c \tag{3.5}$$

$$F(N,c,k) = 0 \ \text{ if } k > N \tag{3.6}$$

Proof: Let us consider first the case when $N > kc$. Each term in the summation is recursively decomposed until the functions generated satisfy equations (3.5) and (3.6). The value of l in each term of the summation corresponds to a part a_i in the ith iteration. The partitions generated will contain a part a_i such that $\lceil N/k \rceil \leq a_i \leq c$. Hence, if $N/k > c$ implies all partitions will have at least the first part $a_1 \geq c$ and $F(N,c,k) = 0$ follows.

Let us examine now the case when $N \leq ck$. $F(N,c,k)$ is decomposed into $(c - \lceil N/k \rceil + 1)$ functions, which in turn are recursively decomposed until the cases of equations (3.5) or (3.6) are reached. In the process those functions in which $N > kc$ or $k > N$ are eliminated.

The number of terms in the sum corresponds to all the possible values a part may take and since the minimum at every iteration is $a_i = \lceil N/k \rceil$, we let $a_1 \leq c$ as an upper bound for the first iteration which gives the number of summands indicated previously. For the next iteration the process is repeated. At the *ith* iteration, $N = N - a_i$ and k has been decremented to indicate that now the new value of N is to be partitioned in $k-i$ parts such that $\lceil N/k \rceil \leq a_i \leq a_{i-1}$.

Equations (3.3-3.6) can also be used to determine the value of each part in a particular partition since in the *ith* iteration the index l in equation (3.4) corresponds to the value of a_i whereas $N-l$ corresponds to the remaining value to be partitioned among the $k-1$ remaining parts. □

Example 3.1: To illustrate the application of the above relations, let us find all the possible distributions of 8 faulty SBL's in a 4×4 array (r=c=n).

Solution: With $N = 8$, $n = 4$ and $r = 4$ we have:

$F(8) = F(8,4,1) + F(8,4,2) + F(8,4,3) + F(8,4,4)$

$F(8,4,1) = 0$ *since* $8 > 4$

$F(8,4,2) = F(4,4,1) = 1$

$F(8,4,3) = F(5,3,2) + F(4,4,2)$

$= F(2,3,1) + F(2,2,1) + F(1,3,1) + F(0,4,1) = 1 + 1 + 1 = 3$

$F(8,4,4) = F(6,2,3) + F(5,3,3) + F(4,4,3)$

$= F(4,2,2) + F(3,2,2) + F(2,3,2) + F(2,2,2) + F(1,3,2) + F(0,4,2)$

$= F(2,2,1) + F(1,2,1) + F(1,1,1) + F(1,1,1) + 0 + 0 = 4$

And $F(8) = 0 + 1 + 3 + 4 = 8$

Table 3.1, shows the order of the resulting partitions, where partition 1) is given by F(8,4,2); partitions 2-4 are given by F(8,4,3); and partitions 5-8 are given by F(8,4,4). □

Table 3.1. Partitions of N = 8

Partition No.	Parts a_1	a_2	a_3	a_4
1	4	4	0	0
2	3	3	2	0
3	4	2	2	0
4	4	3	1	0
5	2	2	2	2
6	3	2	2	1
7	3	3	1	1
8	4	2	1	1

The probability of survival Ps can be found by two methods using in both cases the Total Probability Theorem [Tri82]. In the first method:

$$Ps = \sum_{i=1}^{F(N)} P(s \mid i)P(i) \tag{3.7}$$

The term $P(s \mid i)$ is the probability of success given the ith partition occurs and $P(i)$ is the probability that the ith partition occurs. These probabilities are obtained as follows:

$$P(s \mid i) = \frac{z_i}{r} \qquad P(i) = \frac{\pi_i}{\pi}$$

where z_i is the number of zeros in the ith partition, $\pi_i = \binom{r}{\tau_1 \tau_2 \cdots \tau_m} \prod_{i=0}^{r-1} \binom{c}{a_i}$ corresponds to the number of mappings represented by the ith partition with m distinct parts repeated τ_l $(1 \leq l \leq m)$ times. The term $\pi = \sum_{i=1}^{F(N)} \pi_i = \binom{rc}{N}$ is the total number of possible mappings of N faulty SBL's into an array with $r \times c$ number of PE's. After some manipulations on equation (3.7), the following expression results:

$$Ps = \frac{1}{r\pi} \sum_{i=1}^{F(N)} z_i \pi_i \tag{3.8}$$

A drawback in this procedure is that an enumeration of the possible partitions of N is required and this is a lengthy task for large processor arrays. A second method more attractive for large arrays is given in the following:

Theorem 3.2: Denote by k the number of rows (number of parts in the partition) with i faults (k parts with same value i), let π be defined as above and let $\theta(k,i)$ be the total number of mappings with exactly k rows with i faults then the probability of survival is given by the following expression:

$$Ps = \sum_{k=1}^{m} \begin{pmatrix} \frac{k}{r} \end{pmatrix} \frac{\theta(k,i)}{\pi} \tag{3.9}$$

where

$$\theta(k,i) = \sum_{j=k}^{m} (-1)^{j+k} \binom{j}{k} \binom{r}{j} \binom{(r-j)c}{N-ij} \binom{c}{i}^{j}$$

where $0 \leq i \leq c$ and m is determined such that $0 \leq N-im \leq (r-m)c$.

Proof: To find $\theta(k,i)$, denote by A_k the number of mappings with at least k rows with i faults, then

$$A_k = \binom{r}{k} \binom{(r-k)c}{N-ik} \binom{c}{i}^{k}$$

The first binomial corresponds to the number of ways k is combined in r rows; the second binomial gives the number of mappings of the remaining faults into the remaining spots (PE's) in the array; the last binomial corresponds to the number of ways in which i faults are mapped into c PE's in one column in each of the k rows. Because A_k counts also A_l for $k<l\leq m$ by a factor of $\binom{l}{k}$, $\theta(k,i)$ can be obtained subtracting all mappings with l rows; i.e.:

$$\theta(k,i) = A_k - \sum_{l=k+1}^{m} \binom{l}{k} \theta(l,i)$$

By letting $\theta(m,i) = A_m$ an expression of $\theta(k,i)$ is obtained recursively in terms of A_k. After substituting A_k the expression for $\theta(k,i)$ given above follows.

To obtain Ps, let $a_{k,i}$ be the set of partitions with k rows with i faults then the probability of survival given $a_{k,i}$ is k/r and using the Total Probability Theorem the probability of survival is given by:

$$Ps = P(s) = \sum_{k=1}^{m} P(s \mid a_{k,i}) P(a_{k,i})$$

where $P(a_{k,i}) = \dfrac{\theta(k,i)}{\pi}$ and equation (3.9) follows. □

In the particular case of the SRE scheme, Ps is obtained by letting i=0 in equation (3.9).

3.3. ARCE Reconfiguration Scheme

In the ARCE scheme, a row elimination is followed by a column elimination [FoR85]. Figure 3.1, shows a section of an n×n array supporting ARCE. Since the number of rows and columns varies with each configuration, then at a given configuration, r denotes the number of rows and c the number of columns. The array is in a ***r-configuration*** (r=c) if the reconfiguration algorithm deletes a row; Likewise, it is in a ***c-configuration*** ($r < c$) if the reconfiguration algorithm deletes a column; In Figure 3.1, the following components are identified: PE's; I/O Links, which are the interconnections between switches and PE's; Column Switches (CS), which are the switches in the current c columns of the array; Row Switches (RS), these are the switches in the current r rows of the array. The spare bypass links (SBL) used to bypass a faulty row become active links and are referred to as Row Bypass Links (RBL's). Likewise, the spare bypass links used to bypass a faulty column become active links and are referred to as Column Bypass Links (CBL's).

In the case of ARCE, the following assumptions are made:

1) The failure of a PE or an IOL, causes the elimination of a row if the array is in a r-configuration; it causes the elimination of a column if it is in a c-configuration.

2) All switches in the current configuration are considered active.

3) The failure of a RS in a r-configuration, causes the elimination of that row; in a c-configuration, a column is eliminated followed by the affected row to comply with the alternate column-row deletion process.

4) The failure of a CS in a c-configuration, causes the deletion of the affected column; in a r-configuration a row and the affected column are eliminated.

5) The failure of a RBL, causes the same effect as the failure of a RS.

6) The failure of a CBL, causes the same effect as the failure of a CS.

7) If a row is deleted, the spare bypass links used to bypass that row must be fault free. Otherwise, the row is deleted followed by the column deletion with the faulty spare link; this sequence is referred to as a *row-column deletion* sequence. Likewise, if a column is deleted the spare bypass links used to bypass that column must be fault free; otherwise, the column is deleted followed by the row with the faulty spare link; this sequence is referred to as *column-row deletion* sequence.

3.3.1. MSPN Representation of the ARCE Scheme

Any operational configuration C_i for the ARCE scheme is described as follows:

$$C_i = (\eta_{1i}, \eta_{2i}, \ldots, \eta_{7i}) = (\#PE, \#IOL, \#CS, \#RS, \#RBL, \#CBL, \#SBL)$$

An operational configuration C_i corresponds to an operational marking M_i.

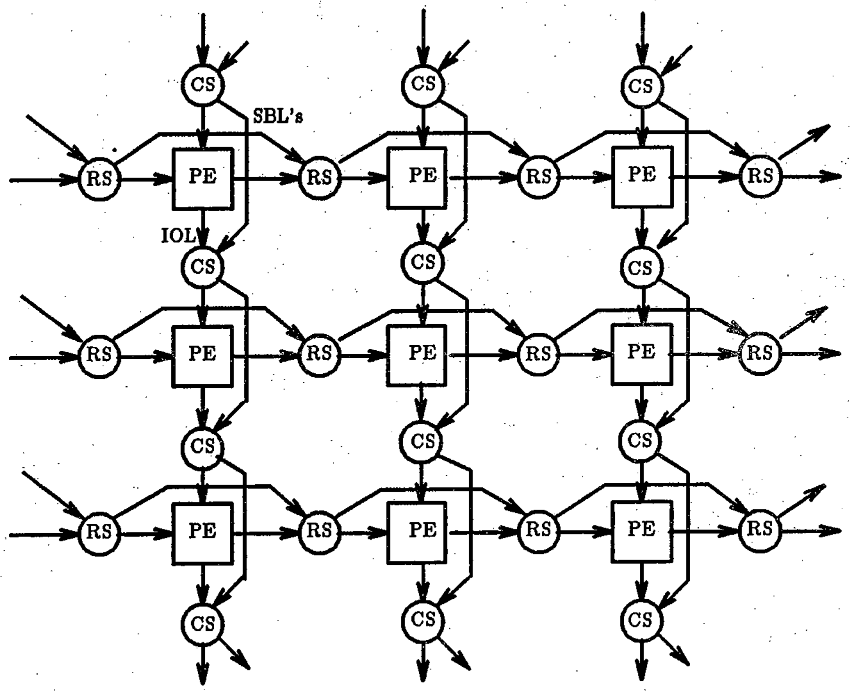


Figure 3.1.- Schematic Layout of the ARCE Structure

An initial configuration C_0 in terms of the size n of the array, is given by the following set of equations:

p_1: $\#PE = n^2$,

p_2: $\#IOL = 4n^2$,

p_3: $\#RS = (n+1)n$,

p_4: $\#CS = (n+1)n$,

p_5: $\#RBL = 0$,

p_6: $\#CBL = 0$,

p_7: $\#SBL = 2n^2$.

In any configuration C_i described by a marking M_i, the number of rows r and columns c are calculated as follows:

$$r = \lfloor (m_{1i})^{1/2} \rfloor \qquad c = \lceil (m_{1i})^{1/2} \rceil$$

The difference $d = c - r = 0$ identifies r-configurations and $d = 1$ identifies c-configurations.

A transition to a failure marking occurs if the current marking contains not enough PE's or not enough SBL's. To identify this condition in each marking M_q, a flag x is set or reset accordingly:

$$x = \begin{cases} 1 & \text{if } m_1 > m \text{ and } m_6 \geq 2m \\ 0 & otherwise \end{cases}$$

where m and $2m$ refer to the number of PE's and SBL's, respectively, eliminated by the reconfiguration scheme. Notice that m takes values depending on the type of the current configuration.

A place p_i and a transition $\hat{t}_i$ is assigned to each component type as listed above. For each $\hat{t}_i$ the corresponding set A_i of I/O places and set of multiplicities V_i are determined next.

According to assumption 1), $\hat{t}_1$ (failure of a PE) affects all the components in a column or a row; hence:

$$A_1 = \{I(\hat{t}_1), O(\hat{t}_1)\} = \{\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}, \{p_6, p_7\}\}$$

The set V_1 of variable multiplicities is given by the following set of equations:

a) $\mu_1^I = rx + m_{1q}(1-x)$

b) $\mu_2^I = 4rx + m_{2q}(1-x)$

c) $\mu_3^I = (n+1)dx + m_{3q}(1-x)$

d) $\mu_4^I = (n+1)(1-d)x + m_{4q}(1-x)$

e) $\mu_5^I = (n-r)dx + m_{5_q}(1-x)$

f) $\mu_6^I = (n-c)(1-d)x + m_{6_q}(1-x)$

g) $\mu_7^I = 2rx + m_{7_q}(1-x)$

h) $\mu_5^O = r(1-d)x - m_{5_q}(1-x)$

i) $\mu_6^O = rdx - m_{6_q}(1-x)$

This set of equations indicate that when $\hat{t}_1$ fires then: a) r PE's are deleted; b) 4 IOL's for each PE eliminated are also eliminated; c) in a c-configuration, $n+1$ CS's are deleted; d) in a r-configuration $n+1$ RS's are deleted; e) $n-r$ RBL's are deleted in a r-configuration; this number corresponds to the RBL's bypassing the PE's in the rows already eliminated; however, no RBL's are deleted if the current configuration is a c-configuration; f) no CBL's are deleted in a c-configuration but in a r-configuration, $n-c$ CBL's need to be deleted which are those CBL's bypassing PE's in the columns already eliminated; g) since there are two SBL's for each PE, $2r$ SBL's are deleted in either configuration; h) the number of RBL added corresponds to the number of PE's in a deleted row; i) the number of CBL's added corresponds to the number of PE's in a deleted column.

The failure of an IOL ($\hat{t}_2$), as stated in assumption 1) causes the same effect as $\hat{t}_1$. Therefore $A_2 = A_1$ and $V_2 = V_1$.

The failure of a CS ($\hat{t}_3$) as stated in assumption 4), will have the following set A_3:

$$A_3 = \{I(\hat{t}_3), O(\hat{t}_3)\} = \{\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}, \{p_5, p_6\}\}$$

Let $m = r + (c-1)(1-d)$ then the set V_3 of variable multiplicities is given as follows:

a) $\mu_1^I = mx + m_{1_q}(1-x)$

b) $\mu_2^I = 4mx + m_{2q}(1-x)$

c) $\mu_3^I = (n+1)x + m_{3q}(1-x)$

d) $\mu_4^I = (n+1)(1-d)x + m_{4q}(1-x)$

e) $\mu_5^I = (n-r)x + m_{5q}(1-x)$

f) $\mu_6^I = (n-c)(1-d)x + m_{6q}(1-x)$

g) $\mu_7^I = 2mx + m_{7q}(1-x)$

h) $\mu_5^O = (c-1)(1-d)x - m_{5q}(1-x)$

i) $\mu_6^O = (r+d-1)x - m_{6q}(1-x)$

Thus when $\hat{t}_3$ fires, equation a) shows that in a c-configuration, only r PE's are eliminated; however in a row configuration, to delete a column a row must be deleted first giving a total of $r+(c-1)$ PE's eliminated. In general given a r-configuration, to delete a column, the sequence row-column deletion applies and given a c-configuration, to delete a row, the sequence column-row deletion applies. In equation b), for every PE that is eliminated 4 IOL's are also eliminated. In c) shows that in either configuration, all the CS's in the affected column are eliminated. Equation d) shows that in a r-configuration all the RS's in the affected row are eliminated. Equation e) indicates that in either configuration $(n-r)$ RBL's are eliminated. Equation f) shows that in a c-configuration, $n-c$ is the number of rows that have been eliminated, so the same number of active CBL's is eliminated. However none are eliminated if a column is deleted. Equation g) deletes 2 SBL's for each PE deleted. Equation h) shows that in a c-configuration the row-column sequence activates only $c-1$ RBL's; none is activated in a r-configuration. Equation i) shows that $r-1$ CBL's are activated corresponding to the column deleted in a row-column sequence that applies in a c-configuration; in a r-configuration only a column is eliminated and so are the r CBL's in it.

The effect of the failure of a RS ($\hat{t}_4$), as stated in assumption 3), is described by the following set A_4:

$$A_4 = \{I(\hat{t}_4), O(\hat{t}_4)\} = \{\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}, \{p_5, p_6\}\}$$

with the following set of variable multiplicities V_4 where $m = r + (c-1)d$:

a) $\mu_1^I = mx + m_{1q}(1-x)$

b) $\mu_2^I = 4mx + m_{2q}(1-x)$

c) $\mu_3^I = (n+1)dx + m_{3q}(1-x)$

d) $\mu_4^I = (n+1)x + m_{4q}(1-x)$

e) $\mu_5^I = (n-r)dx + m_{5q}(1-x)$

f) $\mu_6^I = (n-c)x + m_{6q}(1-x)$

g) $\mu_7^I = 2mx + m_{7q}(1-x)$

h) $\mu_5^O = mx - m_{5q}(1-x)$

i) $\mu_6^O = (r-1)dx - m_{6q}(1-x)$

These equations are essentially the same as for t_3, except that in case of a c-configuration, the failure of a RS, implies the deletion of the affected row therefore the sequence column-row deletion applies. Note that equation h) shows that the number of CBL's activated corresponds to the number of PE's eliminated in a c-configuration; i.e. c CBL's; in the case of a r-configuration $r = c-1$ CBL's are activated. Likewise, the number of RBL's activated by equation i), corresponds to the number of PE's eliminated in a r-configuration; none in a c-configuration.

The failure of a RBL ($\hat{t}_5$) as stated in assumption 6), will have the same effect as the failure of CS; therefore $A_5 = A_3$ and $V_5 = V_3$.

The failure of a CBL ($\hat{t}_6$), has the same effect as the failure of a RS; therefore $A_6 = A_4$ and $V_6 = V_4$.

The failure of a SBL ($\hat{t}_7$), will decrease the number of SBL's by one and:

$A_7 = \{I(\hat{t}_7), O(\hat{t}_7)\} = \{\{p_7\}, \{\emptyset\}\}$

With a set V_7 of variable multiplicities as follows:

$\mu_7^I = 1$

Transitions $\hat{t}_1$ and $\hat{t}_2$ have same set of I/O places and same set of multiplicities; therefore transition t_1 of the MSPN is defined with $A_1^* = A_1 = A_2$ and $V_1^* = V_1 = V_2$.

Likewise $\hat{t}_3$ and $\hat{t}_5$ define transition t_2 of the MSPN with $A_2^* = A_3 = A_5$ and $V_2^* = V_3 = V_5$.

Transitions $\hat{t}_4$ and $\hat{t}_6$ define transition t_3 of the MSPN with $A_3^* = A_4 = A_6$ and $V_3^* = V_4 = V_6$.

Finally transition $\hat{t}_7$ define t_4 with $A_4^* = A_7$ and $V_4^* = V_7$.

The set of transitions t thus obtained with the corresponding sets of I/O places and variable multiplicities describe the MSPN of the ARCE scheme shown in Figure 3.2.

The transition vectors associated with the transitions t are the following:

$$B_1 = [1\ 1\ 0\ 0\ 0\ 0\ 0]$$

$$B_2 = [0\ 0\ 1\ 0\ 1\ 0\ 0]$$

$$B_3 = [0\ 0\ 0\ 1\ 0\ 1\ 0]$$

$$B_4 = [0\ 0\ 0\ 0\ 0\ 0\ 1]$$

Notice that in ARCE, t_1 will not occur if $\alpha_1 = \alpha_2 = 0$, t_2 will not occur if $\alpha_3 = \alpha_5 = 0$, t_3 will not occur if $\alpha_4 = \alpha_6 = 0$ and t_4 will not occur if $\alpha_7 = 0$.

3.3.2. Probability Transition Vectors in ARCE

In ARCE, a successful reconfiguration takes place if 1) In a r-configuration ($r{=}c$), all r RBL's in the row to be deleted are fault free and 2) in a c-configuration ($r{<}c$), all r CBL's, in the column to be deleted are fault free.

In either case we are interested in the distribution of N_1 faulty spare RBL's and N_2 faulty spare CBL's, such that $N = N_1 + N_2$ is the total number of faulty spare links where $0 \leq N_1 \leq rc$, $0 \leq N_2 \leq rc$. In a marking q, the number N of faulty SBL's is obtained as follows:

$$N = 2m_{1q} - m_{7q}$$

where $m_{1q} = rc$ and $0 \leq N \leq 2rc$. The number of SBL's that must be fault free for a successful reconfiguration is always r. Let H(N) be the number of partitions of N into parts N_1 and N_2 such that $0 \leq N_1, N_2 \leq rc$, then

$$H(N) = \begin{cases} N+1 & N \leq rc \\ 2rc-(N-1) & N > rc \end{cases}$$

Assuming all possible partitions of N are equally likely, the probability that any $\hat{n}th$ partition will occur is $pr(\hat{n}) = 1/H(N)$.

Let X be a random variable which denotes the number of faulty sbl's (column spares in case of a r-configuration and row spares bypass links in case of c-configuration). Then $Pr(X{=}i \mid \hat{n})$ is the probability that there are i faulty spare links in the row or column to be deleted given the $\hat{n}th$ partition occurs; this probability is given by equation (3.9).

The total probability $Pr(X=i)$, which corresponds to the probability that i spare row/column links are faulty in the row/column to be deleted out of a total of N faulty spare links is given by:

$$Pr(X=i) = \sum_{\hat{n}=1}^{H(N)} Pr(X=i \mid \hat{n}) pr(\hat{n}) = \frac{1}{H(N)} \sum_{\hat{n}=1}^{H(N)} Pr(X=i \mid \hat{n})$$

$$= \frac{1}{H(N)} \sum_{\hat{n}=1}^{H(N)} \sum_{k=1}^{m} \left(\frac{k}{r} \right) \frac{\theta(k,i)}{\pi} \tag{3.10}$$

Since $0 \leq i \leq r$, expression (3.10) implies that there will be at most $r+1$ transitions from a given state to up to $r+1$ different new states. Each transition is weighted by $Pr(X=i)$ such that $\sum_{x=0}^{r} Pr(X=i) = 1$. Every term in this summation corresponds to a transition probability as stated previously. The size of the set R of states directly reachable from any given state is for this case up to $r+1$ and includes the failure state if $X=r$.

Because every possible value of X, may trigger a different reconfiguration the following cases are observed:

1) *Case of transition* t_1.- If $i=0$, a row or column is deleted in a c or r-configuration respectively. However if $i>0$, i sequences of rc-deletions are required: in a r-configuration i faulty spare row bypass links become faulty active row bypass links so t_2 is applied i times; In a c-configuration i faulty spare column bypass links become faulty active column bypass links so t_3 is applied i times.

2) *Case of transition* t_2.- The application of t_2 implies rc-deletions in a r-configuration for all i. In a c-configuration however, if $i=0$, only the column affected is deleted; if $i>0$, then i rc-deletions occur and since the column spare bypass links in the columns deleted become active column bypass links then t_3 is

applied i times.

3) *Case of transition* t_3.- In this case, i rc-deletions are required in a c-configuration. However in a r-configuration, if i=0, only the affected row is deleted, otherwise i rc-deletions are necessary and since the faulty spare row bypass links become active row bypass links, t_2 is applied i times.

In summary, to capture these cases in an MSPN representation the sequences S_1, S_2 and S_3 are defined as follows:

$$S_1 = S_2 = S_3 = \begin{cases} (t_2, \cdots, t_2) & \text{if } d = 0 \\ (t_3, \cdots, t_3) & \textit{otherwise} \end{cases}$$

i.e., whenever t_1, t_2 or t_3 fire exponentially, a sequence of r immediate firings of t_2 or t_3 will follow depending whether the current marking corresponds to a r-configuration or a c-configuration.

Finally, we adopt the following criteria to determine the number of faulty spares that are passed on to the new configuration. When a single row or column is eliminated, the number of faulty spares passed on to the new configuration is given by:

$$N_1 = N_0 - r$$

In general after k firings of a sequence S_i have occurred, the following recursion can be applied:

$$N_k = N_{k-1} - m$$

where

$$m = \begin{cases} r & \text{for } t_1 \\ r+(c-1)d & \text{for } t_2 \\ r+(c-1)(1-d) & \text{for } t_3 \end{cases}$$

These expressions assure that the spares that become active and make the new configuration possible must be fault-free. The number of faults in a current configuration are distributed in the array between the rows and columns deleted and the new configuration. Therefore the new configuration will contain non-faulty spares if $N_k \leq 0$ otherwise N_k faulty spares are passed on to the new configuration. To represent this condition every time t_1, t_2 or t_3 fires, let a flag y be set or reset as follows:

$$y = \begin{cases} 1 & \text{if } N_k \leq 0 \\ 0 & otherwise \end{cases}$$

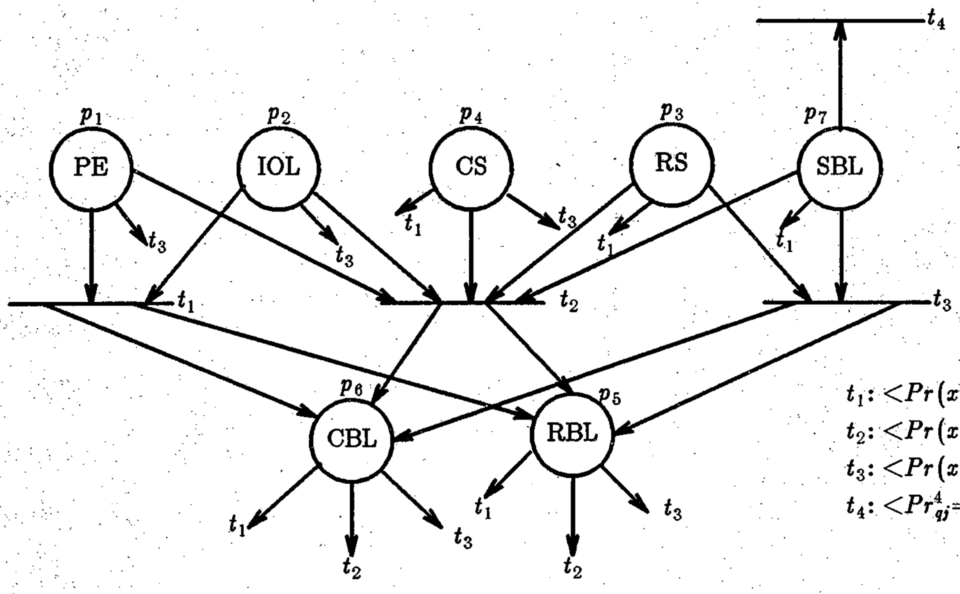
then the desired number of SBL's (m_{7j}) in a new marking j is obtained by modifying the input multiplicities associated with SBL's as follows:

$$\mu_7^I = ((2m - N_{k-1})y - m(1-y))x + m_7(1-x)$$

3.4. Direct Reconfiguration Scheme

In this section an specific case of reconfigurable processor arrays that follow a systematic chained replacement of faulty cells [SaS86a,SaS86b] is analyzed. Figure 3.3 shows a schematic layout of the hardware requirements for the Direct Reconfiguration in an n×n array where interconnections are implemented through multiplexers. For illustration, the Direct Reconfiguration (DR) algorithm is emphasized, since other schemes within this class (fixed fault-stealing, variable fault-stealing, complex fault-stealing, etc.,) [SaS86a,SaS86b] can be analyzed in a similar way.

Upon the application of the reconfiguration algorithm (which considers the total number of non-faulty cells), a faulty cell is replaced by a spare cell by simply reindexing non-faulty cells and bypassing the faulty one. Thus, a shift of



t_1: $<Pr(x \mid M_q, t_1), S_1, B_{1,} c>$
t_2: $<Pr(x \mid M_q, t_2), S_2, B_{2,} c>$
t_3: $<Pr(x \mid M_q, t_3), S_3, B_{3,} c>$
t_4: $<Pr^4_{qj}{=}1, \varnothing, B_{3,} 1>$

Figure 3.2. MSPN of the ARCE Scheme

functional cells or chained replacement is carried out when a faulty cell is detected. For the case of a single spare row and a single spare column, the reconfiguration algorithm scans each column of the array upwards and marks the first faulty cell as a vertical fault. All other faults are classified as horizontal faults. If one row has more than one horizontal faulty, reconfiguration is not possible along that row and the reconfiguration fails.

To analyze the DR scheme, the following components are considered: active cells (PE's); I/O links (IOL's); for each cell there are two Input Multiplexers (IMX's); for each cell there is one Output Multiplexer (OMX); Bypass Links (BL's) are active links which are used to bypass faulty cells; the Interconnection Links (IL's) are sets of links that define the interconnection network and it is assumed (to simplify the analysis) that there is one set per column and that the corresponding failure statistics can be provided; spare cells (SPE's); Spare IOL's (SIOL's); Spare Multiplexers (SMX's) which are the non-active multiplexers associated with spare cells; Spare Bypass Links (SBL's) are non-active bypass links.

In the analysis of this scheme the following assumptions are made:

1) The reconfiguration algorithm is applied to replace either spare or nominal faulty cells. In this analysis only one spare row and one spare column are assumed.

2) A cell is bypassed by any one of the spare bypass links shown in the diagram depending weather the reconfiguration is vertical or horizontal [SaS86a]. Since the functions of the bypassed cell are replaced by another cell, a link in the IL is activated acting as a bypass. Therefore, it can be assumed that always two bypass links will be used to bypass a faulty cell. Also, for a successful reconfiguration both spare bypass links must be fault-free.

3) The failure of an IL or an OMX or a BL is fatal.

5) The failure of an SMX or an SIOL disables the spare cell to which they are attached.

4) The failure of an IMX or an IOL disables the cell to which they are attached.

3.4.1. MSPN of the DR Scheme

The following set of equations shows the initial configuration of an n$\times$n array that supports the DR scheme with one spare row and one spare column. A place assigned to each component type is also indicated:

p_1: $\#PE = n^2$

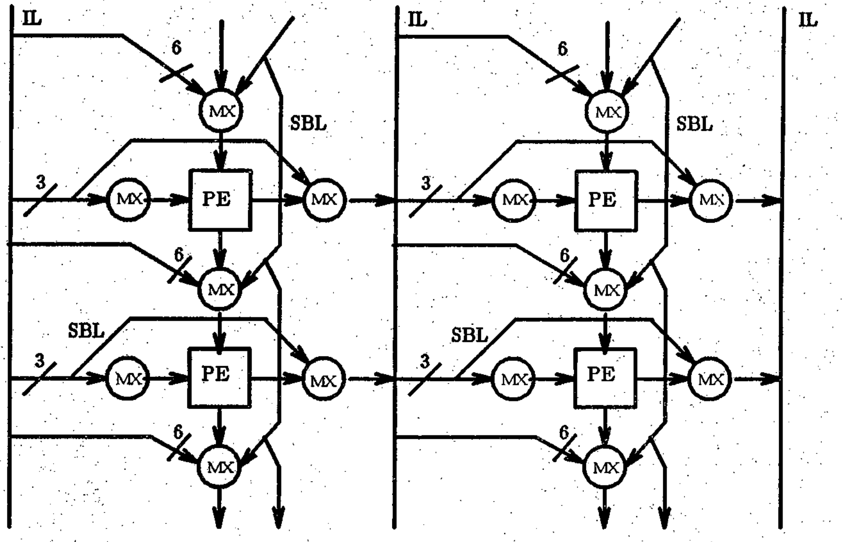


Figure 3.3.- Schematic Layout of the DR Structure

p_2: $\#IOL = 4n^2$

p_3: $\#IMX = 2n^2$

p_4: $\#OMX = n^2$

p_5: $\#BL = 2n$

p_6: $\#IL = n+1$

p_7: $\#SPE = 2n+1$

p_8: $\#SIOL = 4*(2n+1)$

p_9: $\#SMX = 3(2n+1)$

p_{10}: $\#SBL = 2n^2+2(n+1)$

Note that in the initial configuration the number of BL's is given by those active links required to bypass both the spare row and the spare column. The number of SMX's correspond to those multiplexers attached to spare cells; since they are not active a single place is assigned to them assuming all have the same failure rate; when a cell becomes active then its 3 SMX's are separated into 2 IMX's and 1 OMX. The total number of SBL's is given by those existing in the nominal array plus $(n+1)$ in the spare row and $(n+1)$ in the spare column.

To construct the MSPN, let us denote by t_1 the failure of an SBL $(\hat{t}_{10})$ with $A_1^* = A_{10} = \{\{p_{10}\},\emptyset\}$ and $V_1^* = V_{10}$ which contains only the equation $\mu_{10}^I = 1$.

Next, according to assumption 5, the failure of SIOL $(\hat{t}_8)$ or the failure of an SMX $(\hat{t}_9)$, cause the same effect as the failure of an SPE $(\hat{t}_7)$; therefore these three transitions define t_2 with $A_2^* = A_7 = A_8 = A_9 = \{\{p_7,p_8,p_9,p_{10}\},\{p_5\}\}$. The set of multiplicities associated with the set I/O arcs is defined as: $V_2^* = V_7 = V_8 = V_9$ which contain the following equations:

$\mu_7^I = 1$

$\mu_8^I = 4$

$\mu_9^I = 3$

$\mu_{10}^I = 1$

$\mu_5^O = 1$

By assumption 3, the failure of an IL $(\hat{t}_6)$ or the failure of an OMX $(\hat{t}_4)$ or the failure of a BL $(\hat{t}_5)$ will cause the array to fail. These transitions define t_3 in the MSPN with a set of I/O places $A_3^* = A_6 = A_4 = A_5 = \{\{ p_1,p_2,\ldots,p_{10} \},\emptyset\}$ and a set of multiplicities $V_3^* = V_6 = V_4 = V_5 = \{\{ \mu_1^I = m_{i1},\mu_2^I = m_{i2},\ldots,\mu_{10}^I = m_{i10}\}, \emptyset \}$ at any given *ith* marking.

Finally by assumption 4 the failure of a PE $(\hat{t}_1)$, the failure of an IOL $(\hat{t}_2)$ and the failure of an IMX $(\hat{t}_3)$ have the same effect on a given current configuration of the array; therefore transition t_4 of the MSPN with a set of I/O places $A_4^* = A_1 = A_2 = A_3 = \{\{p_7,p_8,p_9,p_{10},p_5\},\{p_5,p_{10}\}\}$. The corresponding set of multiplicities is $V_4^* = V_1 = V_2 = V_3$ which contain the following equations:

$\mu_7^I = 1; \mu_8^I = 4; \mu_9^I = 3; \mu_{10}^I = 2; \mu_5^I = 1; \mu_5^O = 2; \mu_{10}^O = 1; \mu_{10}^I = 2;$

The Binary Transition Vectors associated with the transitions obtained are the following:

$$B_1 = [0\,0\,0\,0\,0\,0\,0\,0\,0\,1]$$

$$B_2 = [0\,0\,0\,0\,0\,0\,1\,1\,1\,0]$$

$$B_3 = [0\,0\,0\,1\,1\,1\,0\,0\,0\,0]$$

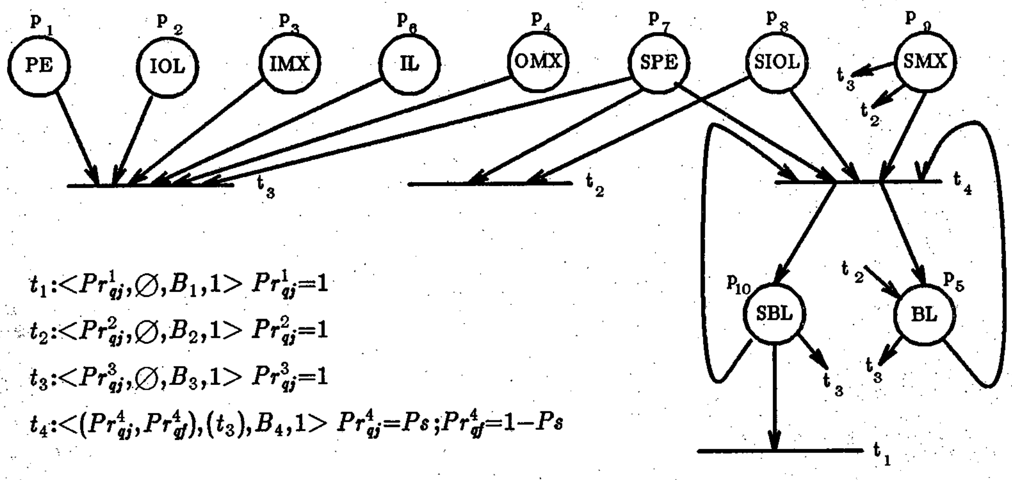
$$B_4 = [1\,1\,1\,0\,0\,0\,0\,0\,0\,0]$$

When t_4 fires because a PE or a IOL or an IMX failed, the the array reconfigures with a probability of survival $Ps{=}P_{jq}^4$ or fails with probability $1{-}Ps$. To capture the transition to the failure state, t_3 fires immediately after t_4 fires exponentially. Hence $S_4 = (t_3)$. The remaining transitions do not cause any immediate firing; therefore, $S_1{=}S_2{=}S_3{=}\emptyset$. The MSPN obtained for the DR scheme is shown in Figure 3.4.

Notice that t_1 will not occur if $\alpha_{10} = 0$, t_2 will not occur if $\alpha_7 = \alpha_8 = \alpha_9 = 0$, t_3 will not occur if $\alpha_4 = \alpha_5 = \alpha_6 = 0$ and t_4 will not occur if $\alpha_1 = \alpha_2 = \alpha_3 = 0$.

3.4.2. Probability Transition Vectors in DR

A successful reconfiguration in the DR scheme depends on the availability of spare cells upon the failure of an active cell and the fault-free condition of the spare bypass links which bypass the faulty cells. The number of faulty cells N_c and the number of faulty SBL's N_s at an *ith* configuration of an n×n array are given by:



t_1:$<Pr^1_{qj},\varnothing,B_1,1>$ $Pr^1_{qj}=1$

t_2:$<Pr^2_{qj},\varnothing,B_2,1>$ $Pr^2_{qj}=1$

t_3:$<Pr^3_{qj},\varnothing,B_3,1>$ $Pr^3_{qj}=1$

t_4:$<(Pr^4_{qj},Pr^4_{qf}),(t_3),B_4,1>$ $Pr^4_{qj}=Ps;Pr^4_{qf}=1-Ps$

Figure 3.4.- MSPN of the DR Scheme

$$N_c = 2nq+q^2-m_{i7}$$

$$N_s = 2n^2+2(n+q)+2nq-m_{i5}-m_{i10}$$

where q is the number of spare rows and columns, $m_{i7} = \eta_{i7}$, $m_{i5} = \eta_{i5}$ and $m_{i10} = \eta_{i10}$ are the number of spare cells, the number of active bypass links (BL's) and the number of spare bypass links (SBL's) in any *ith* configuration respectively.

A probability of transition Pr is one when t_1, t_2 or t_3 fire. However when t_4 fires the reconfiguration scheme is triggered and the array reconfigures with a probability $Ps = Pr^4_{qj}$ or fails with probability $1-Ps = Pr^4_{qf}$. Let $P(s \mid N_c)$ denote the conditional probability of success given N_c number of faulty cells and let $P(s \mid N_s)$ denote the conditional probability of success given N_s faulty SBL's. The probability of success is then given by:

$$Ps = P(s \mid N_c)P(s \mid N_s) \tag{3.11}$$

Thus, the presence of faulty SBL's in the array affects the overall successful reconfiguration rate. The conditional probabilities of success are derived in the next paragraphs.

Estimation of $P(s \mid N_c)$

To derive the probability of successful reconfiguration given N_c faulty cells, the following recursive expression given in [JaR88] is used. Starting from the $(q+1)$th row to the $(n+q)$th row of the array the probability that a fatal failure occurs in the first j rows is given by:

$$P_f(j) = \begin{cases} P_f(j-1)+(1-P_f(j-1))P_{ff}(j) & \text{for } j>r \\ 0 \quad otherwise \end{cases}$$

at the $(n+q)$th row the conditional probability is obtained by:

$$P(s \mid N_c) = 1-P_f(n+q) \tag{3.12}$$

where $P_{ff}(j)$ is the probability that a first fatal failure occurs in the *jth* row. Denote by $P_a(i)$ the probability that a row has exactly i faulty cells then:

$$P_a(i) = \frac{\begin{pmatrix} n+q \\ i \end{pmatrix}\begin{pmatrix} S-(n+q) \\ N_c-i \end{pmatrix}}{\begin{pmatrix} S \\ N_c \end{pmatrix}}$$

where $S=(n+q)^2$ is the total number of cells in the array.

Let $P_b(j)$ denote the probability that a column has at least q faulty cells in the first $j-1$ rows then this probability is obtained as follows:

$$P_b(j) = \sum_{k=q}^{j-1} \frac{\begin{pmatrix} j-1 \\ k \end{pmatrix}\begin{pmatrix} S-(j-1) \\ N_c-k \end{pmatrix}}{\begin{pmatrix} S \\ N_c \end{pmatrix}}$$

Now let $P_c(i,j)$ denote the probability that at least $q+1$ out of i columns have q faults or more in the first $j-1$ rows then:

$$P_c(i,j) = \sum_{l=q+1}^{i} \binom{i}{l} (P_b(j))^l (1-Pb(j))^{i-l}$$

The probability of first fatal failure in the *jth* row $P_{ff}(j)$ is obtained as follows:

$$P_{ff}(j) = \sum_{i=q+1}^{n+q} P_a(i) P_c(i,j)$$

The probabilities of survival given N_c for several array sizes are shown in Figure 3.5. These results compare closely with those obtained using simulation as reported in [SaS86a].

Estimation of $P(s \mid N_s)$

To estimate the probability of success given N_s faulty SBL's, let us assume that up to two SBL's per active cell may be faulty. Denote by p a place where a cell may have either none, one or two faulty SBL's then the number of possible faulty places denoted by N_p varies as follows:

$$\left\lceil \frac{N_s}{2} \right\rceil \le N_p \le \beta$$

where $\beta = \min\{N_s, 2(n^2-N_c)\}$. The probability of successful reconfiguration given N_s is given by:

$$P(s \mid N_s) = \frac{\displaystyle\sum_{N_p=\left\lceil \frac{N_s}{2} \right\rceil}^{\beta} \binom{n^2-N_c}{N_p}}{\displaystyle\sum_{k=\left\lceil \frac{N_s}{2} \right\rceil}^{\beta} \binom{n^2}{k}} \tag{3.13}$$

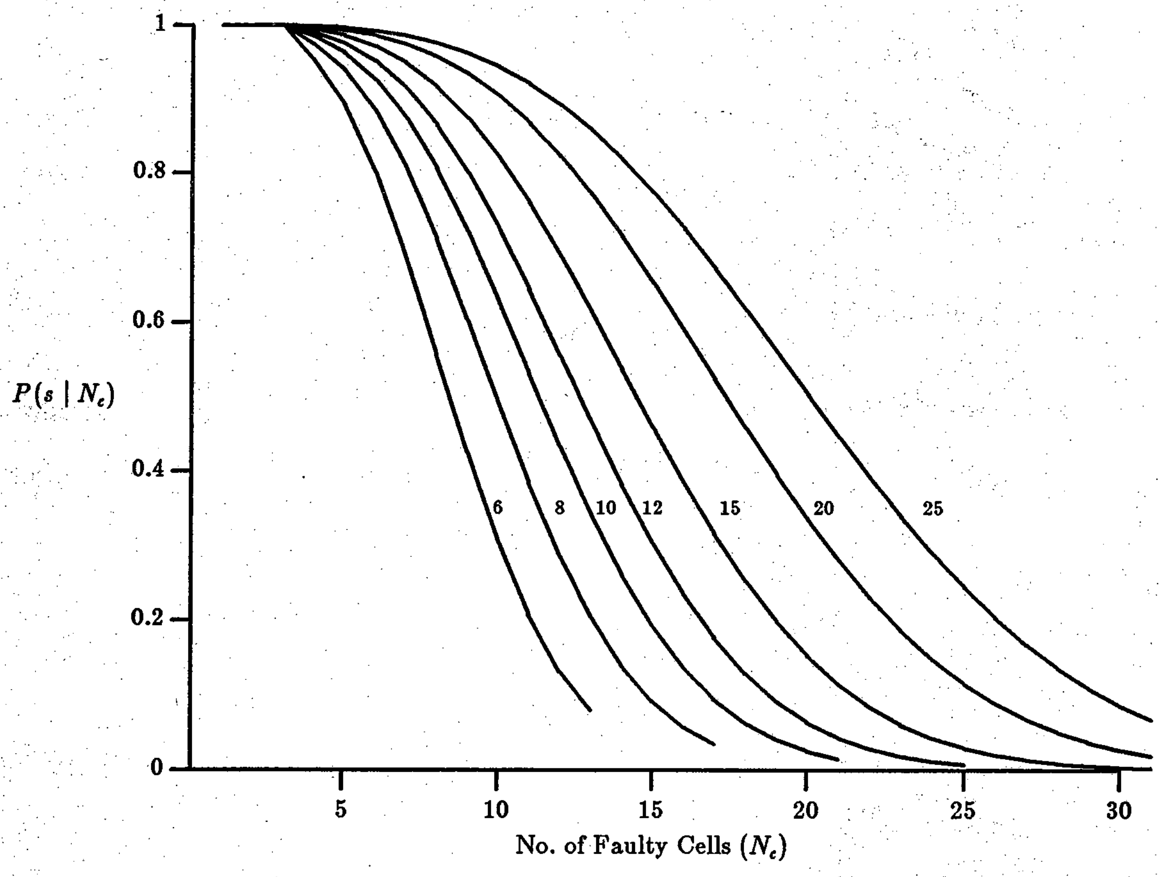


Figure 3.5.- Prob. of Surv. for Different Array Dimensions (DR)

To obtain equation (3.13) let $P(s \mid N_p)$ denote the probability of successful reconfiguration given N_p faulty places, then:

$$P(s \mid N_p) = \frac{\binom{n^2}{N_c}\binom{n^2-N_c}{N_p}}{\binom{n^2}{N_c}\binom{n^2}{N_p}}$$

the numerator of the above expression corresponds to the possible number of mappings of both faulty cells and faulty SBL's in which both types of faults do not coincide in a single place. The numerator corresponds to the total possible number of mappings (fault distributions) given N_p and N_c.

Let $P(N_p)$ be the probability that N_p faulty places exist. This probability is obtained as:

$$P(N_p) = \frac{\binom{n^2}{N_p}}{\sum_{k=\left\lceil \frac{N_s}{2} \right\rceil}^{\beta} \binom{n^2}{k}}$$

By the Total Probability Theorem, the conditional probability given N_s is given by:

$$P(s \mid N_s) = \sum_{N_p=\left\lceil \frac{N_s}{2} \right\rceil}^{\beta} P(s \mid N_p) P(N_p) \tag{3.14}$$

Finally, substituting and simplifying in the above expression yields (3.13).

3.5. Comparative Results

In this section expressions are derived to determine the state space of the three schemes analyzed. The state space size can be obtained through these expressions or through MODELGEN. Also some reliability results are reported for several sets of failure rates to compare the effect of detailed modeling with respect to an analysis in which only failure of PE's are considered.

3.5.1. State Space

For the schemes analyzed, Table 3.2 shows the growth of the state space as function of the size of the array. In this table all failure rates are assumed greater than zero in order to generate in each case, the maximum number of states. The maximum number of states tabulated, can be obtained using for each case closed form expressions.

To derive the number of states for the SRE case, let $S(n)$ denote the maximum number of states for an n×n array; then:

$$S_{SRE}(n) = \frac{1}{2}(n^3 + n^2 + 2n)$$

This expression is derived by observing that n states are generated with non-faulty SBL's; for each state i with non-faulty SBL's, $n(n-i)$ additional states are generated. Hence, the total number of states is obtained by solving the summation $S_{SRE}(n) = \sum_{i=0}^{n-1}[n(n-i)+1]$ to yield the above expression.

For the ARCE case the number of states $S(n)$ can be derived by observing that $(2n-1)$ states are generated with non-faulty SBL's. Each of these states generates additional states with a total number expressed by the following summation:

$$\sum_{i=0}^{2n-2}\{[2(n-i)^2+1] + [2(n-i)(n-i-1)+1]\}$$

from which the following recursion is derived:

$$S_{ARCE}(1) = 3 \qquad S_{ARCE}(n) = S_{n-1} + 4n^2 - 2n + 2$$

with a closed form solution given by::

$$S_{ARCE}(n) = (n-1)[\frac{1}{3}(4n^2 + 7n) + 4] + 3$$

The number of states $S_{DR}(n)$ in the case of DR, can be obtained by the following expression:

$$S_{DR}(n) = \sum_{i=0}^{2qn+q^2} [(2n^2+2(qn+q^2)+1)-i]$$

In this expression the summation is carried out over all states generated with i faulty SPE's. Solving this summation the following closed form results:

$$S_{DR}(n) = (2qn+q^2+1)[2n^2+2(qn+q^2)+1-(2qn+q^2/2)$$

simplifying for $q=1$, then

$$S_{DR}(n) = (n+1)(4n^2+2n+5)$$

Using either MODELGEN or the above equations the number of states generated in each case is shown in Table 3.2.

Table 3.2. State Space Size for three Reconfiguration Algorithms

n	SRE	ARCE	DR
2	8	17	75
3	21	51	188
4	44	107	385
5	80	199	690
6	132	333	1127
7	203	517	1720
8	296	759	2493
9	414	1067	3470
10	560	1449	4675

3.5.2. Reliability Analysis

To illustrate the applicability of MODELGEN, several sets of failure rates ($\alpha's$) have been selected and described in Table 3.3. For each failure rate set, the Reliability (R), the MTTF and the Reliability Improvement Factors (RIF) have been computed for a 4×4 processor array. The results obtained are tabulated in Table 3.4 for both the SRE and the ARCE reconfiguration schemes.

In our analysis, we have used the PE failure rate as a reference normalized with respect to the time unit such that $\alpha t = 1$ and with a coverage factor of $c = .99$. The computations were carried out using the MGRE (Model Generator and Reliability Evaluator) software package described in [Lop89].

The main purpose of the tabulations in Table 3.4, is to show the interdependencies of the different components now contained in the model. Notice for example in column (e), the sensitivity to switch failures of the array with SRE; i.e. columns (d) and (e) show the results obtained under similar failure rate values except for switches; a reliability improvement factor (RIF) is calculated in each case with respect to the simplex (sx) case, which corresponds to the case of the failure of the array when a single processor fails; at $t = .1$ a RIF of 15.91 in column (d) decreases to a RIF of 3.87 in column (e) as the failure rate of the switches increases from 0.01 to 0.1. The same effect is less noticeable in the array with ARCE; Compared to SRE, ARCE is less sensitive to failures in switches and links. The Mean Time to Failure of the array with ARCE improves in each case considerably with respect to the array with SRE. In summary column (b) shows reliability results obtained considering PE's failures only; on the other hand, columns (c-d) show the effect of detailed modeling obtained under different failure rate sets.

Table 3.3.- Failure Rates used for the results shown in Table 3.4.

Array	col.	α_1	α_2	α_3	α_4	α_5	α_6	α_7	Explanation
SRE	a	sx	-	-	-	-	-	-	α_1=pe f.rate
	b	1	.0	.0	.0	.0	.0	-	α_2=IOl f. rate
	c	1	.01	.0	.0	.01	.0	-	α_3=Switch f.rate
	d	1	.01	.01	.01	.01	.01	-	α_4=b.link f.rate
	e	1	.01	.1	.01	.01	.01	-	α_5=h.link f.rate
	f	1	.01	.01	.1	.01	.1	-	α_6=sp.b.link f.rate
	g	1	.01	.01	.0	.01	.0	-	-
ARCE	a	sx	-	-	-	-	-	-	α_3=c.switch f.rate
	b	1	.0	.0	.0	.0	.0	.0	α_4=r.switch f.rate
	c	1	.01	.0	.0	.01	.01	.0	α_5=c.b.link f.rate
	d	1	.01	.01	.01	.01	.01	.01	α_6=r.b.link f.rate
	e	1	.01	.1	.1	.01	.01	.01	α_7=sp.b.l.f.rate
	f	1	.01	.01	.01	.1	.1	.1	sx=simplex

Table 3.4.- Reliability and RIF's for SRE and ARCE with c = .99 and Failure Rates given in Table 3.3.

Array R/RIF	Time	a	b	c	d	e	f
SRE	.1	.201897	.97553	.974196	.949305	.792264	.901154
	.2	.040762	.889824	.882997	.832308	.580682	.703244
R	.3	.008230	.74327	.729796	.663918	.3869727	.489128
	.4	.001662	.578714	.561119	.492554	.239752	.315416
	.5	.000335	.428912	.410413	.347755	.141386	.194080
	.1	----	32.62	3.93	15.91	3.86	8.50
	.2	----	8.71	8.20	5.80	2.30	3.44
RIF	.3	----	3.86	3.67	2.98	1.62	2.04
	.4	----	2.37	2.27	1.98	1.32	1.51
	.5	----	1.75	1.70	1.54	1.17	1.27
MTTF	-	-	.510087	.496435	.447789	.280379	.33785
ARCE	.1	.201897	.986691	.986227	.985946	.983185	.985713
	.2	.040762	.977126	.976359	.975787	.966740	.974176
R	.3	.008230	.969483	.968212	.966729	.939159	.959977
	.4	.001662	.961962	.959513	.955845	.894073	.93758
	.5	.000335	.95268	.947997	.940533	.831963	.903652
	.1	----	59.97	57.95	55.75	46.78	47.29
	.2	----	41.94	4.58	38.91	28.50	31.78
RIF	.3	----	32.50	31.20	29.31	16.17	21.72
	.4	----	26.25	24.66	22.27	9.38	14.47
	.5	----	21.13	19.22	16.61	5.93	9.65
MTTF	-	-	2.07274	1.90644	1.74707	1.02882	1.28279

Some reliability results of a 4$\times$4 array with a DR scheme are shown in Table 3.6. Each column corresponds to a set of failure rates tabulated in Table 3.5. Column *a* considers the reliability of the array when only PE's (SPE's) fail. The remaining columns consider failures in all components in the model; thus, while in column *b* a single failure rate is assumed for all components except PE's column *c* shows the effect of an increased failure rate of *IL's* and column *d* shows the effect of an increased failure rate of multiplexers. The array shows increased sensitivity to failures in multiplexers than to failures in the *IL's* as shown by the reliability and MTTF results shown in columns *c* and *d*. Because the *DR* scheme considers components of greater complexity (such as *IL's* and multiplexers) than those used in SRE and ARCE, suitable comparison between the three schemes is given by column *b* in Table 3.4 and column *a* in Table 3.6 in which only failures of PE's are considered.

Table 3.5.- Failure Rates used for the results shown in Table 3.6

col.	α_{PE}	α_{IOL}	α_{IMX}	α_{OMX}	α_{BL}	α_{IL}	α_{SPE}	α_{SIOL}	α_{SMX}	α_{SBL}
a	1	.0	.0	.0	.0	.0	1	.0	.0	.0
b	1	.01	.01	.01	.01	.01	1	.01	.01	.01
c	1	.01	.01	.01	.01	.1	1	.01	.01	.01
d	1	.01	.1	.1	.01	.01	1	.01	.1	.01

Table 3.6.- Reliability Results for DR with c = 0.99 and Failure Rates given in Table 3.5

time	a	b	c	d
.1	0.971744	0.935879	0.894698	0.795178
.2	0.832624	0.744162	0.680113	0.487772
.3	0.576951	0.461321	0.403064	0.212808
.4	0.326881	0.228668	0.191000	0.070372
.5	0.156860	0.094770	0.075675	0.018868
MTTF	0.346681	0.30104	0.2774	0.207435

Let us consider a simplified model such as the one proposed in [WaF88b] where the reliability of the array is expressed as $R(t) = R_{nr}(t) \times R_r(t)$. The terms R_{nr} and R_r refer to the reliability of non-redundant and redundant hardware respectively. For the SRE case, let the number of PE's in the array be the redundant hardware, then:

$$R_r(t) = \sum_{i=0}^{n-1} c^i \begin{pmatrix} n \\ i \end{pmatrix} (e^{-n\alpha_1 t})^{n-i} (1-e^{-n\alpha_1 t})^i$$

Consider the No. of IOL's, HL's and Switches as the non-redundant hardware, then:

$$R_{nr}(t) = e^{-(2n^2\alpha_2 + (n+n^2)\alpha_3 + n(n+1)\alpha_5)t}$$

Table 3.7 shows the reliability and MTTF results obtained with a set of failure rates as shown in row g in Table 3.3 and with c=.99. The results given by the simplified model show an underestimation of the reliability of a 4X4 array with the SRE scheme as compared to the results obtained by solving the Markov model generated by MODELGEN.

Table 3.7.- Reliability Results of Simplified and Detailed Modeling

time	simplified	Detailed
.1	.915052	.954906
.2	.782915	.848375
.3	.613426	.687296
.4	.448006	.51798
.5	.311454	.371557
MTTF	.41571	.464996

3.6. Summary

In this chapter the application of MSPN's to generate Markov models for the reliability analysis of processor arrays has been shown. Three examples of reconfiguration schemes were thoroughly analyzed using MGRE to generate the models and reliability results. The analysis covers various components of the array and their failure interdependencies. Comparative results show the state space generated in each case. Reliability results derived for SRE and ARCE compare these two schemes and their responses to different types of failures. A possible MSPN representation of the DR scheme has been derived in which dense areas are confined into a block (IL's) to facilitate the analysis. It is assumed that failure statistics for these blocks are available.

The use of MSPN's as a modeling technique implies that the reliability estimation of any fault-tolerant processor array requires a complete understanding of the fault behavior of the array in the presence of any type of faults considered in the model. The influence of faulty components other than PE's in a processor array becomes more important as the ratio of area occupied by PE's decreases. In this case to predict the effect of faults in resources such as switches, buses, links, etc., on the reliability of the array, detailed modeling is justified. However, a complete characterization of the reconfiguration algorithm is clearly needed; i.e., specifications such as resource sharing and type of resources used to establish interconnections, etc.; particularly in cases of overlapping routing paths where conflicts are more likely to occur.

Another advantage of a detailed modeling of an specific array is the possibility of measuring the effect of redundant area increments in the overall reliability of the array. The problem here is to be able to evaluate the effect on the reliability of increased hardware complexity in terms of area requirements. However, a detailed modeling of the array, causes a rapid growth of the state space and traditional solution methods of reliability models require the summation of large number of terms with different signs which increases the effect of round-off errors as the system state space grows. Also, the generation of absolute large numbers increases the possibility of machine overflow. Fortunately, large state spaces can be solved using randomization techniques [GrM84]. On the other hand, approximation and reduction methods can be applied for very large systems to derive lower and upper reliability bounds [SmG86] without having to solve the entire model. In the latter case, suitable criteria should be established to select those operational states whose performance-related measures are of interest.

CHAPTER IV
NUMERICAL RELIABILITY ESTIMATION AND A MODEL REDUCTION TECHNIQUE

4.1. Introduction

A fault-tolerant system can be modeled as a continuous-time discrete state Markov process whose state space corresponds to the possible operational configurations of the system. To derive reliability estimations at a given time, the transient probabilities of the operational states of the Markov process are required. Traditional solution methods require the summation of large number of terms with different signs. Unfortunately, for large state spaces, the likelihood of unreliable results increases as round-off errors are introduced in the solution process. To overcome this problem the randomization technique has been used as an alternate approach to the transient solutions of queueing systems proposed in [Gra77a,Gra77b] and for the reliability calculation of fault tolerant systems [Mil83]. In this chapter, implementation algorithms to estimate both reliability and MTTF using the randomization technique are discussed. Furthermore, by eliminating those states with low mean holding times, a reduced model is obtained via equalization and lumping techniques [SmG86]. Once a reduced model is obtained, the randomization procedure is applied to derive lower and upper reliability bounds. The model reduction process requires an early knowledge of the MTTF of the original model; therefore, an implementation algorithm to obtain an exact evaluation of the MTTF is discussed. To optimize storage requirements and to speed up calculations these algorithms are based on a single vector representation of the transition rate matrix. This evaluation approach is applicable for large state space models which are represented by an upper triangular transition rate matrix. Applications are illustrated through examples of evaluation of models generated by MGRE.

The second section of this chapter introduces a brief background on the randomization algorithm. Implementation details and an algorithm for reliability estimation are presented in the third section. the fourth section discusses a model reduction technique using an exact estimation of the MTTF. Finally some applications are reported.

4.2. The Randomization Algorithm

Let $\{X(t),\ t \geq 0\}$ be a continuous time homogeneous Markov process on a finite state space $S = \{1,2, \cdots ,s\}$. The state probability vector at time t is denoted by $P(t) = [P_1(t),\ P_2(t),\ \cdots\ ,\ P_s(t)]$, where $P_i(t) = P\{X(t)=i\}$, $i \epsilon S$.

The Markov process $X(t)$ can be characterized by a set of differential equations given in matrix form as follows:

$$\dot{P}(t) = P(t)A \qquad (4.1)$$

with a solution:

$$P(t) = P(0)e^{At} \qquad (4.2)$$

where P(0) corresponds to the initial probability vector and A is an $n \times n$ matrix with elements a_{ij} which are the transition rates from state i to state j and $a_{ii} = -\sum_{i \neq j} a_{ij}$

A useful method [Gra77a,Gra77b] to compute the exponential in equation (4.2) called *uniformization* or *randomization* is described next. Denote by Q the transformation of the matrix A as follows:

$$Q = \frac{A}{\Lambda} + I \tag{4.3}$$

where $\Lambda = \max a_{ii,}\ i \epsilon S$.

The resulting matrix Q is a transition probability matrix of a discrete-time Markov chain (i.e. with entries $0 \leq q_{ij} \leq 1$). Substituting $A = -\Lambda I + \Lambda Q$ in equation (4.2), we have:

$$P(t) = P(0) e^{-\Lambda t} e^{\Lambda t Q} \tag{4.4}$$

A series expansion of the exponential $e^{\Lambda t Q}$ yields:

$$\begin{aligned} P(t) &= \sum_{n=0}^{\infty} P(0) Q^n e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \\ &= \sum_{n=0}^{\infty} \Phi(n) e^{-\Lambda t} \frac{(\Lambda t)^n}{n!} \end{aligned} \tag{4.5}$$

where $\Phi(n) = P(0)Q^n$ is a probability vector of a discrete Markov process with a transition matrix Q.

In the probabilistic sense the uniformization algorithm can be interpreted as follows [Mil83]: let $\{Y_{n,}\ n = 0,1,2, \cdots \}$ be a Markov chain on S with transition matrix Q and $\{N(t), t \geq 0\}$ be a Poisson process with rate Λ; assume both processes are independent of each other. Then the process $\{Y_{N(t),} t \geq 0\}$ is a Markov process with a transition matrix Q with an initial probability state $P(0)$ and therefore identical to X(t). Conditioning over the number of occurrences of the Poisson process in $[0,t]$ and using the law of total probability, we have that for a given state i:

$$P_i(t) = P\{X(t)=i\} \tag{4.6}$$

$$= P\{Y_{N(t)} = i, N(t) = n\}$$

$$= \sum_{n=0}^{\infty} P\{Y_{N(t)}=i \mid N(t)=n\}P\{N(t)=n\}$$

$$= \sum_{n=0}^{\infty} P\{Y_n = i\}e^{-\Lambda t}\frac{(\Lambda t)^n}{n!}$$

Defining $\Phi_i(n) = P\{Y_n = i\}$ and $\Phi(n) = [\Phi_1(n), \Phi_2(n), \ldots]$, then equation (4.5) results.

4.3. Implementation Details

The implementation of the uniformization procedure calls for the calculation of $\Phi(n)$ and $P\{N(t)=n\}$. Since Q is a stochastic matrix, converges and $\Phi(n)$ can be calculated recursively by:

$$\Phi(0) = P(0)$$

$$\Phi(n+1) = \Phi(n)Q \tag{4.7}$$

The Poisson probability of exactly n events in an interval of length t can be computed directly. An advantage of the uniformization procedure is that precision errors can be bounded by the user. The infinite series is truncated at some point $n = m$, such that the complementary cumulative value remains below a prescribed bound ϵ; i.e., for every transient state in the system we must have:

$$1 - \sum_{n=0}^{m} \frac{(\Lambda t)^n e^{-\Lambda t}}{n!} \leq \epsilon$$

Let $|\Phi(n)|$ denote the norm of the vector $\Phi(n)$. Since $|\Phi(n)| \leq |\Phi(m)|$ for any $n \geq m$, the following stopping rule can be used:

$$|\Phi(n)|(1 - \sum_{n=0}^{m} \frac{(\Lambda t)^n e^{-\Lambda t}}{n!}) \leq \epsilon \tag{4.8}$$

Other advantage of this algorithm is its numerical stability. Calculations involve only nonnegative numbers.

In [GrM84] an implementation algorithm of (4.7) for sparse matrices is presented. Sparse matrices are transformed to a vector representation to minimize storage requirements and speed up calculations.

The matrix representations of the models generated via an MSPN representation of fault-tolerant processor arrays, exhibit the following features: they are large, sparse and upper triangular. Therefore a vector representation is simplified and an algorithm similar to that of [GrM84] has been implemented to calculate $\Phi(n)$. An estimation of the MTTF is obtained using the following relations:

$$\sigma_i = \frac{1}{\Lambda} \sum_{n=0}^{\infty} \Phi_i(n)$$

$$MTTF = \sum_{i=0}^{s} \sigma_i$$

The summation to derive σ_i is truncated when ϵ is reached.

4.3.1. Reliability Estimation Procedure

An algorithm to derive state probabilities for several time points is outlined below. The following parameters and notation are used: $it =$ initial time; $h =$ length of interval; $ni =$ Number of intervals; $E[j]$ stores the cumulative Poisson probability value $\xi[j]$ at a time t derived in terms of the index j; $P[j]$ stores the state probabilities at a time t which it is derived in terms of the index j;

Procedure STATE_PROBS

```
    Inputs: it, h, ni
    Outputs: Vectors P[j]; j=0,1, ..., ni
Begin
    n=0;
    calculate $\Phi(0)$ and $|\Phi(0)|$
    for j=0,1,2 ... ni, do
        E[j] = 0;
        P[j] = 0;
    end
    ok = 1
    while ok do
        ok = 0
        for j=0,1,2 ... ni, do
            $t = it + h \times j$
            $\xi[j] = e^{-\Lambda t} \frac{(\Lambda t)^n}{n!}$
            $P[j] = P[j] + \Phi(n) \times \xi[j]$
            if $|\Phi(n)| \times (1 - E[j]) < \epsilon$ do
                $E[j] = E[j] + \xi[j]$
                ok = 1
            end
        end for
        if ok do
            n = n+1
            calculate $\Phi(n)$ and $|\Phi(n)|$
        end
    end while
end procedure
```

The norm of vector $P[j]$ gives the reliability estimation at time t. Notice that the vector $\Phi(n)$ which is common for all intervals, is calculated only once. The algorithm stops when all intervals have reached the stopping rule contained in the **if** statement.

4.4. Model Simplification

Given the Markov model, an exact derivation of the MTTF is possible which can give us an 'apriori' knowledge of the stochastic behavior of the Markov chain with respect to different failure rates. This suggests a possible state space reduction based on the contribution of each state to the overall MTTF. The randomization algorithm can then be applied based on a reduced transition rate matrix. Leakage equalization techniques as presented in [SmG86] refer to the procedure by which all transitions to the failure state from all or a set of operational states, are modified to have a single value. The resulting model can be reduced by lumping those states with the same leakage value into a single state with a single transition to the failure state. Thus, equalization of transitions out of those states which exhibit very low mean holding times followed by a lumping procedure, reduces the state space of the model to yield a model whose solution accuracy depends on the number of states eliminated and the transition rates involved. To assess the accuracy of the solution, lower and upper bounds are estimated and MTTF results are compared with those of the original model.

4.4.1. MTTF Exact Derivation and Implementation Procedure

To compute the MTTF, we observe that the Laplace transform of equation (4.1) can be expressed as follows

$$A^T \rho = - \tilde{P}(s)$$

The entries of the vector ρ correspond to the mean holding time of each state. Evaluating at s=0 and using the fact that A is an upper triangular matrix each entry of ρ can be obtained recursively as follows:

$$\rho_j = \frac{1}{a_j} \sum_{i \neq j} \rho_i a_{ij} \qquad (4.9)$$

Hence, each ρ_j is calculated by observing each column of A. The MTTF is then calculated as follows:

$$MTTF = \sum_{j=0}^{s} \rho_j \tag{4.10}$$

Let **u** be a vector whose entries correspond to the non-zero entries of the transition matrix A. Define $T_i=\{s_i, t_i\}$ as a tuple associated with the ith row (state) of A where s_i is an integer that indicates the number of transitions (targets) from state i; t_i is a set of indices corresponding to the targets of state i. An implementation algorithm to calculate the MTTF is outlined below.

```
Procedure MTTF
    Inputs: $T_i = \{s_i, t_i\}$, u
    Outputs: mttf
Begin
    $k = 0$
    for each row $i$, do
        $\rho[i] = 0$
        $\hat{\rho}[i] = \frac{1}{\lceil u[k] \rceil}$
        $k = k + s_i + 1$
    end
    $\rho[0] = \hat{\rho}[0]$
    $k = 0$
    for each row $i$, do
        for(j=0; j<$s_i$; j++) do
            $\rho[t_i[j]] = \rho[t_i[j]] + \hat{\rho}[t_i[j]] \times \rho[i] \times u[k+j+1]$
        end
        $k = k + s_i + 1$
    end
```

```
end procedure
```

The complexity of this algorithm is $O(n^2)$ where n corresponds to the number of operational states of the Markov model represented by A. The first loop initializes the vectors ρ and $\hat{\rho}$. This initialization is needed to evaluate each element of ρ in the main loop as the rows of A are being fetched from the buffer **u**. An advantage of this implementation results in the case of very large models in which on-line memory is restricted. In this case, the rows of A and all T_i's can be fetched directly from a file.

4.4.2. Model Reduction Process

A state i is a *highly probable state* if $\rho[i]>\varsigma$, where ς is a user-given constant. The model reduction process consists of lumping consecutive states that are non-highly probable. The remaining states are systematically reindexed defining a new reduced model. This method is well suited for large models that include failures of spare components or other components with low failure rates. The failure of a single spare component gives place to a new operational state in the Markov model; large blocks of states can be generated with low transition rates due to the failure of spare components; however other transitions to some of these states may or may not turn them into a highly probable states; thus, eliminating states based solely on the transition rates does not guarantee satisfactory results. By this approach, the elimination of non-highly probable states is controlled by the user via the selection of ς, and the desired reliability bounds.

4.4.3. Estimation of Reliability Bounds

It has been shown in [SmG86], that equalizing transitions to the failure state (*leakage*) from states in the entire model or a subset, results in a new model with conservative or optimistic reliability estimations depending whether the equalization is performed with the maximum or the minimum transition to the failure state.

Let L be a subset of non-highly probable states; notice that there may be several subsets or blocks of consecutive states with a $\rho[i] \leq \varsigma$ $(i \epsilon L)$. For a lower bound a state k is selected such that its leakage corresponds to the maximum leakage:

$$l_{\max} = \max_{i \epsilon L} \left| \sum_j a_{ij} \right|$$

Likewise for an upper bound, a state $k \epsilon L$ is selected such that its leakage corresponds to the minimum leakage:

$$l_{\min} = \min_{i \epsilon L} \left| \sum_j a_{ij} \right|$$

The resulting model has a set L of states with the same leakage. All other transitions to states out of L are eliminated. Thus, the set L can be lumped into a single state $k \epsilon L$ with no change in the reliability estimation. The process is repeated for each subset L that can be formed in the original model. The resulting model is a reduced model with a reliability corresponding to a lower/upper bound with respect to the original model.

4.5. Applications

In this section the reliability plots for 4×4 and 10×10 ARCE arrays are shown to illustrate the lower and upper reliability bounds that result upon the selection of the constant ς. Figures 4.1 and 4.2 show the case of a 4×4 array; Figure 4.1 corresponds to the case of ς=.01 where the lower bound reliability curve is shown with a $MTTF$=.911763 and the upper bound reliability curve with a $MTTF$=2.95693; the exact reliability is also shown with a $MTTF$=1.28279. Figure 4.2 shows the case of ς=.005; in this case the lower bound and upper bound reliability curves are closer to the exact reliability curve with $MTTF$=1.1854 for the lower bound and $MTTF$=1.72181 for the upper bound. While the original model contains 107 states, reduced models with ς=.01 and ς=.005 are obtained with 19 and 21 states respectively.

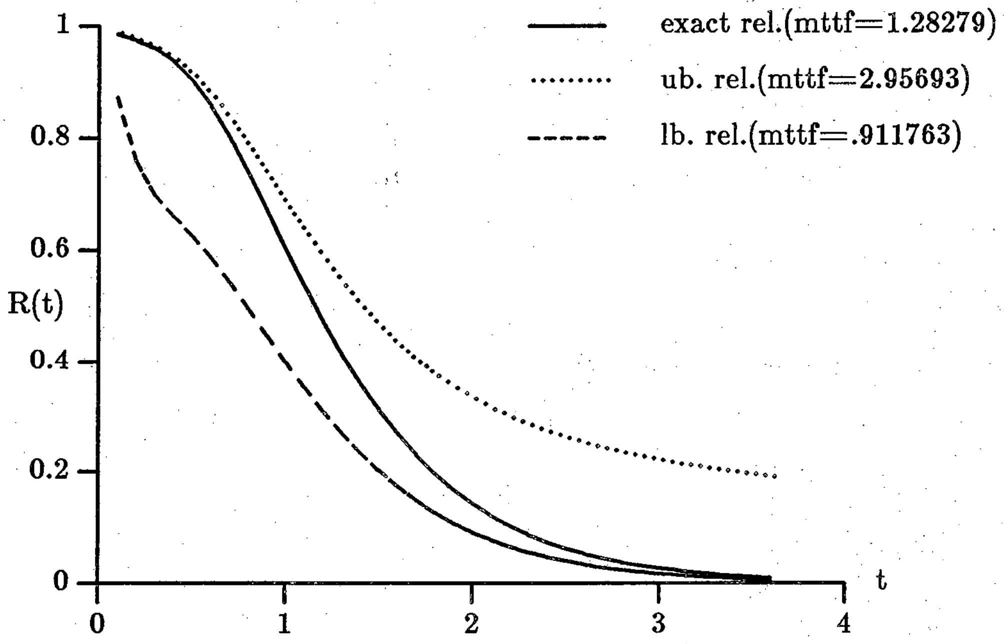


Fig. 4.1. ARCE 4 × 4 Exact Reliability (107 states) and Lower and Upper Bounds (19 states) with ς = .01

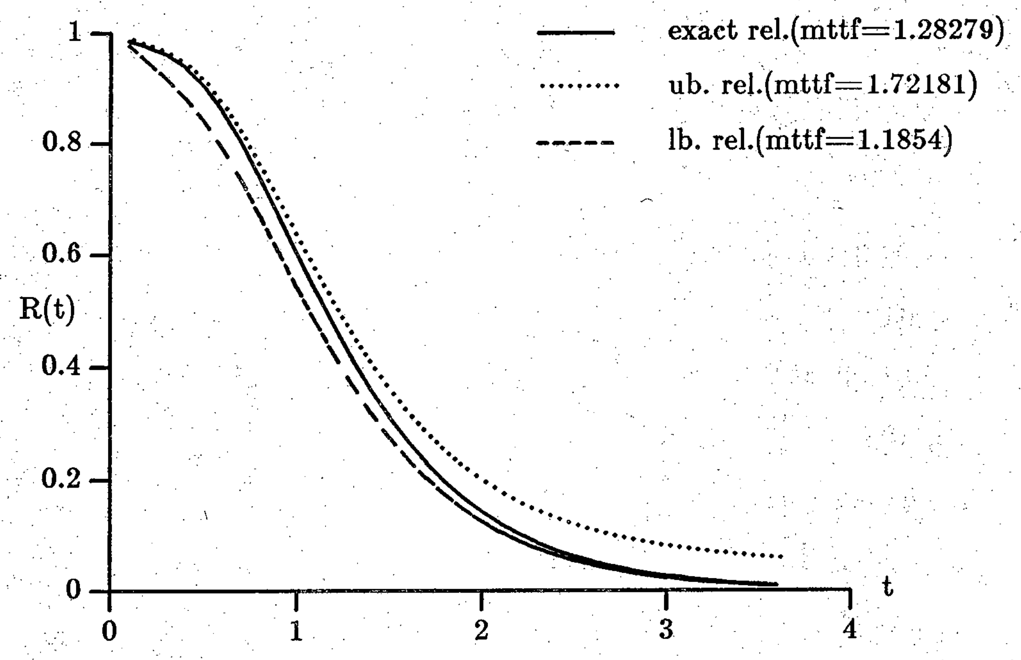


Figure 4.2. ARCE 4 × 4 Exact Reliability (107 states) and Lower and Upper Bounds (21 states) with ς=.005

A 10×10 ARCE array will be represented by a more complex model with 1449 states with a *MTTF*=1.53484. The plots in Figures 4.3-4.5 show the reliability approximations obtained upon the selection of three different constants ς. Figure 4.3 shows the results obtained with ς=.001 with a *MTTF*=1.23601 for the lower bound curve compared to a *MTTF*=1.67142 for the upper bound curve; these results correspond to a reduced model with 44 states. Figure 4.4 shows the results obtained with ς=.0005 with a *MTTF*=1.28975 for the lower bound curve compared to a *MTTF*=1.55848 for the upper bound curve; the reduced model in this case contains 47 states. In Figure 4.5 ς=.0001 with a *MTTF*=1.52796 for the lower bound curve compared to a *MTTF*=1.54118 for the upper bound curve; these results correspond to a reduced model with 47 states; As the lower and upper MTTF values come closer to the exact MTTF, the reliability curves coincide giving an indication of the real reliability values without having to solve the entire model.

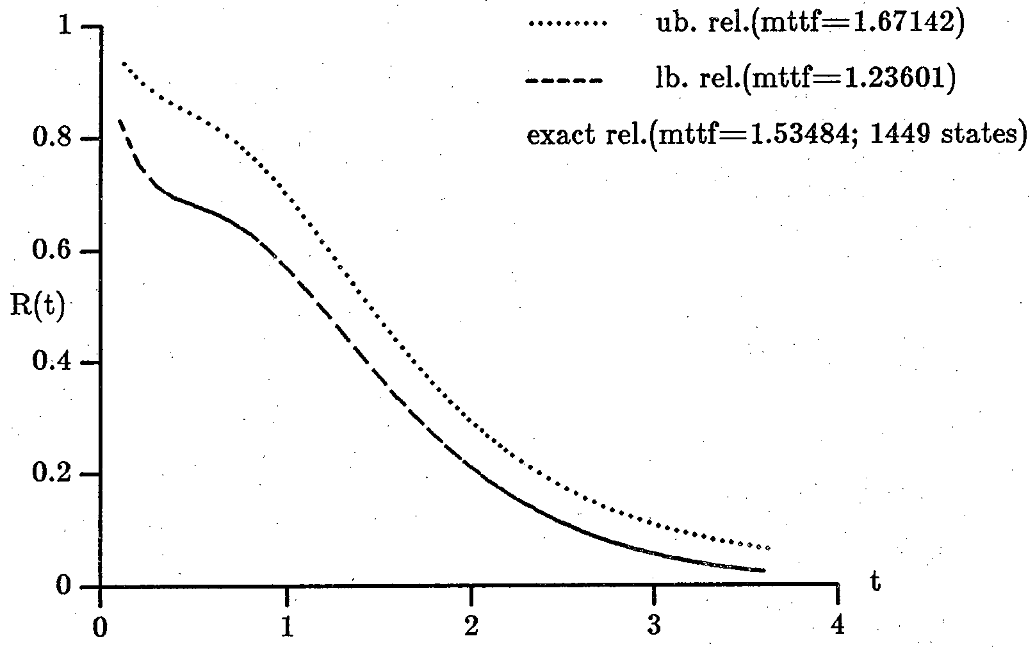


Fig. 4.3. ARCE 10×10 Lower and Upper Bounds (44 states) with $\varsigma = .001$

4.6. Summary

In summary, the general randomization procedure can be used to estimate the reliability of very large models. Using an early estimation of the MTTF of the entire model, fast reliability results can be obtained via the solution of a reduced model. By comparing the mean holding time of each state in the original model, every subset of consecutive states is lumped into a single state. The set of states created by this process is then aggregated with the remaining states to form a new reduced model. A systematic reindexing is carried out and the randomization procedure is applied to obtain lower and upper reliability bounds.

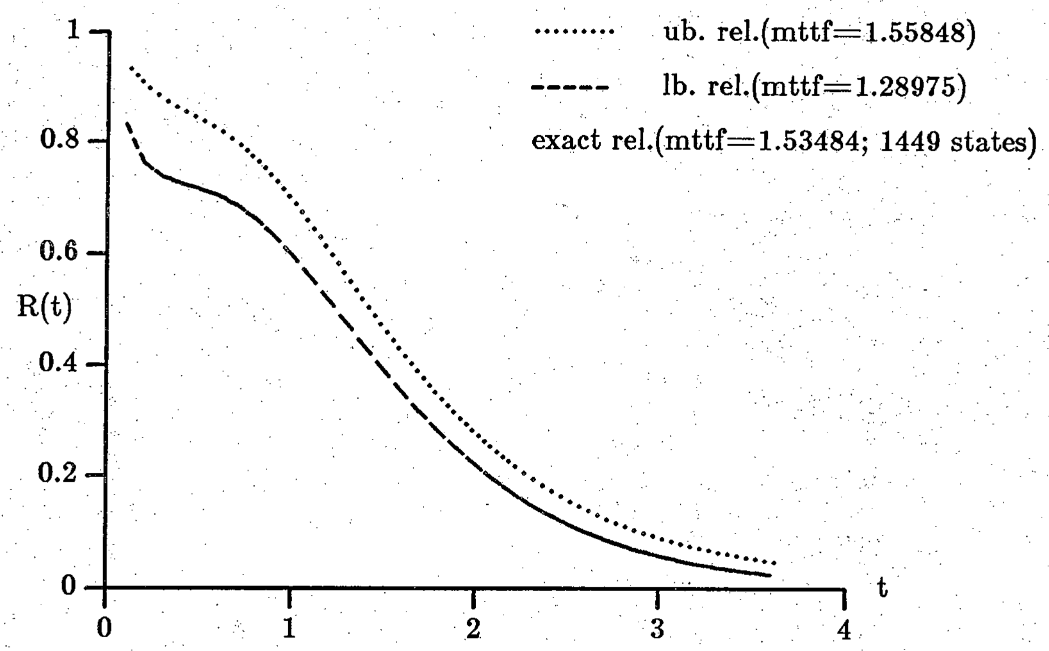


Fig. 4.4. ARCE 10×10 Lower and Upper Bounds (47 states) with $\varsigma = .0005$

Thus, the model reduction process is straightforward and transparent to the user. The solution of the new model corresponds to a lower bound if the state replacement is based on the maximum leakage to the failure state; a minimum leakage replacement yields an upper bound solution. To illustrate the applicability of this method, solutions of small and large models were shown.

One contribution of the work outlined in this chapter consists of the implementation of an algorithm to evaluate reliability models. An additional contribution consists of the possibility of obtaining approximate reliability estimations of large processor arrays; i.e., using reduced models, lower and upper bounds can be obtained. However, a drawback in the reduction process is the fact that a matrix representation is required for an early estimation of the MTTF. Thus the need to reduce storage requirements through a vector-like representation is justified. However it seems reasonable to further explore (in future research) features such as upper triangularity and sparseness of the transition matrix to

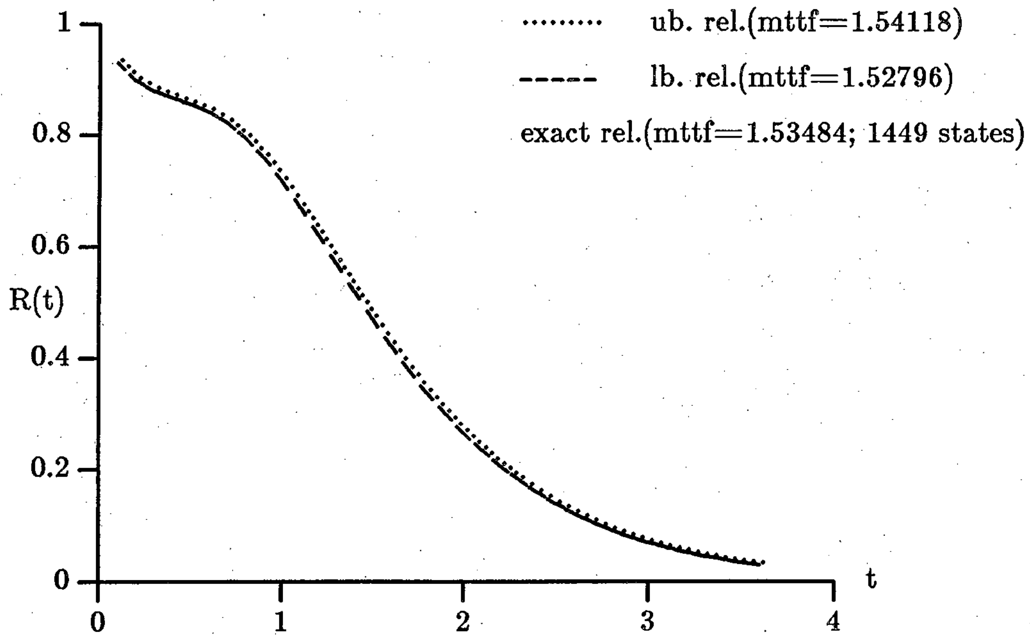


Fig. 4.5. ARCE 10×10 Lower and Upper Bounds (57 states) with $\varsigma = .0001$

calculate the MTTF and select the desired states during the construction of the reachability graph without the need to store the entire model. At this point the only advantage is reaching a solution in shorter time.

CHAPTER V
HIERARCHICAL ARRAYS

5.1. Introduction

In this chapter, 2-level hierarchical structures are analyzed. The components of the hierarchy are fault-tolerant processor arrays that support any of the following reconfiguration algorithms: DR, RR and RCR. DR refers to the direct reconfiguration scheme; the RR (Row Replacement) scheme is a variation of the SRE scheme such that the size of the array remains constant throughout its operational life and can tolerate only a number of faulty processing elements corresponding to the number of spare rows. The RCR (Row or Column Replacement) scheme is a variation of the ARCE scheme such that the size of the array remains constant throughout its operational life and it has a given number of spare rows and spare columns; the number of faults tolerated corresponds to the number of spare rows and columns. The choice of non-degradable processor arrays allows the analysis of several hierarchical schemes by using a given reconfiguration scheme in any place in the hierarchy. Although several measures can be used to compare several schemes, the use of the MTTF permits the comparison of a large number of schemes and helps in determining the effects of detailed modeling in a particular hierarchical scheme; for this purpose, the results obtained are grouped in two tables. One table describes a set of MTTF's obtained when only PE's fail. A second table describes results obtained when faults of several component types are considered. To observe a more detailed behavior in their reliability some hierarchical schemes are selected from the MTTF tables.

5.2. Taxonomy of Hierarchical Schemes

Let $H = <S,K>$ denote a given hierarchical scheme where S refers to a set whose elements identify the reconfiguration algorithms used at different levels in the hierarchy. The subarray size at each level is specified by K as follows:

$$K = (n_1, n_2, \ldots, n_k)$$

i.e., at the ith level the subarray is of size n_i. The subarray sizes satisfy the product:

$$\prod_{i=1}^{k} n_i = n$$

where n corresponds to the size of a single level processor array. Thus, in a given hierarchical structure, a subarray at the ith level is of size n_i and it is formed with subarrays of size n_{i-1}; the subarray at the ith level is implemented using the reconfiguration scheme in the ith position of the set S. For example, if $n=16$, one possible representation of a 3-level structure implemented with only an x reconfiguration scheme is:

$$<(x,x,x),(2,2,4)>$$

which indicates that at each level the array supports the reconfiguration scheme x and that in the first level the size of the subarray is 2, in the second level the size is also 2 and in the third level the array is composed of 4 subarrays.

Note that if there are m different reconfiguration schemes and each can fit at any level in the hierarchical array, then with respect to S there are m^k possible hierarchical configurations. Thus, the selection of an optimal configuration must correspond to an optimal value of a measure of interest such as MTTF, Mission Time, Performability, etc. An analytical determination of an optimal hierarchical configurations is addressed in [WaF88b].

To estimate the reliability and MTTF of a k-level hierarchical array, the following criteria are used:

1) The failure rate of a processing unit (subarray) at the ith level is estimated in terms of the MTTF of the subarray at the (i-1)th level, and denoted as:

$$\alpha_{y,i} = \frac{1}{MTTF_{y,i-1}} \tag{5.1}$$

where y denotes the name of the reconfiguration scheme used in the (i-1)th level.

2) The failure rate of component types other than processing units at the jth level is assumed to be proportional to the failure rate of the corresponding component type at the (i-1)th level such that:

$$\alpha_{i,j} = n_{j-1}\alpha_{i,j-1} \tag{5.2}$$

the subindex i refers to the component type i at the jth and (j-1)th levels; as defined above, the term n_{j-1} corresponds to the size of the subarray at the (j-1)th level. Hence, using this criteria the complexity of a component type is assumed to increase linearly with respect to the same component type in a single level array. Also, the increasing complexity of interconnections as the level of the hierarchy increases is taken into account.

5.3. Fault-Tolerant 2-level Hierarchical Arrays

The use of three different reconfiguration schemes gives a total of $3^2 = 9$ possible choices for a 2-level hierarchical scheme. The schemes used are the DR scheme which was analyzed previously, the RR scheme and the RCR scheme. The

RR and the RCR schemes are a variation of the SRE and ARCE schemes respectively, in the sense that the size of the array remains constant throughout its operational life. Both schemes will tolerate a number of faulty PE's equal to the number of spare rows and columns provided. Since the DR scheme analyzed in chapter 2 considers only one spare row and one spare column, for compatibility two spare rows are considered in the RR scheme and one spare row and one spare column in the RCR scheme.

All results in each table assume a coverage factor $c = .99$; each PE in the first level and in single-level arrays is assumed to fail at a rate $\alpha_{PE} = .01$; each component of every other type is assumed to fail at a rate 100 times less than that of a single PE.

The MTTF's obtained for single-level arrays are shown in Table 5.1, for several array sizes. For each reconfiguration scheme, these results show the effect of the size of the array on the MTTF when only PE's fail and with detailed modeling. High MTTF's of the DR scheme with respect to RR and RCR, reflect the fact that a failure in the DR leads to a single PE substitution; in the RR scheme a single PE failure leads to the substitution of a complete row; in the RCR, a single PE failure causes a row or a column substitution. Notice that the RCR case contains one extra spare processing unit with respect to RR, however, this has the tendency to decrease the MTTF because while in the RR case the failure of any spare unit implies the elimination of n units, in the RCR case however, $n+1$ units are eliminated. Another conclusion that can be drawn from Table 5.1 is the fact that the MTTF of $n \times n$ processor arrays with a fixed structure decreases as the size n of the array increases.

Tables 5.2 and 5.3 show the results for every possible set S; for $n = 64$, each entry displays the MTTF corresponding to a particular set K. Table 5.2

tabulates the MTTF's obtained when only PE's fail and Table 5.3 considers detailed modeling in which other component types in the array can fail. For simplification SBL's in every scheme are assumed fault-free. However SBL's that become active can fail and their complexity increases in the second level according to the size of the subarray in the first level. An immediate advantage of a 2-level hierarchical structure can be seen by comparing the MTTF of a single-level 64 × 64 processor array with any reconfiguration scheme (last row in Table 5.1) with any entry in Tables 5.2 and 5.3; i.e., except for some cases (single level 64×64 DR array), most 2-level choices yield and improvement in its MTTF with respect to the single-level structure. Hierarchical arrays implemented with the DR scheme in at least one level show an improvement in their MTTF with respect to those arrays implemented with RR and/or RCR. Examine, for example, the row corresponding to RR implemented with DR; it is interesting to note the MTTF improvement with the size of the processing units implemented with DR. Two factors influence this effect: 1) the loss in MTTF due to the size increments of the subarray (with DR) are compensated with the gain in MTTF with a lower dimension of the array in the second level implemented with RR; 2) the probability of survival of arrays with DR increases with the size of the array [SaS86a]. This factors also explain the case of (dr,dr) which shows the best MTTF of all the possibilities.

To observe the reliability behavior of hierarchical arrays with DR the following cases are plotted in the figures indicated.

In Figure 5.1 the reliabilities of the schemes ((dr,rr), (4,16)), ((dr,rr), (8,8)) and ((dr,rr), (16,4)) are compared. In this set, the second level is implemented with RR with subarrays implemented with DR. The best structure on this set consists on 4×4 arrays with RCR with 16×16 DR subarrays.

Figure 5.2 plots the reliabilities of the set: ((dr,rcr), (4,16)), ((dr,rcr), (8,8)) and ((dr,rcr), (16,4)).

Figure 5.3 plots: ((rr,dr), (4,16)), ((rr,dr), (8,8)) and ((rr,dr), (16,4)).

Figure 5.4 plots: ((rcr,dr), (4,16)), ((rcr,dr), (8,8)) and ((rcr,dr), (16,4)).

Figure 5.5 plots the set ((dr,dr), (4,16)), ((dr,dr), (8,8)) and ((dr,dr), (16,4)) which correspond to the most reliable structures for the set of failure rates specified previously.

Table 5.1.- MTTF's for Single-level Fault-tolerant Processor Arrays

n	PE fails only			detailed modeling		
	rr	rcr	dr	rr	rcr	dr
1	182.35	174.126	157.668	166.083	155.786	138.393
2	53.7925	52.4267	92.2377	49.9527	48.0654	81.1081
4	15.292	15.1283	34.6681	14.3604	14.0419	30.7144
8	4.164	4.14886	12.1023	3.93578	3.8755	10.7263
16	1.0951	1.09393	4.24583	1.03883	1.02526	3.72798
32	.28152	.281434	1.49453	.267567	.26422	1.28742
64	.071419	.071413	.525113	.067945	.0671025	.438697

Table 5.2.- MTTF Results for Several 2-level Hierarchical Arrays with n=64 where only Processing Units fail.

Reconf. Schemes	(1,64)	(2,32)	(4,16)	(8,8)	(16,4)	(32,2)	(64,1)
(rr,rr)	.13022	.15144	.16746	.17339	.16746	.15144	.13022
(rcr,rr)	.12436	.14759	.16567	.172758	.167284	.15139	.13021
(dr,rr)	.112604	.259664	.379654	.50394	.649272	.803946	.957464
(rr,dr)	.957468	.803936	.649272	.503938	.379653	.259664	.112604
(rcr,dr)	.914358	.783526	.642312	.502108	.379248	.259588	.112596
(dr,dr)	.827933	1.37851	1.47195	1.46466	1.47195	1.12418	.827932
(rr,rcr)	.130211	.15139	.167284	.172758	.165671	.14759	.12436
(rcr,rcr)	.12435	.147546	.165493	.17213	.165493	.147546	.12436
(dr,rcr)	.112596	.259588	.379246	.502108	.64323	.783532	.914359

Table 5.3.- MTTF Results for Several 2-level Hierarchical Arrays with $n=64$ with Detailed Modeling.

Reconf. Schemes	(1,64)	(2,32)	(4,16)	(8,8)	(16,4)	(32,2)	(64,1)
(rr,rr)	.109331	.13363	.152519	.160952	.157168	.142994	.123372
(rcr,rr)	.103054	.128854	.149238	.158533	.15561	.141214	.121843
(dr,rr)	.09231	.210512	.31536	.425479	.548872	.671337	.778616
(rr,dr)	.640056	.64319	.564834	.457919	.352638	.243947	.106487
(rcr,dr)	.634358	.622186	.553217	.451176	.348128	.240927	.105211
(dr,dr)	.570113	.959717	1.11363	1.16951	1.20048	1.12418	.665665
(rr,rcr)	.10717	.131991	.151273	.159744	.155163	.174985	.117712
(rcr,rcr)	.110113	.127298	.148039	.157351	.153161	.137465	.116255
(dr,rcr)	.0907614	.206441	.310446	.419627	.539103	.650625	.739422

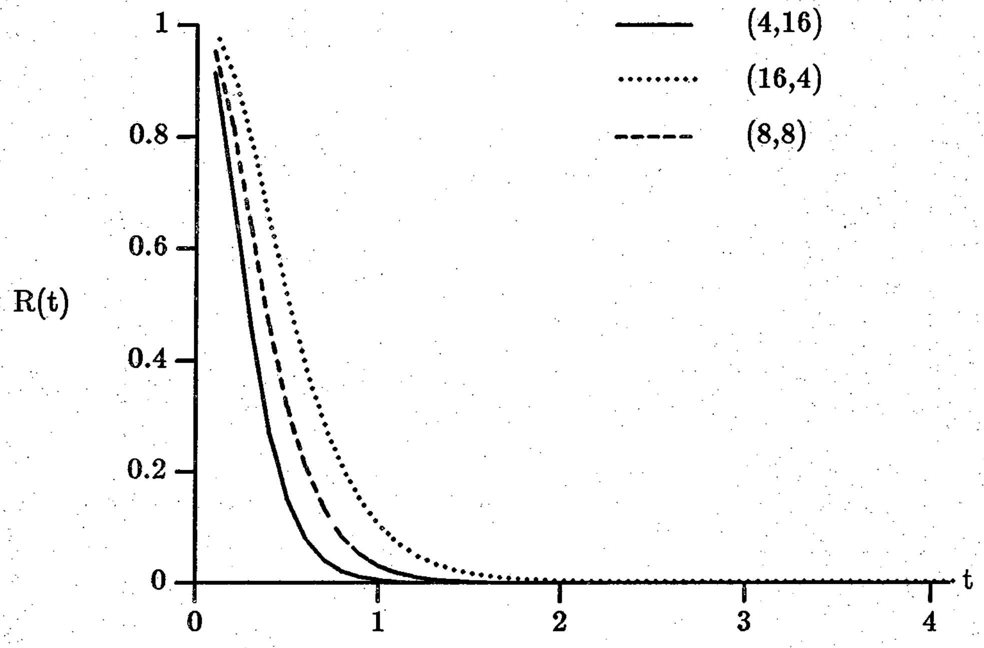


Fig. 5.1.- Reliability plots for (dr,rr) Arrays

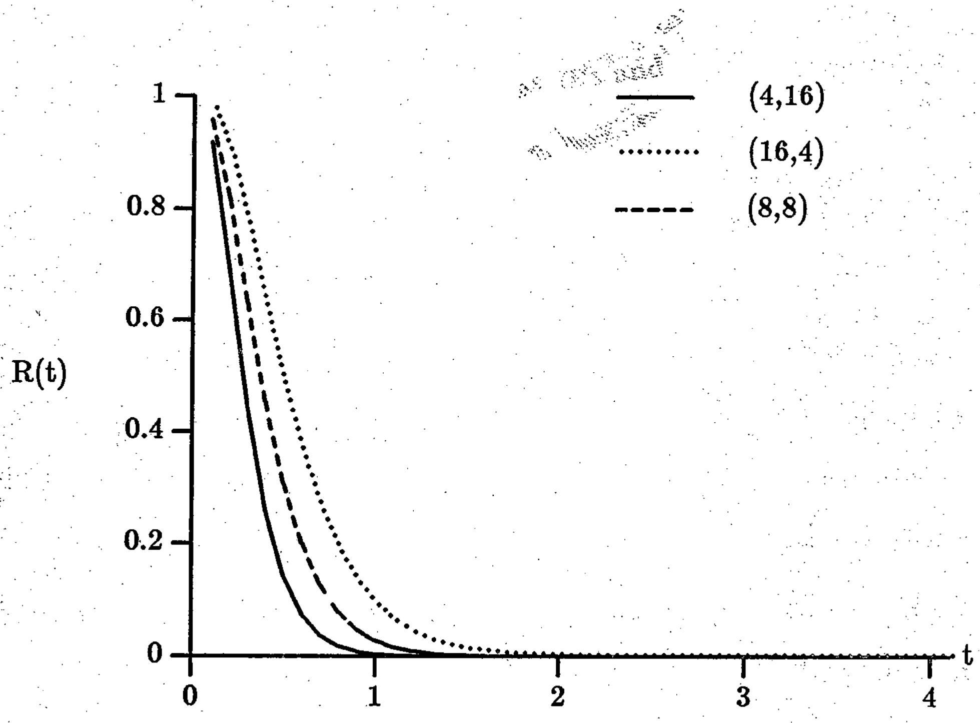


Fig. 5.2.- Reliability plots for (dr,rcr) Arrays

5.4. Summary

In this chapter, possible implementations of hierarchical arrays were analyzed and compared in terms of their MTTF. A brief taxonomy was introduced to relate the reconfiguration algorithms and sizes of the arrays at each level in the hierarchy. For compatibility this analysis was carried out using fixed-structure arrays capable of supporting reconfiguration algorithms such as the DR, RR and RCR. The last two being a variation of the SRE and ARCE reconfiguration algorithms analyzed previously using MSPN's.

Results were derived for cases in which only the processing unit fails and when other component types fail also. The failure rate of a subarray used as a processing unit in the immediately higher level was derived in terms of its MTTF. The failure rates of any other component types were determined in terms of the

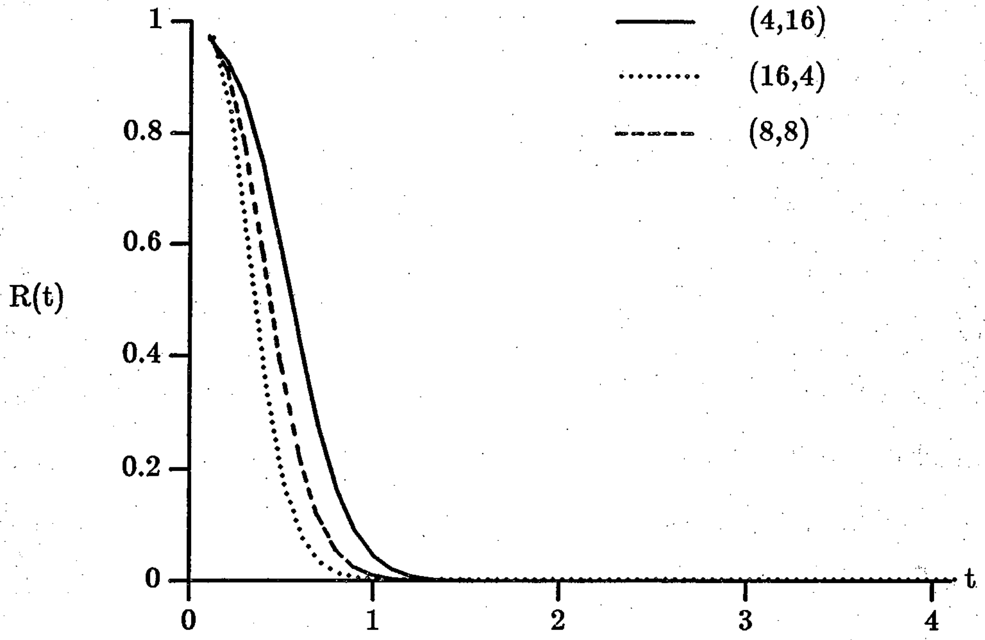


Fig. 5.3.- Reliability plots for (rr,dr) Arrays

size of the processing units and the failure rates used in the immediately lower level. At the first level, the failure rates are chosen by the user. The results shown correspond to a failure rate of a processing element in the first level of .01 failures per time unit; detailed modeling assumes failure rates for any other component type of .0001 failures per unit time. For the cases analyzed, the MTTF decreases as the size of the array increases. However any combination for a hierarchical implementation renders an improvement in its MTTF with respect to a single-level implementation with the same number of PE's. Comparing all the possible hierarchical implementations the ones with DR as a component in the hierarchy render better results; particularly, for the failure rates chosen, the (dr,dr) structures show better MTTF results.

CHAPTER VI
CONCLUSIONS

In this report, a systematic method to construct Markov models to analyze fault-tolerant processor arrays was presented. Since the proposed method rests on the premise that a fault-tolerant processor array can be modeled by a Stochastic Petri Net, a modified version of Stochastic Petri Nets referred to as MSPN's is proposed to model the fault behavior of processor arrays in the presence of failures of components of different types. An MSPN model contains all the information pertaining to the processor array structure and a specific reconfiguration scheme such that the derivation of a detailed Markov model is straightforward. Attributes associated with the transitions in an MSPN model include a probability distribution generated in terms of the number of faulty components in each operational marking. This probability distribution includes probabilities of reconfiguration which together with other attributes, establish a mapping from transitions in the MSPN to transition rates in the corresponding Markov model. Specific reconfiguration algorithms such as SRE, ARCE and DR were analyzed. Reliability results for these three schemes were derived. The analysis covers several components of the array and their failure interdependencies. Analytic expressions to evaluate probabilities of reconfiguration for each scheme were derived. These expressions involve complex combinatorial analysis and for some reconfiguration schemes, simulation has been used to obtain probabilities of reconfiguration. For the DR case, the results obtained via the analytical expressions derived, match those reported in the literature.

A software package, MGRE (Model Generator and Reliability Evaluator) which generates the Markov models and evaluates reliability, has been developed [Lop89]. However, the user is expected to provide files with the probabilities of reconfiguration. Since access to those files may be tailored according to particular applications, the user is also expected to provide subroutines to access the required values. Also, for complex reconfiguration schemes in which sequences of transitions that fire immediately cannot be predetermined for all markings, the user is required to provide subroutines that examine each marking in order to determine the type of reconfiguration or any other parameter used in the selection of a sequence. An extensive use of MGRE for different reconfiguration schemes can lead to a better design of a user interface.

The intrinsic large models generated for even moderate array sizes, cannot be easily solved using existing reliability evaluation packages due to numeric round-off errors introduced during the evaluation process. Also, the large numbers involved in the solution process of large models may cause machine overflow. The solution proposed in this work is based on the general randomization procedure. An algorithm was implemented using this procedure such that the solution of fairly large models is possible. Furthermore, it is shown that given an early estimation of the MTTF of the entire model, fast reliability results can be obtained via the solution of reduced models.

Hierarchical arrays were also discussed as an application of MGRE. Several 2-level structures were compared using as processing units subarrays of different dimensions and with different reconfiguration schemes. For compatibility, this analysis includes processor arrays with a fixed-size structure; i.e., no performance degradation allowed. It was shown that hierarchical structures offer a good potential to increase reliability with respect to single-level arrays with the same number of processing elements.

The main contribution of this research consists of a procedure to analyze fault-tolerant processor arrays using a more general approach and with an optional modeling detail. Given the MSPN of a particular reconfiguration scheme, Markov models of an array of any size can be derived for selected sets of failure rates. Thus, several array architectures can be compared in terms of their reliability and MTTF.

Detailed modeling preserves all the information of the structure of the array in each operational state. Therefore, performability and other performance and area related measures can be derived in terms of each operational state. Furthermore, the approach presented in this thesis can be applied to analyze other hardware systems such as computer networks, interconnection networks, etc.

LIST OF REFERENCES

[Ber81] C. Berge, *Principles of Combinatorics,* Academic Press, 1981.

[BoC69] W.G. Bouricious, W.C. Carter, and P.R. Schneider, "Reliability Modeling Techniques for Self-repairing Computer Systems," *Proc. 24th Nat'l. Conf. of the ACM,* Aug. 1969.

[FoR85] J.A.B. Fortes and C.S. Raghavendra, "Gracefully Degradable Processor Arrays," *IEEE Trans. Comp.,* Vol. C-34, Nov 1985, pp. 1033-1044.

[GeT83] R. M. Geist and K. S. Trivedi, "Ultrahigh Reliability Prediction for Fault Tolerant Computer Systems," *IEEE Trans. Comp.,* Vol. C-32 No. 12, Dec. 1983, pp. 1118-1127.

[Gra77a] W. K. Grassman, "Transient Solutions in Markovian Queueing Systems," *Comput. & Ops. Research,* Vol. 4, 1977, pp. 47-53.

[Gra77b] W. K. Grassman, "Transient Solutions in Markovian Queues ," *European Journal of Operational Research 1,* 1977, pp. 396-402.

[GrM84] D. Gross and D. R. Miller, "The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes," *Opns. Res.,* Vol. 32 No. 2, Mar.-Apr. 1984, pp. 343-361.

[HiE83] E. F. Hitt and D. Eldredge, "A Review and Application of Analytical Models in Fault Tolerant Avionics Validation," *Digital Avionic System Conference IEEE/AIEE,* 1983.

[JaR88] A. Jain and J. Rajski, "Probabilistic Analysis of Yield and Area Utilization of Reconfigurable Rectangular Processor Arrays," *Int'l.*

Workshop on Defect and Fault Tolerance in VLSI Systems, Oct. 1988, to appear.

[KoB84] I. Koren and M.A. Breuer, "On Area and Yield Considerations for Fault-Tolerant VLSI Processor Arrays," *IEEE Trans. Comp.,* Vol. C-33, Jan. 1984, pp. 21-27.

[KoP86] I. Koren and D.K. Pradhan, "Yield and Performance Enhancement Through Redundancy in VLSI and WSI Multiprocessor Systems," *Proc. IEEE,* Vol. 74, May 1986, pp. 699-711.

[KoP87] I. Koren and D.K. Pradhan, "Modeling the Effect of Redundancy on Yield and Performance of VLSI Systems," *IEEE Trans. Comp.,* Vol. C-36, Mar. 1987, pp. 344-355.

[Lop89] N. Lopez-Benitez, *Detailed Modeling and Reliability Estimation of Fault-Tolerant Processor Arrays,* Phd. Dissertation, Dep. of Elec. Eng. Purdue University, 1989.

[MaA82] T.E. Mangir·and A. Avizienis, "Fault-Tolerant Design for VLSI: Effect of Interconnect Requirements on Yield Improvement of VLSI Designs," *IEEE Trans. Comp.,* Vol. C-31, July 1982, pp. 609-616.

[Mil83] D. R. Miller, "Reliability Calculations using Randomization for Markovian Fault-Tolerant Computing Systems," *IEEE Fault Tolerant Computing,* 1983, pp. 284-289.

[Mol82] M.K. Molloy, "Performance Analysis Using Stochastic Petri Nets," *IEEE Trans. Comp.,* Vol. C-39 No. 9, Sept. 1982, pp. 913-917.

[RaA84] C.S. Raghavendra, A. Avizienis, and M. Ercegovac, "Fault-Tolerance in Binary Tree Architectures," *IEEE Trans. Comp.,* Vol. C-33, June 1984, pp. 568-572.

[ShT86] R. A. Sahner and K. S. Trivedi, "A Hierarchical, Combinatorial-Markov Method of Solving Complex Reliability Models," *ACM/IEEE Proc. Fall Joint Comp. Conf.,* Nov. 1986, pp. 817-825.

[SaS86a] M.G. Sami and R. Stefanelli, "Reconfigurable Architectures for VLSI Processing Arrays," *Proc. IEEE,* Vol. 74, May 1986, pp. 712-722.

[SaS86b] M.G. Sami and R. Stefanelli, "Fault-Tolerance and Functional Reconfiguration in VLSI Arrays," *Int'l. Symp. on Circuits and Systems,* 1986, pp. 643-648.

[SmG86] M. Smotherman, R. M. Geist, and K. S. Trivedi, "Provably Conservative Approximations to Complex Reliability Models," *IEEE Trans. Comp.,* Vol. C-35 No.4, Apr. 1986, pp. 333-338.

[Tri82] K.S. Trivedi, *Probability & Statistics with Reliability, Queuing, and Computer Science Applications,* Prentice-Hall, Englewood Cliffs, N.J., 1982.

[Tri85] K.S. Trivedi, *Modeling and Analysis of Fault-tolerant Systems,* Elsevier Science Publishers, North Holland, 1985.

[WaF88] Y.-X. Wang and J.A.B. Fortes, "On the Analysis and Design of Hierarchical Fault-Tolerant Processor Arrays," *Int'l. Workshop on Defect and Fault Tolerance in VLSI Systems,* Oct. 1988.