Fault-Tolerant Computing

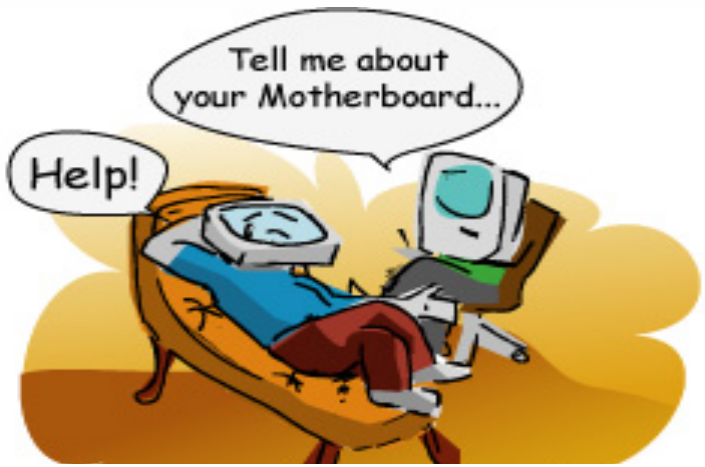
About This Presentation

This presentation has been prepared for the graduate course ECE 257A (Fault-Tolerant Computing) by Behrooz Parhami, Professor of Electrical and Computer Engineering at University of California, Santa Barbara. The material contained herein can be used freely in classroom teaching or any other educational setting. Unauthorized uses are prohibited. © Behrooz Parhami

Edition	Released	Revised	Revised
First	Nov. 2006		

Malfunction Diagnosis and Tolerance

"It's not our fault. We attribute your poor portfolio performance to fund malfunction."



Multilevel Model

Component

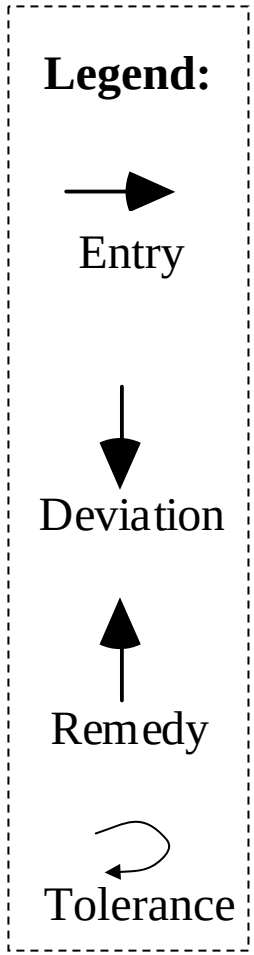
Logic

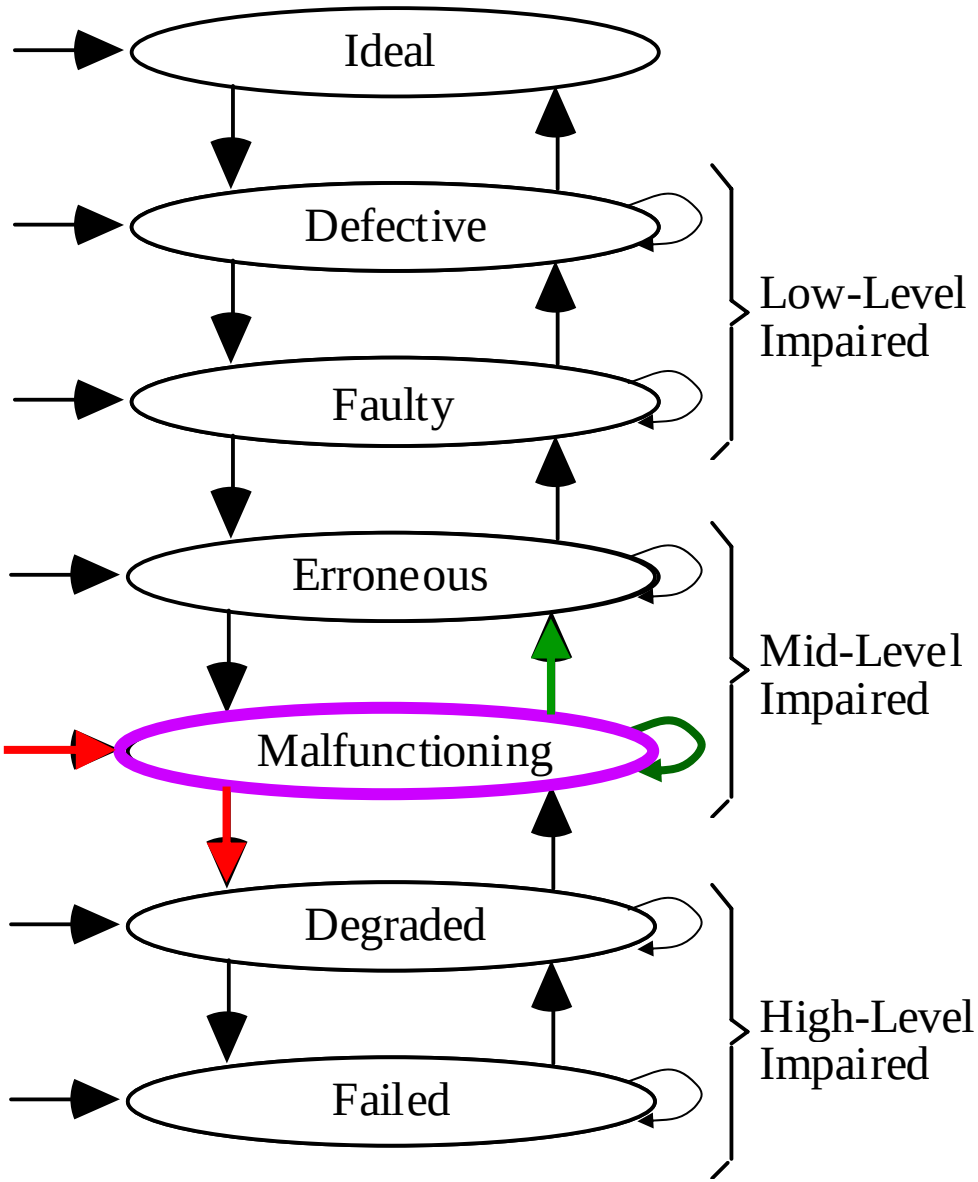
Information

System

Service

Result

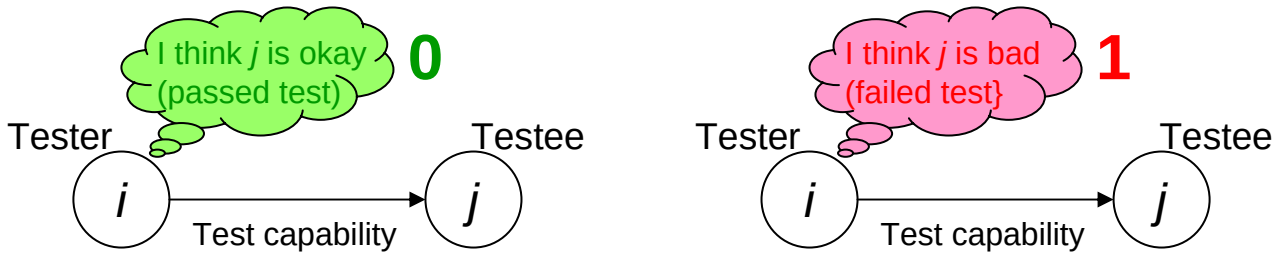




Malfunction Diagnosis Model

Layered approach to self-diagnosis
- A small part of a unit is tested, which then forms a trusted core
- The trusted core is used to test the next layer of subsystems
- Sphere of trust is gradually extended, until it covers the entire unit

Diagnosis of one unit by another
- The tester sends a self-diagnosis request, expecting a response
- The unit under test eventually sends some results to the tester
- The tester interprets the results received and issues a verdict



Testing capabilities among units is represented by a directed graph

The verdict of unit i about unit j is denoted by $D_{ij} \in \{0, 1\}$
All the diagnosis verdicts constitute the $n \times n$ diagnosis matrix D
The diagnosis matrix D is usually quite sparse

More on Terminology and Assumptions

Malfunction diagnosis in our terminology corresponds to "system-level fault diagnosis" in the literature

The qualification "system-level" implies that the diagnosable units are subsystems with significant computational capabilities (as opposed to gates or other low-level components)

We do not use the entries on the main diagonal of the diagnosis matrix *D* (a unit does not judge itself) and we *usually* do not let two units test one another

$$\begin{bmatrix} -- & D_{01} & -- & -- \\ -- & -- & D_{12} & D_{13} \\ D_{20} & -- & -- & -- \\ D_{30} & -- & D_{32} & -- \end{bmatrix}$$

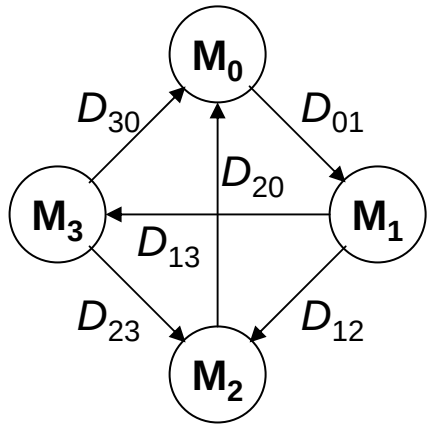
A good unit always issues a correct verdict about another unit (i.e., tests have perfect coverage), but the verdict of a bad unit is arbitrary and cannot be trusted

This is known as the PMC model (Preparata, Metze, & Chien)

We consider the PMC model only, but other models also exist (e.g., in comparison-based models, verdicts are derived from comparing the outputs of unit pairs)

Simple Example with Four Units

Consider this system, with the test outcomes shown



Diagnosis syndromes

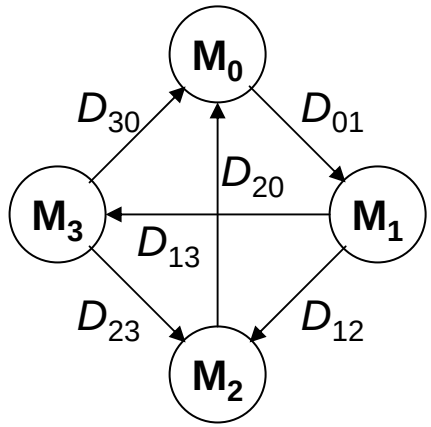
Malfn	D_{01}	D_{12}	D_{13}	D_{20}	D_{30}	D_{32}
None	0	0	0	0	0	0
M_0	0/1	0	0	1	1	0
M_1	1	0/1	0/1	0	0	0
M_2	0	1	0	0/1	0	1
M_3	0	0	1	0	0/1	0/1
M_0,M_1	0/1	0/1	0/1	1	1	0
M_1,M_2	1	0/1	0/1	0/1	0	1

Syndromes for these two conditions may be the same

We say that the system above is 1-step 1-diagnosable (we can correctly diagnose up to 1 malfunctioning unit in a single step)

1-Step *t*-Diagnosability: Requirements

An *n*-unit system is 1-step *t*-diagnosable if the diagnosis syndromes for conditions involving up to *t* malfunctions are all distinct



Necessary conditions:

1. $n \geq 2t + 1$; i.e., a majority of units must be good
2. Each unit must be tested by at least *t* other units

Sufficient condition:

An *n*-unit system in which no two units test one another is 1-step *t*-diagnosable iff each unit is tested by at least *t* other units

So, each unit being tested by *t* other units is both necessary and sufficient

The system above, has each unit tested by 1 or 2 units; it is 1-step 1-diagnosable

It cannot be made 1-step 2-diagnosable via adding more test connections

1-Step Diagnosability: Analysis & Synthesis

Analysis problems:

1. Given a directed graph defining the test links, find the largest value of t for which the system is 1-step t-diagnosable (easy if no two units test one another; fairly difficult, otherwise)
2. Given a directed graph and its associated test outcomes, identify all the malfunctioning units, assuming there are no more than t

There is a vast amount of published work dealing with Problem 1

Problem 2 arises when we want to repair or reconfigure a system using test outcomes (solved via table lookup or analytical methods)

Synthesis problem:

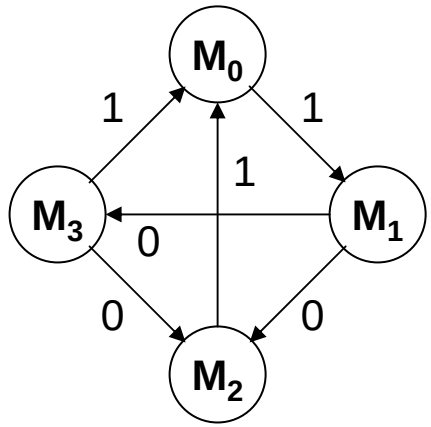
Specify the test links (connection assignment) that makes an n-unit system 1-step t-diagnosable; use as few test links as possible

A degree-t directed chordal ring, in which node i tests the t nodes $i + 1, i + 2, \ldots, i + t$ (all mod n) has the required property

An O(n^3)-Step Diagnosis Algorithm

Input: The diagnosis matrix
Output: Every unit labeled G or B
while some unit remains unlabeled repeat
    choose an unlabeled unit and label it G or B
    use labeled units to label other units
    if the new label leads to a contradiction
    then backtrack
    endif
endwhile

More efficient algorithms exist



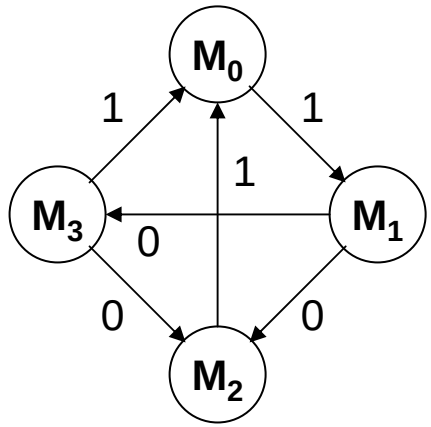
1-step 1-diagnosable system

M_0 is G (arbitrary choice)
M_1 is B
M_2 is B (contradiction, 2 Bs)

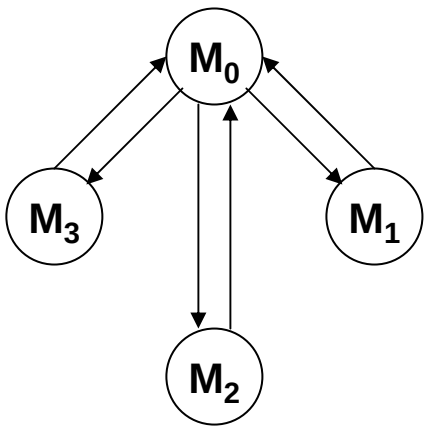
M_0 is B (change label)
M_1 is G
M_2 is G
M_3 is G

An O($n^{2.5}$)-Step Diagnosis Algorithm

From the original testing graph, derive an L-graph
The L-graph has the same nodes
There is a link from node *i* to node *j* in the L-graph
iff node *i* can be assumed to be malfunctioning
when node *j* is known to be good



Testing graph
and test results



Corresponding
L-graph

Definition – *Vertex cover* of a graph:
A subset of vertices that contains at least
one of the two endpoints of each edge

Theorem: The unique minimal vertex cover of the L-graph is the set of *t* or fewer malfunctioning units

Sequential *t*-Diagnosability: Requirements

An *n*-unit system is sequentially *t*-diagnosable if the diagnosis syndromes when there are *t* or fewer malfunctions are such that they always identify, unambiguously, at least one malfunctioning unit

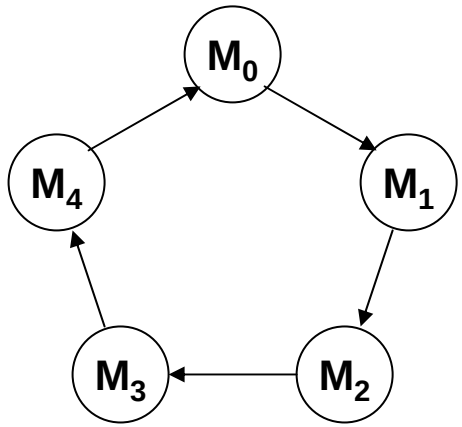
This is useful because some systems that are not 1-step *t*-diagnosable are sequentially *t*-diagnosable, and they can be restored by removing the identified malfunctioning unit(s) and repeating the process

Necessary condition:

$n \geq 2t + 1$; i.e., a majority of units must be good

Sequential diagnosability of directed rings:

An *n*-node directed ring is sequentially *t*-diagnosable for any *t* that satisfies $\lceil (t^2 - 1)/4 \rceil + t + 2 \leq n$



This system is sequentially 2-diagnosable

In one step, it is only 1-diagnosable

Sequential Diagnosability: Analysis & Synthesis

Analysis problems:

1. Given a directed graph defining the test links, find the largest value of t for which the system is sequentially t-diagnosable
2. Given a directed graph and its associated test outcomes, identify at least one malfunctioning unit (preferably more), assuming there are no more than t

These problems have been extensively studied

Synthesis problem:

Specify the test links (connection assignment) that makes an n-unit system 1-step t-diagnosable; use as few test links as possible

An n-node ring, with $n \geq 2t + 1$, with added test links from $2t - 2$ other nodes to node 0 (besides node $n - 1$ which already tests it) has the required property

Other Types of Diagnosability

An n-unit system is 1-step t/s-diagnosable if a set of no more than t malfunctioning units can always be identified to within a set of s units, where $s \geq t$

The special case of 1-step t/t-diagnosability has been widely studied

Given the values of t and s, the problem of deciding whether a system is t/s-diagnosable is co-NP-complete

However, there exist efficient, polynomial-time, algorithms to find the largest integer t such that the system is t/t- or $t/(t + 1)$-diagnosable

An n-unit system is sequentially t/s-diagnosable if from a set of up to t malfunctioning units, $min(s, t)$ can be identified in one step, where $s < t$

Safe diagnosability: Up to t' malfunctions are correctly diagnosed and up to u detected (no danger of incorrect diagnosis for up to u malfunctions; reminiscent of combo error-correcting/detecting codes)

What Comes after Malfunction Diagnosis?

When one or more malfunctioning units have been identified, the system must be reconfigured to allow it to isolate those units and to function without the unavailable resources

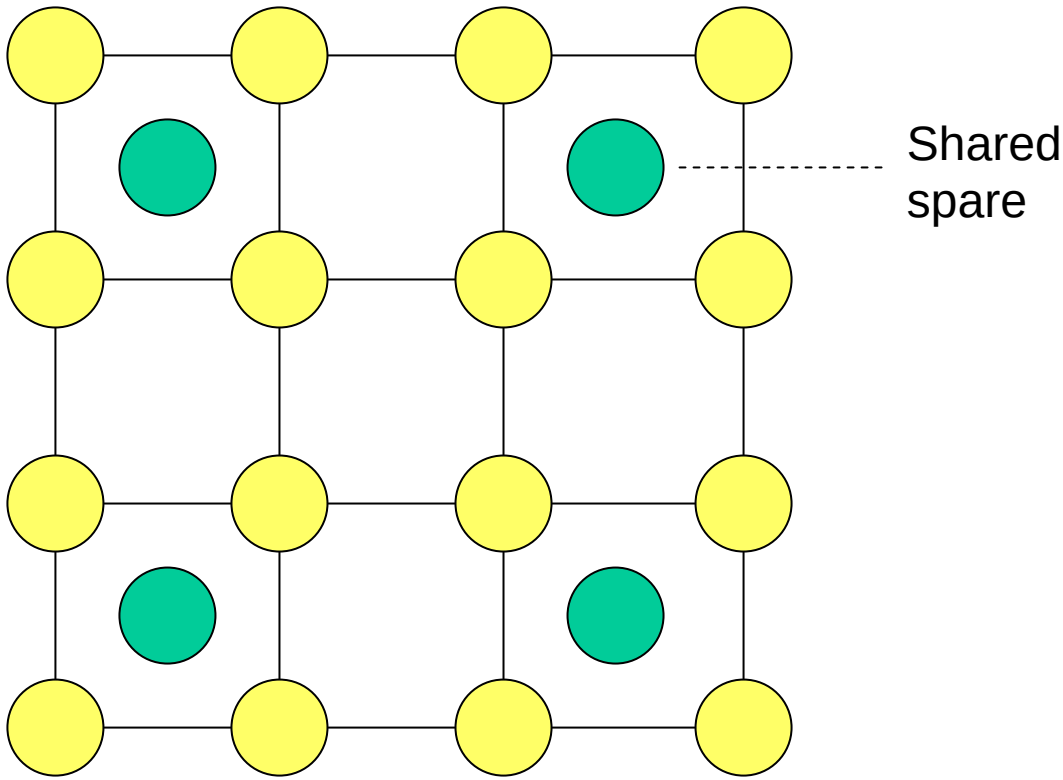
Reconfiguration may involve:

1. Recovering state info from removed modules or back-up storage
2. Reassigning tasks and reallocating data
3. Restarting the computation from last checkpoint or from scratch

In a bus-based system, we isolate malfunctioning units, remove them, and plug in good modules (standby spares or repaired ones)

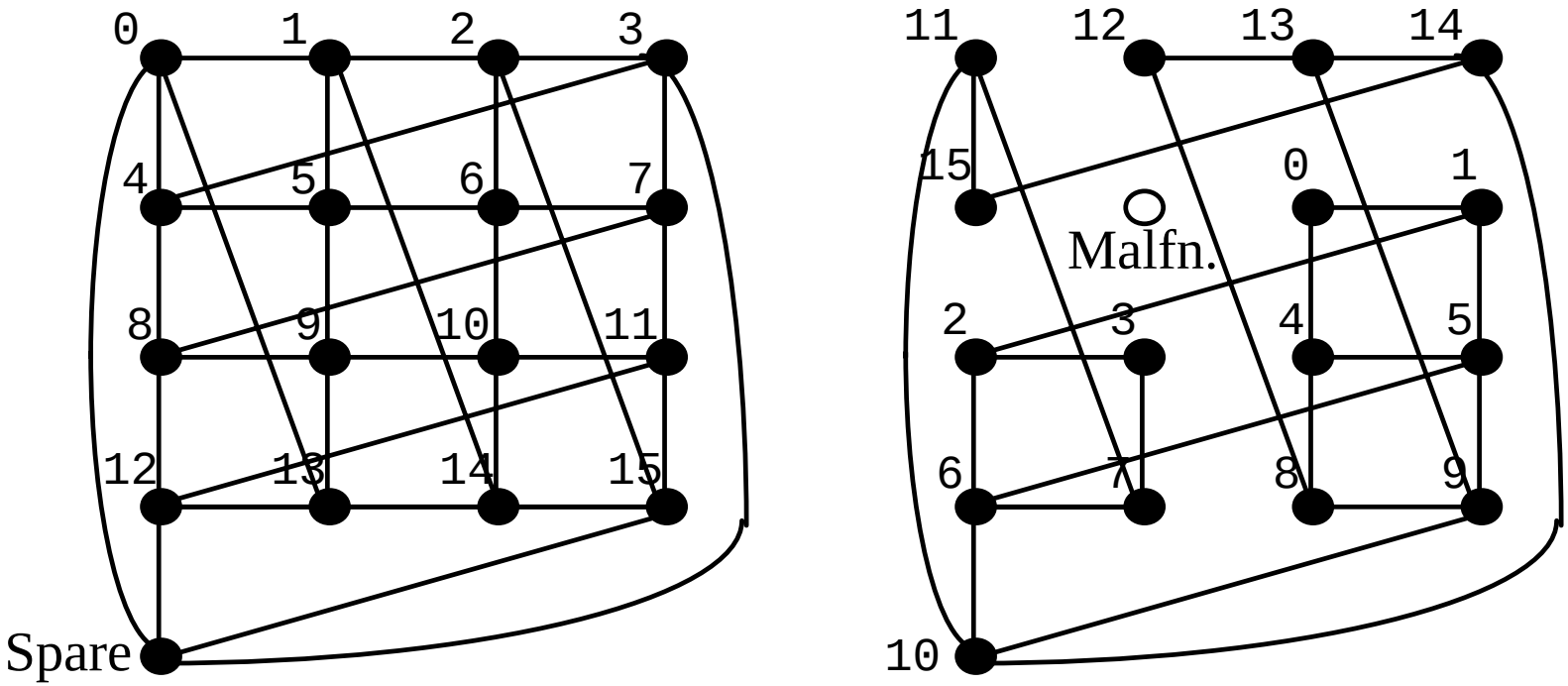
In a system having point-to-point connectivity, we reconfigure by rearranging the connections in order to switch in (shared) spares, using methods similar to those developed for defect circumvention

Reconfiguring Regular Architectures



Malfunction Tolerance with Low Redundancy

The following example scheme uses only one spare processor for a 2D mesh (no increase in node degree), yet it allows system reconfiguration to circumvent any malfunctioning processor, replacing it with the spare via relabeling of the nodes



Reconfigurable 4 × 4 mesh with one spare.