A Parallel Algorithm for Allocation of Spare Cells on Memory Chips

Nobuo Funabiki, Student Member IEEE
Case Western Reserve University, Cleveland
Yoshiyasu Takefuji, Member IEEE
Case Western Reserve University, Cleveland

Key Words — **Artificial neural network, Hysteresis McCulloch-Pitts neuron model, Parallel algorithm, Parallel computing, Spare allocation**

Reader Aids —
Purpose: Present a new algorithm
Special math needed for explanations: Probability
Special math needed to use results: None
Results useful to: Reliability theoreticians and analysts

Summary & Conclusions — **In manufacturing memory chips, Redundant Random Access Memory (RRAM) technology has been widely used because it not only provides repair of faulty cells but also enhances the production yield. RRAM has several rows and columns of spare memory cells which are used to replace the faulty cells. The goal of our algorithm is to find a spare allocation which repairs all the faulty cells in the given faulty-cell map. The parallel algorithm requires $2n$ processing elements for the $n \times n$ faulty-cell map problem. The algorithm is verified by many simulation runs. Under the simulation the algorithm finds one of the near-optimum solutions in a nearly constant time with $O(n)$ processors. The simulation results show the consistency of our algorithm. The algorithm can be easily extended for solving rectangular or other shapes of fault map problems.**

1. INTRODUCTION

With the advancement of VLSI technology, an integrated system interconnecting massive identical elements such as memory cells, CCD cells, and processing elements can be built on a silicon chip or wafer. However, as the density of VLSI systems increases, the probability of defects occurring in them also increases, and the production yield has a tendency to decrease. In manufacturing memory chips, Redundant Random Access Memory (RRAM) technology has been widely used because it not only provides the repairability of faulty cells but also enhances the production yield. RRAM has several rows and columns of spare memory cells. When a faulty cell is detected in a device test, the faulty row or column which contains the faulty cell is disconnected and is replaced by a spare per row or spare per column [1-9]. This redundancy technology can be also used in manufacturing other integrated systems.

A pattern of faulty cells in $n \times n$ matrix cells is described by an $n \times n$ fault map F. In the fault map F, each element f_{ij} represents the condition of the corresponding cell. ($f_{ij}=1$ means that the cell at (i,j) has defects and $f_{ij}=0$ means that the cell has no defect). Figure 1 shows an 8×8 fault map problem where 2 spares per row and 2 spares per column are given. In the fault map, the black squares indicate faulty cells and the white squares indicate normal cells. The fault map contains 7 faulty cells at (1,3), (1,7), (3,5), (4,1), (4,8), (7,5), (8,2). These faulty cells must be repaired by spares per row and/or spares per column.

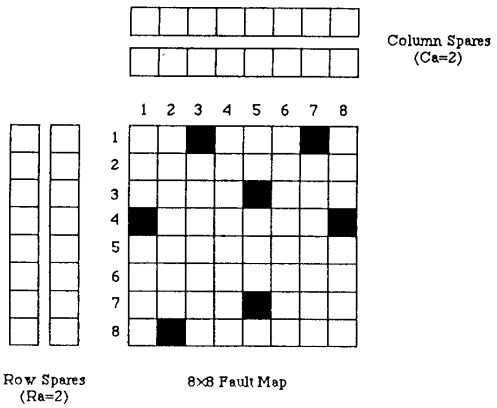


Figure 1. 8×8 Fault Map with Two Row Spares and Two Column Spares

The goal of our algorithm is to find a spare allocation where all the faulty cells in the given fault map are repaired or covered by the given spares per row and spares per column. Several polynomial-time sequential algorithms for this problem have been proposed in RRAM applications [10-15]. In 1984, Tarr, et al proposed two algorithms; the broadside approach and the repair-most approach [10]. The broadside approach scans the memory and replaces each faulty cell with a spare memory per row or a spare memory per column whichever is available where no optimization is performed. The repair-most approach repeatedly repairs the row or column which has the most faulty cells. In 1985, Day proposed the fault-driven approach which repairs the faulty cells according to user-defined preferences [11]. In 1987, Kuo & Fuchs proposed the branch-and-bound approach and the heuristic polynomial-time approximation approach [12]. In 1987, Wey & Lombardi proposed another approach which determines both repairability/unrepairability and a repair solution [13]. In 1990, Huang, et al proposed the faulty-line covering approach and the effective coefficient approach [15]. The problem is proved to be NP-complete [12]. No parallel algorithm has been reported.

More — on the outside.

In this paper we propose a parallel algorithm based on the artificial neural network model for solving the spare-allocation problem. The artificial neural network model uses many simple processing elements which are called neurons because they perform the function of the simplified biological neuron. The sigmoid neural network model for solving combinatorial optimization problems was first introduced by Hopfield & Tank [16]. To enhance the convergence speed, the McCulloch-Pitts neural network [17] has been used for solving several NP-complete or optimization problems [18-25]. However, the McCulloch-Pitts neural network sometimes introduces undesirable oscillatory behavior. It is empirically shown that the hysteresis McCulloch-Pitts neural network suppresses the oscillatory behavior of neural dynamics [24]. Consequently it shortens the convergence time to the global minimum.

The output V_i of processing-element i based on the hysteresis McCulloch-Pitts neuron model is:

$V_i = 1$, if $U_i >$ UTP (Upper Trip Point)

$= 0$, if $U_i <$ LTP (Lower Trip Point)

unchanged, otherwise.

$$\text{UTP} > \text{LTP} \tag{1}$$

Notation

U_i	input of processing-element i
V_i	output (must have an initial value of 0 or 1)
E	computational energy, an n-variable function of $(V_1,\ldots,V_n)$

The change of U_i is the partial derivative of E with respect to V_i. The equation is called a motion equation or a Newton equation. It is:

$$\frac{dU_i}{dt} = -\frac{\partial E(V_1, V_2,\ldots,V_n)}{\partial V_i}. \tag{2}$$

Whatever E is given, the motion equation forces it to monotonically decrease. The proof in the appendix shows that the motion equation forces the state of the system, composed of the hysteresis McCulloch-Pitts neurons, to converge to the local minimum [25].

First we verified our parallel algorithm through solving the Huang problems [15]. Then we created larger problems where the faulty cells are generated randomly and the size of the fault maps is varied from 20×20 to 200×200. The simulation results are shown.

2. SYSTEM REPRESENTATION

Example Problem

Figure 2a shows the system representation for the spare allocation problem of the 8×8 fault map in figure 1. Eight

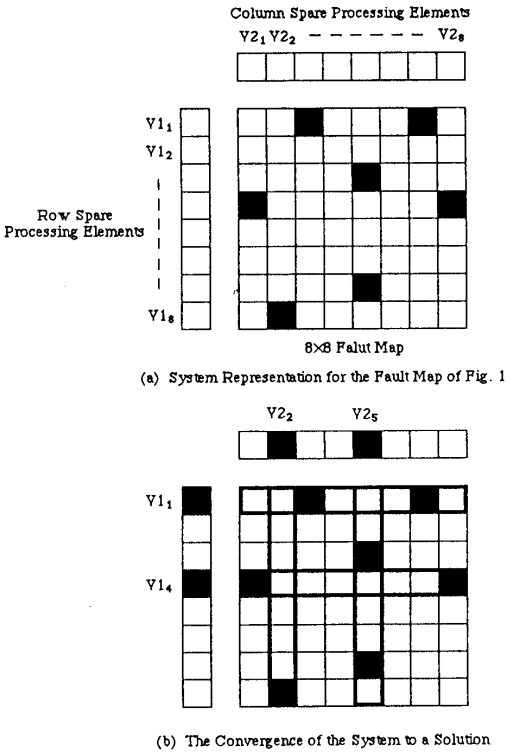


Figure 2. System Representation for the Problem of Figure 1

processing elements are used to describe on which rows (columns) the spares per row (column) should be allocated. A total of 16 processing elements is used. Generally $2n$ processing elements are used to represent n rows and n columns for the $n\times n$ fault map problems. The number of processing elements with the nonzero output among the n row processing elements is Ra (number of given spares per row). The number of processing elements with the nonzero output among the n column processing elements is Ca (number of given spares per column). For example, as shown in figure 2b, the output of 2 processing elements among 8 row processing elements should be nonzero and the output of 2 processing elements among 8 column processing elements should be nonzero, where the black squares indicate the nonzero output of the processing elements and the white squares indicate the zero output of the processing elements. The nonzero output of a processing element means that a spare should be allocated on the corresponding row or column. Figure 2b shows that the spares per row are allocated on rows 1 & 4 and the spares per column are allocated on columns 2 & 5. This spare allocation repairs all the faulty cells.□

Notation

$U1_i, V1_i$	input, output of row i processing-element
$U2_i, V2_i$	input, output of column i processing-element

A,B constant coefficients
$\overline{\Phi}$ $1-\Phi$, where Φ is any symbol

Each faulty cell in the fault map F must be repaired by a spare per row and/or a spare per column. Therefore it is necessary to allocate a spare per row on row i if a faulty cell of row i is not repaired by a spare per column. This condition is:

$$\sum_{k=1}^{n} f_{ik}\,\overline{V2}_k. \tag{3}$$

This condition is nonzero if a faulty cell of row i is not repaired by a spare per column. In other words, it is nonzero if $V2_k = 0$ for the nonzero fault map element f_{ik}.

Similarly it is necessary to allocate a spare per column on column i if a faulty cell of column i is not repaired by a spare per row. This condition is:

$$\sum_{k=1}^{n} f_{ki}\,\overline{V1}_k. \tag{4}$$

This condition is nonzero if a faulty cell of column i is not repaired by a spare per row. In other words, it is nonzero if $V1_k = 0$ for the nonzero fault map element f_{ki}.

The motion equation of row i processing element for the $n\times n$ fault map problem with Ra spares per row is

$$\frac{dU1_i}{dt} = -A\left(\sum_{k=1}^{n} V1_k - \mathrm{Ra}\right) + B\left[\sum_{k=1}^{n} f_{ik}\,\overline{V2}_k\right]\overline{V1}_i \tag{5}$$

The first term in (5) forces the output of Ra row processing elements to be nonzero where the spares per row are allocated. The second term encourages the output of row i processing element to be nonzero if a faulty cell of row i is not repaired by a spare per column and $V1_i = 0$.

Similarly the motion equation of column i processing element for the $n\times n$ fault map problem with Ca spares per column is:

$$\frac{dU2_i}{dt} = -A\left(\sum_{k=1}^{n} V2_k - \mathrm{Ca}\right) + B\left[\sum_{k=1}^{n} f_{ki}\cdot\,\overline{V1}_k\right]\overline{V2}_i \tag{6}$$

The states of the $2n$ processing elements for the $n\times n$ fault map problem can be updated synchronously or asynchronously using (5) & (6). In the synchronous system, the states of all processing elements are updated simultaneously. In the asynchronous system, the states of all processing elements are asynchronously updated. In this paper, the synchronous parallel system is simulated on a sequential machine where the synchronous parallel system can be performed on maximally $2n$ processors. The following procedure/program is used:

```
Program parallel-simulator-on-a-sequential-machine
...
initialization of Ux_i and Vx_i for x:=1 to 2 and for i:=1 to n;
...
/*  Main Program  */
while (a set of conflicts is not empty) do
begin
...
/*  Updating the values of all the inputs  */
    for i:=1 to n
    begin
        U1_i:=U1_i+ΔU1_i;
        U2_i:=U2_i+ΔU2_i;
    end;
/*  End of the first loop  */
...
/*  Updating the values of all the outputs  */
    for i:=1 to n
    begin
        If U1_i>UTP then V1_i:=1 else if U1_i<LTP then
V1_i:=0;
        If U2_i>UTP then V2_i:=1 else if U2_i<LTP then
V2_i:=0;
    end;
/*  End of the second loop  */
...
end;
/*  Main Program end  */
```

It is quite simple to simulate such a synchronous parallel model on a sequential machine. After sequentially updating the values of all the inputs U in the first loop, sequentially updating each one of the outputs V in the second loop is equivalent to simultaneously updating the values of all the outputs — because each output is updated while the states of the inputs are all fixed.

The asynchronous parallel system simulator on a sequential machine is under investigation.

3. PARALLEL ALGORITHM

The following procedure describes our parallel algorithm based on the first order Euler method for the spare-allocation problem where an $n\times n$ fault map, Ra spares per row, and Ca spares per column are given. The data set of A & B, UTP, LTP, $U_$max, $U_$min are empirically determined. $U_$max and $U_$min are the constant upper limit and the constant lower limit of $Ux_i(t+1)$ respectively.

Algorithm

0. Set $t=0$, $A=B=1$, UTP$=5$, LTP$=-5$, $U_$max$=40$, $U_$min$=-160$.

1. The initial values of $Ux_i(t)$ for $x=1,2$ and $i=1,\ldots,n$ are uniformly randomized between 0 and U_min. The initial values of $Vx_i(t)$ for $x=1,2$ and $i=1,\ldots,n$ are zero.

Goto step 3.

2. Evaluate $Vx_i(t)$ for $x=1,2$ and $i=1,..,n$.

$$Vx_i(t) = 1, \text{ if } Ux_i(t) > \text{UTP}$$

$$= 0, \text{ if } Ux_i(t) < \text{LTP}$$

$$\text{unchanged, otherwise} \tag{7}$$

3. Use the motion equations in (5) & (6) to compute $\Delta U1_i(t)$ and $\Delta U2_i(t)$ for $i=1,\ldots,n$.

$$\Delta U1_i(t) = -A\left(\sum_{k=1}^{n} V1_k(t) - \text{Ra}\right) + B\left[\sum_{k=1}^{n} f_{ik}\,\overline{V2}_k(t)\right]\overline{V1}_i(t) \tag{8}$$

$$\Delta U2_i(t) = -A\left(\sum_{k=1}^{n} V2_k(t) - \text{Ca}\right) + B\left[\sum_{k=1}^{n} f_{ki}\,\overline{V1}_k(t)\right]\overline{V2}_i(t) \tag{9}$$

4. Compute $U1_i(t+1)$ and $U2_i(t+1)$ for $i=1,\ldots,n$ based on the first order Euler method.

$$U1_i(t+1) = U1_i(t) + \Delta U1_i(t) \tag{10}$$

$$U2_i(t+1) = U2_i(t) + \Delta U2_i(t) \tag{11}$$

5. If $Ux_i(t+1) > U\_max$ then $Ux_i(t+1) = U\_max$ for $x=1,2$ and $i=1,\ldots,n$. (12)

If $Ux_i(t+1) < U\_min$ then $Ux_i(t+1) = U\_min$ for $x=1,2$ and $i=1,\ldots,n$. (13)

6. If $\Delta Ux_i(t)=0$ for $x=1,2$ and $i=1,\ldots n$ then terminate this procedure, else increment t by 1 and goto step 2.

The range limitation of the input $Ux_i(t+1)$ in step 5 improves the convergence frequency to the global minimum.

The simulator based on our algorithm has been developed on a Macintosh SE/30 in order to verify the algorithm. First we applied the algorithm for solving the Huang problems [15]. Then we created larger problems where the faulty cells are generated randomly and the size of the fault maps is varied from 20×20 to 200×200. Figures 3-19 show the simulation conditions for these problems and the optimum solutions where all the faulty cells are repaired by spares respectively. The algorithm found several optimum solutions from various initial

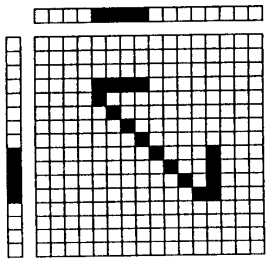


Figure 3. A Solution of the 16 × 16 Fault Map Problem [15] (Ra = Ca = 4)

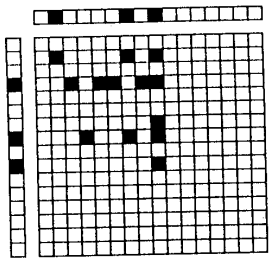


Figure 4. A Solution of the 16 × 16 Fault Map Problem [15] (Ra = Ca = 3)

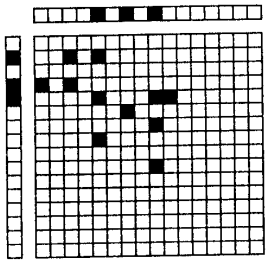


Figure 5. A Solution of the 16 × 16 Fault Map Problem [15] (Ra = Ca = 3)

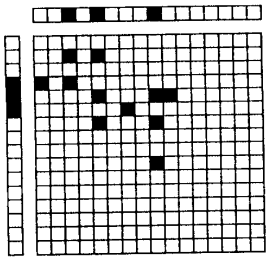


Figure 6. A solution of the 16 × 16 Fault Map Problem [15] (Ra = Ca = 3)

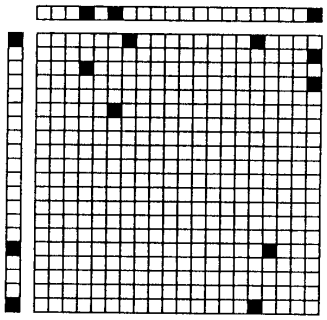


Figure 7. A Solution of the 20 × 20 Fault Map Problem (Ra = Ca = 3)

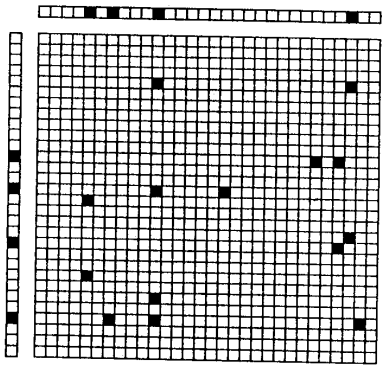


Figure 8. A Solution of the 30 × 30 Fault Map Problem (Ra = Ca = 4)

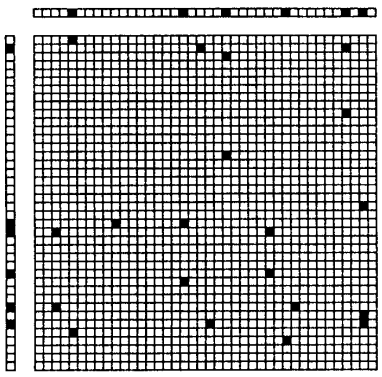


Figure 9. A Solution of the 40 × 40 Fault Map Problem (Ra = Ca = 6)

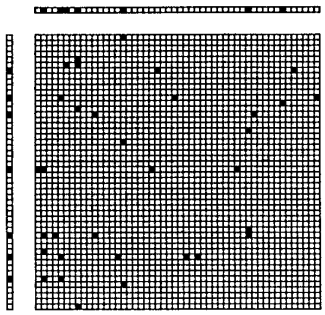


Figure 10. A Solution of the 50 × 50 Fault Map Problem (Ra = Ca = 7)

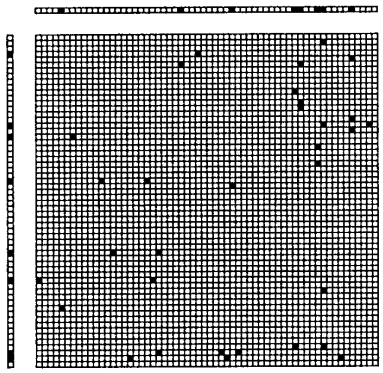


Figure 11. A Solution of the 60 × 60 Fault Map Problem (Ra = Ca = 8)

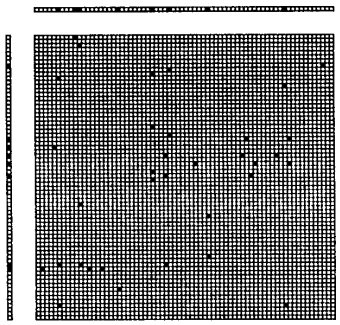


Figure 12. A Solution of the 70 × 70 Fault Map Problem (Ra = Ca = 8)

values of $Ux_i(t)$. Figure 20 shows the simulation results where the number of spares per row and spares per column are varied for the same fault map as that of figure 4. The simulator discovered that 3 spares per row and 1 spare per column are minimum to repair all the faulty cells in this fault map. Table 1 shows the average number of the iteration steps and the frequency of the convergence to the optimum solution where 100 simulation runs were performed for each one of 17 problems. For each one of simulation runs, the different initial values were given. Figure 21 shows the relationship between the frequency and the number of iteration steps to converge to the optimum solution in two of 17 problems. The simulation results show

More — on the outside.

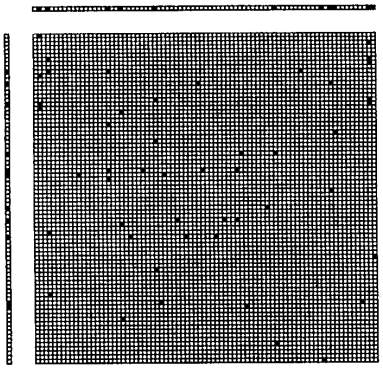


Figure 13. A Solution of the 80 × 80 Fault Map Problem (Ra = Ca = 11)

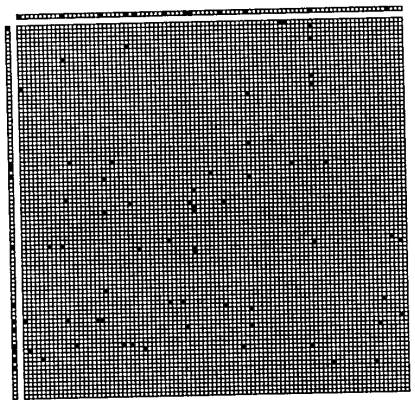


Figure 14. A Solution of the 90 × 90 Fault Map Problem (Ra = Ca = 12)

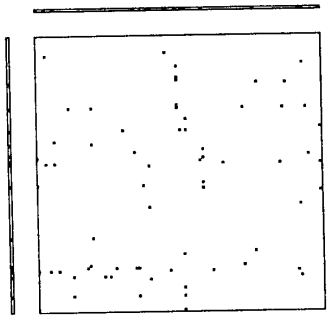


Figure 15. A Solution of the 100 × 100 Fault Map Problem (Ra = Ca = 14)

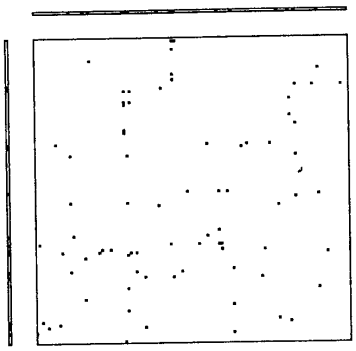


Figure 15. A Solution of the 110 × 110 Fault Map Problem (Ra = Ca = 15)

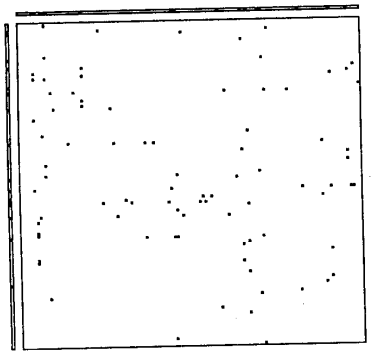


Figure 17. A Solution of the 120 × 120 Fault Map Problem (Ra = Ca = 16)

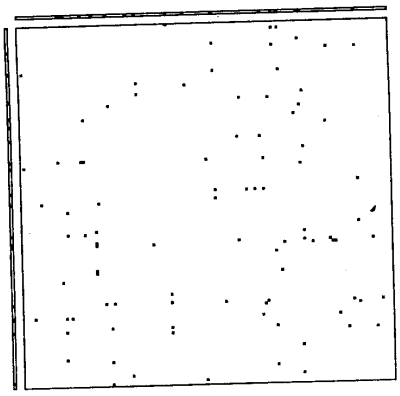


Figure 18. A Solution of the 130 × 130 Fault Map Problem (Ra = Ca = 17)

that the average number of the iteration steps and the frequency of the convergence to the optimum solution is determined not by the problem size but by the degree of the difficulty in the problem.

Our algorithm can approximate solutions to the spare-allocation problems with at least small size of fault maps in a nearly constant time with $O(n)$ processors.

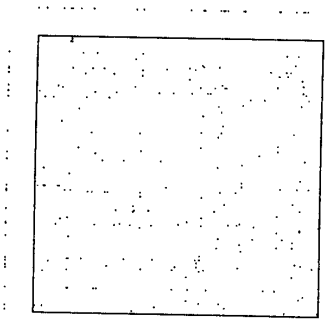
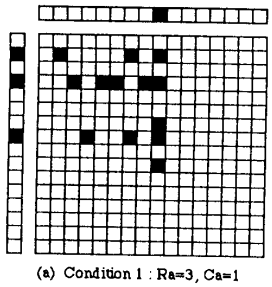
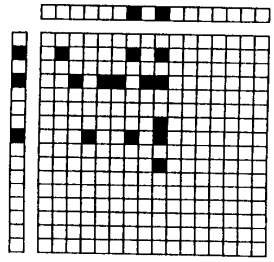


Figure 19. 200 × 200. A Solution of the 200 × 200 Fault Map Problem (Ra = Ca = 24)





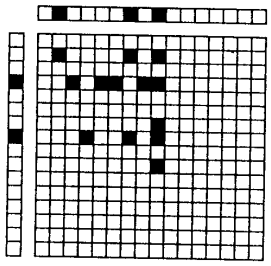
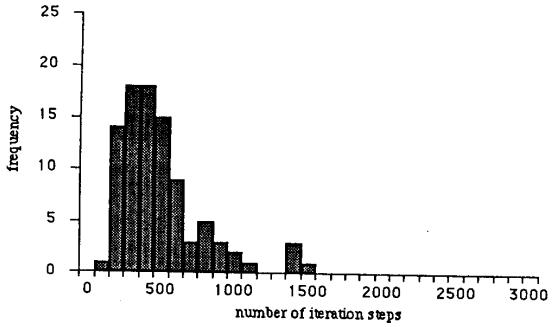
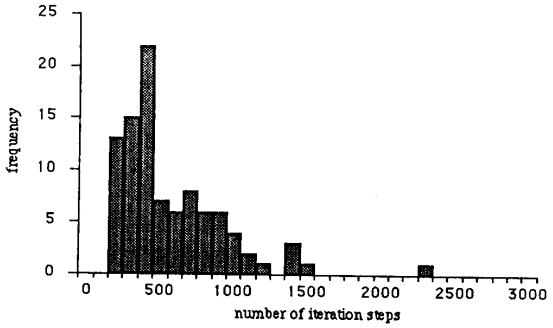


Figure 20. Solutions of the Fault Map Problem of Fig. 4 with the Different Conditions



a. The 40 × 40 Fault Map Problem



b. The 90 × 90 Fault Map Problem

Figure 21. The Relationship between the Frequency and the Number of Iteration Steps to Converge to the Optimum Solutions

APPENDIX

Theorem

The system always satisfies $\Delta E/\Delta t \leq 0$ under two conditions such as $\Delta U_i/\Delta t = -\Delta E/\Delta V_i$ and $V_i = f(U_i)$ where E is the computational Lyapunov energy function and $f(U_i)$ is a hysteresis binary function:

$$f(U_i) = 1 \text{ if } U_i > \text{UTP}$$

$$= 0 \text{ if } U_i < \text{LTP}$$

unchanged otherwise.

Proof: Consider the derivatives of the computational energy E with respect to time t.

$$\frac{\Delta E}{\Delta t} = \sum_i \frac{\Delta V_i}{\Delta t} \frac{\Delta E}{\Delta V_i}$$

$$= \sum_i \frac{\Delta V_i}{\Delta t}\left(-\frac{\Delta U_i}{\Delta t}\right), \text{ where } \frac{\Delta E}{\Delta V_i} \text{ is replaced by}$$

More — on the outside.

TABLE 1
Summary of Simulation Results

Fault Map Problem Size	Average Interation Steps to Optimum Solution	Convergence Frequency to Optimum Solution
16 × 16 (Fig. 3)	430.1	96%
16 × 16 (Fig. 4)	79.9	90%
16 × 16 (Fig. 5)	209.1	99%
16 × 16 (Fig. 6)	211.8	97%
20 × 20 (Fig. 7)	269.3	98%
30 × 30 (Fig. 8)	171.3	93%
40 × 40 (Fig. 9)	481.7	100%
50 × 50 (Fig. 10)	218.7	87%
60 × 60 (Fig. 11)	554.6	100%
70 × 70 (Fig. 12)	495.8	100%
80 × 80 (Fig. 13)	279.7	85%
90 × 90 (Fig. 14)	516.3	95%
100 × 100 (Fig. 15)	278.3	83%
110 × 110 (Fig. 16)	688.7	100%
120 × 120 (Fig. 17)	355.4	81%
130 × 130 (Fig. 18)	328.3	83%
200 × 200 (Fig. 19)	710.1	84%

$$-\frac{\Delta U_i}{\Delta t}$$

$$= -\sum_i \left(\frac{\Delta U_i}{\Delta t}\,\frac{\Delta V_i}{\Delta U_i}\right)\left(\frac{\Delta U_i}{\Delta t}\right)$$

$$= -\sum_i \left(\frac{\Delta V_i}{\Delta U_i}\right)\left(\frac{\Delta U_i}{\Delta t}\right)^2$$

Let $\Delta U_i/\Delta t$ be $U_i(t+\Delta t)-U_i(t)/\Delta t$. Let $\Delta V_i/\Delta U_i$ be $V_i(t+\Delta t)-V_i(t)/U_i(t+\Delta t)-U_i(t)$. It is necessary and sufficient to consider the following four regions:

Region 1: $U_i(t) > \text{UTP}$ and $V_i(t) = 1$

Region 2: $\text{LTP} \le U_i(t) \le \text{UTP}$ and $V_i(t) = 1$

Region 3: $\text{LTP} \le U_i(t) \le \text{UTP}$ and $V_i(t) = 0$

Region 4: $U_i(t) < \text{LTP}$ and $V_i(t) = 0$

In region 1: We must consider four possible cases for $U_i(t+\Delta t)$:

a $U_i(t+\Delta t) > U_i(t)$

b $\text{LTP} \le U_i(t+\Delta t) < U_i(t)$

c $U_i(t+\Delta t) < \text{LTP} < U_i(t)$

d $U_i(t+\Delta t) = U_i(t)$

In a and b, $V_i(t+\Delta t) = V_i(t) = 1 \implies \Delta V_i/\Delta U_i = 0$. Therefore $\Delta E/\Delta t = 0$.

In d, $\Delta U_i/\Delta t = 0 \implies \Delta E/\Delta t = 0$.

In c, $V_i(t+\Delta t) = 0 \implies \Delta V_i/\Delta U_i = 0-1/\text{negative number} > 0$ and $\Delta U_i/\Delta t < 0$. Therefore $\Delta E/\Delta t < 0$.

Thus $\Delta E/\Delta t \le 0$ is always satisfied in region 1.

Similarly, in region 2-4, $\Delta E/\Delta t \le 0$ is always satisfied.

Q.E.D.

REFERENCES

[1] S. E. Schuster, "Multiple word/bit line redundancy for semiconductor memories", *IEEE J. Solid-State Circuits*, vol SC-13, 1978 Oct, pp 698-703.

[2] R. P. Cenker, et al, "A fault-tolerant 64K dynamic random-access memory", *IEEE Trans. Electron Devices*, vol ED-26, 1979 Jun, pp 853-860.

[3] B. F. Fitzgerald, E. P. Thoma, "Circuit implementation of fusible redundant addresses on RAMs for productivity enhancement", *IBM J. Res. Develop.*, vol 24, 1980 May, pp 291-298.

[4] C. H. Stapper, A. N. McLaren, M. Dreckmann, "Yield model for productivity optimization of VLSI memory chips with redundancy and partially good product", *IBM J. Res. Develop.*, vol 24, 1980 May, pp 398-409.

[5] R. Sud, K. C. Hardee, "16-K static RAM takes new route to high speed", *Electronics*, 1980 Sep 11, pp 117-123.

[6] R. C. Evans, "Testing repairable RAMs and mostly good memories", *Proc. Int'l Test Conf.*, 1981, pp 49-55.

[7] Y. Hayasaka, K. Shimotori, K. Okada, "Testing system for redundant memory", *Proc. Int'l Test Conf.*, 1982, pp 240-244.

[8] C. A. Benevit, et al, "A 256K dynamic random access memory", *IEEE J. Solid-State Circuits*, vol SC-17, 1982 Oct, pp 857-862.

[9] R. O. Carlson, C. A. Neugebauer, "Future trends in wafer scale integration", *Proc. IEEE*, vol 74, 1986 Dec, pp 1741-1752.

[10] M. Tarr, D. Boudreau, R. Murphy, "Defect analysis system speeds test and repair of redundant memories", *Electronics*, 1984 Jan 12, pp 175-179.

[11] J. R. Day, "A fault-driven, comprehensive redundancy algorithm", *IEEE Design&Test*, vol 2, 1985 Jun, pp 35-44.

[12] S-Y. Kuo, WS. K. Fuchs, "Efficient spare allocation for reconfigurable arrays", *IEEE Design&Test*, vol 4, 1987 Feb, pp 24-31.

[13] C-L. Wey, F. Lombardi, "On the repair of redundant RAMs", *IEEE Trans. Computer-Aided Design*, vol CAD-6, 1987 Mar, pp 222-231.

[14] R. W. Haddad, A. T. Dahbura, "Increased throughput for the testing and repair of RAMs with redundancy", *Proc. IEEE ICCAD*, 1987, pp 230-233.

[15] W. K. Huang, Y-N. Shen, F. Lombardi, "New approaches for the repairs of memories wirh redundancy by row/column deletion for yield enhancement", *IEEE Trans. Computer-Aided Design*, vol 9, 1990 Mar, pp 323-328.
[16] J. J. Hopfield, D. W. Tank, "Neural computation of decisions in optimization problems", *Biological Cybernetics*, vol 52, 1985, pp 141-152.
[17] W. S. McCulloch, W. H. Pitts, "A logical calculus of ideas immanent in nervous activity", *Bul. Mathematical Biophysics*, vol 5, 1943, p 115.
[18] Y-P. S. Foo, Y. Takefuji, H. Szu, "Binary neurons with analog communication links for solving large-scale optimization problems", *Proc. Int'l Neural Network Soc. Mtg.*, 1988 Sep.
[19] Y. Takefuji, K. C. Lee, "A near-optimum parallel planarization algorithm", *Science*, vol 245, 1989 Sep, pp 1221-1223.
[20] Y. Takefuji, K. C. Lee, "Artificial neural networks for four-coloring map problems and K-colorability problems", *IEEE Trans. Circuits & Systems*, vol 38, 1991 Mar, pp 326-333.
[21] Y. Takefuji, K. C. Lee, "A parallel algorithm for tiling problems", *IEEE Trans. Neural Networks*, vol 1, 1990 Mar, pp 143-145.
[22] Y. Takefuji, C. W. Lin, K. C. Lee, "A parallel algorithm for estimating the secondary structure in ribonucleic acids", *Biological Cybernetics*, vol 63, 1990, pp 337-340.
[23] Y. Takefuji, L. L. Chen, K. C. Lee, J. Huffman, "Parallel algorithms for finding a near-maximum independent set of a circle graph", *IEEE Trans. Neural Networks*, vol 1, 1990 Sep, pp 263-267.
[24] Y. Takefuji, K. C. Lee, "An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviours of neural dynamics", *Biological Cybernetics*, vol 64, 1991, pp 353-356.
[25] Y. Takefuji, K. C. Lee, "A super parallel sorting algorithm based on neural networks", *IEEE Trans. Circuits & Systems*, vol 37, 1990 Nov, pp 1425-1429.

AUTHORS

Nobuo Funabiki; Department of Electrical Engineering; Case Western Rreserve University; Cleveland, Ohio 44106 USA.

Nobuo Funabiki is a graduate student at Case Western Reserve University working toward the PhD degree in the Electrical Engineering. He is also working in Sumitomo Metal Ind., Ltd. (Japan). He received his BS in mathematical engineering and information physics from the University of Tokyo (Japan) in 1984. His research interests include channel routing problems, traffic control problems in three-stage/multistage connecting networks, time slot assignment problems in TDM hierarchical switching systems, and broadcast scheduling problems. He is a student member of the IEEE Computer Society, and American Assoc. for the Advancement of Science.

Dr. Yoshiyasu Takefuji; Department of Electrical Engineering; Case Western Reserve University; Cleveland, Ohio 44106 USA.

Yoshiyasu Takefuji is an assistant professor of Electrical Engineering at Case Western Reserve University. Before joining Case Western in 1988, he taught at the University of South Florida and the University of South Carolina. He received his BS (1978), MS (1980), and PhD (1983) in Electrical Engineering from Keio University (Japan). His research interests focus on neural network parallel computing for solving real problems. He is interested in VLSI applications and silicon architecture. He received the National Science Foundation/Research Initiation Award in 1989 and is an NSF advisory panelist. A member of the IEEE Computer Society, ACM, International Neural Network Society, and American Association for the Advancement of Science, he received the Information Processing Society of Japan's best paper award in 1980. He coauthored two books, *Digital circuits* and *Neural Network Computing* (in Japanese). He will publish the book *Neural Network Parallel Computing* in 1991. He is an Editor of the *J. Neural Network Computing* and an associate editor of *IEEE Trans. Neural Networks*. He has published more than 70 technical papers.

Manuscript TR90-070 received 1990 April 26; revised 1990 October 1; revised 1991 January 25.

IEEE Log Number 43762 ◄TR►

MANUSCRIPTS RECEIVED *MANUSCRIPTS RECEIVED* *MANUSCRIPTS RECEIVED* *MANUSCRIPTS RECEIVED*

"Decision making: Incorrect use of abstract parameters may lead to wrong conclusions", Dr. Telba Z. Irony □ Dept. of Operations Research □ The George Washington University □ Washington, DC 20052 □ USA. (TR91-095)

"Exponentiated Weibull family for analyzing bathtub failure-rate data", Govind S. Mudholkar □ Dept. of Statistics □ University of Rochester □ Rochester, New York 14627 □ USA. (TR91-096)

"Robust multiple linear regression with the observations per estimated-parameter severely limited", Dr. Albert H. Moore, Professor □ Dept. of Mathematics & Computer Science □ AFIT/ENC □ Wright-Patterson AFB, Ohio 45433-6583 □ USA. (TR91-097)

"Systematic Bayes prior-assignment by coupling the mini-max entropy and moment-matching methods", Won Hyo Yoon □ Korea Institute of Nuclear Safety □ POBox 16 □ Daeduk Danji □ Daejeon □ Republic of KOREA. (TR91-098)

"Analysis of complex-system reliability using a symbolic logic system", Dr. S. S. Rao □ Mechanical Engineering □ Purdue University □ West Lafayette, Indiana 47907 □ USA. (TR91-099)

"Adaptive tests for the median", Dr. Hermanus H. Lemmer □ Dept. of Statistics □ Rand Afrikaans University □ POBox 524 □ Johannesburg 2000 □ Republic of SOUTH AFRICA. (TR91-100)

"Reliability of cyclic m-consecutive-k-out-of-n:F systems", Philippos D. Alevizos □ Dept. of Mathematics □ University of Patras □ 261 10 Patras □ GREECE. (TR91-101)

"Estimation of the number of remaining software faults with a redundant system", Robert W. Chen □ Dept. of Mathematics □ University of Miami □ POBox 249,085 □ Coral Gables, Florida 33124 □ USA. (TR91-102)

"Modeling repairable systems with failure rates that depend on age & maintenance", Dr. Jack-Kang Chan □ Norden Systems Inc. □ 75 Maxess Road □ Melville, New York 11747 □ USA. (TR91-103)

"Weibull parameter estimation in a Weibull-type proportional-hazard model", Tsunenori Ishioka □ Software Research Center □ Ricoh Company Ltd. □ Nihon-Seimei-Kasugacho Bldg., 1-33-13 □ Hongo, Bunkyo-ku, Tokyo 113 □ JAPAN. (TR91-104)

"The effects of infrequent, but difficult, input conditions on N-Version programming", Dr. Joshua Etkin, Professor □ College of Engineering □ Boston University □ 44 Cummington Street □ Boston, Massachusetts 02215 □ USA. (TR91-105)

"A class of efficient tests for an increasing-failure-rate-average distribution under random censoring", Dr. Ram C. Tiwari □ Dept. of Mathematics □ University of North Carolina □ Charlotte, North Carolina 28223 □ USA. (TR91-106)

More — on the outside.