アルゴリズム 10－4
（1989. 9. 22）

大規模シストリック・アレイに対する修復パス構成アルゴリズム

小　澤　孝　夫
京都大学工学部

大規模なプロセッサー・アレイが欠陥を持つプロセッサーを含むとき、正常なプロセッサーだけによってシストリック・アレイを構成するためのアルゴリズムを与えている。この問題は、欠陥プロセッサーをどのように予備プロセッサーでおき代えるかを示す修復パスを求めることに帰着されるが、 本文では、欠陥プロセッサーに対するキューを構成し、キューの取り出し口にある欠陥プロセッサーに対し順次修復パスを能率よく求めている。また、格子状アレイの性質を利用して巧妙な分枝限定を行っている。 アルゴリズムの時間複雑度は、欠陥プロセッサーの数を n として、最悪の場合で $O(n^3)$ であるが、 n が大きいとこのような場合は非常に稀であるといえる。領域複雑度は $O(n)$ である。

AN EFFICIENT COMPENSATION-PATH FINDING ALGORITHM FOR A LARGE-SCALE SYSTOLIC ARRAY

Takao OZAWA
Department of Electrical Engineering
Kyoto University, Kyoto, 606, Japan

The problem of constructing a systolic array from a processor array containing faults can be reduced to the problem of finding compensation paths for faulty processors. In this paper an efficient solution algorithm for the latter problem is given. Four queues of faulty processors are constructed, and compensation paths are successively found for the faulty processors at the exits of the queues. A novel branching scheme is used in the search for compensation paths. The worst-case time complexity of the algorithm is $O(n^3)$ where n is the number of faulty processors, but it can be said such worst cases rarely happen, if n is large. The space complexity is $O(n)$.

1. INTRODUCTION

Very large-scale or wafer-scale integrated circuits for achieving a system constituted by a large number of circuits on a chip or wafer inevitablly contain, in today's IC manufacturing technology, faulty circuits in them. In order to enhance production yield, therefore, such integrated circuits are manufactured with redundancy, and construction of the system using only working circuits is tried[1]-[6]. Very often, however, problems which must be solved for such construction have been found NP complete.

In this paper we consider a rectangular(2 dimensional) array of processing elements(PE's) which are connected to form a grid. Such a processor array is regarded as one of the most useful multiprocessor architectures suitable for parallel computation, and the construction of a systolic array(logical array) from the grid of PE's containing faults(physical array) has been considered in various forms depending on the location of spare PE's and the type of communication buses and switches[5]. Very recently Kung et al. considered a processor array having spare PE's on its peripheral and single-track switches for reconfiguration[6]. Their basic idea for reconfiguration is to find paths, called compensation paths, which represent the replacement sequences of PE's from faulty PE's to spare PE's on the grid. To this end they introduced a graph, called a contradiction graph, which indicates compensation paths unable to use at the same time. They reduced the problem of determining whether the reconfiguration is possible to the problem of finding a maximum vertex independent set of the contradiction graph. It is known, however, the latter problem is NP complete, and they gave an branch-and-bound solution algorithm for it based on the algorithm of Bron and Kerborsch for finding all cliques of a graph[7].

We notice that finding compensation paths can be done on a grid which is a very special planar graph, and present an efficient algorithm for it based on this observation.

2. PROBLEM FORMULATION

Our problem formulation is almost the same as that of Kung et al.. Let P be the rectangular region in an x-y plane defined by $0 \leq x \leq N+1$ and $0 \leq y \leq M+1$ where N and M are integers. We assume that PE's are placed on the integer coordinate points (x,y)(x, y:integers) of P except the four corner points of P. The PE's on the peripheral of P are spares. Single-track communication channels are used to connect PE's, and four-state switches for exchanging communication paths are placed at the intersections of the communication channels. See Fig. 1. We are given the distribution of faulty PE's on P(it is assumed that the communication lines and switches are fault-free), and our goal is to obtain an N×M systolic array consisting of good(non-faulty) PE's from the initial setting of a systolic array with PE's being placed on the integer coordinate points (x,y) such that $1 \leq x \leq N$ and $1 \leq y \leq M$. This process is called reconfiguration. A compensation path is defined to be a path which connects a faulty PE and a spare PE going through some other PE's. It represents the sequence of replacement of PE's in the reconfiguration, that is, if the PE's on the compensation path are p_1, p_2, p_3, ..., p_{m-1}, p_m where p_1 is faulty and p_m is a spare, then p_1 is replaced by p_2, p_2 is replaced by p_3, .., and p_{m-1} is replaced by p_m in the reconfiguration. It is assumed that faulty PE's can be converted into connecting elements(communication lines).

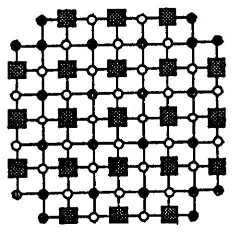
Fig. 1 Array of PE's.

Kung et al. showed that the above defined reconfiguration is possible if, for each faulty PE, a compensation path

COND1: which is a horizontal or vertical straight line, COND2: which does not go through any other faulty PE,

COND3: which does not cross any other compensation path, and

COND4: for which there exists no other compensation path running in parallel and in the opposite direction with distance 1,

can be found. Thus the problem we are going to consider in the following sections is:

PROBLEM: Obtain, whenever possible, compensation paths satisfying the conditions COND1-COND4 for all the faulty PE's.

Because of COND1, we may search compensation paths on a rectangular grid graph(without the vertices at the four corners) whose vertices correspond to PE's. Paths satisfying COND1 and COND2 are called candidate paths, and the candidate path going leftward(resp. rightward, downward, upward) from a faulty PE is called a KL-path (resp. KR-path, KD-path, KU-path). Candidate paths can easily be determined, and thus our main concern is to select, from candidate paths, compensation paths satisfying the remaining two conditions. At a step in the procedure of finding compensation paths a faulty PE for which a compensation path has not yet been found is called a U-PE(unprocessed PE). Initially all the faulty PE's are U-PE's.

3. SOLUTION ALGORITHM

3.1 Lists and Queues for U-PE and Candidate Paths

Consider a U-PE located at point (x,y) on P. Initially it has its KL-path(resp. KR-path, KD-path, KU-path) if there exists no other U-PE which is located at point (x',y') such that $x'<x$ and $y'=y$(resp. $x'>x$ and $y'=y$, $x'=x$ and $y'<y$, $x'=x$ and $y'>y$). Therefore, we sort faulty PE's according to their x-coordinates and y-coordinates, and construct two linear linked-lists named LR-list and DU-list respectively. LR-list(resp. DU-list) is defined to be a double-ended linked-list of U-PE's such that if its i-th cell and j-th cell contain U-PE's located at (x_i,y_i) and (x_j,y_j) respectively and if i<j, then it must be that $x_i \leq x_j$ and, in case $x_i=x_j$, $y_i<y_j$(resp. $y_i \leq y_j$ and, in case $y_i=y_j$, $x_i<x_j$). LR-list is used as two queues of U-PEs, named L-queue and R-queue, whose exits are the left and right ends of LR-list respectively. Likewise, DU-list is used as two queues, named D-queue and U-queue respectively, whose exits are the left and right ends of DU-list respectively. The U-PE at the exit of L-queue (resp. R-queue, D-queue, U-queue) is called UL-PE (resp. UR-PE, UD-PE, UU-PE).

We also construct four linked-lists for KL-paths, KR-paths, KD-paths and KU-paths respectively. The list for KL-paths(resp. KR-paths) is obtained from LR-list by deleting U-PE's each having no KL-path(resp. KR-path). Likewise the list for KD-paths(resp. KU-paths) is obtained from DU-list by deleting U-PE's each having no KD-path (resp. KU-path).

A block in LR-list(resp. DU-list) is defined to be the set of U-PE's placed on the points having the same x-coordinate(resp. y-coordinate). Only the first(resp. last) U-PE of a block in LR-list can have a KD-path(resp. KU-path). Likewise, only the first(resp. last) U-PE of a block in DU-list can have a KL-path (resp. KR-path). Therefore we can use the linked-lists for candidate paths also as the linked-lists for blocks.

3.2 Compensation-Path Finding Algorithm

[I] The major step of our solution algorithm is the selection of compensation paths from candidate paths. We first present the key factors to be attended to in understanding the algorithm.

(I.1) Search for compensation paths: The algorithm repeatedly searches for compensation paths examining the conditions, given later in (II.1)-(II.5), on the candidate paths of UL-PE, UR-PE, UD-PE and UU-PE. If the conditions in (II.2)-(II.5) apply, it selects, according to the conditions, a compensation path for one of these U-PE's or a set of compensation paths for all of these U-PE's at the same time. Note that it may happen that two or more of these U-PE's coincide, that is, the same U-PE appears at the exits of two or more queues.

(I.2) Branching: It may happen that the compensation path(s) for a U-PE(s) at the exits of the four queues can not uniquely be determined. Then a branching occurs in the search for compensation paths. The algorithm selects one(a set) of candidate paths as the compensation path(s) for the U-PE(s) and proceeds to the branch of the search initiated from this selection. The state of the queues and lists at the branching point is stored for possible later use.

(I.3) Deletion operations on the lists and queues: If a compensation path can be determined for a U-PE, then this PE is deleted from both LR-list and DU-list(thus from L-queue, R-queue, D-queue and U-queue). The remaining candidate paths which the U-PE may have are also deleted from the lists of candidate paths.

(I.4) Successful ending: If the four queues for U-PE's become empty as a result of (I.3), the algorithm ends successfully, that is, compensation paths for all faulty PE's are found.

(I.5) Unsuccessful ending or termination: If it is not possible to find a compensation path for any of UL-PE, UR-PE, UD-PE and UU-PE, then the algorithm ends unsuccessfully, or, if it is in a branch of the search for compensation paths, terminates the search unsuccessfully and returns to the branching point. In the latter case the algorithm deletes the candidate path from which the branch of the search started and examines if there remains a branch which has not yet been investigated. If one remains, it cancels the compensation paths determined in the old branch, restores the queues and lists for the remaining U-PE's and candidate paths at the branching point, and proceeds to the new branch.

[II] Next, we present the details of (I.1).

(II.1) Whenever a new U-PE comes to the exit(s) of the queue(s), the following operation END or, if the algorithm is in a branch of the search, TERM is executed. In the latter case the above statement of (I.5) applies.

[END] If COND-E holds, end the search for compensation paths.

[TERM] If COND-E holds, terminate the search for compensation paths in the branch.

(COND-E) The new U-PE has no candidate path.

(II.2) CASE-1: If UL-PE has the KL-path, then this candidate path does not cross any other candidate path and thus it can be chosen as the compensation path for this U-PE without affecting the path selection for other U-PE's. The same can be said regarding the other U-PE's at the exits of the queues. Therefore we define the following operation SEL-1.

[SEL-1] If COND-S1 holds, do EX-1.

(COND-S1) UL-PE has the KL-path, UR-PE has the KR-path, UD-PE has KD-path or UU-PE has the KU-path.

(EX-1) Select the candidate path satisfying the condition as the compensation path for the U-PE which has the candidate path. Delete the U-PE from the four queues for U-PE's. If the queues become empty, stop (the algorithm ends successfully). Otherwise delete, from the lists of candidate paths, the remaining candidate paths which the U-PE may have.

(II.3) CASE-2: If a U-PE has only one candidate path, it must be chosen as its compensation path. Therefore, we define the following operation.

[SEL-2] If COND-S2 holds, do EX-2.

(COND-S2) Any of UL-PE, UR-PE, UD-PE and UU-PE has only one candidate path.

(EX-2) EX-1 plus the operation: delete every candidate path which does not satisfy COND3 or COND4.

The details of the deletion operation stated in EX-2 will be given in section 3.4.

(II.4) CASE-A: Suppose that none of COND-E, COND-S1 and COND-S2 holds for the U-PE's at the exits of the four queues. Then each of these U-PE's has at least two candidate paths. The algorithm proceeds to the following test, which, in general, initiates a branch in the search for compensation paths. The statement of (I.2) above applies. We define:

[SEL-A] If COND-AL(resp. COND-AR, COND-AD, COND-AU) holds, do EX-A.

(COND-AL) UL-PE has the KR-path.

(COND-AR) UR-PE has the KL-path.

(COND-AD) UD-PE has the KU-path.

(COND-AU) UU-PE has the KD-path.

(EX-A) EX-2 plus the operations: store the current state of the queues and lists for U-PE's and candidate paths, and proceed to the branch of the search initiated by the above selection.

Fig. 2 illustrates SEL-A.

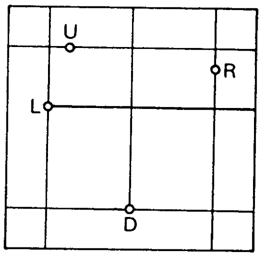

Fig. 2 SEL-A.
L: UL-PE
R: UR-PE
D: UD-PE
U: UU-PE

SEL-A is applied to UL-PE first. If COND-AL holds but the search terminates unsuccessfully in this branch(the statement in (I.5) applies), or if COND-AL does not hold, then SEL-A is applied to UR-PE and so on. If COND-AU does not hold, or if it holds but the search terminates unsuccessfully in this branch, the search in CASE-A terminates unsuccessfully. At this point SEL-2 is applied, as long as it is applicable.

(II.5) CASE-B: If CASE-A does not occur, or if it does but terminates unsuccessfully and the application of SEL-2 is finished, then it must be that the U-PE's at the exits of the four queues are all distinct. Moreover, each of the four U-PE's has exactly two candidate paths, that is, each of UL-PE and UR-PE has the KD-path and KU-path, and each of UD-PE and UU-PE has the KL-path and KR-path. Let the coordinates of UL-PE, UR-PE, UD-PE and UU-PE be (x_L, y_L), (x_R, y_R), (x_D, y_D) and (x_U, y_U) respectively. There can be one of the following four cases.

(COND-B1) $x_D > x_U$ and $y_L < y_R$.
(COND-B2) $x_D < x_U$ and $y_L > y_R$.
(COND-B1) $x_D > x_U$ and $y_L > y_R$.
(COND-B1) $x_D < x_U$ and $y_L < y_R$.

Noting that the compensation paths should not cross each other, we see that, for each of the above four cases, there can be two ways to select the compensation paths for the four U-PE's. Thus a branching in the search for compensation paths occurs. We define:

[SEL-B] Do EX-B-a(resp. EX-B-b).

(EX-B-a) Choose the KU-path of UL-PE, the KR-path of UU-PE, the KD-path of UR-PE and the KL-path of UD-PE as the compensation paths for these U-PE's respectively, store the current state of the queues and list, and proceed to the branch initiated by this selection.

(EX-B-b) Choose the KD-path of UL-PE, the KR-path of UD-PE, the KU-path of UR-PE and the KL-path UU-PE as the compensation paths for these U-PE's respectively, store the current state of the queues and lists, and proceed to the branch initiated by this selection.

Figs. 3 and 4 illustrate SEL-B when COND-B1 and COND-B3 hold respectively.

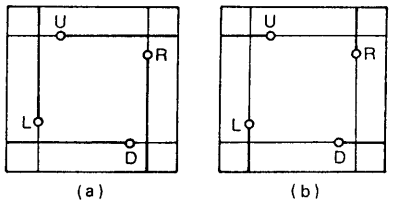

Fig. 3 CASE-B under COND-B-1.
(a) EX-B-a. (b) EX-B-b.

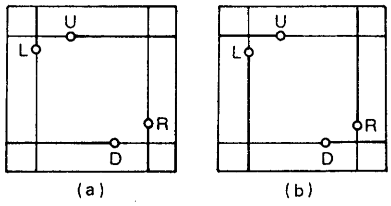

Fig. 4 CASE-B under COND-B-3.
(a) EX-B-a. (b) EX-B-b.

In CASE-B-1 and CASE-B-3 EX-B-a is executed first and if the search for compensation paths in this branch terminates unsuccessfully, EX-B-b is executed. If the search in this branch terminates unsuccessfully, CASE-B terminates unsuccessfully. In CASE-B-2 and CASE-B-4 EX-B-b is executed first.

[III] As a summary we describe how the compensation path finding algorithm proceeds.

Whenever a new U-PE comes to the exit(s) of the queues for U-PE, END or TERM is applied. If

COND-E does not hold, SEL-1 is applied. If neither COND-E nor COND-S1 holds, SEL-2 is applied. If none of COND-E, COND-S1 and COND-S2 holds, SEL-A is applied. If none of COND-E, COND-S1, COND-S2 and COND-AL--COND-AU holds, SEL-B is applied. The application of SEL-A introduces a branching. If the search for compensation paths in the branch terminates unsuccessfully, the next condition of SEL-A is tested. The application of SEL-B also introduces a branching. If the execution of EX-B-a in CASE- B-1 and CASE-B-3(resp. EX-B-b in CASE-B-2 and CASE-B-4) leads to an unsuccessful termination of the search in this branch, EX-B-b(resp. EX-B-a) is executed. If the search in the branch thus initiated terminates unsuccessfully, the entire search for compensation paths ends unsuccessfully.

3.3 Search for Compensation Paths in the Branches

In this section we describe how the search for compensation paths which is initiated at the branching operation in CASE-A or CASE-B, can be done. First we define:

[SEL-3] If COND-S3 holds, do EX-3.

(COND-S3) UL-PE(resp. UR-PE, UD-PE or UU-PE) has KD-path and KU-path(resp. KD-path and KU-path, KL-path and KR-path, KL-path and KR-path) for which there exists no candidate path violating COND3, but there exists a candidate path violating COND4.

(EX-3) Select the candidate paths satisfying the condition as the "temporal" compensation paths for the U-PE. Delete the remaining candidate path(s) which the U-PE may have.

[SEL-4] If COND-S4 holds, do EX-4.

(COND-S4) UL-PE(resp. UR-PE, UD-PE or UU-PE) has KD-path and KU-path(resp. KD-path and KU-path, KL-path and KR-path, KL-path and KR-path) for which there exists no candidate path violating COND3 or COND4.

(EX-4) Select one of the candidate paths satisfying the condition as the compensation path for the U-PE. Delete the remaining candidate path(s) which the U-PE may have. If there exist temporal compensation paths previously selected, choose, from them, the (permanent) compensation paths in such a way that COND4 is satisfied.

[SEL-5] If COND-S5 holds, do EX-5.

(COND-S5) UL-PE(resp. UR-PE, UD-PE or UU-PE) has either KD-path or KU-path(resp. either KD-path or KU- path, either KL-path or KR-path, either KL-path or KR-path) for which there exists no candidate path violating COND3.

(EX-5) Select the candidate path satisfying the condition as the compensation path for the U-PE. Delete the remaining candidate path(s) which the U-PE may have. If there exist temporal compensation paths previously selected, choose, from them, the (permanent) compensation paths in such a way that COND4 is satisfied.

[I] Let us consider CASE-A first. The search for compensation paths in this case can be completed by the applications of SEL-1, SEL-2, SEL-3, SEL-4 and/or SEL-5. COND-S1 for SEL-1 is tested first, since EX-1 is simpler. No more CASE-A or CASE-B occurs. The detail of the selection operation is as follows.

Suppose that UL-PE has the KR-path and that this candidate path is selected as the compensation path of the U-PE. See Fig. 2. Let us consider the search for compensation paths in the region under the compensation path. For this region the U-PE which immediately precedes the (former) UL-PE in DU-list is set to UU-PE, and the exit of U-queue is moved to this U-PE.

Note that no U-PE in this region has the KU-path, since all the candidate paths crossing the compensation path are deleted after the above-stated selection operation. If the new UU-PE has both the KL-path and KR-path, then SEL-4 or SEL-5 is applied(see Fig. 5). If it has either the KL-path or KR-path, SEL-5 is applied. If it has the KD-path only, SEL-2 is applied. The same can be said for the U-PE which becomes UU-PE after the application of SEL-3 or SEL-2.

The same applys to the search in the region above the first compensation path for the UL-PE, if UU-PE is replaced with UD-PE.

We can make similar statements to the above for the other compensation path selections in

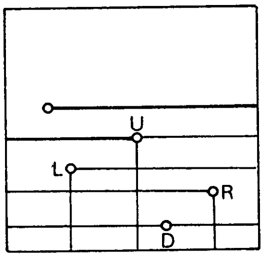

Fig. 5 SEL-3 after CASE-A.

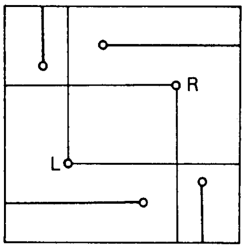

Fig.6 CASE-A after EX-B-a under COND-B3.

CASE-A.

[II] Next we consider CASE-B.

(II-1) Suppose that COND-B1(resp. COND-B2) holds and EX-B-a(resp. EX-B-b) is executed. See Fig. 3(a). The search for compensation paths in this case can be terminated by the applications of SEL-1, SEL-2 and/or SEL-5. Suppose that COND-B1 holds and EX-B-a is executed. If UL-PE has the KL-path and KU-path(resp. KU-path and KR-path), SEL-1(resp. SEL-5) is applied and the KL-path (resp. KU-path) is selected as the compensation path for this U-PE.

(II-2) Suppose that COND-B1(resp. COND-B2) holds and EX-B-b(resp. EX-B-a) is executed. See Fig. 3(b). In this case it may happen later that COND-A, COND-B1, COND-B2, COND-B3 or COND-B4 holds again.

(II-3) Suppose that COND-B3 holds and EX-B-a or EX-B-b is executed. In this case, after the repeated applications of SEL-1 and/or SEL-2, CASE-A may happen once later, but never more than once. CASE-B does not occur. As stated in [I] above the search for compensation paths in CASE-A can be terminated by repeated applications of SEL-1, SEL-2, SEL-3, SEL-4 and/or SEL-5. The same can be said for the case where COND-B4 holds and EX-B-a or EX-B-b is executed.

Fig. 6 illustrates the case where CASE-A occurs after EX-B-a under COND-B3. UL-PE has the KR-path and UR-PE has the KL-path.

3.4 Deletion of Candidate Paths

The deletion of candidate paths to satisfy COND3 can be done as follows. Suppose the KR-path (resp. KL-path) of a U-PE is selected as the compensation path. Then either the KD-path or KU-path, if exists, of each U-PE which follows (resp. precedes) the relevant U-PE in LR-list must be deleted. Which of the KD-path or KU-path should be deleted can be determined by comparing the y-coordinates of these two U-PE's. Similarly, if the KU-path (resp. KD-path) of a U-PE is selected as the compensation path, then either the KL-path or KR-path of each U-PE which follows(resp. precedes) the relevant U-PE in DU-list is deleted.

The deletion of candidate paths to satisfy COND4 can be done by utilizing the block structure of LR-list and DU-list, or more precisely, using the lists for candidate paths. If the KL-path or KR-path of a U-PE is selected as a compensation path, then the U-PE's belonging to the blocks which are adjacent, in DU-list, to the block containing the relevant U-PE, are examined if they have candidate paths violating COND4. Similarly, if the KU-path or KL-path of a U-PE is selected as the compensation path, then the U-PE's belonging to the blocks which are adjacent, in DU-list, to the block containing the relevant U-PE, are examined for the existence of candidate paths violating COND4.

3.5 Evaluation of the Algorithm

We have the following lemma and theorem for our compensation-path finding algorithm. Let n be the number of faulty PE's.

[Lemma] The total number of executions of EX-B-b under COND-B1 and of EX-B-a under COND-B2 is at most n/8.

(proof) At each execution of EX-B-b under COND-B1 or of EX-B-a under COND-B2, the compensation paths for four U-PE's are determined. Before CASE-B occurs there must be four U-PE's for which compensation paths have been determined(UL-PE does not have the KL-path, and thus there must be a U-PE lying to the left of this U-PE. A similar statement can be made for UR-PE, UD-PE and UU-PE). As stated in section 3.3, CASE-B never occurs after EX-A, after Ex-B-a under COND-B1, COND-B3 or COND-B4, or after EX-B-b under COND-B2, COND-B3 or COND-B4. Thus this lemma holds.

[Theorem] The time complexity of the algorithm presented in sections 3.1-3.4 is $O(n^3)$ and the space complexity is $O(n)$.

(Proof) LR-list, DU-list and the lists for candidate paths can be constructed in $O(n \log n)$ time by sorting the x and y coordinates of the faulty PE's. Each of the tests for the conditions on UL-PE, UR-PE, UD-PE and UU-PE can be done in $O(1)$ time. The deletion of candidate paths from the lists after the selection of a compensation path(s) can be done in $O(n)$ time. The selection of (permanent) compensation paths from temporal compendation paths can be done $O(n)$ time. Therefore, the search for compensation paths in the branch initiated at EX-A, at EX-B-a under COND-B1, COND-B3 or COND-B4, or at EX-B-b under COND-B2, COND-B3 or COND-B4, can be completed in $O(n^2)$ time. Now, it is possible that EX-B-b under COND-B1 and/or EX-B-a under COND-B2 are repeated and, furthermore, CASE-A occurs in each of the branches initiated at such repeated executions and each search in the branch initiated at EX-A terminates unsuccessfully. Therefore, we see, from Lemma, the execution of the algorithm takes $O(n^3)$ time in the worst case. Obviously the space complexity of the lists for U-PE's and candidate paths is $O(n)$. An easy way to store these lists in the case of branching is to copy them and use the copy for the search in the branch. Note that in CASE-B-1 or CASE- B-2 the search in the branch where no more branching occurs is executed first. Thus the copy requires $O(n)$ space only.

If the faulty PE's are distributed randomly in P, the probability of COND-B1 or COND-B2 holds is 0.5. Then the probability that worst case stated in the proof of Theorem is extremely small if n is large, and we can say the time complexity of the algorithm is $O(n^2)$ for practical use.

3.6 Example

Suppose that the distribution of the faulty PE's after the repeated applications of SEL-1 is as given in Fig. 7(for simplicity neither the U-PE's for which compensation paths have been determined by the applications of SEL-1 nor candidate paths which have been deleted are shown). The lists for the remaining U-PE's and candidate paths are given in the form of an array(not as linked-lists for easier presentation) in Table 1. The rows denoted by LR and DU show U-PE's in LR-list and DU-list respectively, and the rows denoted by KL, KR, KD and KU show, by 1, the existence of the KL-path, KR-path, KD-path and KU-path respectively of the U-PE in the same column. In the following UL=k(resp. -k) means that UL-PE is k and has (resp. does not have) the KL-path. Similar notations apply to UR-PE, UD-PE and UU-PE.

(1) UL=-1, UR=-15, UD=-14, UU=-2. Branching.

(1.A) PE 15 has the KL-path. CASE-A. This candidate path is selected as the compensation path for PE 15. The search proceeds to the branch initiated by this selection.

(1.A.1) UL=-1, UR=14, UD=-14, UU=-2. The KR-path of PE 14 is selected as the compensation path for the U-PE.

(1.A.2) UL=-1, UR=-13, UD=1, UU=-2. By the applications of SEL-1, SEL-3 and SEL-4, the KD-path of PE 1, the KL-paths of PE 6 and PE 7 are selected as the compensation paths.

(1.A.3) UL=-3, UR=-13, UD=-11, UU=-6. By the repeated applications of SEL-2, the KD-paths of PE 3, PE 4 and PE 13 are selected as the compensation paths.

(1.A.4) UL=-5, UR=-11, UD=-11, UU=-5. PE 11 has no candidate path, and thus the search in this branch terminates unsuccessfully. Back to (1).

(1.B) CASE-B-1. Branching.

(1.B.1) EX-B-a. The search initiated by EX-B-a terminates unsuccessfully. Back to (1.B).

(1.B.2) EX-B-b: The KD-path of PE 1, the KR-path of PE 14, the KU-path of PE 15 and the KL-path of PE 2 are chosen as the compensation paths for these PE's.

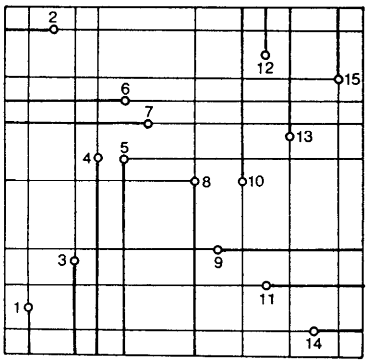

Fig. 7 Example.

Table 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
KL	0	1	0	0	0	1	1	1	1	0	1	0	0	1	1
KR	0	1	0	0	1	1	1	0	1	0	1	0	0	1	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DU	14	1	11	3	9	8	10	4	5	13	7	6	15	12	2
KD	0	1	0	1	0	1	1	1	1	1	0	0	1	0	0
KU	0	1	0	1	0	1	1	1	0	1	0	0	1	1	0

(1.B.2.1) UL=-3, UR=-13, UD=-11, UU=12. The KU-path of PE 12 is chosen as the compensation path.

(2)(after 1.B.2.1) UL=-3, UR=-13, UD=-11, UU=-1. CASE-B-3. Branching.

(2.B.1) EX-B-a. The KU-path of PE 3, the KL-path of PE 11, the KD-path of PE 13 and the KR-path of PE 6 are chosen as the compensation paths.

(2.B.1.1) UL=-4, UR=-10, UD=-9, UU=-7. PE 10 has no candidate path. The search in this branch terminates unsuccessfully. Back to (2).

(2.B.2) EX-B-b. The KD-path of PE 3, the KR-path of PE 11, the KU-path of PE 13 and the KL-path of PE 6 are chosen as the compensation paths.

(2.B.2.1) UL=-4, UR=-10, UD=-9, UU=-7. The KL-path of PE 7, the KD-path of PE 4, the KR-path of PE 9, the KU-path of PE 10, and the KD-path of PE 5 are chosen as the compensation paths by the applications of SEL-1 and/or SEL-2.

(2.B.2.2) UL=-8, UR=-8, UD=8, UU=8. KD-path of PE 8 is chosen as the compensation path. The search in this branch ends successfully. The compensation paths obtained are shown in Fig. 7 by the thick lines.

4. CONCLUDING REMARKS

In this paper we have given an efficient polynomial-order algorithm for the compensation path finding problem which Kung et al. reduced to an NP complete problem. The algorithm takes advantage of the special feature of the problem, that is, the compensation paths should lie on a rectangular grid graph without crossing each other. We expect that the path finding techniques using the queues and the branching scheme are applicable to similar problems to be solved on a plane. Deleting COND1, we can perform, a complementary search of compensation paths for the U-PE's whose compensation path could not be found by the algorithm presented in this paper.

REFERENCES

(1) T. Leighton and C.E Leiserson: "Wafer-scale integration of systolic arrays," IEEE Trans. Comp. C-34, 5, pp.448-461, May, 1985.

(2) W.R. Moore: "A review of fault-tolerant techniques for the enhancement of integrated circuit yield," Proc. IEEE, vol. 74, pp.684-698, May 1986.

(3) M. Sami and R. Stefanelli: "Reconfigurable architectures for VLSI processing arrays," ibid, pp.712-722, May 1986.

(4) S.-Y. Kuo and W.K. Fuchs: "Efficient spare allocation for reconfigurable arrays," IEEE Design and Test of Comp. vol.4, 1, pp.24-31, Feb. 1987.

(5) S.K. Tewksbury: Wafer-Level Integrated Systems, Kluwer Academic Publishers, Boston, 1989.

(6) S.-Y Kung, S.-N. Jean and C.-W. Chang, "Fault-tolerant array processors using single-track switches," IEEE Trans. Comp. C-38, 4, pp .501-514, April 1989.

(7) C. Bron and J. Kerbosch: "Algorithm 457-finding all cliques of an undirected graphs," Comm. ACM, vol.16, 9, pp.575-577, 1973.