Loop-Based Design and Reconfiguration of Wafer-Scale Linear Arrays with High Harvest Rates

Ming-Feng Chang and W. Kent Fuchs, *Senior Member, IEEE*

Abstract—**Design and reconfiguration approaches for high harvest rates and parallel on-wafer diagnosis of linear arrays are described in this paper. The defect-tolerant designs employ multiplexors to switch intercell connections and guarantee that the wire length between any two logically adjacent cells is constant, independent of fault distribution. The designs are appropriate for implementing linear arrays of wafer-scale memory and processor architectures. The harvesting of fault-free cells into a linear array is a *percolation process*; there exists a critical cell yield such that the harvest rate drops to zero (approaches 100%) if the cell yield is below (above) the critical value. Finding a maximum-size linear array for a given set of fault-free cells is polynomial time solvable if only the interconnections between fault-free cells are utilized, but is NP-complete if the interconnections between all cells are utilized. A heuristic reconfiguration algorithm utilizing the interconnections between all cells is presented. Application of boundary scan to parallel testing and on-wafer diagnosis of the arrays is described.**

I. Introduction

ONE OF THE practical applications of wafer-scale integration has been the implementation of linear array architectures for electronic disks and large systolic arrays. For example, a defect-tolerant linear array architecture, originally proposed by Manning [1], has been used by Anamartic to manufacture 200 megabytes of memory [2]. Their implementation is a rectilinear grid of cells on a wafer, in which each cell interconnects with its four nearest neighbors. The target structure is a linear array in which each cell connects to two neighboring cells. The defect-tolerant design allows each cell to bypass defective cells by connecting to any two of its four nearest neighbors. The process of switching interconnection to avoid defective cells (reconfiguration) starts from a cell on the boundary of a wafer, and then links defect-free cells into a chain (linear array). The chain is formed as a *spiral* from the wafer boundary to the center in an attempt to link as many good cells as possible. Aubusson and Catt independently employed a similar design philosophy for a hexagonal array in which each cell has six immediate neighbors [3]. Each cell in their design is capable of connecting to two of its six neighbors. Since only the nearest neighbors are interconnected, the propagation delay between any two logically consecutive cells after reconfiguration is fixed in these spiral designs.

Recently, Horst presented an alternative defect-tolerant loop WSI architecture that provided a simple reconfiguration procedure [4]. In his design, each cell includes four 2-to-1 multiplexors for intercell connections, as shown in Fig. 1. Initially, each cell and its four multiplexors are set to form a closed loop. During reconfiguration, the multiplexors can be set such that two or more neighboring cells form a larger loop. Similar to the spiral approaches, the propagation delay between two logically consecutive cells after reconfiguration is fixed, independent of defect distribution. This is because the path between two logically adjacent cells in a loop has exactly four multiplexor delays. The goal of reconfiguration in this loop-based design is to link as many good cells as possible into a loop. *Harvest rate*, the percentage of good cells that are linked in the largest loop, is used to evaluate the effectiveness of defect-tolerant designs for linear arrays. One advantage of the loop-based approach to defect-tolerant linear arrays is that each cell can merge as many as four neighboring cells into its loop, instead of just two neighbors in the spiral approaches. Thus, the loop-based approach has a better harvest rate for a given cell yield. However, one of the shortcomings of this design is that the harvest rate drops dramatically when the cell yield is below 0.75 [4].

There also exist defect-tolerant linear array structures, other than the spiral and loop-based approaches, that try to harvest as many good cells as possible using a constant upper bound on propagation delay between two logically adjacent cells. Fussel and Varman presented an approach to connecting a linear array from all good cells in a cluster [5]. Leighton and Leiserson developed a scheme to con-

Manuscript received September 20, 1990; revised January 10, 1991. This work was supported in part by the SDIO/IST and managed by Naval Research under Contract N00014-89-0070 and in part by the Semiconductor Research Corporation under Contract 89-DP-109. A brief version of this paper was presented at the IEEE International Conference on Wafer Scale Integration, January 1991.

M.-F. Chang was with the Center for Reliable and High-Performance Computing, Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801. He is now with the Department of Computer Science and Information Engineering, Chiao-Tung University, Taiwan, Republic of China.

W. K. Fuchs is with the Center for Reliable and High-Performance Computing, Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801.

IEEE Log Number 9143195.

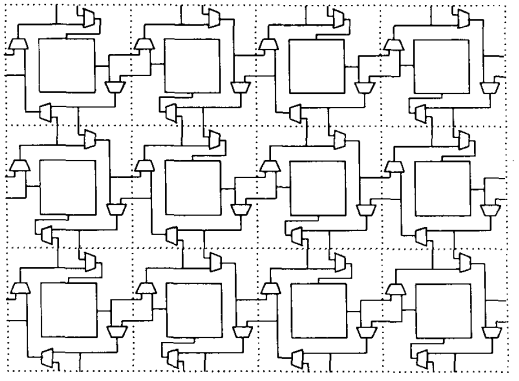


Fig. 1. Loop-based design for defect-tolerant linear arrays.

nect a linear array from all good cells on a wafer with wire of length $O(\sqrt{\log N})$, where N is the number of total cells [6]. Greene and El Gamal presented another scheme that can connect any constant fraction of the good cells on a wafer with a constant upper bound on wire length [7]. A survey of defect-tolerant linear arrays can be found in the book by Negrini *et al.* [8]. These approaches differ from the approach of this paper in that their delay is bounded, while ours is fixed.

Our discussion is restricted to the design and analysis of defect-tolerant linear arrays that guarantee a fixed intercell wire length. We show how the defect-tolerant loop-based approach developed by Horst [4] can be modified to provide high harvest rates. The defect-tolerant designs described allow the interconnection of up to eight neighbors, and maintain constant propagation delay between cells. The harvesting of fault-free cells into a linear array is a percolation process; there exists a critical cell yield such that the harvest rate drops to zero (approaches 100%) if the cell yield is below (above) the critical value. The designs with a larger number of neighbor interconnections reduce the critical cell yield and enhance the harvest rate. A further improvement to harvest rates through reconfiguration strategies that make use of the multiplexors in faulty cells is also described.

II. Defect-Tolerant Linear Arrays

In this section we describe the design approach for enhancing the harvest of good cells into a linear array based on loop interconnections. The cells are placed as a two-dimensional grid on a wafer. Each cell includes l multiplexors; the multiplexors are placed along the cell boundaries, and are used to switch intercell connection buses. Each cell and its multiplexors form a closed loop, as shown in Fig. 2(a). The multiplexors of each cell are labeled such that m_d denotes a multiplexor whose inputs need to propagate through d multiplexors to reach the cell in the closed loop. The interconnection between two neighboring cells is through a pair of multiplexors, one from each cell, with the same subscript. Each multiplexor sets up an interconnection with one neighbor; a cell with l

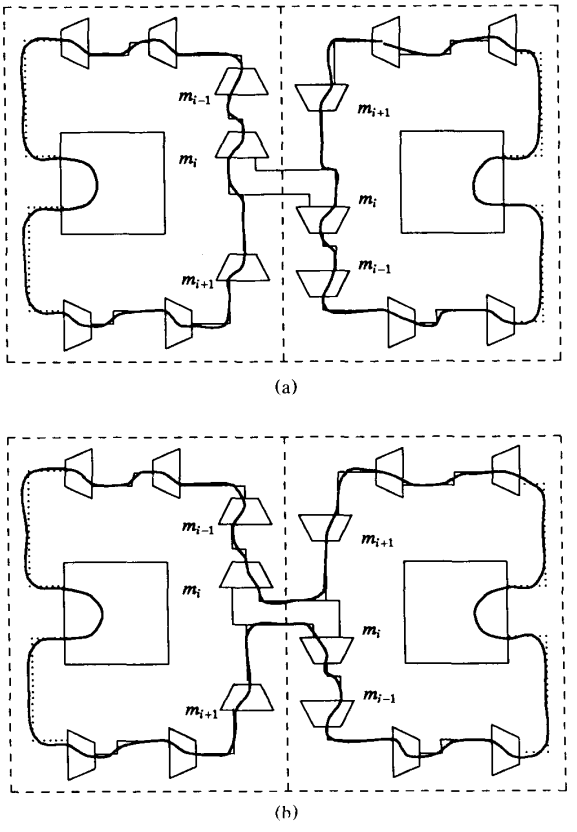


Fig. 2. The interconnection between two neighboring cells: (a) inactive, and (b) active.

multiplexors can have interconnections with l immediate neighbors. Two example interconnections between two neighboring cells are shown in Fig. 2. The interconnection is *active* if the pair of multiplexors selects the input source from their neighboring cell as their outputs, as shown in Fig. 2(b). Otherwise, the interconnection is *inactive*, as shown in Fig. 2(a). Two neighboring cells can be linked in a loop by setting the interconnection between them active. An important feature of the design is that the path between two logically adjacent cells in a loop contains exactly l multiplexors, and the multiplexors are linked in the sequence $m_l, m_{l-1}, \cdots, m_1$. Therefore, the propagation delay between two logically adjacent cells is fixed, if the propagation delay between two consecutive multiplexors (m_{i+1}, m_i) is constant for each i. The design rules are summarized as follows.

Rule #1: Each cell and its l multiplexors $(m_l, m_{l-1}, \cdots, m_1)$ form a closed loop. The interconnection between two neighboring cells is through a pair of multiplexors with the same subscript.

Rule #2: The wire length between any two consecutive multiplexors (m_{i+1}, m_i) is fixed for each i. □

Design rule #1 ensures that the multiplexors between any pair of consecutive cells in a loop is fixed $(m_i, m_{l-1}, \cdots, m_1)$. Design rule #2 ensures the propaga-

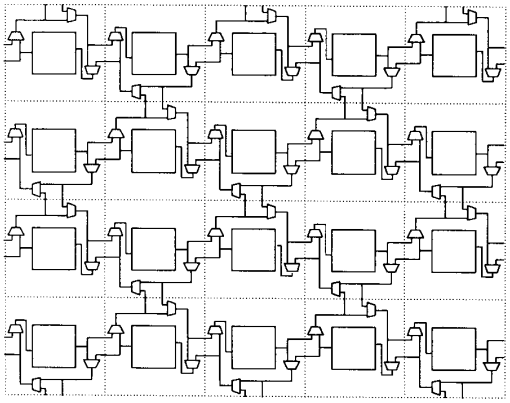


Fig. 3. An example three-neighbor loop.

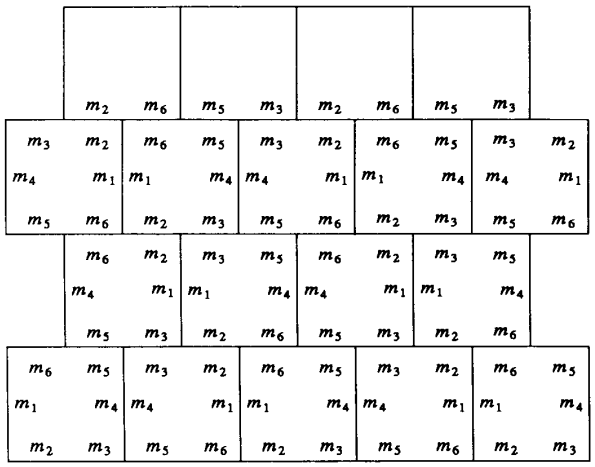


Fig. 4. A six-neighbor loop design violating the fixed wire length rule.

tion delay between any pair of consecutive multiplexors is constant. As a result, the design guarantees a constant propagation delay between any two logically adjacent cells, independent of fault distribution.

Horst's loop-based defect-tolerant architecture follows these design rules and uses four 2-to-1 multiplexors in each cell [4]. Since each cell has interconnections with its four neighbors, Horst's scheme is an example of four-neighbor loops. As a simple variation of four-neighbor loops, a defect-tolerant linear array with three-neighbor interconnections can be obtained by deleting all the m_4 multiplexors, as shown in Fig. 3. A three-neighbor reconfigurable linear array will be shown in Section IV to have a smaller harvest rate than a four-neighbor design, but it uses less interconnection area due to the elimination of multiplexors.

The harvest rate can be improved by increasing intercell connections at the cost of more interconnection overhead and more multiplexors. By skewing the placement of every row of cells by half a cell width, each cell will have six immediate neighbors instead of four, as shown in Fig. 4. However, a straightforward extension of the four-neighbor design, using six 2-to-1 multiplexors to interconnect with six neighbors, does not satisfy the fixed wire length rule. In Fig. 4, every pair of consecutive multiplexors (m_i, m_{i-1}) is physically adjacent for each cell on the second and fourth rows, but (m_6, m_5), (m_4, m_3), and (m_3, m_2) of each cell on the third row are not physically adjacent. The wire length between these logically consecutive, but physically disjoint, multiplexors is longer than the wire length between other consecutive multiplexors. As a result, the propagation delay between two logically adjacent cells after reconfiguration may not be constant.

In order to satisfy the fixed wire length rule for six-neighbor loops, 3-to-1 multiplexors can be used for intercell connections. Each cell uses three 3-to-1 multiplexors, and each multiplexor sets up interconnections with two neighboring cells. A defect-tolerant linear array with six-neighbor interconnections is shown in Fig. 5. It can be verified that the interconnections satisfy the two design rules and hence the propagation delay between two logically adjacent cells is constant.

A reconfigurable linear array with eight-neighbor interconnections and fixed intercell delay can be made by each cell having interconnections not only with the four border-adjacent neighbors, but also with the four neighbors in diagonal directions. There are two possible approaches: one uses only 2-to-1 multiplexors, and the other uses 4-to-1 and 2-to-1 multiplexors. The design with eight 2-to-1 multiplexors in each cell is essentially a simple combination of two 4-neighbor loops; an example is shown in Fig. 6. The design with two 4-to-1 and two 2-to-1 multiplexors in each cell is shown in Fig. 7. By using 4-to-1 multiplexors, the total number of control lines of the multiplexors in each cell is reduced to six, instead of eight if eight 2-to-1 multiplexors are used. The increased neighboring interconnections require a larger number of multiplexors in each cell, and thus lower the cell yield. However, the overall harvest rate will improve because the cells are less likely to be isolated with increased interconnections.

III. Reconfiguration

A. Problem Description

The reconfiguration problem consists of harvesting as many good (defect-free) cells as possible in a linear array. For the loop-based design, a straightforward reconfiguration approach is to use the interconnections between good cells and link all neighboring good cells into a linear array [4]. Therefore, each cluster of good cells forms a linear array. An example reconfiguration for three-neighbor loops is shown in Fig. 8(a). The three linear arrays constructed are of maximal size in the sense that the neighboring cells of each linear array are all faulty.

It is possible to improve the rate of harvesting good cells into a single linear array by utilizing the interconnections between all cells, including the fault-free multiplex-

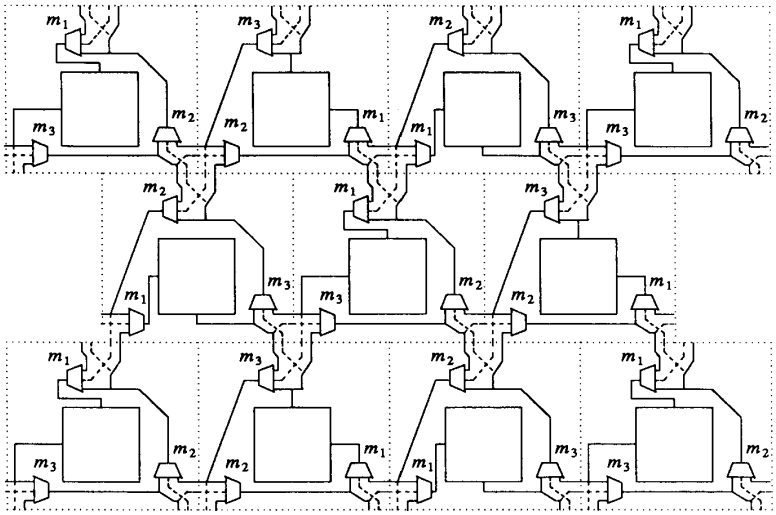


Fig. 5. A six-neighbor loop-based design with fixed wire length.

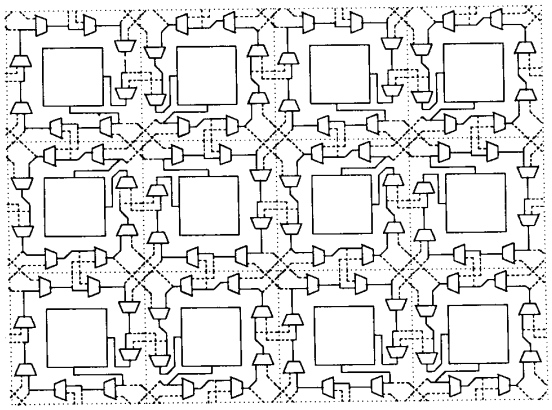


Fig. 6. An example of eight-neighbor loops.

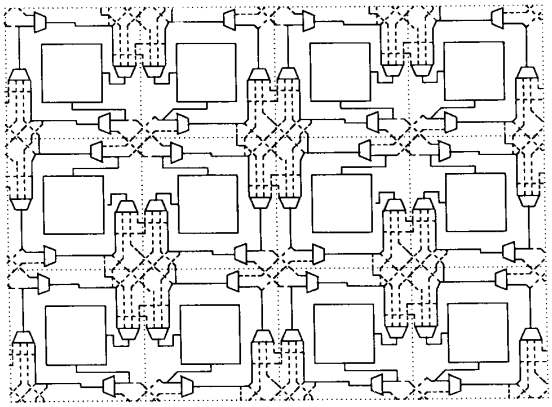
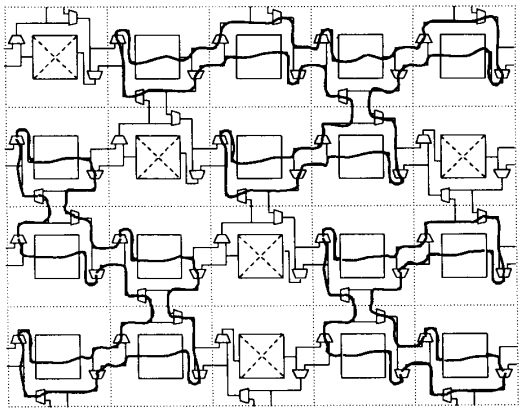
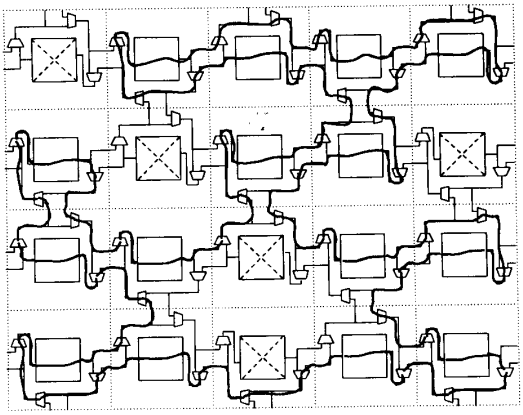


Fig. 7. An alternative eight-neighbor loop design.



(a)



(b)

Fig. 8. Reconfiguration solutions for three-neighbor loops: (a) utilizing interconnections between good cells only, and (b) utilizing interconnections between all cells.

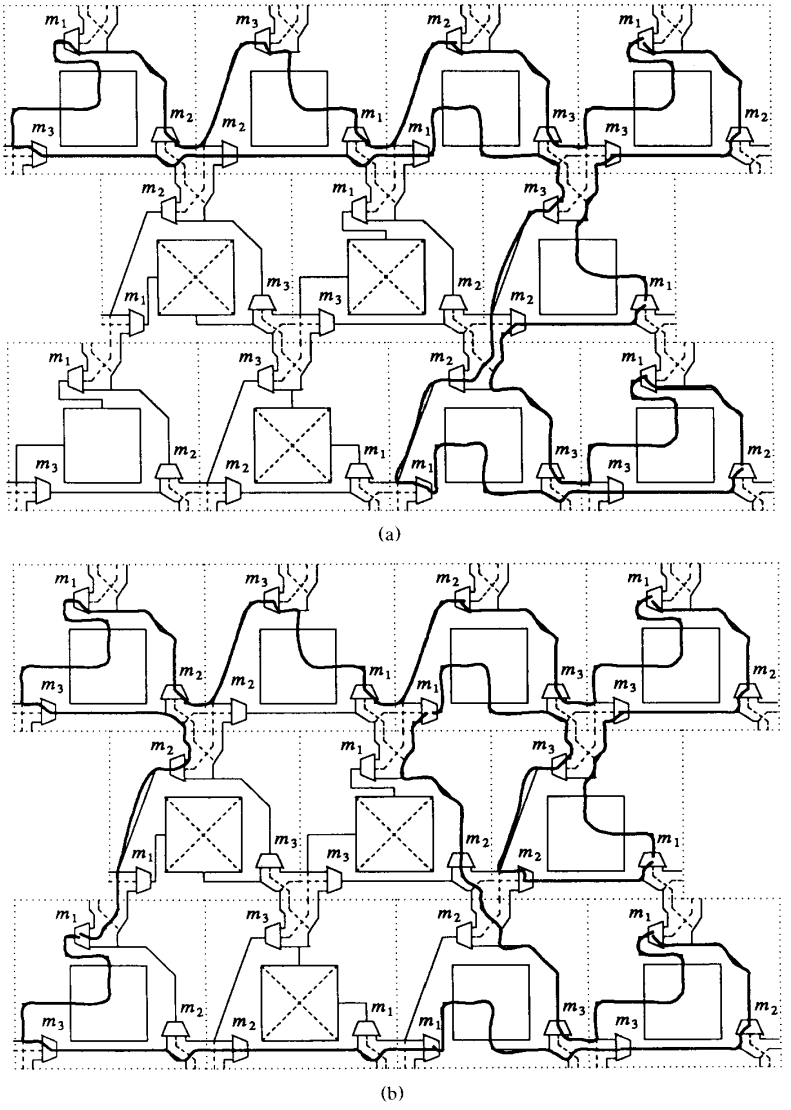


Fig. 9. Reconfiguration solutions for six-neighbor loops: (a) utilizing interconnections between good cells only, and (b) utilizing interconnections between all cells.

ors of defective cells. Fig. 8(b) shows a reconfiguration solution that links all the good cells in a single linear array.

Another set of example reconfigurations for six-neighbor interconnections is shown in Fig. 9. A reconfiguration solution that uses only the multiplexors of good cells is shown in Fig. 9(a). The solution cannot connect the isolated cell in the bottom left corner. By using the multiplexors of all the cells, an optimal reconfiguration solution is able to connect all the good cells in a loop, as shown in Fig. 9(b).

The complexity of the reconfiguration problem is analyzed in the following sections under two different assumptions: one in which only the interconnections between good cells can be used, and the other in which all the interconnections can be used.

B. Graph Model

A graph model is introduced as a basis for analyzing the reconfiguration problems. Only the reconfiguration for four-neighbor loops will be examined, but the results apply also to three-, six-, and eight-neighbor loop reconfigurations.

A rectilinear grid graph consists of vertices which are points at the Cartesian coordinates with integer coordinates, and edges between any two vertices at unit distance. An m-by-n rectilinear grid graph $R(m,n)$ is a

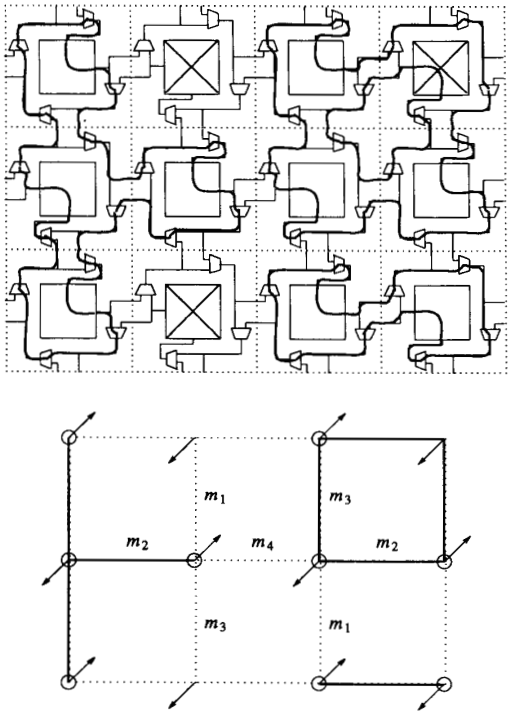


Fig. 10. An example reconfiguration and its grid graph model.

graph (V, E), where $V = \{(x, y) | 1 \leqslant x \leqslant m,\ 1 \leqslant y \leqslant n\}$, and $\{v_i, v_j\} \in E$ if and only if the distance between v_i and v_j is 1.

A wafer of m-by-n cells with four-neighbor loops can be modeled by $R(m, n)$ with some modifications. Each vertex represents a cell. Each edge represents the interconnection between two neighboring cells containing a pair of multiplexors. A vertex is fault-free (faulty) if its corresponding cell is fault-free (faulty). A circle is used to denote a fault-free vertex. An arrow is also used in each vertex to show the location of the cell, which is between m_4 and m_1. The location of each cell is important for the discussion later. An edge is active (inactive) if the corresponding interconnection is active (inactive). An active edge is drawn in a solid line, and an inactive edge in a dotted line. $R'(m, n)$ or R', if its size is not specified, refers to the modified grid graph. An example interconnection setting and its corresponding grid graph model $R'(4,3)$ are shown in Fig. 10.

For purposes of reconfiguration, consider the active edges only and discard all the inactive edges. One can verify the following correspondences between abstract objects in R' and physical cell loops on the wafer:

1) an isolated vertex in R' represents a loop that contains only the cell of the vertex;
2) a single (active) edge in R' represents a loop that contains the cells of the two incident vertices of the edge;

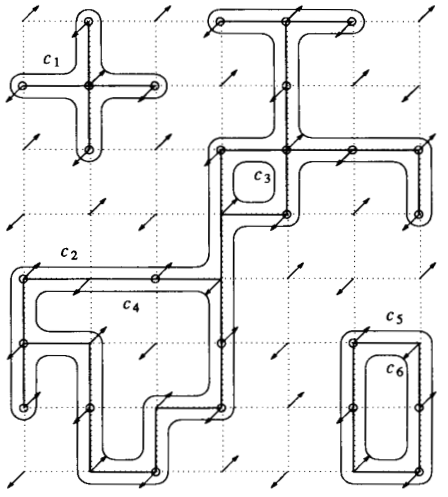


Fig. 11. An example reconfiguration that forms six contours.

3) a tree in R' represents a loop that contains the cells of all vertices on the tree;
4) a simple cycle in R' represents two loops: one consists of all the cells whose vertices are on the cycle and whose arrows point outside of the cycle, and the other consists of the cells of all the other vertices on the cycle.

Assume that the grid graph is embedded on an infinite plane. In general, the active edges may divide the plane into *regions*. *Contours* are the closed lines along the boundaries of all regions. An example reconfiguration solution is shown in Fig. 11; there are four regions and six contours. The contours are labeled as $c_1, c_2, \cdots, c_6$. The region that is unbounded is the *outside region*. The contours that are on the boundaries of the outside region are called outside contours; otherwise, they are inside contours. It can be verified that each contour represents a loop that consists of all the cells (vertices) whose arrows point toward the contour. Thus, besides the isolated vertices, six loops are constructed by reconfiguration in Fig. 11. The size of a contour will be considered to be the number of cells it contains. A contour is fault-free if all of its cells are fault-free. For example, the largest fault-free contour in Fig. 11 is c_2 of size 18.

C. Reconfiguration Complexity

For the spiral approach with four neighbors, each cell can only connect to two of its neighbors. The reconfiguration problem is to find a maximum length path in the grid subgraph induced by good vertices. This maximum-length path problem has been shown to be NP-complete [9].

The reconfiguration complexity for four-neighbor loops is considered under two reconfiguration assumptions. First, suppose that only the multiplexors of good cells can be properly controlled. In this case, only the interconnections between good cells can be active, i.e., only the edges between good vertices in R' can be active. Therefore, only

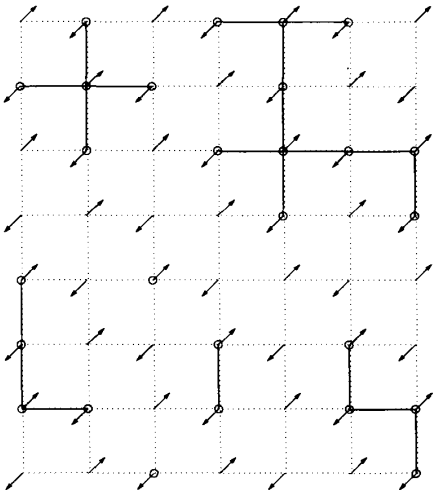


Fig. 12. An example maximum fault-free tree of size 10.

good cells can be linked in a loop of size >1. It is undesirable to generate a cycle in R', because a cycle has an inside contour(s), and hence reduces the size of the outside contour. The reconfiguration problem is to find a maximum-size spanning tree in the grid subgraph induced by good vertices. Clearly, the problem can be solved in linear time [4].

Second, consider the case where the multiplexors of faulty cells can be utilized to set up interconnections. In this case, all the interconnections can be active, i.e., all edges in R' can be active. The reconfiguration problem is to find a fault-free contour of maximum size. The problem will be shown in the following discussion to be NP-complete.

Clearly, the size of the maximum fault-free path is at most equal to the size of the maximum fault-free spanning tree, which in turn is at most equal to the size of the maximum fault-free contour. The examples in Figs. 11 and 12 have the same fault-free cells. Fig. 12 shows that the largest fault-free spanning tree contains ten vertices, while Fig. 11 shows that there exists a fault-free contour of size 18. One can also easily verify that the maximum fault-free path contains only seven vertices.

Finding a maximum-size fault-free contour for a given fault distribution R' is NP-complete. The first lemma was proved by Dolev *et al.* [10].

Lemma 1: For any planar graph G of maximum degree 3, there exist a constant $c>0$ and a planar embedding of G on a rectilinear grid such that the longest edge is of length cn, where n is the number of vertices of G. □

The following lemma shows that all edges can be of the same length.

Lemma 2: For any planar graph G of maximum degree 3, there is a planar embedding of G on a rectilinear grid such that all edges are of the same length $2(cn)^2$, where c is the constant in Lemma 1 and n is the number of vertices of G.

Proof: From Lemma 1, there is a planar embedding $\psi(G,R_1)$ of G on rectilinear grid R_1. A grid point (x,y) is even, if $x+y\equiv 0 \pmod 2$. Without loss of generality, assume all vertices of G are mapped into even grid points, so that the length of any edge is an even integer.

Replace each cell square of R_1 by a grid of size $2cn\times 2cn$. Call the new grid R_2. For any edge e of length $l(e)$ in $\psi(G,R_1)$, we can change the length of e to any even integer between $l(e)\cdot 2cn$ and $l(e)\cdot 2(cn)^2$ in the embedding $\psi(G,R_2)$, because each unit length line in R_1 is enlarged $2cn$ times and has at least $2(cn)^2$ space to extend. Thus, the shortest possible edge, of length 2 in $\psi(G,R_1)$, can be of a length from $4cn$ to $4(cn)^2$ in $\psi(G,R_2)$, and the longest possible edge, of length cn in $\psi(G,R_1)$, can be of a length from $2(cn)^2$ to $2(cn)^3$ in $\psi(G,R_2)$. Therefore, all edges can be of the same length $2(cn)^2$. □

Theorem: The problem of determining whether a given fault distribution of $R'(m,n)$ contains a fault-free contour of size $\geqslant k$ is NP-complete.

Proof: The problem is in NP, since one can arbitrarily make a subset of edges active and check in polynomial time whether a fault-free contour of size $\geqslant k$ exists or not.

In the remainder of the proof, we show a reduction from the following NP-complete problem [11]:

Instance: Planar digraph D with indegree(v) + outdegree$(v)\leqslant 3$ for all vertex $v\in D$.

Question: Does D have a Hamiltonian cycle?

From Lemma 1, we can find an embedding $\psi(D,R_2)$ of D on a rectilinear grid such that all edges are of the same length $2(cn)^2$. Construct an embedding $\psi(D,R_3)$ by replacing each cell square of R_2 by a grid of size 4×4. All arcs are enlarged four times and are of the same length $8(cn)^2$ in $\psi(D,R_3)$. An instance of fault distribution R'_3 can be built from R_3 as follows:

1) all the vertices, in which vertices of D are embedded, are fault-free;
2) all the vertices, whose cells lie on the right-hand side of an arc, are fault-free;
3) all other vertices are faulty.

An *arc-embedding path* is the whole path in which an arc of D is embedded. Each arc-embedding path in R'_3 consists of $4(cn)^2-1\pm 1$ fault-free vertices, and all the fault-free vertices are on the right-hand side of the arc. The remainder of the proof shows that graph D has a Hamiltonian cycle if and only if R'_3 has a fault-free contour of size $\geqslant 4n(cn)^2-2n$.

If D has a Hamiltonian cycle C, let all the edges of R'_3, in which the arcs of C are embedded, be active. All these active edges form a cycle in R'_3. Since all the vertices on the right-hand side of the arcs are fault-free, one contour of the cycle is fault-free and of size $\geqslant 4n(cn)^2-2n$.

The distance between two fault-free vertices is at least four, except for two fault-free vertices on an arc-embedding path. Thus, only these arc-embedding paths contain

all fault-free vertices on one side of the paths. If R_3' has a fault-free contour of size $\geq 4n(cn)^2 - 2n$, the contour must contain at least n arc-embedding paths. Since the degree of each vertex of D is at most 3 and each arc-embedding path contains fault-free vertices only on its right-hand side, the fault-free contour contains at most n arc-embedding paths and these paths must form a cycle. Therefore, the fault-free contour is a contour of a cycle which consists of exactly n arc-embedding paths, and the n corresponding arcs in D are a Hamiltonian cycle. □

In actual implementations, some multiplexors may also be faulty. By a similar argument, one can prove that finding a maximum size fault-free contour for a given set of faulty multiplexors is also NP-complete.

D. Heuristic Reconfiguration Algorithm

Since only the outside region is unbounded, the heuristic algorithm focuses on generating a fault-free contour on the boundary of the outside region. First, all fault-free spanning trees are constructed. Disjoint fault-free spanning trees can be connected by cycles that have fault-free outside contours, and the connected component still has a fault-free outside contour. Therefore, we find all the cycles that have fault-free outside contours and can connect at least two disjoint spanning trees. However, it is possible that a group of cycles may generate inside contours that contain good cells. Moreover, some cycles need to be included before we can use other cycles, i.e., there exist precedence constraints (a partial-order relation) in including these cycles. The following algorithm decides which cycle should be included in the connected component:

Algorithm

1: Form all spanning trees consisting of good cells only.
2: Find all possible cycles that can connect two separated spanning trees.
3: Find the partial-order relation between the cycles found in step 2.
4: While there are still cycles to choose
  4.1: While there are cycles that connect only a single cluster and have no successor.
    4.1.1: Delete the cycles found in step 4.1.
    4.1.2: Include the cycles that cannot cause any inside contour containing good vertices and have all preceding cycles included in some component.
    Merge connected components.
  4.2: Choose one cycle with all preceding cycles included.
  Include the cycle and merge the connected components.
5: Report the largest contour that is built in step 4. □

IV. Harvest Rate and Yield

Computer simulations have been used to estimate harvest rate with respect to the yield of the cells. The distribution of faulty cells is assumed to be random. First, consider the reconfiguration strategy that utilizes the multiplexors of good cells only. The harvest rates of a 256-cell wafer, in a 16×16 layout, for alternative defect-tolerant loop-based designs are shown in Fig. 13. In each case, the harvest rate is the average of 10 000 different simulation samples. The results show that the harvest rate improves consistently as the cell yield increases. The harvest rate also improves consistently with larger numbers of intercell connections. This is because both higher cell yield and more intercell connections generate a larger cluster of good cells. The harvest rate of a large array, a wafer with 1024×1024 cells, was also estimated by simulations. Each harvest rate in Fig. 14 is the average result of 1000 different simulation samples. The results show that there is a rapid transition in the harvest rate, i.e., the harvest rate drops to zero, when the cell yield is below a critical value.

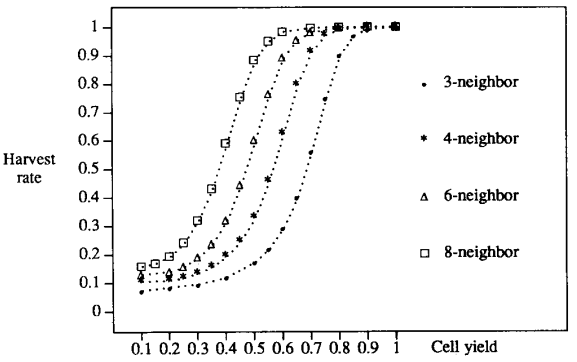


Fig. 13. The expected harvest for a wafer with 16×16 cells using multiplexors of only good cells.

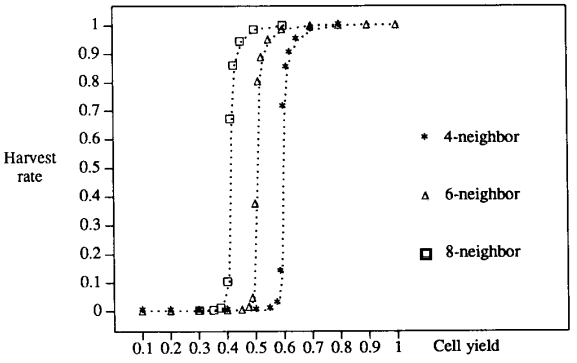


Fig. 14. The expected harvest for a wafer with 1024×1024 cells using multiplexors of only good cells.

The harvest of good cells into a linear array in a loop-based design is an example of a well-known physical process—the percolation process [12]. A percolation process is a random process over an infinite regular lattice of cells in which a fluid starts from a cell and percolates in all directions to neighboring cells. A flawed cell will not allow the passage of the flow. Let q be the probability of each cell being flawed. There exists a critical probability

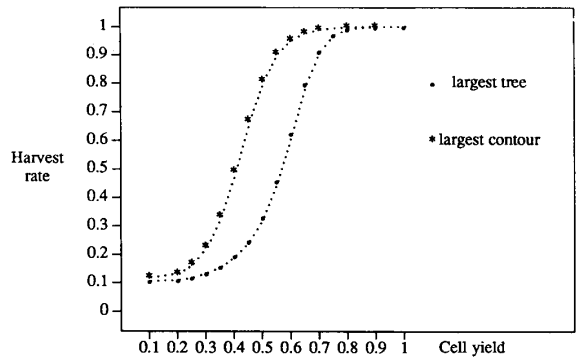


Fig. 15. The effect of using all multiplexors (16×16 cells with four-neighbor interconnections).

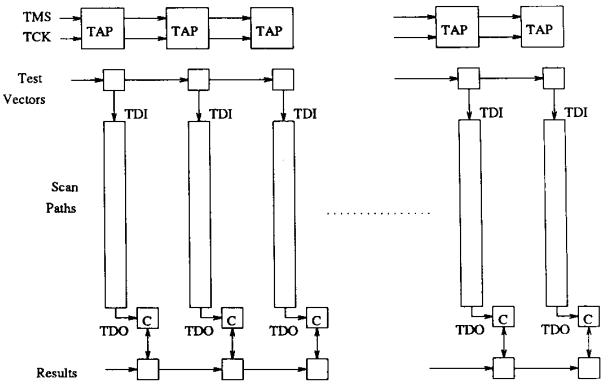


Fig. 16. Parallel on-wafer diagnosis.

q_c, such that if $q < q_c$ the probability of the number of wet cells being infinite is equal to 1. Agrawal has pointed out that the harvest rate of the spiral approaches is bounded by the fraction of good cells that are *wet* in a percolation process [13]. It is clear that the harvest rate of a loop-based design is equal to the percentage of good cells that are *wet* in a percolation process. For the four-neighbor case, $q_c \approx 0.42$ requires cell yields higher than 0.58. The critical cell yield for the six-neighbor interconnections is equal to 0.5 (exact) [14], and the critical cell yield for the eight-neighbor interconnections is approximately 0.40.

The optimal harvest can be achieved by utilizing all the interconnections; however, the reconfiguration problem when all interconnections are allowed is NP-complete. The harvest capability of the heuristic algorithm in Section III was evaluated by simulations. The expected harvest rate for 16×16 cells with four-neighbor interconnections and using all the multiplexors is shown in Fig. 15. One thousand simulation samples are repeated for each case. The results show that by utilizing all the multiplexors, the harvest rate is improved and the critical value of cell yield is reduced significantly.

V. Parallel On-Wafer Diagnosis

In many defect-tolerant designs an effective strategy for diagnosing defective cells is often ignored. One approach to on-wafer diagnosis, which employs both pipelining and parallelism, is illustrated in Fig. 16. Each basic cell in the scan array implements the JTAG boundary scan standard [15]. The test access ports (TAP's) of the cells are linked into a linear chain, and the testing is done in a pipelined fashion. The array cells are connected in parallel scan paths. Since the cells are identical in function, it is possible to perform parallel clocking of test vectors as well as pipelined comparison of test outputs for diagnosis.

The parallel diagnosis requires three extra linear chains for test-mode select, (TMS) test data input (TDI), and test data output (TDO). These linear chains for testing must be defect-tolerant, because any defect in these chains makes some of the cells untestable. Therefore, the loop-based defect-tolerant design is also used for these testing chains. The external bonding pads of the wafer can be placed in any cell on the wafer. The loop interconnections of the cell are replaced by input and output (I/O) pads. The multiplexors are controlled by its cell. At the power-up, each cell controls its multiplexors so that they form a closed loop. During the wafer test, the tester selects a set of I/O pads connecting to a cell first. Then from the first connected cell, the connected cells set up active interconnections with the unconnected neighbors only so that no cycles (in R') are generated. After the parallel on-wafer diagnosis, the external tester pipes in the multiplexor controls for each cell so that a fault-free linear array is connected.

VI. Conclusions

A loop-based approach to defect-tolerant wafer-scale linear arrays was presented. This approach is a generalization of the four-neighbor design of Horst [4]. In terms of harvest rate, the loop-based approach is significantly better than the traditional spiral approaches. Similar to the spiral designs, the loop-based approach guarantees a fixed propagation delay between any logically consecutive cells after reconfiguration, independent of the fault distribution. However, the propagation delay (latency) is larger than that of the spiral designs. Two reconfiguration strategies were also presented: one utilizes the interconnections between good cells only; the other utilizes the interconnections between all cells. Finding a maximum size linear array for a given set of faulty cells was shown to be NP-complete when all the interconnections are utilized. The simulation-based harvest rates for four loop-based designs were described; the results illustrated the improvement in harvest rate when the neighborhood connectivity is enlarged at the cost of more multiplexors and interconnections. Application of boundary-scan design to parallel testing and on-wafer diagnosis of the arrays was also presented.

ACKNOWLEDGMENT

The authors wish to acknowledge the discussions and contributions of R. Horst, J. Patel, and W. Shi in the development of this paper.

REFERENCES

[1] F. B. Manning, "An approach to highly integrated, computer-maintained cellular arrays," *IEEE Trans. Comput.*, vol. C-26, pp. 536–552, June 1977.
[2] F. Baba and A. Sinclair, "200-Mb wafer scale memory," in *Proc. IEEE Int. Conf. Wafer Scale Integration*, Jan. 1990, pp. 5–12.
[3] R. C. Aubusson and I. Catt, "Wafer-scale integration—A fault-tolerant procedure," *IEEE J. Solid-State Circuits*, vol. SC-13, pp. 339–344, June 1978.
[4] R. W. Horst, "A linear-array WSI architecture for improved yield and performance," in *Proc. IEEE Int. Conf. Wafer Scale Integration*, Jan. 1990, pp. 85–91.
[5] D. Fussel and P. Varman, "Fault-tolerant wafer scale architecture for VLSI," in *Proc. 9th Annual Symp. Comput. Architecture*, 1982, pp. 190–198.
[6] T. Leighton and C. E. Leiserson, "Wafer-scale integration of systolic arrays," *IEEE Trans. Comput.*, vol. C-34, pp. 448–461, May 1985.
[7] J. W. Greene and A. El Gamal, "Configuration of VLSI arrays in the presence of defects," *J. Ass. Comput. Mach.*, vol. 31, pp. 694–717, Oct. 1984.
[8] R. Negrini, M. G. Sami, and R. Stefanelli, *Fault Tolerance Through Reconfiguration in VLSI and WSI Arrays*. Cambridge, MA: MIT Press, 1989.
[9] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, "Hamilton paths in grid graphs," *SIAM J. Computing*, vol. 11, pp. 676–686, Nov. 1982.
[10] D. Dolov, T. Leighton, and H. Trickey, "Planar embedding of planar graphs," *Advances Computing Res.*, vol. 2, pp. 147–161, 1984.
[11] J. Plesnik, "The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two," *Inform. Processing Lett.*, vol. 8, pp. 199–201, Apr. 1979.
[12] J. C. Wierman, "Percolation theory," *Annals Probability*, vol. 10, pp. 509–524, 1982.
[13] V. D. Agrawal, "Comments on 'An approach to highly integrated computer-maintained cellular arrays'," *IEEE Trans. Comput.*, vol. C-28, p. 691–693, Sept. 1979.
[14] M. F. Sykes and J. W. Essam, "Exact critical percolation probabilities for site and bond problems in two dimensions," *J. Math. Phys.*, vol. 5, pp. 1117–1127, Aug. 1964.
[15] *Standard Test Access Port and Boundary-Scan Architecture*, IEEE Standard 1149.1–1990.

Ming-Feng Chang received the B.S. and M.S. degrees in electrical engineering from National Taiwan University in 1982 and 1984, respectively, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 1991.

He was an engineer with AOC Corporation, Taiwan, from 1984 to 1986. While pursuing his Ph.D. studies at the University of Illinois, he was partly supported by the Ministry of Education, Republic of China, from 1986 to 1988, and he held a Research Assistantship at the Coordinated Science Laboratory from 1987 to 1990. He is currently an Associate Professor in the Department of Computer Science and Information Engineering, Chiao-Tung University, Taiwan, Republic of China. His research interests include CAD, testing, and fault tolerance of VLSI systems.

W. Kent Fuchs (S'80–M'85–SM'90) received the B.S.E. degree in electrical engineering from Duke University, Durham, NC, in 1977 and the M.S. degree in electrical engineering from the University of Illinois, Urbana, in 1982. In 1984 he received the M.Div. degree from Trinity Evangelical Divinity School in Deerfield, IL, and in 1985 he received the Ph.D. degree in electrical engineering from the University of Illinois.

He is currently an Associate Professor in the Departments of Electrical and Computer Engineering, Computer Science, and the Coordinated Science Laboratory, University of Illinois. He joined the University of Illinois as an Assistant Professor in 1985 and was promoted to Associate Professor in 1989. His research interests include parallel computing and VLSI system design with emphasis on reliable computation.

Dr. Fuchs is the Guest Editor of the May 1992 Special Issue of IEEE TRANSACTIONS ON COMPUTERS on Fault-Tolerant Computing, and Co-Guest Editor of the April 1992 Special Issue of *IEEE Computer* on Wafer-Scale Integration Architectures and Algorithms. He was Technical Program Chairman of the 1989 IEEE International Workshop on Hardware Fault Tolerance in Multiprocessor Architectures. Research awards include appointment as Fellow in the Center for Advanced Studies, University of Illinois 1990, the Xerox Faculty Award for Excellence in Research 1987, College of Engineering, University of Illinois, the Digital Equipment Corporation Incentives for Excellence Faculty Award 1986–1988, the Best Paper Award, IEEE/ACM Design Automation Conference (DAC) 1986, simulation and test category, and nomination for the Best Paper Award, DAC 1987, simulation and test category.