# SPARE ALLOCATION AND RECONFIGURATION IN LARGE AREA VLSI

*Sy-Yen Kuo and W. Kent Fuchs*

Computer Systems Group
Coordinated Science Laboratory
University of Illinois
Urbana, IL

*Abstract-* One approach to enhancing the yield of large area VLSI is through design for yield enhancement by means of restructurable interconnect, logic and computational elements. Although extensive literature exists concerning architectural design for inclusion of spares and restructuring mechanisms in memories and processor arrays, little research has been published on optimal spare allocation and reconfiguration in the presence of multiple defects. In this paper, a summary of a systematic approach developed by the authors for spare allocation and reconfiguration is presented. Spare allocation is modeled in graph theoretic terms in which spare allocation for a specific reconfigurable system is shown to be equivalent to either a graph matching or a graph dominating set problem. The complexity of optimal spare allocation for each of the problem classes is analyzed in this paper and reconfiguration algorithms are provided.

## I. INTRODUCTION

Design for yield enhancement in large VLSI chips and wafer scale integration by means of spare interconnect, logic and computational elements and appropriate restructuring mechanisms has been studied extensively [1]. However, very little has been published regarding effective and general algorithms for spare allocation and reconfiguration in the presence of multiple defects, except for a few simple reconfiguration algorithms appropriate for memory arrays and two dimensional arrays with spare rows and columns of cells [2-4]. Since the number of spares will always be limited and manufacturing time devoted to reconfiguration is costly, efficient spare allocation and reconfiguration algorithms are needed to enhance yield. Simple greedy or exhaustive search algorithms are neither effective nor efficient for problems even as simple as spare allocation for a rectangular array of memory cells, which has been shown to be NP-complete [2].

In this paper, spare allocation problems for reconfigurable structures are modeled in graph theoretic terms. Spare allocation for a specific reconfigurable system is shown to be equivalent to either a graph matching or a graph covering problem. The complexity of optimal spare allocation for each of the problem classes is analyzed and reconfiguration algorithms are provided.

## II. PROBLEM DESCRIPTION

A reconfigurable system (chip or wafer) is represented in this paper as an undirected graph where nodes (vertices) in the graph represent the basic replaceable units, which can be as simple as memory cells or as complex as processors, and edges in the graph represent interconnection links between replaceable units. Each node (replaceable unit) and each edge (interconnection link) can be in one of three states: active, faulty, or spare. Two strategies typically used to replace a faulty node by a spare are direct replacement and shifted replacement . In direct replacement, each faulty node is substituted by a spare node without requiring further substitutions. In shifted replacement, each faulty node is replaced by one of its nonfaulty adjacent nodes, and this replacing node is again replaced by one of its adjacent nodes, and so on, until a spare node is incorporated into the structure.

A *target graph* (*architecture*), $G_t$, is a graph representing the desired system structure with all its nodes and edges in active states. A *host graph* (*architecture*), $G_h$, is a graph representing an interconnection structure in which each node and edge can be active, faulty, or spare, and which contains a subgraph isomorphic to a target graph $G_t$. *Reconfiguration* is the process of modifying the mapping of the target graph onto the host graph to compensate for the presence of faults within the host graph. An optimal reconfiguration occurs when spares are allocated for faults such that a system is successfully reconfigured and the associated cost is a minimum. Reconfiguration cost in actual designs is usually a function of the number of spares used, interconnection, system performance degradation in the reconfigured system, the time to find a reconfiguration solution, and the time spent in actually reconfiguring the chip or wafer. The problem addressed in this paper concerns efficient allocation of spares and reconfiguration.

Section III describes graph theoretic models for spare allocation. Section IV(A) discusses complexity and algorithms for spare allocation. Heuristic reconfiguration algorithms are presented in Section IV(B).

## III. GRAPH THEORETIC SPARE ALLOCATION MODELS

In the following, an *element* indicates either a replaceable unit or a replaceable block of units. Existing reconfigurable designs for large area VLSI and WSI can in general be classified into the following categories.

(1) Single replacement - only one faulty unit is replaced by each spare allocated.
   a. Single option - only one particular spare unit can replace the faulty unit.
   b. Multiple options - any one out of several spare units can replace the faulty unit.

(2) Block replacement - a block of units is replaced by each spare allocated.
   a. Single option - the faulty unit is replaceable by only one type of block.
   b. Multiple options - the faulty unit is replaceable by several types of blocks.

The basic graph models for the (1) single replacement and (2) block replacement strategies are different as discussed below.

### *A. Single Replacement*

(1) Single option (*SRSO*)

In this class, each spare unit replaces one faulty unit and each faulty unit has only one choice to select a spare unit. A host graph can be partitioned into blocks of units with each block including a spare unit as well as spare links such that any faulty unit in this block can be replaced by this particular spare unit only. Figure 1 shows an example *SRSO* scheme as developed by the authors for reconfigurable cube-connected cycles

*Acknowledgement*: This research was supported by the Semiconductor Research Corporation under Contract 87-DP-109.

S.-Y. Kuo is now with the Dept. of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona.

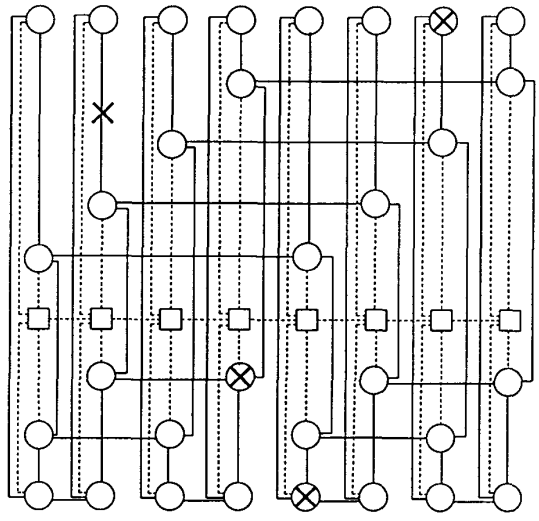

Figure 1. Example *SRSO* scheme with 3 faulty processors and a faulty interconnect bus.

VLSI architectures [5]. Each column is a cycle of four processors with one spare processor represented by a box and associated redundant interconnections. The spare processor has spare links represented by dashed lines to every other processor unit in this cycle. Also shown are three faulty processors and one faulty interconnect link.

The relationship between faulty and spare units is represented by a bipartite graph model. A *bipartite graph* is a graph whose node set can be partitioned into two sets, $A$ and $B$, where each edge has one node in $A$ and one node in $B$. It is denoted by $BG=(A, B, E)$ where $E$ is the set of edges. Let nodes in $A$ represent faulty units and nodes in $B$ represent spare units. An edge exists from a node $\alpha$ in $A$ to a node $\beta$ in $B$ if the faulty unit represented by $\alpha$ can be replaced by the spare unit represented by $\beta$ and the reconfiguration path from $\alpha$ to $\beta$ in $G_A$ contain no faulty link. The degree of each node in $A$ is at most one and the degree of each node in $B$ can be greater than one in the *SRSO* model. Figure 2(a) represents the scenario in Figure 1 with one spare unit left unconnected to indicate some spare units are not utilized. If there is a one-to-one relationship between faults and spares as in Figure 2(a), the system can be reconfigurable. Otherwise, the system is not reconfigurable as in Figure 2(b) in which two nodes in $A$ are connected to the same node in $B$.

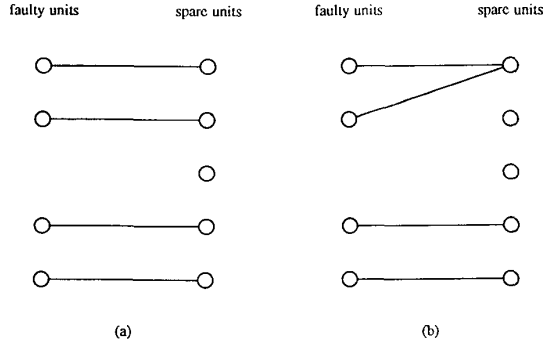


Figure 2. Example bipartite graph models of spare allocation for *SRSO*.

(2) Multiple options (*SRMO*)

In this case, each spare unit still replaces one faulty unit, but there is more than one choice of a spare to replace a fault. An example is the reconfigurable mesh with three faulty units and one faulty link as shown in Figure 3 [6]. The circles in Figure 3 represent spare units while the squares are the units in the target graph. Each faulty unit, as indicated by "•", has either one or two choices to be replaced by a spare unit. The faulty link is indicated by "X".

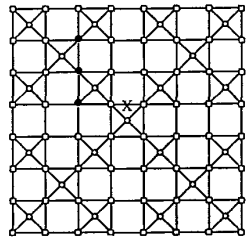

Figure 3. An example *SRMO* scheme with three faulty units and one faulty link.

Figure 4(a) shows the bipartite graph model for the example in Figure 3. This graph is a many-to-many relationship. The degree of any node in $BG$ can be greater than one. The spare allocation problem now becomes a matching problem. A *matching* $M$ of a graph $G=(V,E)$ is a subset of the edges $E$ with the property that no two edges of $M$ share the same node in $V$. The set of bold edges in Figure 4(b) shows a matching for Figure 4(a). The number of spare units must be greater than or equal to the number of faulty units, otherwise the system is not reconfigurable. A matching must be found such that every faulty unit is assigned a spare. This bipartite graph matching problem is known to be solvable in polynomial time [7].

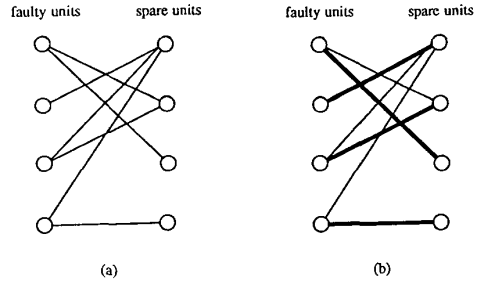


Figure 4. An example bipartite graph model of SRMO.

### *B. Block Replacement*

(1) Single option (*BRSO*)

In block replacement, instead of replacing a single faulty unit by a single spare, a block of units (faulty and nonfaulty) is replaced by a block of spare units. Each faulty unit belongs to only one type of block which means there is only one option to select a spare to cover this faulty unit. An example is shown in Figure 5 which is an array with spare columns of units. Also shown in Figure 5 are six faulty units.

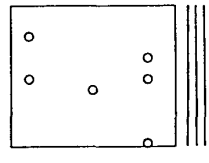

Figure 5. An example scheme of *BRSO*.

A different type of bipartite graph $BG=(A, B, E)$ is used here to describe this problem. Nodes in $A$ still represent all the faulty replaceable units, but nodes in $B$ now represent different replaceable blocks. An edge exists between two nodes $\alpha$ and $\beta$ if the faulty unit denoted by $\alpha$ in $A$ is in the block denoted by $\beta$ in $B$ and the reconfiguration path in $G_h$ from $\beta$ to the available spare block contains no faulty or unavailable link. An example graph model for Figure 5 is shown in Figure 6. The degree of each node in $A$ is at most one since there is only one type of block in the *BRSO* class. The spare allocation is simple in that nodes in $B$ are assigned spare blocks until all the faulty units are replaced or no spares are left.

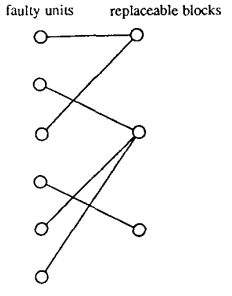


Figure 6. Bipartite graph model of *BRSO*.

(2) Multiple options (*BRMO*)

In this class, more than one type of block exists and each faulty unit can belong to several sets of different types of blocks. There are multiple options to select different types of spare blocks to cover a faulty unit and the allocation of a spare block can cover more than one faulty unit. Hierarchically reconfigurable architectures typically fall into this category [6]. An example *BRMO* scheme with 9 faulty cells is shown in Figure 7. Each rectangle represents an array of cells. For every faulty cell, there are several options to select either a spare row or a column to replace the row or column which contains the faulty cell. A bipartite graph, $BG=(A, B, E)$, is used to represent the spare allocation problem. Nodes in $A$ denote the faulty replaceable units in the system and nodes in $B$ denote blocks of different types. An edge exists between two nodes $\alpha$ and $\beta$ if the faulty unit denoted by $\alpha$ in $A$ is in the block denoted by $\beta$ in $B$ and the reconfiguration path in $G_h$ from $\beta$ to the available spare block contains no faulty link. The nodes in $B$ are divided into sets with each set containing a single type of block, i.e., the nodes in the same set are replaced by the same spare block types. The bipartite model of *BRMO* with 9 types of blocks for the example in Figure 7 is shown in Figure 8. In this model for multiple types of blocks, spare allocation becomes a restricted dominating set problem with constraints [8].

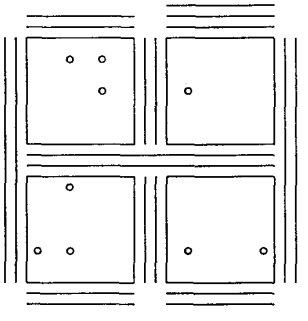

Figure 7. An example *BRMO* scheme.

**DEFINITION :** A *dominating set* of a graph $G=(V,E)$ with |V| vertices and |E| edges is a subset $V' < V$ such that for all $u \in V - V'$ there is a $v \in V'$ for which $\{u,v\} \in E$. □

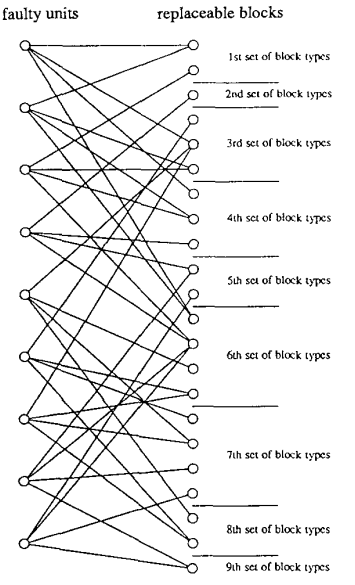


Figure 8. An example bipartite graph model of *BRMO*.

In the model, all the faulty units are on one side of the bipartite graph, and the blocks containing these faulty units are on the other side. The goal is to replace some of the blocks with spare blocks so that all the faulty units are replaced with spare units. Hence the problem can be stated as finding a set of nodes in $B$ which covers all the nodes in $A$ subject to the constraints that the number of nodes chosen in each block can not exceed the number of spares available in each block.

## IV. SPARE ALLOCATION AND RECONFIGURATION

In the following we discuss the complexity of spare allocation and associated algorithms for each of the previous replacement classes.

### *A. Spare allocation*

(1) SRSO and BRSO

For *SRSO*, we first check the bipartite model $BG=(A, B, E)$ to see if any node in $B$ is connected to two or more nodes in $A$. If this is the case, then the reconfiguration structure is declared as unreconfigurable, else if the number of nodes in $A$ is greater than the number of spare units, the structure is still not reconfigurable. Otherwise, each spare node is allocated to the connected faulty node in $BG$.

For *BRSO*, a spare block is allocated to each node in $B$ in the bipartite representation until all the faulty units are covered, otherwise the structure is declared unreconfigurable if no spare block is left and faulty units still remain. For both SRSO and BRSO, the complexity of spare allocation is $O(F)$ where $F$ is the number of faulty units in $G_t$.

(2) SRMO

Spare allocation in this class is a bipartite maximum matching problem. The complexity of the algorithm for the bipartite matching is $O(|V|^{1/2} \cdot |E|)$ [7]. If each edge is assigned a cost, the problem becomes weighted bipartite matching where a matching with minimum cost is to be found. A weighted bipartite graph can be always made complete by adding nodes and edges to make the two sets of nodes equal in size and all the added edges having a cost larger than any cost in the original graph. For a complete bipartite graph $BG=(A, B, E)$, let $c_{ij}$ denote the cost of edge $[v_i, v_j]$ where $v_i \in A$ and $v_j \in B$. The weighted bipartite matching problem is formulated as follows:

$$\min \sum_{i,j} c_{ij}x_{ij}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i=1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j=1, \ldots, n$$

$$x_{ij} \geq 0$$

$x_{ij}$ is a set of variables for $i=1,...,n$ and $j=1,...,n$, where $n$ is the number of nodes in $A$ or $B$. Here $x_{ij}$ means that the edge $[v_i, v_j]$ is included in the matching, whereas with $x_{ij} = 0$ the edge is not.

For the weighted matching problem, the complexity is $O(|V|^3)$ for a complete bipartite graph with 2|V| nodes by using the Hungarian Method [7]. Before applying the matching algorithm, a check can first be made as to whether the number of spare units is greater than or equal to the number of faulty units, if not, then the system is not reconfigurable. Otherwise, the bipartite weighted matching algorithm is applied to obtain the matching $M$ with minimum cost. If all the faulty units are covered by $M$, then $M$ is the solution which has a one-to-one relationship between all the faulty units and spare units, otherwise the system is not reconfigurable.

(3) BRMO

For the restricted case with only two types of blocks, such as for reconfigurable memory arrays with spare rows and columns, spare allocation is a constrained bipartite vertex covering problem. The complexity of this problem has been shown by the authors to be NP-complete [2]. For the general *BRMO* model, with more than two sets of block types, spare allocation is a restricted bipartite dominating set problem with constraints. Vertex covering for the case of two types of blocks is a special case of this problem. The complexity of this new problem is again NP-complete which is shown in the following theorem.

**THEOREM :** Given a bipartite graph $BG = (A, B, E)$ with $B$ partitioned into $n$ sets, $B_1, B_2, \ldots, B_n$, and $n$ positive integers $x_1 \leq |B_1|, x_2 \leq |B_2|, \ldots, x_n \leq |B_n|$. the problem of determining if there is a set $D_1 \cup D_2 \cup \cdots \cup D_n$, which covers all the nodes in $A$, and $D_1 \subseteq B_1, D_2 \subseteq B_2, \ldots, D_n \subseteq B_n$ such that $|D_1| \leq x_1, |D_2| \leq x_2, \ldots,$ and $|D_n| \leq x_n$, is NP-complete.

**PROOF:** The problem is in NP because a feasible subset of $B$ can be checked in polynomial time that it indeed covers all the nodes in $A$ and satisfies all the constraints. For $n = 2$, this problem is equivalent to the problem in the above Lemma, hence it is NP-complete. To prove that the general case is also NP-complete, we use the technique of proof by restriction [8] , which says that if a problem is in NP and contains a NP-complete problem as a special case then this problem is also NP-complete. It has been shown that this general problem is in NP and contains an NP-complete special case, therefore the theorem follows. □

The weighted version of the problem in the above theorem is still NP-complete since the unweighted problem is a special case of the weighted version by letting all the costs be one.

If the number of spares is small, an exhaustive search to find an applicable allocation may be applicable. For the case of a reconfigurable design with two types of blocks, a branch-and-bound with early-abort algorithm as well as an approximation algorithm was previously developed by the authors [2]. In a design with $n$ types of blocks, the early-abort technique does not apply. However, with some modification, the branch-and-bound technique can still be utilized for moderate size problems. For large problem sizes the authors have developed a heuristic dominating set algorithm [9].

### *B. Reconfiguration implementation*

Following spare allocation, node renaming and interconnection rerouting are performed to restore $G_t$. Reconfiguration is discussed for direct replacement and shifted replacement strategies.

(1) Direct replacement

Since each faulty element has either direct spare links or can be directly replaced by the assigned spare element during the spare allocation process, the reconfiguration is completed once the spare allocation is done. The reconfiguration algorithm is straightforward by rerouting the interconnections such that those originally connected to the faulty elements are connected to the replacing spare elements.

(2) Shifted replacement

First a renaming along each reconfiguration path is performed and then interconnection links are reorganized to accommodate the faults and recover $G_t$. However, in some cases we have found that due to switching limitation or resource contention the reconfiguration-controlling circuits may not be able to establish a link in $G_h$ between two elements which are connected in $G_t$. Two heuristics have been implemented to resolve this situation. The first heuristic is to switch the order of reconfiguration paths. In some topologies the resulting configuration of the structure will depend on the order of executing the shifted replacement on different reconfiguration paths. The second heuristic is to assume some non-faulty elements in $G_h$ to be unavailable for inclusion in $G_t$. An active element may be excluded from the system to make the two unconnectable elements change their relative positions. Due to the limited presentation space available for this paper, the reader is referred to [9] for a complete description of the reconfiguration algorithms.

## V. SUMMARY

A summary of a systematic approach to spare allocation and reconfiguration in large area VLSI was presented in this paper. Graph theoretic models have been developed to describe the relationship between faults and spares. By using the graph models, spare allocation becomes either a matching problem or a dominating set problem. The complexity of the problems were analyzed, and spare allocation algorithms were summarized. Reconfiguration algorithms including renaming and rerouting have been developed which utilize heuristics to handle conflicts and contention during the reconfiguration process.


## ACKNOWLEDGEMENT

The authors wish to express their thanks to Ken Kubiak for his assistance in developing the software implementing the spare allocation and reconfiguration algorithms of this paper.


## REFERENCES


[1] W. R. Moore, "A Review of Fault-Tolerant Techniques for the Enhancement of Integrated Circuit Yield," *Proc. of the IEEE*, vol. 74, pp. 684-698, May 1986.

[2] S. Y. Kuo and W. K. Fuchs, "Efficient Spare Allocation for Reconfigurable Arrays," *IEEE Design and Test*, vol. 4, pp. 24-31, Feb. 1987.

[3] S. Y. Kuo and W. K. Fuchs, "Fault diagnosis and Spare Allocation for Yield Enhancement in Large Reconfigurable PLAs," *Proc. 1987 Int'l Test Conf.*, Sept. 1987.

[4] R. W. Haddad and A. T. Dahbura, "Increased Throughput for the Testing and Repair of RAMs with Redundancy," *Proc. Int'l Conf. on Computer-Aided Design*, pp. 230-233, Nov. 1987.

[5] P. Banerjee, S. Y. Kuo, and W. K. Fuchs, "Reconfigurable Cube-Connected Cycles Architectures," *Proc. 16th Int'l Symp. on Fault-tolerant computing*, pp. 286-291, July 1986.

[6] W. K. Fuchs, M. F. Chang, S. Y. Kuo, P. Mazumder, and C. Stunkel, "The Impact of Parallel Architecture Granularity on Yield," in *Yield Modeling and Defect Tolerance in VLSI*, ed., Moore, Maly, Strojwas . Adam Hilger: London, 1988.

[7] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization, Algorithms and Complexity*. New Jersey: Prentice-Hall, Inc., 1982.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability*. New York: W. H. Freeman and Company, 1979.

[9] S. Y. Kuo, "Design for Yield Enhancement and Reconfiguration in Large-Area VLSI/WSI Architectures," *CSG Report #86*, Coordinated Science Laboratory, Ph.D. Thesis, University of Illinois, Urbana-Champaign, Feb., 1988.