# A Defect Tolerance Scheme for Nanotechnology Circuits

S. Ramsundar
Department of
Computer Science and Engineering
Indian Institute of Technology
Guwahati, Assam - 781039
Email: sundark@iitg.ernet.in

Ahmad A. Al-Yamani
Center for Reliable Computing
Stanford University
Stanford, CA
Email: alyamani@crc.stanford.edu

Dhiraj K. Pradhan
Department of Computer Science
University of Bristol
Merchant Venturers Building, Woodland Road
Bristol BS8 1UB UK
Email: pradhan@cs.bris.ac.uk

***Abstract*— Lithography based IC fabrication is rapidly approaching its limit in terms of feature size. The current alternative is nanotechnology based fabrication, which relies on self-assembly of nanotubes or nanowires. Such a process is subject to a high defect rate, which can be tolerated using carefully crafted defect tolerance techniques. This paper presents an algorithm for reconfiguration-based defect tolerance in nanotechnology switches. The algorithm offers an average switch density improvement of $50\%$ to $100\%$ to most recently published techniques. The algorithm is also very consistent in improving the yield through minimizing false rejects. The improvement percentage varies depending on the manufactured switch size and the desired defect-free size.** [1]

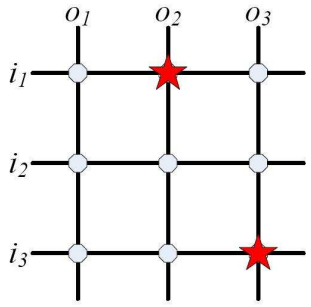


Fig. 1. $3 \times 3$ nano-scale crossbar with 2 defects

## I. Background

As lithography-based silicon VLSI technology is expected to hit its limit in terms of feature size, nanotechnology-based fabrication is emerging as an alternative. It is estimated that molecular electronics can achieve very high densities ($10^{12}$ deivces per $cm^2$) and operate at very high frequencies (of the order of THz)[1]. Several successful nano-scale electronic devices have been demonstrated by researchers, some of the most promising being carbon naotubes (CNT)[2], silicon nanowires (NW)[3][4], and quantum dot cells[5].

It has been shown that using self-assembly techniques, we can overcome the limitations posed by lithography for the smallest feature size[1]. Due to fabrication regularity imposed by the self assembly process, only regular structures such as 2-D crossbars can be built. Several architectures based on crossbars are under investigation. The crossbar plays the key role in these architectures, as it can be used as PLA planes (for programmable logic) and interconnects.

Hewlett-Packard has recently fabricated 8 X 8 crossbar switches using molecular switches at the crosspoints[6]. They observed that only $85\%$ of the switches were programmable while the other $15\%$ were defective. With such a high defect rate, fault-tolerance methods have to be devised for the emerging nanotechnology devices. Various research groups are working on problems related to defect tolerance in nanotechnology [7]-[14].

[1]Research Reported Supported by EPSRC (UK).

A simple calculation shows that even a moderate size crossbar with $15\%$ defect rate would have at least one defect on each of its nanowires. Discarding all parts with any defects in them is not affordable any more as the yield hit will be substantial. So, we have to bypass these defects to make computation possible on these structures. Given a defective crossbar of size $n \times n$, we need to find out a defect-free subset of size $k \times k$. Now, given a defective crossbar, we need to maximize $k$ to use it to its maximum potential. This problem is explored in this paper.

The rest of the paper is organized as follows. Section II introduces the preliminaries for the algorithms to be discussed later. The problem and its intractability are discussed in Section III. Section IV reviews the previous approach. A new algorithm is proposed in Section V, and its results depicted in Section VI. Finally, Section VII concludes the paper.

## II. Preliminaries

A $3 \times 3$ crossbar is shown in Fig.1. Given a 2D cross bar with faults, it can be represented by a bipartite graph. We call this graph the *representative graph* of the 2D crossbar. Formally:

*Definition 2.1:* Given a $a \times b$ crossbar switch with faults $F = \{(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)\}$. A *representative graph* is an undirected bipartite graph $G(U, V, E)$; i.e., with partitions $U$ and $V$, having $|U| = a$ and $|V| = b$, with a bijection between the set of horizontal nanowires and $U$, and between vertical nanowires and $V$. $E$ consists of representative edges

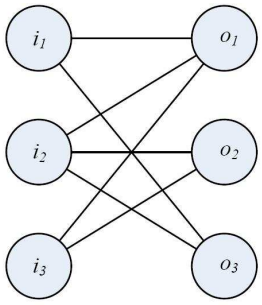


Fig. 2. The representative graph of crossbar in Fig.1

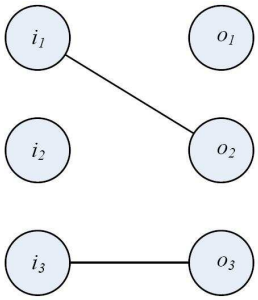


Fig. 3. The bipartite complement of graph in Fig.2

for all the non-defective cross points of the grid. Hence, $|E| = ab - n$.

Fig.2 is the representative graph of the crossbar in Fig.1.

*Definition 2.2:* A bipartite graph $G(U, V, E)$ is said to be a *biclique* if $E = U \times V$; i.e., $E$ consists of all the possible edges in the bipartite graph. The set of all the possible edges is represented by $K_{|U|,|V|}$. Hence, for a biclique $E = K_{|U|,|V|}$.

A biclique $G(U, V, E)$ is said to be a balanced one if $|U| = |V|$.

*Definition 2.3:* Given an undirected bipartite graph $G(U, V, E)$, the graph $\bar{G}(U, V, \bar{E})$ is said to be its *bipartite complement* if $\bar{E} = K_{|U|,|V|} - E$.

Fig.3 shows the bipartite complement of the graph in Fig.2.

*Definition 2.4:* Given an undirected bipartite graph $G = (U, V, E)$, a subset $I \subset U \cup V$ is said to be an independent set on G if for any two nodes $x, y \in I$, $(x, y) \notin E$.

## III. The Problem

To avoid substantial yield hits caused by high defect rates, defect tolerance schemes can be applied to nanotechnology crossbar switches in such a way that an $n \times n$ switch with some defects can be used as a $k \times k$ switch, where $k < n$. Maximizing k for a defect switch leads to higher logic densities on the same defective circuit.

The problem of finding the maximum $k \times k$ defect-free crossbar from a $n \times n$ crossbar corresponds to finding the maximum balanced biclique in its representative graph[12]. This problem is NP complete. This problem is called Balanced Complete Biparite Subgraph [16]. Recently, it has been shown to be NP hard even to approximate within a constant factor [18].

This is a common problem addressed in Operations Research [20], Algorithms [19], and now in nanotechnology.

If we remove the restriction that the biclique should be balanced, the problem is solvable in polynomial time [15]-[17]. However, many applications require balanced crossbars. Our first approach was to get an approximation algorithm based on this polynomial time algorithm. But, the results were poor because the bicliques generated by the polynomial time algorithm were highly unbalanced. So, we had to change the approach.

## IV. Previous Research

Various nano-wire and switch faults were studied in [7] and their impact on the routability of a crossbar was investigated. A crossbar with faults can still be used as a smaller crossbar with reduced functionality; i.e., a smaller defect-free crossbar. Simulation results for the imapact on the routability of the crossbar for various nano-wire and switch faults have been shown.

Another problem related to mapping on crossbars is dealt with in [8]. Given a defective crossbar with a set of functions to be implemented on it, a mapping of functions to the output wires is to be found. This problem is solvable in polynomial time using bipartite matching algorithms[17]. The running time of the algorithm is determined by the time required to construct the bipartite graph model of the crossbar. So, in [8], a greedy algorithm based on probabilistic methods is given to avoid the construction of the biparite graph model of the crossbar. Hence, the time complexity of the algorithm is reduced by trading off its accuracy.

### A. Defect-Unaware Design Flow

In [12], a novel defect-unaware design flow, for mass produciton of crossbar-based logic circuits, was proposed. The aim of the flow model was to design logic circuits on defective crossbar, say of size $n \times n$. The main idea in this defect-unaware flow model was to assume that there was a $k \times k$ crossbar ($k < n$), which is completely defect free and proceed with the logic and architecture design. Finally, mapping the design onto the defective crossbar. Hence, only the final step in this flow model is defect-aware. The entire model depends heavily on the statsistical data relating $n$, $k$ and the defect rate of the fabrication process. There are requirements, advantages and disadvantages of this flow model.

*1) Requirements:*

- A mapping algorithm for locating largest $k \times k$ defect-free crossbar in a $n \times n$ defective crossbar. This problem reduces to the maximum balanced bipartite subgraph problem in the repsentative graph of the $n \times n$ crossbar.
- If the defect-unaware flow requires a $p \times p$ defect-free crossbar and if the defect rate of the fabrication process is known to be $d\%$, the size of the defective crossbar, say $m \times m$, to be manufactured to gurantee that can fulfill the design requirements. Hence, value of $m$ will depend not only on the defect rate, but also on the efficiency of the proposed mapping algorithm.

- Statistical data for $m$; i.e., for an $m \times m$ crossbar with $d\%$ defect rate, the mapping algorithm should consistently produce crossbars of size not less than $p \times p$.

*2) Advantages:*

- In conventional design model, the logic and design steps are customised on per chip basis based on the defect map. But, in this flow model, its universal for all the chips manufactured, as all the chips are assumed to be built on a $k \times k$ defect-free crossbar ($k < n$).
- The conventional defect map is to enumerate all the defective crosspoints. Hence, the size of the conventional defect map is $O(n^2)$. While, the new defect map stores only the information as to which $k$ of the $n$ rows form the rows of the defect-free $k \times k$ crossbar and which $k$ of the $n$ columns form the columns of the defect-free $k \times k$ crossbar. Hence the size of the new defect map is $2k$, which can be utmost $2n$. Hence, the size of the new defect map is $O(n)$.

*3) Disadvantages:*

- Out of the $(n-k) \times (n-k)$ crosspoints which are being neglected, many of them may be working. These working crosspoints are not being used.

The exact algorithm for solving the maximum balanced bipartite subgraph problem as given in [12] takes exponential time. It becomes completely intractable for crossbars of size 64 and larger. So, in [12], comparison of exact algorithm and Algorithm 1 could be done only for $16 \times 16$ and $32 \times 32$ crossbars with defect densities up to $30\%$.

The algorithm given in [12] (Algorithm 1) constructs the bipartite complement of the representative graph, then tries to achieve the approximate maximum balanced independent set of the bipartite complement. For this, the heuristic given is to remove nodes with the highest degree. To keep the resulting bipartite graph almost balanced, the algorithm alternates between $U$ and $V$ while removing the nodes. Hence, the biclique returned from the algorithm has a maximum difference of 1 between the sizes of the two partitions. Notice that working with the complement graph means that the edges represent defective switches. So, the target is to achieve the largest edge-free subgraph possible.

First, the bipartite complement of the representative graph is constructed (line 1). The nodes in each partition are sorted according to their degrees in the complement graph in decreasing order (line 3-4). At each iteration, nodes with zero degree are added to the solution (lines 7-8). The heuristic is to alternate between $U$ and $V$, when removing the highest degree node in the partition, according to the flag value (lines 10-11, 17-18).

**Algorithm 1 Function** Biclique $(G(U, V, E))$[12]

1: Obtain $\bar{G}(U, V, \bar{E})$, $\bar{E} = K_{|U|,|V|} - E$
2: {Find the maximum independent set in $\bar{G}$}
3: Sort $U$ based on $d(u)$ in $\bar{G}$ (decreasing order)
4: Sort $V$ based on $d(v)$ in $\bar{G}$ (decreasing order)
5: $U^b \leftarrow \phi, V^b \leftarrow \phi, flag \leftarrow TRUE$
6: **repeat**
7: $U^b \leftarrow U^b \bigcup \{u | u \in U, d(u) = 0\}, U \leftarrow U - U^b$
8: $V^b \leftarrow V^b \bigcup \{v | v \in V, d(v) = 0\}, V \leftarrow V - V^b$
9: **if** flag **then**
10: $u \leftarrow$ node in $U$ with maximum degree
11: $U \leftarrow U - \{u\}$
12: **for** each $v' \in V$ such that $(u, v') \in \bar{E}$ **do**
13: $d(v') \leftarrow d(v') - 1$
14: **end for**
15: Re-sort $V$ accordingly
16: **else**
17: $v \leftarrow$ node in $V$ with maximum degree
18: $V \leftarrow V - \{v\}$
19: **for** each $u' \in U$ such that $(u', v) \in \bar{E}$ **do**
20: $d(u') \leftarrow d(u') - 1$
21: **end for**
22: Re-sort $U$ accordingly
23: **end if**
24: $flag \leftarrow \neg flag$
25: **until** $U = \phi$ or $V = \phi$
26: **return** $U^b \times V^b$ as the maximum biclique

## V. Proposed Reconfiguration Algorithm

We propose a greedy algorithm (Algorithm 2) for the maximum balanced bipartite subgraph problem. Our aim is to make the bipartite complement of the representative graph (the equivalent of the graph representing the faults) edge-free, by removing the same number of nodes from each of the partitions of the graph. The number of nodes removed should be as low as possible, for the algorithm to be close to optimal. Initially, we tried to approximate this problem by solving the unbalanced version, which is solvable in polynomial time. We solved the unbalanced version using the Hungarian algorithm [21] which as a byproduct, produces the maximum independent set of a bipartite graph. Now if the biclique found out by the Hungarian algorithm was of size $|U| \times |V|$, we gave the solution to the Balanced Complete Biparite Subgraph problem as $min(|U|, |V|) \times min(|U|, |V|)$ biclique. But, this approch did not give any improvement over the previous results known[12].

Given the complement graph which represents the defective switches, our heuristic is to remove the node adjacent to maximum number of minimum-degree nodes in the other partition. As an example, consider the bipartite graph in Fig. 4 to be the complement graph (the graph representing the defects). Now, if we had to remove a node from the left partition, Algorithm 1 would remove $A$ or $B$ since they have the highest degree of 2 in the left partition. But, Algorithm 2 would remove $C$, since it is the only node connected to the minimum degree node on the right partition (in this case $F$). As soon as $C$ is removed, $F$ can be included in the independent set, while removal of $A$ or $B$ does not immediately result in any node being included into the independent set.

Conceptually, Algorithm 1 tries to reduce the number of

edges in the graph, while Algorithm 2 tries to reduce the degree of the least-degree nodes in a partition.

**Algorithm 2 Function** Biclique $(G(U,V,E))$

1: Obtain $\bar{G}(U,V,\bar{E})$, $\bar{E} = K_{|U|,|V|} - E$
2: {Find the maximum independent set in $\bar{G}$}
3: $U^b \leftarrow \phi, V^b \leftarrow \phi, flag \leftarrow TRUE$
4: **repeat**
5: $U^b \leftarrow U^b \bigcup \{u | u \in U, d(u) = 0\}, U \leftarrow U - U^b$
6: $V^b \leftarrow V^b \bigcup \{v | v \in V, d(v) = 0\}, V \leftarrow V - V^b$
7: **if** flag **then**
8: $v \leftarrow$ node in $V$ with minimum degree
9: **for** each $u \in U$ **do**
10: $w(u) \leftarrow 0$
11: **end for**
12: **for** each $v' \in V$ such that $d(v) = d(v')$ **do**
13: **for** each $u \in U$ such that $(u, v') \in \bar{E}$ **do**
14: $w(u) \leftarrow w(u) + 1$
15: **end for**
16: **end for**
17: $u' \leftarrow$ node in $U$ with maximum $w$
18: $U \leftarrow U - \{u'\}$
19: **for** each $v'' \in V$ such that $(u', v'') \in \bar{E}$ **do**
20: $d(v'') = d(v'') - 1$
21: **end for**
22: **else**
23: $u \leftarrow$ node in $U$ with minimum degree
24: **for** each $v \in V$ **do**
25: $w(v) \leftarrow 0$
26: **end for**
27: **for** each $u' \in U$ such that $d(u) = d(u')$ **do**
28: **for** each $v \in V$ such that $(u', v) \in \bar{E}$ **do**
29: $w(v) \leftarrow w(v) + 1$
30: **end for**
31: **end for**
32: $v' \leftarrow$ node in $V$ with maximum $w$
33: $V \leftarrow V - \{v'\}$
34: **for** each $u'' \in U$ such that $(u'', v') \in \bar{E}$ **do**
35: $d(u'') = d(u'') - 1$
36: **end for**
37: **end if**
38: $flag \leftarrow \neg flag$
39: **until** $U = \phi$ or $V = \phi$
40: **return** $U^b \times V^b$ as the maximum biclique

As in Algorithm 1, the bipartite complement is constructed and then we alternate between $U$ and $V$ to remove one node in each iteration and add the nodes with zero degree into our biclique. But, we choose the node to be deleted according to the weight, $w$. As an example, if we want to remove a node from partition $U$, we initialize $w$ of all the nodes in $U$ to be zero. Then we get the least degree in the other partition $V$, say $p$. Now, for each vertex $v \in V$ with degree $p$; i.e., $d(v) = p$, we increase the weight of all the neighbours of $v$ by 1. Now we delete the node with the highest weight in $U$. Similarly, in the next iteration we delete a node from $V$ and so on until at least one of $U$ or $V$ is empty.

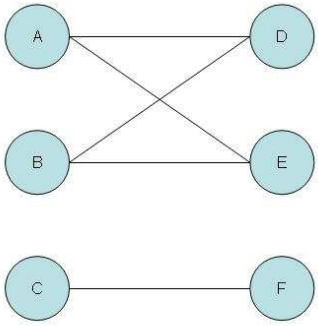


Fig. 4. An example of the differences between Algorithm 2 and Algorithm 1

In Algorithm 2, the bipartite complement of the representative graph is constructed (line 1). A flag is maintained to alternate between $U$ and $V$. If we suppose that the $flag$ is $true$, a node is then to be deleted from $U$ based on the weight $w$. First, a node with the least degree in $V$ is chosen, say $v$ (line 8). Now, the weight $w$ for all the nodes in $U$ is made 0 (lines 9-10). Now, for all the vertices $v' \in V$ with degree equal to that of $v$, the weight of all the neighbours of $v'$ is increased by 1 (lines 12-14). Now, the vertex with maximum $w$ in $U$, say $u'$, is selected to be deleted (lines 17-18). The degrees of the neighbours of $u'$ are adjusted accordingly (lines 19-20). The flag is negated (line 38). When $flag$ is $false$, a node is to be deleted from $V$ in the same way as was done for $U$. Finally, when at least one of $U$ or $V$ is empty, the biclique is returned.

### A. Proof of Correctness

The correctness of the algorithm directly follows from the correctness of Algorithm 1. This is because the only nodes included in the biclique are the zero degree nodes in the bipartite complement.

### B. Time Analysis

Given a $n \times n$ crossbar, the time complexity for constructing the representative graph is $O(n^2)$. Lines 3, 7, 38 take constant time. Loop (lines 4-39) is executed $O(n)$ times, as in each iteration, at least one node is deleted from one of the partitions. Lines 5-6 take $O(n)$ time. Lines 8-21 and 23-36 have the same time complexity by symmetry. Now, lines 8-11 and 17-21 have $O(n)$ time complexity. But, lines 12-16 are of $O(n^2)$ time complexity. Hence, the overall time complexity of Algorithm 2 is $O(n^3)$. The time complexity is tight, as it is easy to see that if the faults form a biclique, and if the faults are of $\Theta(n^2)$ (which has been found to be true), then the running time of Algorithm 2 will be $\Theta(n^3)$. The average running time of Algorithm 2 for $1000 \times 1000$ crossbar, on a machine with $4 \times 2.80$ GHz processors running on Linux OS, was 1.32 seconds.

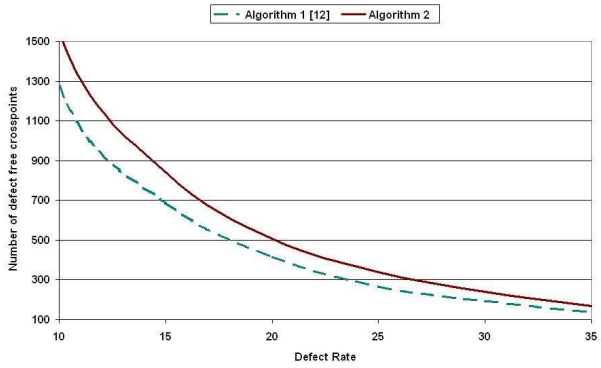


Fig. 5. $250 \times 250$ Crossbars

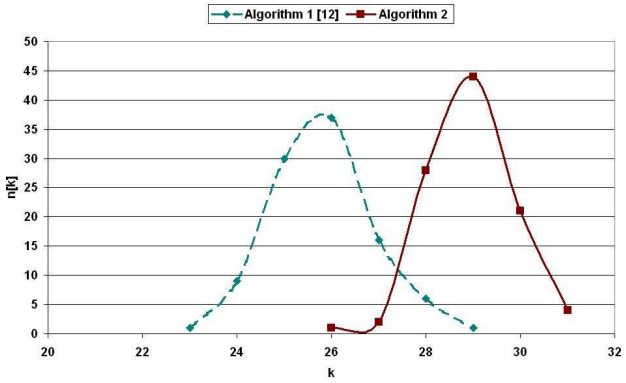


Fig. 6. Distribution of the resulting crossbars from Algorithm 1 and Algorithm 2 for $250 \times 250$ crossbars with $15\%$ defect rate

### C. Space Analysis

We may use an adjacency list or an adjacency matrix representation for the representative graph. In either case, the space complexity for the graph representation dominates the space complexity for the rest of the algorithm, since the number of edges are of $O(n^2)$. Hence, the space complexity of the algorithm is $\Theta(n^2)$.

## VI. Simulation Results

Algorithm 1 from [12] and the proposed algorithm (Algorithm 2) were both simulated using C++. The two algorithms were compared in terms of efficiency, consistency, yield and cost.

### A. Efficiency

If a $p \times p$ defect-free crossbar is required for the defect-unaware design flow (see IV-A.1). The value of $m$ is directly related to the efficency of the mapping algorithm. The more efficient the mapping algorithm, the smaller will be the value of $m$. Hence, a smaller defective crossbar will fulfill the requirements for the defect-unaware design flow. For each of the crossbar sizes 250, 500, 1000 and 10000, with defect rates considered were $5\%$ to $35\%$ (in steps of 5).

*1)* $250 \times 250$ *Crossbars:* For the $250 \times 250$ crossbars, the results of Algorithm 2 were just better than those of Algorithm 1, with the maximum increase in the results being $27.42\%$, while the average of the results is $22.38\%$ better. The sizes of the resulting crossbars are given in Fig.5 (lines smoothed).

*2)* $500 \times 500$ *Crossbars:* For the $500 \times 500$ crossbars, the results of Algorithm 2 are better than those of Algorithm 1, with the maximum increase in the results being $37.14\%$, while the average of the results is $29.93\%$ better. The sizes of the resulting crossbars are given in Fig.7.

*3)* $1000 \times 1000$ *Crossbars:* For the $1000 \times 1000$ crossbars, the results of Algorithm 2 are better than those of Algorithm 1, with the maximum increase in the results being $51.89\%$, while the average of the results is $44.03\%$ better. The sizes of the resulting crossbars are given in Fig.9.

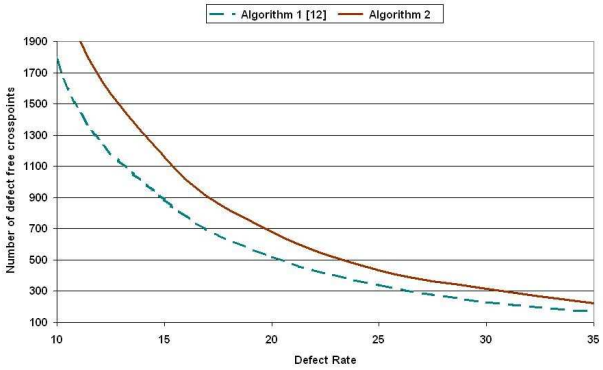


Fig. 7. $500 \times 500$ Crossbars

*4)* $10000 \times 10000$ *Crossbars:* For the $10000 \times 10000$ crossbars, the results of Algorithm 2 are substantially better than those of Algorithm 1, with the maximum increase in the results being $113.73\%$, while the average of the results is $97.14\%$ better. The sizes of the resulting crossbars are given in Fig.11.

The percentage improvement in defect-free switch densities for different manufactured switch sizes is shown in Fig.13 for different defect densities. The figure shows that the proposed algorithm (Algorithm 2) is much more efficient (50% to 100%) than Algorithm 1 from [12].

### B. Consistency

For each of the crossbar sizes (n) 250, 500, 1000 and 10000, with defect rates (p) considered were $5\%$ to $35\%$ (in steps of 5), 100 random samples were generated and the resulting number of crossbars from the two algorithms are shown in Fig. 6, 8, 10, 12. $n[k]$ is number of crossbars of size equal to $k$. Fig. 6, 8, 10, 12 are four representave examples of the distributions obtained (p=$15\%$ in all the four cases).

The distribution of the resulting crossbars for Algorithm 2 were better than those of Algorithm 1 (The range of defect-free switch densities spanned by Algorithm 1 is consistently higher than that spanned by Algorithm 2) as seen in Fig. 6, 8, 10, 12. $SD(1)$ and $SD(2)$ are the standard deviation of the

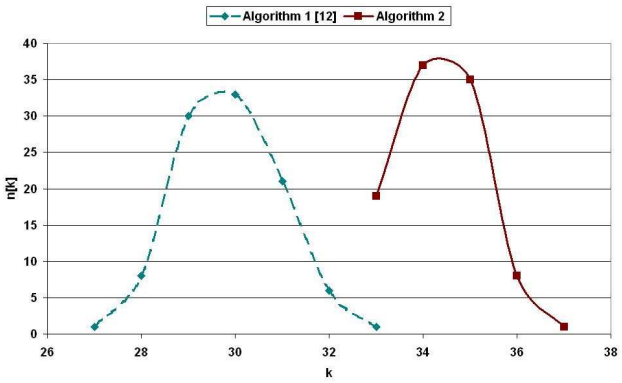


Fig. 8. Distribution of the resulting crossbars from Algorithm 1 and Algorithm 2 for $500 \times 500$ crossbars with $15\%$ defect rate

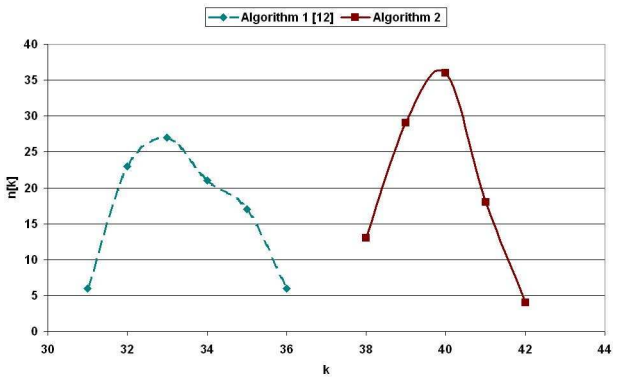


Fig. 10. Distribution of the resulting crossbars from Algorithm 1 and Algorithm 2 for $1000 \times 1000$ crossbars with $15\%$ defect rate

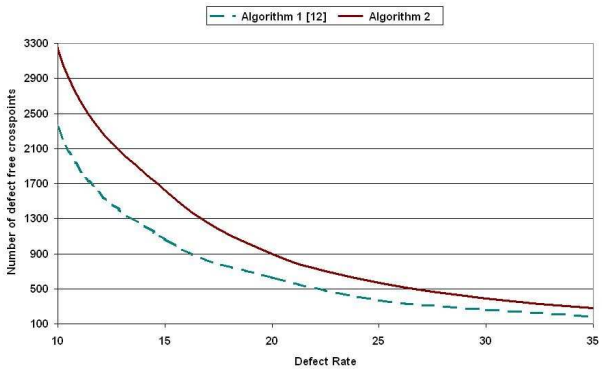


Fig. 9. $1000 \times 1000$ Crossbars

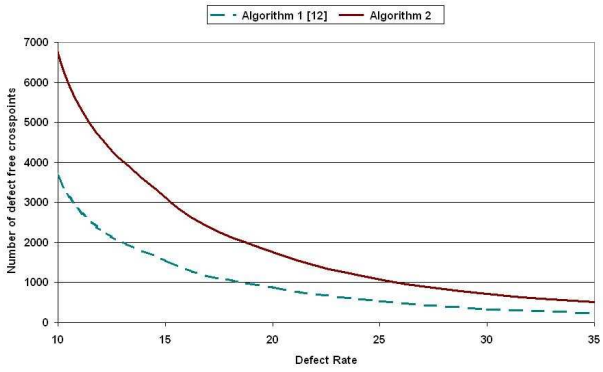


Fig. 11. $10000 \times 10000$ Crossbars

distribution curve of Algorithm 1 and Algorithm 2 respectively (assuming uniform distribution of defects). Figure 14 shows $SD(1) - SD(2)$ averaged over the defect rates.

$SD(1) > SD(2)$ directly implies that accuracy of $m$ (see IV-A.1) given by Algorithm 2 is better than Algorithm 1. The consistency of the algorithm has a direct impact on the yeild as seen below.

### C. Yield

As Algorithm 2 outperforms Algorithm 1, both in terms of efficiency and consistency, it may be expected that the same holds true for yield. Its indeed the case, as can be seen in Fig. 16, here $n = 250$ and the defect rate $p = 15\%$. Now the yield of the two algorithms can be seen as a function of $k$; i.e, given a $k$ what percentage of the $n \times n$ crossbars with defect rate $p$ can result in defect-free crossbars of size atleast $k$. In Fig. 16, for $k < 24$ the yield of both the algorithms is $100\%$ and for $k > 31$ the yield of both the algorithms is $0\%$. But, for the values of $k \in [24, 31]$, the yields of the algorithms differ. As an example for $k = 27$, the yeild of Algorithm 1 is less than $10\%$ while that of Algorithm 2 is greater than $90\%$.

As can be seen from Fig. 6, 8, 10, 12, the resulting average value of $k$ of proposed algorithm is always greater than the previous algorithm. Hence, yield of the proposed algorithm is always better than previous algorithm.

### D. Cost

Assuming that the cost of manufacturing is proportional to the number of cross-points in the crossbar, the cost per chip for Algorithm 2 is always lower than that of Algorithm 1.

As can be seen from Fig. 6, 8, 10, 12, the resulting average value of $k$ of proposed Algorithm 2 is always greater than the previous Algorithm 1. Hence, cost per chip of the proposed algorithm is always lower than previous algorithm.

## VII. Summary and Conclusions

This paper proposed a greedy approximation algorithm for the Balanced Complete Bipartite Subgraph problem. The algorithm can be directly applied as a reconfiguration-based defect tolerance algorithm for nanotechnology crossbar switches. Our algorithm is shown to have better results than the previous approaches. One interesting finding is that as the size of the crossbar increases, the performance of the proposed algorithm, as compared to the previous algorithm, increases substantially, in terms of efficiency Fig.15, consistency Fig.14 and yield Fig.16. This has a direct impact on the cost per chip as seen in Sec. VI. The comparison of the proposed algorithm with the exact algorithm was not delineated because for even small crossbars (size $64 \times 64$), the exact algorithm is completely intractable [12]. Also, the proposed algorithm yields better results than [12] which gave close results to the exact algorithm.

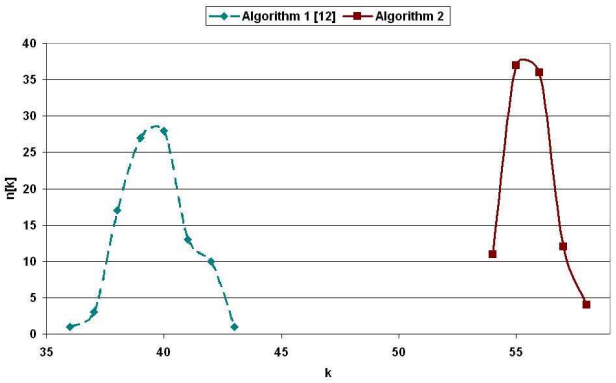


Fig. 12. Distribution of the resulting crossbars from Algorithm 1 and Algorithm 2 for $10000 \times 10000$ crossbars with $15\%$ defect rate

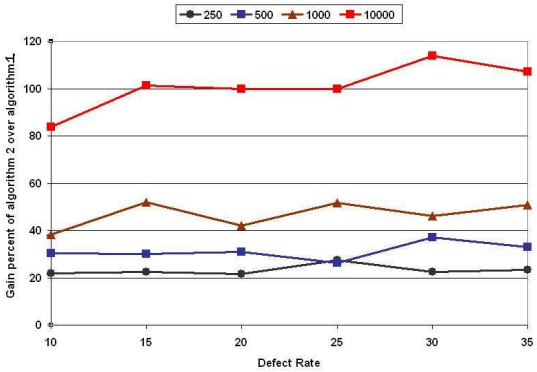


Fig. 13. Percentage increase of results of proposed algorithm over previous algorithm for various crossbar sizes

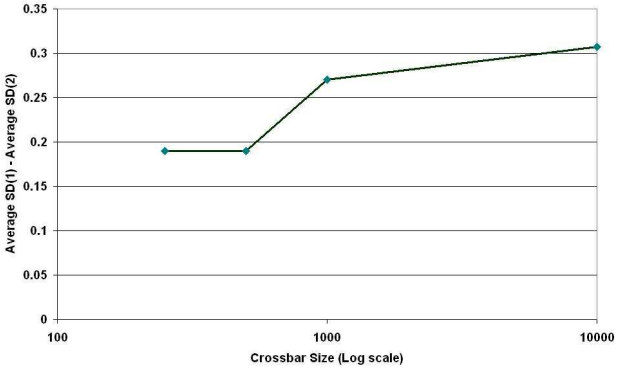


Fig. 14. The average of $SD(1) - SD(2)$ vs crossbar size

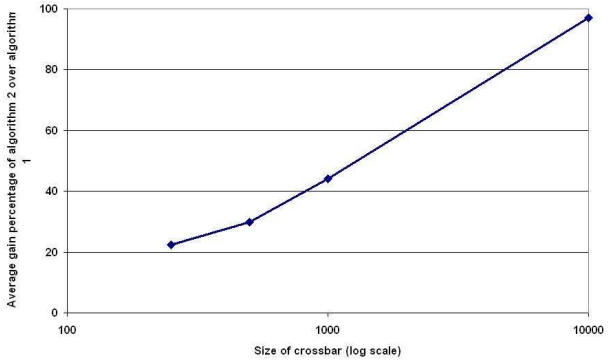


Fig. 15. Average gain percentage of algorithm 2 over algorithm 1 for various crossbar sizes

Possible improvements to the given algorithm might include making it faster. Another improvement could be to find an algorithm which can be parallelized optimally.

## REFERENCES


[1] M. Butts, André DeHon, S. C. Glodstein, "Molecular electronics: devices, systems and tools for gigagate, gigabit chips", *Proc. International Conference on Computer-Aided Design*, pp. 443-440, 2002.

[2] A. Bachtold, P. Harley, T. Nakanishi, C. Dekker, "Logic circuits with carbon nanotube transistors", *Science*, vol. 294, pp. 1317-1320, 2001.

[3] Y. Cui, C. M. Lieber, "Functional nanoscale electronics devices assembled using silicon nanowire building blocks", *Science*, vol 291, pp. 851-853, 2001.

[4] Y. Huang, X. Duan, Y.Cui, L. J. Lauhon, K. H. Kim, C. M. Leiber, "Logic gates and computation from assembled nanowire building blocks", *Science*, vol 294, pp. 1313-1317, 2001.

[5] P. D. Tougaw, C. S. Lent, "Logical devices implemented using quantum cellular automata", *Applied Physics*, vol 75(3, pp. 1818-1825, 1994.

[6] Y. Chen, G.-Y. Jung, D. A. A. Ohlberg, X. Li, D. R. Stewart, J. O. Jeppesen, K. A. Nielsen, J. F. Stoddart, and R. S. Williams, "Nanoscale molecular-switch crossbar circuits", *Nanotechnology*, vol. 14, pp. 462-468, 2003.

[7] Jing Huang, Mehdi B. Tahoori and Fabrizio Lombardi, "On the defect tolerance of nano-scale two-dimensional crossbars", *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2004.

[8] Helia Naeimi and André DeHon, "A greedy algorithm for tolerating defective crosspoints in NanoPLA desing", *IEEE International Conference on Field-Programmable Tehnology*, 2004.

[9] André DeHon, "Array-based architectures for FET-based, nanoscale electronics", *IEEE Trans. on Nanotechnology*, vol 2, No. 1 pp. 23-32,2003.

[10] André DeHon, M. J. Wilson, "Nanowire-based sublithographic programmable logic arrays", *Proc. International Symposium on Field-Programmable Gate Arrays (FPGA 2004)*, pp. 123-132, 2004.

[11] Seth Copen Goldstein and Mihai Budiu, "Nanofabrics: spatial computing using molecular electronics", *International Symposium on Computer Architecture*, 2001.

[12] Mehdi B. Tahoori, "A mapping algorithm for defect-tolerance of reconfigurable nano-architectures", *Proc. Int'l Conf. on Computer-Aided Design,* pp. 668-672, 2005.

[13] G. Snider, P. Kuekes, and R. S. Williams, "CMOS like logic in defective, nanoscale crossbars", *Nanotechnology*, vol. 15, pp. 881891, June 2004.

[14] M. M. Ziegler, M.R. Stan, "Design and analysis of crossbar circuits for molecular nanoelectronics", *Proc. IEEE International Conference on Nanotechnology*, 2002.

[15] M. Yannakakis, "Node-deletion problems on bipartite graphs", *SIAM J. Comput*, Vol. 10, No. 2, May 1981.

[16] M. Garey and D. Johnson, *Computers and intractability*, Freeman, New York, 1979.

[17] Thomas H. Cormen, Charles E. Lierson, and Ronald L. Rivest, Clifford Stein. 2/e *Introduction to algorithms*. MIT Press, Cambridge, MA.

[18] Feige, U. and Kogan, S. (2004) "Hardness of approximation of the balanced complete bipartite subgraph problem". Technical Report MCS04-04, Department of Computer Science and Applied Math., The Weizmann Institute of Science.

[19] Milind Dawande, Pinar Keskinocak, Jayashankar M. Swaminathan and Sridhar Tayur, "On bipartite and multiparite clique problems", *Journal of Algorithms,* **41-2,** 388-403 (2001).

[20] Dorit S. Hochbaum, "Approximating clique and biclique problems", *Journal of Algorithms,* **29-1,** 174-200 (1998).

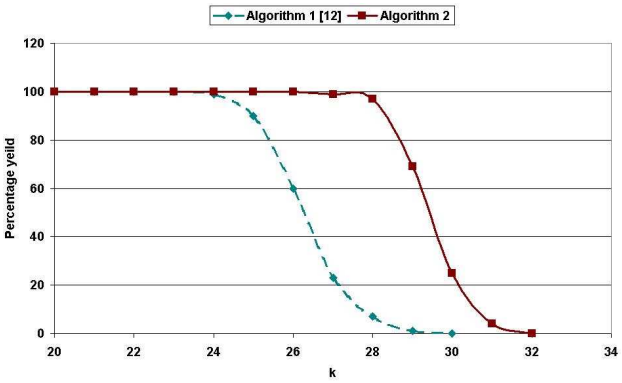


Fig. 16. Percentage yield of the algorithms for $250 \times 250$ cross-bars with $15\%$ defect rate

[21] Harold W. Kuhn, "The Hungarian method for the assignment problem", *Naval Research Logistic Quarterly* 2 (1955), pp. 83-97.