A Project Report on

Defect Tolerance of Nanoscale Crossbar Circuits

Submitted to the Department of Computer Science and Technology

**For the partial fulfilment of the degree of B.E. in**

**Computer Science and Technology**

by

**SAYANTAN PANDIT**

Examination Roll Number : 111205022

Registration number: 110512022 of 2015-16

B.E., 4th year

Under the supervision of

**Prof. Malay Kule**



Department of Computer Science and Technology

INDIAN INSTITUTE OF ENGINEERING SCIENCE AND

TECHNOLOGY, SHIBPUR

*May, 2016*

****

**Department of Computer Science and Technology**

**Indian Institute of Engineering Science and Technology,**

**Shibpur**

**CERTIFICATE**

This is to certify that the work presented in this report entitled “**Defect Tolerance of Nanoscale Crossbar Circuits**”, submitted by **Sayantan Pandit**, having the examination roll number **111205022**, has been carried out under my supervision for the partial fulfilment of the degree of **Bachelor of Technology** in **Computer Science and Technology** during the session **2015-16** in the **Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur.**

—————————————– —————————————–

Prof. Malay Kule Prof. (Dr.) Biplab Kumar Sikdar

Assistant/Associate Professor Head of the Department

Department of Computer Science and Technology Department of Computer Science and Technology

Indian Institute of Engineering Science Indian Institute of Engineering Science

and Technology, Shibpur and Technology, Shibpur

—————————————– Date:

Prof. (Dr.) Amit Kumar Das

Dean (Academic)

Indian Institute of Engineering Science

and Technology, Shibpur

**Acknowledgements**

It gives us the immense pleasure to express our deepest sense of gratitude and sincere thanks to my highly respected and esteemed guide, Prof. Malay Kule, for his valuable guidance, encouragement and help for partial completing of this project. His useful suggestions for this whole work and cooperative behavior are sincerely acknowledged.

Date: ————————————––––

SAYANTAN PANDIT

Department of Computer Science and Technology

Indian Institute of Engineering Science

and Technology, Shibpur

**INDEX**

|  |  |  |
| --- | --- | --- |
| **1.** | **INTRODUCTION** | 5 |
|  | |  |  |  | | --- | --- | --- | | 1.1 | Motivation | 5 | | 1.2 | Previous Study | 5 | | 1.3 | Problems with the Previous Study | 5 | | 1.4 | About the Project | 5 | | 1.5 | Objectives | 6 | | 1.6 | Achievements | 6 | | 1.7 | About this Report | 6 | | |
| **2.** | **LITERATURE SURVEY** | 7 |
|  | |  |  |  | | --- | --- | --- | | 2.1 | Crossbar Architecture | 7 | | 2.2 | Terminology | 8 | | 2.3 | Problems | 9 | | 2.4 | Previous Research | 9 | | |
| **3.** | **ALGORITHMS** | 14 |
|  | |  |  |  | | --- | --- | --- | | 3.1 | Algorithm 1 | 14 | | 3.2 | Algorithm 2 | 15 | | |
| **4.** | **PROPOSED ALGORITHMS AND EVALUATION** | 17 |
| **5.** | **REMOVING ALL CLOSED DEFECT** | 18 |
|  | |  |  |  | | --- | --- | --- | | 5.1 | Preamble | 18 | | 5.2 | Description | 18 | | 5.3.1 | Algorithms – K1 x K2 | 18 | | 5.3.2 | Example of K1 x K2 | 19 | | 5.3.3 | Graph Analysis | 20 | | 5.3.4 | Advantages | 23 | | 5.4.1 | Algorithm – K x K | 24 | | 5.4.2 | Graph Analysis | 25 | | 5.4.3 | Advantages | 28 | | 5.4.4 | Disadvantages | 28 | | |
| **6.** | **DEFECT-FREE SOLUTION USING MER** | **29** |
|  | |  |  |  | | --- | --- | --- | | 6.1 | Introduction | 29 | | 6.2 | Description | 29 | | 6.3 | Algorithm | 29 | | 6.4 | Our Modification | 30 | | 6.5 | Advantages | 30 | | 6.6 | Disadvantages | 30 | | |
| **7.** | **APPLICATION OF GRAHAM SCAN ALGORITHM** | **31** |
|  | |  |  |  | | --- | --- | --- | | 7.1 | Introduction | 31 | | 7.2 | Algorithm: Defected Area | 31 | | 7.3 | Explanation | 32 | | 7.4 | Time Complexity | 33 | | 7.5 | Advantages | 33 | | 7.6 | Disadvantages | 33 | | 7.7 | Implementation and Results | 34 | | |
| **8.** | **CONCLUSION** | **36** |
| **9.** | **REFERENCES** | **37** |

**1. INTRODUCTION**

**1.1 Motivation**

Nano-scale devices and molecular electronics promise to overcome the fundamental physical limitation of lithography-based silicon VLSI technology. It is projected that molecular electronics can achieve density of 1012 devices per cm2 and operated at THz(TeraHertz) frequencies. Researchers have demonstrated several successful nano-scale electronic devices including carbon nanotubes (CNT), silicon nanowires (NW) and quantum dot cells.

During manufacturing in the industrial process, the nanoscale crossbar circuits are produced with a number of defective permanent faults. But, the fully populated 2D crossbar provides defect tolerance capabilities even in the presence of some defects. A defect crossbar circuits with N x N crossbar can still be used as a defect-free K x K (where K <= N) crossbar circuits at an architectural level. This approach is similar to the conventional memory defect tolerant scheme which utilizes spare rows and columns. So, by implementing algorithms and passing through various tests at every corner cases, defective junctions can be abled to detect. Hence with the proper techniques to bypass the defective junctions, the circuit with the defective-free junctions is utilized.

**1.2 Previous Study**

The defects caused by the manufacturing process in the industrial process can be bypassed developing algorithms which are briefly described in chapter 3. A defective cross-bar of dimension of N x N can be minimized to a K x K sized defect-free cross bar using algorithm 1.

**1.3 Problems with the Previous Study**

Using algorithm 1 from [3], a maximum K x K defect-free crossbar from a N x N crossbar (K<=N) can be developed. But, many of the junctions in the region (N-K) x (N-K) can still be operational, which is not used further. This drawback is minimized in larger manner by developing an algorithm 2. Again, there is a problem in time complexity, which takes O (n3).

**1.4 About the Project**

This report presents a technique that supersede the existing technique for finding the maximum defect-free sub-switch in a defective crossbar switch. As shown by the results,the proposed algorithm has higher efficiency, meaning that the required size to be manufactured in order to achieve a certain defect-free sub-switch is 22% to 97% smaller than the size required by the existing techniques. The proposed algorithms are consistent in outperforming the previous work with a large number of samples with a much smaller standard deviation across the samples. The yield is also better with the new algorithm compared to the previous work. The advantages of proposed algorithm over the previous work are directly proportional to the switch size.

**1.5 Objectives**

The objective of this project are discussed below:

1. Algorithm 1 develops a maximum defect free k x k square zone bipartite graph techniques from a given N x N defect crossbar.
2. Algorithm 2 develops a maximum defect free area or zone from a given crossbar whose defect set points are given.

**1.6 Achievements**

The project claims these following achievements:

1. Developed a maximum k x k defect-free sub-matrix from a given N x N defective crossbar.
2. Developed a maximum defect free zone from a given set of fault points of a defective crossbar.

**1.7 About this Report**

In this report, the **chapter 2** specifies the background of the project stating brief discussions about the crossbar architectures, it’s produced faults during the industrial manufacturing periods and the importance of developing defect tolerant algorithms. In **chapter 3**, the two previous algorithms are introduced. In **chapter 4**, the implementation and evaluation of our proposed algorithm is going to be discussed in brief. In **chapter 5**, the algorithms for removing all closed defects is mentioned along with its analysis. In **chapter 6,** the rectangular area of defect-free zone is developed and the convex polygon is obtained using defect point is mentioned in **chapter 7** and finally, the report is concluded in **chapter 8**.

**2. LITERATURE SURVEY**

**2.1 Crossbar Architecture**

Many of the nanoscale computing architectures proposed in recent years are clock driven. These architectures are mainly based on the two-dimensional nanowire crossbar architecture. In this architecture two sets of parallel, doped silicon nanowires or carbon nanotubes, are crossed over each other orthogonally to form a grid-like structure. The crossing over of these nanowires forms programmable junctions called crosspoints. The primary challenge in designing clocked architectures is to route the clock to all the components of the circuit. Due to imperfections in nanowires fabricated using current manufacturing processes, high defect densities are anticipated and realizing complex synchronous circuits on them is intricate. Hence, nanowire crossbars offers both an opportunity and a challenge. The opportunity is to achieve ultra-high density which has never been achieved by photolithography. The challenge is to make them simple enough to be manufactured and reliable enough to be used in everyday computing applications, since high-density systems consisting of nanometer-scale elements assembled in a bottom-up manner are likely to have many 161 imperfections (much higher raw fabrication defect densities, as high as 10%, are expected) and parametric variations. Asynchronous crossbar architecture efficiently helps utilize the opportunity and keep up to the challenge of fabricating reliable complex circuitry. The following conditions are required to be met to make nanowire crossbar to be a viable nanotechnology:

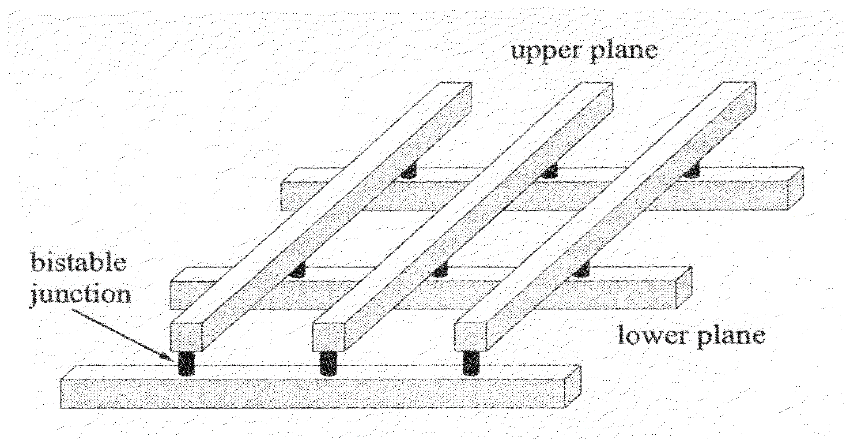
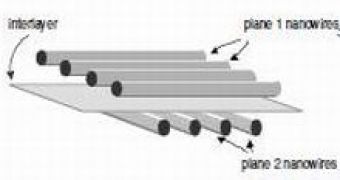
1. Structurally simple and scalable enough to be fabricated by bottom-up manufacturing technique,

2. Robust enough to tolerate extreme parametric variations

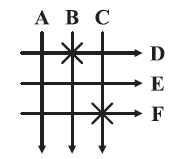
3. Defect and fault-tolerant enough to overcome the extreme defect densities, aging factors and transient faults

4. Able to support at-speed verification and reconfiguration. The proposed asynchronous nano-architecture is based on a delay-insensitive data encoding and self-timed logic - therefore, it is totally clock-free. Thus, no clock distribution network is needed and all failure modes related to the timing will be also eliminated. Potential benefits from the proposed asynchronous architecture includes enhanced manufacturability, scalability, robustness and defect and fault-tolerance.

Recently, almost all the industries related to nanoscale technology started finding the efficient algorithm that reduces the time and space so that they can utilise the crossbar circuits efficiently.

An example of a defective 3 x 3 nanoelectronic crossbar is shown in Fig. below. The vertical nanowires can be defined to be the inputs whereas the horizontal nanowires can be defined to be the outputs. There is a programmable switch at each crosspoint. The non-programmable defective switches at the crosspoints are represented by an “X” for each.



**2.2 Terminology**

|  |  |
| --- | --- |
| **1.** | **Bipartite Graph Representation :** A crossbar with a defect map can be represented by a bipartite graph. A bipartite graph of an n × n crossbar is an undirected bipartite graph G(U, V, E) with partitions U and V, having |U| = n and |V| = n. U represents the set of input nanowires, and V represents the output nanowires. E consists of representative edges for all the programmable non-defective crosspoints of the crossbar. Hence, if assuming the defect density is p, |E| = (1–p) · n2 . |
|  |  |
| **2.** | **Biclique :** In a defect-free crossbar, there is a programmable switch at each crosspoint, that is, |E| = n2. This graph is a complete bipartite graph. The complete subgraph of a bipartite graph is called the biclique. A biclique G(U, V, E) is said to be balanced if |U| = |V|. |
|  |  |
| **3.** | **Bipartite Complement Graph** : Given a bipartite graph G(U, V, E), the bipartite complement graph of G is Gc(U, V, Ec) with the same set of vertices such that two vertices of Gc are adjacent if and only if they are not adjacent in G. Hence, |Ec| = n2 −|E|. |
|  |  |
| **4.** | **Independent Set :** An independent set S in a graph G(U, V, E) is a subset of nodes that are disconnected: ∀(x, y) ∈ S, (x, y) / ∈ E. The maximum independent set is an independent set with the maximum number of nodes. |
|  |  |

**2.3 Problems**

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 x n switch with some defects can be used as a k x k switch, where k < n. Maximizing for a defect switch leads to higher logic densities on the same defective circuit.

The problem of finding the maximum k x k defect-free crossbar from a n x n crossbar corresponds to find the maximum balanced biclique in its representative graph. This problem is NP complete. This problem is called Balanced Complete Biparite Subgraph. Recently, it has been shown to be NP hard even to approximate within a constant factor. This is a common problem addressed in Operations Research, Algorithms, and now in nanotechnology.

If we remove the restriction that the biclique should be balanced, the problem is solvable in polynomial time. 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.

**2.4 Previous Research**

Various nanowire and switch faults were studied 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 impact on the routability of the crossbar for various nanowire and switch faults have been shown.

Another problem related to mapping on crossbars is dealt with in H. Naeimi and A. DeHon, “A greedy algorithm for tolerating defective crosspoints in nanoplate design,” in Ptov. IEEE Int. Conf. Field-Programmable Technol., 2004. 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. The running time of the algorithm is determined by the time required to construct the bipartite graph model of the crossbar. So, a greedy algorithm based on probabilistic methods is given to avoid the construction of the bipartite graph model of the crossbar. Hence, the time complexity of the algorithm is reduced by trading off its accuracy.

In “A mapping algorithm for defect-tolerance of reconfigurable nano architectures,” in Proc. Int. Conf. Computer-Aided Design, 2005, written by M. B. Tahoori, a novel defect-unaware design flow, for mass production 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 x n. The main idea in this defect-unaware flow model was to assume that there was a k x 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 statistical 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 x k defect free crossbar in a n x n defective crossbar. This problem reduces to the maximum balanced bipartite subgraph problem in the representative graph of the n x n crossbar.
* If the defect-unaware flow requires a p x 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 x m, to be manufactured to guarantee 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 x m crossbar with d% defect rate, the mapping algorithm should consistently produce crossbars of size not less than p x p.

1. **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 x 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(n2). 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 x k crossbar and which k of the n columns form the columns of the defect-free k x 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).

1. **Disadvantages**

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

**2.5 PREVIOUS WORKS ON NANOSCALE CROSSBAR ARCHITECTURE**

**2.5.1 EFFICIENT FUNCTION MAPPING IN NANOSCALE CROSSBAR ARCHITECTURE[10]**

For the function mapping technique, the logic function is represented by a matrix. Assume that the function f, expressed as sum-of-products, is

f = O f1 + O f2 --------- (1)

where O f1 = I a I b I c and O f2 = I b I c .

The function f require two wires for the two product terms (O f1 , O f2 ) and three wires, one each for the three literals (I a , I b , I c ), to implement the function in a nanowire crossbar array. Input and output relations can be represented by a matrix. If the logic function, represented as sum-of-products, has k variables and l product terms,a function matrix of size l x k is generated. Each literal is designated by a column and each row signifies a product term. However, the mapping of each column to a particular variable is not fixed at this time and is assigned later.

After a nanowire array is manufactured, the location and type of defects are identified using test and diagnostic procedures and a defect map is generated. Since crosspoint between O 2 and I 1 has a shorted switch defect and O 2 and I 3 has an open switch, 0 and 2 are assigned to the corresponding columns and rows. In this manner, nanowire crossbar in the presence of defects can be represented as a matrix.

Given a function to be mapped along with a defect map that describes the defects in the crossbar architecture, the constraint equations are derived which would govern a valid function mapping. A variable X is defined for row mapping and X i\_j denotes the mapping of the i th row in the function matrix to j th row in the crossbar matrix. If there is a valid row to row mapping found, X i\_j will be assigned ‘1’ and ‘0’ will be assigned if no mapping is found.

For a k 1 x k 2 function matrix to be mapped to a n 1 x n 2 nanowire matrix, six constraint relations are generated mapping the rows in the function matrix. Using these six constraint relations, a valid mapping between the function matrix and the crossbar matrix can be obtained.

The above constraint relations if used as input to an ILP solver yield the desired mapping, if one exists. However, for a nanowire array with high defect density, the computational complexity could be fairly high if the full state-space is used. When the constraint equations are being solved, a bigger solution space would require more time before a valid mapping could be obtained, if one exists.

**2.5.1.1 RESULTS**

I. 100% valid mapping is found for defect rates of 5% ~ 25%.

II. For a defect rate of 45%, the proposed mapping succeeds in 96% of the cases.

**2.5.2 RUNTIME ANALYSIS FOR DEFECT-TOLERANT LOGIC MAPPING ON NANO SCALE CROSSBAR ARCHITECTURES[11]**

**2.5.2.1 LOGIC FUNCTION MODEL**

Just as the crossbar, a two-level logic function in the form of SOP (sum of product) can be modeled by a bipartite graph,based on the relationship between its variable set and product set. Unlike CBGs, edges in an Logic Bipartite Graph (LBG) fall into only two types: inclusion and exclusion. Inclusion indicates a particular variable included in the product, and exclusion indicates otherwise.

**2.5.2.2 SOLUTION DENSITY EXPECTATION**

The solution space volume depends on (i) logic function size(variable and product numbers) and (ii) crossbar size. When both are given, the solution space volume is fixed. Defect spose constraints in the mapping process and result in a drastic decrease in the number of solutions in the solution space. Naturally, more defects lead to lower solution density. When a larger crossbar is used, there could be more solutions existing in the enlarged solution space. When the size of logic function increases, the number of solutions decreases in the yet increased solution space. During the mapping process, large logic functions inevitably hit into more defects at a mapping trial, since more wires as well as switches need to be used. Therefore, when the logic function size gets larger, we expect μ sd to drop sharply.

**2.5.2.3 SOLUTION DENSITY STANDARD DEVIATION**

Solution density standard deviation σ sd reveals the variability, which, in turn, indicates dropping of accuracy when using μ sd for runtime prediction.

Large crossbars lead to low σ sd uniformly at all defect rate, and σ sd reduction speed becomes slower as crossbar size further increases. Thus, σ sd decreases with crossbar size increasing.

**2.5.2.4 RESULTS**

**I.** As μ sd increases, runtime expectation always decreases.

**II.** Large crossbar size leads to reduced runtime expectation.

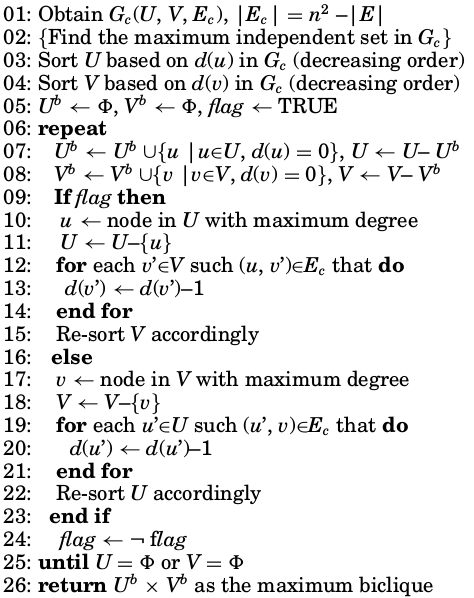
**III.** Runtime estimation based on 1/μ sd can predict runtime expectation accurately when crossbar size is above two times larger than the logic function size.

**3. Algorithms**

**3.1 Algorithm 1**

The algorithm proposed in Tahoori [2006] (Algorithm 1) constructs the complement graph of the bipartite graph, and then tries to find the maximum balanced independent set in the complement graph. For this, the heuristic is based on removing nodes with the maximum degree. To keep the resulting bipartite graph almost balanced, the algorithm alternates between U and V while removing the nodes.

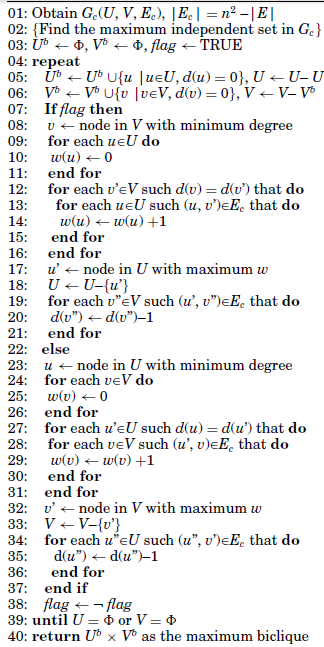
**Algorithms 1 :** Function Biclique (G(U,V,E))



The complement graph is formed first (line 1), then the iteration loops start. In each iteration, nodes with zero degree are added to the solution list first (lines 7–8). Then the heuristic is applied alternately between U and V while removing the maximum degree node in the partition according to the flag value (lines 10–11, 17–18).

**3.2 Algorithm 2**

The algorithm given in Al-Yamani et al. [2007] (Algorithm 2) also converts the maximum biclique problem into the maximum independent set problem, similar to Algorithm 1. The heuristic removes the node in one partition that is adjacent to the maximum number of minimum degree nodes in the other partition. Therefore, in brief, to achieve the maximum biclique, 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.



In Algorithm 2, the complement graph is constructed first (line 1), then the iteration loops start. In each iteration, first a node v with the minimum degree in V is chosen (line 8). The weight w for each node in U is made 0 (lines 9–11). Then, for all vertices v’ ∈ V with degree equal to that of v, the weight of all the neighbors of v’ is increased by 1 (lines 12–16).

Finally, the vertex with maximum w in U, say u’, to be deleted (lines 17–18) is selected. The heuristic alternates between U and V while removing the node with maximum w in the partition according to the flag value (lines 8–18, 23–33). The loop between lines 5–38 will be executed O(n) times, since in each iteration, at least one node is removed. The body of the loop will take O(n) time complexity to find the node with minimum degree v (line 8).

The computation of weight w is time-consuming, with time complexity O(n2) (lines 9–16). O(n × p) with time complexity is needed to adjust the degree of the neighbors of u’ (lines 19–21). Lines 8–21 and 23–36 have the same time complexity by symmetry. Hence, the total time complexity of the algorithm is O(n) × (O(n) + O(n2) + O(n × p)) = O(n3) [Al-Yamani et al. 2007].

**4. PROPOSED ALGORITHMS AND EVALUATION**

**Step 1:** Choose an element of the ON-set

**Step 2:** Find "maximal" groupings of 1s and Xs adjacent to that element

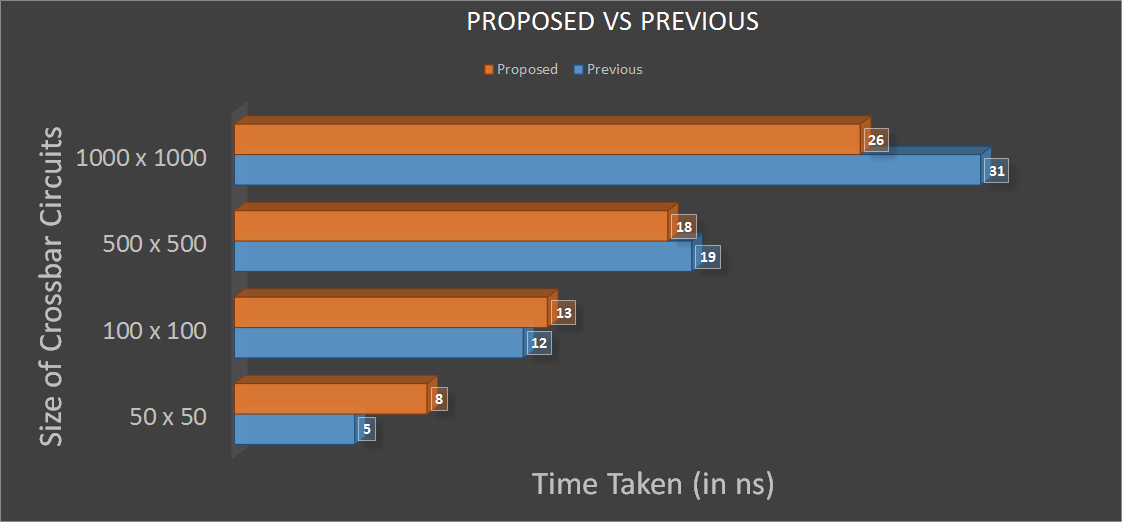
* consider top/bottom row, left/right column, and corner adjacencies
* this forms prime implicants Repeat Steps 1 and 2 to find all prime implicants

**Step 3:** Revisit the 1s in the K-map

* if covered by single prime implicant, it is essential, and participates in final cover
* 1s covered by essential prime implicant do not need to be revisited

**Step 4:** if there remain 1s not covered by essential prime implicants

* select the smallest number of prime implicants that cover the remaining 1s



**5. Removing all Closed Defect**

**5.1 Preamble**

A crossbar consists of a large number of junction which may includes the various types of errors like open defect, closed defect etc. But it is found from the literature survey that closed defect can’t be greater than the open defect. So, the main task of this section is to eliminate the closed defect in the rows or columns and thereby reducing the open defect also if present in that rows or columns.

**5.2 Description**

Since the crossbar contains the various type of errors, but the proposed algorithm only deals with the Open and the Closed Defect. The algorithm shows, in the crossbar circuit of **K x K** matrixsize consists of defect junctions which can be eliminated by taking the maximum closed defect in the rows or columns. So, by eliminating closed defect in the row or column, open defect if present in that rows or columns is also eliminated. By this way, this process is continued till all the closed defect are eliminated.

**5.3.1 Algorithms - K1xK2**

**Input : 2D-array Matrix\_Details** and **Defect Percentage**

**Output: 2D Matrix** containing the elimination of Closed Defect.

**Assumption**

1. No.1 -> Junction is Perfect

2. No.0 -> Junction is Opened

3. No.2 -> Junction is Closed

4. No.3 -> Junction is Covered Opened or Closed

5. No.4 -> Rejected Defective Columns

**Steps**

1: Declared the 2D array **Matrix\_Details** which contains the details of the junction of the Open and Closed Defect.

2: Also, take input the defect percentage and calculate the total no. of defect from the **Matrix\_Details** and set that value in the variable **Total\_Defect**.

2a: Take two variable **OPEN\_DEFECT** and **CLOSED\_DEFECT**

2b: Calculate the value of **OPEN\_DEFECT** **:** {60%-90%} of the **Total\_Defect.**

2c: Calculate the value of **CLOSED\_DEFECT :** the remaining percentage from the Open\_Defect taken i.e. {10%-40%} of the **Total\_Defect.**

3: Take the two 2D array **Row\_Defect** and **Col\_Defect** to record the details of every rows and columns i.e. first cell and second cell contains the number of the open and closed defect respectively.

4. for loop: i=1 to Matrix Size

4a: for loop: j=1 to Matrix Size

4b: Set all cells of **Matrix\_Details** to 1

4c: Initialise **Row\_Defect** and **Col\_Defect** to 0

5. for loop: i=1 to Matrix Size

5a: for loop: j=1 to Matrix Size

5b: Set the cell to 1 if the junction is Open

5c: Set the cell to 0 if the junction is Closed

6: Infinite loop whose terminating condition is when there is no closed defect

6a: Finding the maximum defect in the Rows and Columns and set the value accordingly in the array **Row\_Defect** and **Col\_Defect.**

6b: Then, find the number of Closed defect which is greater in Rows or Columns

6b.1: If the number of Closed defect is same in both Rows or Columns, then Calculate the Sum of Closed Defect and Open Defect.

6b.1.1: After calculation, again the result is same, then take the value of the columns, otherwise

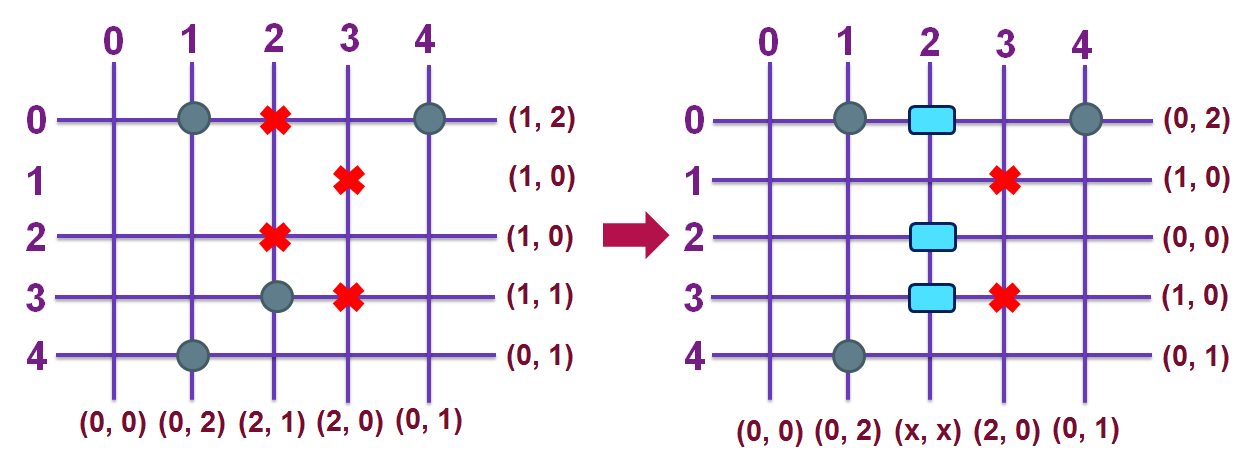
6b.1.2: Take the Row or Column whichever is maximum

6b.2: Take the Row or Column whichever is maximum

6c: Set the value : no.4 where there is a maximum no. of defect in the Row or Column

7: Print the resultant matrix

**5.3.2 Examples**



**Fig. 6.1.1 Fig. 6.1.2**

|  |  |
| --- | --- |
| **Fig. 6.1.3** |  |

**5.3.3 Graph Analysis**

**5.3.3.1. Matrix-wise Defect**

1. **Matrix Size - 10**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **10 v 10** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 82 | 90 | 90 | 100 |
| **10** | 68 | 74 | 81 | 90 |
| **15** | 58 | 68 | 75 | 90 |
| **20** | 51 | 58 | 67 | 81 |
| **25** | 44 | 53 | 62 | 81 |

**2. Matrix Size - 20**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **20 v 20** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 279 | 303 | 330 | 361 |
| **10** | 216 | 245 | 280 | 328 |
| **15** | 163 | 194 | 241 | 301 |
| **20** | 129 | 166 | 213 | 276 |
| **25** | 104 | 137 | 184 | 255 |

**3.Matrix Size - 40**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **40 v 40** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 867 | 980 | 1137 | 1327 |
| **10** | 561 | 687 | 868 | 1122 |
| **15** | 379 | 513 | 686 | 985 |
| **20** | 251 | 355 | 540 | 865 |
| **25** | 158 | 277 | 447 | 772 |

**4. Matrix Size - 80**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **80 v 80** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 2302 | 2854 | 3522 | 4535 |
| **10** | 1182 | 1623 | 2349 | 3544 |
| **15** | 550 | 946 | 1572 | 2775 |
| **20** | 222 | 561 | 1076 | 2270 |
| **25** | 62 | 230 | 705 | 1872 |

**5. Matrix Size - 100**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **100 v 100** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 3000 | 3878 | 4986 | 6678 |
| **10** | 1181 | 2002 | 3105 | 4970 |
| **15** | 489 | 1000 | 1962 | 3815 |
| **20** | 157 | 407 | 1212 | 2853 |
| **25** | 50 | 192 | 703 | 2255 |

**6. Matrix Size - 120**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **120 v 120** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 3723 | 4853 | 6506 | 9015 |
| **10** | 1401 | 2130 | 3703 | 6471 |
| **15** | 327 | 919 | 2216 | 4679 |
| **20** | 88 | 370 | 1118 | 3360 |
| **25** | 17 | 77 | 525 | 2284 |

**5.3.3.2. Ratio-wise Defect**

1. **Ratio-wise : 60%-40%**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Defect Percentage** | | | | | |  |
|  | **5%** | **10%** | **15%** | **20%** | **25%** |
| **10 v 10** | 82 | 68 | 58 | 51 | 44 |
| **20 v 20** | 279 | 216 | 163 | 129 | 104 |
| **40 v 40** | 867 | 561 | 379 | 251 | 158 |
| **80 v 80** | 2302 | 1182 | 550 | 222 | 62 |
| **100 v 100** | 3000 | 1181 | 489 | 157 | 50 |
| **120 v 120** | 3723 | 1401 | 327 | 88 | 17 |

1. **Ratio-wise : 70%-30%**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Defect Percentage** | | | | | |  |
|  | **5%** | **10%** | **15%** | **20%** | **25%** |
| **10 v 10** | 90 | 74 | 68 | 58 | 53 |
| **20 v 20** | 303 | 245 | 194 | 166 | 137 |
| **40 v 40** | 980 | 687 | 513 | 355 | 277 |
| **80 v 80** | 2854 | 1623 | 946 | 561 | 230 |
| **100 v 100** | 3878 | 2002 | 1000 | 407 | 192 |
| **120 v 120** | 4853 | 2130 | 919 | 370 | 77 |

1. **Ratio-wise : 80%-20%**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Defect Percentage** | | | | | |  |
|  | **5%** | **10%** | **15%** | **20%** | **25%** |
| **10 v 10** | 90 | 81 | 75 | 67 | 62 |
| **20 v 20** | 330 | 280 | 241 | 213 | 184 |
| **40 v 40** | 1137 | 868 | 686 | 540 | 447 |
| **80 v 80** | 3522 | 2349 | 1572 | 1076 | 705 |
| **100 v 100** | 4986 | 3105 | 1962 | 1212 | 703 |
| **120 v 120** | 6506 | 3703 | 2216 | 1118 | 525 |

1. **Ratio-wise : 90%-10%**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Defect Percentage** | | | | | |  |
|  | **5%** | **10%** | **15%** | **20%** | **25%** |
| **10 v 10** | 100 | 90 | 90 | 81 | 81 |
| **20 v 20** | 361 | 328 | 301 | 276 | 255 |
| **40 v 40** | 1327 | 1122 | 985 | 865 | 772 |
| **80 v 80** | 4535 | 3544 | 2775 | 2270 | 1872 |
| **100 v 100** | 6678 | 4970 | 3815 | 2853 | 2255 |
| **120 v 120** | 9015 | 6471 | 4679 | 3360 | 2284 |

**5.3.4 Advantages**

1. The code is useful when the circuits having closed defect, as in closed defect the current always flows through that junction.

**5.4.1. Algorithm - KxK**

**Input : 2D-array Matrix\_Details** and **Defect Percentage**

**Output: 2D Matrix** containing the elimination of Closed Defect.

**Assumption**

1. No.1 -> Junction is Perfect

2. No.0 -> Junction is Opened

3. No.2 -> Junction is Closed

4. No.3 -> Junction is Covered Opened or Closed

5. No.4 -> Rejected Defective Columns

**Steps**

1: Declared the 2D array **Matrix\_Details** which contains the details of the junction of the Open and Closed Defect.

2: Also, take input the defect percentage and calculate the total no. of defect from the **Matrix\_Details** and set that value in the variable **Total\_Defect**.

2a: Take two variable **OPEN\_DEFECT** and **CLOSED\_DEFECT**

2b: Calculate the value of **OPEN\_DEFECT** **:** {60%-90%} of the **Total\_Defect.**

2c: Calculate the value of **CLOSED\_DEFECT :** the remaining percentage from the Open\_Defect taken i.e. {10%-40%} of the **Total\_Defect.**

3: Take the two 2D array **Row\_Defect** and **Col\_Defect** to record the details of every rows and columns i.e. first cell and second cell contains the number of the open and closed defect respectively.

4. for loop: i=1 to Matrix Size

4a: for loop: j=1 to Matrix Size

4b: Set all cells of **Matrix\_Details** to 1

4c: Initialise **Row\_Defect** and **Col\_Defect** to 0

5. for loop: i=1 to Matrix Size

5a: for loop: j=1 to Matrix Size

5b: Set the cell to 1 if the junction is Open

5c: Set the cell to 0 if the junction is Closed

6: for loop: i=1 to Matrix Size

6.1: Alternate with the Rows and Columns

6a: Finding the maximum defect in the Rows orColumns and set the value accordingly in the array **Row\_Defect** and **Col\_Defect.**

6b: Then, find the number of Closed defect which is greater in Rows or Columns

6b.1: If the number of Closed defect is same in both Rows or Columns, then Calculate the Sum of Closed Defect and Open Defect.

6b.1.1: After calculation, again the result is same, then take the value of the columns, otherwise

6b.1.2: Take the Row or Column whichever is maximum

6b.2: Take the Row or Column whichever is maximum

6c: Set the value : no.4 where there is a maximum no. of defect in the Row or Column

7: Print the resultant matrix

**5.4.2 Graph Analysis**

**5.4.2.1 Matrix-wise Defect**

1. **Matrix Size - 10**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **10 v 10** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 81 | 81 | 81 | 100 |
| **10** | 65 | 69 | 81 | 81 |
| **15** | 57 | 65 | 69 | 81 |
| **20** | 49 | 56 | 65 | 81 |
| **25** | 45 | 52 | 61 | 81 |

**2. Matrix Size - 20**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **20 v 20** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 277 | 297 | 325 | 361 |
| **10** | 218 | 244 | 275 | 324 |
| **15** | 174 | 203 | 243 | 296 |
| **20** | 146 | 177 | 217 | 276 |
| **25** | 122 | 150 | 193 | 257 |

**3.Matrix Size - 40**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **40 v 40** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 897 | 996 | 1141 | 1317 |
| **10** | 617 | 730 | 887 | 1131 |
| **15** | 463 | 569 | 734 | 1002 |
| **20** | 402 | 457 | 608 | 885 |
| **25** | 400 | 406 | 522 | 802 |

**4. Matrix Size - 80**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **80 v 80** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 2516 | 2979 | 3607 | 4561 |
| **10** | 1613 | 1924 | 2534 | 3656 |
| **15** | 1600 | 1600 | 1896 | 2977 |
| **20** | 1600 | 1600 | 1617 | 2498 |
| **25** | 1600 | 1600 | 1600 | 2175 |

**5. Matrix Size - 100**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **100 v 100** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 3409 | 4107 | 5145 | 6744 |
| **10** | 2500 | 2573 | 3439 | 5156 |
| **15** | 2500 | 2500 | 2548 | 4131 |
| **20** | 2500 | 2500 | 2500 | 3419 |
| **25** | 2500 | 2500 | 2500 | 2914 |

**6.** **Matrix Size - 120**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **120 v 120** | | | | |  |
|  | **60-40** | **70-30** | **80-20** | **90-10** |
| **5** | 4348 | 5319 | 6775 | 9158 |
| **10** | 3600 | 3600 | 4383 | 6758 |
| **15** | 3600 | 3600 | 3600 | 5308 |
| **20** | 3600 | 3600 | 3600 | 4329 |
| **25** | 3600 | 3600 | 3600 | 3663 |

**5.4.2.2 Ratio-wise Defect**

1. **Ratio-wise : 60%-40%**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Defect Percentage** | | | | | |  |
|  | **5%** | **10%** | **15%** | **20%** | **25%** |
| **10 v 10** | 81 | 65 | 57 | 49 | 45 |
| **20 v 20** | 277 | 218 | 174 | 146 | 122 |
| **40 v 40** | 897 | 617 | 463 | 402 | 400 |
| **80 v 80** | 2516 | 1613 | 1600 | 1600 | 1600 |
| **100 v 100** | 3409 | 2500 | 2500 | 2500 | 2500 |
| **120 v 120** | 4348 | 3600 | 3600 | 3600 | 3600 |

**2. Ratio-wise : 70%-30%**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Defect Percentage** | | | | | |  |
|  | **5%** | **10%** | **15%** | **20%** | **25%** |
| **10 v 10** | 81 | 69 | 65 | 56 | 52 |
| **20 v 20** | 297 | 244 | 203 | 177 | 150 |
| **40 v 40** | 996 | 730 | 569 | 457 | 406 |
| **80 v 80** | 2979 | 1924 | 1600 | 1617 | 1600 |
| **100 v 100** | 4107 | 2573 | 2500 | 2500 | 2500 |
| **120 v 120** | 5319 | 3600 | 3600 | 3600 | 3600 |

**3. Ratio-wise : 80%-20%**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Defect Percentage** | | | | | |  |
|  | **5%** | **10%** | **15%** | **20%** | **25%** |
| **10 v 10** | 81 | 81 | 69 | 65 | 61 |
| **20 v 20** | 325 | 275 | 243 | 217 | 193 |
| **40 v 40** | 1141 | 887 | 734 | 608 | 522 |
| **80 v 80** | 3607 | 2534 | 1896 | 1617 | 1600 |
| **100 v 100** | 5145 | 3439 | 2548 | 2500 | 2500 |
| **120 v 120** | 6775 | 4383 | 3600 | 3600 | 3600 |

**4. Ratio-wise : 90%-10%**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Defect Percentage** | | | | | |  |
|  | **5%** | **10%** | **15%** | **20%** | **25%** |
| **10 v 10** | 100 | 81 | 81 | 81 | 81 |
| **20 v 20** | 361 | 324 | 296 | 276 | 257 |
| **40 v 40** | 1317 | 1131 | 1002 | 885 | 802 |
| **80 v 80** | 4561 | 3656 | 2977 | 2498 | 2175 |
| **100 v 100** | 6744 | 5156 | 4131 | 3419 | 2914 |
| **120 v 120** | 9158 | 6785 | 5308 | 4329 | 3663 |

**5.4.3 Advantages**

1. Same number of rows and columns are produced, therefore same of number of inputs and outputs can be used.
2. Better than the previous algorithm

**5.4.4 Disadvantages**

1. Still some defects are exists in the circuit

**6. To consider multiple type of defects simultaneously, developed a defect-free solution using Maximum Rectangle Problem**

**6.1 Introduction**

Now multiple type of defects (stuck-open, stuck-fault) have to be considered in one arena. A rectangle which is defect free to be obtained and the area of rectangle will be maximum. The job has to be done iteratively, so that a rectangle will be obtained which can tolerate the defect.

**6.2 Description**

For solving such problem, maximum rectangle problem solving method is taken according to [9]. The algorithm shows, there are certain number of points given within a rectangle and it finds a rectangle from the given rectangle such that no point should be there within the rectangle. Now in case of nanoscale crossbar architecture, the defect points are taken as those points in maximal empty rectangle. In first iteration, the defect-free maximal empty rectangle is determined and it is excluded from the nanoscale crossbar architecture. Then again the same algorithm is executed to the remaining part of the rectangle. This iteration goes on until a very small maximum empty rectangle is obtained which can tolerate the defects.

**6.3 Algorithm**

Now the algorithm to find maximum empty rectangle in [9] has been applied here with some modifications in some steps. The algorithm is:

**Input :** Size of crossbar circuit and defect percentage in the circuit

**Output :** MAXR, the area of the Maximum Empty Rectangle

**Assumptions**

Here defect points are generated randomly.

Number of defect points, n = (Size of crossbar circuit)\*(defect percentage in the circuit)/100.

Set of defect points,S = {P1,P2,....,Pn}.

Coordinate of Pi=(Xi,Yi).

Boundary of the crossbar circuit is:

At:Top boundary

Ab:Bottom boundary

Al:Left boundary

Ar:Right boundary

It is assumed that,Ab=Al=0 (X-axis and Y-axis respectively)

Therefore,At=Ab+Size of crossbar circuit.

And Ar=Al+Size of crossbar circuit.

**Method**

1. Let MGAP be the maximum gap in {Al,Ar,X1,X2, . . . ,Xn}.

2. MAXR = MGAP\*(At-Ab).

3. Sort S according to the Y coordinates of the points in descending order.

4. For i= 1 to n do steps 5-8.

5. Tl=Al, Tr=Ar.

6. For j=i+ 1 to n do step 7.

7. If Tl<Xj<Tr

Then do steps 7.1-7.2.

7.1 MAXR=MAX(MAXR,(Tr-Tl)\*(Yi-Yj)).

7.2If Xj>Xi

then Tr=Xj

else Tl=Xj

8.MAXR=MAX(MAXR,(Tr-Tl)\*(Yi-Ab)).

9.For i = 1 to n do steps 10-12.

10.Ri=MIN(Ar∪{Xj|(Xj,Yj)∈S,Yj>Yi and Xj>Xi}).

11.Li=MAX(Al∪{Xj|(Xj,Yj)∈S,Yj>Yi and Xj<Xi}).

12.MAXR=MAX(MAXR,(Ri-Li)\*(At-Yi)).

**6.4 Our modification:**

Now the above algorithm has been further modified here.This algorithm is iterated and at each iteration we get an MER and exclude the MER from the nanoscale crossbar circuit.This iteration is done until we get ultimately a small MER (5x5 for convenience).

**Time Complexity:**

Time complexity of above algorithm is O(n2). After the above modification it will be O(n3).

**6.5 Advantages**

1. It is more reliable than speculating each defect point at a time.

2. It excludes a region rather excluding some defect points.

**6.6 Disadvantages**

1. It has higher time complexity.

2. It does not differentiate stuck-open and stuck-close defects.

**7. To obtain a convex polygon that contains all the defect points using Graham Scan Algorithm:**

**7.1 Introduction**

The objective of this algorithm is to find a set of points which forms the smallest convex polygon that contains all the points from a given set of error points denoted by Q. The |Q| defines the total number of error points. This algorithm uses Graham Scan approach to find the convex hull of the set. After finding the defected region using complement of the set we can use the defect-free area to implement the logical functions.

**7.2 Algorithm: Defected\_Area(Q)**

**Input:** a set of error points (Q)

**Output:** vertices of a convex polygon (a defected area)

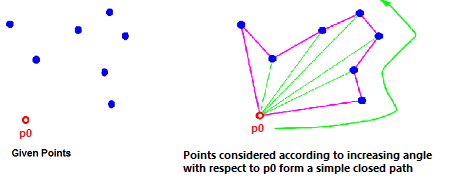
**Pseudo-code:**

1. Let Q[0,1,2,.....n-1] be the input array.
2. Let p0 be the point in Q with minimum ordinate, or with minimum abscissa in case of a tie.
3. Consider the remaining n-1 points and sort them by polar angle in counterclockwise order around p0. If polar angle of two points is same, then put the nearest point first.
4. After sorting, check if two or more points have same angle. If two more points have same angle, then remove all same angle points except the point farthest from p0. Let the new array be <p0,p1,p2,........,pm-1>
5. If m is less than 3, return (Convex Hull not possible).
6. Let S be an empty stack.
7. S.**PUSH**(p0);
8. S.**PUSH**(p1);
9. S.**PUSH**(p2);
10. **For** i=3 to m
11. **While** orientation(Next\_to\_Top(S),Top(S) and pi) is clockwise
12. S.**POP**();
13. S.**PUSH**(pi);
14. **RETURN** S.

**7.3 EXPLANATION:**

The above algorithm can be divided into two phases.

***Phase I (Sort Points)****:* We first find the bottom-most point. The idea is to pre-process points be sorting them with respect to the bottom-most point. Once the points are sorted, they form a simple closed path (See following diagram).



***Phase II (Accept or Reject points)****:* Once we have the closed path, the next step is to traverse the path and remove concave points on this path. How to decide which point to remove and which to keep?Orientation helps here. Orientations means whether the point is clockwise or counterclockwise. Orientation depends on the sign of the below expression.

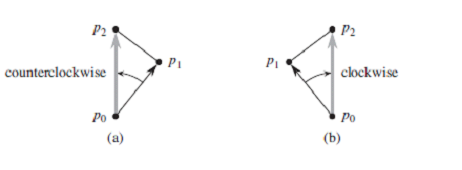
U = (y2 - y1)\*(x3 - x2) - (y3 - y2)\*(x2 - x1)

where p0(x1,y1),p1(x2,y2) and p2(x3,y3) are three consecutive sorted points with respect to their polar angle.

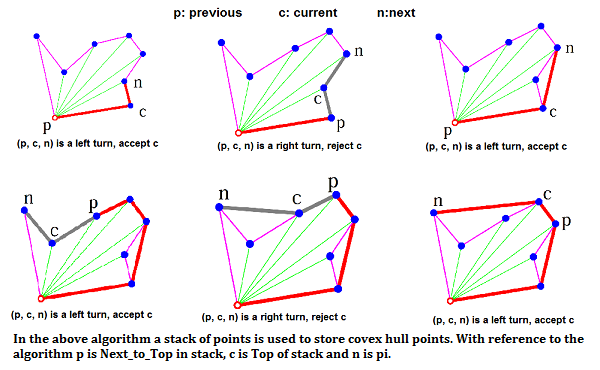
If U<0 orientation is counterclockwise.

If U=0 orientation is collinear(In our case this does not happen).

If U>0 orientation is clockwise.



The first two points in sorted array are always part of Convex Hull. For remaining points, we keep track of recent three points, and find the angle formed by them. Let the three points be prev(p), curr(c) and next(n). If orientation of these points (considering them in same order) is not counterclockwise, we discard c, otherwise we keep it. Following diagram shows step by step process of this phase.



**7.4 Time Complexity:**

Let n be the number of input points. The algorithm takes O(n log n) time if we use an O(n log n) sorting algorithm. The first step (finding the bottom-most point) takes O(n) time. The second step (sorting points) takes O(n log n) time. Third step takes O(n) time. In third step, every element is pushed and popped at most one time. So the sixth step to process points one by one takes O(n) time, assuming that the stack operations take O(1) time.

Overall complexity is O(n) + O(n log n) + O(n) + O(n) which is O(n log n).

**7.5 Advantages:**

1. The runtime of this algorithm is far better than the previous algorithms.
2. The defect free area we get by this algorithm (after complementation of convex-hull) is larger than the previous algorithms.

**7.6 Disadvantages:**

1. In this algorithm we assumed that the defects are always in clusters. So, in case of scattered defects this algorithm would not be able to give proper defect-free region. Or if we find different regions of clusters, we can apply the algorithm in different regions and enclosed all the defect points in different convex sub-hulls.
2. There are some functional parts are enclosed in the polygon.

**7.7 Implementation and Results:**

We have taken different set of error points considering a crossbar with fixed size (100x100). And we shuffled the defect percentage in each case to compare the algorithm 1 (proposed by M.B. Tahoori) and this algorithm. In case of this algorithm the area for the defected zone is calculated using the following formula:

**Area = Mod of[(x1y2-y1x2) + (x2y3-y2x3) +.......+(xny1-y1xn)]/2**

where {(x1,y1),(x2,y2),(x3,y3),..........,(xn,yn)} are the co-ordinates of the points of the calculated polygon with n vertices.

The results regarding the data is given in the following chart:

|  |  |  |
| --- | --- | --- |
| **Defect Rate(d%)** | **Area for algorithm 1** | **Area for this algorithm** |
| 5 | 180 | 480 |
| 10 | 165 | 410 |
| 15 | 147 | 350 |
| 20 | 125 | 310 |
| 25 | 110 | 240 |
| 30 | 100 | 198 |
| 35 | 80 | 150 |
| 40 | 65 | 120 |
| 45 | 47 | 90 |
| 50 | 35 | 50 |

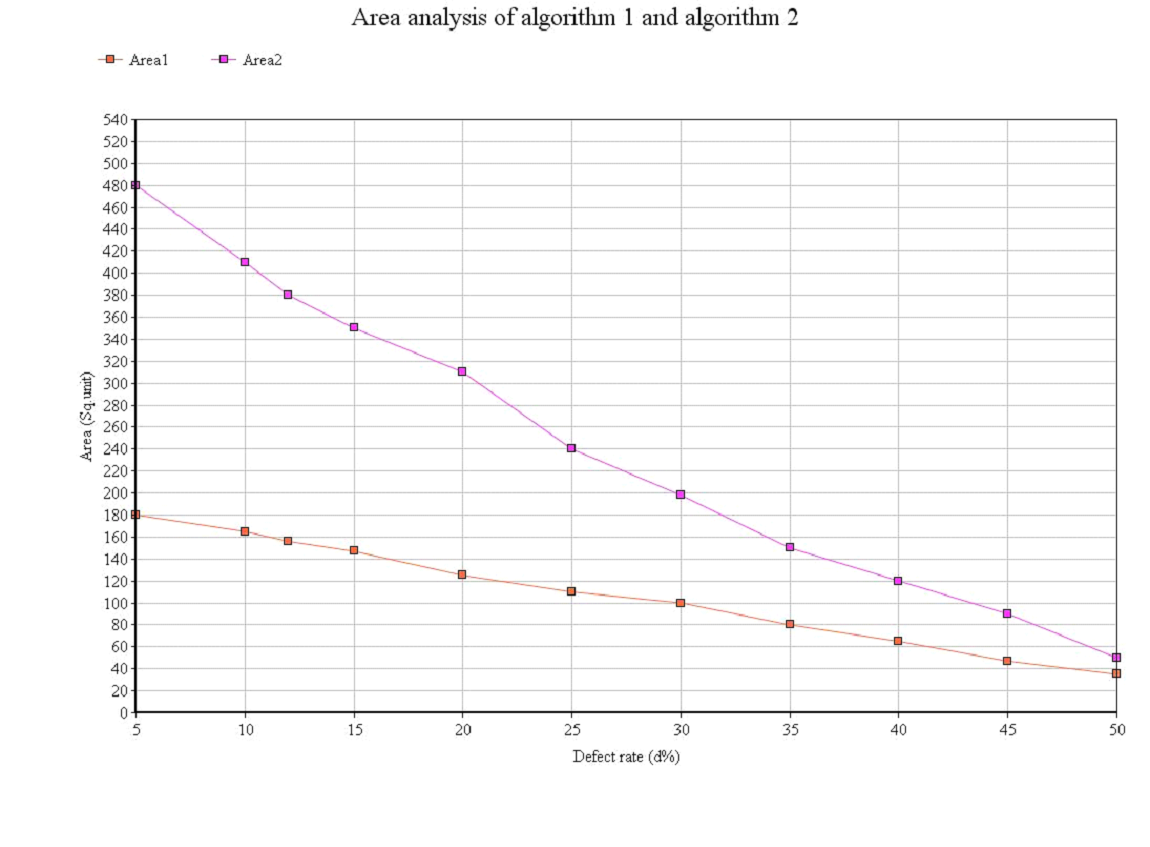


Table describes the dataset of inputs and outputs of the proposed algorithm. The table is created with the given Data input as described above and related to the graph.

We can see from the graph that area2 calculated from proposed algorithm shows a higher growth than area1 calculated from algorithm by M.B. Tahoori with decreasing defect rate.

**8. CONCLUSION**

This paper proposed an approximation algorithm by K-Map for finding the maximum defect-free junctions. 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. This has a direct impact on the cost per chip. The comparison of the proposed algorithm with the exact algorithm was not delineated because for even small crossbars (size 64 64), the exact algorithm is completely intractable. Also, the proposed algorithm yields better results than which gave close results to the exact algorithm. Possible improvements to the given algorithm might include making it faster. Another improvement could be to find an algorithm which can be parallelized optimally.

**9. REFERENCES**

1. J. Huang, M. B. Tahoori, and F. Lombardi, “***On the defect tolerance of nanoscale two-dimensional crossbars***,” in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Syst.*, 2004
2. M. M. Ziegler and M. R. Stan, “***Design and analysis of crossbar circuits for molecular nanoelectronics***,” in Proc. IEEE Int. Conf. Nanotechnology, 2002
3. 3.Ahmad A. Al-Yamani, Sundarkumar Ramsundar and Dhiraj K. Pradhan, “***A Defect Tolerance Scheme for Nanotechnology Circuits***”, in Proc. IEEE Int. Conf. 2007
4. 4.Bo Yuan and Bin Li, University of Science and Technology of China, “***A fast extraction Algorithm for Defect-Free Subcrossbar in Nanoelectronic Crossbar***”, ACM Journal Emerging Technology Computer System, April 2014
5. Malay Kule, Hafizur Rahaman, Bhargab B. Bhattacharya, “***On Finding a Defect-free Component in Nanoscale Crossbar Circuits***”, 4th International Conference on Eco-friendly Computing and Communication Systems (ICECCS 2015)
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. M. B. Tahoori, “***A mapping algorithm for defect-tolerance of reconfigurable nano architectures***,” in Proc. Int. Conf. Computer-Aided Design, 2005, pp. 668–672.
8. H. W. Kuhn, “***The Hungarian method for the assignment problem***,” Naval Res. Logistic Q., vol. 2, pp. 83–97, 1955.
9. A. NAAMAD,D.T. LEE, **“ON THE MAXIMUM EMPTY RECTANGLE PROBLEM”**

**10. “Efficient Function Mapping in Nanoscale Crossbar Architecture “** Joon-Sung Yang and Rudrajit Datta Computer Engineering Research Center The University of Texas at Austin.

**11. ”Runtime Analysis for Defect-tolerant Logic Mapping on Nanoscale Crossbar Architectures”** Yehua Su and Wenjing Rao