Non-square Meshes for Improved Yield in Nanotechnology Circuits

Costas Argyrides , S. Ramsundar[+], Ahmad Al-Yamani* , Dhiraj Pradhan
Bristol University, Department of Computer Science, Bristol, UK
KFUPM, Department of Electrical Engineering, Saudi Arabia
{costas, pradhan @cs.bris.ac.uk}, sundak@iitg.ernet.in[+], aaa@ieee.org*

Abstract – In this paper we propose a novel algorithm for non-square meshes. In best of our knowledge all algorithms that have been proposed till now, are all for square and non-rectangular meshes. We did several experiments for different defect rates for different Crossbars sizes. The new effective technique will provide cost free big square or rectangular circuits. The use of rectangular meshes allows us to construct the cost free circuits in the high defect rate conditions. The savings in yield may reach 1000% or even 3000% in some cases.

I. Introduction

As lithography-based silicon VLSI technology is expected to reach its limit, nanotechnology-based fabrication is emerging as an alternative. It is estimated that molecular electronics can achieve very high densities (10^{12} devices per cm^2) and can 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 nanotubes (CNT) [2], silicon nano-wires (NW) [3,4], and quantum dot cells [5].

It has been shown that using self-assembly techniques, it is possible to overcome certain technological limitations posed by lithography for the smallest feature size [2]. 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 [7]. The crossbar plays the key role in these architectures, as it can be used as programmable logic array (PLA) planes 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. Such high defect rates, fault-tolerance methods have to be employed. Various research groups are working on problems related to defect tolerance in nanotechnology [7-14].

A simple calculation shows that even a moderate sized crossbar with a high defect rates would have at least one defect on each of its nanowires. Discarding all parts with any defects is not cost-effective any more as the yield hit will be substantial, resulting in very high production costs. Therefore, we need to learn how to bypass these defects to make computation possible. Given a desired crossbar, our approach is to get a minimum-cost crossbar size that can be used to construct a defect-free subset, connecting such subsets together using the technique described in [22]. Given a defective crossbar, we need to maximize the defect-free area and use it to its maximum potential while reducing the yield loss and increasing the profit.

The rest of the paper is organized as follows. Section II reviews the previous approaches. A new algorithm is proposed in Section III, results and cost analyses are depicted in Section IV. Finally, Section V concludes the paper.

II. Previous Work

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 impact 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 bipartite graph model of the crossbar. Hence, the time complexity of the algorithm is reduced by trading off its accuracy.

2.1 Defects-Unaware Design Flow

In [12], 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 $n \times n$. 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, we map 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.

2.1.1 Requirements

a) 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 sub graph problem in the representative graph of the $n \times n$ crossbar.

b) 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 guarantee that can fulfil 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.

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

2.1.2 Advantages

a) 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, it's universal for all the chips manufactured, as all the chips are assumed to be built on a $k \times k$ defect-free crossbar $(k < n)$.

b) 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 $O(n)$.

2.1.3 Disadvantages

a) Out of the $(n-k) \times (n-k)$ cross points 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 proposed Algorithm could be done only for 16x16 and 32x32 crossbars with defect densities up to 30%.

The algorithm given in [12] constructs the bipartite complement of the representative graph, and then tries to achieve the approximate maximum 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.

III. Rect Algorithm

We propose a greedy algorithm for the maximum balanced bipartite sub graph problem for minimum cost. 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 by-product 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 Sub graph problem as $\min(|U|,|V|) \times \min(|U|,|V|)$ biclique. But, this approch did not give any improvement over the previous results known [12].

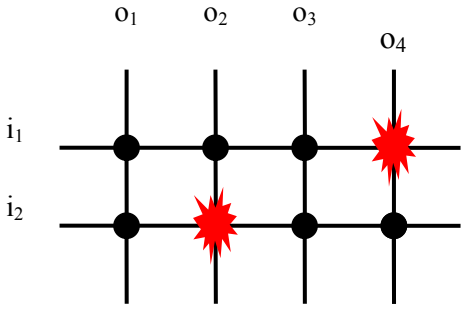

Figure 1: 2x4 Crossbar with defects

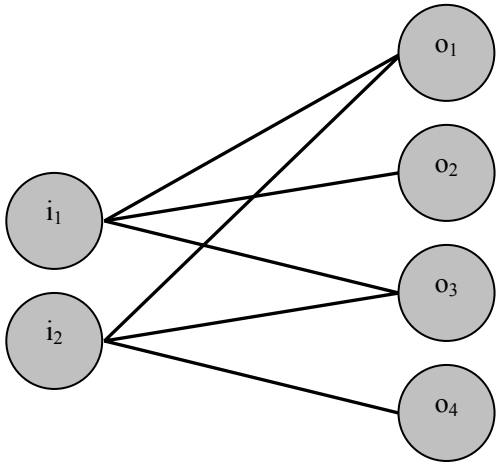

Figure 2: Representative graph of crossbar in Figure 1

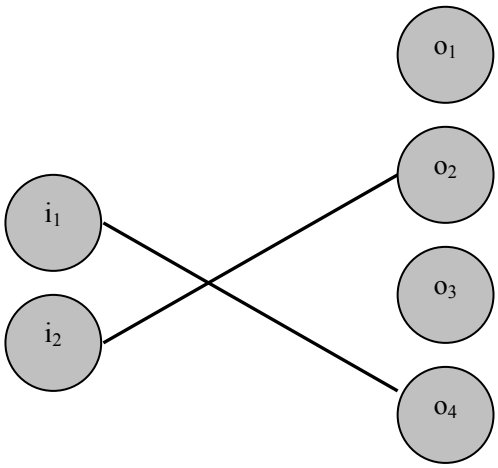

Figure 3: The bipartite complement of Figure 2

Conceptually, Algorithm in [12] tries to reduce the number of edges in the graph, while our Algorithm tries to reduce the degree of the least-degree nodes in a partition.

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 Figure 4 to be the complement graph (the graph representing the defects).

Now, if we had to remove a node from the left partition, Algorithm in [12] would remove *E* since they have the highest degree of *3* in the left partition. But, our Algorithm would remove *F,* since it is the node with the highest degree and connected to a node with the minimum degree node on the right partition (in this case A). As soon as *F* is removed, A can be included in the independent set, while removal of *E* does not immediately result in a node being included into the independent set.

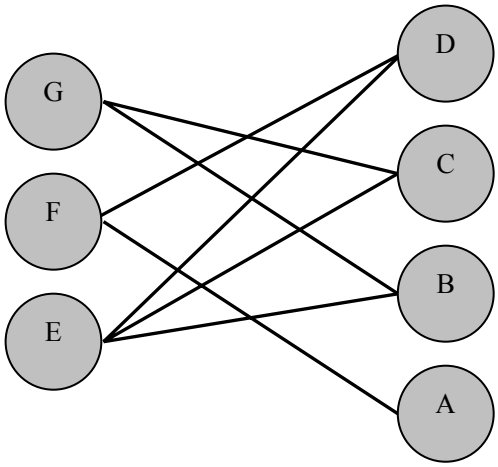

Figure 4: Bipartite graph

Rect Algorithm, Function Biclique $(G(U,V,E))$

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

As in Algorithm in [12], the bipartite complement is constructed and then we alternate between *U* and *V* to remove

A 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.

In our, Rect Algorithm 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.

IV. Cost Analysis

The way that nanocircuits are fabricated is described in [22]. Smaller crossbars are assembled together to create a bigger circuit. For a given number of inputs and outputs we analyze different configurations to achieve the minimum cost given a defect rate. For example a 20x20 circuit may be implemented using 16 of 5x5, 4 of 10x10 etc. Using our approach this can be done using 4 of 5x20 or 8 of 5x10 etc. Firstly we analysed the yield loss using square meshes and then we compared the results with our approach with different configurations. In Table 1 one can see the results of the analysis using algorithm [12] for producing big circuits using square meshes for defect rate 5%. The yield loss is calculated using formula (1)

Table 1: Yield Loss using different Nano-block sizes using square crossbars (5% defect rate)

Nano-block Size $(k \cdot k)$	5	10	15	20	25	30	40
Min. Required size $(n \cdot n)$	8	18	28	40	54	70	110
Yield lost (%)	0	25	36	56	81	226	285

$$Yield\ loss\,\% = \frac{{n_2}^2 \cdot 100}{\left(\frac{k_2}{k_1}\right)^2 \cdot {n_1}^2} - 100 \qquad \textbf{(1)}$$

Where k_1 is the desired nano-block, n_1 is the minimum sized nano block required to produce k_1, k_2 is the next desired nano-block and n_2 the minium sized nanoblock required to produce k_2. (Note that in formula one we calculate the defect rate of different square meshes.)

In figure 5 one can see the yield loss while creating a 20X40 chip using a 5x5(less expensive using algorithm in [12]), 10x10 and 20x20 crossbar meshes and our technique using various non-square meshes in 5% defect rate.

$$Yield\ loss\,\% = \frac{(k_a * k_b) * n^2 * 100}{(n_a * n_b) * k^2} - 100 \qquad \textbf{(2)}$$

Where $n_a * n_b$, is the minimum sized nano block required to produce $k_a * k_b$ mesh size and n is the minimum desired nano-block mesh required to produce k size meshes. It is important to note that this formula is desired circuit size free. The reason that we are using 20x40 circuit in the requirements is to restrict the number of available mesh sizes produce by our algorithm.

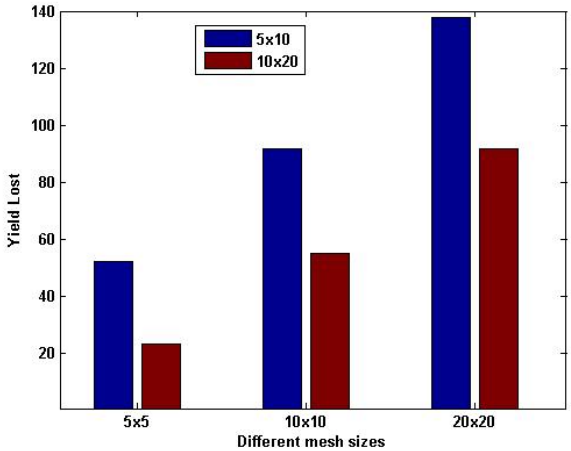

Figure 5: Yield loss using Non-Square meshes in 5% defect rate

It is known that with the advent of new technology the defect rate is high, In table 2 we are presenting an analysis for the yield loss in different defect rates using only square meshes. Formula (1) was used to calculate the yield loss.

Table 2: Yield Loss using different Nano-block sizes and different defect rates

k	Defect Rate					
	15%		20%		25%	
	n	%	n	%	n	%
5	10	0	12	0	14	0
10	28	196	37	238	57	314
15	57	361	98	741	178	1796
20	119	885	199	1718	528	8889

In figure 6 and figure 7 we are creating a 20x40 circuit using different squarer meshes from table 2 and compared to 5x10 and 10x20 respectively.

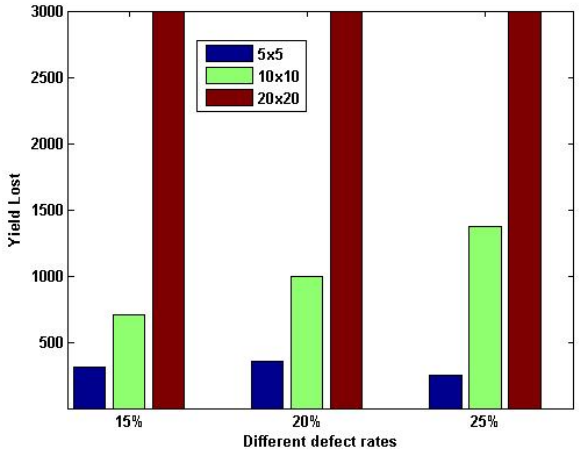

Figure 6: Yield Lost using different square meshes over 5x10 mesh

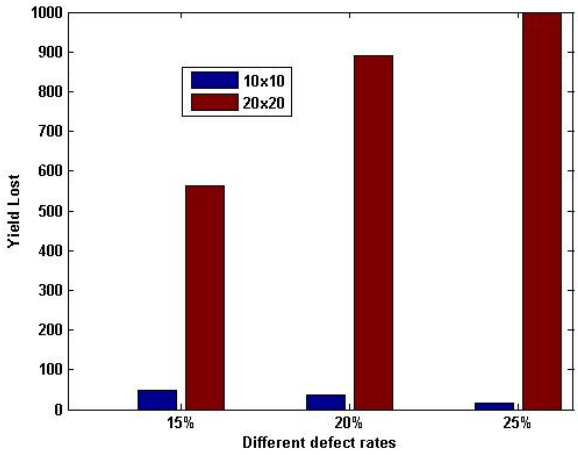

Figure 7: Yield Lost using different square meshes over 10x20 mesh

V. Conclusions

An effective technique to reduce chip size and improve the yield in nanotechnology circuits is proposed. Using the proposed technique yield loss will be decreased significantly. A reduction of 120% was shown in 5% defects per chip. In higher defect rates our approach may achieve a reduction in lost yield more than 3000% (figure 6) using 5x10 mesh and more than 1000% using 10x20 mesh.

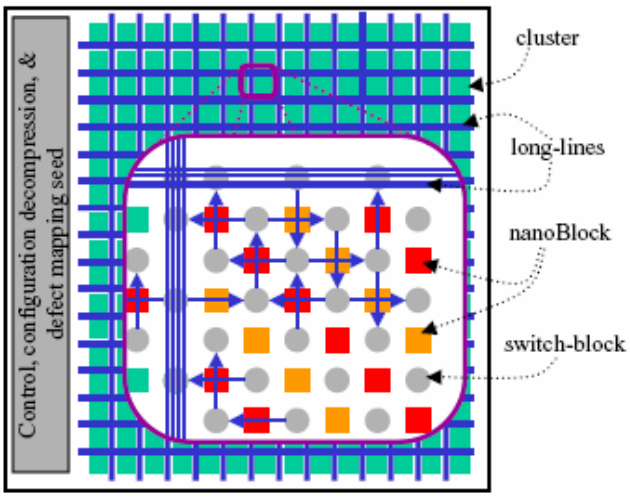

Figure 8: nanoFabric Layout [23]

References

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

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

[3] Y. Cui and 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, and C. M. Leiber, "Logic gates and computation from assembled nanowire building blocks", Science, Vol. 294, pp. 1313-1317, 2001.

[5] P. D. Tougaw and 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 molecularswitch crossbar circuits", IEEE Trans. on Nanotechnology, Vol. 14, pp. 462-468, 2003.

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

[8] H. Naeimi and A. De Hon, "A greedy algorithm for tolerating defective crosspoints in NanoPLA desing", IEEE International Conference on Field-Programmable Tehnology, 2004.

[9] A. De Hon, "Array-based architectures for FET-based, nanoscale electronics", IEEE Trans. on Nanotechnology,

Vol. 2, No. 1 pp. 23-32, 2003.
[10] Andre DeHon and M. J. Wilson, "Nanowire-based sublithographic programmable logic arrays", International Symposium on Field-Programmable Gate Arrays (FPGA 2004), pp. 123-132, 2004.
[11] S. C. Goldstein and M. Budiu, "Nanofabrics: spatial computing using molecular electronics", International Symposium on Computer Architecture, 2001.
[12] M. B. Tahoori, "A mapping algorithm for defect-tolerance of recon_gurable nano-architectures", International Conference on Computer-Aided Design, pp. 668-672, 2005.
[13] G. Snider, P. Kuekes, and R. S. Williams, "CMOS like logic in defective, nanoscale crossbars", IEEE Trans. on Nanotechnology, Vol. 15, pp. 881-891, June 2004.
[14] M. M. Ziegler and M.R. Stan, "Design and analysis of crossbar circuits for molecular nanoelectronics", 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] T. H. Cormen, " Introduction to algorithms", MIT Press, Cambridge, MA. , 2001
[18] U. Feige and S. Kogan, "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, 2004.
[19] M. Dawande, P. Keskinocak, J. M. Swaminathan, and S. Tayur, "On bipartite and multiparite clique problems", Journal of Algorithms, 41-2, 388-403, 2001.
[20] D. S. Hochbaum, "Approximating clique and biclique problems", Journal of Algorithms, 29-1, 174-200, 1998.
[21] H. W. Kuhn, "The Hungarian method for the assignment problem", Naval Research Logistic Quarterly 2, pp. 83-97, 1955.
[22] S. C. Goldstein and M. Budiu "NanoFabrics: Spatial Computing Using Molecular Electronics", 28th Annual International Symposium on Computer Architecture, June 2001.
[23] Seth Copen Goldstein and Mihai Budiu "NanoFabrics: Spatial Computing Using Molecular Electronics", in Proc. of The 28th Annual International Symposium on Computer Architecture, June 2001