

Eliminate Wire Crossings by Node Duplication

Jian Li

June 13, 2008

Fudan University, Department of Computer Science and Engineering
Shanghai 200433, China

Abstract

For many circuit design problem, it is imperative to carefully study the effect of physical implementation constraint. Under some condition, it is very difficult to fabricate wire crossings. In this paper, we introduce a crossing elimination model based on a node duplication method and we want to minimize the number of duplication. We relate it with an artificial problem, called maximum simple sharing problem. First we prove it is NP-hard, then we show a simple greedy algorithm can achieve a approximation factor of 3. We introduce maximum disjoint simple sharing problem, which is naturally a 2-approximation of the maximum simple sharing problem, and show it can be solve optimally by reducing to the perfect matching problem in a series carefully constructed graph. At last, we further improve the approximation factor to $12/7$ through a local search technique.

1 Introduction

In some circuit design problem, wire crossings are forbidden or to implement wire crossings is very hard or expensive. For example in the molecular quantum dot cellular automata(QCA) circuit[1, 3, 4], it is very difficult for chemists to fabricate wire crossings(at least in the near to midterm)[5]. QCA circuits perform logical operations and data movement via Coulombic interaction rather than electric flow. As a result, they are able to overcome two obstacles to developing logic gates at the nanoscale that molecular transistors (based on traditional CMOS technology) will not be able to do: Achieving low power dissipation and giving a natural way to connect devices at the molecular scale. Therefore, there has been a great deal of research efforts on developing QCA devices and circuits that are capable of facilitating the implementation of real QCA circuits and systems (e.g., see [1, 2, 3, 4]). Eliminating wire crossing also finds applications in physical synthesis of Binary Decision Diagram(BDD). None-Crossing Ordered BDD are used to overcome the timing closure problem by Cao and Koh in [6]. Many other works focus on design the embedding of the circuit to minimize the number of crossing, such as [9, 10].

In our model, the circuit can be presented as a bipartite graph, $G(U, V; E)$. The data should be transferred from nodes in U to nodes in V . Nodes in U (resp V) are arranged in

a row. The wire must be embedded in the space between the row of U and the row of V . We can eliminate wire crossings by node duplications on one color class U of G . There is another constraint: each duplication of one node has connection limitation, that it can connect only one node in V while each original node can connect arbitrarily many nodes in V . This forced constraint is required by many practical situations since the function of the duplication may be rather restricted sometimes [5]. To avoid wire crossings in the layout of G , we can choose an order of the vertices in U and V . Figure 1(a)(b) is an illustration of our crossing elimination method. The node duplication is expensive usually, so naturally we aim at minimize the number of node duplications.

The role of our wire crossing elimination method plays in the emerging QCA technology is similar to the key role played by the widely researched *crossing minimization problem* in traditional VLSI technologies. The crossing minimization problem aims to minimize wire crossings in a planar circuit layout [6, 12, 13, 14].

In this paper, we introduce the maximum simple sharing(MSS) problem, and show to minimize the number of node duplications in our wire crossing elimination method is in fact equivalent to MSS problem. So we focus on the MSS problem in the main part of the paper. First, we prove the NP-hardness of the problem. So one promising way to tackle the problem is to design polynomial time approximation algorithms. Recall for maximization problem, we call it has a α -approximation, if its optimal solution is at most α times the solution our polynomial time algorithm produce[8].Then we give a rather simple greedy algorithm and show it can guarantee a approximation factor of 3. Next, we introduce maximum disjoint simple sharing problem, which is naturally a 2-approximation of the MSS problem. Interestingly, we show it can be solve optimally. The work is done by reducing it to perfect matching problem in a series carefully constructed graph. Based on the maximum disjoint simple sharing, we further improve the approximation factor of MSS problem to 12/7 through a local search technique.

It is useful to note that our problem is a variant of the node-duplication based crossing elimination problem(NDCE) in [5]. In their model, duplicated nodes have the same connection ability as the original nodes. The NDCE problem is closely related to the so-called maximum sharing problem[5, 11]. The maximum sharing problem is also NP-hard. But with respect to approximation algorithm, the maximum sharing problem and maximum simple sharing problem are quit different. The maximum sharing problem can be approximated by using a path cover technique and it admits a 3/2-approximation[11].

2 Minimizing Node Duplication and MSS Problem

In this section, we show to minimize the number of node duplications is in fact equivalent to the following *maximum simple sharing(MSS) problem*. Given a bipartite graph $G(U, V; E)$, a simple sharing is two adjacent edges (u, v) and (u, w) where $u \in U$ and $v, w \in V$. Let $((u, v)(u, w))$ denote this simple sharing. Note that a simple path of length k with both of its ending points in V are composed of $\frac{k}{2}$ simple sharings. We want to find a series vertex disjoint simple paths with their ending points in V and maximize the number of simple sharings contained in these paths. We call a simple path with both of its ending points in V a *path* for convenience. Figure 1(c) shows a two simple sharings in

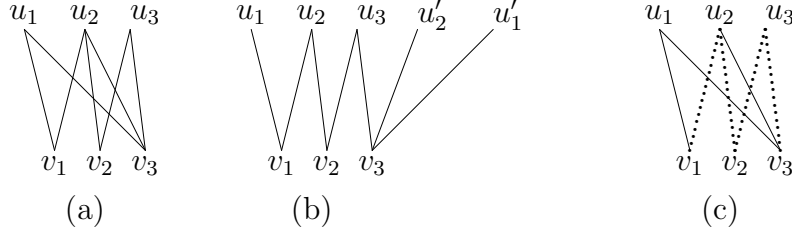


Figure 1: (a)Original bipartite graph G .(b)The layout of G . We duplicate u_1 and u_2 to eliminate all the wire crossings.(c)Dotted lines is a simple path which contains two simple sharings

the graph.

W.l.o.g, we assume in G , every node has degree at least 2 in G .

Theorem 1 *Given a bipartite graph $G(U, V; E)$ which every node has degree at least two, there is a solution of MSS problem which contain m simple sharings, if and only if we can duplicate $|E| - |U| - m$ nodes to eliminate all wire crossings.*

Proof: Denote the layout of the circuit without wire crossings $G'(U', V'; E')$. U' is composed of $|U|$ original nodes and newly duplicated nodes.

First, we can see the job can be done by trivially duplicating $|E| - |U|$ nodes, that is every edge has a distinct corresponding nodes in U' . How can we do better? The idea is original nodes(not duplicated nodes) in U can connect to more than one nodes, thus reducing the duplication number.

Consider a permutation of nodes in V' , let v_i and v_{i+1} be a pair of consecutive nodes. it is easy to see v_i and v_{i+1} can have at most one common neighbor in U' , otherwise there would be some wire crossings. It also can be seen that in the layout G' , the degree of a node in U' cannot be bigger than 2, otherwise edge crossing can not be avoided by noting that we cannot duplicate nodes in V

In G , if there are m simple sharings, we can arrange m pairs of nodes in V consecutively so that each pair of nodes share a common neighbor in U , so we can reduce the duplication number by m . The other direction can be proved by a similar argument. \square

For example, see figure 2(c), there are two simple sharing in G , so we can duplicate $|E| - |U| - m = 7 - 4 - 2 = 2$ nodes to eliminate all wire crossings. If there is some node with degree one, it is easy to see we can insert it somewhere without incurring any crossing. So to minimize the number of duplications, we need to maximize the number of simple sharings.

Now, the equivalence of the minimization of node duplication and maximum simple sharing problem is established. In the following section, we will focus on the MSS problem. We call vertices in U *upper vertices* and vertices in V *lower vertices*.

3 Maximum Simple Sharing Problem

First we prove maximum simple sharing problem is a NP-hard problem. Then we give a simple greedy algorithm which can achieve a 3-approximation. Then, we introduce a new problem, called maximum disjoint simple sharing problem, whose optimal solution is a 2-approximation for maximum simple sharing problem. We show maximum disjoint simple sharing problem can be solved optimally in polynomial time. Finally, we further improve the approximation ratio to $\frac{12}{7}$ by combining the 2-approximation with a local search technique.

3.1 NP-hardness of MSS problem

Theorem 2 *Maximum simple sharing problem is NP-hard.*

Proof: The decision version of the problem is given a bipartite graph $G(U, V; E)$ and a parameter k , is there a series vertex disjoint simple paths that contain at least k simple sharings.

We show the a reduction from a well-know NP-complete problem, Hamitonian path problem[7]: Given a graph $G(V, E)$, is there a simple path go through every vertex in V exactly once?

Now we construct a instance of maximum simple sharing problem, $G'(U, V; E')$. G' is formed as following. Break every edges of G into two edges and introduce a new node connecting them. Let V be the set of original nodes in G and U be the newly introduced nodes. It can be see G' is a bipartite graph since all edges have one ending points in U and the other in V . The key observation is G has a hamitonian path if and only if G' has $|V| - 1$ simple sharings. Moreover G' can have at most $|V| - 1$ simple sharings. Set the parameter $k = |V| - 1$, then the proof is completed. □

3.2 A Simple Greedy Approximation Algorithm

We first give out the algorithm and then analysis its approximation guarantee in Theorem 3.

ALGORITHM: GREEDY EXTENSION:

Initially, all vertices are unused.

Do the following until nothing can be done: In the graph induced by all unused vertices, choose an vertex v in V , arbitrarily extend a maximal vertex disjoint path in both direction. Mark the vertices in this path used.

Theorem 3 *GREEDY EXTENSION achieve a approximation factor of 3 for MSS problem*

Proof: When the algorithm terminates, the original vertices set is naturally partitioned into four sets, A, B, C and D , where $U = A \cup B$, $V = C \cup D$ and A and C are sets of used vertices. It is obvious that the subgraph induced by $B \cup D$ does not contain any simple sharing.

Clearly, the number of simple sharings which our algorithm has found is $|A|$. Suppose we have extended m distinct path, so $|C| = |A| + m$. Moreover, every path we have found has two endpoints in C which we will call *external nodes*, other vertices in C *internal nodes* and vertices in D *outside nodes*. Let $OPT(G)$ be an optimal solution for G , and $|OPT(G)|$ be the its value. With a little bit abuse of notation, we use $B \cup C \cup D$ to denote the subgraph induced by $B \cup C \cup D$. It is easy to see that $|OPT(G)| \leq |A| + |OPT(B \cup C \cup D)|$. let $OPT_b = OPT(B \cup C \cup D)$. We claim that $|OPT_b| \leq 2|A| - m$, thus $|OPT| \leq |A| + |OPT_b| \leq 3|A| - m \leq 3|A|$, which completes the proof.

Then the remain thing is to prove the claim that $|OPT_b| \leq 2|U| - m$. Now let us consider a simple sharings of OPT_b . If the simple sharing's two lower points are both external nodes that belongs to a single path our algorithm finds. Then we delete this simple sharing in OPT_b . Note that there are at most m such simple sharings because there are m paths in our solution. Then the leftover edges in OPT_b form several paths. Consider such a path P . We write down the nodes in P sequentially, namely $v_1, u_1, v_2, u_2, v_3, \dots, u_{k-1}, v_k$, where $v_i \in V$ for $1 \leq i \leq k$ and $u_i \in B$ for $1 \leq i \leq k - 1$. This path contributes $k - 1$ simple sharings in B . The key observation is for any $1 \leq i < k$, at least one node of v_i and v_{i+1} is an internal node. Suppose not, the current solution can be further enlarged by adding a simple sharing $((v_i, u_i), (v_{i+1}, u_i))$. So there are at least $\frac{k-1}{2}$ internal nodes in P which implies a internal node can contribute to at most two simple sharing in OPT_b . Since there are $|A| - m$ internal nodes, so OPT_b has at most $2(|A| - m)$ simple sharing plus at most m simple sharing we delete previously. So $OPT_b \leq 2|A| - m$ \square .

We first define a *un-extendable* solution as a solution which cannot be enlarged by simply extending a path from its endpoint without destroying the current solution, otherwise we call it *extendable*. It is easy to see the algorithm GREEDY EXTENSION will obtain a un-extendable solution. It is also useful to note that we can enlarge any solution to an un-extendable solution in polynomial time.

3.3 Maximum Disjoint Simple Sharing Problem

In this subsection, we introduce maximum disjoint simple sharing problem. "Disjoint simple sharing" means the simple sharing must be vertex disjoint, that implies to find a series paths of length 2. See the left hand side of figure 4, the dotted line form a maximum disjoint simple sharing. The objective is similar as before, to maximize the number of disjoint simple sharings. The optimal solution of maximum disjoint simple sharing problem is obvious a 2-approximation of the MSS problem.

Algorithm for Maximum Disjoint Simple Sharing Problem. For every vertex $u \in U$, we split it into two vertices u and u' and add an edge of zero weight linking them. u and u' connect the same set of vertices in V as original, and these edges have weight 0.5. We name the current graph G_0 . then we add to G_0 a vertex w_1 which will connects all the vertices in V with zero weight edges, we call this graph G_1 . By repeatedly adding one such vertex one time we get graph $G_2, G_3, \dots, G_{|V|}$. An example of this construction are illustrated in Figure 2. Run maximum weight perfect matching algorithm in $G_0, G_1, \dots, G_{|V|}$. Suppose G_k has a perfect matching of maximum weight W_k (Note that some G_i cannot be perfectly matched, then let $W_k = 0$). Let $W = \max_k(W_k)$. We claim that the maximum number of disjoint simple sharing is W .

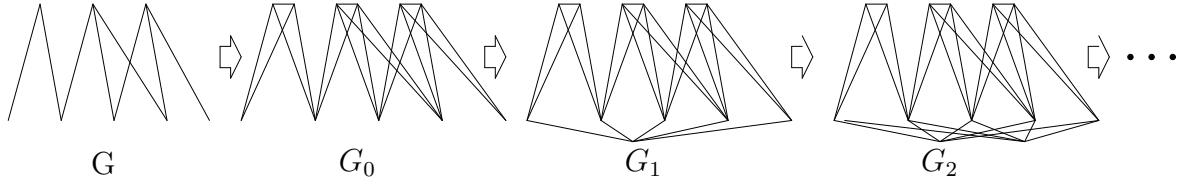


Figure 2: **Construction of G_i**

Lemma 4 if G_i has a perfect matching, nodes u and u' (both of which are obtained from splitting the node u in G) are matched either by edge (u, u') , or by (u, v_1) and (u', v_2) where $v_1, v_2 \in V$ and $v_1 \neq v_2$.

Lemma 5 if G_i has a perfect matching M_i of weight W_i , then G has W_i disjoint simple sharings.

Proof: by Lemma 4, if M_i contain edges (u, v_1) and (u', v_2) where $v_1, v_2 \in V$ and $v_1 \neq v_2$, we construct a simple sharing $((u, v_1), (u, v_2))$ in G . it is easily seen these simple sharings we construct are disjoint and we can get W_i such simple sharings. \square

Lemma 6 if G has W disjoint simple sharing S , then $G_{|V|-2W}$ has a perfect matching M of weight W .

Proof: consider $G_{|V|-2W}$, we construct a perfect matching M from the W disjoint simple sharings in G . If there is a simple sharing $((u, v_1)(u, v_2))$ in G , we add two matching edges (u, v_1) and (u', v_2) to M . if $u \in U$ (in G) doesn't participate any disjoint simple sharing, we add (u, u') to M . Moreover there are still $|V| - 2W$ unmatched vertices in V . They can be matched by the $|V| - 2W$ newly added vertices in $G_{|V|-2W}$. the it is easily seen the edges we add into M are really a matching and have total weight W . \square

Figure 3 shows a example of the correspondence between maximum disjoint simple sharing in G and maximum weight perfect matching in G_1 .

Now, we can conclude the following theorem.

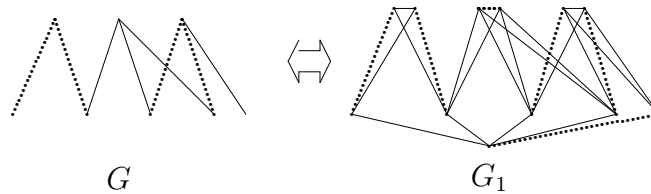


Figure 3: **The correspondence between maximum disjoint simple sharing in G and maximum weight perfect matching in G_1 . Dotted lines in G are a solution of maximum disjoint simple sharing problem, while dotted lines in G_1 form a maximum weight perfect matching.**

Theorem 7 *The maximum disjoint simple sharing problem can be solved optimally in polynomial time.*

Corollary 8 *There is a 2-approximation for MSS problem.*

3.4 Improve the Approximation Factor to $\frac{12}{7}$

Algorithm:LOCAL SEARCH

(Step 1)Initially, the solution is the maximum disjoint simple sharing we have found.

(Step 2)We arbitrarily extend the current solution to a un-extendable solution.

(Step 3)For every singleton simple sharing(a simple sharing doesn't participate in any path of length greater than 2) $((v, u), (v, w))$, for every unused vertex v' in U , we exchange the top vertex v to v' (i.e. the new simple sharing becomes $((v', u)(v', w))$). If the current solution becomes extendable, goto Step 2. If the solution remain un-extendable for any choice of singleton sharing and unused upper vertex, algorithm terminates.

It is easy to see our algorithm runs in polynomial time, since in each iteration we add one more simple sharing, and there are at most $|U|$ simple sharings. All three steps can be obviously implemented in polynomial time.

Theorem 9 *Algorithm LOCAL SEARCH is a $\frac{12}{7}$ approximation for MSS problem.*

Proof:

Let OPT denote the optimal solution of maximum simple sharing problem, OPT_D denote the optimal solution of maximum disjoint simple sharing problem and SOL is the solution we find.

Suppose $|SOL| = (1 + \alpha)|OPT_D|$, that is we add $\alpha|OPT_D|$ more simple sharings to the maximum disjoint simple sharing we have obtained in step 1. It is easy to see in SOL there are at most $2\alpha|OPT_D|$ internal node and at most $\alpha|OPT_D|$ paths of length more than two.

Suppose SOL is a β approximation, that is $|OPT| = \beta|SOL|$, where β is at most 2 obviously.

We refer the set of vertices in U that appear in OPT but don't appear in SOL as $OPT \setminus SOL$ and similarly the set of vertice in U that in our solution SOL but not in OPT as $SOL \setminus OPT$. It is easy to see $|OPT \setminus SOL| = (\beta - 1)|SOL| + |SOL \setminus OPT|$ since $|OPT \setminus SOL| + |SOL| = |SOL \setminus OPT| + |OPT|$.

Now we consider the vertices in $OPT \setminus SOL$ and classify them into four categories. See Figure 4. Recall every path in SOL has two endpoints which we will call *external nodes*, other vertices in V along these paths *internal nodes* and all other vertices in V do not appear in these paths *outside nodes*.

If there is a simple sharing $((u, v)(u, w))$ in OPT , we say u OPT-connects to v and w and visa verse.

Category 1: $u \in OPT \setminus SOL$ OPT-connects to two external nodes both of which are the ending points of a path in SOL whose length is at least 4.

Category 2: $u \in OPT \setminus SOL$ OPT-connects two external nodes both of which are the ending points of a path $P = \{a, b, c\}$ of length 2 in SOL . Moreover b is in $OPT \cap SOL$.

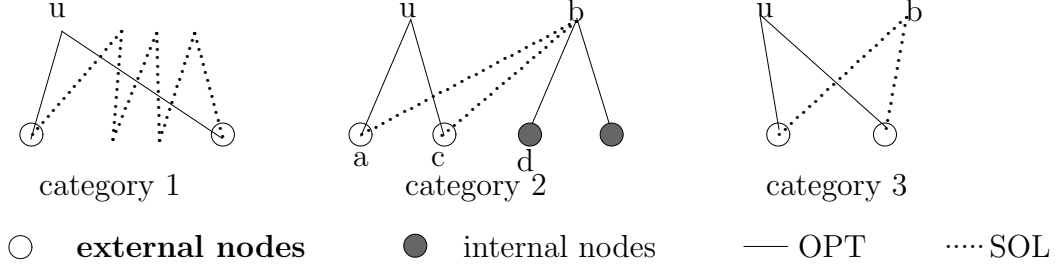


Figure 4: **Vertices in $OPT \setminus SOL$ of first three categories**

we claim that all nodes b connects except a and c are internal nodes. Otherwise suppose b connects a node d which is not a internal node , then in some iteration of our algorithm, we would shift simple sharing $((b, a)(b, c))$ to $((u, a)(u, c))$ (in Step 2) and then we could extend the current solution by adding one more simple sharing $((b, a)(b, d))$, rendering contradiction. So, we can associate u with a internal vertex d which OPT-connects to b . We call we associate u with d through b .

Category 3: $u \in OPT \setminus SOL$ OPT-connects two external nodes both of which are the ending points of a path $P = \{a, b, c\}$ of length 2. moreover b is in $SOL \setminus OPT$.

Category 4: $u \in OPT \setminus SOL$ OPT-connects at least one internal nodes i . We can associate u with i . This time, we call we associate u with i through u .

Note that no other cases exist. This fact can be easily proved through a case by case analysis. For example, suppose the vertex $u \in OPT \setminus SOL$ connects a external node v and an outside node w , we can simply extend the simple sharing by adding $((u, v)(u, w))$.

We use k_i to denote the number of the vertices of category i . We have $k_1 + k_2 + k_3 + k_4 = |OPT \setminus SOL| = (\beta - 1)|SOL| + |SOL \setminus OPT|$. It is easy to see $k_3 \leq |SOL \setminus OPT|$, thus $k_1 + k_2 + k_4 \geq (\beta - 1)|SOL|$. Recall that there are at most $\alpha|OPT_D|$ paths of length more than two, so $k_1 \leq \alpha|OPT_D|$.

Now we mention a useful lemma which can be used to bound the value of $k_2 + k_4$.

Lemma 10 *Each internal node can be associated by at most 2 vertices in $OPT \setminus SOL$*

Proof: Consider a internal node i , it can OPT-connect at most two nodes , say a and b . It is easy to see there are at most one node in $OPT \setminus SOL$ that are associated with i through a (or b). No node in $OPT \setminus SOL$ can be associated with i through other nodes except a and b . \square

Recall that there are at most $2\alpha|OPT_D|$ internal node, and by Lemma 10, we can see $k_2 + k_4 \leq 4\alpha|OPT_D|$. So

$$5\alpha|OPT_D| \geq k_1 + k_2 + k_4 \geq (\beta - 1)|SOL|. \quad (1)$$

Because $|OPT_D| \geq \frac{1}{2}|OPT|$,

$$|SOL| = (1 + \alpha)|OPT_D| \geq \frac{(1 + \alpha)|OPT|}{2} = \frac{(1 + \alpha)\beta|SOL|}{2}. \quad (2)$$

Rearrange (2), we get

$$\beta \leq \frac{2}{(1 + \alpha)}$$

Moreover, by substituting $|SOL|$ in the right hand side of (1) by $(1 + \alpha)|OPT_D|$, we can easily get $5\alpha \geq (\beta - 1)(1 + \alpha)$, so

$$\beta \leq \frac{(6\alpha + 1)}{(1 + \alpha)}.$$

then

$$\beta \leq \min\left\{\frac{(6\alpha + 1)}{(1 + \alpha)}, \frac{2}{(1 + \alpha)} \mid \alpha \geq 0\right\}.$$

the right hand side maximizes at $\alpha = 1/6$, then we obtain $\beta \leq \frac{12}{7}$.

□

4 Conclusion

In this paper, we introduce a node duplication based wire crossing elimination method and show minimizing the number of node duplication is in fact equivalent to the maximum simple sharing problem. Then we prove the NP-hardness of the MSS problem and give several approximation algorithm. The best approximation ratio we have achieved so far is $\frac{12}{7}$. It would be interesting to show any inapproximability result of this problem or improve the approximation ratio significantly.

References

- [1] D.A. Antonelli, D.Z. Chen, T.J. Dysart, X.S. Hu, A.B. Khang, P.M. Kogge, R.C. Murphy, and M.T. Niemier. Quantum-dot Cellular Automata (QCA) Circuit Partitioning: Problem Modeling and Solutions, *Proc. 41st ACM/IEEE Design Automation Conf. (DAC)*, 2004, pp.363–368.
- [2] I. Amlani, A. Orlov, G.L. Snider, and C.S. Lent, Demonstration of a Functional Quantum-Dot Cellular Automata Cell, *Journal of Vacuum Science and Technology B*, 16(1998), pp. 3795-3799.
- [3] M.T. Niemier and P.M. Kogge. Exploring and Exploiting Wire-level Pipelining in Emerging Technologies, *Proc. 28th Annual International Symp.on Computer Architecture*, 2001, pp.166-177.
- [4] P.D.Tougaw and C.S.Lent, Logical Devices Implemented Using Quantum Cellular Automata, *J. of App. Phys.*, 75(1994), p.1818.

- [5] A. Chaudhary, D.Z. Chen, X.S. Hu, M.T. Niemier, R. Ravinchandran, and K.M. Whitton, Eliminating Wire Crossings for Molecular Quantum-Dot Cellular Automata Implementation, *Proc.of IEEE/ACM International Conference on Computer-Aided Design*, 2005, pp.565-571.
- [6] A. Cao and C.-K. Koh, Non-Crossing Ordered BDD for Physical Synthesis of Regular Circuit Structure, *Proc. International Workshop on Logic and Synthesis*, 2003, pp.200-206.
- [7] M.R. Garey and D.S. Johnson. Computers and Intractability — A Guide to the Theory of NP-Completeness. W.H.Freeman and Company, New York, 1979.
- [8] V.V.Vazirani. Approximation algorithms. Springer-Verlag, 2001.
- [9] W.T.Tutte. Toward a Theory of Crossing Numbers. *Journal of Combinatorial Theory*, 1970.
- [10] F.Shahrokhi, O.Sýkora, L.A.Szély and I.Vrto. Crossing number: bounds and applications. em In I.Bárány and K.Böröczky, editor, Intuitive Geometry, Bolyai Society Mathematical Studies 6,pages 179-206 . Akadémiai Kiadó, Budapest, 1997.
- [11] J.Li, A.Chaudhary, D.Z.Chen, R.Fleisher, X.S.Hu, M.T.Niemier, Z.Xie, H.Zhu. Approximating the Maximum Sharing Problem. *manuscript*, 2006
- [12] T. Lengauer, Combinatorial Algorithms for Integrated Circuit Layout, Wiley, New Yory, 1990.
- [13] M. Marek-Sadowska and M. Sarrafzadeh. The Crossing Distribution Problem, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(4) (1995), pp. 423-433.
- [14] C.D. Thompson. Area-time Complexity for VLSI, *Proc. 11th Annual ACM Symp. on Theory of Computing*, 1979, pp. 81-88.