MATEMATIKA, 2007, Volume 23, Number 2, 99–120

Placement and Routing in VLSI design Problem Using Single Row Routing Technique

Farhana Johar & [1]Shaharuddin Salleh
Department of Mathematics, Faculty of Science, Universiti Teknologi Malaysia Skudai
81310 Johor, Malaysia
e-mail: [1]ss@mel.fs.utm.my

Abstract Two major problems are involved in VLSI design, namely, the placement of components and routing between these components. Single row routing problem is a combinatorial optimization problem of significant importance for the design of complex VLSI multi layer printed circuit boards (PCB's). The design involves conductor routing that makes all the necessary wiring and interconnections between the PCB modules, such as pins, vias, and backplanes. In very large systems, the number of interconnections may exceed tens of thousands. Therefore, we have to optimize the wire routing and interconnections and thus determine the efficient designs. Kernighan-Lin algorithm, traveling salesman problem, simulated annealing algorithm and single row routing problem are used to find the best design. Included here are some simple examples to find the results. A simulation program using Microsoft Visual C++ is developed to simulate the single row routing problem using the simulated annealing algorithm.

Keywords Placement and Routing in VLSI, Single Row Routing, Kernighan-Lin Algorithm, Simulated Annealing

1 Introduction

Issues in VLSI design

Due to the growth of the computer and communication industries, there is a significant demand for digital designers. VLSI system design deals with the design of integrated circuit chips, boards that include these chips and systems that use the boards. In addition, as the complexity of integrated circuit chips increases, analog hardware often accompanies the digital hardware on a chip. As a consequence, analog design and mixed signal design are also potential components of this area. There are many approaches and techniques in solving the VLSI design problems.

A comprehensive survey were presented by Shahookar and Mazumder [1] with emphasis on standard cell and macro placement. Five major algorithms for placement are discussed: simulated annealing, force-directed placement, min-cut placement, placement by numerical optimization and evolution-based placement. The first two classes of algorithms owe their origin to physical laws, the third and fourth are analytical techniques, and the fifth class of algorithms is derived from biological phenomena.

Chandy and Banerjee [2] investigates two new approaches that have been proposed for generalized parallel simulated annealing, multiple Markov chains and speculative computation. These two algorithms are compared with the parallel moves approach to perform parallel cell placement.

Issues in Single Row Routing (SRR)

A systematic suboptimal approach to multilayer rectilinear wire routing: single row approach, was investigated by Raghavan and Sahni [3]. The method involves decomposing the general rectilinear wiring problem into a number of conceptually simpler single row problems. The complexity of the decomposition process will be first considered. Under interesting optimization criteria, each step in the decomposition process shown to be NP-hard. Then the single row wiring problem itself is considered under a variety of optimization criteria. In some cases, either efficient or "usually good" algorithms are possible. In other cases, the problem turns out to be NP-hard.

Han and Sahni [4] developed two fast algorithms for the layering problem that arises when the single row routing approach to wire layout used. Both of these algorithms are for the case when the upper and lower street capacities are two. While neither of these algorithms guarantees the production of an optimal layering, it has been empirically determined that both will produce better layerings than an earlier proposed algorithm for this problem. In addition, these algorithms run much faster than an earlier algorithm.

Battacharya et al [5], Hossain et al [6], and Deogun and Sherwani [7] consider some restricted single row routing problems. Graph theoretic approach is used to obtain restricted classes of single row routing problems. Optimal street congestion algorithms are proposed for single row routing problems that have overlap graphs isomorphic to path, binary tree and clique.

Salleh and Zomaya [8] introduce the different approach, which is based on the simulated annealing algorithm (SAA). By performing slow cooling, the nets in the SRR problem align themselves according to a configuration with the lowest energy. The energy is taken as the absolute sum of the heights of the net segments, given as follows:

$$E_L = \sum_{i=1}^{m} \sum_{r=1}^{m} |h_{i,r}|$$

where m_i is the number of segments in the net N_i for $i = 1, 2, 3, \ldots, m$.

Based on these literature reviews, clearly there are many methods to solve the placement and the routing problem in VLSI design. The most popular one is using graph theoretical method. But we interested to use another approach to solve this problem, which is simulated annealing. Routing problem solved by the single row technique, which is considered the interconnection and the placement of the components.

2 Placement Strategy by Partitioning

Placement problem is a problem of determining the relative position of each transistor within the target cell architecture. Figure 2.1 shows layout set of component. A common strategy is to minimize the number of connections that may be made between the devices

through common diffusions and vertically aligned gate wires. Figure 2.2 shows two different placements of the component in the same problem.

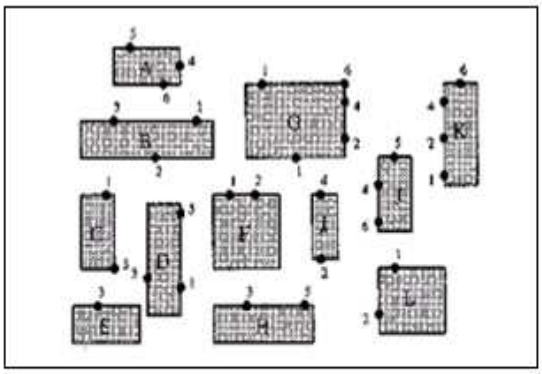

Figure 2.1: Blocks and set of interconnecting nets

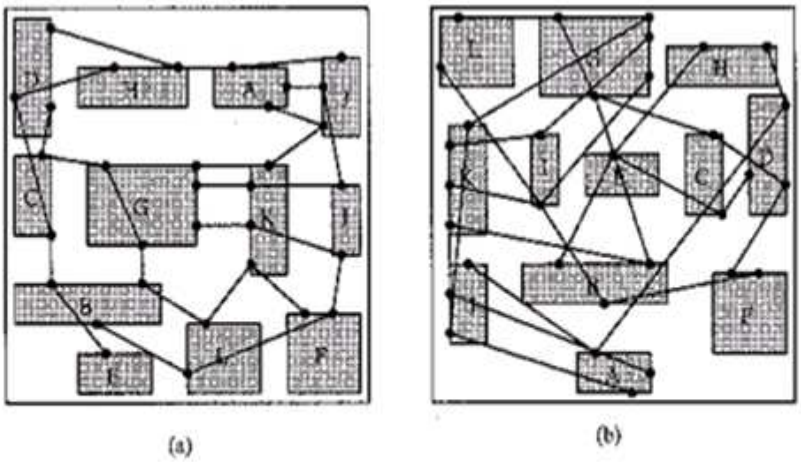

Figure 2.2: Two different placements of the same problem

Shahookar and Mazumder [1] defines the placement problem as follows. Given an electrical circuit consisting of modules with predefined input and output terminals and interconnected in a predefined way, construct a layout indicating the positions of the modules so the estimated wire length and layout area are minimized. The inputs to the problem are the module description, consisting of the shapes, sizes and the terminals locations and the netlist, describing the interconnections between the terminals of the modules. The output is a list of x- and y-coordinates for all modules.

Partitioning circuits with costs and edges into partitions of specified sizes while trying

to minimize total cost of edges cut has many applications in design automation. Many techniques have been developed to solve this problem. Among the techniques are direct partitioning techniques, group migration and metric allocation techniques. Combinatorial optimization techniques such as simulated annealing have also been applied to partitioning. Many heuristics have been developed based on the work done by Kernighan-Lin and Fiduccia-Mattheyses.

2.1 Kernighan-Lin Algorithm

The Kernighan-Lin algorithm partitions a given graph $G = (V, E)$ of $2n$ into two equal subgraphs of n nodes minimizing the edges cut. A cost matrix

$$C = (c_{i,j}), i, j = 1, 2, \ldots, 2n, i \neq j$$

is associated with the graph. The algorithm starts with an arbitrary partition of V into two subsets A and B. For each node $a \in A$, an *external cost* E_a is defined by

$$E_a = \sum_{y \in B} c_{ay}$$

and an *internal cost* I_a by

$$I_a = \sum_{x \in A} c_{ax}$$

and

$$D_a = E_a - I_a$$

is a difference between external and internal costs. The gain g_i, produced when a and b $(a \in A, b \in B)$ are interchanged, is given by $D_a + D_b - 2c_{ab}$. In each iteration the algorithm interchanges k $(k \leq n)$ pairs of vertices between the two subsets. The Kernighan-Lin algorithm begins with an arbitrary partition of V into two equal-sized for all vertices. A pair of vertices, one from each subset that generates the maximum gain through interchange, is chosen. The gain is stored and the selected pair locked to prevent it from being considered for interchange again. This procedure continues until all n vertex pairs are generated and a sequence of gains, $g_1, g_2, \ldots, g_n$ is generated. The total gain Gain(h), of interchanging the first k pairs of vertices, $1 \leq k \leq n$ is computed:

$$\text{Gain}(k) = \sum_{i=1}^{k} g_i = G.$$

The algorithm interchanges the first k pairs of vertices for which G is maximal. Intermediate individual gains g_i and cumulative gain G can be negative. If the best gain found in an iteration is less than or equal to zero, the iteration is stopped (Kernighan and Lin [9]).

2.2 Travelling Salesman Problem

In the case of the traveling salesman problem, the mathematical structure is a graph where each city is denoted by a point (or node) and lines are drawn connecting every two nodes

(called arcs or edges). Associated with every line is a distance (or cost). When the salesman can get from every city to every other city directly, then the graph is said to be *complete.* A round-trip of the cities corresponds to some subset of the lines, and is called a tour, or a Hamiltonian cycle, in graph theory. The length of a tour is the sum of the lengths of the lines in the round-trip.

The problem then becomes

$$\text{minimize} \sum_{j=1}^{m} \sum_{i=1}^{m} c_{ij} x_{ij}$$

such that

$$\begin{aligned}
&\sum_{j=1}^{m} x_{ij} = 1 && \text{for } i = 1, \ldots, m, \\
&\sum_{i=1}^{m} x_{ij} = 1 && \text{for } j = 1, \ldots, m, \\
&\sum_{i=1}^{|K|} \sum_{j=1}^{|K|} k_{ij} \leq |K| - 1 && \text{for } K = \{k_i\}\ i = 1, \ldots |K|, \text{ and } K \subset \{1, 2, \ldots, m\}
\end{aligned}$$

where $|K|$ denotes the number of elements (cities) in K. The cost c_{ij} is allowed to be different from the cost c_{ji}. Note that there are $m\,(m-1)$ zero-one variables in this formulation.

To formulate the *symmetric* traveling salesman problem, one notes that the direction traversed is immaterial, so that $c_{ij} = c_{ji}$. Since direction does not now matter, one can consider the graph where there is only one arc (undirected) between every two nodes. Thus, we let $x_j \in \{0, 1\}$ be the decision variable where j runs through all edges E of the undirected graph and c_j is the cost of traveling that edge. To find a tour in this graph, one must select a subset of edges such that every node is contained in exactly two of the edges selected. Thus, the problem can be formulated as a 2-matching problem in the graph G^{v} having $m(m-1)/2$ zero-one variables, i.e. half of the number of the previous formulation. As in the asymmetric case, subtours must be eliminated through subtour elimination constraints.

The problem can therefore be formulated as:

$$\text{minimize} \quad \frac{1}{2} \sum_{j=1}^{m} \sum\nolimits_{k \in J(j)} c_k x_k$$

such that

$$\begin{aligned}
\textstyle\sum_{k \in J(j)} x_k &= 2 && \text{for all } j = 1, \ldots, m, \\
\textstyle\sum_{j \in E(K)} x_j &\leq |K| - 1 && \text{for all } K \subset \{1, \ldots, m\}, \\
x_j &= 0 \text{ or } 1 && \text{for all } j \in E(k).
\end{aligned}$$

where $J(j)$ is the set of all undirected edges connected to node j and $E(K)$ is the subset of all undirected edges connecting the cities in any proper, nonempty subset K of all cities. Of course, the symmetric problem is a special case of the asymmetric one, but practical experience has shown that algorithms for the asymmetric problem perform, in general, badly on symmetric problems. Thus, the latter need a special formulation and solution treatment.

A problem in graph theory requires the most efficient (i.e., least total distance) Hamiltonian circuit a salesman can take through each of n cities (Hoofman and Padberg [10]). No general method of solution is known, and the problem is NP-hard.

2.3 Simulated Annealing Algorithm

Simulated annealing was introduced by Metropolis et al. This method is used to approximate the solution of very large combinatorial optimization problems such as the TSP or VLSI-CAD problems including PLA folding, partitioning, routing, logic minimization, floorplanning or placement (Shahookar and Mazumder[1]. The technique originates from the theory of statistical mechanics and is based upon the analogy between the annealing of solids. It is very time consuming but yields excellent results. It is an excellent heuristic for solving any combinatorial optimization problem.

Assume that we are looking for the configuration that minimizes a cost function E. Starting off at an initial configuration, a sequence of iterations is generated. Each iterations consists of the random selection of a configuration from the neighbourhood of the current configuration and the calculation of the corresponding change in cost function ΔE.

Typically, we use the Boltzmann probability distribution

$$\text{Prob}(E) \sim \exp\left(\frac{-\Delta E}{kT}\right)$$

where k is a constant and at a given temperature T, a system can be in a range of possible energy states-the higher the temperature the more likely it is to be in a high energy state. In practise, the temperature is decreased in stages and at each stage the temperature is kept constant until thermal quasi-equilibrium is reached.

The simulated annealing program consists of a pair of nested loops. The outer-most loop sets the temperature and the inner-most loops runs the simulation at that temperature. The way in which the temperature is decreased is known as the *cooling schedule.* In practice, two different cooling schedules are predominantly used: a linear cooling schedule ($T_{\text{new}} = T_{\text{old}} - dT$) and a proportional cooling schedule ($T_{\text{new}} = C^* T_{\text{old}}$) where $C < 1.0$. These are not only possible cooling schedules, they are just the ones that appear in most literature (Kirkpatrick et al [11]).

3 Routing Problem

The routing problem is typically the assignment of physical wires on the board to individual nets in a design. Ideally, placement and routing should be performed simultaneously as they depend on each others results. This is however, too complicated. In real problem, placement is done prior to routing. The objectives of the routing are to minimize the interconnection lengths and provide the shortest connections between terminals.

The routing problem can be defined as follows. Given a placement and a fixed number of metal layers, we have to find a valid pattern of horizontal and vertical wires that connect the terminals of the nets. Our objectives here are to minimize the respective cost components such as area, wire delays, number of layer and the other additional cost component like the number of bends and vias. There are two levels of abstraction: global routing and detailed routing.

Solving the routing problem, minimizes the total of area and the wire length. It is important to optimize chip area usage in order to fit more functionality into a given chip area. With the optimal wire length being used, the capacity delays associated with longer nets will be reduced and this will speed up the operation of the chip. This, in turn,

reduces the communication cost between the terminals and therefore, improves the circuit performance.

3.1 Single Row Routing Technique

The classical single row routing problem is one of the important problems in the layout design of multilayer circuit boards (Battacharya et al [5]). Single row routing problem, as applied to VLSI design, is a specific subproblem of the floor plan problem, which not only considers the interconnection of components, but also the placement of the component such as that the total chip area used is minimized. The goal of area minimization is paramount in VLSI design because area dominates the cost of a chip. The problem of finding an optimal placement for the general floor plan problem is NP-Complete e (Looges [12]).

In the single row routing problem, we are given n points (pins / terminals) $V = 1, 2, ..., n$ evenly spaced along a line, and a set of nets $L = \{N_1, N_2, \ldots, N_m\}$ over V. Without loss of generality, we may assume that the line of points is oriented horizontally.

Laying out a single row wiring problem entails defining conductor paths in order to realize all the nets, subject to each of the following:

- Each conductor path is made up of horizontal and vertical segment only.
- Conductor paths do not cross.
- The layout of each net should be free of "backward moves". In other words, it should not be possible to draw a vertical line anywhere that intersects more than one conductor segment of any given net. A layout that satisfies the above constraints is called a realization.

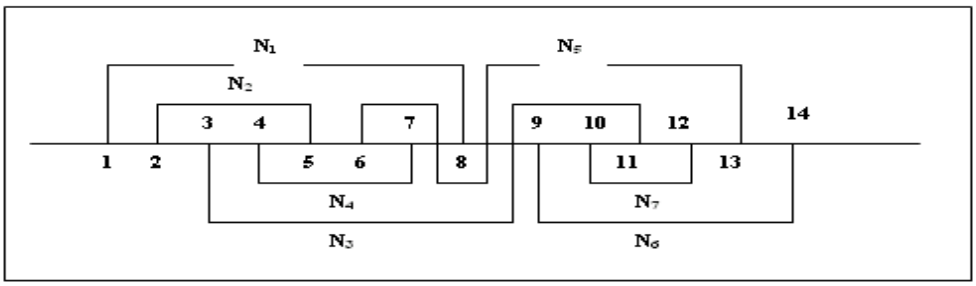

Figure 3.1: Single row realization of the net list L_1

The wiring channel above the line of points is called the *upper street* and the one below, the *lower street.* The number of wiring tracks needed in the upper street is *upper street congestion,* C_{us} and the below one is *lower street congestion,* C_{ls}. The quantity $\max\{C_{us}, C_{ls}\}$ is called width.

When the layout of the conductor path contains a vertical segment from the upper street to the lower street, the path is said to make *interstreet crossing.* The *between -nodes congestion* of a realization is

$$\max\{\text{number of interstreet crossings between an adjacent pair of points}\}.$$

In laying out a wire the number of right-angled *bends* in the layout of that wire is related to the number of interstreet crossings made by it, as follows:

$$\text{the number of bends} = 2(\text{number of interstreet crossings}) + 2.$$

In general, $p-1$ wires are required to realize a net with p points. Let p_i be the number of points in net $N_i, 1 \leq i \leq m$. Then, for the complete realization,

$$\text{The number of bends} = 2(\text{number of interstreet crossings}) \; + \; 2 \sum_i (p_i \; - \; 1\,)$$

There are many single rows wiring problem under a variety of constraints and optimization measures such as minimizing the width, between-nodes congestions and the total number of bends.

Here, we are concerned with finding a realization that minimizes the total number of bends (in all wires). This optimization measure is important in at least this context (Raghavan and Sahni [3]):

- In fabricating microwave and millimetric wave integrated circuits, the conductor paths in the metallization layer act as waveguides, rather than as simple electrical wires. These paths are called microstrip line. The effect of a bend in a microstrip line is the creation of reflections and this reduces the transmission efficiency of the line. This forces the drivers to send out stronger signals in order to effect a proper transmission. So, in this context, counting the total number of bends in a realization can help form a rough estimate of the power requirements of the chip. Minimizing the total number of bends will help reduce the power consumption of the integrated system.

- As mention earlier, the total number of bends in a realization is intimately related to the total number of interstreet crossings. If external circumstances (e.g fixed placement procedure) do not force the point locations to be fixed, a realization that minimizes the total number of interstreet crossings can be followed by a suitable relocation of the point positions along the line of points. The result is a "minimum length" realization. i.e., a realization in which the separation between the two extreme points is minimized.

4 Examples

4.1 Model Desciption 1

Figure 4.1 below shows a graph representation of a set of gates that are partitioned into two sets.

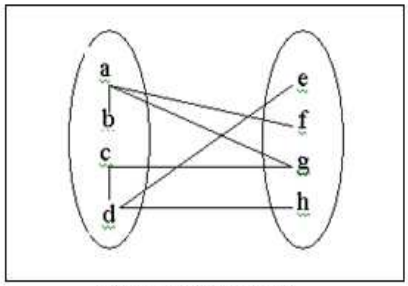

Figure 4.1: Model 1

Analysis of Results

Kernighan-Lin algorithm is one of an iterative improvement techniques that produces good placement but requires enormous amounts of computation time. For the above example,

For $i = 1$ to 4 :

- iteration 1 : compute the gain for all free pairs takes 16 time
  - ◯ choose the largest gain takes 4 time
  - ◯ update all the D's takes 4 time
- iteration i : compute gain for all free pairs takes $(n - i + 1)^2$

Generally, when we are given a graph $G(V, E)$, where $|V| = 2n, \{A, B\}$ is the initial partition such that $|A| = |B|$:
inner (for) loop:

- iterates n times
- iteration 1 : (n^*n) time
- Iteration i : $(n - i + 1)^2$

4.2 Model Description 2

Given a set of gates that need to be partitioned into two sets, as shown in Figure 4.4:

By applying Kernighan- Lin algorithm, the final solution for our model is $A = \{1, 2, 3\}$ and $B = \{4, 5, 6\}$. Now we apply the single row routing approaches to this problem to find the best routing.

Analysis of Results

For this example, we have to iterate twice before achieving $G < 0$. Then, we show it as single row routing.

Figure 4.5 is the layout of the single row routing before the circuit is partitioned. All the nodes are in one row. The dots in each area nodes show a number of connection wires.

Figure 4.6 is the layout of single row routing after the circuit is partitioned. Red lines are referred to as graph partition line. Our objective is to minimize a number of connection

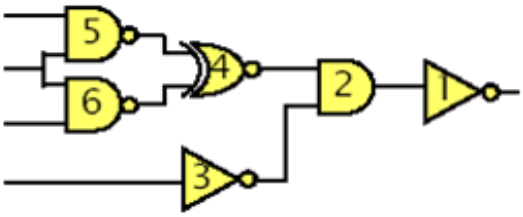

Figure 4.2: Our Original Model

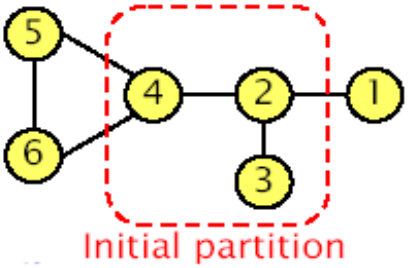

Figure 4.3: Initial partition of a set of gates

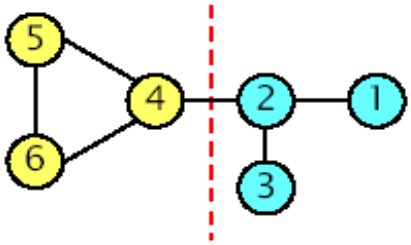

Figure 4.4: Graph representation after first iteration, component 1 and 4 swap.

First view:

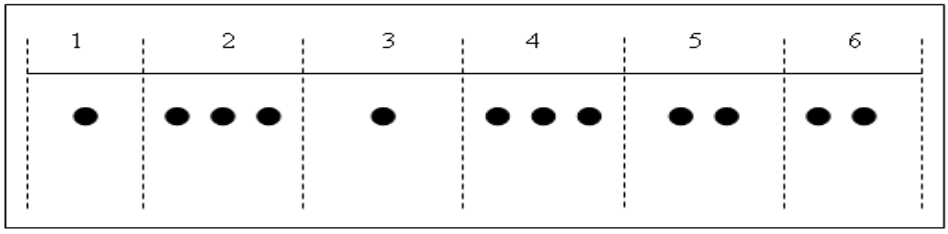

Figure 4.5: Single row routing before partition

At first iteration:

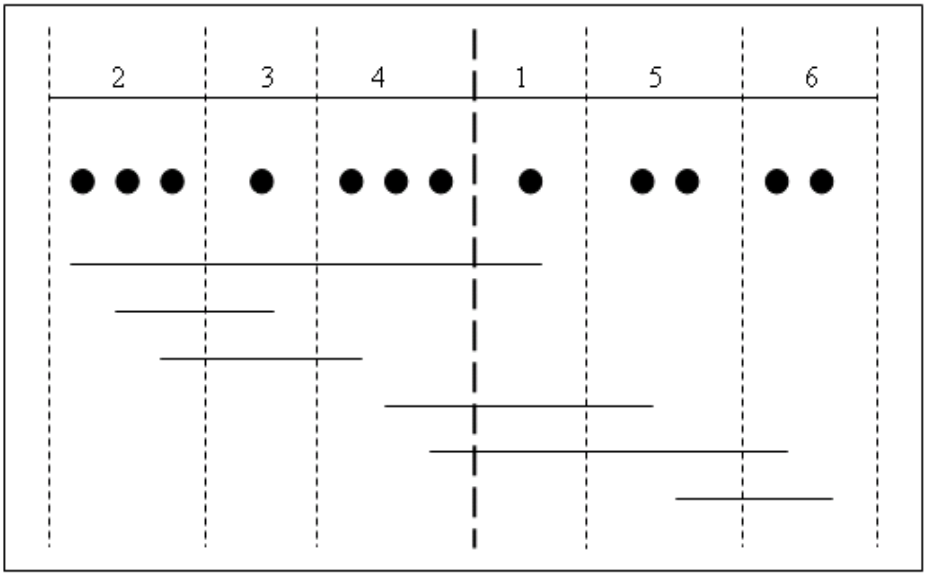

Figure 4.6: Single row routing after partition into two sets. Bold line shows a partition graph line. The number of cutsize is 3

For the final solution:

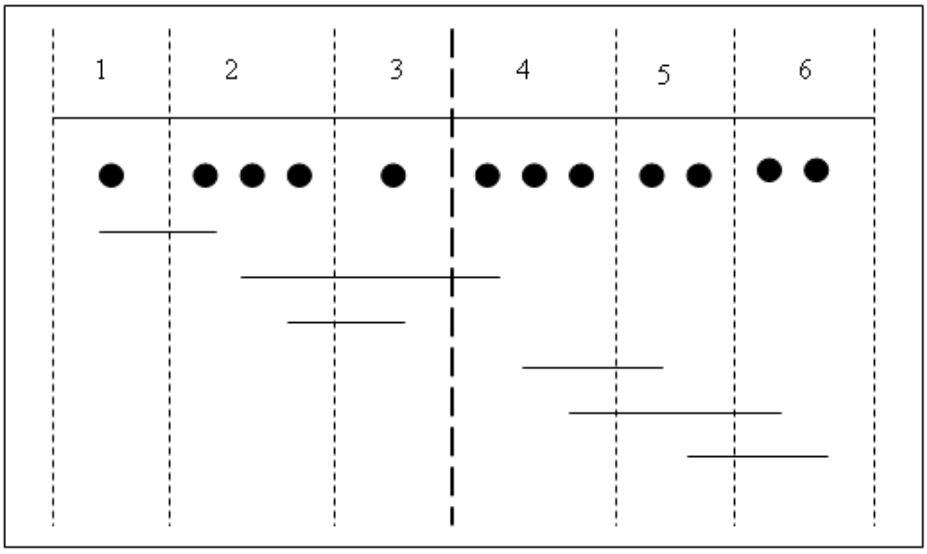

Figure 4.7 Single row routing for the final solution. The number of cutsize is reduced to 1. This is the best placement.

wires between nodes that interstreet that red line. After first iteration, the number of cutsize is 3.

Figure 4.7 shows a final solution for this example. After swapping the node 1 and 4 by Kernighan-Lin algorithm, it shows an improvement in cutsize number.

5 Single Row Routing Simulation Model

A computer simulation model has been developed to study the simulated annealing and single row routing technique. In this simulation, only 80 nets to be considered. Our simulation was based on single row routing technique using the simulated annealing algorithm.

We used the MFC with Microsoft Visual C++ to simulate our model. MFC was created to make programming in Windows easier. As an object-oriented wrapper for Win32, it automates many routine programming tasks (mostly passing references around). Paradigms like the document/view architecture were added to automate even more tasks for the programmer, but in the process, some control was taken away.

5.1 Model Description

Our simulation consists of 160 terminals/pins, $V = \{1, 2, 3, \ldots, 160\}$, evenly spaced along a line, and a set of nets $L = \{N_1, N_2, \ldots, N_{80}\}$ over V. We put all the data in our input file. We assume that the line of terminals oriented horizontally. Our simulation has been developed using Microsoft Visual C++ based on the single row technique.

Firstly, we have defined the initial temperature, $T_o = 100^oC$, constant, $k = 1$ and Boltzmann probability of accept, $\varepsilon = 0.9$. We have used the proportional cooling schedule $T_{new} = a^*T_{old}$ where $\alpha = 0.95$. This simulation will start as in Figure 5.1. The first and forth column are the nets number, the second and fifth column are terminals start while the rest are terminals end.

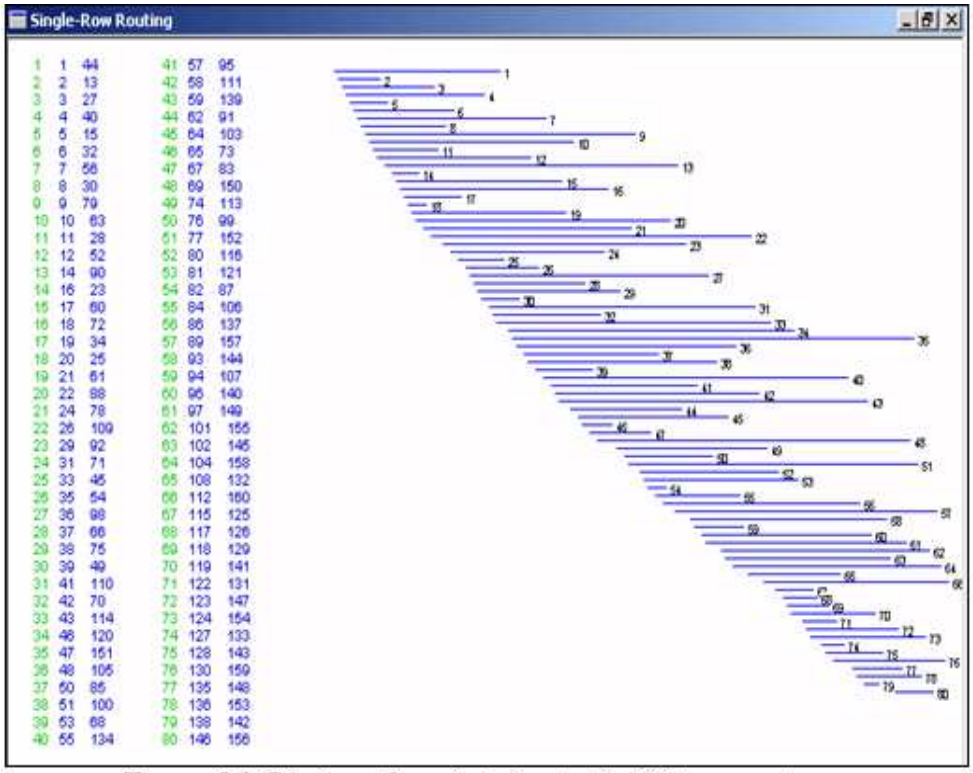

Figure 5.1: Display of our data input at initial temperature

Some parameters involved in this simulation are as follows:

T	:	temperature that will change after we press the enter or space bar key
ColPosition(int)	:	display the nets into two column in our output
Energy	:	energy of the nets
Q	:	maxima qm,qM or width
qm	:	lower street congestions
qM	:	upper street congestions
nDogLegs	:	number of doglegs or a vertical line joining two horizontal lines of the same track in a given realization

5.2 Data Structure

A data structure is constructs that can be defined within a programming language to store a collection data. The following data structure describes our simulation:

```
struct nn
        {
                int b,e,q,bq,eq;
                int order,bqU,bqL,eqU,eqL;
                bool Hide;
                int nDog;
                int vbd[2*MaxNets+1];
                int sh[MaxNets+1];
                int energy;
        {       N[MaxNets+1];

struct mm
        {
                bool Hide;
                int Net;
        {       v[2*MaxNets+1];
```

Some of the member functions in the simulation are:

ClearScreen()	:	to clear the screen after iteration
Update(void)	:	update the output display after press the enter and a spacebar key
Perturb()	:	doing all the calculation
DisplayNets()	:	to display all of nets
OnPaint()	:	to draw a horizontal line/interval line for each nets
OnKeyDown()	:	to let the node i moving either randomly or in one way direction.

5.3 Simulation Model

In this simulation, a perturbation is performed to examine the neighbors by moving a net at random to a new position. The resulting change in energy (E is then evaluated. If the energy is reduced, that is $\Delta E < 0$, the new configuration is accepted as the starting point for the next move. However, if the energy is increased, $\Delta E > 0$, the move generates the probability Prob(E) $\sim \exp\left(\frac{-\Delta E}{kT}\right)$. The move is accepted if this probability is greater than a Boltzmann probability of acceptance, $\varepsilon = 0.94$ and rejected otherwise. Within a higher value, the number of moves accepted for $\Delta E > 0$ are reduced and the same rule applies vice versa.

Figure 5.2 , Figure 5.3 and Figure 5.4 show an output of display net for T=95^{o}C, 85.737^{o}C and 81.451^{o}C. From the right side lines drawn, we can see that every terminal/pins are oriented horizontally and each net was free of backward moves. From these figures also, seen that the move is accepted because there exist an amount reduction in total of energy, width and number of doglegs. This simulation will continue until the best result obtain.

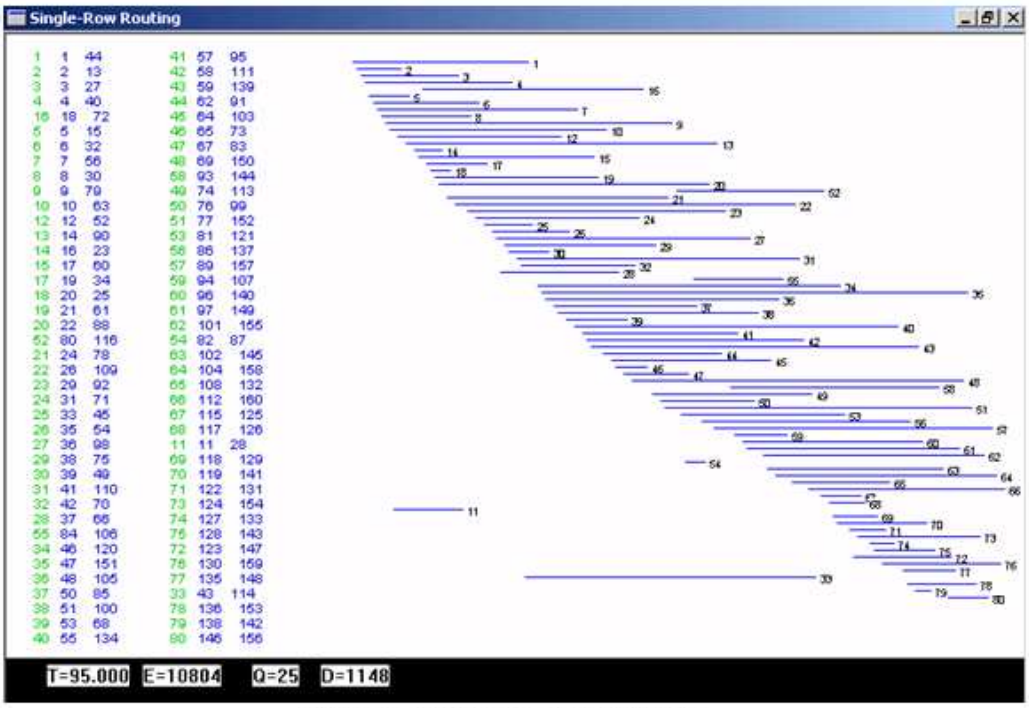

Figure 5.2: Data output at T=95°C

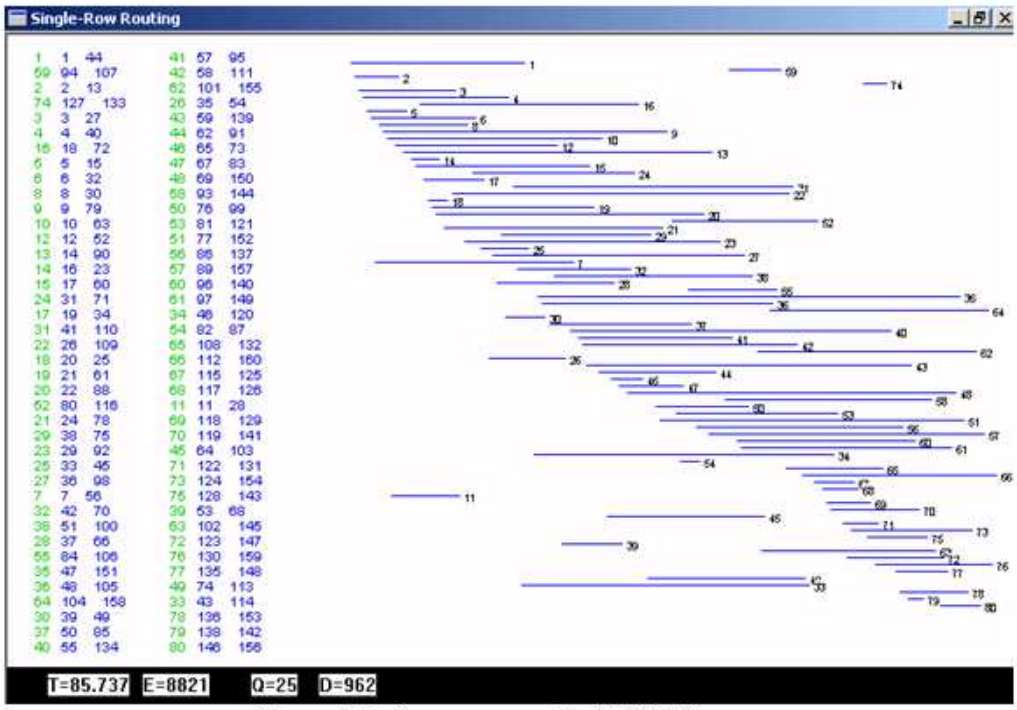

Figure 5.3: Data output at T=85.737°C

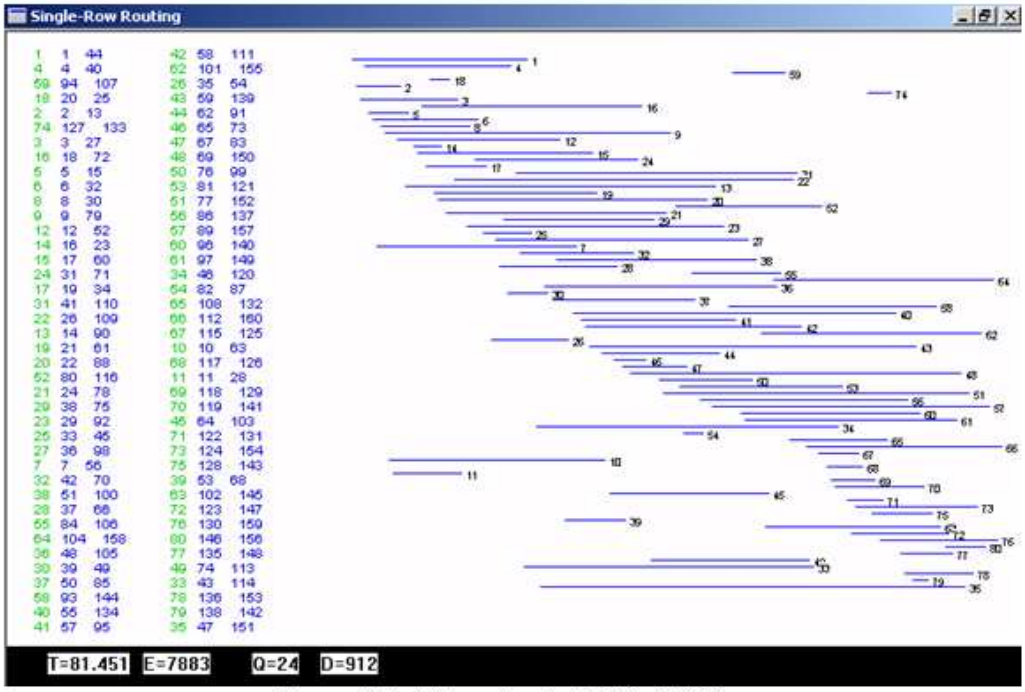

Figure 5.4: Data output at T=81.451°C

5.4 Results

Based on the results in appendix, shown that this process lowers the temperature by slow stages and at T=0.656°C, we can see that the system 'freezes' and no further changes occur. In this simulation, the total of energy is 3244, the width is 24 and the number of doglegs is 440. Figure 5.5 below shows the final result for this simulation.

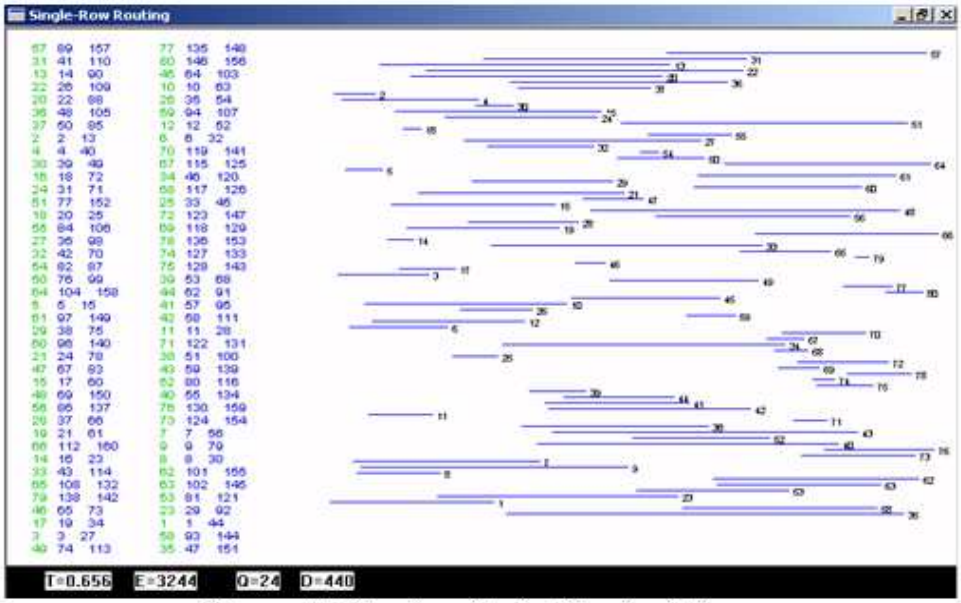

Figure 5.5: Final results for this simulation

The main drawback of simulated annealing is the long computation time needed before converging to the stable state. This is due to the fact that the algorithm performs a large number of random searches at each temperature step before arriving at the equilibrium state.

6 Conclusion

Single row routing is a component of many complex VLSI design problems. The difficulty with the problem stems from the fact that no solid mathematical approach has been able to provide a general solution. The simulated annealing algorithm was applied by using an objective function, which is expressed in terms of the track height. The energy in the objective function represents both the street congestion and the number of interstreet crossings. By reducing the two cost into one, the new energy now represents the degree for the compactness of the routing design.

References

[1] K Shahookar & P. Mazumder, *VLSI Cell Placement Techniques, ACM Computing Surveys,* Vol. 23(1991), No.2, pp 143-173.

[2] J.A. Chandy & P. Banerjee, *Parallel Simulated Annealing Strategies for VLSI Design,* 9th International Conference Proceedings on VLSI Design: VLSI in Mobile Communication, pg. 37, 1996

[3] R. Raghavan & S. Sahni, *The Complexity of Single Row Routing,* IEEE Transactions on Circuits and Systems, Vol. CAS-31(1984), No.5, pp 462-472.

[4] S. Han & S. Sahni, *Layering Algorithms for Single Row Routing,* IEEE Transactions on Computer Aided Design, Vol.CAD-6(1987), No.1, pp 95-102.

[5] B. Battacharya et al, *A Graph Theoretic Approach to Single Row Routing Problems,* IEEE Transaction on Computer-Aided Design of Integrated Circuit and Systems, Vol. 2(1988), pp 1437-1440

[6] M. Hossain et al , *Optimal Algorithms for Restricted Single Row Routing Problems,* Proc. 34th Midwest Symposium on Circuit System, pp 823-826, 1992.

[7] J.S. Deogun & N.A. Sherwani, *A Decomposition Scheme for Single Row Routing Problems,* Report series #68, Department of Computing Science, University of Nebrasca, 1988.

[8] S. Salleh & A.Y. Zomaya, *Scheduling in Parallel Computing Systems,* Kluwer Academic Publishers, pp 127-146, 1999.

[9] B.W. Kernighan & S.Lin , *An Efficient Heurictic Procedure for Partitioning Graphs,* The Bell System Technical Journal, Vol.49(1970), pp 291-307.

[10] K. Hoffman & M. Padberg, *The Travelling Salesman Problem,* Encyclopedia of Operational Research & Management Science, Kluwer Academic Publication, Boston, 1995.

[11] S. Kirkpatrick et al , *Optimization by Simulated Annealing Science,* Vol. 220(1983), pp 671-680.

[12] P. Looges,P , *Indifference Graph and Single Row Routing,* M.SC thesis, Computer Science Department, Old Dominion University, 1991.

Appendix

Results for 80 nets

Output file

Temp.	Energy	Width	Doglegs
95.000	10804	25	1148
90.250	9607	24	1046
85.737	8821	25	962
81.451	7883	24	912
77.378	7386	24	851
73.509	7239	26	854
69.834	7204	26	846
66.342	7204	26	846
63.025	7164	26	827
59.874	7164	26	827
56.880	6790	26	832
54.036	6768	26	830
51.334	6768	26	830
48.767	6362	26	801
46.329	6337	26	797
44.013	6333	26	796
41.812	6308	26	791
39.721	6298	26	791
37.735	6298	26	791
35.849	6180	26	769
34.056	6113	26	751
32.353	6066	26	749
30.736	5508	26	703
29.199	5508	26	703
27.739	5508	26	703
26.352	5496	26	695
25.034	5476	26	695
23.783	5476	26	695
22.594	5445	26	674
21.464	5445	26	674
20.391	5325	26	662
19.371	5325	26	662
18.403	5325	26	662
17.482	5247	26	653
16.608	5239	26	651
15.778	5082	25	671
14.989	4878	25	643
14.240	4833	25	639
13.528	4756	25	633
12.851	4756	25	633
12.209	4735	25	627
11.598	4735	25	627

Temp.	Energy	Width	Doglegs
11.018	4658	25	619
10.467	4538	25	600
9.944	4529	25	598
9.447	4508	25	596
8.974	4492	25	592
8.526	4492	25	592
8.099	4487	25	589
7.694	4398	25	587
7.310	4285	25	578
6.944	4207	24	571
6.597	4202	24	570
6.267	4155	24	562
5.954	4143	24	566
5.656	4079	24	554
5.373	4079	24	554
5.105	3966	24	551
4.849	3965	24	549
4.607	3932	24	544
4.377	3899	24	546
4.158	3895	24	544
3.950	3893	24	542
3.752	3845	24	536
3.565	3845	24	536
3.387	3816	24	533
3.217	3767	24	526
3.056	3631	23	503
2.904	3631	23	503
2.758	3608	23	501
2.620	3608	23	501
2.489	3608	23	501
2.365	3608	23	501
2.247	3608	23	501
2.134	3608	23	501
2.028	3529	23	492
1.926	3518	23	497
1.830	3515	23	495
1.738	3494	23	491
1.652	3488	23	489
1.569	3485	23	487
1.491	3475	23	483
1.416	3459	23	483
1.345	3446	23	480

Temp.	Energy	Width	Doglegs
1.278	3446	23	480
1.214	3446	23	480
1.153	3446	23	480
1.041	3430	23	481
0.989	3408	23	466
0.939	3408	23	466
0.892	3366	23	465
0.848	3366	23	465
0.805	3319	23	467
0.765	3319	23	467
0.727	3308	23	463
0.691	3303	23	461
0.656	3244	24	440
0.623	3244	24	440
0.592	3244	24	440
0.562	3244	24	440
0.534	3244	24	440
0.508	3244	24	440