Force Directed Graph Drawing Algorithms for Macro Cell Placement

Meththa Samaranayake , Helen Ji and John Ainscough

Abstract— **Macro cells are used more and more in current designs as they provide the benefit of reusability directly resulting in the decrease of design cost and time. However, there lies a gap in the EDA industry for Macro cell placement tools. This paper would like to introduce the idea of using graph-drawing algorithms as the basis for a Macro cell placement tool in order to obtain successful layouts.**

Index Terms— **design automation, Macro cell, placement tool, force directed algorithm, graph drawing**

I. Introduction

The past few years have seen an exponential rise in the growth rate of the semi-conductor industry. The increase in usage and demand of electronic devices among consumers has resulted in the need to provide better and faster design methods. The designers are pushed to their limits in meeting these demands whilst juggling the constraints of power and performance of ever shrinking circuits. To help designers meet their targets, EDA (Electronic Design Automation) tools are used to help fully or partially automate the design processes. One of such important backend processes is the placement component.

The placement problem simply is the problem of finding the ideal locations for each cell in a circuit achieving as many or all of the placement objectives. The two main objectives that every placement tool has to achieve are,

- overlap free layout
- fit in the given placement area.

Other objectives may include minimization of wirelength, area, congestion, run time etc. The optimal solution will be one that satisfies all of the given criteria. Achieving such a placement solution is far from possible and even the simplest of cell placement problems are defined to be NP-hard. The consequence of falling short of a good placement could result in an unroutable design, a slower and/or larger chip etc. This will cost time and money to either manually correct the placement or start the design from the beginning.

The input to the placement component consists of the description of all the cells including their size and pin locations and a netlist. On successful placement, the output will hold the locations of the cells that are non-overlapping and fitted into the placement area.

In the past, designs mainly carried standard cells that were of uniform height and width. Macro cells were introduced as an answer to the growing complexity of circuits. Macros can mainly be seen as black boxes that are designed to carry out specific tasks such as implementation of a logic function (e.g. an IP block). Increased use of Macro cells help designer's reuse of their designs which in turn helps minimise design time and cost.

This paper introduces the idea of using graph-drawing algorithms as the basis of a Macro cell placement tool. Two force directed algorithms, one authored by Kamada and Kawai [1] and the other by Fruchterman and Reingold [2] are the main focus of this research (these will be referred to as KK and FR respectively within the rest of this paper). They were chosen mainly for their ability to handle undirected straight line drawing graphs, their simplicity in implementation, their speed as well as the criteria they follow to produce aesthetically pleasing graphs. In many cases, these criteria are shared by good placements. More details on these two algorithms will be discussed in section IV.

Graph theory is widely used in solving problems in subjects such as electronics, computer science, physics, chemistry and even geography and more. The different types of graph implementations allow easy application to different situations making them a common choice of solution[3]. A Graph is a collection of nodes and edges where pairs of nodes will be connected with the edges. Graph drawing algorithms will take these nodes and edges and represent them in an aesthetically pleasing graphical manner. Graphs can have different characteristics; connected or disconnected, directed or undirected etc [4]. Undirected straight line drawing graph algorithms are required when considering placement tools.

The rest of this paper is organized as below. Section II will look at the different mixed size and macro cell placement tools. Section III discusses the issues that need to be given consideration when developing a macro cell placement tool. The graph drawing algorithms are discussed in Section IV whilst Section V highlights the experiments conducted on the algorithms and their performance results. Section VI gives brief details of the proposed placement tool before concluding in Section VII.

Manuscript received March 13, 2008. This research is jointly funded by the Department of Engineering and Technology of Manchester Metropolitan University, UK, and the ORSAS (Overseas Research Students Awards Scheme).

Meththa Samaranayake is with the Department of Engineering and Technology, Manchester Metropolitan University, John Dalton Building, Chester Street, Manchester M1 5GD, UK (phone: +44 (0) 1612473683; e-mail: Meththa.t.samaranayake@student.mmu.ac.uk).

Helen Ji is with the Department of Engineering and Technology, Manchester Metropolitan University, John Dalton Building, Chester Street, Manchester M1 5GD, UK (e-mail: h.ji@mmu.ac.uk).

John Ainscough is with the Department of Engineering and Technology, Manchester Metropolitan University, John Dalton Building, Chester Street, Manchester M1 5GD, UK (e-mail: j.ainscough@ mmu.ac.uk).

II. Placement tools

There are many standard cell placement tools available both academically and commercially. Several of them are capable of mixed-mode cell placement i.e. designs that contain both standard cells and macro cells, but there are only a few placement tools specifically for macro cells. This is in fact because standard cells govern most of the circuit designs. Recent changes have seen designs to contain macro-cell based designs such as memory blocks and IP blocks (Intellectual Property), and furthermore, the hierarchical design methodology intended to tackle design complexity results in macro-dominated designs at the top level. Even though mixed mode placement tools can handle macro cells, for designs that contain a majority of macro cells these tools may not place the cells in the best interest of the macro cells.

Some leading edge mixed mode placement tools identified are Capo[5], Dragon [6], FastPlace[7] and APlace[8]. The Capo tool is mainly based on simulated annealing and it handles macro cells by shredding them to smaller sub cells. These sub cells are connected by two pin nets ensuring that they are placed close to one another. The circuit is then considered as a standard cell placement problem. FastPlace and APlace tools are based on analytical techniques and incorporates macro cell placement in to its normal placement flow. In FastPlace the macro cells are given priority during legalization stage where overlaps are resolved between macros before standard cells. Dragon is a hybrid placement tool that combines the use of simulated annealing with min-cut partitioning. To handle macro cells, it has modified the min-cut partitioning algorithm so that the partitions can be of different sizes. All these placement tools were designed for standard cells and later modified to support macro cells. As a result they do not consider macro cell pin locations and cell orientation which are important factors for placing macros.

In [9] a Java based macro cell placer based on a force directed placement algorithm is described. In this work, unlike the traditional force directed algorithms, the cell shapes and sizes have been considered when developing the force equation. A limitation of this tool is that it does not handle placement on a fixed placement area and instead treats the chip as a soft cell with a variable size and aspect ratio. The pads of the chip are also not fixed; therefore the positions are found with the use of the force directed algorithm at a later stage.

A macro cell placement method based on net clustering and force directed method is proposed in [10]. This approach is unique such that, it treats the nets as the placement components. In the graphs they draw, the nodes represent the nets whilst an edge only exists for the nets that share one or more cells. The forces on the nets determine the initial locations for the cells. Pin locations are determined last, suggesting that this placement tool is mainly focused on soft cell macros. This work reiterates the importance of the pin locations and cell orientation in macro cell placement. Another limitation seen is that the tool only handles connected graphs, again limiting the type of designs that can be processed.

III. Placement Considerations

Macro cell placement is not as straightforward as standard cell placement. In standard cell placement, the cells are of uniform height and are restricted to rows in which they must sit in. These restrictions allow the placement tools to be more precise in choosing locations for the standard cells. Macro cells on the other hand do not have such restrictions. They can be of any height, width and shape and are not restricted to a specific location of the placement area with the exception of fixed cells.

When using graph-drawing methods for cell placement, cell size is an issue needs to be looked at. Work carried out regarding using different size and shape nodes for graph drawing has been considered in [11]. This work is mainly aimed towards general graph drawing algorithms and the criteria they use for graph drawing include 1) Vertices are not to overlap 2) Edges are not to cross vertices. For this research, the first criterion directly applies, as the objective of the placement tool is to produce a non-overlapping placement. The second criterion also applies as it tends to place directly connected cells together, but it could be too conservative if routing is allowed to be over-the-cell. . One of the limitations of the work in [11] is that the node orientation is fixed and cannot be mirrored or rotated. However, [11] has been able to successfully implement nodes of different sizes and shapes and place them in a visually pleasing manner.

As the cell size can be a significant amount of the total area, sometimes even up to half of the placement areas, the pin positions play a key role in generating a placement. Unlike in standard cell placement, pin locations can have a significant impact on wirelength, routability and congestion of the chip as seen in Figure 1. To overcome this, the placement tool will need to handle extra features such as cell mirroring and cell rotation to consider the best possible orientation in order to minimize costs.

Another difference between standard cell and macro cell placement is that macro cells do not have rows in which they should be placed in. The macro cells can be placed anywhere within the placement area.

Fixed cells are also an important factor that needs to be looked at during cell placement. There are designs that may want one or more cells to be placed in a fixed position within the placement area. In force directed algorithms, it should be noted that even the fixed cells exert a force even though the forces it encounters by others will not make any changes to its placement.

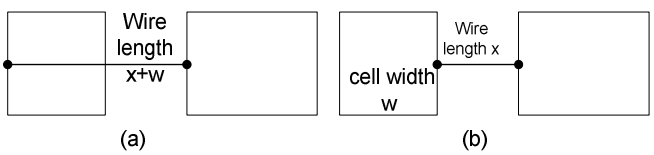

Figure 1 Effect of pin locations to wirelength (a) before cell mirroring wirelength includes the cell width (b) after cell-mirroring wirelength is reduced

IV. Force Directed Graph Algorithms

Force directed algorithms generally tend to be analogous to the classic problem of Hookes law for a spring system. Most of the current force directed algorithms follow the foot steps of Eades' spring embedded algorithm[12]. Hooke's law simply stated that the force exerted by an extended spring is proportional to the length of the spring. Eades modelled the graph as a system of rings in place of the nodes and springs for edges. His formula for the forces exerted by the springs differed Hooke's law by the former taking both attraction and repulsion forces in to consideration as seen in Figure 2. The aim of all the force directed algorithms is to find zero-force locations for all nodes – i.e. state of equilibrium for that system.

Traditional force directed algorithms tend to treat the cells as points that do not posses any size or shape. The edges do not connect to any pins but to the nodes that represents the cell. This method may be acceptable for standard cell design as identified in [9] but in Macro cell placement it can cause inaccuracies of positions, wirelength, area, congestion etc. due to the cell dimensions.

In [13] a comparison of several Force directed algorithms has been carried out where KK and FR algorithms were the two main contenders. It was identified that KK is successful in achieving high computation speed, minimising the computation time. Even though FR is quick in giving aesthetically pleasing layouts, it does suffer from long run times when the number of nodes/edges exceeds 60. As mentioned in [13] one cannot declare a certain algorithm to be the best. Each has its pros and cons and what is important is how relevant each algorithm is, to the work that needs to be done. The implementations for both algorithms used for this research work were taken from the Boost C++ library[14].

KK Algorithm[1] is concerned about general undirected, connected graphs. It has the ability to handle weighted graphs though it was seen that the weights had an opposite effect than what would be generally expected i.e. the higher the weight the closer the nodes should be. One advantage in this algorithm is that it introduces a "graph theoretic distance" which defines a minimum edge length ensuring the nodes do not overlap each other at any point. The main objective of the algorithm is to find a balanced formulation of the spring forces within the system. This algorithm though based on Eades work also makes use of Hooke's law in order to produce an optimized layout.

The graph drawing criteria followed by [1] are, 1) reduce number of edge crossings 2) distribute the vertices and edges uniformly. Comparing these criteria with those of the macro-cell placement tool, it can be seen that both are related to the 'good placement criteria'. Reducing number of edge crossings results in directly connected cells being placed close to each other. The second criterion allows the nodes to be evenly distributed within the placement area as well as show any symmetry within the layout. This not only is an advantage for graph drawing where the aesthetics are improved, but for cell placement, by illustrating the cell connections in an uncomplicated manner. It is worth

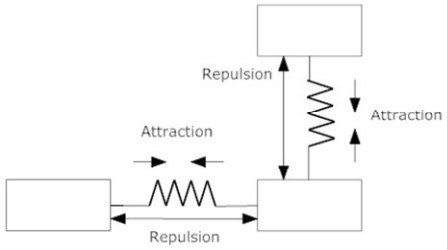

Figure 2 Attraction and Repulsion forces between masses

pointing out that symmetry is a very important heuristic for placement. While most of placement tools have difficulty in incorporating it into their algorithms, the KK algorithm handles it neatly.

The algorithm is implemented by connecting all the nodes on a plane with springs with strength of k_{ij}. Other important variables within the algorithm are, $l_{ij,}$ the desirable or ideal length between nodes and $d_{ij,}$ the shortest distance between the nodes v_i and $v_{j.}$ l_{ij} is calculated by the user providing the side length (*side_length*) of the placement area. The value for the ideal edge length can be directly provided by the user if needed. The algorithm assumes that the initial layout of the graph is one where all the nodes are placed in a circle and during graph drawing only considers one node at a time.

As mentioned above, the KK algorithm defines the placement area using the *side_length* variable that gives the length of the side of the area. Unfortunately, this does not guarantee that the placement will be bounded to a bounding box of height and width equal to the *side_length*. Through direct experimentation, it was seen that at times the nodes can be placed out of the placement area, however it was noted that this was rare and the amount of displacement is quite small compared to the overall placement area. It is believed that limitation on placing components within the placement area can be imposed upon in later stages when being used in the placement tool.

The main difference between the FR algorithm and KK algorithm is that the FR algorithm can handle disconnected graphs. Even though this is not an absolute requirement compared to the objectives of the placement tool, it does give an advantage as to the type of designs the algorithm will be able to handle. In [1] it is pointed out that even though KK algorithm does not support disconnected graphs, it can be easily extended to do so without a significant delay in time.

The main objectives of the FR algorithm are to achieve a visually pleasing graph with increased speed and simplicity. Following Eades work, the FR algorithm also makes use of both attraction and repulsion forces, but takes it one-step further by defining that the attraction forces are only calculated for neighbouring nodes whilst repulsion forces are calculated for all nodes within the graph.

A main feature of the FR algorithm is that it uses a method similar to simulated annealing to control the 'cooling schedule' of the algorithm. This is to help limit the

displacement prohibiting the algorithm to be trapped in local minima. The authors of FR algorithm had also been successful in implementing a placement border in order to keep the nodes within the given area. Unlike the KK algorithm, FR implements support for a customisable placement area. This is quite a useful attribute in cell placement as this defines the placement area and ensures that the cells will be placed within the placement area.

In order to handle disconnected graphs, FR algorithm[2] introduces a separate method which is based on the idea of Kamada and Kawai[1]; partition the graph to its connected components giving each component a region of area proportional to its size, with each component laid out independently. The implementation of a grid variant option is used by FR where the placement area is divided into a grid and nodes are given locations within the grid. Changes are made to the method of calculation of the repulsion forces; it is only calculated for those nodes that are in the neighbouring grids.

Looking at the criteria followed by [2] when drawing graphs, it is seen that two main points are considered. 1) Vertices connected by an edge should be drawn near each other 2) Vertices should not be drawn too close to each other. The first criteria does apply for the cell placement tool as the cells connected to one another will need to be close to each other in order to minimise wirelength. This can be further enhanced by edge weights to ensure that cells connected to edges with higher weights are as close as possible. Unfortunately, the current implementation of the FR algorithm does not contain support for edge weights. The second criteria is set quite vaguely and according to [2] it depends on the number of nodes and the placement area. Literally, this should mean that the nodes do not overlap each other, which is directly applicable to the objectives of the placement tool.

V. Experimentation

Using the Boost implementation of the two algorithms, they were simulated under different conditions to identify their strengths and weaknesses. The inability to handle disconnected graphs of the KK algorithm has proven to be a minor setback in the type of simulations that could be carried out. To start, the two algorithms were subjected to a subset of graphs taken from [1, 2]. The simulations were run on an Intel Pentium IV PC running at 3.2GHz and with 2 GB of RAM.

Table 1 shows the performance results of the two algorithms for the subset of graphs. The runtime, HPWL (half perimeter wire length) and the average edge length are the statistics

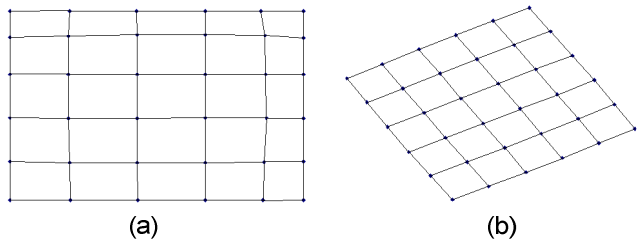

Figure 3 placement of graph7 (a) by FR algorithm with grid variant option (b) by KK algorithm

used in comparing the performance of the two algorithms. Average edge length can be defined as,

$$\text{average edge length} = \frac{\sum \text{edge length}}{\text{number of edges}}.$$

The KK algorithm used in these simulations is a modified version of the boost implementation. One such modification made was to define the initial layout circle to have a radius equal to half the size of the side length. Another was to give a smaller value for the convergence precision in order for it to iterate more. For both algorithms, the placement area was defined to be a square with the dimensions of 100x100 units. When simulating *graph7* the FR algorithm was unable to place it in a visually pleasing manner using the general algorithm. With the grid variant option, it was seen that the algorithm needed to be iterated 15 times before obtaining a pleasing layout. It was observed that for graphs with higher number of nodes and edges, an iteration stage needed to be performed in order to produce aesthetically pleasing layouts.

From the results of Table 1, it can be seen that FR algorithm tends to condense the overall graph resulting in a lower wirelength than the KK algorithm. This condensation would indirectly reduce the overall area when used by a cell placement tool. From the runtime data, it can be observed that KK performs far better, particularly when the number of nodes and edges are higher as can be seen for *graph7*. As the graph-drawing algorithm will be the basis of the placement tool, its speed in generating a layout is very important in ensuring that the overall tool performs just as quickly.

A disadvantage identified with the KK algorithm with respect to the results obtained is shown in Figure 3. It can be seen that the layout produced by the KK algorithm for graph7 is somewhat tilted. For a cell placement algorithm, this introduces an additional obstacle.

Similarly, it was observed that the FR algorithm tends to overlap nodes at times. This is thought to be solvable by introducing a minimum distance between nodes as accomplished by the KK algorithm.

Table 1 Performance results of KK vs FR algorithm for a subset of graphs

Graph	No. of Edges	No. of Nodes	Run time			HPWL			Avg. Edge length		
			KK	FR	GRID	KK	FR	GRID	KK	FR	GRID
1	10	5	0.016	0.109	0.110	1200.59	541.11	541.11	94.72	42.51	42.50
2	6	4	0.015	0.078	0.093	724.27	369.89	369.89	97.14	48.48	48.48
3	15	6	0.031	0.156	0.171	1737.83	734.44	734.44	92.85	38.12	38.11
4	1	2	0	0.031	0.031	100.0	98.81	98.81	100.0	98.81	70.60
5	7	6	0.046	0.109	0.140	217.83	331.77	331.77	25.21	38.55	38.55
6	18	10	0.094	0.281	0.328	801.97	859.92	859.92	34.68	38.16	38.16
7	60	36	2.578	-	24.328	975.79	-	1210.15	11.90	-	20.00

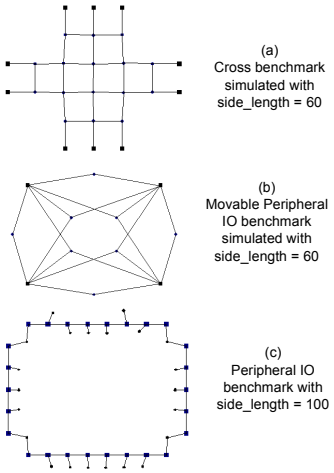

Figure 4 Results from simulating a set of designs from Feature benchmarks on the KK algorithm

The authors have also modified the KK algorithm to implement support for fixed-point nodes. The algorithm was modified with the intention that positional updates are not carried out for the fixed nodes, but the forces of attraction and repulsion are still exerted by them on to the movable nodes. Simulations were run on a set of designs based on the widely used Feature benchmark set which is designed to compare placement quality [15], and the results are shown in Figure 4. The modified designs did not contain the cell dimensions or the initial placement locations that the Feature benchmarks contain. The fixed cells are shown in the nodes represented by a square shape in the results. The *peripheral_IO* design was further modified so that all the fixed cells are interconnected, as KK algorithm does not support disconnected graphs.

Figure 5 looks at a few more simulation results after setting selected nodes to have fixed coordinates. It can be observed from the above figures that the resultant placements have the intuitively correct overall topology. One point to note is that the edge lengths between fixed nodes were smaller than the ideal edge length calculated by the algorithm. This has resulted in the slight change in shape of the designs as can be seen in Figure 5 (b) and (c). Due to the nature of the algorithm in distributing the nodes over the placement area, the movable nodes of Figure 5 (b) have been placed quite far apart. This is not seen as an obstacle for using this algorithm to be integrated in to a placement tool. The distribution of nodes can be limited by, 1) giving a smaller placement area 2) bringing the minimum edge length to be smaller or 3) having a separate stage in the placement tool to remove whitespace and reduce the wirelength.

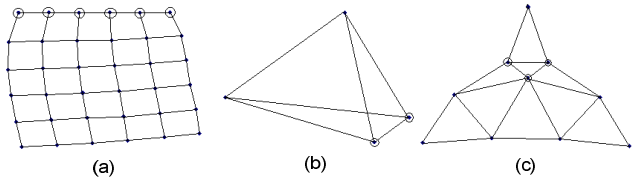

Figure 5 Simulation results of (a) graph7 (b) graph2 (c) graph6 with a selected number of nodes fixed (circled nodes)

Table 2 HPWL calculations for a subset of graphs

Graph No.	Ideal HPWL	Actual HPWL
2	600	724.27
5	212.62	217.83
6	750.79	801.97
7	600	975.69
9	469.59	609.55
10	166.67	341.73
11	372.94	413.352

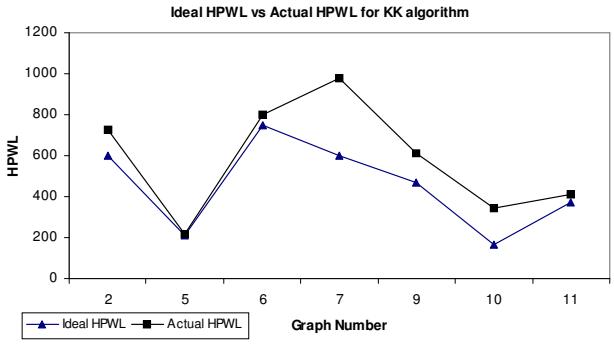

Figure 6 Ideal HPWL and Actual HPWL plotted for a subset of graphs using KK algorithm

Table 2 looks at the variation of HPWL of several graphs as calculated by KK algorithm. Figure 6 displays the data in a graphical format to appreciate the deviation of data. Ideal HPWL was calculated first using a generalised formula for each of the graphs resulting in a unit length HPWL value. This was then multiplied by the minimum edge length calculated by the KK algorithm to give the ideal HPWL. For all the graphs, *side_length* was set to be 100 units. It can be seen for *graph7* the deviation is a significant value. It can be identified that this is due to tilting of the graph as was seen in Figure 3 that causes the increase in wirelength. The same can be said for *graph9* and *graph10*; both hold a grid like topology.

Overall, it was seen that both algorithms were successful in generating a layout with a good topology. The preliminary results obtained from these algorithms suggest that they will be successful in being used as a tool to give a good initial placement for the overall placement tool.

VI. Proposed Placement Tool

The main aim of this research is to pave the way for an improved macro-cell placement tool. The force directed graph-drawing algorithm is to be used for the first stage of the placement tool in determining good topology for the cells. A brief overview of the proposed method is given in Figure 7 and is discussed more in detail below.

Stage 1

The first stage of the placement tool is to construct the graph using the netlist. The necessary data will be extracted from the LEF/DEF netlist files. Important information such as initial cell positions (if any), cell dimensions, nets and net weights etc. will be extracted and stored in a database enabling quick recall of the data at necessary times.

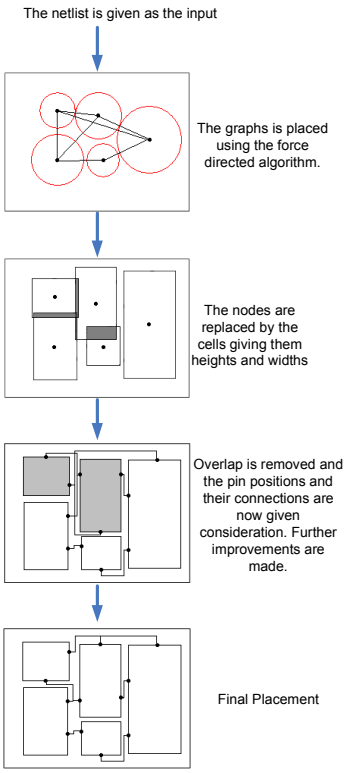

Figure 7 the placement tool overview

In constructing the graph, one of two methods can be used. The first is to consider the cells as zero size points. The pins will be disregarded at this stage and an initial placement will be found by applying the force directed algorithm. In order to minimise the amount of overlap that will be introduced in the later stages it will be needed to introduce a minimum distance between vertices. It is believed that creating individual 'halo' effects will help minimise overlaps as well as obtain more realistic values for the wire lengths. Ref [11] discusses the possibility of ending up with a distorted graph when scaling zero size points to have dimensions. With this in mind, it is suggested that cell sizes should be taken into account when calculating the 'halo' of a given cell.

The second method is to follow the footsteps of [11] and to consider the cell size at the initial placement removing the need for Stage 2 in this overview. A cause for concern of this method is the runtime due to the increased number of calculations when considering cell sizes. Another consideration at this stage is the edge weights. The higher the edge weight the smaller the edge length should be.

Stage 2

In the next stage of the tool, the nodes will be replaced by the actual cell dimensions. As can be seen in Figure 7 there can be some overlap that will be removed at this stage. In [11] this method is described to have the disadvantage of edges not being of uniform length in addition to the possibility of edges being very long. It should be noted that for a placement tool, the edges need not be uniform and since routing can be performed at different layers, the wires will not be routed around the cells.

The pins will be introduced in the placement area to be used for further improvements.

Stage 3 and 4

Once the initial placement with good topology is achieved, the cell placement can be further improved using traditional methods for minimising wirelength and area. Possibility of changing the cell orientation in order to reduce wirelength and congestion will be investigated. Another feature that may need to be investigates is overlap removal.

VII. Conclusion

In this work, a method of using graph-drawing algorithms as a building block for a Macro cell placement has been proposed. Future work will focus on implementing further capabilities on to the graph drawing algorithms. Further simulations will be conducted in order to finalise fine details such as values for the various options presented by the algorithms as well as the different variables that allow tweaking the algorithm to maximise performance. The experiment has demonstrated that force directed graph-drawing algorithm can achieve successful provisional placement, and by subsequently applying traditional wirelength minimisation techniques such as simulated annealing, min-cut partitioning and greedy optimisation techniques, it is believed that this will lead to a high-quality macro-cell placement tool.

References

[1] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters,* vol. 31, p. 15, 1989.

[2] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software- Practice and Experience,* vol. 21, pp. 1129-1164, 1991.

[3] N. A. Sherwani, *Algorithms for vlsi physical design automation*, 3rd ed.: Kulwer Academic, 1999.

[4] R. J. Wilson, *Introduction to graph theory*, Fourth ed. Essex: Prentice Hall, 1996.

[5] J. A. Roy, J. F. Lu, and I. L. Markov: 'Seeing the forest and the trees: Steiner wirelength optimization in placement'. International Symposium on Physical Design, California, USA, 2006, pp. 78-85.

[6] B. K. Choi, M. Sarrafzadeh, T. Taghavi, M. Wang, and X. Yang: 'Dragon2005: Large scale mixed-sized placement tool'. International Symposium on Physical Design, April, 2005, pp. 42-47.

[7] N. Viswanathan, M. Pan, and C. Chu: 'Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control'. Asia and South Pacific Design Automation Conference, 23-26 Jan 2007, pp. 135-140.

[8] A. B. Kahng, S. Reda, and Q. Wang: 'Aplace: A general analytic placement framework'. International Symposium of Physical Design, California, USA, April 2005, pp. 233-235.

[9] F. Mo, A. Tabbara, and R. K. Brayton: 'A force-directed macro-cell placer'. IEEE/ACM International conference on Computer-aided design, November, San Jose, USA, pp. 177-180.

[10] S. Alupoaei and S. Katkoori, "Net-based force-directed macrocell placement for wirelength optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 10, pp. 824-835, December 2002 2002.

[11] D. Harel and Y. Koren: 'Drawing graphs with non-uniform vertices'. Proceedings of Working Conference on Advanced Visual Interfaces, pp. 157–166.

[12] P. Eades, "A heuristic for graph drawing," *Congressus Numerantium,* vol. 42, pp. 194-202, 1984.

[13] F. J. Brandenburg, M. Himsholt, and C. Rohrer, "An experimental comparison of force-directed and randomized graph drawing algorithms," in *Proceedings of the Symposium on Graph Drawing*, 1996, pp. 76 - 87

[14] Boost,http://www.boost.org/, accessed Sep 2007

[15] D. A. Papa, A. Saurabh, and I. Markov, Feature benchmarks for placement,http://vlsicad.eecs.umich.edu/BK/FEATURE/, accessed 03-01-2007