Graph Drawing for Floorplanning with Flexible Blocks

Joanna Ellis-Monaghan
St. Michael's College
Department of Mathematics
Colchester, VT 05439
(802) 654-2660
jellis-monaghan@smcvt.edu

Jamey Lewis
St. Michael's College
Colchester, VT 05439
(802) 654-9451
jlewis@smcvt.edu

Paul Gutwin
Cadence Design Systems
San Jose, CA 95134
(802) 872-0762
pgutwin@cadence.com

Greta Pangborn
St. Michael's College
Department of Computer Science
Colchester, VT 05439
(802) 654-2791
gpangborn@smcvt.edu

ABSTRACT

We discuss further development of a modified force-directed graph drawing placement algorithm for reducing wire length while placing flexible blocks during the floorplanning stage of computer chip design. An effective repelling perimeter allows floorplanning blocks and to pass through each other during early stages of a run in response to spring tensions on the edges, yet repel just enough to avoid overlap in later stages. Pressure equalization equations permit flexible blocks to reshape dynamically in reaction to penetration from neighboring blocks. We present a number of experimental results demonstrating the feasibility of this approach, achieving up to 31% wirelength improvement over commercial tools.

Keywords
Graph-drawing, vertex overlap, floorplanning, spring embedding, force-directed layout, geometric graph theory, geometric visualization.

1. Introduction

A major component of computer chip design consists of generating a netlist layout. A layout provides a geometrically constrained placement of the functional elements and their interconnections as specified by a netlist, which is a large data set encoding the abstract logical structure of a computer chip. Graphs and hypergraphs naturally model the functional elements and interconnections of a netlist. However, rather than being dimensionless points, the vertices represent rectangular chip elements with physical dimensions that must be placed on a chip without overlap, while the edges represent wires

whose lengths are restricted by timing delay constraints. Balancing the conflicting demands of spreading out elements to prevent overlap while simultaneously bringing them close together to minimize timing delays presents significant challenges in the layout process.

Here, we build on our prior work [E-MGLP05], developing a physically motivated graph theoretical model for floorplanning, an early step in the overall layout process. Given a rough high-level grouping of related gates into blocks of fixed area, together with their interconnections, a floorplan arranges these blocks without overlap in the chip region, ideally minimizing wire length (and hence timing delays, thus improving chip performance) by placing connected blocks in close proximity. Often complicating floorplanning are flexible blocks that may be reshaped as long as their areas remain constant, and a very large range of block sizes. For a comprehensive introduction to chip design see [Len90], for graph theoretical approaches, [Cac89], and more briefly, [E-MG03].

Viewed as a purely abstract problem in graph drawing, floorplanning may be formalized as follows. We start with a hypergraph whose vertices are rectangular blocks, each with a fixed area and range of permitted aspect ratio, and whose hyperedges correspond to the wiring connection among a set of blocks. We the convert the hypergraph to a graph by replacing a hyperedge by a claw, that is a copy of $K_{1,n}$, as in Figure 1. For a minor reduction in computation time, we model hyperedges of size 2 or 3 by cliques instead. This introduces a set of dimensionless vertices, *netstars*, which may freely overlap the block vertices. Vertices of a third type, *pins*, also dimensionless, occur in fixed positions along the perimeter of the layout area and represent connections between a block and the edge of the chip.

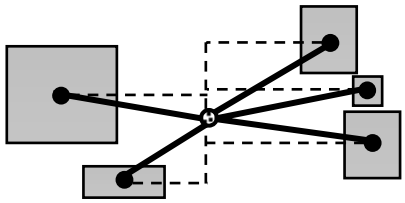

Figure 1. A wiring connection (dotted lines) modeled by a claw on a dimensionless center netstar vertex.

The graph drawing challenge is then to place the rectangular block vertices without overlap within a fixed rectangular area (the chip layout region) so as to minimize the overall edge lengths.

The goal of reducing edge lengths naturally suggests force-directed graph drawing techniques (see [Ead84]), wherein the graph is modeled as a physical

system of masses (the vertices) and springs (the edges), with an iterative algorithm based on Hooke's Law then used to lay out the graph so that the spring system is in equilibrium.

Heuristics that better integrate the three processes of floorplanning, placement (locating the individual components), and routing (locating the wires), have recently led to improvements in the layout process. This approach has been taken in [CCPY00], one of the trailblazers in considering the signal delays during the floorplanning process. By moving optimization techniques usually not implemented until the routing stage into the floorplanning process, they were able to achieve significant improvement in delay reduction for their test case. Similarly, a number of publications and patents (see [VC04], [MTB00], [EJ98], [Kim95], [CKP99], [CCX03], [Shi96], [SMMS94]) describe notable improvements in automated layout by using force directed or interconnect driven methods.

In a netlist layout application, the spring force values correspond to timing delays on the wires, since timing delay (the amount of time for a signal to travel along a wire from one block to another) roughly correlates to wire length. Ideally, this forces floorplan blocks with short delays between them to be placed close together. Unfortunately, the physical dimensions of the blocks usually prevents them from being able to occupy the position determined by a simple force-directed graph drawing algorithm. Various approaches, including bin packing in [MTB00], cell shifting in [VC04], and congestion-based repulsive forces in [EJ98], have been used to address the problem of preventing overlapping. See also [CRS05, DETT99, DH96, FLM94, FR91, FSW99, KK89, Tho04].

We have chosen to work directly with the blocks as physical objects, in part because of the need for a floorplanning model to incorporate the malleability of these floorplan blocks. We considerably modify force-directed graph drawing techniques by developing a physical model that allows the vertex blocks to 'pass through' each other and to adjust aspect ratios as needed while approaching a solution.

The current paper builds on prior work, [E-MGLP05], further refining the various force equations and adapting the algorithm to better incorporate timing the interactions of the attraction, repulsion, and pressure equalization forces. We demonstrate the effectiveness of these efforts with a number of experimental results, achieving up to 31% wirelength improvement over commercial tools on industry data sets.

2. Overview of the Algorithm

As in general for force-directed graph drawing, at the core of our algorithm is an iterative process that repeatedly computes attractive and repulsive forces, with temperature controls to assure convergence. Our use of both local and global temperatures to control oscillation and force the system to

converge to a solution, as well as our spring force computations, are fairly standard. However, we modify the repulsive forces between blocks in three ways. First, the blocks repel from a rectangular perimeter rather than a dimensionless point, and repel only to this perimeter rather than simply according an inverse square law. Second, this repelling perimeter grows outward from the center of the block as the algorithm progresses, beginning very small, and then eventually encompassing the given area of the block. This allows the blocks to pass through one another early in the algorithm as the system grossly positions the components, but prevents blocks from overlapping as the system nears stability. We refer to this perimeter as the *effective* width and height of the block at a given stage of the iteration. Thirdly, we compute boundary pressures (depth of penetration by neighboring blocks) for each block, and then adjust the aspect ratio of the block to equalize these pressures, thus allowing blocks to find good shapes as well as positions in the final floorplan.

Specifics of the spring and repulsive forces are as previously announced in [E-MPGL05], and we discuss the timing curves and pressure equalization advances in further detail below.

Our input consists of:

- the rectangular chip layout area,
- the set of blocks, each with a prescribed area and degree of aspect ratio flexibility (maximum and minimum allowed height and width),
- information about how the blocks are connected among themselves and to the perimeter of the layout area.

The principal components of our algorithm are:

- the spring equations on the edges,
- the repulsion equations on the blocks,
- the pressure equalization equations to reshape overlapping blocks
- timing/temperature controls on the interaction of the above forces.

The output is a floorplan, where we seek to minimize the overlap and wire lengths among connected blocks.

ALGORITHM OUTLINE:

```
while (system has not cooled)
  For each vertex v (block or netstar)
    For each adjacent vertex w (block, netstar, or pin)
      Calculate Spring Force between v and w
  For each vertex v (block or netstar)
    Apply Spring Forces
```

```
If temperature < threshold
  For each vertex v (block)
    For each vertex w (block)
      Calculate the overlap for v and w
      Update the maximum overlap (North, East, South, or West)
    Calculate Pressure Equalization for v
  For each vertex v (block)
    Update the dimensions of v
For each vertex v (block)
  Consider each vertex w (block)
    If v and w overlap
      Calculate the repulsion force of w on v
For each vertex v (block)
  Apply Repulsion force on v
For each block
  Decrease the temperature of the block
Decrease global gate relaxation constants
end while.
```

Critical to the algorithm is how the spring, repulsion, and pressure equalization forces interact with one another. Currently the timing curves for these interactions (under the guise of temperature controls in the code) look roughly as shown in Figure 2. We presently adjust these by hand, but will ultimately initialize these parameters based on raw netlist data such as the number of nets and blocks and the distribution of their sizes.

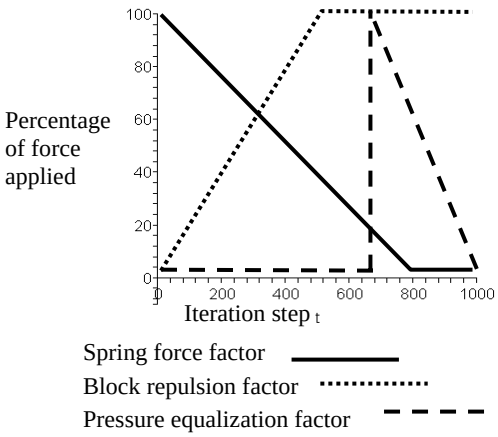

Figure 2. Timing Curves

We have also improved the pressure equalization heuristics. During early floorplanning, it is common for a design to include blocks which do not have a fixed aspect ratio, due to either undesigned logic (black boxes) or modules which only have an area constraint. These flexible blocks may be reshaped as long as their areas remain constant and their aspect ratios stay within given

bounds. Intuitively, this flexibility should lead to better floorplanning solutions than might be possible with rigid blocks. We are able to transform this challenge into an advantage by dynamically reshaping flexible blocks toward the end of a run in response to overlap from neighboring blocks. This has allowed us to find floorplanning solutions with smaller overall wirelengths than with fixed blocks.

For a block *u* we determine the maximum horizontal and the maximum vertical penetrations by any blocks that overlap it. If the penetrations come from neighbors (including possibly the bounding box of the chip region) on all four sides, we reshape the block to equalize the maximum horizontal and vertical penetrations, as in Figure 3. In the case that the penetrations come from opposite sides, with at least one side not penetrated, as in Figure 4, we compress the block only in that direction. If the block *u* is only penetrated on only one side, or only on two adjacent sides, we do not reshape *u*, instead letting the repulsion equations resolve the overlap in this situation. In any case, we do not want to decrease a dimension if the penetration in that direction is greater than half the dimension, since such an overlap is likely to be better handled by either changing the shape of the overlapping block or by the action of the spring and repulsion forces. Thus, we only consider penetrations into less than half the height or width of a block.

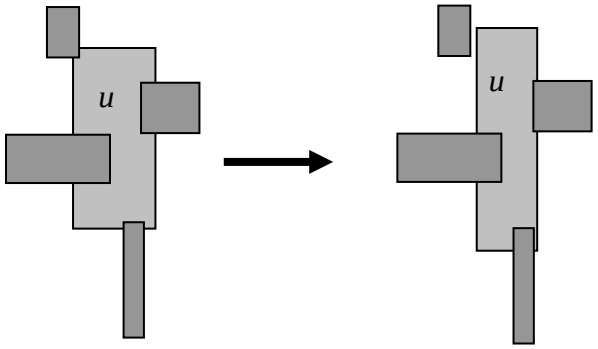

Figure 3. Block *u* reshaping to equalize pressures from all sides.

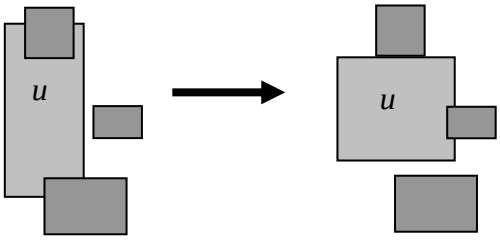

Figure 4. Block *u* responding to penetrations from two opposite sides. The new penetration from the right will be resolved by repulsion forces, moving *u* towards the unoccupied space to the left.

When a block *u* is penetrated on all four sides, then we equalize the horizontal and vertical penetration pressures by setting them equal and resizing

u accordingly, as in Figure 3. For penetrations from two opposite sides, we reduce the appropriate dimension of *u* just enough to clear both penetrations (or to the minimum allowed dimension), as in Figure 4.

As with the spring and repulsion equations, we incorporate the temperature into this scheme in order to prevent oscillation and ensure the termination of the algorithm. Thus, we set the new width of the block to be a convex combination of the previous width and the equilibrium width we have just computed. As the temperature of the vertex goes to zero as the algorithm progresses, the weight placed on the previous width increases. The block reshaping is most critical for fine tuning toward the end of a run, after the blocks have been grossly arranged, so we only implement pressure equalizations later in the process (see Figure 2).

Our present Java implementation is an unoptimized development prototype, with a current $O(n^2)$ run time resulting from the electrical repulsion and pressure equalization equations considering every possible pair of the n blocks. This has been reasonable since floorplans have relatively few components. However, the average run time for large instances may easily be improved by sorting the blocks by location (which will take time $O(nlogn)$) and then for each block examining only the blocks which overlap it. The work required to compute the spring forces is $O(m)$, where m is the number of edges, since these forces only act on adjacent vertices; as long as the graph is reasonably sparse this will not impede the speed of the algorithm.

Currently we randomize block positions as the first step in the run. This was for preliminary stability testing, to be sure we got good solutions independent of the starting configuration, and to test that the algorithm would successfully 'untangle' randomly snarled wires despite the obstruction of the blocks. Although we do see consistently good results independent of the starting position, when there are a small number of very large blocks relative to the chip area then the amount of improvement may depend on the random seed.

3. Experimental Results

We have used several sample floorplans (both synthetic and realistic), as well as small (gate level) data sets to develop our algorithm. The floorplan examples I, II, and III shown below are derived from an ASIC-style demonstration chip and incorporate challenges such as flexible blocks and widely varied block areas. Floorplan I assumes fixed blocks, and further reduction in wirelength was achieved in Floorplan II by allowing up to a 20% change in aspect ratio. Floorplan III, in addition to large blocks with 50% flexibility, incorporates the further challenge of many very tiny blocks that must be arranged around the much larger blocks, a common occurrence in many design examples. To adapt our approach to the presence of these small standard cells, we ran two phases of our algorithm. In the first phase, standard cell components were not subjected to repulsion forces, so they were able to overlap

larger blocks at the end of the first phase. In the second phase, the positions of the large floorplanning blocks were fixed and only standard cell components were subjected to spring forces and to repulsion forces that moved them out of the floorplaning blocks.

Our floorplanning heuristics also apply to small gate-level layout problems, as can be seen from the standard cell ALU and Tdsp_Glue examples. These data sets involve blocks representing individual gates which are rectangles with fixed aspect ratios and fairly uniform sizes.

In all examples, an initial floorplan or layout was created using a commercial auto-floorplanning tool. In a typical chip design flow, the results of auto-floorplanning would be used as a starting point for further floorplan refinement. We compare our results with the initial auto-floorplan results, using total wire length as the figure of merit.

In some cases the commercial tool may impose constraints such as row placement not implemented in our prototype, and also our current implementation leaves some residual block overlap (noted in Table 1, as a percentage of the chip area, and shown in black in the figures) which is prohibited by the commercial tool. However, the improvements are of sufficient magnitude not to be lost during minor post-processing adjustment.

Table 1.

Data Set	# blocks	# pins	# nets	Block overlap	**% wire length improvement**
ALU (fixed blocks)	84	31	107	0.06%	**9%**
Tdsp_Glue (fixed blocks)	550	501	757	0.26%	**17%**
Floorplan I (fixed blocks)	15	57	269	0.03%	**13%**
Floorplan II (flexible blocks)	15	57	269	1.43%	**16%**
Floorplan III (flexible blocks-wide size range)	15	57	269	0.96%*	**31%**

*Large block overlap

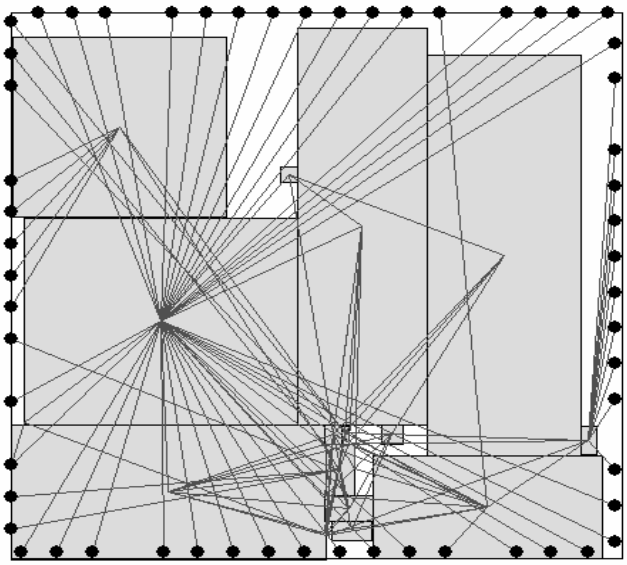

Figure 5: Floorplan I.

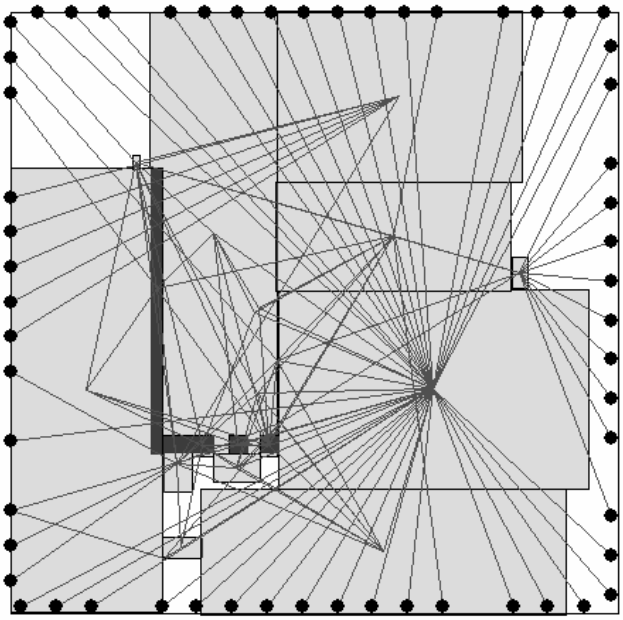

Figure 6: Floorplan II.

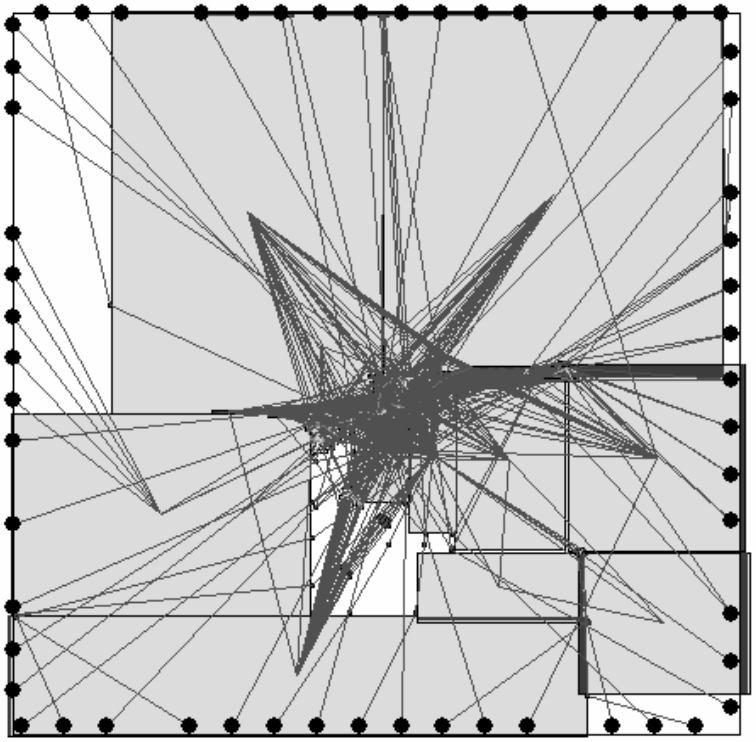

Figure 7: Floorplan III.

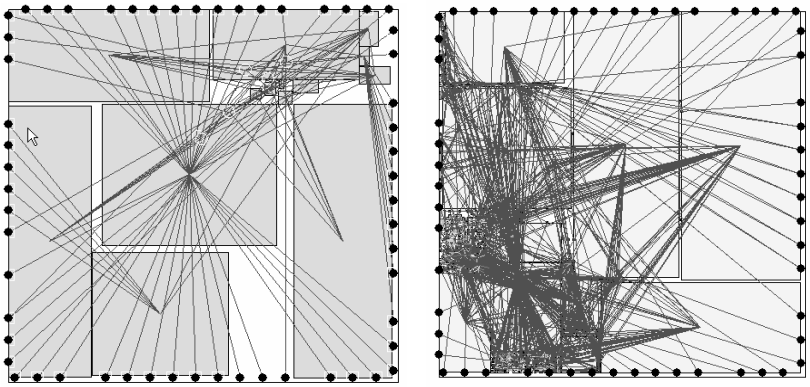

Figure 8: Commercial Layouts for Floorplans I & II (left),
and Floorplan III (right).

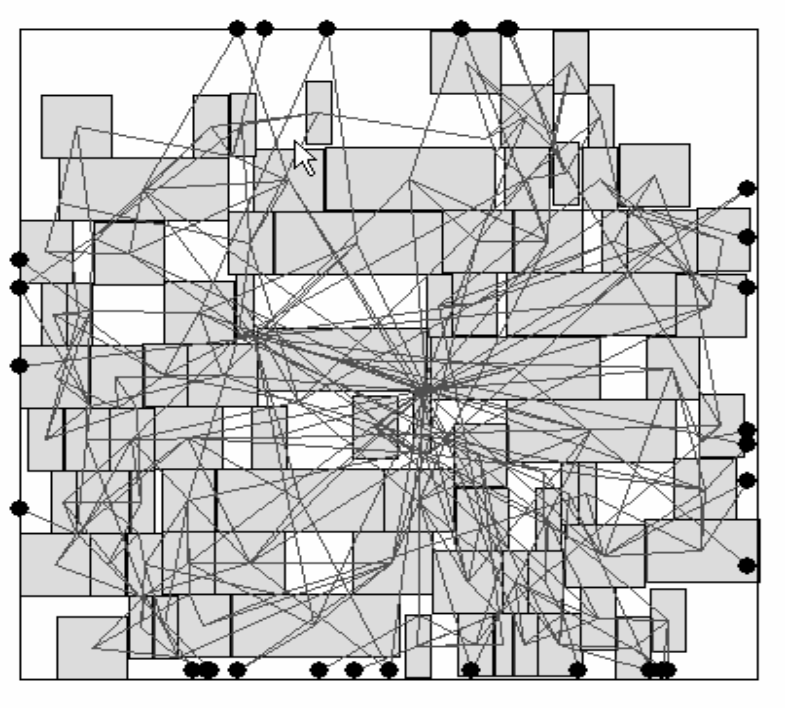

Figure 9: Layout for ALU example.

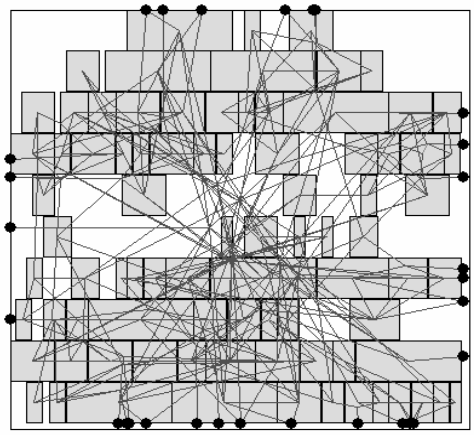

Figure 10: Commercial Layout for ALU example.

Acknowledgements

This work has been supported by VT EPSCoR under grant NSF EPS 0236976, by NASA under Training Grant NGT5-40110 to the Vermont Space Grant Consortium., and by Cadence Design Systems.

4. REFERENCES

[Cac89] L. CACCETAA, Graph theory in network design and analysis, *Recent Studies in Graph Theory*, V.R. Kulli, ed., Vishwa International Publications, 29-63, 1989.

[CCPY00] C.-C. CHANG, J. CONG, D. Z. PAN, X. YUAN, Interconnect-driven floorplanning with fast global wiring planning and optimization, *Proceedings SRC TechCon Conference, September 21-23*, Phoenix, 2000

[CCX03] C.C. CHANG, J. CONG, M. XIE, Optimality and scalability study of existing placement algorithms, ASP-DAC, 621-627, 2003.

[CKP99] J. CONG, T. KONG, Z. PAN, Buffer Block Planning for Interconnect-Driven Floorplanning, *Proc. of Internat. Conf. on Computer-Aided Design*, San Jose, California, 358-363, November 1999.

[CRS05] J. CONG, M. ROMESIS, AND J. SHINNERL, Fast floorplanning by look-ahead enabled recursive bipartitioning. In Asia South Pacific Design Automation Conf., 2005.

[DH96] R. DAVIDSON, D. HAREL, Drawing Graphs Nicely Using Simulated Annealing, ACM Transactions on Graphics, **15**, 301-333, 1996.

[DETT99] G. DIBATTISTA, P. EADES, R. TAMASSIA, I.G. TOLLIS, Graph Drawing, Prentice Hall, NJ, 1999.

[EJ98] H. EISENMANN, F.M. JOHANNES, Generic global placement and floorplanning, *Proc. 35th Annual Conference on Design Automation Conference*, 1998.

[E-MGLP05] J. ELLIS-MONAGHAN, P. GUTWIN, J. LEWIS, G. PANGBORN, *Principles of force directed floorplanning*, Congressus Numerantium, 175, (2005), 81-96.

[E-MP03] J. ELLIS-MONAGHAN, P. GUTWIN, *Graph theoretical problems in next generation chip design*, Congressus Numerantium, 163 (2003), 143-159.

[FLM94] A. FRICK, A. LUDWIG, H. MEHLDAU, A Fast Adaptive Layout Algorithm for Undirected Graphs, *Proc. of Graph Drawings '94, Lect. Notes in Comput. Sci.*, **894**, Springer, Berlin, 388-403, 1994.

[FR91] T. FRUCHTERMAN, E. REINGOLD, *Graph Drawing by Force-Directed Placement, Software-Practice and Experience*, **21** no. 11, 1129-1164, 1991.

[FSW99] A. FRICK, G. SANDER, K. WANG, Simulating Graphs as Physical Systems, *Dr. Dobb's Journal*, 58-64, August 1999.

[E84] P. EADES, A Heuristic for Graph Drawing, Congressus Numerantium, 42, 149-160, 1984.

[KK89] T. KAMADA, S. KAWAI, An Algorithm For Drawing General Undirected Graphs, *Inform. Process. Lett.*, **31**, 7-15, 1989.

[Kim95] M. KIM, *Method and system for providing a non-rectangular floor plan*, Patent 5,398,195, March 14, 1995.

[Len90] T. LENGAUER, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley, 1990.

[MTB00] F. MO, A. TABBARA, R.K. BRAYTON, A force-directed macro-cell placer, *IEEE/ACM International Conference on Computer Aided Design*, November 2000, ICCADOO, Santa Clara.

[Shi96] H. SHIKATA, *Method of and apparatus for placing blocks in semiconductor integrated circuit*, Patent 5,493,510, February 20, 1996.

[SMMS94] H. SHIKATA, Y. MURAISHI, S. MORIYA, N. SEKI, *Method of and apparatus for designing circuit block layout in integrated circuit*, Patent 5,309,371, May 3, 1994.

[Tho04] C. THOMASSEN, Tutte's Spring Theorem, published online, Wiley InterScience (http://www.interscience.wiley.com), DOI 10.1002/jgt.10163, February 2004.

[VC04] N. VISWANATHAN, C. CHU, FastPlace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net, *ISPD*, April 18-21, Phoenix, AZ pp. 26-33, April 2004.