Graph Theoretical Problems in Next-Generation Chip Design

Joanna A. Ellis-Monaghan
Department of Mathematics
St. Michael's College
Colchester, VT 05439

Paul Gutwin
Cadance Design Systems
San Jose, CA 95134

Abstract

A major component of computer chip design is creating an optimal physical layout of a netlist, i.e., determining where to place the functional elements and how to route the wires connecting them when manufacturing a chip. Because of its basic structure, the overall problem of netlist layout contains many questions that lend themselves to graph theoretical modeling and analysis. We will describe the basic principles of netlist layout and present several graph theoretical questions inherent in the problem. Possible approaches to these questions include concepts from hypergraphs, graph partitioning, graph drawing, graph and geometric thickness, tree width, grid graphs, planar embeddings, and geometric graph theory.

Keywords: Netlist layout, hypergraphs, graph partitioning, graph drawing, graph and geometric thickness, tree width, grid graphs, planar embeddings, and geometric graph theory, routing, wiring, placement.

Introduction

A netlist is the logical description of the function of a computer chip. The netlist contains functional elements (often basic Boolean operators like AND and NOT) and the interconnections between them. The process of converting this logical description into the physical design of a chip is a major portion of the complete chip design. The physical design of the chip is the detailed specification of the location of each functional element and wire interconnection.

Because of the basic structure of a chip (wires connecting nodes), graphs and hypergraphs are natural choices for modeling various aspects of the layout problem. We seek to describe integrated circuit layout in terminology familiar to graph theorists and thus make a potentially rich area of research more accessible. We discuss five components of netlist layout in which a graph theoretical approach is particularly applicable: partitioning the netlist, leveraging the hierarchy, identifying congestion, measuring congestion, and automating small configurations.

As might be expected, almost every aspect of netlist layout involves problems that are known to be NP hard. Thus, the goal is to find either fast approximating heuristics or, what may be of more interest to the theoretician, graph theoretical results from approximating models that in some way inform the larger problem.

The process of converting a netlist into the physical design of a chip is traditionally broken into three phases: *floorplanning*, in which the very large and "special" functional elements are assigned locations in the chip area; *placement*, in which the remaining functional elements are assigned locations in the chip; and *wiring* or *routing*, in which the interconnect wires are assigned locations. The area for placing the functional elements and legal locations for the wires is constrained, sometimes severely, by the implicit demand of minimizing chip cost. This often leads to wiring "congestion," where a large number of wires need (ideally) to be routed through the same area to minimize the length of the wires.

The performance (operational speed) of a chip is determined by the delay of the worst path through the netlist. The delay of a path through the netlist is the sum of the delays through the functional elements and the delays due to the interconnect wires. An upper and lower bound exists for every path on the chip, with the upper bound dictated by the operational speed of the chip. Historically, the delay of a functional element was much greater than the delay of the wires. This allowed the wire delay to be effectively ignored during placement and wiring. Today, the interconnect (wire) delay has become a significant portion of the total path delay and is one of the compelling reasons researchers are seeking effective models for predicting wireability [SBG99] or methods for minimizing the wiring resources needed [HKR94].

Netlist partitioning is a fundamental part of the process of chip design. The size of a netlist, upwards of 3 million functional elements today, makes it computationally intractable. Thus, the netlist must be partitioned into more tractable components, while considering the delays on the wires. This amounts to a hypergraph partitioning problem that is NP hard. Since an optimal solution is not feasible, the goal is to find fast approximation heuristics for partitioning the netlist while respecting the delays.

Leveraging the hierarchy is actually a possible approach to the netlist partitioning problem. A chip is not designed on the netlist level. Rather, designers think in terms of a hierarchy of functional modules. Often (but not necessarily), it may be efficient to place the nodes comprising one of these modules in the same area on the chip. The question is whether, and precisely how, it is possible to use this structural hierarchy to inform decisions about netlist partitioning.

Wiring congestion occurs when too many nodes and too many wires with very short delays try to occupy the same small area on a chip. Given that a chip has finite wiring resources, congestion comes in two forms: *legal* congestion, where all the wires in a given area have legal locations; and *illegal* congestion, where some wires have no legal assignment. The problem is to identify potential legal and illegal congestion from the netlist prior to the layout and wiring process. The terms "legal" and "illegal" congestion are not standard terminology in the VLSI design industry. We use them here simply to reflect the complex rules governing placement and routing without introducing technical electrical issues.

A physical design with an illegally congested netlist cannot be either manufactured or functional. A physical design with a legally congested netlist will incur high manufacturing costs and may not function correctly. Because of this, the industry seeks a "congestion metric," namely, a way to estimate the amount of congestion from the netlist before attempting placement and wiring of the chip [HF99].

A related problem is to identify potentially congested substructures from the netlist prior to the layout and wiring process. These substructures, if they could be identified and "cut out" from the netlist, might then be processed separately from the netlist as a whole [CCPY02].

Once the netlist has been roughly partitioned, the ability to automate the layout of small, often repeated configurations, or even the potentially congested substructures mentioned above, becomes valuable. While there are preset layouts for some very regular configurations, the industry currently seeks improved general placement and routing heuristics with provable optimization bounds.

Basic Definitions

Encouraging the extension of graph and hypergraph results into the topology of the wiring space, or reasonable reductions of it, is one of the primary goals of this attempt to make netlist problems more accessible to graph theorists. To this end, we provide somewhat more detailed information defining and formalizing the essentials of netlist layout, and in particular the wiring

space, than is given in the subsequent problem descriptions. The topology of the wiring space, both with and without the application-driven delays, is of intrinsic theoretical interest with respect to graphs and hypergraphs, in the same sense as are graph and hypergraph embedding and optimization questions in other spaces such as grids, lattices, planes, and other surfaces.

We first need to define and formalize the essentials of netlist layout. Since solutions will be scalable, we assume unit spacing for simplicity.

The *placement layer* is a rectangle $[-a,a]\times[-b,b]$ in the *XY* plane, and the *placement points* are the integer lattice points in the placement layer: $\{(x,y,0)|x,y\in\mathbb{Z},\ -a<x<a,\ -b<y<b\}$.

The *wiring layers* are the sets

$$L_i=\begin{cases}\{(x,y,i)|y\in\mathbb{R},\ x\in\mathbb{Z},\ -a<x<a,\ -b<y<b\},\\ \qquad \text{if } i \text{ an odd positive integer (a horizontal layer),}\\ \{(x,y,i)|x\in\mathbb{R},\ y\in\mathbb{Z},\ -a<x<a,\ -b<y<b\},\\ \qquad \text{if } i \text{ an even positive integer (a vertical layer)}\end{cases}.$$

Currently, there are typically fewer than ten wiring layers.

A *via* is a connector between layers above a placement point — in other words, a line segment of the form (r,s,t) where *r* and *s* are integers and *t* ranges from 0 to 1 (to connect the placement layer to the wiring layer above it) or from $i-1$ to i (to connect a vertical with a horizontal layer).

The *wiring space* consists of a placement layer with specified placement points, plus several wiring layers. Figure 1 shows a wiring space with vias and one horizontal and one vertical wiring layer.

A *pin* is a vertex and may be labeled as an input, denoted p_i, or an output, denoted p_o. Pins are placed on placement points.

A *gate* is a set of pins, and the size of a gate is the number of pins in it (from 2 to 1,000, with a mean of roughly 3.5). A gate occupies a rectangular region of the placement layer that contains all its pins. The area of the rectangle is the area of the gate. Other than this containment requirement, the size and the area of a gate are not related, so a gate may occupy a much larger region than the minimum rectangle containing all its pins. By abuse of language, the term *gate* will also be used to refer to the rectangular region it occupies. Gates may not overlap on the placement layer.

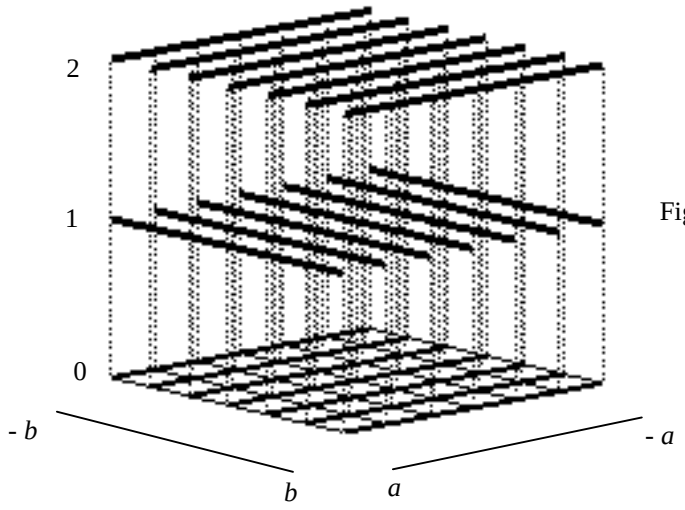

Figure 1.

A *signal* is a set of pins, no two of which are in the same gate, and exactly one of which is an output pin on its gate, which is also called the source pin of the signal. Thus, if we ignore the input/output information, a signal can be thought of as a hyperedge on the pins it connects. The size of a signal is the number of pins in it (from 2 to 1,000, with a mean of 3.5). Each pin is in at most one signal.

A *wire* of a signal *s* is a subset of the layout space that is (topologically) connected and that contains *s*, but no pins not in *s*. The *length* of a wire is the sum of the lengths of its segments in the L_i's. The *total length* includes the lengths of any vias in the wire; however, length is more important than total length in most models. We ignore the other physical dimensions and electrical properties of the wires.

Note that in industry literature, the word *wire* is used in two different contexts to mean two different things. When considering the netlist, "wire" and "logical wire" are synonymous with "signal" as defined above. When considering the physical layout, "wire" and "physical wire" mean "wire" as defined above. Since, in practice, the netlist and the physical design are usually considered independently, the meaning of "wire" is clear from the context. However, in this paper, we will use "signal" and "wire" as defined above to distinguish the abstract from the physical connection among a set of pins.

The *distance* between two pins p_i and p_o of different types in the same signal is the sum of the segments in the L_i's of a shortest path from p_i to p_o within the wire. The *delay* between two pins p_i and p_o is the maximum distance between p_i and p_o allowed in any wire connecting them (we assume that delay is proportional to distance, and that via delay is negligible). A delay may be infinite.

The *fanout* of a gate is the number of output pins it contains, and the *fanin* is the number of input pins it contains. The *fanout* of an output pin is the number of input pins in its signal.

Formally, then, a *netlist* is a quadruple, $N(P, G, S, D)$, where P is a set of pins, G is the set of gates (so G is a partition of P), S is the set of signals (so, since each pin is in at most one signal, S is also a partition of P), and D is an assignment of delays between every two pins of opposite type in the same signal.

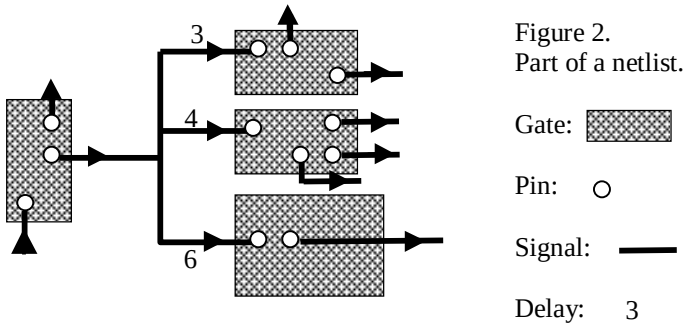

Figure 2.
Part of a netlist.

A *layout* of a netlist $N(P, G, S, D)$ consists of three separate but closely related problems. The first is *floorplanning*; a floorplan is a rough high-level grouping and locating of related gates within the chip. The second is the specific placement of the gates on the placement layer so that the pins are on the placement points and the gates are pair-wise disjoint. The third is the routing of the wires in the wiring space, so that the distances between every pair of pins of opposite type in the same signal are less than their delays and so that the set of wires is pair-wise disjoint (topologically).

Currently, the problems of floorplanning, placement, and routing are typically treated separately. A floorplan is developed for the netlist, often by hand. Then a set of heuristics is used to place the gates/pins. Finally, considerable effort is devoted to the routing problem. Heuristics that better integrate these procedures may lead to improvements in the layout process. Recent efforts in this direction may be found in [CCPY00].

The area of a layout is the area of the minimum closed rectangle containing all the gates, and the volume of the layout is the volume of the minimum closed box containing all the wires.

A layout is *optimal* if the area is minimal given the number of wiring layers, if the distance between pins is always less than or equal to the prescribed delay, and if the total length of wire (not counting vias) is minimal. The overall problem is to find a heuristic for generating optimal layouts.

Some Approximating Reductions

Note that, as described above, a netlist is a hypergraph on the set of pins, such that each pin is in exactly two hyperedges (a gate and a signal), together with the delay information. However, as with any complex problem, it is often productive to reduce the general problem by choosing models that, although perhaps less accurate, are more tractable. In various contexts, researchers have simplified the above model in several ways. Among these are treating the gates as the vertices of a hypergraph — in other words, contracting the hyperedge associated with each gate to a single point. Further, each signal hyperedge may be replaced by a clique or directed claw.

Similarly, the topology of the wiring space may be simplified, for example by approximating it by a grid, or by one or more layers of grids, or even by several planar layers, ignoring orthogonality. It may be modeled by a grid in which edges are allowed to cross at grid points (see [Tho83] for a formalization). In some approaches, the delay information may be ignored, at least temporarily.

Results that apply to such approximating structures, if they can be made to inform the original problem, are potentially beneficial to the industry.

Netlist Partitioning

Because a netlist can contain several million pins, the most natural approach is to subdivide it into more manageable portions. The basic idea is to do the placement and routing of smaller pieces separately and then make the necessary connections between these larger components. The difficulty is that after these larger components have been placed, it may be problematic to route the connecting wires, or it may not be possible to meet the delays between pins in different components.

Thus, the goal is to partition the gates in the netlist so that a minimum number of signals have gates in different parts of the partition. Furthermore, the partitions cannot be very uneven in size. Since in this context the netlist is simply a hypergraph with the gates as the vertices and the signals as the

hyperedges, this is a question of hypergraph partitioning. However, even graph bisection into bounded sets is NP hard (see [GJS76], and for the hardness of approximation see [ALMSS92]), so optimal netlist partitioning is unfeasible.

Alpert and Kahng [AK95] provide a comprehensive survey of both problem formulation and solution approaches to netlist partitioning. Combinatorial approaches include max flow/min cut, labeling, and covering. An important consideration in modeling the problem is that reducing the hypergraph structure to a graph model may fail to adequately capture essential characteristics, in particular the delays. For example, if the signals are converted to cliques, Ihler, Wagner, and Wagner [IWW93] have shown that there is no satisfactory assignment of weights to the clique edges to model the cost of cutting the hyperedge.

However, there are graph and hypergraph partitioning algorithms such as MeTiS and hMeTiS that show experimental success (see [KAKS99]). Also, for graphs, under certain conditions such as edge density, there are polynomial time approximation algorithms for NP problems, many based on the algorithmic version of the graph regularity lemma of [ADLRY94]. Analogously, an algorithmic regularity lemma for hypergraphs was given in [CR01], and applied to hypergraph partitioning in [Czy02]. The challenge remains in the possibility of adapting such techniques for practical application to the partitioning problem.

Leveraging the Hierarchy

One possible approach to the netlist partitioning problem is to use information from the conceptual hierarchy that results from the netlist design process. The netlist design team provides this potential partitioning of the netlist in the form of a logical design hierarchy. This nested hierarchy partitions the design in a way that is conducive to the netlist design challenge (creating and verifying the netlist from the design specifications) although not necessarily to the physical design (layout) task. Experienced design teams with adequate resources often create a netlist with a hierarchy that has a one-to-one mapping with an excellent physical partitioning. To the degree that this is not true (due to schedule constraints or lack of designer resources, for example), the physical design team tries to create a one-to-one mapping from the logical design hierarchy to physical design partitions.

Figure 3 gives a simple example of a logical hierarchy that illustrates the inclusive relationships between different modules in the hierarchy. Here, the designer has chosen to break the “DSP” unit into a “MEM” unit and a “LOGIC” unit. All of the functional elements for the DSP are specified in that section of the netlist. Note, however, that there may be many signals connecting different nodes of the hierarchy tree — for example, connecting a functional element in the CACHE with another functional element in the MEM_CTL.

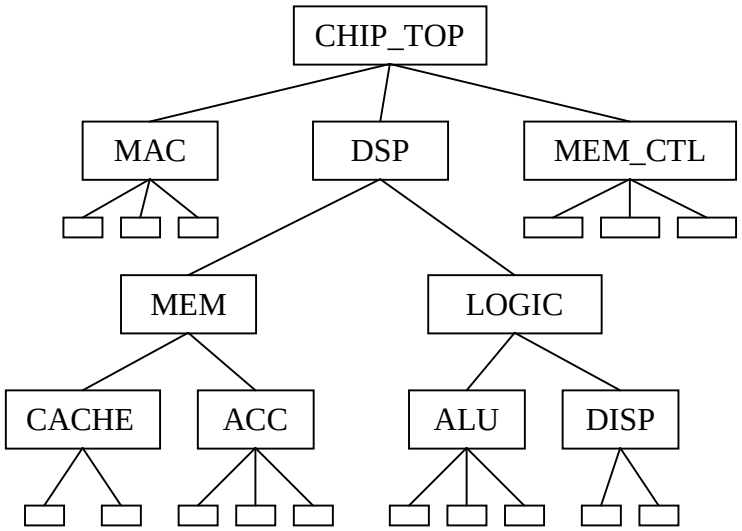

Figure 3.

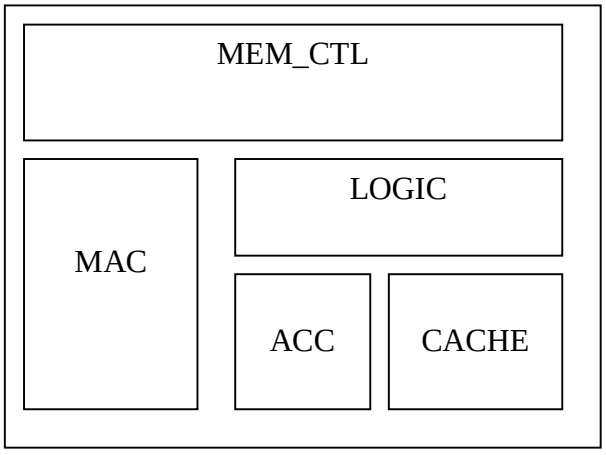

Figure 4.

During the floorplanning process, a separate physical hierarchy is created, and a mapping from the logical hierarchy to this physical hierarchy is derived. The physical hierarchy is used to group together elements from the netlist that need to be physically close together. The physical hierarchy is often a proper subset of the logical hierarchy in that the physical hierarchy is a two-level hierarchy (a root with a complete cut set of the logical hierarchy tree as children). It is generally represented as a plan view of the actual chip, similar to the rooms of a house (thus, the term "floorplan"). A simple floorplan of the logical hierarchy in Figure 3 is given in Figure 4.

Ultimately, the goal is to determine which groupings in the hierarchy tree should remain together during the floorplanning process, and which should be subdivided or regrouped in order to minimize delays and routing difficulties. Treating this as an abstract problem, one possible approach would be to improve the initial design hierarchy incrementally and iteratively by implementing "local" operations — for example, those within a given level or between parent and child nodes — to transform the design hierarchy into a usable physical partition.

A more pragmatic engineering approach has been taken in [CCPY00], one of the trailblazers in this relatively new approach of forcing the floorplanning process to consider the signal delays. By moving optimization techniques such as interconnect topology optimization and layer assignment — 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.

Measuring Congestion

Congestion occurs in a physical layout when the pin density (pins per unit placement layer area) and average fanout are high in a localized area, and the wires must be as short as possible to meet the timing constraints. Automated wiring tools are used to find routes for each signal in the design. A congested design with a legal wiring assignment is prone to longer wires (with the attendant larger wire delays) and signal crosstalk (interference between two adjacent signals). A congested design with an illegal wiring assignment must be manually modified to make it legal (if possible) either by changing the wiring route or by removing a subset of the routed wires and rerunning the wiring tool or both. These problems are very expensive to address in terms of both scheduling and workforce resources.

Since the degree of wiring congestion is so central to the quality of the physical design of the chip, and the process of creating a physical layout for a chip from the netlist is so time consuming, companies need a means of estimating the congestion inherent in a given netlist *before* attempting the physical layout processes.

Three major factors affect congestion: the number of wiring layers available, the interconnectedness of the netlist, and the delay constraints.

Since the number of layers available in the wiring space obviously determines a critical parameter for potential congestion, various measures of the thickness of a graph may inform this problem (see [DH91] for an overview of and relations among several embedding parameters of graphs). Recall that graph thickness is the minimum number of planar subgraphs whose union is the original graph (see [MOS98] for a survey). Geometric thickness adds the restriction (very important in this context) that the planar subgraphs must be on the same arrangement of the vertices in the plane and have straight line segments as the edges (see [DEH98], or [HSV96] and [HSV99], where geometric thickness two is called doubly linear). A critical concern about this approach is that graph thickness is in general NP hard ([Man83]). Although there are heuristics for approximating graph thickness (several are compared in [MOS98]), this computation complexity presents an implementation obstacle.

Furthermore, the problem actually requires something more specific: what might be described as the thickness-dependent area of a hypergraph. The number of wiring layers available is known. The question then becomes how to find a hypergraph measure that, given the number of wiring layers, returns the minimum area on the placement layer necessary to embed the hypergraph.

Working along these lines, Wood [Woo], by modifying geometric thickness slightly to allow each edge to have a single bend, has shown that $O\sqrt{m}$ layers are required above a grid of size $\lceil\sqrt{n}\rceil\times\lceil\sqrt{n}\rceil$, where m is the number of edges and n is the number of vertices of a graph. Furthermore, he provides an algorithm that with high probability produces a drawing in $O\left(m\log^3 n\log\log n\right)$ time. Archdeacon [Arc], has also proposed (and found some preliminary bounds for) the *grid-width* as a possible measure, where the grid-width of a graph G is the smallest number n such that G is a minor of a two-layer $n\times n$ grid.

Since congestion is a consequence of the interconnectedness of the netlist, it also seems reasonable that various measures of graph and hypergraph connectivity might inform this problem. The most common connectivity measures, edge and vertex connectivity, are too sensitive to local phenomena to measure global congestion. More promising candidates include tree- and path-width and the bramble number. Excellent surveys of these more complex measures can be found in [Ree97] and [Bod93]. In general, connectivity measures need to be adapted to account for the hypergraph structure, the delays, and the computational magnitude of the problem; as always, the challenge remains bridging the gap between theory and practice.

Effectively modeling the delay constraints is as large a stumbling block in this problem as in others, and congestion measures that incorporate delay information in a meaningful way are highly desirable.

Identifying Localized Congestion

In some cases, congestion may be localized. In other words, there may be identifiable substructures (of roughly 2,000 to 200,000 pins) in the netlist in which the confluence of numbers of pins, gates, signals, and shortness of delays makes that small piece of the netlist difficult to place and route, independent of the rest of the netlist. If such substructures could be identified, they could be placed and routed separately, or their delays reassessed or even redesigned, depending on the severity of the congestion.

A typical example of such localized congestion occurs in a crossbar switch. Conceptually, this is just a bipartite graph on two sets of pins, with every pin in one set of pins communicating with every pin in the other set, as in Figure 5. The difficulty lies in the nature of the communication, which is through an intermediary set of gates and signals, typically with very short delays, as illustrated in Figure 6, which details the triangular region highlighted in Figure 5. This intermediary set of gates and signals may or may not be regular and may or may not be the same for each pin on the left-hand side of Figure 5.

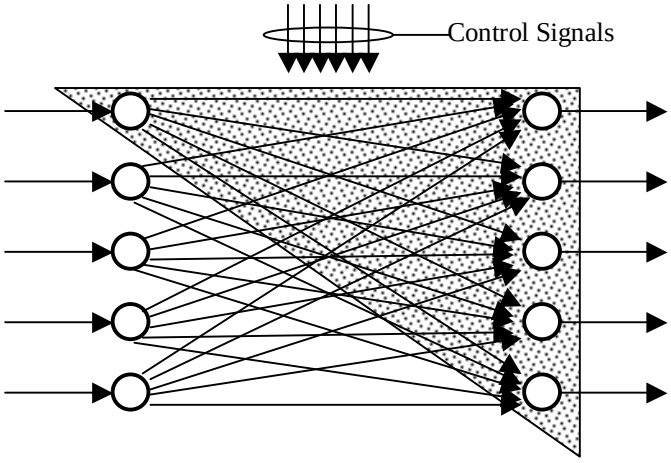

Figure 5.

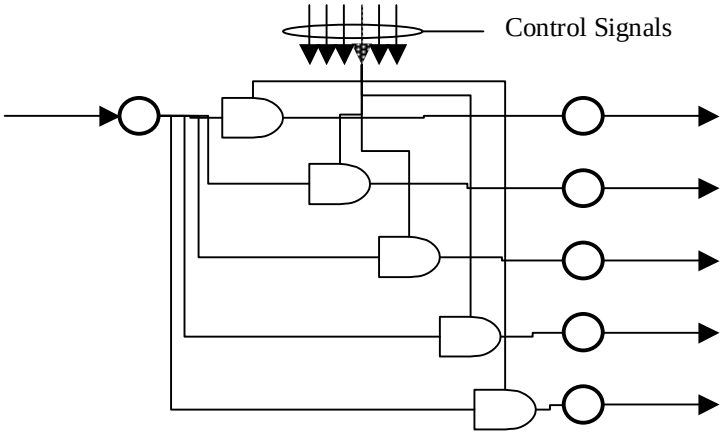

Figure 6.

While some such configurations are readily identified, having been deliberately introduced to meet specific design requirements, other such snarls of interconnections evolve insidiously as part of the design process. These latter may not become apparent until they cause serious problems during the wiring process. Thus, a means of identifying them from the netlist information prior to placement and routing would be valuable.

One possible approach to this problem would be to consider the restrictions inherent in the topology of the wiring space. For example, obstruction sets for hypergraph planarity have recently been determined [Deb02]. It would be very useful to find the set of hypergraph obstructions in the wiring space, since such structures would then be known to be impossible to place and wire. Even a partial list of such obstructions would be of benefit to the industry, since structures containing any of them might then be avoided even in the design stage and thus would never appear in a netlist.

Automating Small Configurations

Some configurations (substructures of a netlist known as *datapath* structures) are relatively easy to place and route: The fanouts may be low and uniform; the logic may permit a simple left-to-right topology; there may be no, or very few, circuits formed. In such cases, common statistical models apply and can accurately predict the congestion of the design.

Unfortunately, problematic configurations, such as the crossbar switches described above, still need to be placed and routed. Complications include

irregular fanouts, many circuits, high interconnectivity. There are probably minimal layouts for some regular configurations such as butterfly switches (see [MPSS99] for several examples on an integer grid model). However, the industry advances so rapidly that highly specific results may be obsolete even before implementation.

As when analyzing congestion, reasonable bounds for small configurations would be valuable here — for example, being able to analyze a small configuration and determine that, irrespective of delays, it will require a footprint of area at least A on the placement layer given that there are n wiring layers. Even better would be a constructive approach, which then provides a possible layout, such as is given in [Woo], for graphs with one-bend edges.

The "inverse" problem of determining whether it is possible and, if so, how to fit a given hypergraph into a fixed volume is also of interest. Significant progress has. recently been made for doing something very close to this in [BCMW], at least in the case of three-dimensional grid drawing. [BCMW] gives an exact formula for the maximum number of edges possible in a graph drawn in a three-dimensional grid of fixed dimensions. However, grid-drawings permit vertices to lie on any lattice point, not just the placement layer: They permit both vertical and horizontal movement on the wiring layers; and they apply only to graphs, not general hypergraphs. Thus, more work must be done to adapt such techniques to the exact demands of the placement and routing problem.

A wide range of other graph drawing techniques (see [BETT94] for an extensive survey, and [BETT99] for a comprehensive text) have been brought to bear not only on automating small configurations, but also on other aspects of the layout process, often with considerable success. A random sample of some recent techniques might include force-directed drawing ([MTB00]), rectangular drawing ([RNN02]), and orthogonal grid drawing ([TTV00]). The important considerations are: applicability to the actual wiring space, modeling the delays, realizing the areas occupied by the gates, minimizing the footprint in the layout space, computational speed, and provable optimality.

Conclusion

We have tried to provide a brief, graph-theorist-friendly introduction to some of the interesting combinatorial problems that arise in the area of integrated circuit layout design. Although the industry advances so rapidly that any resource becomes quickly dated, [Len90] is an excellent introduction to the technical considerations of chip layout. The text provides a systematic and detailed discussion of a wide variety of layout problems, including those briefly described in this paper. Since it was written to "accelerate the process of unification" between combinatorial theorists and engineering practitioners, it

succeeds in bringing many technical engineering problems in circuit layout design into the reach of graph theory tools.

Bibliography

[ADLRY94] N. ALON, R. A. DUKE, H. LEFMANN, V. RÖDL, R. YUSTER, The algorithmic aspects of the regularity lemma, *J. Algorithms*, **16**, 80-109, 1994.

[AK95] C. ALPERT, A. KAHNG, Recent directions in netlist partitioning, *Integration, the VLSI Journal*, **19** (1-2), 1-81, 1995.

[ALMSS92] S. ARORA, C. LUND, R. MOTWANI, M. SUDAN, M. SZEGEDY, Proof verification and hardness of approximations problems, *Proceedings of the 33rd IEEE Symposium on Foundations of Computing*, 14-23, 1992.

[Arc] D. ARCHDEACON, A stroll through the brambles and the trees, *private communication*, 2001.

[BCMW] P. BOSE, J. CZYZOWICZ, P. MORIN, D. WOOD, The maximum number of edges in a three-dimensional grid-drawing, *preprint.*

[BETT94] G. DI BATTISTA, P. EADES, R. TAMASSIA, I. G. TOLLIS, Algorithms for drawing graphs: An annotated bibliography, *Comput. Geom. Thry. Appl.*, **4**, 235-282, 1994.

[BETT99] G. DI BATTISTA, P. EADES, R. TAMASSIA, I. G. TOLLIS, *Graph Drawing*, Prentice-Hall, 1999.

[Bod93] H. L. BODLAENDER, A tourist guide through treewidth**,** *Acta Cybernetica*, **11** (1-2), 1-21, 1993.

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

[CCPY02] C.-C. CHANG, J. CONG, D. Z. PAN, X. YUAN, Physical Hierarchy Generation with Routing Congestion Control, *Proc. International Symposium on Physical Design, 2002*

[CR01] A. CZYGRINOW, V. RÖDL, An algorithmic regularity lemma for hypergraphs, *SIAM Journal on Computing*, **30** (4), 1041-1066, 2001.

[Czy02] A. CZYGRINOW, Partitioning problems in dense hypergraphs, *Discrete Applied Mathematics*, **116**, 179-191, 2002.

[Deb02] M. DEBOWSKI, Results on planar hypergraphs and on cycle decompositions, *Master's Thesis, University of Vermont, 2002.*

[DEH98] M. DILLENCOURT, D. EPPSTEIN, D. HIRSCHBERG, Geometric thickness of complete graphs, *Graph drawing (Montréal, QC, 1998), Lecture Notes in Comput. Sci.*, **1547**, *Springer, Berlin*, 102-110, 1998.

[DH91] A. DEAN, J. HUTCHINSON, Relations among embedding parameters for graphs, in: Y. Alavi et al. (Eds.), *Graph Theory, Combinatorics, and Applications*, 287-296, Wiley, 1991.

[GJS76] M. R. GAREY, D. JOHNSON, L. STOCKMEYER, Some simplified NP-complete graph problems, *Theor. Comput. Sci.*, **1**, 237-267, 1976.

[HF99] P. HUNG, M. FLYNN, Deep Submicron VLSI Floorplanning Algorithm, *Electronic Devices and Systems Conference, November 1999.*

[HKR94] L. HAGEN, A. B. KAHNG, F. J. KURDAHI, C. RAMACHANDRAN, On the intrinsic Rent parameter and spectra-based partitioning methodologies, *IEEE Transactions on Computer-Aided Design* **13** (1), 27-37, *January 1994.*

[HSV96] J. P. HUTCHINSON, T. SHERMER, A. VINCE, On representations of some thickness-two graphs, extended abstract, in: F. Brandenburg (Ed.), *Lecture Notes in Comput. Sci.*, **1027**, *Symposium on Graph Drawing DG '95, Passau, Germany, Springer, Berlin*, 324-332, 1996.

[HSV99] J. P. HUTCHINSON, T. SHERMER, A. VINCE, On representations of some thickness-two graphs, *Comput. Geom.*, **13** (3), 161-171, 1999.

[IWW93] E. IHLER, D. WAGNER, F. WAGNER, Modeling hypergraphs by graphs with the same mincut properties, *Information Processing Letters*, **45** (4), 171-175, 1993.

[KAKS99] G. KARYPIS, R. AGGARWAL, V. KUMAR, S. SHEKHAR, Multilevel hypergraph partitioning: Application in VLSI domain, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **7**, 69-79, 1999.

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

[Man83] A. MANSFIELD, Determining the thickness of graphs is NP-hard, *Math. Proc. Cambridge Philos. Soc.*, **93** (1), 9-23, 1983.

[MOS98] P. MUTZEL, T. ODENTHAL, M. SCHARBRODT, The thickness of graphs: A survey, *Graphs Combin.*, **14** (1) 59-73, 1998.

[MPSS99] S. MUTHUKRISHNAN, M. PATERSON, S. C. SAHINALP, T. SUEL , Compact Grid Layouts of Multi-level Networks, *31st ACM*

Symposium on Theory of Computing, Atlanta, GA, 455-463, 1999.

[MTB00] F. MO, A. TABBARA, R. K. BRAYTON, A force-directed macro-cell placer, *IEEE/ACM International Conference on CAD, ICCAD 00, Santa Clara, November 2000.*

[Ree97] B. REED, Tree width and tangles: A new connectivity measure and some applications, in: R. A. Bailey (Ed.), *Surveys in Combinatorics, 1997,* Cambridge University Press, 1997.

[RNN02] S. RAHMAN, S. NAKANO, T. NISHIZEKI, Rectangular drawings of plane graphs without designated corners, *Computational Geometry,* **21** 121-138, 2002.

[SBG99] M. STALLMANN, F. BRGLEZ, D. GHOSH, Evaluating iterative improvement heuristics for bigraph crossing minimization. *In IEEE 1999 International Symposium on Circuits and Systems -- ISCAS'99,* May 1999.

[Tho83] C. D. THOMPSON, Area-time complexity for VLSI, *Proc. of the 11th ACM Symposium on Theory of Computing,* 81-88, 1983.

[TTV00] R. TAMASSIA, I. G. TOLLIS, J. S. VITTER, A Parallel algorithm for planar orthogonal grid drawings, *Parallel Processing Letters,* **10** (1), 141-150, 2000.

[Woo] D. WOOD, Geometric thickness in a grid, *to appear.*

[Woo03] D. WOOD, Optimal three-dimensional graph drawing in the general position model, *Theoretical Computer Science,* 299, 151-179, 2003.