

# Linear Decomposition Algorithm for VLSI Design Applications \*

Jianmin Li, John Lillis, and Chung-Kuan Cheng  
Dept. of Computer Sci. & Engr.  
University of California, San Diego  
La Jolla, CA 92093-0114

## Abstract

*We propose a unified solution to both linear placement and partitioning. Our approach combines the well-known eigenvector optimization method with the recursive max-flow min-cut method. A linearized eigenvector method is proposed to improve the linear placement. A hypergraph maxflow algorithm is then adopted to efficiently find the max-flow min-cut. In our unified approach, the max-flow min-cut provides an optimal ordered partition subject to the given seeds and the eigenvector placement provides heuristic information for seed selection. Experimental results on MCNC benchmarks show that our approach is superior to other methods for both linear placement and partitioning problems. On average, our approach yields an improvement of 45.1% over eigenvector approach in terms of total wire length, and yields an improvement of 26.9% over PARABOLI[6] in terms of cut size.*

## 1 Introduction

The linear placement problem has several applications in the VLSI designs. The linear placement problem is a special case of the 2D placement problem, which is considered crucial for circuit layout quality; In [5][6][11][12], Techniques for the linear placement problem have been applied for circuit partitioning, and it has been demonstrated that placement-based partitioning approaches obtain very good results. Thus, the trend leads us to conjecture that there is strong relation between the linear placement problem and the partitioning problem.

In [1], Adolphson and Hu have derived the relationship between linear placement and partitioning. By applying max-flow min-cut method of network flow, an ordered partition which is consistent with an optimal placement can be defined. However, if the cut operations cannot create new cut configuration, no additional partitioning information is obtained to further refine the linear placement, and thus heuristic algorithm is needed to further the operation[3].

Yang and Wong[2] proposed repeated max-flow min-cut method to search for the balanced partitioning. When the cut operation does not create new cut

configuration, a node incident to the source/sink will be collapsed to the source/sink, then max-flow min-cut is applied again. The new min-cut defines an optimal ordered partition with respect to new seed configuration.

However, with their approach, selection of the node to be collapsed is very crucial to the final partitioning quality. This can be observed in their experiment results. In [2], the experimental data provided the best partitioning result and the average cutsize and running time of 10 runs for benchmark circuits. We observe that the average cutsize and running time are much worse than the best ones. For example, the average cutsize and running time of 10 runs on S35932 are 213.3 and 1115.9 seconds respectively, while the best cutsize and its corresponding running time are only 49 and 280.8 seconds respectively. The results are very sensitive to the selection of initial network seeds and the nodes to be collapsed in subsequent cut operations.

Compared to Yang and Wong's FBB algorithm, our approach has the following features:

- We use eigenvector placement to give an initial ordering, we can fix the left extreme and right extreme nodes, for example  $10\%|V|$  on each end, to form  $s$  and  $t$  to speed up the computation; in contrast, Wong's approach selects seeds randomly and starts with single  $s$  and  $t$  nodes.
- The selection of node to be collapsed is based on eigenvector placement instead of random selection, implying the result is more accurate and stable.
- We employ equivalent min-cut search mechanism to speed up the computation, where the equivalent mincut search is done in  $O(m)$  complexity by breadth-first search.
- The max-flow min-cuts are implemented directly on hypergraphs instead of transforming hypergraphs into graphs.

In this paper, we propose a unified approach to linear placement and partitioning problems. We propose an algorithm which linearizes the quadratic objective function by iteratively revising the edge weights based on the current placement. We adopt a max-flow min-cut algorithm for hypergraphs, so that we need not transform hypergraphs into graphs for the max-flow min-cut. We efficiently combine the flow network method with the eigenvector approach, where

\*This work was supported in part by grant from NSF MIP-9315794 and MICRO program.

the eigenvector placement provides a guide for the seed selection during the recursive max-flow min-cut operations. Experiments on MCNC benchmarks show that our approach yields an improvement of up to 55% for linear placement (compared to the eigenvector approach) and an improvement of up to 51% for balanced partitioning (compared to PARABOLI approach).

The rest of this paper is organized as follows. The next section provides some preliminaries. In section 3, we briefly review the eigenvector approach, and propose an iterative approach to linearize the quadratic objective function. In section 4, we present an efficient preflow implementation for the hypergraph max-flow min-cut problem. Section 5 presents our unified approach to linear placement and partitioning problems, which combines recursive max-flow min-cut approach with the eigenvector placement approach. In section 6, we provide the experimental results for both linear placement and balanced partitioning. Finally we conclude this paper in section 7.

## 2 Preliminaries

A circuit is modeled by a hypergraph  $G_H = (V_H, E_H)$ , where vertex set  $V_H$  represents the modules and hyperedge set  $E_H$  represents the nets connecting vertices. The linear placement problem of a circuit is then to put these vertices in equally spaced slots, one vertex to one slot, such that the total wire length over all nets is minimal. The wire length of a net is defined by the distance between the extreme left vertex and the extreme right vertex of the net. The two-way partitioning problem of a circuit is a bipartition of  $V_H$  into  $\Theta$  and  $\bar{\Theta}$  such that the connection demand  $C(\Theta, \bar{\Theta})$  between  $\Theta$  and  $\bar{\Theta}$  is minimal.

In general, a hypergraph is approximated by a graph  $G = (V, E)$ , where a hyperedge is represented by a set of edges. For the hyperedge approximation, clique and star are the two most common models used for VLSI design problems. In this work we adopt the star model; i.e. for each net  $e$  we introduce a dummy vertex  $v_e$  and for each vertex  $v \in e$  we introduce edge  $(v, v_e)$  with weight  $1/(|e| - 1)$ .

Let  $n = |V|$  be the number of vertices and  $m = |E|$  be the number of edges in  $G$ . Then, this graph  $G$  may be described by an  $n \times n$  adjacency matrix  $A = [a_{ij}]$ , where the matrix element  $a_{ij}$  is the weight of the connection between vertices  $i$  and  $j$ . After transforming a hypergraph into a graph, we can formulate the objective functions of linear placement and partitioning as below:

Linear Placement View:

$$\text{minimize } \sum_{i>j} \sum_j a_{ij} |x_i - x_j| \quad (1)$$

Partitioning  $(\Theta, \bar{\Theta})$  View:

$$\text{minimize } \sum_{i \in \Theta} \sum_{j \in \bar{\Theta}} a_{ij}$$

where  $x_i$  is the coordinate of vertex  $i$  in the linear placement.

## 3 Optimal Continuous Linear Placement

In this section we propose an iterative linearization procedure to improve the quality of linear ordering using the spectral method. We first summarize the main concepts of spectral placement, and then propose a linearization method, and a heuristic variant.

**Background:** Given a weighted graph  $G = (V, E)$ , let  $a_{ij}$  be the weight of the connection between vertices  $i$  and  $j$ . The Laplacian of  $G$  is defined as the matrix  $B$  where

$$B = \begin{cases} \sum_{j=1}^{j=n} a_{ij} & \text{if } i = j \\ -a_{ij} & \text{otherwise.} \end{cases} \quad (2)$$

By computing the second smallest eigenvalue and its corresponding eigenvector  $X$ , we obtain a non-trivial solution to the following *continuous* placement problem.

$$\text{min } \sum_{(i,j) \in E} a_{ij} (x_i - x_j)^2 = X^T B X \quad (3)$$

$$\text{s.t. } I^T X = 0, X^T X = \alpha$$

Where  $X$  is the vector of  $x_i$ , and  $\alpha$  is the squared deviation of the linear placement.

**Linearization:** One objection to the spectral approach is that it minimizes the squared wire length (Equation 3) rather than the linear wire length (Equation 1) of the placement. To remedy this situation, we propose an algorithm which iteratively applies an eigenvalue solver, revising the edge weights based on the placement of the previous iteration. Our approach is similar to the method used by Sigl et. al. [10] for analytical placement. However, their focus was not the spectral method, but rather a force-balanced solution.

The following equation specifies the weight revising method. We introduce  $\epsilon$  for numerical stability.

$$a'_{ij} = \begin{cases} \frac{a_{ij}}{|x_i - x_j|} & \text{if } |x_i - x_j| > \epsilon \\ \frac{a_{ij}}{\epsilon} & \text{otherwise} \end{cases} \quad (4)$$

**Algorithm 1** *Linearized Eigenvector Approach*

1. Construct the matrix  $B$ , the Laplacian of  $G$ ;
  2. Find  $X_1$  the solution of  $\text{min } X^T B X$  s.t.  $I^T X = 0, X^T X = \alpha$
  3.  $k=1$ ;
  4. Do { Construct a new matrix  $B_k = (a'_{ij})$  using the placement  $X_k$  and Equation 4; Find  $X_{k+1}$  the solution of  $\text{min } X^T B_k X$  s.t.  $I^T X = 0, X^T X = \alpha$   $k = k + 1$ ;
- } Repeat until the placement  $X_k$  converges.

**Theorem 1** *Convergence of Linearized Eigenvector Approach: The wire length (Equation 1) in Algorithm 1 is monotonically decreasing.*

**Heuristic Discrete Placement:** The solution to the continuous placement problem provides a heuristic solution to a discrete placement problem. A discrete linear placement can be obtained by ordering the vertices according to the continuous placement  $X$ . Experimentally, we have found that a modified revising strategy

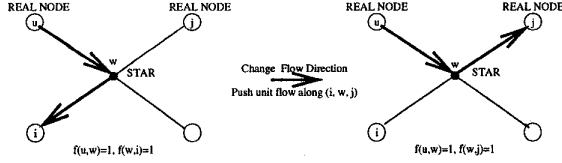


Figure 1: Push Flow to Change the Direction

often yields better results than the above method. We order the vertices according to the value of previous placement  $X_k$ . Let  $\pi(i)$  be the position of vertex  $i$  in the permutation. Our new strategy uses the function  $|\pi(i) - \pi(j)|$  as the divisor of  $a_{ij}$ .

#### 4 Max-Flow Min-Cut of a Hypergraph

The hypergraph max-flow min-cut algorithm is derived from the preflow algorithm[9]. The preflow algorithm has a feature that it first finds a min-cut in the first stage and then a maximum flow in the second stage, while the augmenting path algorithms first find a maximum flow and then a min-cut. Therefore, the preflow algorithm seems more efficient and direct if only min-cuts are needed.

Before we provide the algorithm, we first give some definitions and notations.

With a star model, a hypergraph is represented as a graph  $G = (V_r \cup V_s, E)$  with two types of vertices, which are called as nodes and stars. The node set  $V_r$  represents modules, and the star set  $V_s$  represents nets, and each edge  $e \in E$  connects between a node and star.

Let  $s, t \in V_r$  be the two distinguished source and sink nodes in the flow network. To each star  $i \in V_s$ , we assign a weight  $c(i)$  called the net capacity. In this paper, we set  $c(i) = 1$  for each star  $i$ .

For each edge  $e(i, j) \in E$ ,  $f(i, j)$  represents a directed flow from  $i$  to  $j$ . To each star  $i \in V_s$  we assign a weight  $h(i)$  called the flow through star  $i$ , defined by the total flow into this star, i.e.,  $h(i) = \sum_j f(j, i)$ .

Let  $g(i)$  be the flow excess defined for each node  $i \in V_r$  as  $g(i) = \sum_j f(j, i) - \sum_j f(i, j)$

Nodes  $i, j \in V_r$  are said to be neighboring in  $G$  if  $i$  and  $j$  are connected to a same star  $w$ , and we define the path  $(i, w, j)$  as a superedge. The residual capacity is the non-negative function on superedge  $(i, w, j)$  defined by:

$$R_p(i, w, j) = c(w) - h(w) + \max(f(w, i), f(j, w))$$

where the last term is the residual capacity contribution by changing flow direction. For example, as shown in Figure 1, even though  $c(w) = h(w)$ , because  $f(w, i) = 1$  and  $f(j, w) = 0$ , we still have  $R_p(i, w, j) = 1$ ; thus we can push one unit of flow along  $(i, w, j)$ , resulting in changed direction of flow. If  $R_p(i, w, j) > 0$ , then the superedge  $(i, w, j)$  is said to be a residual superedge. If all superedges of a path are residual, then this path is said to be a residual path. A node  $i \in V_r$  is said to be reachable from  $t$  if there exists a residual path from  $i$  to  $t$ .

A node  $i \in V_r$  is said to be active if its flow excess  $g(i) > 0$  and it is reachable from the sink  $t$ .

A valid labeling  $d$  for a preflow  $p$  is a non-negative integer function on nodes, such that

$$\begin{cases} d(t) = 0, \\ d(i) > 0 \\ d(j) \geq d(i) - 1 \end{cases} \text{ for all } i \neq t, \text{ and} \\ \text{if } R_p(i, w, j) > 0.$$

With these definitions and notations, now we describe the max-flow min-cut algorithm for a hypergraph as in Algorithm 2:

#### Algorithm 2 max-flow min-cut of a hypergraph

1. Initialize the hypergraph with zero flow,  $g(s) = \infty$ , and initialize the labeling  $d(i)$  with the shortest distance from  $i$  to  $t$ .
2. Push and relabel procedure:
  - While there exists active nodes{
    - Select an active node  $i \in V_r$
    - select a neighbor  $j \in V_r$  with  $d(j) = d(i) - 1$ ;
    - Push  $\delta = \min(g(i), R_p(i, w, j))$  flow from  $i$  to  $j$  if any flow arrives at  $t$
    - then relabel node  $i \in V_r$  with  $0 < d(i) < n$  by  $d(i) = \min(d(j) + 1 | R_p(i, w, j) > 0)$
3. Find a cut  $(\Theta, \bar{\Theta})$ :
  - $\bar{\Theta} = \{i | i \text{ is reachable from } t\}$ ;  $\Theta = \text{the rest}$ ;
4. Find a maximum flow:
  - Push flow excess back to the source.

**Lemma 1** The cut  $(\Theta, \bar{\Theta})$  derived by Algorithm 2 is a minimum cut separating node  $s$  and  $t$ .

**Lemma 2** Let  $(\Theta, \bar{\Theta})$  be the min-cut found by Algorithm 2, and  $(\Phi, \bar{\Phi})$  be any other equivalent mincut solution; then we have  $\bar{\Theta} \subset \bar{\Phi}$ .

This Lemma implies that we can use breadth-first search, which has complexity of  $O(m)$ , to efficiently find more equivalent min-cuts, as shown in Algorithm 3.

#### 5 Decomposition Algorithm for the Unified Solution

The well-known max-flow min-cut method finds a cut that separates vertices  $s$  and  $t$  with a minimum capacity; at the same time, this min-cut also defines an optimal ordered partition subject to the seeds  $s$  and  $t$ . Here we have following formal statement[1][3].

**Theorem 2** Let  $s$  be the node at the left end and  $t$  be the node at the right end in the optimal linear placement problem, A max-flow min-cut defines an optimal partition that is consistent with an optimal node order in the linear placement problem.

This theorem provides the theoretical foundation of our decomposition algorithm. With the network flow approach, the selection of seeds is very important, because all optimal ordered partitions are subject to these seeds. Therefore, instead of randomly choosing seeds for the flow network, we use the eigenvector linear placement to assign the extreme left/right node to the seeds  $s/t$ . Then a maxflow from  $s$  to  $t$  provides a partition in the optimal linear placement problem.

According to the above theorem, the subset of nodes at the left or right side of the min-cut can be condensed into  $s$  or  $t$  to form new seeds. To simplify the notation, we still use  $s$  and  $t$  to represent the subsets of nodes which form the new condensed seeds. We do a maxflow again to find a new optimal partition.

Since the new cut configuration  $(s, V - s)$  or  $(V - t, t)$  is a min-cut after condensing nodes into  $s$  or  $t$ , the subsequent maximum flow operation does not create new cut configuration and no additional partition information is obtained. In order to repeat max-flow min-cut to create a new cut configuration, a floating node  $i \notin s \cup t$  must be selected to collapse into  $s$  or  $t$ .

In Wong's approach[2], the node is randomly selected among the nodes incident to  $s$  or  $t$ , resulting in unstable performance. To remedy this problem, we combine the max-flow min-cut method with the eigenvector approach under one objective, i.e, the optimal ordering of nodes. In our approach, when max-flow min-cut can not create a new cut configuration, we use the eigenvector placement to select a node, which is incident and the nearest to the seeds, to collapse into the seeds.

Suppose we obtain three subsets  $(V_L, V_W, V_R)$  after recursive calls of min-cut operations. We choose a node  $\varpi \in V_W$  to collapse into  $V_L$  for further partition. The following theorem describes the error bound on the optimal solution.

**Theorem 3** After recursive calls of Algorithm 2, we have the subsets  $V_L, V_W,$  and  $V_R$ . Let  $O = (N_L, N_W, N_R)$  be optimal ordering in the linear placement problem, and  $\varpi$  be a node in  $V_W$ . If we place node  $\varpi$  next to  $V_L$ , and we find the new optimal ordering  $O' = (N_L, \varpi, N_W', N_R)$ , then the total wire lengths of the new ordering  $O'$  is increased by no more than  $\Delta = (|V_W| - 1) \times \max\{(C_R + C_W - C_L), 0\}$ ; where  $|V_W|$  is the size of the subset  $V_W$ , and  $C_u | u \in \{L, W, R\}$  represents the sum of net capacities between node  $\varpi$  and subset  $u$ .

Theorem 3 gives an error bound if a floating node  $i$  is selected to collapse into seeds to continue the max-flow min-cut operations for the linear placement problem.

Below is the algorithm for finding the min-cuts with given seeds and eigenvector ordering of nodes.

Input: hypergraph  $G = (V_r \cup V_s, E)$ ,  
the condensed seeds  $s$  and the sink  $t$ , and  
the eigenvector ordering of nodes  $O$ .

**Algorithm 3** Find min-cuts

1.  $W = V_r - (s \cup t)$ , let  $s' = s, t' = t$  be two working seeds
2. Set  $doflow = 0$ ; Call Algorithm 2 to find a min-cut  $(s \cup W_1, W_r \cup t)$ ;
3. if  $(W_1 = \emptyset)$   
then based on  $O$ , select a node  $v \in W$ ,  
Collapse  $v$  into the source:  $s' \leftarrow s \cup \{v\}$ ;  
set  $doflow = 1$ ;
- if  $(W_r = \emptyset)$   
then based on  $O$ , select a node  $v \in W$ ,  
Collapse  $v$  into the sink:  $t' \leftarrow t \cup \{v\}$ ;

set  $doflow = 1$ ;

if  $(doflow = 1)$

Call Algorithm 2 with new seeds to find a min-cut  $(s \cup W_1, W_r \cup t)$

4. /\* Looking for more equivalent mincuts \*/  
Do { Collapse  $W_r$  to  $t'$ :  $t' \leftarrow t' \cup W_r, W_r \leftarrow W_1$   
based on  $O$ , select a node to collapse to  $t'$   
if  $(s'$  is not reachable from  $t')$   
then we have a new cut  $(s \cup W_1, W_r \cup t')$   
} until  $s'$  is reachable from  $t'$

Output: the ordered partition  $P(s, W_1, W_2, \dots, t)$

**Lemma 3** The cuts found with Step 4 in Algorithm 3 are equivalent mincuts.

The decomposition algorithm for linear placement recursively applies max-flow min-cut to partition sets of nodes until their sizes are small enough. In implementation, residual flow in the network can be used to speed up the max-flow min-cut computation. The algorithm is provided below:

**Algorithm 4** Linear Placement

1. Eigenvector approach to provide the initial linear placement
2. Let  $s/t$  be the subset which includes the extreme left/right node of the initial placement; Let  $W = V - (s \cup t)$ , and we have initial ordered partition  $P(s, W, t)$ .
3. For each subset  $W$  in  $P$  {  
Call Algorithm 3 to partitioning  $W$   
into  $W_1, W_2, \dots$ ;  
Replace  $W$  in  $P$  with  $W_1, W_2, \dots$   
} Until all subsets in  $P$  are small enough

The solution to linear placement can be used as a heuristic guide for multi-way partition. For balanced two-way partition, we only care about cuts satisfying balancing constraint. For example, the balancing constraint may be from  $\frac{1}{2}(1 - \beta)|V|$  to  $\frac{1}{2}(1 + \beta)|V|$ . In this case, the above algorithm can be modified as below:

**Algorithm 5** Balanced two-way Partitioning

1. Eigenvector approach to provide the initial linear placement
2. Let  $s/t$  be the subset which includes the extreme left/right node of the initial placement; Let  $W = V - s \cup t$ , and we have initial ordered partition  $P(s, W, t)$ .
3. For subset  $W$  spanning the cutting range in  $P$  {  
Call Algorithm 3 to partitioning  $W$   
into  $W_1, W_2, \dots$ ;  
Replace  $W$  in  $P$  with  $W_1, W_2, \dots$   
} Until a cut satisfying the constraint is found.

To this decomposition algorithm for the balanced two-way partitioning, we have the following formal statements:

**Lemma 4** The max-flow min-cut found in step 3 of Algorithm 5 is monotonically increasing.

**Theorem 4** If the cutsize of the balanced partitioning is  $K$ , then Algorithm 5 terminates after at most  $K$  residual flow operations.

The above Theorem shows that Algorithm 5 is very efficient.

## 6 Experiments

We implemented our decomposition algorithm, which we call PANZA, for linear placement as well as balanced two-way partitioning, and tested it with a set of large benchmarks from MCNC. These benchmark circuits were also used by B.M. Riess in PARABOLI[6] and Yang in FBB[2] for the balanced two-way partitioning problem.

For the linear placement problem, we compare PANZA to the eigenvector linear placement(EIG1) and the eigenvector linear placement with linearized objective function(EIG2) in terms of the total wire length. For these eigenvector approaches, we use the star models. The net length is measured by the span of its node set.

The placement results are summarized in Table 1. The results of PANZA are superior to the EIG1 and EIG2 approaches in terms of the total wire length. On the average, PANZA yields improvements of 45.1% and 28.4% compared to EIG1 and EIG2 respectively.

For the balanced two-way partitioning problem, we compare our approach PANZA to PARABOLI[6] and FBB[2] in terms of cutsizes. For all these three methods, we allow  $\beta = 10\%$  deviation from bisection. The results are summarized in Table 2<sup>1</sup>, where the results of PARABOLI are from [6] and those of FBB are from [2]. On the average, PANZA yields improvements of 14.3% and 26.9% over FBB method and PARABOLI method respectively.

It is also important to note that the cut capacity of PANZA is consistently less than those achieved by EIG1 and EIG2 approaches. For example, Figure 2 gives the cut diagrams of different approaches for the circuits S13207 and S35932. The area under the cut diagram corresponds to the total wire length. This observation leads us to conjecture that the linear placement problem is strongly related to the partitioning problem. A high quality placement solution also derives high quality partitioning solution.

Note that our circuit characteristics of node count and net count are consistent with those of PARABOLI, but are different from those of FBB[2]. One major difference is on circuit S1423. FBB reported that S1423 circuit has 731 nodes and 743 nets, while PARABOLI and our PANZA report that it has only 619 nodes and 538 nets. Even though the underlying netlist structure in different netlist translations are the same, the number of nodes of S1423 in FBB is about 20% more than that in PARABOLI or PANZA. Because of this discrepancy, the comparison may not be fair in this case.

## 7 Conclusion

In this paper, We developed an efficient decomposition algorithm for a unified solution to the linear placement problem as well as partitioning problem. A linearized eigenvector approach is derived to improve

<sup>1</sup>The circuits marked with \* under CUT SIZE in Table 2 are not included in FBB[2]

the linear placement. The recursive max-flow min-cut approach is a very efficient method[2], but it is also very sensitive to the seed selection during subsequent cut operations. Therefore, it is not surprising that our approach, which combines the recursive max-flow min-cut method with the eigenvector placement approach, produces superior linear placement and partitioning results. Our placement-based two-way partitioning results are at least 14% better than the state-of-the-art.

## References

- [1] D.Adolphson, and T.C. Hu, "Optimal Linear Ordering," SIAM J. Appl. Math. Vol.25(3), Nov. 1973, pp403-23.
- [2] H. Yang, and D.F. Wong, "Efficient Network Flow Based Min-Cut Balanced Partitioning," Proc. IEEE Int. Conf. on Computer-Aided Design, 1994, pp50-5.
- [3] C.K. Cheng, "Linear Placement Algorithms and Applications to VLSI Design," NETWORKS, Vol.17, 1987, pp439-64.
- [4] C.K. Cheng, and E.S. Kuh, "Module Placement Based on Resistive Network Optimization," IEEE Trans. on Computer-Aided Design, Vol.CAD-3(3), Jul. 1984, pp218-25.
- [5] L. Hagen, and A.B. Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering," Proc. IEEE Int. Conf. on Computer-Aided Design, 1991, pp10-3.
- [6] B.M. Riess, K. Doll, and F.M. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques," Proc. 31th ACM/IEEE Design Automation Conference, 1994, pp646-51.
- [7] E. Ihler, D. Wagner, and F. Wager, "Modeling Hypergraphs by Graphs with the same mincut properties," Information Processing Letters, Vol.45(4), Mar. 1993, pp171-5.
- [8] T.C. Hu, and K. Moerder, "Multiterminal Flows in a Hypergraph," Hu and Kuh eds., IEEE Press, 1985.
- [9] A.V. Goldberg, and R.E. Tarjan, "A New Approach to the Maximum Flow Problem," Journal of the Association for Computing Machinery, Vol.35(4), Oct. 1988, pp921-40.
- [10] J.M. Kleinhans, G. Sigl, F.M. Johannes, and K.J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," IEEE Trans. on Computer-Aided Design, Vol.10(3), Mar. 1991, pp356-365.
- [11] K.M. Hall, "An r-Dimensional Quadratic Placement Algorithm, Management Science," Vol.17(3), Nov. 1970, pp219-29.
- [12] R.S. Tsay, and E.S. Kuh, "A unified approach to partitioning and placement (VLSI layout)," IEEE Trans. on Circuits and Systems, Vol.38(5), May 1991, pp521-33.
- [13] G. Sigl, K. Doll, and F.M. Johannes, "Analytical Placement: A Linear or a Quadratic Objective Function?" Proc. 28th ACM/IEEE Design Automation Conference, 1991, pp427-32.

CIRCUITS	WIRE LENGTH			%improv over	
	EIG1	EIG2	PANZA	EIG1	EIG2
s1423	12,416	9,924	9,060	27.0	8.7
s9234	463,210	290,472	266,617	42.4	8.2
s13207	790,430	608,820	451,210	42.9	25.9
s15850	1,377,295	1,072,068	728,103	47.1	32.1
s35932	2,052,699	1,693,484	920,607	55.2	45.6
s38417	3,115,066	2,386,239	1,711,139	45.1	28.3
s38584	3,915,095	3,480,634	1,728,237	55.9	50.3
Average				45.1	28.4

Table 1: Linear Placement Results.

CIRCUITS	BALANCED CUT SIZE			%improv over	
	FBB	PARABOLI	PANZA	FBB	PARABOLI
s1423	13	16	15	-15.4	6.3
s9234	70	74	40	42.8	45.9
s13207	74	91	66	10.8	27.5
s15850	67	91	44	34.3	51.6
s35932	49	62	43	12.2	30.6
s38417	58	49	49	15.5	0.0
s38584	47	55	47	0.0	14.5
s1488	*	50	44	*	12.0
sioo	*	45	25	*	44.4
struct	*	40	33	*	17.5
balu	*	41	27	*	34.1
biomed	*	135	83	*	38.5
Average				14.3	26.9

Table 2: Partitioning Results Allowing  $\beta = 10\%$  deviation from bisection.

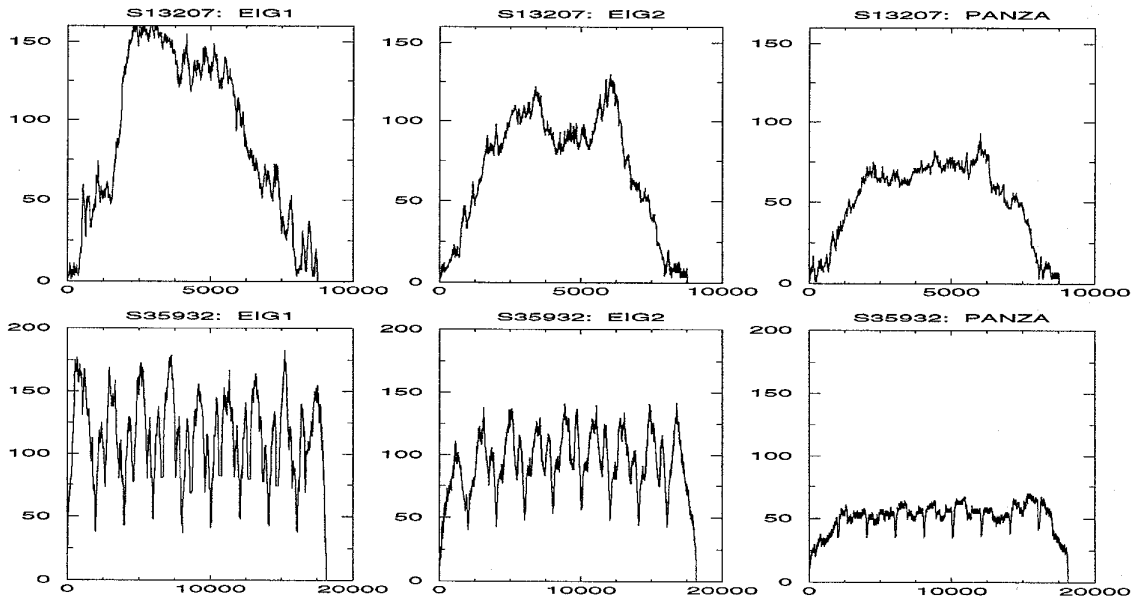


Figure 2: Histogram of linear placement: cut capacity vs. cut position for circuits S13207 and S35932