Proceedings IEEE Intern. Conf. on Computer Design: VLSI in Computers, 1983

RC 10069 (#44763) 7/12/83
Computer Science 9 pages

Research Report

HEURISTIC IMPROVEMENT TECHNIQUE FOR BISECTION OF VLSI NETWORKS

Mark K. GOLDBERG

Clarkson College of Technology
Potsdam, New York

Michael BURSTEIN

IBM T. J. Watson Research Center
Yorktown Heights, New York

IBM Research Division
San Jose · Yorktown · Zurich

Copies may be requested from:

IBM Thomas J. Watson Research Center
Distribution Services 38-066
Post Office Box 218
Yorktown Heights, New York 10598

RC 10069 (#44763) 7/12/83
Computer Science 9 pages

HEURISTIC IMPROVEMENT TECHNIQUE FOR BISECTION OF VLSI NETWORKS

Mark K. GOLDBERG

Clarkson College of Technology
Potsdam, New York

Michael BURSTEIN

IBM T. J. Watson Research Center
Yorktown Heights, New York

A B S T R A C T

We address the mincut partitioning problem for VLSI networks and experimentally evaluate the performance of several heuristic algorithms. For example, we observed the the quality of final partition obtained by Kernighan-Lin algorithm for graphs is heavily dependent on the ratio of the number of edges to the number of vertices: namely, the algorithm performs poorly when this ratio is less then 3 and produces nearly optimal solutions when the ratio is higher 5. We also develop a hierarchical improvement technique that can be applied to any mincut algorithm and results in an improved algorithm. We use the Kernighan-Lin-Fiduccia-Mattheyses (KLFM) technique as a base for the improvement. The resulting algorithm outperforms the KLFM precisely in the "poor performance areas" for the latter.

INTRODUCTION

The problem of partition of the set of nodes of a network into subsets arises at several stages of VLSI design and layout. The objective of such partition is usually to reduce the number of external interconnections. The importance of the problem is well recognized. We consider here a particular instance of the problem where the nodes of a network have to be distributed among two subsets in such a way that the number of dissected nets is minimal. We address it as a bisection problem in order to distinguish it from general partition. Bisection is a key feature of the several automatic gate array layout systems ([3,7,8]). The problem can be formalized in the following way. Given a network $H = (V, E)$, where V is a set of ***nodes*** and E is a set of subsets of V (called ***nets***) consisting of at least two nodes. The problem is to partition the set V into two parts $V = A \cup B$ of equal size ($|A| = |B|$) such that the number of nets having nonempty intersection with both A and B is minimal. For the sake of simplicity we assume that the total number of nodes is always even. In practice it may not be so: the nodes may be of different weight, they may not be interchangeable or free to be assigned to any subset, the desired partition may not be one that dissects the set into exactly equal parts. But the problem, as it is formulated above, is an important extraction of the practical problem and preserves the key difficulties pertinent to general partitioning.

The first nontrivial heuristic algorithm for graph bisection problem was developed by Kernighan and Lin [1]. This algorithm, when implemented straightforward results in a $O(n^{2.5})$ execution time routine. Recently Fiduccia and Mattheyses developed a modification of Kernighan and Lin method [2] but utilization of efficient data structures enabled them to come up with linear time heuristic practically without any loss in quality of partitions and applicable to general networks. This bisection technique, which further will be referred to as KLFM, is superior to several local optimization algorithms [4]. However there was still no significant performance study of this technique. As a result of numerous experiments we concluded that the KLFM performance is heavily dependent on certain network parameter, which we call ***network ratio*** and define as follows: for a network $H = (V,E)$, the ratio

$$r(H) = \frac{\sum_{e \in E} |e| - 1}{|V|}.$$

It turns out that KLFM performs relatively poor when $r(H)<3$ but performs nearly optimally when $r(H)>5$. Practical networks, however lie in the range of $1.8<r(H)<2.5$.

We also develop a mechanism that can be applied to any bisection algorithm and results in another bisection algorithm which we call a ***derivative.*** Our main result is the experimentally derived conclusion: the derivative of KLFM significantly outperforms KLFM in the $r(H)<3$ area.

TEST MODEL

Since an analytical technique for evaluating performance of bisection algorithms does not exist, experimental methods must be employed. This is explicitly pointed out in [6] ; moreover [6] suggests that the bisection algorithms should be evaluated on specific networks for which the mincut value is known, and describes the construction of such networks. However, we argue that the suggested networks are still not very suitable for algorithm testing, since the ratio of the generated networks is very high. In order to demonstrate this denote A and B two sets with p elements, $n = 2p$ and let k be an integer. Let U_i be a random $(p + 1)$ element subset of $A \cup B$, $(i = 1, 2, ..., k)$. Let T_i be an arbitrary tree connecting the vertices of U_i such that only one edge of it connects a node from A with a node from B. Let $KM(n,k)$ (for Krishnamurthy & Mellema) denote a graph which is the union of these $T_i - s$ on the node set $A \cup B$. It is easy to see that the mincut number of KM is exactly k. However, $r(KM) = k/2$, which easily puts us into the "good performance area" of KLFM, as it will be demonstrated in the next section (since practical values of k are larger then 10). The networks generated in [6] in our optinion are not good test cases for bisection algorithms. They practically will never occur in real logical networks. It has been observed that in practical networks the total number of nets is approximately equal to the number of active nodes. Moreover, the number of 2-terminal nets is about 45% of the total number of nets, the number of 3-terminal nets is about 15% and so on. In general [7] the following "net-size distribution" criteria is approximately satisfied. Denote m_i the number of nets connecting i nodes of the network, $M(H)$ - the vector, whose i -th component is m_i/m , where $m = \Sigma m_i$. Then

$$M(H) = 0\ 0.45\ 0.15\ 0.12\ 0.11\ 0.8\ 0.6\ 0.3\ .$$

We suggest that the test networks be generated according to this criteria. It was pointed out in [4] that the parameters m_i play an important role in bisectioning. The average number of dissected nets over all possible bisections (an upper bound for mincut) can be expressed in terms of these parameters.

In order to evaluate the bisection algorithm performance we need not know the exact value of the mincut. It is enough to know a good upper bound. So, we suggest the following construction. Denote $G(n,m)$ a random network on n nodes and m nets satisfying the above distribution formula. Let $B(n,m,k)$ be a network constructed as follows: take two copies of $G(n/2,(m - k)/2)$ (assuming that n and $(m - k)$ are even) and add k random nets connecting these two networks and also satisfying the size distribution. Then obviously k will be an upper bound for the mincut of $B(n,m,k)$. We will be testing the bisection algorithms in these $B(n,m,k)$ -s. The quality of the algorithm will be evaluted by the difference between the resulting number of dissected nets and k. The resulting number may easily be less then k if k is large enough. In order to model practical networks the values of n,m and k should be selected in such a way, that $m \sim 1.1n$ and $k \sim O(\sqrt{m})$.

KLFM

We have implemented a simplified version of Fiduccia-Mattheyses algorithm [2] under the assumptions that all nodes are equivalent and free to move between both buckets. The algorithm runs in linear time and our testing experimentally confirmed that fact. Using the same data structures we also implemented the original Kernighan-Lin algorithm, which theoretically may have on the average $O(n \log n)$ running time, but practicly runs almost as fast as Fiduccia-Mattheyses. It usually takes more time per pass, but requires slightly less passes. The test results on Krishnamurthy-Mellema graphs $KM(n,k)$ are presented in the Table 1. We have performed 6 runs for each setting of parameters n and k : n = 500 1000 1500 and k = 10 20 30. The optimal solution was found in each of these 6 experiments. Running times are given per pass in seconds for PL/1 implementation on IBM 370/3081.

Table 1.

	Kernighan-Lin		Fiduccia-Mattheyses	
	n. passes	time per pass	n. passes	time per pass
KM(500,10)	3	0.8	3	0.7
KM(500,20)	3	1.4	3	1.2
KM(500,30)	3	1.7	4	1.6
KM(1000,10)	4	0.9	5	0.8
KM(1000,20)	3	1.8	5	1.4
KM(1000,30)	3	2.1	3	1.9

We also tested the algorithms on several different $K(2000,50)$ -s and the optimal solution was obtained every time (with perhaps different numbers of passes) by both of them.

The results of testing on $B(n,m,k)$ -s are, however, more interesting. Table 2 contains the results of typical runs with different n,m,k setting. We have performed 5 runs with each parameter setting and the results are averaged over these 5 runs. The deviation from these average figures was also small. These results led us to the conclusion that the quality of KLFM performance increases with the growth of ratio. Figure 1 contains an illustration of this phenomenon, when on X axis we measure the ratio, and on Y the $\frac{c_r}{c_o}$, where c_r is the resulting number of dissected nets by KLFM and c_o the minimal number of them.

Table 2.

	Kernighan-Lin		Fiduccia-Mattheyses	
	avrg cut	total time	avrg cut	total time
B(500,500,50)	64	1.9	63	2.1
B(500,1000,150)	149	4.3	148	4
B(500,1500,200)	194	5.8	194	6.3
B(500,2000,250)	246	9.1	246	8.7
B(1000,1000,100)	123	5.6	126	5.4
B(1000,1600,200)	200	6.7	200	6.4
B(1000,3200,300)	298	10.1	297	9.3
B(1000,6400,400)	381	16.3	382	16.2

We conclude that $\frac{c_r}{c_o} \to 1$ when ratio increases.

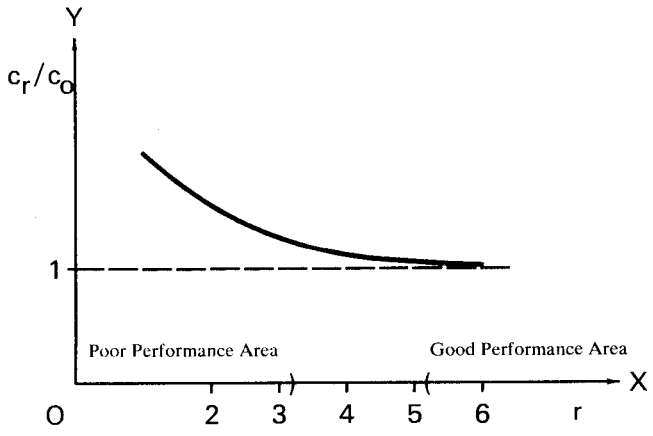

Figure 1.

DERIVATIVE

In this section we describe a mechanism for generating bisection algorithms: starting with an arbitrary algorithm we generate another one which we call a ***derivative.*** The idea of this construction was suggested in [3]. We need the following definitions. Let $H = (V,E)$ be a network. Notation: if $X \subseteq V$, then H_X denotes a subnetwork of H having X for a set of nodes and a set $\{X \cap e \mid e \in E \text{ and } \mid X \cap e \mid \geq 2\}$ for a set of nets.

A ***matching*** of the set V is the partition of it into $\mid V/2 \mid$ disjoint 2-element subsets. For a matching F, H^F denotes a factor-network of H in which F serves as a set of nodes and nets are the contracted nets of H (2-element nets that contracted completely are deleted).

Let Y be an abstract algorithm that can be applied to any network $H = (V,E)$ and the result of this application is the partition $V = A \cup B$ which we denote $Y(H)$. Let F be an arbitrary matching of V. Then the derivative of Y (denoted by dY) is defined as follows.

1. **Construct H^F ;**
2. **Apply Y to H^F ; let $A' \cup B'$ be the resulting partition of H^F ;**
3. **Generate $A \cup B$ - reconstructed partition of V ;**
4. **Construct H_A and H_B and apply Y to each to those networks; let $A = A_1 \cup A_2$ and $B = B_1 \cup B_2$ denote the resulting partitions: $Y(H_A)$ and $Y(H_B)$;**
5. **Select arbitrary matchings F_1, F_2, F_3, F_4 of each of the sets A_1, A_2, B_1, B_2 ; let $F_* = \bigcup_i F_i$ (matching of V);**
6. **Construct H^{F_*} and apply Y to it; let $V = A'' \cup B''$ be the resulting partition.**
7. **Announce $V = A'' \cup B''$ to be the resulting partition of the derivative algorithm dY .**

Remark: the definition of the derivative algorithm is dependent on the matching selection.

The derivation mechanism may be very useful in case when the initial algorithm Y is space consuming, since if the initial network has n nodes, the original algorithm Y is applied to $n/2$ node networks only. The derivative of Y is generally faster than Y in case when the computational complexity of Y is more than $O(n^2)$. If Y is a linear algorithm, then dY is linear as well (with a larger constant).

Our main purpose is to investigate the performance of the derivative of KLFM. Since KLFM is an improvement routine for a given partition, we specify two variations of it. KLFM1 starts with a random bisection and iteratively applies KLFM until no further improvement is made. dKLFM1 denotes a derivative of KLFM1. This derivative also assumes a starting bisection, so it may be iterated (applied to the result) as well. KLFM2 denotes

only one pass of KLFM and dKLFM2 is the corresponding derivative. We applied those derivatives in iterative mode until the iterations showed no improvement. The test networks were the $B(n,m,k)$ -s with net size distributions described above. We have performed 10 runs with each of the three parameter settings:

A: $n = 512$, $m = 600$, $k = 100$;

B: $n = 1024$, $m = 1200$, $k = 200$;

C: $n = 2048$, $m = 2400$, $k = 400$;

Table 3 contains the data on execution time averaged over the 10 runs (times are given in seconds both total and per pass).

Table 3.

	KLFM		dKLFM1		dKLFM2	
	ttl	p. p.	ttl	p. p.	ttl	p. p.
A	3.5	0.7	5.2	1.3	5.1	0.4
B	8.1	1.9	10	2.9	12.3	0.8
C	20.4	4.2	23	6.1	24.5	2.2

Table 4 presents the averaged results and averaged numbers of passes.

Table 4.

	KLFM		dKLFM1		dKLFM2	
	cut	n. p.	cut	n. p.	cut	n. p.
A	96	5	84	4	83	9
B	205	4	190	3	196	12
C	418	5	360	3	361	10

The deviations from these average results are not significant. In none of these experiments KLFM resulted in better bisection then both derivatives. Note, that the ***ratio*** of networks in these experiments belongs to the "poor performance area" of KLFM. We believe that the derivatives will perform similar to KLFM when the ratio is greater than 5.

We performed another five tests on $B(512, 2500, 1000)$ -s (ratio is 12.3). Table 5 contains the averaged results for these tests (average cut and number of passes).

Table 5. Tests on B(512,2500,1000).

KLFM		dKLFM1		dKLFM2	
cut	n. p.	cut	n. p.	cut	n. p.
749	5	750	4	753	12

The following explanation for the improved performance of derivatives in the *ratio*<3 area is suggested. At the derivation step 2 we apply KLFM to H^F. The ratio of H^F is approximately twice as higher than the ratio of H , because its number of nodes is twice as less, and, at the same time, the numerators are changed insignificantly. The numerator of the ratio for H^F differs from the numerator of $r(H)$ by the number pairs of adjacent nodes of H that got contracted (two nodes of a network are adjacent if they belong to the same net). Let q denote the total number of adjacent node-pairs of H. Since total number of node-pairs is $n(n-1)/2$ the expected number of contracted pairs will be

$$\frac{|F|}{\frac{n(n-1)}{2}} q = \frac{q}{n-1}.$$

It is easy to see, that in our case $q = \sum_i m_i(m_i - 1)/2 \sim 6m$, and since $m \sim n$, on the
average only 6 or 7 pairs will be contracted. This means that KLFM is applied to H^F in the "better performance area". However, H^F is a different network, and the mincut value of it may be significantly higher than the original (we might have contracted the edges of the real mincut). But we reconstruct H at step 3 and try to eleminate as many disection candidates as we can in step 4. Finally, in the step 6, we apply KLFM for H^{F*} which has almost doubled ratio as well.

ACKNOWLEDGEMENT

We are grateful to Jerome Kurtzberg, Jonathan Turner and Jacob Peysackh for helpful comments, suggestions and assistance.

REFERENCES

[1] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", Bell System Technical Journal, Vol. 49, February 1970, pp. 291-307.

[2] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", Proc. 19th Design Automation Conference, Las Vegas, June 1982, pp. 175-181.

[3] M. K. Goldberg and R. Gardner, "On the minimal cut problem", Proc. of the Silver Jubilee Conf. on Combinatorics, 1983.

[4] M. Burstein, "Analysis of a Network Partitioning Technique", Proc. ISCAS-82, Rome, May 1982, 477-480.

[5] M. A. Breuer, A. D. Friedman, A. Iosupovicz, "A Survey of the State of the Art of Design Automation", Computer, Vol. 14, No. 10, October 1981, pp. 58-75.

[6] B. Krishnamurthy and P. Mellema, "On the Evaluation of Mincut Partitioning Algorithms for VLSI Networks", Proc. ISCAS-83, Newport Beach, May 1983, pp. 12-15.

[7] M. Burstein, S. J. Hong, R. Pelavin, "Hierarchical VLSI Layout: Simultaneous Placement and Wiring of Gate Arrays", Proc. IFIP VLSI-83, Trondheim, August 1983.

[8] A. E. Dunlop, Automatic Layout of Gate Arrays, Proc. ISCAS-83, Newport Beach, May 1983, pp. 1245-1248.