Efficient Network Flow Based Ratio-Cut Netlist Hypergraph Partitioning

Sachin B. Patkar
Department of Mathematics
Indian Institute of Technology, Bombay
Mumbai 400 076, India

Himanshu Sharma
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Mumbai 400 076, India

H. Narayanan
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Mumbai 400 076, India

Abstract: -
We present a new efficient heuristic that finds good ratio-cut bipartitions of hypergraphs that model large VLSI netlists. Ratio cut measure is given by the ratio of the number of nets cut between two blocks and the product of the cardinality (size) of each block. Hypergraphs model VLSI netlists in a more natural fashion than do graphs. Our new heuristic may be considered a hybrid between the approaches of [10] and [12]. It makes use of maxflow-mincut algorithm as a subroutine that is invoked only a (small) constant number of times. Comparisons with the state-of-art partitioners such as hmetis [6](hmetis was run several times to yield min-cuts for diffently balanced bipartitions) show that the heuristic successfully finds better ratio-cut bipartitions in most of the cases. Although maxflow-mincut algorithms are theoretically slower than multilevel algorithm of hmetis the actual running times of our heuristic is found to be of the same order. The experiments have been conducted on ISPD98 benchmarks of sizes upto 200K cells.

Key-Words: - VLSI physical design, Circuit partitioning, Computer aided design tool

1 Introduction

Netlist Partitioning is of immense importance in many VLSI CAD applications [1, 8]. The objective is to divide the circuit(or a netlist) into blocks such that each block falls within prescribed sizes and the complexity of connections between these blocks is reduced.

We use $netlist(C, H)$ to denote a circuit(or a netlist) where C is the set of cells and H is the set of nets. We call $\{C_1, C_2\}$ a bipartition of the set of cells C if $C_1, C_2 \neq \emptyset$, $C_1 \cap C_2 = \emptyset$ and $C_1 \cup C_2 = C$. We say that a bipartition is *balanced* if the sizes of the blocks of the bipartition are roughly the same.

Although many algorithms are highly successful in balanced bipartitioning and multiway partitioning, one limitation most of them have is that they do not reveal the hierarchical nature of the real netlists. The presence of hierarchy gives rise to natural clusters of cells. Most of the widely used algorithms tend to ignore this clustering and divide the netlist in a balanced (or within the bounds of acceptable imbalance) partitioning and frequently, the resulting partitions are not "natural" (in the sense of the small value of ratio of the cut value of a block to its weight). Very often it is desired to have a possibly unbalanced partition that is more "natural" [8]. This "naturalness" is captured by the well-known concept of *ratio-cut* (Ratio cut measure is given by the ratio of the number of nets cut between two blocks and the product

of the cardinality (size) of each block) (see [2, 7]) which is defined for $\emptyset \neq W \subset C$ as follows:

$$\text{ratio-cut}(\{W, C-W\}) = \frac{|\delta(W)|}{w(W) * w(C-W)}.$$

where $\delta(W)$ (or equivalently, $\delta(C-W)$) is the set of nets crossing W (A net is said to cross W if atleast one but not all of its cells are in W), while $w(U)$ is the sum of the weights of cells in any subset of cells $U \subseteq C$.

The objective of this ratio-cut formulation, is to identify the natural clusters in the circuit and prevent these from being separated by the cut. In a ratio-cut formulation, the algorithm focuses on minimizing the ratio-cut, as opposed to finding minimum balanced cut. Although the denominator is maximum when both sides are of equal size, the minimum ratio cut may be obtained for a specific cut set that may divide them into two unequal size blocks.

In this paper, we describe a new efficient network flow based approach that aims at finding a partition of a circuit into 2 parts that has better ratio-cut.

The problem of finding a bipartition, say $\{V_1, V_2\}$ of the vertex set V representing cells of a netlist, that minimizes the **ratio-cut**$(\{V_1, V_2\})$ is known to be *NP-hard* (see for details [7]).

Approaches for ratio-cut partitioning has been proposed and evaluated by several researchers. The paradigms and techniques used for ratio-cut partitioning range from the local search based iterative improvement technique of [11] and genetic algorithm of [8] to spectral partitioning approach used in [2] and the submodular function optimization based idea of [10]. Our new technique *resembles* the one proposed in [10], however the similarity ends in the use of maxflow-mincut algorithm [5]. The network flow model used in this paper is different from the one used in [10] and it is based on a true hypergraph model of a netlist rather than the graph model of a netlist [10]. It may be remarked at this point that the network flow model used in this paper is an extension of (*however not identical to*) the one used by [12] in their proposal of FBB algorithm for balanced bipartitioning of netlists. Thus our new approach may be thought of as a hybrid between the approaches of [10] and [12].

The problem of finding minimum balanced bipartitioning has been even more well-researched over many years and the best of these ideas include the multi-level algorithms such as hmetis of Metis package [6], the MLPart of [3], dohpart from [4]. These algorithms are reputed to be extremely fast and able to produce high quality bipartitions. Indeed a few of these implementations, such as hmetis can also be used for producing unbalanced bipartitions (however, the user has to specify the "balance factor") so as to give better output in the sense of ratio-cut measure.

Generally, the existing pure ratio-cut algorithms have been slower and their results are not available for larger benchmark netlists. Therefore, we focussed on evaluating our new approach by comparing with the results from application of hmetis partitioner as described below.

Our experimental setup is described below. Hmetis ([6]) is among the most efficient and effective hypergraph partitioning systems. As netlists are typically modeled using hypergraphs, hmetis presents itself as an obvious choice for benchmarking purposes. We make use of the best (in terms of the ratio-cut) bipartition (by varying constraints on sizes of the 2 parts of a bipartition) obtained by hmetis as the reference for comparisons. These bipartitions were compared with those produced by equivalently many invocations of our new heuristic algorithms.

The paper is organized as follows. Section 2 describes relevant notation and definitions. In Section 3 we discuss how to model a netlist as a flow network. In Section 4 we describe our new algorithm. In Section 5 we present the results and the comparison with the hmetis results and conclude the paper in Section 6.

2 Preliminaries and Notation

Flow Network [5] is a directed graph $G(V, E)$ with specified source $s \in V$ and a specified sink $t \in V$. Furthermore, each edge (u, v) has a non-negative capacity $cap(u, v)$ where $cap(u, v) = 0$ if $(u, v) \notin E$. Flow is a real-valued function $f : V \times V \rightarrow \mathbf{R}$ which satisfies the following 3 properties :

- Capacity Constraint : $f(u, v) \leq cap(u, v)$
- Skew Symmetry : $f(u, v) = -f(v, u), \forall u, v \in G$
- Flow Conservation : $\sum_{v \in V} f(u, v) = 0, \forall u \in V - \{s, t\}$

Flow Value is the sum of flows from the source s to all the vertices $\subseteq V$ or from all the vertices $\subseteq V$ to the sink t .

$$|f| = \sum_{u \in V} f(s,u) = \sum_{u \in V} f(u,t)$$

Maximum flow problem is to find a flow f such that $|f|$ is maximum.

It is convenient to use implicit summation notation

$$f\{X,Y\} = \sum_{x \in X} \sum_{y \in Y} f(x,y)$$

where X, Y are sets of vertices, $v \in V$.

A **cut** in a flow network is a partition of V into S and $T = V - S$ s.t. $s \in S$, $t \in T$ The net flow across the cut $\{S,T\}$ is defined to be $f\{S,T\}$. An edge whose starting (ending) vertex is in S and ending (starting) vertex is in T is called a forward (backward) **cut-edge**. The capacity of the $cut\{S,T\}$ is the sum of the capacities of the forward cut-edges and is denoted by $cap\{S,T\}$. It is easy to observe that net flow $f\{S,T\} = |f|$.

Max-Flow Min-Cut Theorem states that if f is a flow in a flow network $G(V,E)$ with source s and sink t, then the following conditions are equivalent:

- f is a maximum flow in G.
- $|f| = cap\{S,T\}$ for some cut $\{S,T\}$ of G.

In other words there exists a cut $\{S,T\}$ of minimum capacity (equal to $|f|$) and $|f|$ saturates all forward cut-edges from S to T.

3 Modeling a Net in a Flow Network

Circuits are best described as netlists consisting of cells and nets. Hypergraphs are a natural representation of the circuits with cells being represented as vertices and net being described by hyperedge. Package hmetis uses hypergraph modelling of netlists for partitioning.

A net $\eta = (v; v_1, v_2, ..., v_m)$ connects the output of the cell v to the inputs of $v_1, v_2, \ldots, v_m$. For example in Figure 1, cells are given by set $\{r_1, r_2, g_1, g_2, r_3\}$ and net $\eta = (r_1; g_1, g_2)$ connects the output of cell r_1 to inputs of cells g_1 and g_2. Also for the net η, we use the notation $cells(\eta)$ to denote the set of the incident cells (r_1, g_1, g_2).

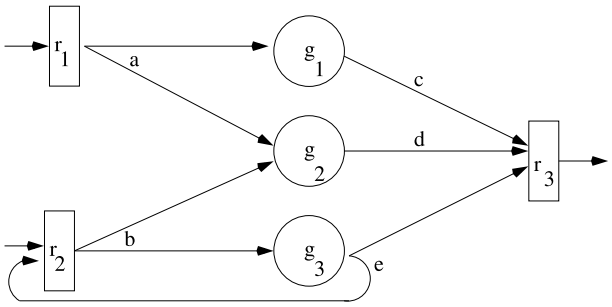

Figure 1: A netlist.

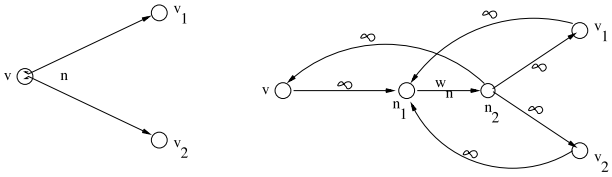

Figure 2: A net and its equivalent representation in Flow Network.

Let $\{X, X'\}$ be a bipartition of the set of cells of a netlist. $\delta(X)$ is the set of nets that cross X (or equivalently cross X'). A bipartition $\{S,T\}$ is a minimum net-cut if $|\delta(S)|(= |\delta(T)|)$ is minimum.

In order to find a bipartition with good ratio-cut, we will make use of flow-network ideas. Towards this, we will use the model described in [12] for representing nets in a flow network. Let us construct a flow network $N(V', E')$ from the given netlist as follows (see Figure 2). A net η can be represented as shown in Figure 2.

To model the net in the flow network following steps need to be taken.

- V' is initialized to represent cells of the netlist.
- For each net $\eta = (v; v_1, v_2...v_m)$ add two vertices $sink(\eta) = n_1$ and $source(\eta) = n_2$ to V' and a bridging edge (n_1, n_2) in E'.
- For each cell $u \in \{v, v_1, v_2....v_m\} = cells(\eta)$ incident on net η, add two edges (u, n_1) and (n_2, u) in E'.
- Assign appropriate capacity (specified later in this paper) to the bridging edge and infinite capacity to the rest of the edges in E'.

In addition following steps are taken to complete the flow network

- Each cell in C correspond to a unique vertex in V'
- For a vertex $v \in V'$ corresponding to a cell in the original $netlist(C, H)$ the weight $w(v) : v \in V'$ is the same as $w(u) : u \in C$, which could reflect the area requirement of the cell.

Thus all vertices incident on the net η are connected to n_1 and are connected from n_2. It is easy to see that the size of N is only a constant factor larger than the size of $netlist(C, H)$.

4 Algorithm and its Analysis

A high level description of the subroutine which forms the basis of our heuristic, is given below. We will later show that it gives partitions with better ratio-cut.

Subroutine DeltaMod_hyp(Input: $A \subset C$, Output: $\{A', C - A'\}$)

1. Given a subset $A \subset C$ (a proper subset of the set of cells C)
   - Build a flow network $N(V', E')$ for the complete $netlist(C, H)$ as described in Section 3.
   - The "bridging" edges in the flow network modeling the nets are assigned capacity equal to $w(A)$. Note that the other arcs in the flow network model of the nets have infinite capacity.
2. Next we make the following modifications to the flow network obtained above. These modifications are based on ideas used [10].
   - Introduce a new vertex, and call it $SOURCE$. For each vertex $v \in V'$ corresponding to a cell in A add edge $(SOURCE, v)$ of capacity $|\delta(A)| * w(v)$ where $\delta(A)$ is the net-cut of A and $w(v)$ is the weight of cell v.
   - Introduce a new vertex, and call it $SINK$. For each vertex $v \in V'$ corresponding to a cell **not** in A, add edge $(v, SINK)$ of capacity ∞
3. For the above model obtain the maxflow based mincut with $SOURCE$ as s and $SINK$ as t. Let $\{Y, Y'\}$ be a min-cut separating $SOURCE$ and $SINK$.
4. Output $A \cap Y$ as A'. $\{A', C - A'\}$ is our new bipartition.

End Subroutine DeltaMod_hyp

The outline of our new heuristic stratergy which uses good selection of A is as follows:

Algorithm DeltaMod_hyp_Metis
Input: $netlist(C, H)$
Output: Good ratio-cut bipartitions $\{C'_1, C - C'_1\}, \{C'_2, C - C'_2\}$
begin
Call hmetis (of Metis) or dohpart [4] on $netlist(C, H)$ to obtain a good bipartition $\{C_1, C_2\}$.
Call DeltaMod_hyp with C_1 as input and obtain the bipartition $\{C'_1, C - C'_1\}$ from its output.
Call DeltaMod_hyp with C_2 as input and obtain the bipartition $\{C'_2, C - C'_2\}$ from its output.
end
End Algorithm DeltaMod_hyp_Metis

We now justify our strategy. We will show that the bipartition of the cells $\{Y \cap A, C - (Y \cap A)\}$ output by subroutine DeltaMod_hyp has lower (or equal) ratio-cut that the bipartition $\{A, C - A\}$ that is fed to it. Thus, we would be justified in feeding the blocks of the bipartitions, obtained from hmetis invocation, to the subroutine DeltaMod_hyp, in order to get bipartitions with lower ratio-cut.

Since $cap\{\{SOURCE\}, V' - \{SOURCE\}\}$ is finite, the min-cut in the flow network N is of finite capacity. Suppose we get a $cut\{Y, Y'\}$ as a min-cut in the flow-network above. If $cells(\eta) \cap Y \neq \emptyset$, clearly $sink(\eta) \in Y$, for otherwise there would exist an infinity capacity arc from a cell of η in Y to $sink(\eta)$ which would cross across $\{Y, Y'\}$ contradicting the fact that it is min-cut in the flow-network. Similarly, if $cells(\eta) \cap Y' \neq \emptyset$ then $source(\eta) \in Y'$, for otherwise, there would exist an infinity capacity arc to a cell of η in Y' from $source(\eta)$.

If $cells(\eta) \subseteq Y$, then both $source(\eta), sink(\eta) \in Y$. For otherwise, by pulling both $source(\eta)$ and

$sink(\eta)$ to the same side as of $cells(\eta)$ one would get a lesser cut in the flow network.. Similarly, in the case, $cells(\eta) \subseteq Y'$ both $source(\eta), sink(\eta) \in Y'$.

Thus, a net η is cut by the flow-network min-cut $\{Y, Y'\}$ **if and only if** the $cells(\eta)$ are split across $\{Y, Y'\}$. Furthemore, if a net η is cut by the flow-network min-cut $\{Y, Y'\}$, then we know that the edge $(sink(\eta), source(\eta))$ is a **forward cut-edge**.

Therefore, $(sink(\eta), source(\eta))$ is a forward arc in the min-cut $\{Y, Y'\}$ of the flow-network **iff** $cells(\eta) \cap Y \neq \emptyset$ and $cells(\eta) \cap Y' \neq \emptyset$, that is, η lies in $\delta(Y \cap A)$. Furthermore, the contribution of the capacities of above edges in the min-cut $\{Y, Y'\}$ is $w(A) * |\text{net-cut}\{Y \cap A, C - (Y \cap A)\}|$.

Thus the capacity of a minimum cut $\{Y, Y'\}$ is equal to

$$\begin{aligned} &= \sum_{v \in Y' \cap A} cap(SOURCE, v) + \\ &\quad \sum_{\eta \text{ cut by } \{Y,Y'\}} cap(sink(\eta), source(\eta)) \\ &= (|\delta(A)| * \sum_{v \in Y' \cap A} w(v)) + \\ &\quad (w(A) * |\text{net-cut}\{Y \cap A, V - (Y \cap A)\}|) \end{aligned}$$

Hence

$$\begin{aligned} &capacity(cut\{Y, Y'\}) \\ = \; &(w(A) * |\text{net-cut}\{Y \cap A, V - (Y \cap A)\}|) + \\ &(|\delta(A)| * w(Y' \cap A)) \\ = \; &w(A) * \delta(Y \cap A) - |\delta(A)| * w(Y \cap A) + \\ &|\delta(A)| * w(A) \end{aligned}$$

However, $capacity(cut(SOURCE, V - \{SOURCE\})) = |\delta(A)| * w(A)$. Thus,

$$\begin{aligned} &w(A)\delta(Y \cap A) - |\delta(A)|w(Y \cap A) \\ &+|\delta(A)|w(A)| \leq |\delta(A)|w(A) \end{aligned}$$

Therefore,

$$w(A) * \delta(Y \cap A) - |\delta(A)| * w(Y \cap A) \leq 0,$$

that is,

$$\frac{|\delta(Y \cap A)|}{w(Y \cap A)} \leq \frac{\delta(A)}{w(A)}$$

Thus the block of cells $Y \cap A$ output by the use of the network flow based approach has better ratio of $\frac{|\delta()|}{w()}$ than that of the block of cells A itself. Note that A is the block of cells output by a balanced bipartitioner such as hmetis.

As argued in [10], it is easy to see that

$$\text{ratio-cut}\{Y \cap A, C - (Y \cap A)\} \leq \text{ratio-cut}\{A, C - A\}.$$

Summarizing,

Theorem 4.1 *The ratio-cut of bipartitions $\{C_1', C - C_1'\}$ and $\{C_2', C - C_2'\}$ obtained by Delta-Mod_hyp_Metis is less or equal to that of the bipartition $\{C_1, C_2\}$ output by hmetis.*

The worst case running time complexity of our heuristic is clearly dominated by the time required for finding a maximum flow in the flow network. Noting that the number of vertices in the flow network that we use is proportional to the number of nets (we use two special vertices per net), we get the following.

Theorem 4.2 *Our heuristic has a theoretical worst case running time of $O(|H|^2 \log(|H|))$ where $|H|$ is the number of nets in the netlist.*

However, we shall see that the actual running times of the execution of our heuristic was of the same order as that of hmetis. Indeed, often in combinatorial optimization applications, the preflow-push based implementations of maximum flow algorithms have been surprisingly fast.

5 Experimental Results

We have implemented the algorithm in C/C++ using LEDA library [9] (compilation was done without optimizations). We tested our approach on ISPD98 Circuit Benchmark Suite (available at http://vlsicad.cs.ucla.edu/cheese/). It is a collection of large circuits with number of cells ranging from 12K to 210K.

For each benchmark circuit dohpart was run using various random seeds to produce different bipartitions to be used by Deltamod_hyp. Similarly hmetis was run with various balance factors (balance factor is the allowed percentage deviation in the sizes of two bipartitions, e.g. a balance factor of 1 allows the blocks of the bipartition to be in the range of 49-51%

Circuit Name	Number of Cells	Number of Nets	rc (hM)	rc (best)	rc(hM)/ rc(delM)	percentage improvement	time hM	time delM
ibm01	12752	14111	3.13E-6	3.07E-6	1.0165	1.6	4.0	2.4
ibm02	19601	19584	1.38E-6	1.38E-6	1.0	0.0	10.7	2.6
ibm03	23136	27401	6.35E-6	3.78E-6	1.67	67.0	12.4	3.6
ibm04	27507	31970	2.16E-6	2.16E-6	1.0	0.0	13.2	3.7
ibm05	29347	28446	8.75E-6	5.16E-6	1.63	63.0	19.1	4.7
ibm06	32498	34826	2.87E-6	2.63E-6	1.086	8.6	20.3	6.3
ibm07	45926	48117	9.45E-7	9.08E-7	1.041	4.1	24.5	10.1
ibm08	51309	50513	1.53E-6	1.53E-6	1.0	0.0	43.0	11.0
ibm09	53395	60902	7.76E-7	7.72E-7	1.002	0.2	30.7	9.1
ibm10	69429	75196	7.91E-7	6.18E-7	1.28	28.0	55.1	14.2
ibm11	70558	81454	6.06E-7	6.09E-7	1.00017	0.017	42.6	19.9
ibm12	71076	77240	8.77E-7	6.57E-7	1.334	33.4	54.6	15.5
ibm13	84199	99666	1.05E-7	1.05E-7	1.0	0.0	47.0	13.5
ibm14	147605	152772	1.31E-7	1.19E-7	1.096	9.6	100.1	44.0
ibm15	161570	186608	4.74E-8	4.71E-8	1.007	0.7	113.6	41.4
ibm16	183184	190048	1.67E-7	1.43E-7	1.171	17.1	153.7	36.2
ibm17	185495	189581	2.42E-7	2.37E-7	1.02	2.0	177.5	38.7
ibm18	210613	201920	1.4E-7	8.22E-8	1.704	70.4	215.3	37.6

Table 1: Experiment Results

of original circuit) to produce different bipartitions to be used by Deltamod_hyp.

The results of the experiments are reported in Table 1. In the table **rc** stands for ratio-cut, **hM** for hmetis, **delM** for DeltaMod_hyp_Metis and **doH** for dohpart.

The first column tells the name of the benchmark circuit, second and third tells the number of cells and

the number of nets respectively in the circuit.

The fourth column corresponds to the best ratio-cut obtained when hmetis was run with balance factor of 1,2,5,10,20,30,40 with 10 runs (one of the command-line arguments needed by hmetis) per experiment. This was done for benchmarking purpose, and the results with the least ratio-cut are reported.

For example, with ibm01 we obtained results [6273,203], [6231,213], [5912,181], [5430,188], [4438,112], [1297,101], where in [a,b] a corresponds to size of smaller partition and b being the cut. The result [4438,112] has the least ratio-cut and is reported.

Similarly fifth column tell the least ratio-cut obtained among the bipartitions obtained when

dohpart/DeltaMod_hyp(ie. dohpart applied on netlist to find A followed by DeltaMod_hyp which outputs $\{A', C - A'\}$) and hmetis/DeltaMod_hyp (ie. hmetis applied on netlist to find A followed by Deltamod_hyp which outputs $\{A', C - A'\}$) algorithms were applied to the circuits as defined in Section 4. The result reported has the best ratio-cut obtained among many results obtained. For example,with ibm07 we obtained results [1076,60], [4694,342], [5905,254], [5644,205], [9521,588], [15054,872]. The result [5644,205] has the best ratio-cut and is reported.

The sixth column corresponds to the ratio of reported ratio-cut obtained by hmetis to ratio-cut obtained from our heuristic for all benchmark circuits, while seventh column contains the percentage improvement obtained in ratio-cut.

Column eighth, describes the time taken by 10 runs of hmetis. The time reported here corresponds to the time taken by hmetis to find the bipartition with the best ratio-cut. Ninth coulum describes the time taken by DeltaMod_hyp subroutine as defined in Section 4. (It may be noted that the reported time doesn't include that taken by partitioners to get the initial bipartitions.) Surprisingly, the maxflow-

mincut based subroutine Deltamod_hyp takes about the same order of time as hmetis does. It may be pertinent here to remark that often in several applications the running time of network flow algorithms are unpredictably better than the worst case theoretical analysis hints at.

These tables reveal that our new heuristic found better ratio-cut partitions in all cases except for 5 (out of 18) benchmark circuits. Furthermore, it can be seen that time taken by 10 runs of hmetis to partition 210K cell circuit is nearly 210 seconds and by DeltaMod_hyp to do further processing is additional 37 seconds(All the experiments were done on a Pentium-IV Machine with a 1GHz processor and 1GB RAM). Hence the algorithm is quite efficient in the sense of running time.

6 Conclusion

We have designed and implemented a new heuristic for ratio-cut bipartitioning based on ideas from [12, 10]. We also made use of a truer hypergraph based modelling of VLSI netlist. The experiments show that running time of our heuristic is good and the quality of the resulting partitions improve upon the best of hmetis results. The quality and the speed of this new heuristic would be of considerable use in identification of natural clusters in netlists[11].

References

[1] C.J. Alpert and A.B. Kahng, Recent Directions in Netlist Partitioning: A Survey In *INTEGRATION, the VLSI Journal*, vol. 19, 1995, pp.1-81.

[2] C.J. Alpert, A.B. Kahng and S.Z. Yao, Spectral Partitioning with Multiple Eigenvectors *Discrete Applied Mathematics*, vol. 90, 1999, pp. 3-26.

[3] A. E. Caldwell, A. B. Kahng, and I. L. Markov, Improved Algorithms for Hypergraph Bipartitioning. *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 2000, pp. 661-666.

[4] S. Dutt and W. Deng, A probability-based approach to VLSI partitioning. *Proc. ACM/IEEE, Design Automation Conf*, June, 1996, pp. 100-105.

[5] A.V. Goldberg and R.E. Tarjan, A new approach to the maximum flow problem, *J. Assoc. Computing Mach.*, vol. 35, 1988, pp.921-940.

[6] G. Karypis and Vipin Kumar, Multilevel k-way Hypergraph Partitioning *Proc. ACM/IEEE Design Automation Conf.*, 1999, pp. 343-348.

[7] F.T. Leighton and S. Rao, An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximate algorithms 29^{th} *ann. Symposium on Foundations of Computer Science*, 1988, pp. 422-431.

[8] P. Mazumder, E. Rudnick Genetic Algorithm for VLSI Design, Layout and Test Automation, Prentice Hall, 1999.

[9] Kurt Melhorn, Stefan Naeher, LEDA, A platform for combinatorial and geometric computing, http://www.algorithmic-solutions.com

[10] Sachin B. Patkar, H. Narayanan, An Efficient Practical Heuristic For Good Ratio-Cut Partitioning In 16^{th} *International Conference on VLSI Design*, 2003.

[11] Y.C. Wei, C.K. Cheng An improved two-way partitioning algorithm with stable performance. *IEEE Trans. Computer Aided Design*, vol 10. no.12, 1991 pp.1502-1511.

[12] Honggua Yang and D. F. Wong, Efficient Network Flow Based Min-Cut Balanced Partitioning In *Proc. IEEE Intl. Conf. Computer Aided Design*, vol. 19, 1994, pp.50-55.