VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques*

Shantanu Dutt and **Wenyong Deng**
EECS Department — Cadence
University of Illinois-Chicago — Design Systems
Chicago, IL 60607-7053 — San Jose, CA 95134

Principal Contact: Shantanu Dutt
Phone: 312-355-1314; Fax: 312-413-0024
E-mail: dutt@eecs.uic.edu

Abstract

Move-based iterative improvement partitioning (IIP) methods, such as the Fiduccia-Mattheyses (FM) algorithm [11] and Krishnamurthy's Look-Ahead (LA) algorithm [15], are widely used in VLSI CAD applications, largely due to their time efficiency and ease of implementation. This class of algorithms is of the "local/greedy improvement" type, and they generate relatively high quality results for small and medium size circuits. However, as VLSI circuits become larger, these algorithms suffer a rapid deterioration in solution quality. We propose new IIP methods CLIP and CDIP that select cells to move with a view to moving clusters that straddle the two subsets of a partition, into one of the subsets. The new algorithms significantly improve partition quality while preserving the advantage of time efficiency. Experimental results on 25 medium to large size ACM/SIGDA benchmark circuits show up to 70% improvement over FM in mincut, and average mincut improvements of about 35% over all circuits and 47% over large circuits. They also outperform state-of-the-art non-IIP techniques, the quadratic-programming-based method Paraboli [20] and the spectral partitioner MELO [3] by about 17% and 23%, respectively, with less CPU time. This demonstrates the potential of sophisticated IIP algorithms in dealing with the increasing complexity of emerging VLSI circuits. We also compare CLIP and CDIP to hMetis [16], one of the best of the recent state-of-the-art partitioners that are based on the multilevel paradigm (others include ML_c [1] and LSR/MFFS [5]). The results show that one scheme of hMetis is 8% worse than CLIP/CDIP and the other two schemes are only 2-4% better. However, CLIP/CDIP have advantages over hMetis and other multilevel partitioners that outweigh these minimal mincut improvements. The first is much faster times-to-solution (for example, one of our best schemes CLIP-LA2 is 6.4 and 11.75 times faster than the two best hMetis schemes) and much better scalability with circuit size (e.g., for the largest circuit with about 162K nodes, CLIP-LA2 is 10.4 and and 21.5 times faster and obtains better solution qualities than the two best hMetis schemes). Secondly, CLIP/CDIP are "flat" partitioners, while multilevel techniques perform a sequence of node clustering/coarsening before partitioning the circuit. In complex placement systems such as timing-driven placement in the presence of multiple constraints, such circuit coarsening can hide crucial information (needed for a high quality solutions) thus making the partitioning process oblivious to them. This, however, is not a problem with flat partitioners like CLIP/CDIP that can take all important parameters into account while partitioning. All these advantages make CLIP/CDIP very suitable for use in complex physical design problems for large deep submicron VLSI circuits.

(1) Categories and Subject Descriptors: *B.7 Integrated Circuits: B.7.2 Design Aids, J.6 Computer-Aided Engineering: J.6.0 Computer-aided design (CAD)*
(2) General Terms: *Algorithms, Design*
(3) Keywords and Phrases: *clusters, iterative-improvement, mincut, physical design/layout, VLSI circuit partitioning*

*Support for this work was partly provided by an UIC research grant. This is a significantly extended version of the paper that appeared in ICCAD-96 (Ref. [9]).

1 Introduction

Partitioning plays a key role in the design of very large scale integrated (VLSI) circuitry. With the growing complexity of modern VLSI design, a divide-and-conquer approach is routinely used to partition a complex system into smaller and simpler subcircuits. At the system or board level, a circuit is divided into many subcircuits which can be mapped to a chip in a printed circuit board (PCB) or a multichip module (MCM) system. The interconnections between the chips are minimized either to achieve high performance or to meet their I/O pin constraints. At the chip level, partitioning is used to minimize the total wire length so as to reduce the chip area. It can also be used to reduce the critical path length to achieve better performance.

The essence of VLSI circuit partitioning is to divide a circuit into a number of subcircuits with minimum interconnections between them. This can be accomplished by recursively partitioning a circuit into two parts until we reach the desired level of complexity. Thus two-way partitioning is a basic problem in circuit partitioning and placement.

Kernighan and Lin [14] proposed the well-known KL heuristic for graph partitioning. The KL algorithm starts with a random initial two-way partition and proceeds by swapping pairs of cells iteratively in order to maximally improve the cutsize (or cause minimal deterioration in it) at every swapping step. Schweikert and Kernighan [22] extended KL to hypergraphs so that it can partition actual circuits. Fiduccia and Mattheyses [11] reduced the complexity of the algorithm to linear-time with respect to the number of pins in the circuit. This is done by moving one cell at a time and using an efficient bucket data structure. Krishnamurthy [15] enhanced FM by adding higher level lookahead gains and improved the results for small circuits. All these algorithms improve an initial partition through a sequence of node moves (based on node "gains"), and thus fall under the class of iterative-improvement partitioners (IIPs). Recently, a number of non-IIP algorithms [3, 4, 12, 20, 21] have been proposed and excellent results have been obtained.

FM and LA are the most commonly used two-way partitioning algorithms largely due to their excellent run times, simple implementations and flexibility. However, this class of IIP algorithms have a common weakness, viz., they only find solutions corresponding to local minima. Because of their localized/greedy improvement nature, they can only evolve from an initial partition through very shortsighted moves. Thus the results strongly depend on the initial partition. In order to get a local minimum that is close to the optimum partition, multiple runs on randomly generated initial partitions are needed. As the circuit size becomes large, the probability of finding a good local minimum in one run will drop significantly. This makes FM an unattractive choice for partitioning large circuits. As will become clear later, FM gives good results on small to medium size circuits, but performs very poorly on large circuits. Some clustering [24] or compaction [21]

techniques have been proposed to remedy the above weakness. Very good results have been obtained at the cost of considerable CPU time increases and implementation complexities. In this paper, we propose a technique CLIP (CLuster-oriented Iterative-improvement Partitioner) that significantly improves the ability of IIP methods like FM and LA for finding good local minima. The new technique pays more attention to the neighbors of moved cells and encourages the successive moves of closely connected cells. This implicitly promotes the move of an entire densely-connected group (a cluster) into one subset of the partition. The large reduction in both the minimum and average cutsize of multiple runs indicates that CLIP is a more robust and stable approach. The increase in implementation complexity and run time is minimal. We also propose a sophisticated extension CDIP (Cluster-Detecting Iterative-improvement Partitioner) of this basic technique, which explicitly identifies clusters during the move sequence, and dynamically adapts the move sequence to facilitate the move of clusters into one subset of the partition. Very good results have been obtained at reasonable CPU times.

Around the same time as well as since the the first appearance of of the CLIP/CDIP algorithms in [9], a number of other powerful partitioners like PANZA [18], GMetis [2], STRAWMAN [13], hMetis [16], ML_c [1] and LSR/MFFS [5] were developed, that yield very good results for the ACM/SIGDA benchmark circuits; all except PANZA are of the IIP variety. Under the unit node-size assumption (made by almost all partitioners in order to obtain compatible results[1]), the CLIP/CDIP mincut results reported for the ACM/SIGDA benchmark suite are either better or within 3-6% of those reported by these partitioners. Except PANZA and STRAWMAN, the other state-of-the-art partitioners use the multilevel paradigm, which consists of a series of $l > 1$ coarsening or clustering stages of a given netlist into smaller netlists, followed by l stages of iterative-improvement partitioning (e.g., using FM, PROP [8], CLIP) and uncoarsening applied to the sequence of clustered netlists. Multilevel algorithms are in general recognized to produce the best mincut results to date. In this paper, we also compare CLIP/CDIP to one of the best multilevel partitioners, hMetis, for a number of medium to very-large industrial circuits.

In spite of the current trend towards multilevel partitioning, there are two major issues that make it important to pursue advances in flat IIP algorithms like CLIP/CDIP. First, the multilevel paradigm relies heavily on a good flat partitioner at each stage of partitioning and uncoarsening in order to obtain good results. For example, ML_c uses CLIP as the core partitioner, and LSR/MFFS's core partitioner was inspired by CLIP/CDIP

[1] LSR/MFFS [5] uses non-unit node areas. Thus it can be either easier or harder for it to satisfy a given balance constraint, e.g., 45%-55%, and it can hence obtain either better or worse mincuts than under the unit-area assumption. In other words, the space of feasible and thus optimal solutions are different under the two assumptions, and hence the LSR/MFFS results are not compatible with those of other partitioners.

concepts. Secondly, flat partitioners are important, especially in complex placement systems (e.g., timing-driven placement [6, 19] in the presence of multiple constraints [7, 17]), where it may be detrimental to "hide" crucial information like constraint parameters and net criticalities, that are needed for a high quality solutions, within meta-nodes obtained by clustering subcircuits. This, however, is not a problem with flat partitioners like CLIP/CDIP, which can take all important parameters into account while partitioning. Further, as we show in Sec. 4.5, CLIP's and CDIP's mincut results for the industrial circuits are between 8% better and 4% worse than various hMetis schemes, and their solution qualities and times-to-solution scale much better with circuit size than those of hMetis. Thus advanced flat partitioners like CLIP/CDIP offer an attractive alternative to multilevel methods for use in complex physical design problems for emerging large deep submicron (DSM) VLSI circuits.

In the next section, we briefly describe the FM and LA algorithms and point out their shortcomings. Then in Section 3 we present our rationale for the new partitioning algorithm CLIP. An extended algorithm CDIP that detects clusters as they are moved out of the cutset (thereby being able to reset node gains when a cluster removal is completed) is also proposed in this section. Extensive experimental results along with discussions are presented in Section 4 in which we compare CLIP/CDIP to three classes of partitioning algorithms: (1) Classical IIP methods (FM [11] and LA [15]); (2) State-of-the-art non-IIP techniques (Paraboli [20] and MELO [3]); and (3) Multilevel IIP algorithms (hMetis [16]). Conclusions are in Section 5.

2 Previous Iterative Improvement Algorithms

A circuit netlist is usually modeled by a hypergraph $G = (V, E)$, where V is the set of cells (also called nodes) in the circuit, and E is the set of nets (also called hyperedges). We denote the number of nodes by n and the number of nets by e. Each net connects two or more cells in the circuit. We will represent a net n_i as a set of the cells that it connects. A *two-way partition* of G is two disjoint subsets V_1 and V_2 such that each cell $v \in V$ belongs to either V_1 or V_2. A net is said to be *cut* if it has at least one cell in each subset and *uncut* otherwise. We call this the *cutstate* of the net. All the nets that are cut form a set called the *cutset*. The objective of a two-way partitioning is to find a partition that minimizes the size of the cutset (called the *cutsize*). Usually there is a predetermined *balance criterion* on the size of the subsets V_1, V_2, for example, $0.45 \leq |V_i|/|V| \leq 0.55$, where $i = 1, 2$.

The FM algorithm [11] starts with a random initial partition. Each cell u is assigned a gain $g(u)$ which

is the *immediate* reduction in cutsize if the cell is moved to the other subset of the partition:

$$g(u) = \sum_{n_i \in E(u)} c(n_i) - \sum_{n_j \in I(u)} c(n_j) \tag{1}$$

where $E(u)$ is the set of nets that will be immediately moved out of the cutset on moving cell u, and $I(u)$ is the set of nets that will be newly introduced into the cutset. Put in another way, a net in $E(u)$ has only u in u's subset, and a net in $I(u)$ has all its cells in u's subset. $c(n_i)$ is the weight (cost) of the net n_i which is assumed to be unity in this paper unless otherwise specified.

The goal of FM is to move a cell at a time from one subset to the other subset in an attempt to minimize the cutsize of the final partition. The cell being selected for the current move is called the *base* cell. At the start of the process, the cell with maximum gain value in both subsets is checked first to see if its move will violate the balance criterion. If not, it is chosen as the base cell. Otherwise, the cell with maximum gain in the other subset is chosen as the base cell. The base cell, say u_1, is then moved to the other subset and "locked"—the locking of a moved cell is necessary to prevent thrashing (a cell being moved back and forth) and being trapped in a bad local minimum. The reduction in cutsize (in this case, the gain $g(u_1)$) is inserted in an ordered set S. The gains of all the affected neighbors are updated—a cell v is said to be a *neighbor* of another cell u, if v and u are connected by a common net. The next base cell is chosen in the same way from the remaining "free" (unlocked) cells and the move process proceeds until all the cells are moved and locked. Then all the partial sum $S_j = \sum_{t=1}^{j} g(u_t), 1 \leq j \leq n$, are computed, and p is chosen so that the partial sum S_p is the maximum. This corresponds to the point of minimum cutsize in the entire moving sequence. All the cells moved after u_p are reversed to their previous subset so that the actually moved cells are $\{u_1, \ldots, u_p\}$. This whole process is called a *pass*. A number of passes are made until the maximum partial sum S_p is no longer positive. This is a local minimum with respect to the initial partition $[V_1, V_2]$.

The FM algorithm has been criticized for its well-known shortsightedness [15, 8]—it moves a cell based on the immediate decrease in cutsize. Thus it tends to be trapped in local minima that strongly depend on the initial random partition. We will later point out some other consequences of this shortsightedness that are related to the removal of natural clusters, which occur in real circuits, from the cutset.

The FM gain calculation only considers *critical* nets whose cutstate will change *immediately* after the move of the cell. It is conceivable there will be many cells having the same gain value since the gain is bounded above by p_{max} and below by $-p_{max}$, where p_{max} is the maximum degree of a cell. Krishnamurthy suggested a refined gain calculation scheme which includes less critical nets. A net is critical in some degree if the move of cell u possibly followed by moves of one or more of its neighbors will change the cutstate of

the net. Thus we obtain a gain vector $\mathbf{g}(u)$ whose elements are ordered by the degree of criticality of the nets from which the kth gain element is computed:

$$g_k(u) = \sum_{n_i \in E_k(u)} c(n_i) - \sum_{n_j \in I_k(u)} c(n_j) \tag{2}$$

where a net in $E_k(u)$ has exactly k cells (including u) in u's subset and a net in $I_k(u)$ has exactly $k-1$ cells in the other subset. A gain vector $\mathbf{a}$ is said to be greater than a gain vector $\mathbf{b}$ if either $\mathbf{a}[1] > \mathbf{b}[1]$ or if there exits an $i < k$ such that $\mathbf{a}[j] = \mathbf{b}[j]$ for all $1 \leq j \leq i$ and $\mathbf{a}[i+1] > \mathbf{b}[i+1]$.

Before we proceed to the next section, we distinguish between the "actual" move and the "virtual" move of a cell in a pass. In an iterative improvement scheme like FM and LA, all cells are *virtually moved*, and a cell is *actually moved* from its original subset to the other one, if its virtual move lies at or before the maximum partial sum of gains point calculated over all virtual cell moves. Cells beyond this point are not actually moved and revert back to their original subset. In Section 3, except in algorithm descriptions, whenever we refer to the moving of a cell in the discussion, we mean the actual move of the cell, not its virtual move.

3 Cluster-Oriented Iterative Improvement Methods

3.1 Case for a Cluster-Oriented Approach

A real VLSI circuit netlist can be visualized as an aggregation of a number of highly connected subcircuits or clusters. It is conceivable that there are many levels of clusters with different degrees in the density of their connectivities. A small group of densely interconnected cells may be part of a larger but less densely connected cluster. The goal of two-way partitioning is to determine a cut that goes through the most weakly connected groups.

Iterative improvement algorithms like FM and LA start with a randomly assigned partition that results in a binomial distribution of cells in V_1 and V_2. In such a distribution, the probability of finding r $(r \leq m)$ cells in V_1 and $m-r$ cells in V_2 in some group of m cells is:

$$P(m, r) = \binom{m}{r} p^r q^{m-r} \tag{3}$$

where p (q) is the probability of a cell being assigned to V_1 (V_2). For a random partition, $p = q = 0.5$. The probability distribution maximizes at $r = m/2$ with standard deviation $\sigma = \sqrt{m}/2$. For example, a group of 100 cells will have the expected distribution of 50 cells in each subset with a standard deviation of 5 cells. Therefore, for a cluster with a fair number of cells, there is a very high probability that it will initially be cut.

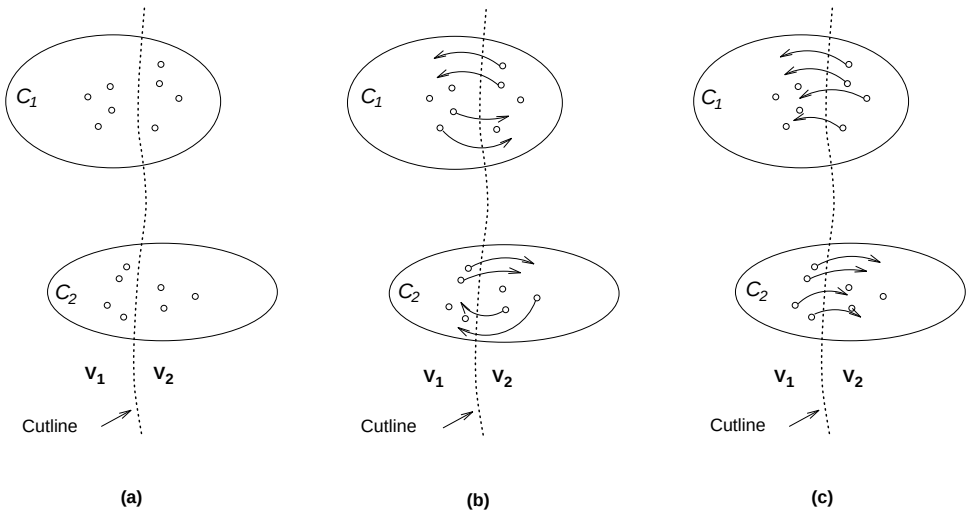

Figure 1: (a) In the initial partition, clusters straddle the cutline as a result of random cell assignment. (b) FM locks clusters on the cutline by moving cells within one cluster in both directions. (c) Better approaches pull clusters out from the cutline by moving cells within one cluster in a single direction.

Hence in an initial random partition, most clusters will straddle the *cutline*, which is an imaginary line that divides the cells into the two subsets of the partition. This situation is illustrated in Figure 1(a).

For an IIP algorithm to succeed, it must be able to pull clusters straddling the cutline into one subset. It is easy to visualize that there will be many cells in different clusters with similar situations and therefore the same gain values. Since there is no distinction between cells with the same gain values but belonging to different clusters, the FM algorithm may well start to work on many clusters simultaneously, trying to pull them out of unfavorable situations. However, cell movement is a two-way process, and while some cells in a cluster are moved from V_2 to V_1, other cells in the same cluster might be moved from V_1 to V_2. Thus the cluster can be locked in the cutset at an early stage —a cluster is said to be *locked* in the cutset if it has locked cells in both subsets of the partition. This is the situation of clusters C_1 and C_2 in Figure 1(b). Unfortunately, the moves made at an early stage (before the maximum partial sum point) are the actual moves. Hence these clusters will not be pulled out from the cutset in the current pass, and in later passes the same scenario may reappear.

Undoubtedly, FM does a good job of pulling small and highly connected clusters to one side. This is because many nets of a small cluster will be incident on each cell of the cluster, and thus their gains capture significant information about the cluster. Once a cell is moved from V_1 to V_2, its neighbors in V_1 might have their gains raised and hence have a better chance of being moved to V_2, while its neighbors in V_2 will get a negative gain contribution from nets connected to the moved cell and hence tend to stay in V_2. Therefore a small cluster has a good likelihood of being moved entirely to V_2. However, when the size of a cluster is

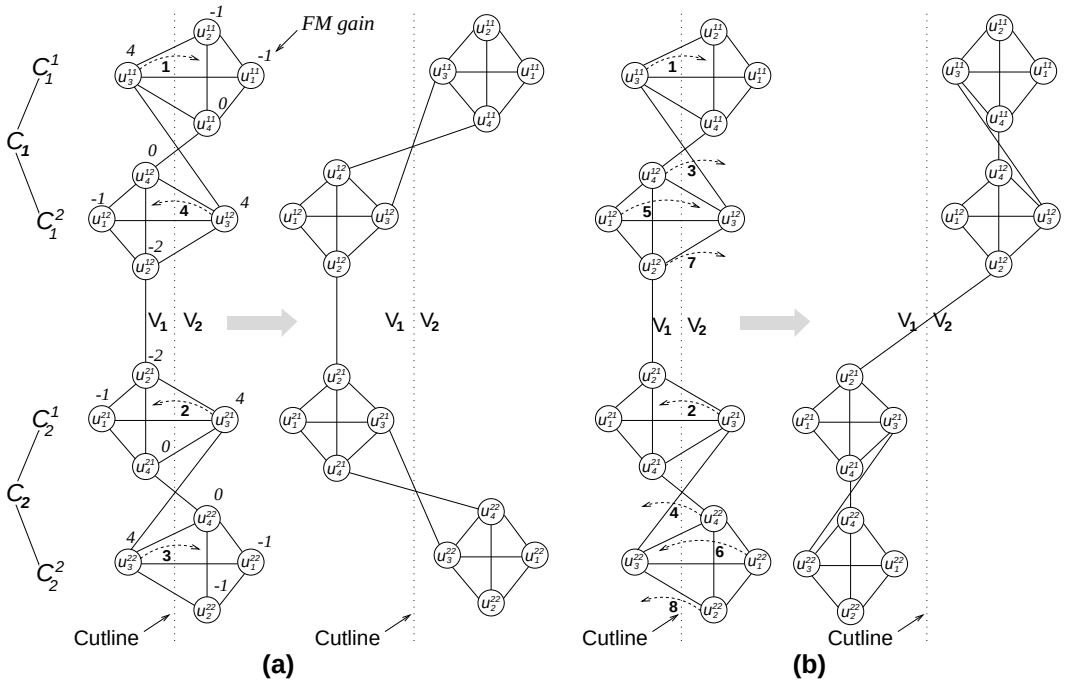

Figure 2: (a) FM only pulls out subclusters and finds a local minimum in cutsize. (b) The new approach pulls out clusters and finds the optimum cut.

large, its structural properties cannot be simply represented in the *immediate* connectivities of cells such as in FM and LA gain calculations; for example, the cluster may consist of subclusters. Although it might be easy for FM to move out subclusters, it is also very likely that it will lock the bigger clusters in the cutset.

To demonstrate the validity of the above observations, let us consider the simple graph shown in Figure 2(a) that shows a two-level clustered structure. Cells u_1^{11}, u_2^{11}, u_3^{11} and u_4^{11} form a strongly connected subcluster C_1^1; other subclusters C_1^2, C_2^1 and C_2^2 are similarly formed. Subclusters C_1^1 and C_1^2 construct a higher level but less densely interconnected cluster C_1. Similar is the case for C_2, which is composed of sublusters C_2^1 and C_2^2. After a random partition, all the clusters as well as the subclusters straddle the cutline. Initial FM gain calculation gives the gain values indicated beside each cell in the figure. Cells u_3^{11} and u_3^{22} belong to different clusters, but have similar situations, and hence the same gain of 4. We assume a 50%-50% balance criterion in which cells move alternately between the two subsets V_1 and V_2. The first four moves are shown by the numbered dashed arrows in Figure 2(a). FM quickly reaches the local minimum of cutsize 4 (further moves will be reversed finally since the current point is the maximum partial sum point). While FM succeeded in moving out subclusters, it locked the higher level clusters C_1 and C_2 on the cutline. Therefore it missed the optimal cut of one that can be easily identified in the figure.

It is obvious now that a mechanism is needed to aid IIP algorithms in pulling out clusters from the cutset.

We propose a cluster-oriented framework for gain calculation and base-cell selection that focuses on nets connected to moved cells. It can be overlaid on any IIP algorithm with any cell-gain calculation scheme. It implicitly promotes the move of an entire cluster by dynamically assigning higher weights to nets connected to recently moved cells. This greatly enhances the probability of finding a close-to-optimum cut in a circuit. We also propose an extended version of this algorithm that identifies clusters explicitly as they are being moved out of the cutset.

3.2 Considering Clusters in Iterative Improvement Methods

We first re-examine the cell gain calculation of FM. Initially, cell gains are calculated based on the immediate benefits of moving cells. After a cell is moved, the gains of its neighbors are updated. At any stage in the move process, the *total* gain of a cell can be broken down as the sum of the initial gain component and the updated gain component. The total gain indicates the overall situation of a cell, while the updated gain component reflects the change in the cell's status due to the movements of its neighbors.

An intuitive solution to the problem of an IIP scheme "jumping around" and working on different clusters simultaneously, as illustrated in Figs. 1(b) and 2(a), thus locking them in the cutset, is to make node movement decisions based primarily on their updated gain components. This minimizes distractions during the cluster-pulling effort caused by nodes not in the cluster currently being moved, but with high total gains. In other words, it allows the algorithm to concentrate on a single cluster at a time for moves in one direction; note that the updated gain component of a node reflects its goodness for moving with regard to the cluster currently being pulled from the cutset. The initial gain of a node, however, provides useful information for choosing the starting seed for removal of a cutset-straddling cluster—the cell with the highest gain is most likely in such a cluster, and thus a very good starting point. Once the move process has begun, nets connected to moved cells should be given more weights so that the updated gain components of cells become more important than their initial gains. The utility of giving more weight to nets connected to moved nodes (and hence to the updated gain components of nodes), in facilitating the movement of clusters from the cutset is established in the following set of results.

For a cluster C, the set of C's internal nets $I(C)$ and the set of C's external nets $E(C)$ are defined as $I(C) = \{n_i \mid n_i \subset C\}$, $E(C) = \{n_j \mid \exists u, v \in n_j, \text{ s.t. } u \in C, v \notin C\}$. We assume that a cluster has the following two reasonable properties:

(1) For any node $u \in C$,

$$\sum_{u \in n_i, n_i \in I(C)} c(n_i) \quad > \sum_{u \in n_j, n_j \in E(u)} c(n_j) \tag{4}$$

i.e., a cell in a cluster has stronger internal connections than external connections.

(2) The probability that an edge connects a pair of nodes inside the cluster is uniformly distributed and equal to f_{int}, the probability that an edge connects a node in C to a node outside C is also uniformly distributed and equal to f_{ext}, and $f_{int} > f_{ext}$. This is similar to the *uniformly distributed random graph* model used by Wei and Cheng in [23].

(3) An "average" node u in C is connected to more nodes in C than outside C. This means that $f_{int}(|C| \times (|C|-1)/2) > f_{ext}(|C| \times (|V|-|C|))$ and this translates approximately to

$$f_{int} > 2\frac{|V|}{|C|} \times f_{ext}$$

.

We consider an IIP process like FM, and assume for simplicity of exposition that all original net weights are one; our techniques are easily adapted to cases where nets have different weights. When updating the gain of a node, we also assume that each net connected to moved cells is assigned a weight of at least $2p_{max}$, where recall that p_{max} is the maximum cell degree in the circuit.

Theorem 1 *If a cluster C is divided by the cutline into subsets $C_1 \in V_1$ and $C_2 \in V_2$, and C_1 is moved to V_2 by a sequence of moves of its nodes, then the cutsize of the partition will decrease if initially $|C_1| \leq |C_2|$, or first increase then decrease if initially $|C_1| > |C_2|$.*

Proof: The change in the cutsize comes from the cutstate change of nets in $I(C)$ and $E(C)$.

First, consider the subset $C_{c_1c_2}$ of internal edges $I(C)$ that are in the cutset. The expected cardinality of $C_{c_1c_2}$ is:

$$|C_{c_1c_2}| = f_{int} \times |C_1| \times |C_2| = f_{int} \times (|C| \times |C_2| - |C_2|^2) \tag{5}$$

Eqn. 5 is a quadratic function of $|C_2|$, and maximizes at $|C_2| = |C|/2$. When the removal of cluster C starts from $|C_1| \leq |C_2|$ (i.e., $|C_2| \geq |C|/2$), $C_{c_1c_2}$ will decrease as $|C_2|$ increases; if the removal of cluster C starts from $|C_1| > |C_2|$ (i.e., $|C_2| < |C|/2$), $C_{c_1c_2}$ will increase until $|C_2| = |C|/2$, then decrease to zero at $|C_2| = |C|$, and the overall effect is a decrease of $C_{c_1c_2}$. In general, the rate of change in $|C_{c_1c_2}|$ as nodes are being moved is

$$\frac{d\,|C_{c_1c_2}|}{d\,|C_2|} = f_{int} \times (|C| - 2|C_2|)$$

With $|C_2|$ ranging from 0 to $|C|$, the average absolute value of this rate of change is

$$2\frac{f_{int}}{|C|}\sum_{i=0}^{|C|/2}(|C|-2i) = \frac{f_{int}}{|C|}\sum_{j=0}^{|C|/2} j \approx 0.25 f_{int}|C|$$

Next, consider the cutsize change from external edges $E(C)$. The cutsize change from the external edges by moving a node $u \in C$ from V_1 to V_2 is

$$\frac{d\,|E(C)|}{d\,|C_2|} = f_{ext} \times (|V_1| - |C_1|) - f_{ext} \times (|V_2| - |C_2|) \approx f_{ext} \times (|V_1| - |V_2|) < \frac{f_{int}}{2} \times \frac{|C|}{|V|} \times (|V_1| - |V_2|)$$

using property 3 of a cluster and the fact that (for large circuits) $|V_1| \gg |C_1|, |V_2| \gg |C_2|$. For a roughly equal size partition, such as 45-55% partition sizes, $||V_1| - |V_2|| \leq 0.1|V|$, and thus $\frac{d\,|E(C)|}{d\,|C_2|} < 0.05 f_{int} \times |C|$. Thus it can be seen that the absolute average value of $\frac{d\,|C_{c_1c_2}|}{d\,|C_2|}$ is much larger than that of $\frac{d\,|E(C)|}{d\,|C_2|}$, and hence the cutsize change as C is being moved out of the cutset is dominated by the former, and its effect is as established above. □

Figure 5 illustrates Theorem 1.

Observation 1 *Once a cluster C starts to move from V_1 to V_2, there is a high probability that cells of C in V_1 will be actually moved to V_2.*

Justification : Let $C_1 = C \cap V_1$ and $C_2 = C \cap V_2$. There are two cases.

Case 1: $|C_1| \leq |C_2|$

From Theorem 1, since $|C_1| \leq |C_2|$, the movement of the nodes of C_1 from V_1 to V_2 will reduce the cutsize. This means that all nodes in C_1 have high gains even if none of their incident edges have been heavily weighted (otherwise, the cutsize will not decrease on moving them to V_2). The probability that all edges incident on those nodes of C moved to V_2 lie entirely in V_2 is $(\frac{|C_2|}{|C|})^{ql}$, where q is the average node degree of cluster C and l is the number of moved nodes in C_2. For a cluster with average node degree three[2], the above probability is very low, due to the exponential factor ql, until almost all nodes in C_1 are moved to V_2, i.e., until $\frac{|C_2|}{|C|} \approx 1$. For example, when $|C| = 100$, and initially $|C_1| = 30$ and $|C_2| = 70$, the above probability is only 0.18 at the point when $|C_1|$ is only two; it is 0.1 after the first two cells have been moved from C_1 to C_2. Therefore there is a high probability that some nodes in C_1 will have connections to moved nodes in C_2. Due to the extra weight of such edges, the gains of these nodes will be raised even higher (than that predicted by Theorem 1, as mentioned above). It is possible for some external nodes to have high gains due to their connections to moved nodes in C_2. However, since they do not necessarily belong to a single cluster, the

[2]The average node degree over all the benchmarks in this paper is 3.22.

moves of these nodes will not, on the average, change the cutsize significantly. Therefore the gains of nodes in C_1 will dominate, and these nodes will be moved to V_2 even though there might be a few external nodes also being moved in this sequence.

Case 2: $|C_1| > |C_2|$

Since the cluster has started movement, at least one node u has been moved from V_1 to V_2. The probability that some node $v \in C_1$ has an edge connecting u is $1 - (\frac{|C_2|}{|C|})^q$, which is very high due to the fact that $|C_1| > |C_2|$. This gives v a weighted gain of $2p_{max}$ for that edge. In the worst case, when all the other $p_{max} - 1$ edges incident on v are in V_1, v still has a gain of $p_{max} + 1$. It will then become the node with the highest gain in V_1 and will be moved to V_2. Such a process continues until $|C_1| = |C_2|$, at which point Case 1 applies. □

Observation 2 *During the move of cluster C from V_1 to V_2, a node $u \in C_2$ has a high probability of not being actually moved from V_2 to V_1, where $C_2 = C \cap V_2$.*

Justification: The maximum gain of u is p_{max} when all its edges are initially in the cutset. To satisfy the size balance of V_1 and V_2, there must be another cluster C' which is being moved from V_2 to V_1. Denoting the subsets of C' in V_1 and V_2 as C'_1 and C'_2, respectively, the probability that all edges incident on the locked nodes in C'_1 are contained in V_1, viz., $(\frac{|C'_1|}{|C'|})^{ql}$, is very low even when just a few nodes in C' have been moved from V_2 to V_1. (Assuming that only two nodes of C' are moved to V_1, the above probability is 0.09 when $|C'_1| = 2 \times |C'_2|$). Therefore there is a high probability that some node $v \in C'_2$ has an edge connected to a moved node in C'_1. Hence v will have at least a gain of $p_{max} + 1$ due to the $2p_{max}$ weight on that edge. Therefore node $v \in C'_2$ will be moved to V_2 instead of node $u \in C_2$. Note that u will eventually be virtually moved to V_2, since all nodes are so moved in an IIP algorithm. However, since u's move will be after almost all "good" nodes have been moved, as just explained, it is unlikely that it will lie at or before the maximum partial sum point. □

Observation 3 *Once a cluster starts to move from V_1 to V_2, there is a high probability that the whole cluster will be removed from the cutset.*

Justification: Follows from Observations 1 and 2. □

Although the above results are for clusters that meet the criteria we specified earlier, they give an idea of the efficacy of our approach to removing more general clusters in a real circuit from the cutset.

From the justifications of Observations 2 and 3, it can also be concluded that with high probability the two move processes, $M_{1\to 2}$ (moving of cells from V_1 to V_2) and $M_{2\to 1}$ (moving of cells from V_2 to V_1), work

on different clusters with the least interference with each other. Such a situation is shown in Figure 1(c). As a result, clusters are more easily pulled out of the cutset instead of getting locked in it, as in the original FM and LA schemes.

The example of Fig. 2 can be used to demonstrate the advantage of the above approach. The cell gains are first computed and the base cell is again u_3^{11}. The first two moves of u_3^{11} and u_3^{21} bring large negative gains to cells u_3^{12} and u_3^{22} through the weighted edges. Therefore in the subsequent two moves, they are not selected as would be the case in FM. Instead, u_4^{12} becomes the top cell in V_1, and is selected as the next cell to move. Thus begins the process in which $M_{1\rightarrow 2}$ works on cluster C_1 and $M_{2\rightarrow 1}$ works on cluster C_2. The sequence of the first few moves are indicated by numbered arrows in the figure[3]. After eight moves, C_1 and C_2 are moved out from the cutset, and we obtain the optimal cut of one. Thus this process escapes the local minimum of four in which the original FM algorithm was trapped.

3.3 A Cluster-Oriented Iterative-Improvement Partitioner

From the above discussion, we propose a general gain calculation and base-cell selection framework CLIP (for CLuster-oriented Iterative-improvement Partitioner) presented in Figure 3, that can be applied to any IIP algorithm. For implementation convenience, we set the cell gains to zero after the initial gain calculation. Cell gains are updated as in the original algorithm over which CLIP is overlaid. Zeroing of initial gains followed by gain updating is equivalent to giving nets connected to moved cells a weight of infinity. The initial gain information is only reflected in the initial ordering of cells.

After the first pass, most strongly connected clusters will probably have been removed from the cutset. The few clusters left in the cutset can be removed in the subsequent passes. In later passes, another advantage of the above cluster-oriented scheme is that clusters lying entirely in one subset can be easily moved to the other subset. This is because cell gains being cleared to zero in the initial stage causes cells in a cluster to have less inertia in staying inside their original subset. The benefit of cluster movement between subsets is that larger but less densely connected clusters (we call them *superclusters*) can be removed from the cutset by moving their densely-connected constituent clusters from one subset to the other. By rearranging clusters between the two subsets in this way, subsets V_1 and V_2 of the final partition will become the two largest but most weakly connected superclusters, which implies that the cutsize will be small. Figure 4 illustrates the movement of clusters C_1^2 and C_2^1 across the cutline due to which a better cut through the loosely connected

[3]The details of the gain updating and cell moves for this example are given in the appendix of [10], an extended version of this paper that is available at ftp site ftp-mount.ee.umn.edu in file /pub/faculty/dutt/vlsi-cad/papers/pap-pdw96-ext.ps.

Algorithm `CLIP`

1. Calculate the initial gain of all cells according to the IIP algorithm of choice (e.g., FM, LA, PROP[8]).
2. Insert the cells into some sorted data structures (free-cell store) T_1 and T_2 for subsets V_1 and V_2, respectively. Select the maximum gain cell $u \in V$ as the first base cell to move.
3. Clear the gain of all cells to zero while keep their original ordering in the data structure.
4. Move u and update the gain of its neighbors and their ranks in the data structure as done in the chosen IIP algorithm. The gain of a cell now only contains the updated part.
5. Choose the base cell based on the cell's updated gain and the balance criterion. Move the cell, update its neighbors.
6. **Repeat** Step 5 until all cells are moved.
7. Find the point in the move sequence which corresponds to the minimum cutsize, and reverse all the moves after this point.

Figure 3: One pass of CLIP (CLuster-oriented Iterative-improvement Partitioner)

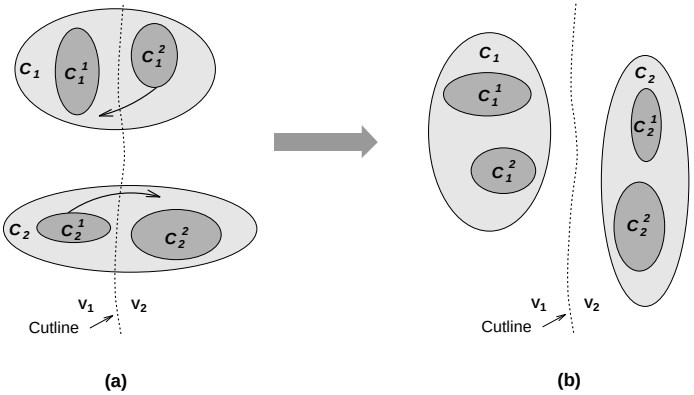

Figure 4: In later passes, larger but less densely connected clusters (superclusters) can be moved across the cutline. In previous passes, the more densely connected clusters (represented by a darker shade) were removed from the cutset.

groups C_1 and C_2 is achieved. As opposed to FM, which tends to do only local improvement within large clusters, the above new scheme can explore a wider solution space, and hence has less dependence on the initial partition.

Compared to other clustering-based approaches, such as bottom-up compaction [21], top-down clustering [24] and vertex ordering [4], CLIP does not explicitly bind cells together as inseparable clusters. Instead, cells can be implicitly regrouped into different clusters in subsequent passes. Both the moving and possible regrouping of clusters are guided directly by the ultimate objective—cutsize reduction. An advantage of this approach is that CLIP has more freedom in searching for the optimum cut.

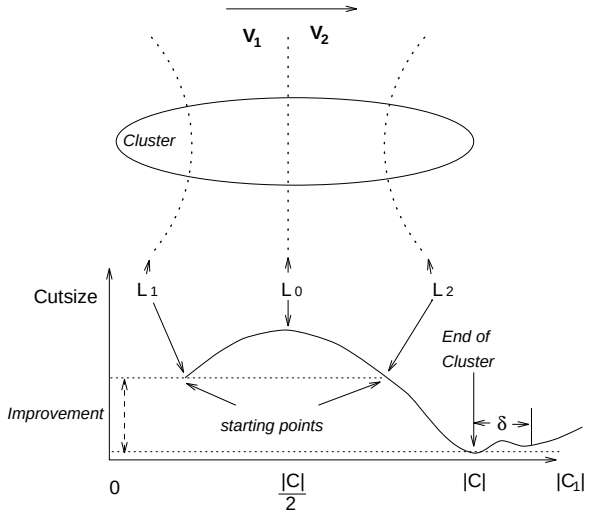

Figure 5: The cutsize change with the move of a cluster as indicated in Theorem 1. When the cutsize does not improve, it indicates the end of a cluster.

3.4 A Cluster Detection Method

Although the new framework CLIP has significant advantages over the traditional IIP approach, it is possible to do even better. We start by asking the following questions:

1. How do we know when a cluster has been pulled out?
2. Once we finish pulling out the first cluster, how do we select the next starting point?

We address the first question by determining what happens when a cutline sweeps across a cluster—this is equivalent to a cluster being pulled out from the cutset. From Theorem 1, as we move a cluster from the cutset, the overall cutsize will decrease until the cluster is entirely removed from the cutset. In a practical partitioner, some external cells may be moved across the cutline due to their connections to moved cells in the cluster. However, since they are randomly distributed across many clusters (and thus do not belong to a specific cluster), their contribution to the overall cutsize change will not be significant. As a result, we obtain a cutsize change similar to that illustrated in Figure 5, which is a pictorial depiction of Theorem 1. Referring to this figure, the movement of a cluster starts from either point L_1 or L_2. After it is removed from the cutset, there is an overall improvement in cutsize. If in subsequent moves no other cluster starts getting pulled out from the cutset, the cutsize will not improve further.

From this reasoning we propose the following cluster detecting criterion:

> *After the move process reaches a positive maximum improvement point, and there is no further improvement in the following δ moves, we say that a cluster has been moved out at the maximum improvement point; δ is a parameter of the algorithm.*

Referring to the cutsize curve in Figure 5, the δ cells moved after the minimum cutsize is reached do not belong to the previous cluster. This is in contrast to the two cluster detection criteria that Saab proposed in his compaction algorithm [21], viz., (1) The cutsize decreases for the first time after a sequence of moves; (2) The last moved cell in the sequence has a positive gain. We derived our detection criterion from the analytical results in Section 3.2, and we use the partial sum of gains of the sequence of node moves in our criterion, instead of individual cell gains as in [21].

We now address the second question raised at the beginning of this subsection. After reversing the δ moves, we come back to the end point of the current cluster. Subsequently, we need to select the next seed to start the move of another cluster. For this purpose, the updated gain components of cells should not be the determining factor. Rather, the total gains of cells, which reflect their overall situation, is more useful. Similar to the situation at the beginning of the pass, the cell with the maximum total gain is a good seed to start with. Following this, however, unlike at the beginning of the pass, gain components of nodes are selectively zeroed. Negative components due to connections to locked nodes are retained, since they signify attachment to a moved cluster, and a node with a significant negative gain most probably belongs to such a cluster and should not be moved; see Fig. 6 for more details.

From the above discussion, a more sophisticated algorithm CDIP (Cluster-Detecting Iterative-improvement Partitioner) is presented in Figure 6. This is an extension of CLIP and can also be applied to any IIP algorithm. The detection of the end of the kth cluster is done by monitoring the partial sum $S_j^{i,k}$ ($i = 1, 2$) for each cluster and each subset separately. After $S_j^{i,k}$ becomes positive and does not increase for δ moves, we infer that the move of the current cluster ended at the point of maximum partial sum $S_p^{i,k}$. We then reverse the moves to this point, and select the cell with the maximum total gain, which very likely belongs to an unmoved cluster, as the base cell. After selecting this cell, the free cells are reordered in a manner that reflects their relevant connectivities to previously moved clusters. Since some cluster C was previously moved from the cutset, say, into V_2, it is not desirable to move the free cells of C in V_2 to V_1—doing so will lock the cluster in the cutset. Therefore, as at beginning of the pass, all gain components of cells are cleared to zero, with the exception of those components that correspond to negative gains corresponding to connections to locked neighbors in the same subset. These high-weight negative gains that are retained reflect the fact that some clusters were moved previously in the pass, and that free cells strongly connected to moved cells in these clusters likely belong to them and thus should be given lower priorities for movement.

By using either of the two new partitioning algorithms, CLIP or CDIP, we can find a good cut through weakly connected clusters. Since most clusters will have been successfully moved at the end of the

Algorithm `CDIP`

1. Calculate the initial gain of all cells according to the IIP algorithm of choice.
2. Insert the cells into sorted data structures T_1 and T_2 for subsets V_1 and V_2, respectively. Select the maximum gain cell $u \in V$ as the first base cell to move.
3. Clear the gain of all cells to zero while keeping their original ordering in T_i ($i = 1, 2$).
4. Move u and update the gain of its neighbors and their ranks in T_i ($i = 1, 2$) as done in the chosen IIP algorithm. Start to count the move index j and to calculate the partial sum $S_j^{i,1}$ for this first cluster, where $u \in V_i$. The gain of a cell now only contains the updated part.
5. **Repeat** Steps 6 to 7 until all cells are moved and locked.
6. Choose the base cell based on the cell gain, and the balance criterion Move the cell, and update the gain of neighbors.
7. **If** the current maximum partial sum $S_p^{i,k} > 0$, and the current move index $q \geq p + \delta$, **then**
   (a) Reverse the moves from q to $p + 1$.
   (b) Choose the free cell $v \in V_i$ with the maximum *total* gain as the next base cell.
   (c) For each free cell in V_i, clear the cell gain except the gain component from nets connected to the locked cells in the same subset V_i. Reorder cells in T_i according to the modified gain.
   (d) Move the base cell v, update neighbors and start the count of new move index j and the calculation of new partial sum $S_j^{i,k+1}$ from this move.
8. Find the maximum prefix point in the move sequence that satisfies the balance criterion and reverse all the moves after this point.

Figure 6: One pass of CDIP (Cluster-Detecting Iterative-improvement Partitioner)

CLIP/CDIP algorithms, to obtain even better results, we can apply the original IIP algorithm as a post-processing phase to fine-tune the partition without cluster-oriented considerations.

3.5 Complexity Analysis

In an implementation, both CLIP and CDIP have to be overlaid on some chosen IIP algorithm. Let n be the number of cells, e the number of nets, p the number of pins and d the average number of neighbors of a cell. The only additional operation of CLIP over the chosen IIP algorithm is to clear the gain of all cells, which requires $O(n)$ time. For a bucket list structure as proposed in [11], we need to remove linked lists from all buckets except bucket $B(0)$ which contains zero gain cells, and concatenate them to $B(0)$ according to their original ranks. This can be done in $O(b)$ time, where $b = 2p_{max} + 1$ is the number of buckets. Thus CLIP-FM (CLIP applied to FM) with bucket list structure has the same time-complexity as FM, which is $O(p)$ as derived in [11]. CLIP-LA (CLIP applied to LA) can also be implemented with an $O(p)$ time complexity, the same as that of LA [15].

CDIP has two major additional operations beyond those in the CLIP algorithm. After each detection of

Test Case	# Cells	# Nets	# Pins	Test Case	# Cells	# Nets	# Pins
s1423	619	538	1528	p2	3014	3029	11219
sioo	664	408	1882	s9234	5866	5844	14065
s1488	686	667	2079	biomed	6514	5742	21040
balu	801	735	2697	s13207	8772	8651	20606
p1	833	902	2908	s15850	10470	10383	24712
bm1	882	903	2910	industry2	12637	13419	48404
t4	1515	1658	5975	industry3	15406	21924	68290
t3	1607	1618	5807	s35932	18148	17828	48145
t2	1663	1720	6134	s38584	20995	20717	55203
t6	1752	1541	6638	avq.small	21918	22124	76231
struct	1952	1920	5471	s38417	23949	23843	57613
t5	2595	2750	10076	avq.large	25178	25384	82751
19ks	2844	3282	10547				

Table 1: ACM/SIGDA benchmark circuit characteristics.

a cluster: (1) The δ moves are reversed, and (2) The free cells are reordered. Assuming c clusters are detected in a pass, the first operation implies $c\delta$ additional moves over the entire pass, and the second operation causes c reorderings of all free cells. For a bucket list structure, where the insertion of a cell into a bucket is a constant time operation, the complexity of the first operation is $O(c\delta d)$ (each extra move requires $O(d)$ time for updating d neighbors and reinserting them in the bucket data structure), and the complexity of the second operation is $O(cn)$ over the entire pass. Thus the complexity of CDIP is $O(max(c\delta d, cn, p))$ for CDIP-FM and CDIP-LA for one pass. Empirical results presented later show that CDIP is quite fast.

4 Experimental Results

The initial set of experiments have been done on ACM/SIGDA benchmark circuits whose characteristics are listed in Table 1. The circuit netlists were acquired from the authors of [4] and [20][4]. All cutset results are for the 45-55% balance criterion. Applications of CLIP and CDIP to FM, LA and PROP are compared to FM [11] and LA [15], and two state-of-the-art non-IIP algorithms, Paraboli [20], which uses mathematical programming, and MELO [3], which uses graph spectral techniques.

The next set of experiments compare CLIP/CDIP to one of the newest and current state-of-the-art partitioning technique, hMetis [16], which uses multilevel clustering methods, for a set of medium to very large industrial circuits whose characteristics are given in Table 6; hMetis is available at the URL http://www.cs.umn.edu/~metis.

In every comparison of an algorithm A to another algorithm B for a certain metric X, the percent im-

[4]Benchmarks *s1423, s9234, s13207, s15850, industry3* and *s38584* have some disconnected components and/or isolated cells which amount to less than 3% of the total number of cells. In all our experiments, the disconnected components and isolated cells are not specially treated, and left as they are.

provement of A over B is computed as $\frac{(X_B - X_A) \times 100}{X_B}$, where we assume that X is a minimization metric (as is the case of all metrics, viz., mincut, average-cut and time-to-solution, in this paper). Thus a positive value for the percent improvement means A is better than B and a negative value indicates that A is worse.

4.1 Comparisons to FM

Table 2 presents the results of applying CLIP to FM and LA3 (LA algorithm with lookahead level of 3). Both the minimum and average cutsizes over 20 runs are greatly improved compared to their corresponding original schemes. The overall minimum-cutsize improvements are 24.5% for CLIP-FM over FM and 45.4% for CLIP-LA3 over LA3. Note also from the table that while LA3 performs slightly better than FM for small to medium-size circuits, it performs much worse for large circuits—for an explanation of this phenomenon the interested reader is referred to [10]. However, CLIP-LA3 now performs much better than FM (by 24.5% for medium-size benchmarks and 46.5% for large-size benchmarks, for an overall improvement of 34.3%), as does CLIP-FM (by 8.6% and 19.7% for medium and large size circuits, respectively). As is clearly evident, the most improvements of the new schemes over FM are obtained for large circuits. The largest improvement of CLIP-LA3 over FM is about 70% for the circuit *s38417*. This clearly demonstrates the ability of the new cluster-oriented schemes to tackle large circuits.

The cluster detection method CDIP-LA3, obtained by overlaying the CDIP scheme of Figure 6 on LA3, performs even better. The minimum and average cutsizes are improved over those of FM by 35.3% and 45.5%, respectively. Both the minimum and average cutsizes are also superior to those of CLIP-LA3. This indicates that CDIP is a better and more stable partitioner than CLIP.

4.2 Empirical Validation of Theorem 1

A closer look at cell moves reveals the differences between the original algorithms and the new cluster-oriented schemes. Figure 7 shows the cut improvement as we move cells from V_1 to V_2 (not including cell moves from V_2 to V_1) for the first 3 passes in the partitioning of circuit *industry2* under the 50%-50% balance constraint. A figure showing the total cut improvement from moves in both directions is similar. In the initial random partition, most clusters straddle the cutline (see Section 3.1) and give a huge cutsize. In the first pass, the original LA3 algorithm, from its short-sighted local improvement nature, first makes those moves that immediately reduce the cutsize and quickly reaches the maximum improvement point. On the other hand, CLIP-LA3 focuses on pulling out clusters, and reaches its maximum improvement point later, getting a better cutsize reduction. Later passes are even more interesting. LA3 again quickly exhausts the moves that have immediate benefits. For CLIP-LA3, on the other hand, the cutsize first increases (negative improvement in the diagram) and then decreases. These CLIP-LA3 curves for passes 2 and 3 are inverted

Test Case	Minimum of 20 runs								Average of 20 runs							
	Cut Size					% Improvement over FM			Cut Size					% Improvement over FM		
	FM	LA3	CLIP -FM	CLIP -LA3	CDIP -LA3	CLIP -FM	CLIP -LA3	CDIP -LA3	FM	LA3	CLIP -FM	CLIP -LA3	CDIP -LA3	CLIP -FM	CLIP -LA3	CDIP -LA3
s1423	17	16	15	16	15	11.8	5.9	11.8	24.2	22.3	21.8	19.6	20.6	9.9	18.8	14.7
sioo	31	25	37	25	25	-16.2	19.4	19.4	47.5	25.4	49.9	25.3	25.1	-4.8	46.7	47.1
s1488	48	42	43	42	41	10.4	12.5	14.6	53.5	49.1	46.9	45.5	46.4	12.3	14.8	13.2
balu	27	27	27	27	27	0.0	0.0	0.0	38.1	36.1	38.9	32.5	33.9	-2.1	14.6	11.2
p1	47	52	52	52	52	-9.6	-9.6	-9.6	74.9	68.5	65.8	61.8	61.5	12.1	17.6	17.9
bm1	54	53	49	52	52	9.3	3.7	3.7	79.5	67.5	65.0	60.4	59.8	18.3	24.1	24.8
t4	87	82	56	51	52	35.6	41.4	40.2	129.3	117.2	77.0	72.9	71.7	40.4	43.6	44.6
t3	75	80	57	57	57	24.0	24.0	24.0	106.8	106.2	72.3	71.6	66.8	32.2	32.9	37.4
t2	149	126	89	92	90	40.3	38.3	39.6	182.1	148.1	105.2	106.8	101.0	42.3	41.4	44.5
t6	67	70	60	60	60	10.4	10.4	10.4	94.2	84.4	70.1	71.8	70.0	25.5	23.7	25.6
struct	46	44	37	33	36	19.6	28.3	21.7	58.0	49.6	45.6	46.8	46.8	21.3	19.3	19.4
t5	127	99	75	80	74	40.9	37.0	41.7	183.6	165.0	89.0	90.7	89.8	51.5	50.6	51.1
19ks	140	130	119	107	105	15.0	23.6	25.0	171.7	169.0	150.3	136.1	134.6	12.4	20.7	21.6
p2	212	149	149	142	152	29.7	33.0	28.3	273.9	233.4	233.2	208.8	193.2	14.8	23.8	29.5
s9234	59	43	49	47	44	16.9	20.3	25.4	84.7	81.1	89.5	80.8	77.2	-5.4	4.5	8.9
biomed	83	90	84	84	83	-1.2	-1.2	0.0	117.4	170.7	108.4	102.1	102.2	7.7	13.0	13.0
s13207	98	85	98	68	70	0.0	30.6	28.6	122.6	118.9	123.5	100.5	93.9	-0.8	18.0	23.4
s15850	109	87	80	73	67	26.6	33.0	38.5	176.9	140.4	140.9	106.7	107.0	20.3	39.7	39.5
industry2	264	422	260	205	190	1.5	22.3	28.0	627.5	732.4	369.6	332.8	293.6	41.1	47.0	53.2
Subtotal	1740	1722	1436	1313	1292	17.5	24.5	25.7	2646	2585	1963	1773	1694	25.8	33.0	36.0
industry3	272	504	261	261	243	4.0	4.0	10.7	506.0	758.0	376.6	421.6	358.6	25.6	16.7	29.1
s35932	85	168	102	73	73	-16.7	14.1	14.1	210.3	231.4	127.1	79.0	83.0	39.6	62.4	60.5
s38584	100	85	49	55	47	51.0	45.0	53.0	299.8	271.2	83.2	103.3	95.7	72.2	65.5	68.1
avq.small	347	608	223	146	148	35.7	57.9	57.3	578.6	815.5	335.1	309.9	311.7	42.1	46.5	46.1
s38417	240	284	78	73	79	67.5	69.6	67.1	384.1	408.2	136.7	114.0	108.7	64.4	70.3	71.7
avq.large	350	398	216	138	145	38.3	60.6	58.6	755.0	693.4	305.2	349.1	280.8	59.6	53.8	62.8
Subtotal	1394	2047	929	746	735	33.4	46.5	47.3	2734	3178	1363	1377	1238	49.1	48.6	53.8
Total	3134	3769	2365	2059	2027	24.5	34.3	35.3	5380	5763	1963	1773	1694	38.2	41.4	45.5

Table 2: Comparisons of CLIP and CDIP (applied to FM and LA3) to FM. CLIP-LA3 results are for $\delta = 50$. Subtotals shown correspond first to medium-size circuits and then to large circuits.

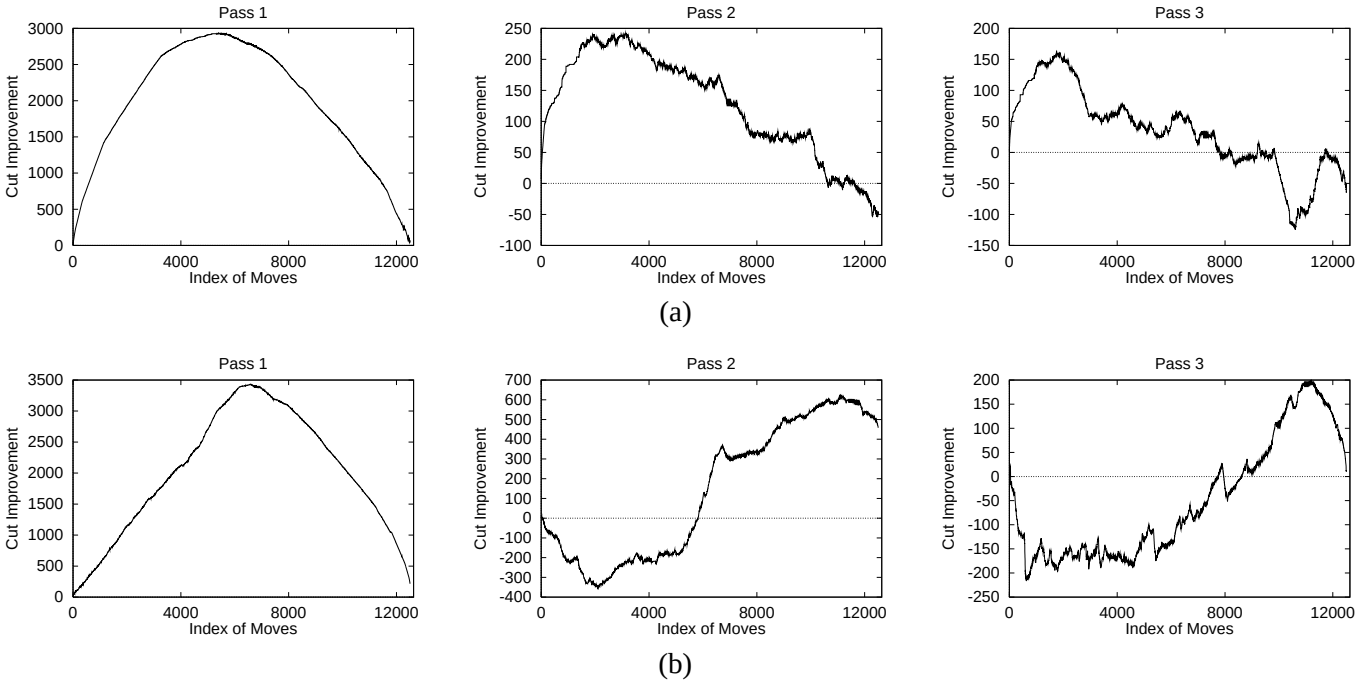

Figure 7: Cut improvement from moves $V_1 \rightarrow V_2$. (a) LA3. (b) CLIP-LA3.

versions of Figure 5, which shows total cutsize rather than cutsize improvement. They are thus consistent with the theoretical curve of Figure 5 and with Theorem 1, and most probably indicate the movements of clusters from one subset to the other in which the cutsizes first increase and then decrease. The minimum cutsize point comes at a much later stage for CLIP-LA3 compared to LA3. This demonstrates the ability of the new schemes to rearrange clusters on a global scale. As a result, LA3 gives a final cut of 986, while CLIP-LA3 gives 362 (note that these are the results of only one run, not the minimum of 20 runs).

4.3 Comparisons to Paraboli and MELO

In Tables 3 and 4, we compare the original IIP and the new cluster-oriented IIP algorithms to two state-of-the-art non-IIP methods, the quadratic-programming based algorithm Paraboli [20] and the spectral partitioner MELO [3]. Here, all the new cluster-oriented IIP algorithms (i.e., CLIP/CDIP schemes) are further improved by the corresponding original schemes as indicated at the end of Section 3.4 (e.g., CLIP-FM$_f$ is CLIP-FM followed by FM improvement, CLIP-PROP$_f$ is CLIP-PROP followed by PROP improvement, etc.). Since FM is very fast, we perform 100 runs of both FM and CLIP-FM$_f$; all other IIP algorithms' results are for 20 runs.

First, it is clear from the tables that the original FM algorithm can obtain good results for medium-size circuits. For this set of benchmarks, it is about 4% better than MELO, and 1% better than Paraboli in total cut. However, for large size circuits, it falls far behind Paraboli (by -22.5% in total cut). This confirms our

Test Case	Cut Size						% Improvement over Paraboli				
	Paraboli	FM	CLIP-FM$_f$	CLIP-LA3$_f$	CDIP-LA3$_f$	CLIP-PROP$_f$	FM	CLIP-FM$_f$	CLIP-LA3$_f$	CDIP-LA3$_f$	CLIP-PROP$_f$
s1423	16	17	15	15	15	15	-5.9	6.2	6.2	6.2	6.2
sioo	45	25	25	25	25	25	44.4	44.4	44.4	44.4	44.4
s1488	50	46	43	42	41	43	8.0	14.0	16.0	18.0	14.0
balu	41	27	27	27	27	27	34.1	34.1	34.1	34.1	34.1
p1	53	47	47	51	47	51	11.3	11.3	3.8	11.3	3.8
struct	40	41	33	33	36	33	-2.4	17.5	17.5	10.0	17.5
p2	146	182	148	142	151	152	-19.8	-1.4	2.7	-3.3	-3.9
s9234	74	51	44	45	44	42	31.1	40.5	39.2	40.5	43.2
biomed	135	83	83	83	83	84	38.5	38.5	38.5	38.5	37.8
s13207	91	78	76	66	69	71	14.3	16.5	27.5	24.2	22.0
s15850	91	104	75	71	59	56	-12.5	17.6	22.0	35.2	38.5
industry2	193	264	174	200	182	192	-26.9	9.8	-3.5	5.7	0.5
Subtotal	975	965	790	800	779	791	1.0	19.0	17.9	20.1	18.9
industry3	267	263	241	260	243	243	1.5	9.7	2.6	9.0	9.0
s35932	62	85	83	73	73	42	-27.1	-25.3	-15.1	-15.1	32.3
s38584	55	63	47	50	47	51	-12.7	14.5	9.1	14.5	7.3
avq.small	224	297	200	129	139	144	-24.6	10.7	42.4	37.9	35.7
s38417	49	147	66	70	74	65	-66.7	-25.8	-30.0	-33.8	-24.6
avq.large	139	350	185	127	137	143	-60.3	-24.9	8.6	1.4	-2.8
Subtotal	796	1205	822	709	713	688	-33.9	-3.2	10.9	10.4	13.6
Total cut	1771	2170	1612	1509	1492	1479	-22.5	8.8	14.8	15.8	17.5

Table 3: Comparisons of various IIP algorithms to Paraboli [20]. The results for the cluster-oriented IIP algorithms (i.e., CLIP/CDIP schemes) in the table have been further improved by their corresponding original IIP algorithms (indicated by the subscript $_f$). FM and CLIP-FM$_f$ results are the best cutsizes from 100 runs, while results for CLIP-LA3$_f$, CDIP-LA3$_f$ and CLIP-PROP$_f$ are the best cutsizes from 20 runs. CDIP-LA3$_f$ results are for $\delta = 50$. Subtotals shown correspond first to medium-size circuits and then to large circuits.

Test Case	Cut Size						% Improvement over MELO				
	MELO	FM	CLIP-FM$_f$	CLIP-LA3$_f$	CDIP-LA3$_f$	CLIP-PROP$_f$	FM	CLIP-FM$_f$	CLIP-LA3$_f$	CDIP-LA3$_f$	CLIP-PROP$_f$
balu	28	27	27	27	27	27	3.6	3.6	3.6	3.6	3.6
p1	64	47	47	51	47	51	26.6	26.6	20.3	26.6	20.3
bm1	48	49	47	51	47	47	-2.0	2.1	-5.9	2.1	2.1
t4	61	80	53	49	48	52	-23.8	13.1	19.7	21.3	14.8
t3	60	62	56	56	57	57	-3.2	6.7	6.7	5.0	5.0
t2	106	124	87	92	89	87	-14.5	17.9	13.2	16.0	17.9
t6	90	60	60	60	60	60	33.3	33.3	33.3	33.3	33.3
struct	38	41	33	33	36	33	-7.3	13.2	13.2	5.3	13.2
t5	102	104	74	80	74	77	-1.9	27.5	21.6	27.5	24.5
19ks	119	130	109	104	104	104	-8.5	8.4	12.6	12.6	12.6
p2	169	182	148	142	151	152	-7.1	12.4	16.0	10.7	10.1
s9234	79	51	44	45	44	42	35.4	44.3	43.0	44.3	46.8
biomed	115	83	83	83	83	84	27.8	27.8	27.8	27.8	27.0
s13207	104	78	76	66	69	71	25.0	26.9	36.5	33.7	31.7
s15850	52	104	75	71	59	56	-50.0	-30.7	-26.8	-11.9	-7.1
industry2	319	264	174	200	182	192	17.2	45.5	37.3	42.9	39.8
Total cut	1554	1486	1193	1210	1177	1192	4.4	23.2	22.1	24.3	23.3

Table 4: Comparisons of various IIP algorithms to MELO [20]. The results in the table have been further improved by the original IIP algorithms (indicated by the subscript $_f$). FM and CLIP-FM$_f$ results are the best cutsizes from 100 runs, while results for CLIP-LA3$_f$, CDIP-LA3$_f$ and CLIP-PROP$_f$ are the best cutsizes from 20 runs. CDIP-LA3$_f$ results are for $\delta = 50$.

Test Case	FM $\times 100$	CLIP-FM$_f$ $\times 100$	CLIP-LA3$_f$ $\times 20$	CDIP-LA3$_f$ $\times 20$	CLIP-PROP$_f$ $\times 20$	MELO	Paraboli
s1423	0.15	0.20	0.54	0.68	0.61		7.7
sioo	0.10	0.11	0.35	0.44	0.77		15.8
s1488	0.19	0.20	0.59	0.74	1.44		17.0
balu	0.25	0.35	1.59	1.55	1.69	7	15.5
p1	0.33	0.44	1.86	2.36	1.81	8	18.3
bm1	0.27	0.43	1.78	2.42	1.84	9	
t4	0.58	0.90	4.07	5.29	5.70	24	
t3	0.76	0.96	4.41	5.35	4.74	27	
t2	0.58	0.99	4.97	6.20	5.46	29	
t6	0.57	0.88	2.54	2.75	8.73	31	
struct	0.54	0.69	2.25	2.72	3.75	38	35.2
t5	1.13	1.62	7.07	8.11	9.39	67	
19ks	1.59	2.23	8.92	10.83	10.96	79	
p2	1.81	2.37	8.33	10.52	17.64	89	137.4
s9234	2.78	3.14	8.74	13.48	13.22	516	490.3
biomed	3.89	3.34	11.56	18.13	28.61	496	710.9
s13207	4.23	5.01	11.03	21.47	19.02	710	2060.4
s15850	4.24	6.87	13.33	27.13	28.80	1197	2730.9
industry2	9.10	13.17	56.47	72.64	106.33	1855	1367.3
industry3	11.07	16.35	70.93	97.18	96.01		760.7
s35932	11.82	13.21	23.16	48.19	54.26		2626.7
s38584	13.70	15.73	37.38	66.93	97.48		6517.5
avq.small	18.44	20.50	63.01	125.35	104.08		4098.9
s38417	15.36	17.70	40.58	86.65	84.49		2041.5
avq.large	19.49	24.07	71.50	157.26	106.28		4135.0
Total(16 ckts.)	3264	4336	2995	4218	5354	5177	
Total(18 ckts.)	11754	14346	8464	15070	15325		27787

Table 5: Comparisons of CPU times of various algorithms in secs per run, and total times over all circuits and all runs made. MELO was run on SUN SPARC10, Paraboli on DEC3000 Model 500 AXP, all others on SUN SPARC5 Model 85.

earlier discussion on the shortcomings of previous IIP methods. After using the cluster-oriented techniques, CLIP and CDIP, all of the four new algorithms CLIP-FM$_f$, CLIP-LA3$_f$, CLIP-PROP$_f$ and CDIP-LA3$_f$, are able to obtain cutsizes that are overall better than Paraboli's. Total cut improvements range from 8.8% to 17.5%. This demonstrates that IIP algorithms can also partition large-size circuits very effectively.

The best results are obtained by applying CLIP to PROP, a probability-based IIP partitioner [8]. PROP calculates cell gains based on the probability of a cell being actually moved in a pass. Thus the cell gain calculation is more accurate than that of either FM or LA. Further, when CLIP is applied to FM and LA, neighbors of moved nodes get updated only if they are connected to them by critical or somewhat critical (for LA) nets. On the other hand, in PROP, all nets are probabilistically critical, and thus all neighbors get updated, leading to a more accurate reflection of a cell's move on them. The PROP algorithm combined with CLIP overcomes the two fundamental shortcomings in traditional IIP methods—inaccurate gain calculation and lack of a global view of the cluster-oriented structure of circuits. Thus it emerges as a very powerful partitioning tool and performs about 17% better than Paraboli. When compared to MELO for which only results on medium-size circuits are given in [3], the various CLIP/CDIP algorithms show similar improvements of about 23% in total cutsize over MELO.

4.4 Run Time Comparisons

The run times of the IIP algorithms are very favorable compared to the non-IIP algorithms Paraboli and MELO (Table 5). Run times of Paraboli and MELO are reported for the DEC3000 Model 500 AXP and SUN SPARC10 workstations, respectively, while we have executed all CLIP and CDIP based algorithms, as well as FM and LA, on the SUN SPARC5 Model 85 workstation[5]. The data structures used to store free cells for FM and LA are bucket structures as proposed in [11] and [15], respectively. Despite the $O(max(c\delta d, cn, p))$ worst-case time complexity of CDIP-LA3$_f$, in practice it uses less than twice the CPU time of CLIP-LA3$_f$, which has a linear run time. PROP uses a tree structure which makes it easy to accommodate arbitrary net weight as is required in performance-driven partitioning [19]. Yet the run time of CLIP-PROP$_f$ is quite reasonable. The total CPU times of all four new algorithms are less than that of Paraboli. Even assuming the same speed for the three different workstations, CLIP-FM$_f$ is 1.9 times faster, CLIP-LA3$_f$ is 3.3 times faster, and both CDIP-LA3$_f$ and CLIP-PROP$_f$ are 1.8 times faster than Paraboli. CLIP-PROP$_f$ is comparable with MELO in run time, while CLIP-FM$_f$, CLIP-LA3$_f$ and CDIP-LA3$_f$ are faster than MELO by factors of 1.2, 1.7 and 1.2, respectively. This also means that if equal CPU times are allocated, the new algorithms can

[5] SPECmarks of these machines show that the SPARC10 and SPARC5 are comparable in speed, while the DEC3000 is about 1.5 times faster than the SPARC5.

Test Case	# Cells	# Nets	# Pins
test0	2741	2521	8292
test1	11471	11390	33894
test2	12146	10508	28906
test3	20392	21703	67290
test4	25991	27970	87881
test5	35549	44004	172056
test6	57093	49186	185855
test7	117440	101696	375660
test8	161951	152627	460779

Table 6: Characteristics of various industrial circuits tested at Cadence Design Systems.

perform more runs and generate even better cutsizes than those presented in Tables 3 and 4.

In summary, the new cluster-oriented IIP algorithms CLIP and CDIP are very promising partitioning tools. They integrate cluster awareness with traditional IIP schemes, and generate much better mincut results than state-of-the-art non-IIP methods such as Paraboli and MELO. The superior performance of this class of algorithms makes them very attractive for various VLSI CAD applications.

4.5 New Industrial Circuits and Comparisons to hMetis

As mentioned in Sec. 1, the flatness of CLIP/CDIP can be a major advantage for complex placement problems vis-a-vis multilevel techniques [1, 2, 5, 16]. Further, results for the ACM/SIGDA benchmark suite reported in the above works show that under the unit node-size assumption (as stated in Sec. 1, [5] does not make this assumption and its results thus are not comparable to those of other partitioners), CLIP/CDIP mincut results are within just 3-6% of those of the multilevel algorithms. However, in this section, we still compare CLIP/CDIP to one of the best multilevel partitioners hMetis [16] for a number of medium to very-large industrial circuits whose characteristics are given in Table 6. All experiments reported in this section were run on the 167MHz Ultra1 Sparc workstation. We first establish in Table 7 that the industrial circuits present challenging partitioning problems, and then give the results of running CLIP/CDIP and hMetis on them in Tables 8 and 9, respectively.

Table 7 compares the cutsizes and times-to-solution for two classic IIP techniques FM and LA to that the newer IIP algorithm PROP [8]. In [8], it was shown to obtain large improvements over FM and LA for the ACM/SIGDA benchmark circuits of Table 1. As shown in Table 7, large improvements in both mincut and average-cut over all circuits are again obtained by PROP-20 over FM-40 (52.7% and 61% improvements, respectively) and LA2-40 (69.5% and 77.7% improvements, respectively), while it is only 1.64 times slower than FM-40 and slightly faster (by 12%) than LA2-40. The maximum improvement obtained by PROP-20

Test Case	FM 40-runs			LA2 40-runs			PROP 20-runs			PROP 40-runs		
	Min	**Avg**	**Time**	**Min**	**Avg**	**Time**	**Min**	**Avg**	**Time**	**Min**	**Avg**	**Time**
test0	50	93.8	23.1	42	93.2	34.1	24	32.0	62.2	24	31.9	114.0
test1	62	245.9	155.2	62	223.0	214.3	55	93.5	196.2	55	88.5	374.7
test2	133	234.7	152.3	138	247.6	192.2	144	184.3	241.2	135	188.8	461.9
test3	387	947.3	260.6	313	1196.1	365.1	245	462.3	728.4	200	423.7	1537.3
test4	514	886.0	365.5	698	1229.0	527.0	220	319.4	1177.6	220	319.0	2106.1
test5	844	1798.1	848.1	1508	3329.9	1399.6	361	768.0	2312.2	361	653.6	4422.9
test6	85	206.2	904.2	229	819.3	1547.5	85	247.2	1525.8	85	266.3	3120.9
test7	95	183.0	2345.0	417	1167.0	3907.8	79	273.6	3230.4	79	291.3	6195.8
test8	433	1593.2	2321.9	634	2477.8	5371.1	19	30.9	2650.4	18	27.9	5781.3
Total	2603	6188.2	7375.9	4041	10782.3	13558.7	1232	2411.2	12124.4	1178	2289.0	24114.9
% Impr. of PROP-20	52.7	61	-64	69.5	77.7	12	-	-	-	-4.6	-5.3	49.7
% Impr. of PROP-40	54.8	63	-227	70.8	78.8	-78	4.4	5.1	-99	-	-	-

Table 7: Comparisons of cutsizes for the 45-55% balance criterion produced by FM, LA and PROP for the industrial circuits of Table 6. The **Min** column is the best result from all the runs, the **Avg** column shows the average cutsize over those runs, and the **Time** column is the total CPU time in seconds over all the runs.

over FM-40 is for the largest circuit `test8` for which it produces a mincut that is 95.61% better than that of FM-40 (a FM-40 mincut of 433 reduces to 19) and obtains a 98% improvement in average-cut (an average cut of 1593.2 produced by FM-40 is reduced to 30.9 by PROP-20)! Over all circuits, PROP-40 improves slightly over PROP-20, by 4.4% in mincut and 5.1% in average cut. Table 7 establishes two distinct levels of results (by FM and PROP) to which hMetis and CLIP/CDIP can be compared, and because of the large gap in results produced by FM/LA and PROP, it also shows that the industrial circuits are indeed interesting and non-trivial circuits to partition.

Table 8 shows the cutsizes and times for various CLIP/CDIP versions obtained for the industrial circuits. The salient points of the results are

- The results produced by CLIP-LA2$_f$, CLIP-PROP$_f$ and CDIP-LA2$_f$ are comparable to each other and about 7.7% and 13.6% better than CLIP-FM$_f$ in mincut and average-cut, respectively.
- CLIP-LA2$_f$, CLIP-PROP$_f$ and CDIP-LA2$_f$ improve in mincut and average-cut over FM-40 by about 57% and 73%, respectively, and over PROP-20 by about 11% and 30%, respectively. For the circuit `test2` these CLIP/CDIP partitioners produce mincut improvements of 23-40% and for the second-largest circuit `test7` improvements of 18% over PROP-20/40 runs.
- Ignoring the results for `test3`, CLIP-PROP$_f$ produces as good or better mincuts than CLIP-LA2$_f$. CDIP-LA2$_f$ produces the best mincut results.
- The times increase by factors of 5 and 1.5 as we go from CLIP-LA2$_f$ to CLIP-PROP$_f$ to CDIP-LA2$_f$.
- The CLIP/CDIP algorithms produce substantially better results than FM as well as PROP.

Test Case	CLIP-FM$_f$ 20-runs			CLIP-LA2$_f$ 20-runs			CLIP-PROP$_f$ 20-runs			CDIP-LA2$_f$ 20-runs		
	Min	**Avg**	**Time**	**Min**	**Avg**	**Time**	**Min**	**Avg**	**Time**	**Min**	**Avg**	**Time**
test0	27	37.4	14.6	24	34.0	19.8	24	32.0	74.8	25	31.2	29.6
test1	53	133.6	104.6	53	90.0	115.8	53	82.0	459.0	53	93.7	228.4
test2	158	220.9	117.6	103	159.5	148.0	100	164.0	365.2	86	150.3	317.8
test3	179	354.8	222.6	177	331.6	326.0	199	317.6	2260.6	177	316.9	1060.0
test4	220	327.2	253.6	220	330.1	328.4	206	313.1	1887.8	209	310.5	1048.4
test5	360	583.6	365.8	360	487.6	547.8	360	628.6	4224.4	360	553.4	4207.0
test6	85	92.7	303.0	85	91.7	458.2	85	113.5	2519.4	85	87.8	1906.2
test7	97	117.6	788.8	65	91.0	772.0	65	74.6	4765.2	65	91.6	4848.6
test8	17	32.8	1149.0	17	26.4	1493.8	17	19.2	5272.0	17	25.8	18370.6
Total	1196	1900.6	3319.6	1104	1641.9	4209.8	1109	1744.6	21828.4	1077	1661.2	32016.6
% Impr. over FM-40	54.1	69.3	55	57.6	73.5	42.9	57.4	71.8	-196	58.6	73.2	-334
% Impr. over PROP-20	2.9	21.2	72.6	10.4	31.9	65.3	10.0	27.7	-80	12.6	31.1	-164
Total for test6-test8	199	243.1	2240.8	167	209.1	2724.0	167	207.3	12556.6	167	205.2	25125.4

Table 8: Cutset results for the 45-55% balance criterion for various CLIP/CDIP versions for the industrial circuits of Table 6. The **Min** column is the best result from all the runs, the **Avg** column shows the average cutsize over those runs, and the **Time** column is the total CPU time in seconds over all the runs.

Test Case	hMetis1 20-runs			hMetis2 20-runs			hMetis3 20-runs		
	Min	**Avg**	**Time**	**Min**	**Avg**	**Time**	**Min**	**Avg**	**Time**
test0	26	28.80	9.03	24	27.55	11.55	25	30.50	13.84
test1	54	54.45	55.19	54	55.40	128.14	54	57.90	177.33
test2	77	88.20	70.65	76	83.00	120.69	81	93.25	190.79
test3	171	256.30	177.41	167	203.90	309.60	167	186.50	570.71
test4	204	224.45	214.17	204	216.75	456.09	209	234.40	799.79
test5	442	670.20	217.72	360	364.00	411.84	360	412.80	1128.60
test6	85	140.35	582.95	85	86.90	1972.40	86	104.35	2756.08
test7	85	124.25	2087.07	65	121.30	8106.00	65	110.95	11610.07
test8	19	30.95	4999.00	19	31.65	15489.00	33	46.90	32201.00
Total	1163	1617.95	8413.19	1054	1190.45	27005.32	1080	1277.5	49448.21
Comp. to CLIP-LA2$_f$	-5.3%	1.5%	-100%	4.5%	27.5%	-541%	2.2%	22.2%	-1075%
Comp. to CDIP-LA2$_f$	-8%	2.6%	73.7%	2.1%	28.3%	15.6%	-0.3%	23.1%	-54%
Total for test6-test8	189	295.55	7669.02	169	239.85	25567.4	184	262.20	46567.15
Comp. to CLIP-LA2$_f$ for test6-test8	-13.2%	-41.3%	-182%	-1.2%	-14.7%	-840%	-10.2%	-25.4%	-1610%
Comp. to CDIP-LA2$_f$ for test6-test8	-13.2%	-44%	39%	-1.2%	-16.9%	-0.02%	-10.2%	-27.8%	-85%

Table 9: Cutset results for three hMetis schemes (45-55% balance criterion) and comparison to CLIP/CDIP for the industrial circuits. A negative % in the comparison rows means a deterioration for the hMetis scheme in the metric concerned, i.e., an increase in mincut, average cut and time, while a positive % means an improvement in these metrics. hMetis$x$ refers to a run of hMetis with "Ctype" parameter equal to x, $x = 1, 2, 3$. The Ctype parameter value refers to a particular clustering/coarsening heuristic that are as follows for the three different values. **1**: Nodes are grouped if they are present in multiple hyperedges and the grouping is biased in favor of faster reduction in the number of the hyperedges that remain in the coarse hypergraphs. **2**: Nodes are grouped together if they are present in multiple hyperedges. **3**: Nodes are grouped together that correspond to entire hyperedges. Preference is given to hyperedges that have large weight.

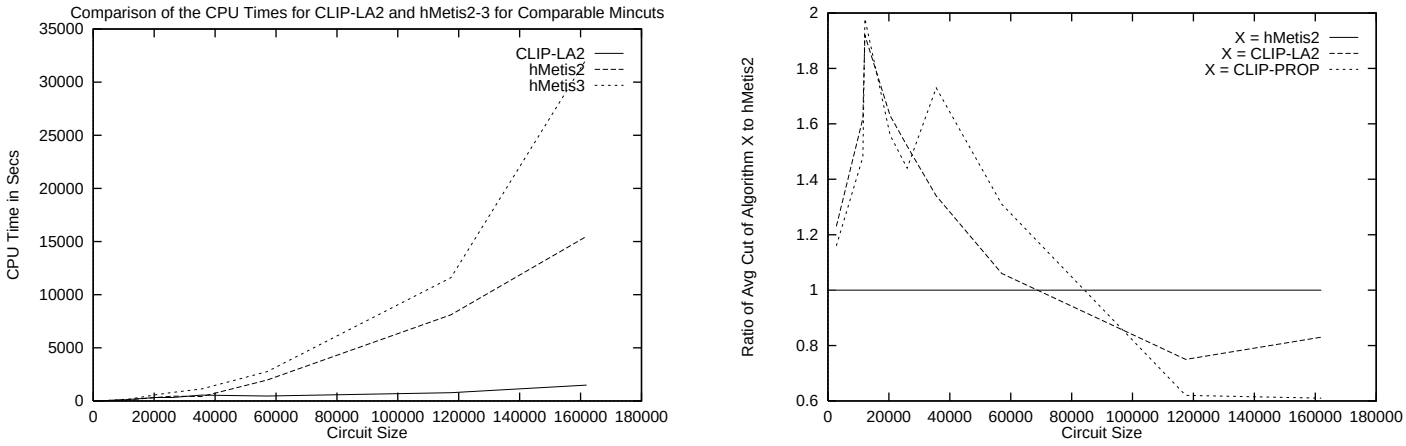

Figure 8: Scalability plots for CLIP: (a) Comparison of the CPU times for CLIP-LA2$_f$ and hMetis2 & 3 versus industrial circuit size (# of nodes)—CLIP-LA2$_f$ and hMetis2 & 3 produce comparable mincuts (Table 9). (b) Ratio of average cuts obtained by CLIP-LA2$_f$ and CLIP-PROP$_f$ to those obtained by hMetis2 versus circuit size—these three partitioners produce comparable mincuts and the CLIP algorithms are faster than hMetis2.

Table 9 shows the results of three hMetis schemes (described in the caption of the table) on the industrial circuits. As shown in the two comparison rows corresponding to all circuits, hMetis1 produces 8% worse mincuts than CDIP-LA2$_f$, while the mincut results of hMetis2 and hMetis3 are almost the same as those obtained by CLIP-LA2$_f$ and CDIP-LA2$_f$. However, the hMetis results are obtained at the expense of much larger times-to-solution. For example, hMetis1 is twice as slow, hMetis2 is 6.5 times slower and hMetis3 is 11.75 times slower than CLIP-LA2$_f$, while the mincut results are virtually identical. Compared to CDIP-LA2$_f$, hMetis1 is 73.7% faster but has 8% worse mincut, while hMetis3 has almost identical mincut but is 54% slower. hMetis2 and hMetis3 have better average-cuts over all circuits than CLIP-LA2$_f$ and CDIP-LA2$_f$ by about 27% and 22%, respectively, but as mentioned above, at the expense of enormously more computation times, especially compared to CLIP-LA2$_f$. However, it is very significant to note that all three hMetis schemes have much worse average cuts for the largest three circuits (`test6-test8`) than CLIP-LA2$_f$ and CDIP-LA2$_f$ by 15-41% and 17-44%, respectively—see last two rows of Table 9. The greater computation time for hMetis is also more pronounced for large circuits. These results clearly indicate that CLIP-LA2$_f$ is much more scalable than hMetis with increasing circuit sizes both in terms of computation time and solution quality. This is also portrayed in Fig. 8, which shows: (a) The computation times of hMetis2 and hMetis3 increase at much higher rates with circuit size than that of CLIP-LA2$_f$; and (b) The average cuts of CLIP-LA2$_f$ and CLIP-PROP$_f$ improve with circuit size after a certain threshold compared to that of hMetis2 (it has the best average cuts of all hMetis schemes) until they are much better than hMetis2 for the largest circuits.

In summary, CLIP/CDIP mincuts are very close to those obtained by hMetis and are produced much faster; the only advantage of hMetis is better average cuts for medium-size circuits (but worse average cuts for the three largest circuits), though it is not clear how this advantage can be translated to applications like placement, where mincut is the primary metric. Finally, as mentioned earlier, flat partitioners like CLIP/CDIP are more appropriate in complex placement systems. The fact that our flat partitioners produce comparable mincut results to those of multilevel partitioners at higher speeds augurs additionally well for their use in such placement modules for large VLSI circuits.

5 Conclusions

We proposed new cluster-oriented partitioning approaches CLIP and CDIP that can be combined with fast IIP methods to greatly enhance their performance. Our new algorithms generate much better mincut results than classic IIP techniques such as FM and LA, and state-of-the-art non-IIP methods such as Paraboli and MELO. They also obtain comparable mincuts but at much higher speeds than state-of-the-art multilevel IIP algorithms such as hMetis.

In spite of being implicitly cognizant of natural circuit clusters, neither CLIP nor CDIP requires nodes to be clustered. These algorithms are thus flat partioners that are cluster-aware. As evident by the appearance of several new partitioning algorithms such as hMetis [16], ML_c [1] and LSR/MFFS [5], the multilevel paradigm that applies a sequence of clustering followed by unclustering and partitioning phases to circuits is becoming quite popular. This is indeed a good trend, since it is a powerful paradigm that can produce high-quality results especially when the desired optimization metric is purely mincut. For complex placement problems like timing-driven placement [6, 19] in the presence of multiple constraints [7, 17], however, it may be detrimental to hide useful information about the circuit by clustering subcircuits into large meta-nodes. Thus flat partitioners like CLIP/CDIP (or their timing-driven variations, for example) are more suited for such placement problems. Further, for the ACM/SIGDA benchmark suite, hMetis, one of the best and fastest of the multilevel genre, produces results that are only about 5% better [16] than those of CLIP/CDIP. We also compared CLIP/CDIP to hMetis for some medium to very large industrial circuits, and found that one hMetis scheme is 8% worse while two others are only about 2-4% better than CLIP/CDIP. Significantly, the experiments also showed that one of our best algorithms CLIP-$LA2_f$ is an order of magnitude faster, and is much more scalable for large circuits in both computation time and solution quality (better mincuts and average cuts for large circuits) than all hMetis schemes. Along with the fact that CLIP/CDIP are flat partitioners, these results show that they are very attractive techniques for use in complex placement problems

for emerging large VLSI circuits, and also in many VLSI CAD applications that apply divide-and-conquer to solve large problems.

References

[1] C.J. Alpert, J-H. Huang and A.B. Kahng, "Multilevel circuit partitioning", *Proc. Design Automation Conf.*, June 1997, pp. 530-533.

[2] C.J. Alpert and A.B. Kahng, "A hybrid multilevel/genetic approach for circuit partitioning", *Physical Design Workshop*, 1996, pp. 100-105.

[3] C. J. Alpert and S-Z Yao, "Spectral Partitioning: The More Eigenvectors, the Better", *Proc. ACM/IEEE Design Automation Conf.*, 1995.

[4] C. J. Alpert and A. B. Kahng, "A General Framework for Vertex Orderings, With Applications to Netlist Clustering", *Proc. IEEE Intl. Conf. Computer-Aided Design*, 1994, pp. 63-67.

[5] J. Cong, et al., "Large scale circuit partitioning with loose/stable net removal and signal flow based clustering", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, Nov. 1997, pp. 441-446.

[6] S. Dutt, "A Stochastic Approach to Timing-Driven Partitioning and Placement with Accurate Net and Gain Modeling", *TAU97: IEEE/ACM Int. Workshop on Timing Issues in Digital Systems*, Dec. 1997, pp. 246-256.

[7] S. Dutt and H. Theny, "Partitioning Around Roadblocks: Tackling Constraints with Intermediate Relaxations", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, IEEE/ACM ICCAD, Nov., 1997, pp. 349-355.

[8] S. Dutt and W. Deng, "A probability-based approach to VLSI circuit partitioning", *Proc. Design Automation Conf.*, June 1996, pp. 100-105.

[9] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, Nov. 1996, pp. 92-99.

[10] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques", Technical Report, Department of Electrical Engr., University of Minnesota, Nov. 1995.
This report is available at ftp site ftp-mount.ee.umn.edu in file pap-pdw96-ext.ps in directory /pub/faculty/dutt/vlsi-cad/papers.

[11] C.M. Fidducia and R.M. Mattheyses, "A linear-time heuristic for improving network partitions", *Proc. Nineteenth Design Automation Conf.*, 1982, pp. 175-181.

[12] L. Hagen and A. B. Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering", *Proc. IEEE Intl. Conf. Computer-Aided Design*, 1991, pp. 10-13.

[13] S. Hauck and G. Borriello, "An Evaluation of Bipartitioning Techniques", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 8, pp. 849-866, August 1997.

[14] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs", *Bell System Tech. Journal*, vol. 49, February 1970, pp. 291-307.

[15] B. Krishnamurthy, "An improved min-cut algorithm for partitioning VLSI networks", *IEEE Trans. on Comput.*, vol. C-33, May 1984, pp. 438-446.

[16] G. Karypis, et al., "Multilevel hypergraph partitioning: Application in VLSI domain", *Proc. Design Automation Conf.*, June 1997, pp. 526-529.

[17] R. Kuznar, F. Brglez and K. Kozminski, "Cost minimization of partitions into multiple devices", *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 315-320.

[18] J. Li, J. Lillis and C-K. Cheng, "Linear decomposition algorithm for VLSI design applications", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, 1995, pp. 223-228.

[19] M. Marek-Sadowska, “Issues in timing driven layout”, in *Algorithmic Aspects of VLSI Layout*, pp. 1-24, M. Sarrafzadeh and D.T. Lee, eds., World Scientific Publ. Co.

[20] B. M. Riess, K. Doll and F. M. Johannes, “Partitioning Very Large Circuits Using Analytical Placement Techniques”, *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 646-651.

[21] Y. G. Saab, “A Fast and Robust Network Bisection Algorithm”, *IEEE Trans. Computers*, 1995, pp. 903-913.

[22] D. G. Schweikert and B. W. Kernighan, “A Proper Model for the Partitioning of Electrical Circuits”, *Proc. 9th Design automation workshop*, 1972, pp. 57-62.

[23] Y.C. Wei and C.K. Cheng, “Towards efficient hierarchical designs by ratio cut partitioning”, *Proc. Int’l. Conf. Computer-Aided Design*, 1989, pp. 298-301.

[24] Y. C. Wei and C. K. Cheng, “An Improved Two-way Partitioning Algorithm with Stable Performance”, *IEEE Trans. on Computer-Aided Design*, 1990, pp. 1502-1511.