New Algorithms for Min-Cut Replication in Partitioned Circuits*

Hannah Honghua Yang†
Intel Development Labs
Intel Corporation
Hillsboro, OR 97124

D. F. Wong
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712

Abstract

Hwang and El Gamal [HE92, HE95] formulated the min-cut replication problem, which is to determine min-cut replication sets for the components of a k-way partition such that the cut size of the partition is minimized after the replication. They gave an optimal algorithm for finding min-cut replication sets for a k-way partitioned digraph. However, their optimal min-cut replication algorithm does not guarantee min-cut replication sets of minimum sizes. Furthermore, their algorithm is not optimal for hypergraphs. In this paper, we optimally solve the min-area min-cut replication problem on digraphs, which is to find min-cut replication sets with the minimum sizes. More importantly, we give an optimal solution to the hypergraph min-area min-cut replication problem using a much smaller flow network model. We implemented our algorithms in a package called Hyper-MAMC, and interfaced Hyper-MAMC to the TAPIR package [HE95]. On average, Hyper-MAMC produces 57.3% fewer cut nets and runs much faster than MC-Rep in the TAPIR package, on the same initial partitions of a set of MCNC Partition93 benchmark circuits.

1 Introduction

VLSI circuit partitioning [PL88] underlies many important problems in VLSI layout designs, such as floorplanning, placement, and multi-chip/multi-FPGA partitioning. Minimizing the number of edges connecting different components has been an important objective in circuit partitioning. This problem is NP-complete if the sizes of the components are constrained, but effective heuristics [KL70, FM82, KGV83, WC89, HK91, CHK92, YW94] are known for solving this problem.

Recent research results [KN91, HE95, MBSV91, RW93] show that node replication from one component to another can be used to reduce the number of cut edges, wiring density and the critical path delay in a partitioned circuit. The node replication approach is particularly useful for fully utilizing pin-limited devices, such as multiple-FPGAs, and can reduce the number of devices needed to implement a design. Kring and Newton [KN91] extended the FM heuristic [FM82] to allow cells to be replicated in the components. The first graph theoretical treatment of the min-cut replication problem was given by Hwang and El Gamal in [HE92, HE95]. They first formulated the *min-cut replication problem* to be the problem of determining min-cut replication sets for the components of a k-way partition such that the cut size of the partition is minimized after the replication. They gave a network flow based algorithm for finding min-cut replication sets for a k-way partitioned digraph. Their algorithm is optimal only in terms of the number of cut edges. As pointed out in [HE95], their optimal min-cut replication algorithm does not guarantee min-cut replication sets of minimum sizes. Often, many nodes are replicated unnecessarily, which causes the components to exceed their size limits after replication, thus making the min-cut replication sets infeasible[1]. This drawback limits the application of their otherwise elegant optimal min-cut replication algorithm. Hence, it is of both theoretical and practical interests to find the smallest replication sets that achieves the same min-cut size. Hwang and El Gamal [HE95] used a heuristic approach to extend their min-cut replication algorithm to hypergraphs.

More recently, Liu, Kuo, Cheng, and Hu [LKCH95] presented an optimal algorithm for the two-way replication partitioning problem. Their formulation is different from that used in [HE95] in that it is only sensitive to a pair of source-sink nodes, rather than the initial bipartition. However, the result of [LKCH95] is not easily generalized to k-way replication partitioning, does not guarantee the minimum size for the replication sets, and is not optimal for hypergraphs.

In this paper, we give optimal solutions to the min-area min-cut replication problem for both digraphs and hypergraphs. Both solutions rely on a procedure for finding a min-area min-cut in a flow network. The solution for digraphs is based on the same flow network model as used in [HE95]. The solution for hypergraphs requires a new flow network model. The new flow network model not only exactly models a (directed) hypergraph,[2] but also is much smaller than the flow network used in [HE95] due to searching the components

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459, by an Intel Foundation Graduate Fellowship, and by a DAC Design Automation Scholarship.

†This work was done while the author was a Ph.D. student at the University of Texas at Austin.

[1]The experimental results in [HE95] show that the max-flow replication solution was infeasible for many circuits and was replaced by the FM [FM82] based heuristic solution.

[2]This is different from the flow network model given in [YW94]. [YW94] models an undirected hypergraph without distinguishing the source node and the sink nodes of a net.

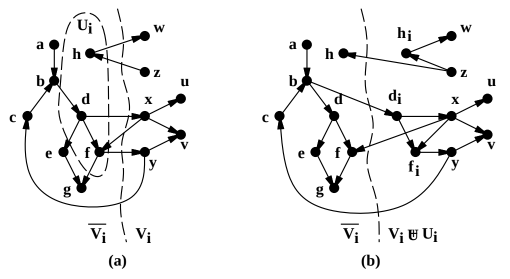


Figure 1: (a) A digraph G, and a partition $\{\overline{V_i}, V_i\}$ with 6 cut edges. (b) After replicating $U_i = \{d, f, h\}$ into V_i, the number of cut edges is reduced to 4.

in the reverse order of [HE95]. Hence we were able to obtain better cut sizes with less runtime.

We applied our optimal algorithm for hypergraphs as a basic procedure to solve the more practical *area and pin constrained* min-cut replication problem, and implemented it in a package called Hyper-MAMC. Experiments were conducted to compared Hyper-MAMC with the MC-Rep implementation in the TAPIR package [HE95] on a set of MCNC Partition93 benchmark circuits, using the same k-way initial partitions. On average, Hyper-MAMC produces 57.3% fewer cut nets, and runs much faster than MC-Rep. For example, for a circuit s35932 of almost 20K gates and 24 initial components, Hyper-MAMC took 3 minutes while MC-Rep took 2.5 hours on a Sparc10 workstation with 32MB of memory.

The rest of the paper is organized as follows. In Section 2, we formally define the min-area min-cut replication problem. Section 3 presents an optimal algorithm for the min-area min-cut replication problem on a k-way partitioned digraph. In Section 4, we give an exact optimal solution to the hypergraph min-area min-cut replication problem. This optimal solution is the basis for solving the more practical area and pin constrained replication problem, which is described in Section 5. We show experimental results in Section 6, and conclude the paper in Section 7.

2 Problem Formulation

We adopt similar notations to those used in [HE95]. Given a digraph $G = (V, E)$, a *k-way partition* $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ is a partition of the node set V into k disjoint subsets called *components*. The set of *cut edges* of the partition is the set $C \subseteq E$ of edges that connect vertices in different components. Let $in(V_i) = \{(u, v) \in E \mid u \notin V_i, v \in V_i\}$ denote the set of all incoming edges to V_i. Then $C = \bigcup_i in(V_i)$, and $|C| = \Sigma_i |in(V_i)|$ since $in(V_i)$ and $in(V_j)$ are disjoint for different i and j in a digraph.

The complement, $V \setminus V_i$, of a component V_i is denoted $\overline{V_i}$. $\overline{V_i} = \bigcup_{j \neq i} V_j$. Given a subset $U_i \subseteq \overline{V_i}$, the *replication of* U_i *into* V_i (see Fig. 1) is obtained by adding to V_i a new node w_i for each node $w \in U_i$. The new node w_i is the *clone* of w in V_i. To ensure functional correctness, the input edges to the clones need to be added (see edges (b, d_i), (x, f_i), (z, h_i) and (d_i, f_i) in Fig. 1 (b)). To reduce the cut size, output edges from an original node in $\overline{V_i}$ to nodes in V_i need to be redirected from its clone which is already in V_i (see edges (d_i, x), (f_i, y) and (h_i, w)).

We use $V_i \biguplus U_i$ to denote the component V_i after replicating U_i into V_i. Observe that, after replicating U_i to V_i, the original cut edges from nodes in U_i to nodes in V_i are deleted, and new cut edges from nodes in $\overline{V_i} \setminus U_i$ to nodes in $V_i \biguplus U_i$ are added. The cut edges directing from V_i to $\overline{V_i}$ are not changed. Hence replicating U_i to V_i affects $in(V_i)$ only, and does not affect $in(\overline{V_i})$. A set U_i is a *min-cut replication set* to V_i if $|in(V_i \biguplus U_i)|$ is minimum among all $U_i \subseteq \overline{V_i}$. Let $w(v)$ denote the area of a node $v \in V$, and let $area(U_i) = \Sigma_{u \in U_i} w(u)$ denote the total area of the nodes in U_i. A min-cut replication set is a *min-area min-cut replication set* if $area(U_i)$ is minimum among all min-cut replication sets U_i to V_i. The following problem formulation was given in [HE95].

The min-cut replication problem: Given a digraph $G = (V, E)$, and a k-way partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$, find a collection of sets of vertices, $\{U_{ij} \mid 1 \leq i, j \leq k\}$, such that $U_{ij} \subseteq V_i$ and after replicating U_{ij} into V_j, the number of cut edges in the new partition is minimum, for $i, j = 1, 2, \ldots, k$.

The authors in [HE95] showed that the min-cut replication problem can be solved by solving k independent min-cut replication problems, one for each V_i, $1 \leq i \leq k$, by observing the fact that replicating any set $U_i \subseteq \overline{V_i}$ into V_i cannot change $in(V_j)$ for $j \neq i$, since $in(\overline{V_i})$ is not changed and $\overline{V_i} = \bigcup_{j \neq i} V_j$. Hence an equivalent statement of the min-cut replication problem is the following.

The equivalent min-cut replication problem: Given a digraph $G = (V, E)$, and a k-way partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$, find min-cut replication sets $U_i \subseteq \overline{V_i}$ that minimizes $|in(V_i \biguplus U_i)|$, for $i = 1, 2, \ldots, k$.

Although this formulation ignores size constraints in the partition components after the replication, it still has practical applications when the size constraints are sufficiently loose. As we will see later in Section 5, there are heuristics that reduce the size of a replication set without increasing the cut size significantly. Nevertheless, the effect of the size of a replication set can be significant when the components in the original partition are close to their capacity. Hence, it is of both theoretical and practical interests to find a smallest replication set that achieves the same min-cut size. The authors of [HE95] did not solve the problem of finding a smallest min-cut replication set.

The min-area min-cut replication problem: Find a solution $\{U_i \mid 1 \leq i \leq k\}$ to the equivalent min-cut replication problem such that the area of the replicated nodes, $\Sigma_i area(U_i)$, is minimum.

3 Optimal Min-Area Min-Cut Replication for Digraphs

We first describe our method for finding a min-area min-cut replication set for a component V_i. Given a digraph $G = (V, E)$ and a bipartition $\{\overline{V_i}, V_i\}$, we want to find a smallest replication set $U_i \subseteq \overline{V_i}$ that minimizes $|in(V_i \biguplus U_i)|$.

We construct a digraph flow network $G' = (V', E')$ from G as described in [HE95]. The flow network constructed from the digraph in Fig. 1 (a) is shown in Fig. 2

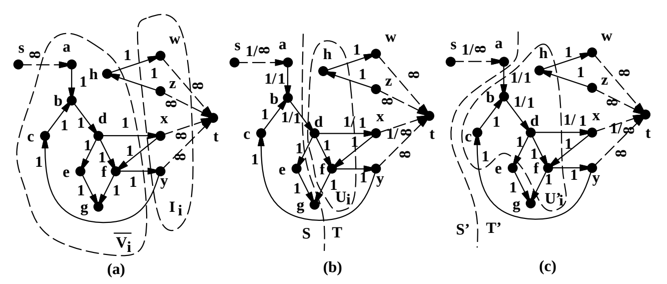


Figure 2: (a) A flow network G'. The labels indicate the capacities of the edges. (b) A min-area min-cut (S,T) in G'. $e(S,T) = \{(b,d)\}$. $U_i = \{d,f,h\}$. The label x/y on an edge e indicates that $cap(e) = y$ and the flow value on e is x. The flow value on other edges is 0. (c) A max-area min-cut (S',T') in G'. $e(S',T') = \{(a,b)\}$. $U_i' = \{b,c,d,f,h\}$.

(a). $V' = \overline{V_i} \cup I_i \cup \{s,t\}$, where $I_i = \{v \in V_i \mid v$ is incident on a cut edge between $\overline{V_i}$ and $V_i\}$, and s and t are the newly added *super source* and *super sink* nodes. $E' = E_i \cup E_s \cup E_t$, where E_i is the set of edges in the subgraph of G induced by $\overline{V_i} \cup I_i$, $E_s = \{(s,u) \mid u \in \overline{V_i}$, u is a primary input node in $\overline{V_i}\}$, and $E_t = \{(v,t) | v \in I_i\}$. Let $cap(e)$ denote the capacity of an edge $e \in E'$. We assign $cap(e) = 1$ if $e \in E_i$, $cap(e) = \infty$ if $e \in E_s \cup E_t$.

A *cut* (S,T) of G' is a partition of nodes into S and T such that $s \in S$ and $t \in T$. The edge cut $e(S,T)$ induced by the cut (S,T) is the set of edges in E' whose starting node is in S and ending node is in T. Note that $e(S,T)$ contains forward edges from S to T only (see Fig. 2 (b)). A cut (S,T) of G' is a *min-cut* if its induced edge cut $e(S,T)$ contains the minimum number of edges among all cuts of G'. A min-cut (S,T) of G' is a *min-area min-cut* if the total area of T is minimum among all min-cuts of G'.

Lemma 3.1 *There is a one-to-one correspondence between a cut (S,T) in G' and a replication set $U_i \subseteq \overline{V_i}$, by letting $U_i = T \setminus (I_i \cup \{t\})$ and $T = U_i \cup I_i \cup \{t\}$. Further, $|in(V_i \uplus U_i)| = |e(S,T)|$.*

Corollary 3.1.1 *Let (S,T) be a min-area min-cut in the flow network G'. Then $U_i = T \setminus (I_i \cup \{t\})$ is a min-area min-cut replication set to V_i.*

Hence the problem of finding a min-area min-cut replication set to V_i is reduced to the problem of finding a min-area min-cut in the flow network G'.

An *augmenting path* from u to v in a flow network is a simple path from u to v in the undirected graph obtained from the network by ignoring edge directions that can be used to push additional flow from u to v. The *capacity of a cut* (S,T) is the sum of the capacities on the *forward edges* in $e(S,T)$. A max-flow in G' is a flow of maximum value. A max-flow in a flow network defines many min-cuts with varying areas for T (see Fig. 2 (b) & (c)). The following theorem is the most commonly used version of the max-flow min-cut theorem, which finds a min-cut with the maximum area for T.

Theorem 3.1 Max-flow min-cut theorem [FF62]
Given a max-flow f in G', let $S = \{v \in V' : \exists$ an augmenting path from s to v in $G'\}$, and let $T = V' \setminus S$. Then (S,T) is a cut of minimum capacity $|f|$, and f saturates all forward edges from S to T.

The algorithm of [HE95] finds a min-cut replication set based on the set T of the max-area min-cut in the above max-flow min-cut theorem. More specifically, the min-cut replication set U_i' found by the algorithm in [HE95] is the set of all nodes $u \in \overline{V_i}$ for which no augmenting path exists from s to u in the flow network G' (after the max-flow computation), *and* a directed path exists from u to t in the initial flow network G' (before the max-flow computation). Such a min-cut replication set contains many unnecessary nodes. For example, in Fig. 2 (c), the min-cut replication set found by the algorithm in [HE95] is $U_i' = \{b,c,d,f,h\}$. The nodes b and c in U_i' are redundant since they do not contribute to further reduction of the cut size. In Fig. 2 (b), The min-area min-cut replication set U_i found by our algorithm (corresponding to the min-area min-cut (S,T)) achieves the same reduction in cut size as U_i' but contains the least number of nodes. When the flow network G' is large, the difference in $area(T)$ between the max-area min-cut and the min-area min-cut is significant. This is confirmed by the experimental results shown in Section 6. Our algorithm for min-area min-cut replication in a digraph is listed below.

Algorithm 1: Finding Min-Area Min-Cut Replication Sets in a Digraph

```
1. for $i = 1$ to $k$
2.     Construct the digraph flow network $G' = (V', E')$
       with respect to component $V_i$;
3.     Find a max-flow in $G'$ from $s$ to $t$;
4.     Let $T = \{v \in V' | \exists$ an augmenting path from $v$ to $t\}$,
       and let $S = V' \setminus T$;
       /* Steps 2-4 find a min-area min-cut in $G'$. */
5.     Let $U_i = T \setminus (I_i \cup \{t\})$.
```

Theorem 3.2 *Algorithm 1 finds the min-area min-cut replication sets for a digraph G, and runs in $O(km^2)$ time, where m is the number of edges in G.*

4 Optimal Min-Area Min-Cut Replication for Hypergraphs

In the previous section, we gave an algorithm for finding a min-area min-cut replication using a digraph model for a circuit. The optimal algorithm for min-cut replication in [HE95] also works on digraphs. However, a more accurate model for a circuit with multi-terminal nets is a hypergraph.

A *hyper-digraph* (or *hypergraph* for short) is represented by $H = (V, \tilde{E})$, where V is the set of nodes, and $\tilde{E}$ is the set of hyperedges representing the interconnections among the nodes in V. For each hyperedge $\tilde{e} \in \tilde{E}$, $\tilde{e} = (u, \{v^1, \ldots, v^l\})$ represents a net with u being the source node, and $v^1, \ldots, v^l$ being the sink nodes of the net.

Given a subset $U_i \subseteq \overline{V_i}$, the *replication of* U_i *into* V_i is obtained by adding to V_i a new node v_i for each node $v \in U_i$. The new node v_i is the *clone* of v in V_i. Further, for each hyperedge $\tilde{e} = (u, \{v^1, \ldots, v^l\}) \in E$,
1. if $u \notin U_i$, then for each sink node $v^m \in U_i$ of $\tilde{e}$, $1 \leq m \leq l$, make v_i^m a new sink node of $\tilde{e}$ (see hyperedges $(b, \{c, d, d_i\})$ in Fig. 3 (b));

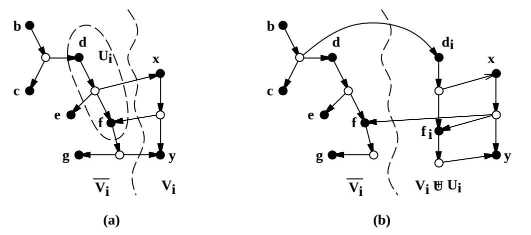


Figure 3: (a) A hypergraph, and a partition $\{\overline{V_i}, V_i\}$ with 3 cut hyperedges. A white node represents a net. (b) After replicating $U_i = \{d, f\}$ into V_i, the number of cut hyperedges is reduced to 2.

2. if $u \in U_i$, then create a new hyperedge $\tilde{e'} = (u_i, X)$ such that

- for each $v^m \in V_i$, $1 \le m \le l$, remove v^m from $\tilde{e}$ and add v^m to X, and
- for each $v^m \in U_i$, $1 \le m \le l$, add v_i^m to X (see hyperedge $(d_i, \{x, f_i\})$ in Fig. 3 (b)).

The hypergraph min-cut replication problem: Given a hypergraph $H = (V, \tilde{E})$, and a k-way partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ of V, find a collection of sets of vertices $\{U_{ij} \mid 1 \le i, j \le k\}$ such that $U_{ij} \subseteq V_i$ and after replicating U_{ij} into V_j, the number of cut hyperedges is minimum, for $i, j = 1, 2, \ldots, k$.

We modify some definitions related to hypergraphs. Let $out(V_i) = \{\tilde{e} \in \tilde{E} \mid$ the source node of $\tilde{e}$ is in V_i, and at least one sink node of $\tilde{e}$ is in $\overline{V_i}\}$ denote the set of hyperedges outgoing from V_i, and let $in(V_i) = \{\tilde{e} \in \tilde{E} \mid$ the source node of $\tilde{e}$ is in $\overline{V_i}$, and at least one sink node of $\tilde{e}$ in $V_i\}$ denote the set of hyperedges incoming to V_i, for $1 \le i \le k$.

The authors of [HE95] described a heuristic method to extend their digraph min-cut replication algorithm to hypergraphs, by replacing each hyperedge with a directed tree and then finding a min-cut replication set $U_i \subseteq \overline{V_i}$ for each V_i to minimize $|in(V_i \uplus U_i)|$. However, this heuristic does not guarantee a minimum cut after replication for a given k-way partition of a hypergraph. This is because $in(V_i)$ and $in(V_j)$ for different i and j may contain many common hyperedges in a hypergraph, and the number of actual cut hyperedges is usually less than $\Sigma_i |in(V_i)|$. Hence minimizing $\Sigma_i |in(V_i)|$ does not mean that the number of cut hyperedges will be minimum after replication.

In this section, we give an exact solution to the hypergraph min-cut replication problem. Our solution is based on the observation that although $in(V_i)$ and $in(V_j)$ for different i and j maybe overlapping in a hypergraph, $out(V_i)$ and $out(V_j)$ for different i and j are disjoint in a hypergraph, since each hyperedge has only one source node. Hence $\Sigma_i |out(V_i)|$ is the number of actual cut hyperedges in a partitioned hypergraph.

Note that minimizing $\Sigma_i |out(V_i)|$ is not a simple operation symmetric to minimizing $\Sigma_i |in(V_i)|$ in the min-cut replication problem, since $\overline{V_i} = \bigcup_{j \ne i} V_j$ contains many components. Our strategy is to find a set $W_i \subseteq V_i$ such that by replicating W_i to $\overline{V_i}$, $|out(V_i)| = |in(\overline{V_i} \uplus W_i)|$ is minimized. Replicating W_i to $\overline{V_i}$ is defined as replicating W_{ij} into V_j, for each $j \ne i$, where $W_{ij} = \{w \in W_i \mid$ w is reachable to V_j through only nodes in $W_i\}$. Let $\overline{V_i} \uplus W_i$ denote the set $\overline{V_i}$ after the replication of W_i. For example in Fig. 4, $W_1 = \{c, d, e\}$, $W_{12} = \{c\}, W_{13} = \{d\}$, and $W_{14} = \{e\}$; after the replication of W_1 into $\overline{V_1}$, $out(V_1) = in(\overline{V_1} \uplus W_1) = \bigcup_{j \ne 1} in(V_j \uplus W_{1j})$.

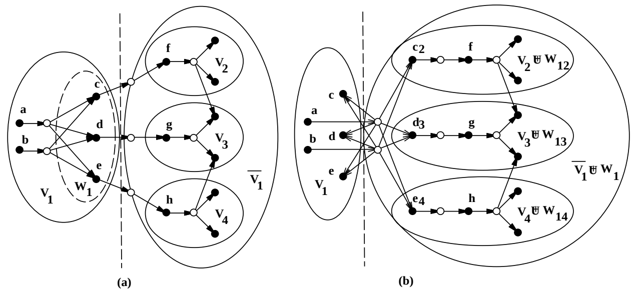


Figure 4: (a) A hypergraph H and a partition $\{V_1, V_2, V_3, V_4\}$ with 3 cut hyperedges. A white node represents a hyperedge. (b) A min-area min-cut replication of H with 2 cut hyperedges.

We show that the hypergraph min-cut replication problem can be solved by solving k independent problems of finding a min-cut replication set $W_i \subseteq V_i$ to $\overline{V_i}$ to minimize $|out(V_i)| = |in(\overline{V_i} \uplus W_i)|$, for each i, $1 \le i \le k$ (see Fig. 4). The set of the cut hyperedges $\tilde{C} = \bigcup_i out(V_i)$, and $|\tilde{C}| = \Sigma_i |out(V_i)|$. By the definition of replication in a hypergraph, replicating a set $W_i \subseteq V_i$ to $\overline{V_i}$ changes $out(V_i) = in(\overline{V_i} \uplus W_i)$ only, and does not change $out(V_j)$ for $j \ne i$. Hence the hypergraph min-cut replication problem can be broken into k independent problems of finding a min-cut replication set $W_i \subseteq V_i$ to $\overline{V_i}$ for each i, $1 \le i \le k$.

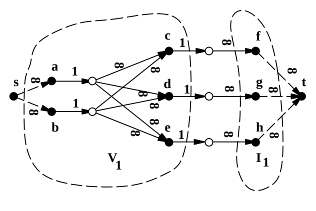
The equivalent hypergraph min-cut replication problem: Given a hypergraph $H = (V, \tilde{E})$, and a k-way partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ of V, find min-cut replication sets $W_i \subseteq V_i$ such that after replicating W_i to $\overline{V_i}$, $|out(V_i)| = |in(\overline{V_i} \uplus W_i)|$ is minimized, for $i = 1, 2, \ldots, k$.

The hypergraph min-area min-cut replication problem: Find a solution $\{W_i \mid 1 \le i \le k\}$ to the equivalent hypergraph min-cut replication problem such that the area of the replicated nodes, $\Sigma_i area(W_i)$, is minimum.

Hence finding a min-area min-cut replication set W_i to $\overline{V_i}$, for each i, gives us an optimal solution to the hypergraph min-area min-cut replication problem.

4.1 Finding a Min-Area Min-Cut Replication Set in a Hypergraph

In this subsection, we consider the problem of finding a min-area min-cut replication set $W_i \subseteq V_i$ for $\overline{V_i}$ to minimize $|out(V_i)| = |in(\overline{V_i} \uplus W_i)|$. As in the flow network construction of a digraph given in Section 3, we construct a hypergraph flow network $H' = (V', E')$ from a hypergraph $H = (V, \tilde{E})$. The main differences are 1) s is connected to the primary input nodes in V_i rather than in $\overline{V_i}$, 2) for each hyperedge, we add a node called *hypernode* in H', and 3) the capacity is 1 for an edge incoming to a hypernode, and ∞ for other edges. Fig. 5 (a) shows the flow network constructed from the hypergraph in Fig. 4 (a). Since each hyperedge has only



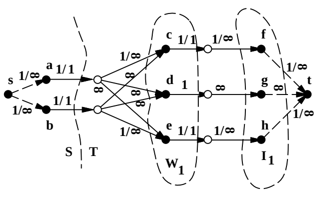


Figure 5: (a) A flow network H' constructed from a hypergraph. The label on an edge indicates the capacity of the edge. A white node indicates a hypernode. (b) A min-area min-cut (S,T) in H'. $W_1 = \{c,d,e\}$ is the min-area min-cut replication set to $\overline{V_1}$ found in our algorithm. The label x/y on an edge e indicates that $cap(e) = y$ and the flow value on e is x. The flow value on other edges is 0.

one source node, each hypernode has only one incoming edge. This capacity assignment guarantees that only the edges incoming to hypernodes (corresponding to hyperedges) will be cut. Hence the capacity of a cut is exactly the number of cut hyperedges.

Specifically, $V' = \overline{V_i} \cup I_i \cup \{s,t\} \cup V_h$, where $I_i = \{v \in V_i \mid v$ is incident on a cut hyperedge between V_i and $\overline{V_i}\}$, s and t are the newly added *super source* and *super sink* nodes, and V_h contains a hypernode for every hyperedge in $\tilde{E}$. $E' = E_i \cup E_s \cup E_t$, where E_i is the set of pins in the subgraph of H induced by $\overline{V_i} \cup I_i$, $E_s = \{(s,u) \mid u \in V_i$, u is a primary input node in V_i, or u is pre-determined to be fixed in $V_i\}$, and $E_t = \{(v,t) | v \in I_i\}$. The capacity of an edge $e \in E'$ is assigned as follows: $cap(e) = 1$ if e is an incoming edge to a hypernode, $cap(e) = \infty$ otherwise.

Algorithm 2: Finding Min-Area Min-Cut Replication Sets in a Hypergraph

```
1. for $i = 1$ to $k$
2.     Construct a hypergraph flow network $H' = (V', E')$
       with respect to component $V_i$;
3.     Find a max-flow in $H'$ from $s$ to $t$;
4.     Let $T = \{v \in V' | \exists$ an augmenting path from $v$ to $t\}$,
       and let $S = V' \setminus T$;
5.     Let $W_i = T \setminus (I_i \cup \{t\})$;
6.     for $j = 1$ to $k$, $j \neq i$
7.         Let $W_{ij} = \{ w \in W_i \mid w$ is reachable to $V_j$
               through only nodes in $W_i\}$.
```

For example, given a hypergraph H and a 4-way partition as in Fig. 4 (a), the heuristic method given in [HE95] cannot further improve the cut size, since the incoming cut hyperedges to each partition is already minimum. However, Algorithm 2 finds the optimal min-area min-cut replication solution shown in Fig. 4 (b).

Theorem 4.1 *Algorithm 2 finds the min-area min-cut replication sets for a hypergraph H, and runs in $O(knP)$ time, where n is the number of hyperedges in H and P is the total number of pins in H.*

4.2 Further Advantages of the Hypergraph Flow Network

In addition to exactly modeling net cuts in a hypergraph, the hypergraph flow network described in the previous subsection has the following advantages compared with the digraph flow network given in [HE95] and Section 3. The time complexity of a network flow based algorithm depends on the size of the flow network. The dominating factor of the flow network size is V_i for the hypergraph flow network, and $\overline{V_i}$ for the digraph flow network. Since $\overline{V_i} = \bigcup_{j \neq i} V_j$ contains $k-1$ components of a partition and V_i is just one component, the digraph flow network is significantly larger than the hypergraph flow network. This difference contributes to not only longer runtime when using the digraph flow network, but also larger min-cut replication sets found in [HE95], since the min-cut replication sets they find are based on the max-area min-cuts in the already large digraph flow network.

5 Min-Cut Replication with Area and Pin Constraints

The *area and pin constrained min-cut replication problem* is a min-cut replication problem with the additional requirement that after the replication, each component has to satisfy an area constraint and a pin constraint. It was shown in [HE95] that the area and pin constrained min-cut replication problem is NP-complete. Hence heuristics are necessary to solve the constrained min-cut replication problem efficiently.

We assume that each component in the initial partition satisfies both area and pin constraints. Min-cut replication will not increase the pin count for any component. However, some min-cut replication sets might cause some components to violate the area constraint after replication, in which case heuristics are needed to find other feasible min-cut replication sets (though not optimal in terms of cut nets) that will satisfy the area constraint after replication.

The heuristic implemented in MC-Rep in the TAPIR package [HE95] is described below. Given a k-way partition $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$, for each bipartition $\{\overline{V_i}, V_i\}$, where $i = 1, 2, \ldots, k$, the following steps are performed.
1. The optimal flow based min-cut algorithm in [HE95] is applied to $\{\overline{V_i}, V_i\}$ to find a min-cut replication set $U_i \subseteq \overline{V_i}$.
2. If the flow based solution U_i is feasible, then U_i is replicated into V_i.
3. If U_i is infeasible, then a *failure* is recorded, the flow based solution U_i is ignored, and an FM heuristic for replication is applied to $\{\overline{V_i}, V_i\}$ returning a feasible replication set.

In our package Hyper-MAMC (Hypergraph Min-Area Min-Cut replication), we apply an efficient max-flow min-cut heuristic proposed in [YW94] to repeatedly cut the oversized min-cut replication sets to obtain smaller replication sets with gradually increased cut sizes. For each bipartition $\{\overline{V_i}, V_i\}$, where $i = 1, 2, \ldots, k$, our steps are the following.
1. The optimal flow based hypergraph min-area min-cut algorithm (Algorithm 2 in Section 4) is applied to $\{\overline{V_i}, V_i\}$ to find a min-area min-cut replication set $W_i \subseteq V_i$ (notice the difference from MC-Rep). Let $W_{ij} = \{w \in W_i \mid w$ is reachable to V_j through only nodes in $W_i\}$ for $j \neq i$. Each W_{ij} is to be replicated into V_j for $j \neq i$.
2. If the solution W_i is feasible, i.e., W_{ij} is feasible for V_j for every $j \neq i$, then W_{ij} is replicated into V_j for every $j \neq i$.

Circuit Size			Constraints	
Circuit	#gates	#nets	#nodes	#pins
c1355	708	618	750	50
c1908	534	365	750	50
c3540	1401	1016	750	50
c2670	999	860	1500	100
c5315	2271	1655	1500	100
c6288	3286	2824	1500	100
c7552	2719	2140	1500	100
s15850	14839	10385	1500	100
s35932	29313	17830	1500	100
s5378	4525	2995	1500	100
s9234	7775	5846	1500	100

Table 1: MCNC Partition93 benchmark circuits and their constraints.

3. If U_i is infeasible for some V_j, then a *failure* is recorded, W_i is not ignored, but rather used as a basis for repeated max-flow min-cut to find a feasible replication set.

The repeated max-flow min-cut process was implemented efficiently using incremental flow computation. Thus the time spent in computing the oversized min-cut replication sets is not wasted as in MC-Rep, and the repeated cut process takes time proportional to one max-flow computation [YW94].

Note that in both MC-Rep and Hyper-MAMC, the number of failures recorded indicates the number of oversized replication sets found by the optimal flow based min-cut replication algorithms, and in those failed cases, feasible replication sets were found using the heuristics. We claim that for the area and pin constraint replication problem, the repeated flow based heuristic in [YW94] is better than the FM heuristic given in [HE95] in terms of the final cut size, since the repeated flow based heuristic starts from the optimal min-cut replication set (oversized) and gradually relaxes the min-cut size until the replication set is feasible, while the FM heuristic ignores the optimal min-cut replication set (oversized) and uses the FM heuristic from scratch, thus wasting the time spent in computing the oversized replication set, and not taking advantage of the max-flow theory. The experimental results shown in Section 6 strongly support our claim. For example, for circuits s15850, s35932, and s9234 where both heuristics were applied to most components, our method results in 51.2%, 57.1%, and 48.8% fewer cut nets respectively, and uses much less runtime.

6 Experimental Results

The experiment was conducted with the help of the original TAPIR package in [HE95]. The initial partitions of a set of MCNC Partition93 benchmark circuits, that satisfy predefined area and pin constraints (see Table 1), were obtained using MW-Part in TAPIR. MW-Part is a k-way partitioning program based on a recursive FM bipartitioning heuristic. MC-Rep in TAPIR is the implementation of the optimal min-cut replication algorithm given in [HE95].

We implemented our optimal hypergraph min-area min-cut replication algorithm (i.e., Algorithm 2 in Section 4) in a package called Hyper-MAMC, and interfaced Hyper-MAMC to TAPIR. We compared the results for Hyper-MAMC with those obtained for MC-Rep on the same initial partitions. For the oversized min-cut replication sets, Hyper-MAMC uses a repeated max-flow min-cut heuristic to find smaller replication sets with small cut size, while MC-Rep applies an FM replication heuristic to minimize pin counts for those components, as described in Section 5. As shown in Table 1, the benchmark circuits ranged in size from six hundred to twenty thousand gates, after removing all size-one nets. Table 1 also shows the corresponding partition constraints for each circuit.

Init. Partition		MC-Rep	Hyper-MAMC
Circuit	k	#failures	#failures
c1355	6	6	0
c1908	5	5	0
c3540	12	7	0
c2670	4	4	0
c5315	8	6	0
c6288	4	4	3
c7552	5	5	0
s15850	13	13	12
s35932	24	24	23
s5378	8	8	0
s9234	7	7	6

Table 2: Comparison of Hyper-MAMC and MC-Rep on failure rate.

In Table 2 we compare the number of failures (i.e., the number of min-cut replication sets that cause the area constraint violation) using MC-Rep and Hyper-MAMC. Hyper-MAMC has a much lower failure rate since it finds a min-area min-cut replication set, which is always no bigger than the min-cut replication set found by MC-Rep. Note that the majority of the min-cut replication sets found by MC-Rep are oversized, and hence counted as failures. This is consistent with the statement made in [HE95] that "most of the time the max-flow replication solution was infeasible and was ignored by MC-Rep" (and was replaced by the FM based heuristic solution). Two major factors contributed to the large min-cut replication sets found by MC-Rep: 1) the digraph flow network constructed by MC-Rep is as large as the whole circuit, while the hypergraph flow network constructed by Hyper-MAMC is only of the size of one component; 2) the min-cut replication sets found by MC-Rep are based on max-area min-cuts in the already large digraph flow network, while the min-area min-cut replication sets found by Hyper-MAMC are based on a min-area min-cut in the much smaller hypergraph flow network. The two factors compounded together significantly worsen the performance of MC-Rep. Therefore the unnecessarily large min-cut replication sets significantly limit the application of the flow based min-cut replication algorithm in [HE95] in practice, and MC-Rep basically performs the FM replication heuristic to reduce cut size.

Table 3 compares the min-cut replication results of MC-Rep and Hyper-MAMC. The first two columns give the circuit name and the number of cut nets[3] in the initial partition obtained by MW-Part in TAPIR. The next two columns show the number of cut nets after applying MC-Rep, and the runtime of MC-Rep. The last four columns show the number of cut nets after applying Hyper-MAMC, the improvement percentage in cut size of Hyper-MAMC over the initial partition and over MC-Rep, and the runtime of Hyper-MAMC.

[3]The external PI nets and PO nets are not counted in cut nets in both TAPIR and Hyper-MAMC.

Init. Partition		MC-Rep		Hyper-MAMC			
Circuit	cut nets	cut nets	rtime (sec.)	cut nets	imp.% Init.	imp.% MC-Rep	rtime (sec.)
c1355	73	63	3.5	21	71.2%	66.7%	0.7
c1908	59	52	2.9	12	79.7%	76.9%	0.4
c3540	188	117	19.8	13	93.1%	88.9%	1.5
c2670	46	39	4.6	31	32.6%	20.5%	0.7
c5315	138	101	21.2	38	72.5%	62.4%	2.6
c6288	122	101	24.0	32	73.8%	68.3%	6.7
c7552	55	53	15.1	49	10.9%	7.5%	2.9
s15850	321	285	1155.1	139	56.7%	51.2%	66.6
s35932	516	487	2.5hrs	209	59.5%	57.1%	184.7
s5378	254	182	66.3	33	87.0%	81.9%	4.3
s9234	197	166	170.2	85	56.9%	48.8%	22.4
Ave.					63.1%	57.3%	

Table 3: Comparison of Hyper-MAMC and MC-Rep on cut size and runtime.

The runtime of both MC-Rep and Hyper-MAMC was measured in elapsed seconds on a Sparc10 workstation with 32 MB of memory. Table 3 shows that on average, Hyper-MAMC generates 57.3% fewer cut nets than MC-Rep, and 63.1% fewer cut nets than the initial partition. In addition, Hyper-MAMC runs much faster than MC-Rep. For example, for a circuit s35932 of almost 20K gates and 24 initial components, Hyper-MAMC took 3 minutes while MC-Rep took 2.5 hours on a Sparc10. Note that the reported runtime for s35932 given in [HE95] is 7695.1 seconds, or 2.1 hours on a Sun4.

The larger cut size of MC-Rep [HE95] is mainly caused by the high failure rate as shown in Table 2, and MC-Rep basically performs FM replication heuristic to reduce cut size, which does not guarantee the minimum cut size as their flow based algorithm does. The larger cut size of MC-Rep is also caused by the heuristic approach used in [HE95] for hypergraph min-cut replication, while Hyper-MAMC exactly models a hypergraph and solves the hypergraph min-area min-cut replication problem optimally.

The longer runtime of MC-Rep is explained by the much larger flow network size, while Hyper-MAMC uses smaller flow network and the repeated max-flow min-cut process takes time proportional to one max-flow computation [YW94]. The reason that both Hyper-MAMC and MC-Rep generate many oversized min-cut replication sets for c6288, s15850, s35932 and s9234 is that, the components of the initial partitions of the four circuits are very close to their capacity of 1500 (i.e., 80% to 90% utilized capacity). Still Hyper-MAMC was able to reduce the cut size significantly while MC-Rep was not for the these circuits, since the repeated max-flow min-cut heuristic used in Hyper-MAMC can find smaller (non-oversized) replication sets with gradually increased cut sizes.

Finally, we note that the replication sets generated by both MC-Rep and Hyper-MAMC in Table 3 satisfy both the area and pin constraints as specified in Table 1. We did not compare Hyper-MAMC with FM-Rep in TAPIR (an FM based heuristic for min-cut replication mainly to speedup MC-Rep) since the cut size of FM-Rep is at most as good as that of MC-Rep.

7 Conclusions

We have presented optimal solutions to the min-area min-cut replication problem for both the digraph model and the hypergraph model. Our algorithms are a successful application of the theoretical network flow technique to circuit partitioning. We implemented our algorithms in the Hyper-MAMC package, and the experimental results show that our package outperforms the best previously known package TAPIR by a significant margin.

Acknowledgment We thank Jim Hwang for providing us a version of the TAPIR system that contains MW-Part, MC-Rep, and FM-Rep functions, and for helpful discussions and comments on our paper.

References

[CHK92] J. Cong, L. Hagen, and A. Kahng. Net Partitions Yield Better Module Partitions. In *Proc. of the 29th ACM/IEEE Design Automation Conf.*, pages 47–52, 1992.

[FF62] J. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[FM82] C. M. Fiduccia and R. M. Mattheyses. A Linear Time Heuristic for Improving Network Partitions. In *Proc. of the ACM/IEEE Design Automation Conf.*, pages 175–181, 1982.

[HE92] J. Hwang and A. El Gamal. Optimal Replication for Min-Cut Partitioning. In *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pages 432–435, Nov. 1992.

[HE95] J. Hwang and A. El Gamal. Min-Cut Replication in Partitioned Networks. *IEEE Trans. on CAD*, 14(1):96–106, Jan. 1995.

[HK91] L. Hagen and A. B. Kahng. Fast Spectral Methods for Ratio Cut Partitioning and Clustering. In *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pages 10–13, Nov. 1991.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, pages 671–680, May 1983.

[KL70] B. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning of Electrical Circuits. *Bell System Technical Journal*, pages 291–307, Feb. 1970.

[KN91] C. Kring and A. R. Newton. A Cell-Replicating Approach to Mincut-Based Circuit Partitioning. In *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pages 2–5, Nov. 1991.

[LKCH95] L.-T. Liu, M.-T. Kuo, C.-K. Cheng, and T. C. Hu. A Replication Cut for Two-Way Partitioning. *IEEE Trans. on CAD*, 14(5):623–630, May 1995.

[MBSV91] R. Murgai, R. K. Brayton, and A. Sangiovanni-Vincentelli. On Clustering for Minimum Delay/Area. In *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pages 6–9, 1991.

[PL88] B. Preas and M. Lorenzetti. *Physical Design Automation of VLSI Systems.* Benjamin/Cummings, 1988.

[RW93] R. Rajaraman and D. F. Wong. Optimal Clustering for Delay Minimization. In *Proc. 30th ACM/IEEE Design Automation Conf.*, pages 309–314, 1993.

[WC89] Y. C. Wei and C. K. Cheng. Towards Efficient Hierarchical Designs by Ratio Cut Partitioning. In *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pages 298–301, Nov. 1989.

[YW94] H. Yang and D. F. Wong. Efficient Network Flow Based Min-Cut Balanced Partitioning. In *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pages 50–55, Nov. 1994.