Min-Cut Replication in Partitioned Networks

L. James Hwang and Abbas El Gamal, *Senior Member, IEEE*

***Abstract*— Logic replication has been shown empirically to reduce pin count and partition size in partitioned networks. This paper presents the first theoretical treatment of the min-cut replication problem, which is to determine replicated logic that minimizes cut size. A polynomial time algorithm for determining min-cut replication sets for k-partitioned graphs is derived by reducing replication to the problem of finding a maximum flow. The algorithm is extended to hypergraphs and replication heuristics are proposed for the $\mathcal{NP}$-hard problem with size constraints on partition components. These heuristics, which reduce the worst-case running time by a factor of $O(k^2)$ over previous methods, are applied to designs that have been partitioned into multiple FPGA's. Experimental results demonstrate that min-cut replication provides substantial reductions in the numbers of FPGA's and pins required.**

I. Introduction

GRAPH partitioning permeates integrated circuit layout and design, arising naturally in placement, floorplanning, and the mapping of large designs into multiple chips and multiple printed circuit boards [13]. The problem can be formulated as follows: Given a graph $G = (V, E)$, find a partition of V into disjoint components with constrained sizes such that the number of edges with vertices in distinct components is minimal. This partitioning problem is known to be $\mathcal{NP}$-hard, but fast and effective partitioning heuristics have been developed and are widely used (e.g., [9], [2]).

A variation of the partitioning problem arises when mapping large designs into multiple field-programmable gate arrays (FPGA's) for rapid hardware prototyping and logic emulation. In addition to the size constraint imposed by the gate capacity of an FPGA, there is a pin constraint that limits the number of cut edges incident on a component. The pin constraint often leads to *pin-limited* partitions, in which all of the pins on the FPGA's are used, but much of the available gate capacity remains unused [17].

One approach to reducing the adverse effects of the pin constraint on partitioning is to use logic replication as demonstrated in Fig. 1. In the example, it can be seen that replicating the vertex u into both components reduces the cut size from 2^n edges to n edges. We refer to this type of replication as *min-cut replication*, since its primary objective is to minimize cut size after partitioning.

This paper represents the first theoretical treatment of min-cut replication. The earliest published reference merely alluded to its usefulness without presenting any details [14]. In 1971,

Manuscript received February 16, 1993; revised August 22, 1994. This research was supported in part by DARPA under Contract J-FBI-89-101. This paper was recommended by Associate Editor G. Zimmerman.

The authors are with Information Systems Laboratory, Stanford University, Stanford, CA 94305 USA.

IEEE Log Number 9406356.

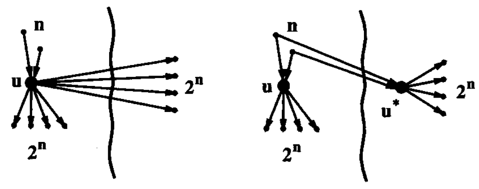


Fig. 1. Replicating u reduces cut size from 2^n edges to n edges.

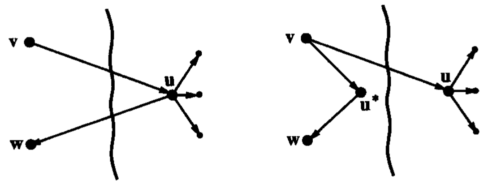


Fig. 2. Replicating u reduces path delay from v to w.

Russo *et al.* described an automated partitioning system that supported replication but required the user to specify the logic to be replicated [16]. Automated replication was not seriously considered until the early 1990's, when several experimental studies showed that combining replication with partitioning can reduce both cut size and the number of partition components. Preliminary experimental results were presented by the authors in [3], and in independent work, Kring and Newton proposed a replication heuristic based on the Fiduccia-Mattheyses partitioning algorithm [10]. Some of the theoretical results in this paper were stated without proof in [8].

In addition to reducing cut size, min-cut replication can also be used to reduce partition size, i.e., the number of partition components. By reducing the number of pins required in a partitioned network, min-cut replication effectively relaxes the pin constraint. We have observed that substantially smaller partitions can be found by first relaxing the pin constraint and then using min-cut replication to reduce the number of pins so that the original pin constraint is satisfied.

Replication has proved essential to partitioning for minimum delay [15]. Although the problem is formulated differently, min-cut replication can have the side benefit of reducing delay by eliminating cut edges along signal paths [7]. As demonstrated in Fig. 2, replicating the vertex u removes two cut edges along the path from v to w. Such reduction in delay, however, is not guaranteed in general.

The paper is organized as follows. Section II contains background material. In Section III, we present a graph-theoretic formulation and a solution to the min-cut replication problem, establishing a connection between replication and classical network flow theory. In Section V, we extend the solution from graphs to hypergraphs, and in Section VI indicate that the replication problem with component size constraints is $\mathcal{NP}$-hard. In Section VII, we describe new replication heuristics based on the solution to the min-cut replication problem. Section VIII contains experimental results demonstrating that min-cut replication can substantially improve the mapping of designs into multiple FPGA's both by reducing the number of FPGA's needed and by reducing the number of pins. Section IX contains concluding remarks.

II. Preliminaries

Let $G = (V, E)$ be a directed graph with vertex set V and edge set $E \subseteq V \times V$. If $S \subseteq V$ is a set of vertices, we denote its complement by $\overline{S} = \{v \in V \mid v \notin S\}$. If S, T, are sets, their relative difference is defined to be $S - T = \{x \in S \mid x \notin T\}$.

Definition 2.1: A **k-cut** $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ is a partition of the vertex set V into k disjoint nonempty subsets or **components**. If $k = 2$, we refer to the k-cut as a **bipartition**. A k-cut may also be referred to simply as a **cut**. Indexing does not imply order; for example, the bipartition $\{V_1, V_2\}$ is identical to $\{V_2, V_1\}$.

Definition 2.2: A **cut edge** is an edge (u, v) where $u \in V_i$ and $v \notin V_i$. The **cut set** is the set of all cut edges in G. We denote the number of cut edges incident into a component V_i by

$$\mathrm{in}(V_i) = |\{(u, v) \in E \mid u \notin V_i, v \in V_i\}|.$$

Definition 2.3: The **cut size** of a k-cut, denoted $|\mathcal{V}|$, is the number of cut edges in G. As each cut edge is incident into exactly one component, we observe that

$$|\mathcal{V}| = \sum_i \mathrm{in}(V_i). \tag{1}$$

Definition 2.4: A vertex u is a **source** if it has in-degree zero or a **sink** if it has out-degree zero; we assume there are no isolated vertices. Signals are introduced into the network via the sources and are externally observable via the sinks. For this reason, we also refer to sources (sinks) as **primary inputs (outputs)**. The set of all primary inputs in a partition component V_i is denoted by $\mathcal{P}_i$.

Definition 2.5: For vertices u and v, a **u-v path** is a directed path from u to v. For any subset $S \subseteq V$ and vertex $v \in V$, we say there is an **S-v path** in G if there exists a u-v path for some $u \in S$. If $F \subseteq E$ is a subset of edges, the set **reachable**(S,F) consists of all vertices v for which there exists an S-v path containing only edges in F. When $S = \{u\}$, we write reachable(u, F). We refer to a vertex $v \in$ reachable(u, E) as a **descendant** of u.

A. Network Flow

We briefly review standard concepts in network flow theory that will be used in our solution of the min-cut replication problem (see e.g., [11]).

Definition 2.6: A **flow network** $G = (V, E)$ is a directed graph with a single source s, a single sink t, and a nonnegative **capacity** function $c: E \rightarrow \mathbf{R}^+$ defined on the edge set. If $(u, v) \notin E$, we define $c(u, v) = 0$.

Definition 2.7: A **flow** in G is a function $f: V \times V \rightarrow \mathbf{R}$ that satisfies three properties.

1) $f(u, v) \leq c(u, v)$ for all $u, v \in V$ (capacity constraint).
2) $f(u, v) = -f(v, u)$ for all $u, v \in V$ (skew symmetry requirement).
3) $\sum_{v \in V} f(u, v) = 0$ for all $u \in V - \{s, t\}$ (flow conservation constraint).

If X and Y are disjoint subsets of V, we define $f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y)$ to be the sum of the flow over all edges between X and Y.

Definition 2.8: The **value** of a flow, $|f| = \sum_{v \in V} f(s, v)$ is the total net flow out of the source. A **maximum-flow** is a flow that has maximal value.

Definition 2.9: A **cut** in a flow network G is a bipartition $\{S, T\}$ such that $s \in S$ and $t \in T$.

As a notational device, we will denote a cut in a flow network by (S, T) as a reminder of the directionality. We assume the convention that the source (sink) is contained in the first (second) coordinate of the ordered pair of sets.

Definition 2.10: The **net flow** across a cut (S, T) is defined to be $f(S, T)$. The **capacity** of the cut is $c(S, T) = \sum_{x \in S} \sum_{y \in T} c(x, y)$. If $f(S, T) = c(S, T)$, we say that f **saturates** the cut. A **minimum cut** is a cut with minimal capacity.

Definition 2.11: If f is a flow in G, then for any vertices u and v, the **residual capacity** of (u, v) is defined to be $c_f(u, v) = c(u, v) - f(u, v)$. If p is a path in G, then we define the residual capacity $c_f(p) = \min_{(u,v) \in p}\{c_f(u, v)\}$. The **residual graph** $G_f = (V, E_f)$ is the graph having edge set $E_f = \{(u, v) | c_f(u, v) > 0\}$. A source-sink path in G_f is called an **augmenting path**, because the flow f can be increased (augmented) along the path p by an amount not exceeding $c_f(p)$.

We state without proof several fundamental results due to Ford and Fulkerson [11].

Theorem 2.1: (max-flow min-cut theorem) The following statements are equivalent:

1. f is a maximum flow;
2. there exists no augmenting path in the residual graph G_f;
3. f saturates some (minimum) cut in G.

Theorem 2.2: A maximum flow can be determined by the **augmenting path method** as follows. Begin with $f(u, v) = 0$ for all edges, then repeat the following step until obtaining a flow with no augmenting path: find an augmenting path p and increase the value of the flow by pushing $c_f(p)$ units of flow along p.

Theorem 2.3: If $c(u, v)$ is integral for all u and v, then the augmenting path method determines an integral maximum flow f. In addition, $f(u, v)$ is integral for all u and v.

B. Replication

Definition 2.12: Let $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$ be a k-cut of G. The **replication** of a vertex $u \in V_i$ into a different component

V_j is the graph $\hat{G} = (\hat{V}, \hat{E})$ defined as follows. The vertex set $\hat{V} = V \cup \{u_j\}$ is obtained by adding a new vertex u_j to the component V_j. The edge set $\hat{E}$ is the same as E except for the following modifications to edges incident to u.

- For every incoming edge (w, u), $\hat{E}$ contains a new edge (w, u_j).
- Every outgoing cut edge (u, v) where $v \in V_j$, is replaced by an edge (u_j, v) in $\hat{E}$.

We refer to the vertices u and u_j as **clones**. If $V_i = V_j$ or if there already exists a clone of u in V_j, then replication is defined to be the identity map. It is not hard to see that replication is commutative and associative. As a result, any permutation of a sequence of replication steps from a component V_i to V_j results in the same partition. We can therefore extend the definition in the natural way to subsets of vertices; the actual order of replications need not be specified, or can be chosen for convenience. We will assumes this extended definition throughout this paper. It should also be clear that the transformation is reversible; the inverse transformation, referred to as **unreplication**, is defined in a similar manner.

We denote by $V_j \uplus S$ the component V_j after replication of a set S.

Definition 2.13: Let $S \subseteq V$ be a vertex set and V_j be a component in a partitioned network. Define

$$O_j(S) = \{(u, v) \in E \mid u \in S - V_j \text{ and } v \in V_j\}$$

to be the set of all **outgoing** edges removed from the $\{\overline{V}_j, V_j\}$ cut set when S is replicated into V_j. Similarly, define

$$I_j(S) = \{(w, u) \mid w \in \overline{V}_j - S \text{ and } u \in S - V_j \text{ and } u_j \notin V_j\}$$

to be the set of all **incoming** edges added to the cut set. The **replication gain** of S is the change in cut size, given by

$$\text{gain}_j(S) = |O_j(S)| - |I_j(S)| = \text{in}(V_j) - \text{in}(V_j \uplus S).$$

When the component V_j is understood, we write $O(S)$, $I(S)$, and gain(S), and when $S = \{u\}$, we write gain(u).

III. The Min-Cut Replication Problem

Formally, the replication problem we consider first is the following.

Min-Cut Replication Problem (MCRP):

Given a directed graph $G = (V, E)$ and k-cut $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$, determine a collection of sets, $\{V_{ij}^* : 1 \leq i, j \leq k\}$, that minimizes the cut size $|\mathcal{V}^*|$, where $\mathcal{V}^*$ is the partition that results when V_{ij}^* is replicated from V_i to V_j for all i and j.

Although the MCRP abstracts away important practical capacity constraints, it remains an important and nontrivial problem, and as we will show, has an elegant solution. Moreover, the solution is of more than theoretical interest; it is applicable whenever the size constraints are sufficiently loose. As we will see, the solution also forms the basis for heuristic approaches to $\mathcal{NP}$-complete constrained replication problems.

A. The Bipartition Replication Problem

We begin our discussion by considering the simplest nontrivial version of the MCRP, namely finding a min-cut replication set for one component of a bipartition. In this case, the MCRP can be restated as follows.

Simple Min-Cut Replication Problem:

Given a directed graph $G = (V, E)$ and bipartition $\{V_1, V_2\}$, determine a replication set $V_{12}^* \subseteq V_1$,

$$V_{12}^* = \arg \min_{R \subseteq V_1} \text{in}(V_2 \uplus R),$$

which minimizes the number of cut edges into V_2. Equivalently, find a replication set $V_{12}^* \subseteq V_1$,

$$V_{12}^* = \arg \max_{R \subseteq V_1} \text{gain}(R),$$

having maximal gain.

We refer to any subset having maximal gain as a **min-cut replication** set. Clearly, since the empty set has zero gain, the maximal gain is nonnegative. If the maximal gain in fact equals zero, we will simply assume the empty set as the solution to the SMCRP.

Our method of solving the SMCRP will be to reduce the replication problem to a maximum flow problem. We will construct a flow network $G' = (V', E')$ derived from G, for which a maximum flow will define a cut that separates the replication set from the network source. The replication set V_{12}^* will be found by breadth-first search in the flow network. Since efficient algorithms are known for breadth-first search and computing a maximum flow, we will thereby obtain an efficient algorithm for the SMCRP. We will refer to this method of solution as the **max-flow replication algorithm**.

The flow network G' is defined as follows.

Definition 3.1: Let $I_{12} = \{v \in V_2 \mid \exists u \in V_1 : (u, v) \in E\}$ be the set of vertices in V_2 that are incident on a cut edge **into** V_2, and let source s and sink t be new vertices. We define the vertex set $V' = V_1 \cup I_{12} \cup \{s, t\}$. It is easy to see that the sets $V_1' = V_1 \cup \{s\}$ and $V_2' = I_{12} \cup \{t\}$ define a cut (V_1', V_2') in G' that corresponds to the cut edges in G from V_1 to V_2.

The edge set E' consists of selected edges in E and new edges between vertices in V and the new source s and sink t.

Definition 3.2: Let $E_s = \{(s, u) | u \in \mathcal{P}_1\}$ be a set of new edges from the source s to every primary input in V_1. Let $E_t = \{(v, t) \mid u \in I_{12}\}$ be a set of new edges from every vertex in I_{12} to the sink t. Letting $E_{12} = E \cap (V_1 \times V')$ be the set of all edges in G within V_1 or from V_1 to I_{12}, we define the flow network edge $E' = E_s \cup E_t \cup E_{12}$.

Definition 3.3: The edge capacity function $c : E' \rightarrow \mathbf{R}^+$ is defined on E' as follows:

$$c(u, v) = \begin{cases} \infty & \text{if } (v, t) \in E_t, \\ \infty & \text{if } (s, u) \in E_s \text{ and } u \text{ cannot be replicated}, \\ 1 & \textit{otherwise} \end{cases}$$

This definition allows replication of selected primary inputs to be enabled and prevented if so desired.

The max-flow replication algorithm proceeds by determining a maximal s-t flow in the flow network G' and defining a replication set based on the flow. Let f be a maximum flow

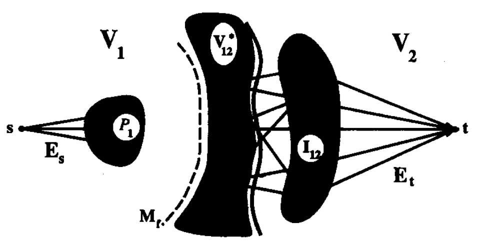


Fig. 3. Flow network derived from the original graph G.

computed by the augmenting path method of Theorem 2.2. By Theorem 2.1, we know that there is no augmenting s-t path in the residual graph G'_f. We define the replication set V_{12}^* to consist of selected vertices that are unreachable from the source s in G'_f.

Definition 3.4: Let $T = \{u \in V_1' \mid t \in \text{reachable}(u, E')\}$ be the set of all vertices in V_1' that lie on a path to t in the original flow network G', and let $T_f = \{u \in V_1' \mid u \notin \textit{reachable}(s, E'_f)\}$ be the set of all vertices unreachable from the source s in the residual graph G'_f. We define the replication set

$$V_{12}^* = T \cap T_f$$

to be the intersection of T and T_f, and define M_f to be set of edges that separates V_{12}^* from $V_1 - V_{12}^*$.

It follows from Theorem 2.3 that the flow value $|f|$ is integral since all edge capacities are integral. In fact, since the capacity $c(u,v) = 1$ for every saturated cut edge, $|f|$ is equal to the size of the cut set in G'. Identifying the set V_{12}^* can be accomplished using breadth-first search in G and G_f. We note that the set M_f is the cut set for the cut $(V_1' - V_{12}^*, V_2' \cup V_{12}^*)$, and by Theorem 2.1, is a minimum cut.

The flow network construction is depicted in Fig. 3. With this definition of V_{12}^*, we have the following result.

Theorem 3.1: V_{12}^* is a solution to the SMCRP. That is,

$$V_{12}^* = \arg \max_{R \subseteq V_1} \textit{gain}(R).$$

We begin the proof of Theorem 3.1 with several lemmas concerning two subsets of V_1. These subsets characterize the maximum possible gain of replicating any subset of V_1 into the component V_2.

Definition 3.5: Let $C = \textit{reachable}(s, E \cap V_1 \times V_1)$ be the set of all vertices in V_1 that are **connected** to the source s in the flow network G'.

Definition 3.6: Let $\mathcal{D} = \{u \in V_1 \mid u \notin C$ and reachable $(u, E) \cap V_2 \neq \emptyset\}$ be the set of all vertices in V_1 that are **disconnected** from the source s in G', but have descendants in V_2.

It is intuitively clear that $\mathcal{D}$ should be replicated, as doing so will remove cut edges from the network without introducing any new cut edges. We state this formally as a lemma.

Lemma 3.1: For any set $R \subseteq V_1$, the gain of replicating R into V_2 satisfies the inequality

$$\textit{gain}(R) \leq \textit{gain}(R \cup \mathcal{D}),$$

with strict inequality if $\mathcal{D} \nsubseteq R$.

Proof: It follows from the definition of the vertex set $\mathcal{D}$ that the edge set $I(\mathcal{D})$ is empty. Therefore, after replicating R, we have $\textit{gain}(\mathcal{D}) = \textit{gain}(\mathcal{D} - R) \geq |O(\mathcal{D} - R)| \geq 0$. Furthermore, $u \in \mathcal{D} - R$ implies that after replicating R, reachable$(u, E) \cap (V_2 \uplus R)$ is not empty, which in turn implies that $O(\mathcal{D} - R)$ is not empty. Hence, if $\mathcal{D} - R$ is not empty, the inequality is strict. □

An immediate consequence of Lemma 3.1 is the fact that $\mathcal{D}$ must be contained in any min-cut replication set. The next lemma states that the replication set V_{12}^* does contain $\mathcal{D}$.

Lemma 3.2: If $\mathcal{D}$ is defined as in Defn. 3.6, then $\mathcal{D} \subseteq V_{12}^*$.

Proof: If $\mathcal{D}$ is empty the lemma is trivially true, so suppose it is not empty. Let $u \in \mathcal{D}$ be arbitrarily chosen. With C defined as in Defn. 3.5, we know that $u \notin C$. By construction, this implies that $u \notin \textit{reachable}(s, E')$, which implies $u \notin \textit{reachable}(s, E'_f)$. Hence $u \in T_f$. Also, $u \in \mathcal{D}$ implies that $\textit{reachable}(u, E) \cap V_2$ is nonempty, which implies that there exists a u-t path in G', hence $u \in T$. It follows that $u \in T \cap T_f = V_{12}^*$. □

The gain of replicating a vertex set is invariant under permutation of the elements of the set. Therefore the gain of replicating a set R containing $\mathcal{D}$ is simply the gain of replicating $\mathcal{D}$ plus the gain of replicating $R - \mathcal{D}$ after replicating $\mathcal{D}$. The next lemma provides an upper bound on the gain of replicating any set that contains $\mathcal{D}$.

Lemma 3.3: Let f be the maximum flow computed by the augmenting path method in the definition of V_{12}^* and let M_f, C, and $\mathcal{D}$ be defined as above. Then after replicating $\mathcal{D}$, we have the following inequality for the gain of replicating any set $R \subseteq V_1$.

$$\textit{gain}(R) \leq |O(C)| - |M_f|.$$

Proof: The disjoint union $(O(C) - O(R)) \cup I(R)$ is an s-t cut set in G'. Therefore, since M_f is a minimum cut, we have

$$\begin{aligned} |M_f| &\leq |(O(C) - O(R)) \cup I(R)| \\ &= |O(C) - O(R)| + |I(R)|. \end{aligned}$$

Furthermore, after replicating $\mathcal{D}$, we have $O(R) \subseteq O(C)$, which implies that

$$|O(C) - O(R)| = |O(C)| - |O(R)|.$$

It follows that

$$\textit{gain}(R) = |O(R)| - |I(R)| \leq |O(C)| - |M_f|,$$

as desired. □

Essentially, all that remains to prove Theorem 3.1 is to prove that the gain of replicating V_{12}^* achieves the bound of Lemma 3.3.

Proof: (of Theorem 3.1) We wish to show that V_{12}^* has maximal gain of any subset of V_1. We will first prove that V_{12}^* achieves the bound of Lemma 3.3. By Lemma 3.2, we know that $\mathcal{D} \subseteq V_{12}^*$, and we may assume that in replicating V_{12}^*, the subset $\mathcal{D}$ is replicated first. It is then easily verified from the definitions that after replicating $\mathcal{D}$, the set of original cut edges that remain, $O(V_{12}^*) = O(C) - M_f$, and the set of new cut edges introduced by replication, $I(V_{12}^*) = M_f - O(C)$, where $\mathcal{C}$ is given by Defn. 3.5 and M_f is given by Defn. 3.4. Therefore,

$$\begin{aligned} gain(V_{12}^*) &= |O(V_{12}^*)| - |I(V_{12}^*)| \\ &= |O(\mathcal{C}) - M_f| - |M_f - O(\mathcal{C})| \\ &= |O(\mathcal{C}) - M_f| + |M_f \cap O(\mathcal{C})| \\ &\quad - |M_f \cap O(\mathcal{C})| - |M_f - O(\mathcal{C})| \\ &= |O(\mathcal{C})| - |M_f|. \end{aligned}$$

Now suppose that $R \subseteq V_1$ is an arbitrary min-cut replication set. By Lemma 3.1, we know that if $\mathcal{D} \not\subseteq R$, then $gain(R \cup \mathcal{D}) \geq gain(R)$, so it must be that R contains $\mathcal{D}$. After replicating $\mathcal{D}$, by Lemma 3.3 we have that

$$gain(R) = gain(R - \mathcal{D}) \leq |O(\mathcal{C})| - |M_f| = gain(V_{12}^*),$$

which completes the proof. □

Having established that V_{12}^* is a min-cut replication set, we can determine the running time of the max-flow replication algorithm to obtain an upper bound on the run-time complexity of the SMCRP.

Corollary 3.1: Given a directed graph $G = (V, E)$ and a cut $\{V_1, V_2\}$, a min-cut replication set into V_2 can be determined in time $O(nm \log(n^2/m))$, where $n = |V|$ and $m = |E|$.

Proof: By Theorem 3.1, it is sufficient to show that V_{12}^*, computed by the max-flow replication algorithm, can be determined in $O(nm\log(n^2/m))$ time.

Clearly the flow network G' can be computed in $O(m)$ time since the vertex set $|V'| = O(n)$ and the edge set $|E'| = O(m)$. The maximum flow f can be determined in time $O(nm\log(n^2/m))$ using the algorithm in [5]. The set V_{12}^* can be identified by a breadth-first search of G' and G'_f in time $O(m)$. Hence, $O(nm\log(n^2/m))$ time is sufficient to determine V_{12}^*. □

We see that the SMCRP can be solved efficiently. Furthermore, since the replication problem can be reduced to a flow problem, speed-ups in flow computations can lead to speed-ups of the max-flow replication algorithm. However, we observe that the max-flow replication algorithm does not guarantee a min-cut replication set of minimal size. In fact, the difference in size between V_{12}^* and the smallest min-cut replication set can be unbounded due to the existence of many minimum cut sets defined by the flow f. Nevertheless, when component size constraints are sufficiently loose, the solution can be feasible.

IV. The Min-Cut Replication Algorithm

In the section we will show that the general MCRP can be solved by solving k independent instances of the SMCRP. We will refer to this method of solution as the **min-cut replication algorithm**.

We begin by making a simple, but important observation about replication in k-partitioned networks.

Lemma 4.1: If $\{V_1, V_2, \ldots, V_k\}$ is a k-cut of G and $R_i \subseteq \overline{V_i}$ is an arbitrary subset of vertices, then replicating R_i into the component V_i cannot alter the number of cut edges incident into a different component V_j.

Proof: By definition, replicating vertices of $\overline{V_i}$ into V_i removes edges from E or adds edges to E only that are incident into V_i. Hence, $\text{in}(V_j)$ is unaffected by replicating R_i into V_i for all components except V_i. □

This innocent looking lemma is really quite powerful; it implies that the k instances of the SMCRP corresponding to the cuts $\{\overline{V_i}, V_i\}$ for $i = 1, 2, \ldots, k$, are independent. Each instance of the SMCRP minimizes the number of cut edges into a particular component. Consequently, a general solution can be obtained by minimizing all of them independently.

We formalize this observation in the following theorem, which provides a solution to the MCRP.

Theorem 4.1: Let $\{V_i^* \mid i = 1, 2, \ldots, k\}$ be a collection of vertex sets, where

$$V_i^* = \arg \min_{R \subseteq \overline{V_i}} \text{in}(V_i \uplus R)$$

is the subset of $\overline{V_i}$ that minimizes the number of cut edges incident into the component V_i when replicated into V_i. Then the collection of vertex sets

$$\{V_{ij}^* \mid V_{ij}^* = V_i \cap V_j^*, 1 \leq i \neq j \leq k\}$$

is a solution to the min-cut replication problem.

Proof: We first assert that replicating the sets V_i^*, for $i = 1, 2, \ldots, k$, results in a minimal cut size over any choice of replication sets. To see this, observe that replicating any set $R_i \subseteq \overline{V_i}$ results in some value for $\text{in}(V_i \uplus R_i)$, the number of cut edges incident into V_i. We know that by Lemma 4.1, replicating R_i into V_i has no effect on $\text{in}(V_j \uplus R_j)$ when i, j are not equal. But since the cut size after replicating all of the sets V_i^* is $\sum_i \text{in}(V_i \uplus R_i)$, choosing the sets V_i^* to be those R_i that minimize each term of the sum minimizes the total sum.

Finally, it is clear that replicating the sets, $V_{ij}^* = V_i \cap V_j^*$, from V_i to V_j, for all i and j, is identical to replicating V_j^* into V_j. Therefore, replicating the collection of vertex sets $\{V_{ij}^*\}$ minimizes the total cut size, as we wished to prove. □

By decomposing the MCRP into k instances of the SMCRP with the additional set intersection operations, we can easily determined the running time of the min-cut replication algorithm. Simply, the running time is on the order of k times the running time of each instance of the SMCRP.

Corollary 4.1: Given a directed graph $G = (V, E)$ and a cut $\{V_1, V_2, \ldots, V_k\}$, the MCRP can be solved in $O(knm\log(n^2/m))$ time, where $n = |V|$ and $m = |E|$.

Proof: It is sufficient to show that the min-cut replication algorithm runs in $O(knm\log(n^2/m))$ time. Applying the max-flow replication algorithm to the k instances of the SMCRP corresponding to the cuts $\{\overline{V_i}, V_i\}$ produces replication sets V_i^*, for $i = 1, 2, \ldots, k$. By Corollary 3.1, this can be done in $O(knm\log(n^2/m))$ time. Since computing the set intersections $\{V_{ij}^* \mid V_{ij}^* = V_i \cap V_j^*, 1 \leq i \neq j \leq k\}$ can be done in $O(kn\log n)$ time by sorting the $\{V_i\}$ and $\{V_i^*\}$ and comparing V_i with V_j^* for all i and j, and result follows. □

V. Replication in Hypergraphs

Since a logic network may contain multi-terminal nets in addition to two-terminal nets, a network is better represented by a hypergraph than by a graph. Although max-flow techniques are known for *undirected* hypergraphs [6], they do not apply to directed hypergraphs as required for replication. Therefore it is necessary to extend the min-cut replication algorithm to determine replication sets in hypergraphs.

Definition 5.1: A **directed hypergraph** $\tilde{G} = (\tilde{V}, \tilde{E})$ consists of a vertex set $\tilde{V}$ and hyperedge set $\tilde{E} \subseteq \tilde{V} \times (2^{\tilde{V}} - \emptyset)$, where a hyperedge $e = (u, S)$ consists of a single source vertex u and a nonempty set of sink vertices (there should be no confusion between this use of source and sink and the usage in flow networks). We say that e **contains** the vertices in $\{u\} \cup S$, and we assume that $u \notin S$ (no self loops).

Definition 5.2: A **cut** $\tilde{\mathcal{V}} = \{\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_k\}$ is a partition of $\tilde{V}$ into nonempty components. A **cut hyperedge** is a hyperedge containing vertices in two or more components. The **cut size** of the cut set, $|\tilde{\mathcal{V}}|$ is the number of cut hyperedges in $\tilde{G}$.

Definition 5.3: Let $\tilde{\mathcal{V}} = \{\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_k\}$, be a partition of the hypergraph $\tilde{G}$. Then the **replication** of a vertex $u \in \tilde{V}_i$ into a different component $\tilde{V}_j$ is the hypergraph obtained by adding a new vertex u_j to component $\tilde{V}_j$, and modifying the edge set $\tilde{G}$ as follows.

- Every incoming hyperedge (w, S), where $u \in S$ is replaced by the hyperedge (w, S'), where $S' = S \cup \{u_j\}$.
- Every outgoing cut hyperedge (u, S) where $S \cap V_j$ is nonempty, is replaced by the hyperedges (u, S_i) and (u, S_j), where $S_i = S \cap \overline{\tilde{V}_j}$ and $S_j = S \cap \tilde{V}_j$.

As for graphs, replication in hypergraphs is extended to sets of vertices.

Because a single hyperedge may have many sinks, replicating a sink vertex into more than one component does not add a new cut hyperedge for each clone, as is the case for graphs. In fact, for hypergraphs we have the inequality, $|\tilde{\mathcal{V}}| \le \sum_i \mathrm{in}(\tilde{V}_i)$, so minimizing the sum is not equivalent to minimizing the cut size. Nevertheless, in practice this distinction is not as critical as it may seem and we will continue to refer to replication sets that minimize the sum as min-cut replication sets. This is particularly true for multiple FPGA partitioning, where the objective is to minimize the number of *pins* rather than the number of *cut nets*. Since the number of pins is $|\tilde{\mathcal{V}}| + \sum_i \mathrm{in}(V_i)$, it is reasonable to minimize the sum.

We would like to apply the min-cut replication algorithm to partitioned hypergraphs, but the max-flow replication algorithm is not directly applicable, since the flow techniques do not apply to hypergraphs. Our approach is to transform a hypergraph into a graph and then apply the min-cut replication algorithm.

Given a hypergraph, $\tilde{G} = (\tilde{V}, \tilde{E})$, with bipartition $\{\tilde{V}_1, \tilde{V}_2\}$, the graph $G = (V, E)$ is defined as follows.

Definition 5.4: Let $\tilde{E}_m = \{e \in \tilde{E} \mid e \in (u, S), |S| > 1\}$ be the set of all hyperedges containing more than two vertices. The vertex set

$$V = \tilde{V} \cup \{v_1^e \mid e \in \tilde{E}_m\} \cup \{v_2^e \mid e \in \tilde{E}_m\}$$

consists of $\tilde{V}$ and two new vertices, v_1^e and v_2^e, for every hyperedge e in $\tilde{E}_m$.

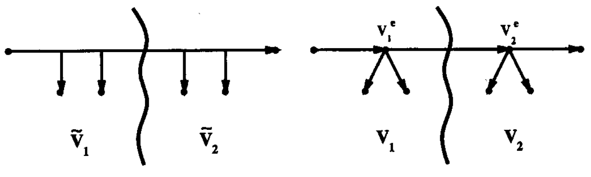


Fig. 4. Replacing a multiple sink hyperedge by a tree.

Definition 5.5: The edge set E contains an edge (u, v) for every two-vertex hyperedge $(u, \{v\}) \in \tilde{E} - \tilde{E}_m$. Each hyperedge in $\tilde{E}_m$ is replaced by a tree as shown by example in Fig. 4. If $e = (u, S)$ is a hyperedge with source u in $\tilde{V}_1$ and sinks S, then E contains edges

- (u, v_1^e),
- (v_1^e, v_2^e) between the two new vertices corresponding to e,
- (v_1^e, v) for every v in $\tilde{V}_1 \cap S$ and
- (v_2^e, v) for every u in $\tilde{V}_2 \cap S$.

To determine a replication set $\tilde{V}_{12}^*$ that minimizes the number of cut hyperedges into $\tilde{V}_2$, we construct a graph $G = (V, E)$ as follows, determine a min-cut replication set V_{12}^*, and set $\tilde{V}_{12}^*$ equal to V_{12}^*.

The following lemma justifies the use of this tree construction for transforming a hypergraph into a directed graph. It states that even though a single hyperedge may be replaced by many graph edges, at most one edge for a hyperedge will ever occur in the cut set defined in the max-flow replication algorithm. Note that this would certainly not be the case if a hyperedge were replaced by a clique.

Lemma 5.1: If G' is the flow network constructed from G in the max-flow replication algorithm, then the minimum cut set M_f defined in Def. 3.4 never contains two edges of a tree corresponding to a single hyperedge.

Proof: Suppose to the contrary that M_f contains two such edges corresponding to a hyperedge $e \in \tilde{E}$. Then replacing these two edges by the edge (u_e, v_1^e) from the source vertex for e to the vertex v_1^e preserves the cut and reduces $|M_f|$, which is impossible. □

This lemma also ensures that applying the max-flow replication algorithm to find a min-cut replication set in the transformed graph G also finds a replication set in the hypergraph that minimizes the sum $\sum \mathrm{in}(\tilde{V}_i)$ for $\tilde{G}$ as well as G'.

Theorem 5.1: Corresponding to the cut set in G' computed by the max-flow replication algorithm is a minimum cut set in $\tilde{G}$ of the same size.

Proof: Let M_f be the minimum cut set in G' defined in Def. 3.4. Then by Lemma 5.1, each hyperedge can have at most one corresponding edge in the cut set. Let $\tilde{M}_f$ be the set of hyperedges corresponding to M_f. We claim that $\tilde{M}_f$ is a minimum cut set in $\tilde{G}$. It is clearly a cut set, since M_f is a cut set in G'. Furthermore, if $\tilde{C}$ is any cut set in $\tilde{G}$ with $|\tilde{G}| < |\tilde{M}_f|$, then $C = \{(u_e, v_1^e) \mid e \in \tilde{C}\}$, is a cut set in G' with $|C| < |M_f|$, a contradiction. □

Therefore, we can extend the min-cut replication algorithm in a natural way to hypergraphs. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be a hypergraph with cut, $\tilde{\mathcal{V}} = \{\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_k\}$. For each bipartition, $\{\overline{\tilde{V}_i}, \tilde{V}_i\}$, construct a bipartitioned graph, $G = (V, E)$, having the cut $\{\overline{V_i}, V_i\}$. When the max-flow replication algorithm is applied to G, we obtain a set V_i^* that minimizes, $\text{in}(V_i)$, the number of cut edges incident into V_i. Theorem 5.1 asserts that V_i^* also minimizes $\text{in}(\tilde{V}_i)$ in the hypergraph $\tilde{G}$. Therefore, applying in min-cut replication algorithm to obtain the collection of vertex sets, $\{V_i^* \mid i = 1, 2, \ldots, k\}$, minimizes $\sum_i \text{in}(\tilde{V}_i)$ in the original hypergraph. The transformation can be done in time linear in the network size, so, as for graphs, the min-cut replication algorithm for hypergraphs runs in $O(knm \log(n^2/m))$ time, where $n = |V|$ and $m = |E|$.

VI. Constrained Replication is $\mathcal{NP}$-Hard

By formulating the MCRP without component size constraints, we were able to derive an efficient solution by reducing the replication problem to a maximum flow problem. However, if we add size constraints to the problem, unsurprisingly, the problem becomes $\mathcal{NP}$-hard. Formally, consider the following decision problem:

Constrained Min-Cut Replication Problem:

Given a directed graph $G = (V, E)$, cut $\{V_1, V_2\}$, integers K and L, and weighting functions $w_V\colon V \to \mathbf{N}$ and $w_E\colon E \to \mathbf{N}$, determine if there exists a replication set $V_{12}^* \subseteq V_1$ for which the resulting cut $\{V_1, V_2, \uplus V_{12}^*\}$ has size less than or equal to K and the resulting sets V_1 and $V_2 \uplus V_{12}^*$ have size less than or equal to L.

Theorem 6.1: The constrained min-cut replication problem is $\mathcal{NP}$-complete.

A proof of this result, based on a polynomial time reduction from the $\mathcal{NP}$-complete PARTITION problem [4], is contained in [7]. Although we do not have a proof of it, we conjecture that the constrained replication problem remains $\mathcal{NP}$-complete when all vertices and edges have unit weight.

VII. Min-Cut Replication Heuristics

We have observed that the max-flow replication algorithm may fail to satisfy a component size constraint and that the replication problem with size constraints is $\mathcal{NP}$-hard. Consequently, in practice, heuristics must be used. The heuristics presented in this section are based on the solution to the MCRP, using a partitioning heuristic to approximate the max-flow replication algorithm.

One of the most widely used partitioning heuristics is a quasi-linear time algorithm due to Fiduccia and Mattheyses (FM) [2], which can be modified to reflect the inherent edge directionality necessary for replication. We denote the modified FM partitioning heuristic by `FM-Dir`. The primary differences between `FM-Dir` and FM are as follows.

- The gain calculations must be modified to reflect the directionality of replication. Like FM, `FM-Rep`'s gain is defined to the change in the number of *pins* rather than the number of cut nets. An example gain calculation is shown in Fig. 5.
- The size constraint must apply only to V_i, the component being replicated into, since all vertices in $\overline{V_i}$ are candidates for replication.

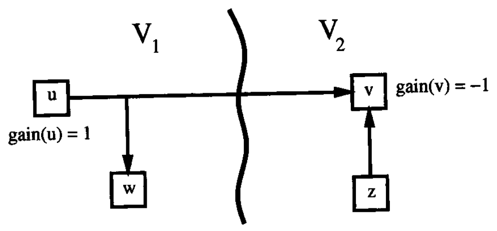


Fig. 5. Gain calculation for directional FM-based replication.

TABLE I
Benchmark Designs and Partitioning Constraints

Design			Constraints	
MCNC	gates	pins	gates	pins
c1355	708	73	750	50
c1908	534	58	750	50
c3540	1401	72	750	50
s1196	827	30	750	50
s1238	829	30	750	50
c2670	999	221	1500	100
c5315	2271	301	1500	100
c6288	3286	64	1500	100
c7552	2719	313	1500	100
s13207	13440	154	1500	100
s15850	14839	103	1500	100
s35932	29313	357	1500	100
s38584	33553	292	1500	100
s5378	4525	86	1500	100
s9234	7775	43	1500	100
Industrial	modules		modules	
addresspart	623	91	295	57
biga	1210	95	546	69
cat	719	41	295	57
control	517	66	295	57
entmisc	1529	72	546	69
gme_a	2013	59	546	69
look1240	619	60	295	57
pdt	453	63	295	57
seq	1824	76	546	69
sgv	569	57	295	57
vrcl	544	43	295	57

With these modifications, `FM-Dir` approximates the maximum-flow computation in the max-flow replication algorithm.

`FM-Dir` forms the basis of a replication heuristic for constrained min-cut replication that we denote by `FM-BiRep`[1]. To determine a set of replicate into a component V_i, we consider the cut $\{V_i, \overline{V_i}\}$. Every vertex $u \in V_i$ is fixed to

[1]referred to as `FM-Rep` in [8].

prevent it from being moved during any of the partitioning passes. FM-Dir is then applied to the resulting bipartition, with the size constraint enforced only for component V_i. After partitioning is completed, any vertex that has been moved into V_i is replicated. Like FM, each pass of FM-Dir runs in time, $O(p)$, where p is the number of pins in the network.

FM-BiRep forms the basis for two k-way replication heuristics, both suggested by the min-cut replication algorithm. The first, denoted by FM-Rep simply applies FM-BiRep to each bipartition $\{\overline{V_i}, V_i\}$ for $i = 1, 2, \ldots, k$. The second, denoted by MC-Rep[2], is similarly simple to describe. First, the max-flow replication algorithm is applied to each of the cuts $\{\overline{V_i}, V_i\}$ independently, for $i = 1, 2, \ldots, k$. If the max-flow solution is feasible, it is replicated. If, however, replicating this set leads to a violation of the component size constraint, FM-BiRep is applied to the bipartition, returning a feasible replication set. If the max-flow solution is empty, then FM-BiRep is not invoked.

Both FM-Rep and MC-Rep make a single pass over the partition components. It is shown in [7] that this approach reduces the worst-case running time by a factor $O(k^2)$ over a naive approach which computes replication pairwise between all component pairs.

The running times for FM-Rep and MC-Rep are easily determined. Each call to FM-BiRep takes time $O(p)$, so the running time of FM-Rep is $O(kp)$. Moreover, in practice the total number of pins in the network is $O(n)$, and k is bounded by a constant, so we see that FM-Rep runs in time essentially linear in the input size.

For MC-Rep, The max-flow step dominates the running time. If the push-relabel method of [5] is used for the max-flow replication algorithm, each flow step takes $O(nm \log(n^2/m))$ time, where $n = |V|$ and $m = |E|$. FM-BiRep runs in time $O(p)$ [2], where p, the total number of pins in the network, is clearly $O(nm)$. Therefore, MC-Rep runs in time $(knm \log(n^2/m))$.

VIII. Experimental Results

We have implemented the min-cut replication algorithms in TAPIR, a tool for automatic partitioning that incorporates replication [7]. Initial partitions were generated by MW-Part, an FM-based multiple-way partitioning algorithm that computes good quality min-cut partitions as described in the Appendix. We applied FM-Rep and MC-Rep to fifteen designs from the MCNC Partitioning93 benchmark suite and ten industrial design implemented in the Actel FPGA library. As shown in Table I, the designs ranged in size from about five hundred to over thirty thousand gates.

In the first experiment, each design was partitioned using MW-Part and the number of cut nets and the total number of pins in the partition were recorded. FM-Rep and MC-Rep were each applied to the partitioned designs, and the corresponding numbers were again recorded. The results are shown in Table II.

[2]referred to as Flow-FM in [8].

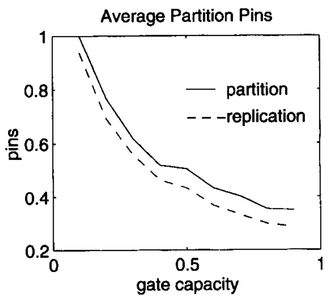


Fig. 6. Average total number of partition pins vs. gate capacity (normalized), with and without replication.

The relative reduction in the number of pins,

$$\Delta p = \frac{\text{partition pins} - \text{replication pins}}{\text{partition pins}},$$

expresses the change in pin count obtained by applying replication. Although the reduction Δp varied substantially from design to design, it is apparent that replication substantially reduced both the numbers of cut nets and total pins for most of the designs. The number of total pins was reduced by 15% on average using FM-Rep and by 18% on average using MC-Rep. The gate utilization was on average, 42% after partitioning and 54% after replication. The running times indicate that the flow steps dominate the MC-Rep computation; FM-Rep runs substantially faster, especially on the large designs. The running time was measured in CPU seconds on a Sun4, running C code compiled by gcc without the optimizer.

The relatively small difference in pin reduction between FM-Rep and MC-Rep indicates that most of the time, the max-flow replication solution was infeasible and was ignored by MC-Rep. Another approach would be to use the infeasible max-flow solution as the starting point for the replication heuristic. If the gate capacity constraint is *nearly* met, this approach may do better than the current method.

In the second experiment, we partitioned each design D, with the size constraint varying from $0.1|D|$ to $0.9|D|$ and the pin constraint set to allow feasible partitions. To avoid excessive running time in partitioning, we considered only the twenty smallest designs. For each size constraint, the design was partitioned, MC-Rep was applied, and the total pin reduction was recorded. Fig. 6 contains the results.

Fig. 6 is a plot of the total numbers of pins in the partition, before and after replication, as functions of the size constraint. The gate capacity is normalized to the design size $|D|$, and the pin counts are normalized to the values when the size constraint is 0.1 $|D|$. Fig. 7 is a plot of the average pin reduction Δp as a function of size constraint, where the error bars depict plus or minus one standard deviation. As can be seen, the total number of pins after partitioning decreased with increased capacity and Δp increased on average from about 10% to 20%. Although for smaller capacities, the results agreed well with the first experiment, when the capacity exceeded about $0.5|D|$, the variance in Δp increased significantly.

TABLE II
CUT SIZE AND TOTAL PINS, WITH AND WITHOUT REPLICATION

Design	M-W-Partition			FM-Rep			run	MC-Rep			run
	FPGAs	cuts	pins	cuts	pins	Δp	time	cuts	pins	Δp	time
c1355	6	73	232	63	215	.07	.8	63	215	.07	11.1
c1908	5	59	197	52	181	.08	.3	52	181	.08	4.8
c3540	12	188	542	153	457	.16	3.6	117	420	.23	38.7
s1196	6	84	239	25	130	.46	.8	10	114	.52	8.7
s1238	8	119	340	27	150	.56	1.2	0	135	.60	11.1
c2670	4	46	321	39	308	.04	.8	39	308	.04	8.7
c5315	8	138	610	115	550	.10	4.4	101	539	.12	47.6
c6288	4	122	335	101	279	.17	3.1	101	279	.17	55.0
c7552	5	55	432	53	424	.02	3.6	53	424	.02	35.4
s13207	12	268	841	269	813	.03	44.5	269	813	.03	960.5
s15850	13	321	869	285	777	.11	46.4	285	777	.11	1812.0
s35932	24	516	1767	487	1552	.12	348.2	487	1552	.12	7695.1
s38584	28	631	1844	609	1739	.06	357.6	609	1739	.06	10849.8
s5378	8	254	663	182	520	.22	6.0	182	520	.22	128.6
s9234	7	197	493	166	418	.15	8.5	166	418	.15	233.1
addresspart	6	74	283	56	223	.21	1.0	30	195	.31	8.8
cat	3	31	110	31	110	0.0	.4	31	110	0.0	3.7
control	6	94	272	81	245	.10	.8	79	243	.11	7.0
look1240	10	150	481	113	395	.18	1.9	67	335	.30	17.6
pdt	4	51	175	37	148	.15	.5	31	141	.19	3.4
vrc1	3	39	130	40	130	0.0	.5	40	130	0.0	3.7
biga	10	154	490	143	462	.06	4.1	62	372	.24	35.3
entmisc	10	218	599	115	360	.40	5.0	102	345	.42	45.7
gme_a	7	131	366	116	326	.11	3.2	116	325	.11	41.1
seq	6	73	339	60	313	.08	3.6	23	268	.21	32.6
avg						.15				.18	

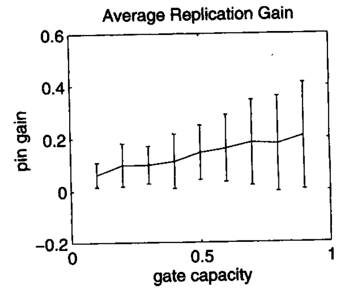


Fig. 7. Average replication gain vs. component gate capacity, with one standard deviation error bars.

The next experiment demonstrates that min-cut replication can reduce the number of FPGA's required to implement a design. The approach was to first increase the pin constraint by η pins, apply partitioning, and then apply MC-Rep to the partition. For each design, binary search was used to determine the maximal pin constraint relaxation η for which MC-Rep rendered the partition feasible under the original pin constraint. Again, to avoid excessive run time we restricted attention to the twenty smallest designs.

Letting k_p and k_r be the numbers of FPGA's in the partitions with the original and expanded pin constraints respectively, the relative reduction in the number of FPGA's is given by $\Delta_F = \frac{k_p}{k_r}$. The data shown in Table III includes k_p, k_r, Δ_F, the maximum value of η obtained for each design, and η expressed as a fraction of the original pin constraint. As can be seen, the number of FPGA's was reduced for 14 of the 20 designs, with a 23% average reduction in the number of FPGA's required.

IX. CONCLUSION

Motivated by the problem of mapping designs into multiple FPGA's, we have formulated and solved the min-cut replication problem. For the $\mathcal{NP}$-hard constrained replication problem, we proposed FM-Rep, an essentially linear time replication heuristic, and MC-Rep, a flow-based heuristic. We applied both algorithms to standard partitioning benchmarks and demonstrated that combining replication with partitioning substantially reduces the number of pins as well as the partition size.

Our formulation of the min-cut replication problem assumed the existence of an initial partition. A natural question is whether or not decoupling replication and partitioning limits the resulting cut size. In theory the answer is no.

TABLE III
TOTAL NUMBER OF FPGA'S REQUIRED WITH AND WITHOUT REPLICATION

Partitioning		Replication			
Design	FPGAs	FPGAs	Δ_F	η	η/γ_p
c1355	6	5	.17	7	.14
c1908	5	5	0.0	0	0
c3540	12	8	.33	13	.26
s1196	6	2	.67	59	1.18
s1238	8	2	.75	59	1.18
c2670	4	4	0.0	0	0
c5315	8	6	.25	15	.15
c6288	4	4	0.0	0	0
c7552	5	5	0.0	0	0
s5378	8	6	.25	35	.35
addresspart	6	4	.33	18	.26
cat	3	3	0.0	0	0
control	6	5	.17	8	.14
biga	10	8	.20	13	.19
entmisc	10	6	.40	10	14
gme_a	7	6	.14	19	.28
look1240	10	4	.60	22	.32
pdt	4	3	.25	29	.42
seq	6	5	.17	18	.26
vrcl	3	3	0.0	0	0
avg			.23		

Proposition 9.1: Let $G = (V, E)$ be a directed graph and $\mathcal{U} = \{U_1, U_2, \ldots, U_k\}$ be a partition of V, possibly with replicated vertices, with minimal cut size. Then there exists a disjoint partition $\mathcal{V}$ to which applying the min-cut replication algorithm results in a cut of the same minimum size.

Proof: Within the partition $\mathcal{U}$, arbitrarily choose one vertex to be the representative for each set of clones, and consider the disjoint partition, $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$, obtained from $\mathcal{U}$ by unreplicating every clone to its representative. Obviously $\mathcal{U}$ can be derived from $\mathcal{V}$ by a sequence of replication steps. Hence if $\mathcal{V}^*$ is the partition obtained from $\mathcal{V}$ by the min-cut replication algorithm, then by Theorem 4.1, $|\mathcal{V}^*| \leq |\mathcal{U}|$. As $|\mathcal{U}|$ was assumed minimal, the result follows. □

We observe that in practice, a *min-cut* partition may not be the best initial partition. This can be demonstrated by the simple example shown in Fig. 8. Suppose a partition component cannot contain more than three vertices. Then, as indicated by the bold dotted line in Fig. 8(a), the min-cut partition has cut size equal to three. However, it is easily verified that the initial partition depicted by the lighter dotted line, having cut size equal to four, results in the min-cut replication set of size two, shown in Fig. 8(b). In contrast, min-cut replication does not improve the original min-cut partition.

Although our approach separates replication from partitioning, we should emphasize that a complete decoupling is *not* required. The min-cut replication approach applies replication at the *component-level*, performing replication with respect to a bipartition. It can therefore be applied to a candidate component at any time during partitioning. Whether

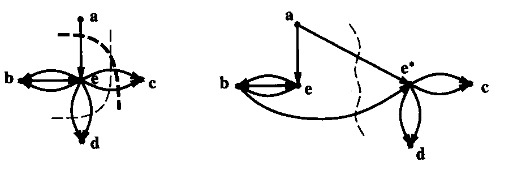


Fig. 8. Min-cut partition (in bold) is not the best initial partition.

a top-down partitioning algorithm or a bottom-up clustering approach is used, min-cut replication can be applied during the partitioning process to produce a partition component. For example, replication can usefully be applied whenever a component *almost* satisfies the pin constraint.

There are a number of important open problems related to min-cut replication. We conclude by listing several such problems.

1) Find the *smallest* solution to the MCRP.
2) Prove that the Constrained MCRP remains $\mathcal{NP}$-hard when all vertex and edge weights are unity.
3) Find an exact minimum cut size solution to the hypergraph MCRP.
4) Characterize initial partitions for which min-cut replication is most effective.

APPENDIX

TAPIR was developed as part of a project to investigate FPGA-based architectures and CAD for rapid hardware prototyping and logic emulation [3], [7]. It is currently integrated with a CAD system for field-programmable multi-chip modules [12]. TAPIR is written in C, with `Flow-Rep` based on a modified version of the `maxflow` module from **misII** [1]. The input to TAPIR is a netlist specified in either the Actel ADL or Xilinx XNF format.

The multiple-way partitioning algorithm used in TAPIR, `MW-Part`, is based on the FM bipartitioning heuristic (see Fig. 9). `MW-Part` generates partitions that satisfy components size and pin constraints as follows. In a series of rounds, `MW-Part` maintains a set of infeasible components, initially containing the entire design. In each round, the largest infeasible component is selected from this set and bipartitioned using the FM heuristic, taking the best result from a number of random initial bipartition (for the experiments described in Section VIII, ten random bipartitions were computed for each component splitting step). After splitting this infeasible component, FM is applied to all component pairs to equalize their sizes while reducing the cut size. If components can be merged without violating the size and pin constraints, they are merged instead of bipartitioned. In the next round, the largest infeasible component is selected, bipartitioned, and again, all component pairs are bipartitioned to equalize the component sizes. The process continues until all components are feasible.

ACKNOWLEDGMENT

We thank D. How, J. Gill, and S. Lan for helpful discussions, and J. Kouloheris and P. Siegel for assistance in running

```
MW-Part($G$) {
    $\mathcal{V} \leftarrow \{V\}$;
    while ($\exists$ an infeasible component) {
        $V_i \leftarrow$ largest infeasible cmpt;
        $\mathcal{V} \leftarrow \mathcal{V} - \{V_i\}$;
        $\{V_{i0}, V_{i1}\} \leftarrow$ SplitCmpt($V_i$);
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{V_{i0}, V_{i1}\}$;
        PartAllCmptPairs($\mathcal{V}$);
    }
    return($\mathcal{V}$);
}

SplitCmpt ($V_i$) {
    $\{V_{i0}, V_{i1}\} \leftarrow$ FM(RandomBipartition($V_i$));
    repeat for max iterations
        $\{U_0, U_1\} \leftarrow$ FM(RandomBipartition($V_i$));
        if ( cut size ($\{U_0, U_1\}$) $<$ cut size ($\{V_{i0}, V_{i1}\}$))
            $\{V_{i0}, V_{i1}\} \leftarrow \{U_0, U_1\}$;
    return($\{V_{i0}, V_{i1}\}$);
}

PartAllCmptPairs ($\mathcal{V}$){
    foreach (component pair ($V_i, V_j$))
        if ( $|V_i \cup V_j| \leq$ size constraint ) {
            $V_i \leftarrow V_i \cup V_j$;
            $\mathcal{V} \leftarrow \mathcal{V} - \{V_j\}$;
        }
        else
            FM ($\{V_i, V_j\}$);
    return ($\mathcal{V}$);
}
```

Fig. 9. MW-Part: a multiple-way partitioning algorithm.

the experiments. Dana deserves special thanks for his careful reading of early versions of this paper. We would also like to thank the reviewers for comments that helped improve the readability of this paper.

REFERENCES

[1] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. Computer-Aided Design*, vol. 6, no. 6, pg. 1062–1081, Nov. 1987.

[2] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. 19th Design Automation Conf.*, 1982, pp. 175–181.

[3] A. El Gamal *et al.*, "Architectures, circuits and computer-aided design for electrically programmable VLSI," *Semi-Annual Tech. Report*, Defense Advanced Research Projects Agency, Mar. 1991.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: Freeman, 1979.

[5] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," in *Proc. 18th ACM Symp. Theory Comput.*, pp. 136–146, 1986.

[6] T. C. Hu and K. Moerder, "Multiterminal flows in a hypergraph," in *VLSI Circuit Layout: Theory and Design*, T. C. Hu and E. S. Kuh, Eds., New York: IEEE Press, pp. 87–93.

[7] L. J. Hwang, "Replication in partitioned networks," Ph.D. dissertation, Stanford Univ., 1994.

[8] J. Hwang and A. El Gamal, "Optimal replication for min-cut partitioning," in *Dig. Tech. Papers, ICCAD-92*, pp. 432–435, IEEE and ACM, 1992.

[9] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell Sys. Tech. J.*, vol. 49, pp. 291–307, 1970.

[10] C. Kring and A. R. Newton, "A cell-replicating approach to mincut-based circuit partitioning," in *Dig. Tech. Papers, ICCAD-91*, pp. 2–5, 1991.

[11] L. R. Ford, Jr. and D. R. Fulkerson, *Flows In Networks.* Princeton, NJ: Princeton Univ. Press, 1962.

[12] S. Lan, A. Ziv, and A. El Gamal, "Routing algorithms for field programmable multi-chip modules," in *Proc. 31st Design Automation Conf.*, IEEE and ACM, 1994.

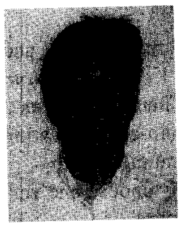
[13] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout.* Sussex, UK: Wiley, 1990.

[14] W. A. Notz, E. Schischa, J. L. Smith, and M. G. Smith, "Benefitting the system designer," *Electronics*, pp. 130–141, Feb. 1967.

[15] R. Rajaraman and D. F. M. Wong, "Optimal clustering for delay minimization," in *Proc. 30th Design Automation Conf.*, 1993, pp. 309–314.

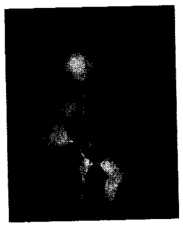
[16] R. L. Russo, P. H. Oden, and P. K. Wolff, Sr., "A heuristic procedure for the partitioning and mapping of computer logic graphs," *IEEE Trans. Comput.*, vol. 20, no. 12, pp. 1455–1462, Dec. 1971.

[17] S. Walters, "Computer-aided prototyping for ASIC-based systems," *IEEE Design Test Comp.*, pp. 4–10, June 1991.



James Hwang received the B.A. degree in computer and information sciences from the University of California, Santa Cruz, in 1986, and the M.S. degree in electrical engineering from Stanford University, in 1988. He is currently a doctoral student of electrical engineering at Stanford, specializing in design automation for configurable VLSI systems.

His current research interests include algorithms and CAD for field-programmable multichip systems, and formal verification



Abbas El Gamal (S'71-M'73-SM'83) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1978.

He is currently an Associate Professor of Electrical Engineering at Stanford University and Chief Technical Officer of SiArc. From 1978 to 1980, he was an Assistant Professor of Electrical Engineering at the University of Southern California. From 1981 to 1984, he was an Assistant Professor of Electrical Engineering at Stanford. He was on leave from Stanford from 1984 to 1987, first as Director of LSI Logic Research Lab, then as cofounder and Chief Scientist of Actel Corporation. His research interests include VLSI circuits, architectures, and synthesis, FPGA's and mask programmable gate arrays, smart sensors, displays, image compression, error correction, and information theory. He has written or co-written more than 60 papers and holds 15 patents in these areas.