A Replication Cut for Two-Way Partitioning

Lung-Tien Liu, Ming-Ter Kuo, Chung-Kuan Cheng, *Senior Member, IEEE*, and T. C. Hu

***Abstract*—Graph partitioning is crucial in multiple-chip design, floorplanning and mapping large logic networks into multiple FPGA's. Replication logic can be used to improve the partitioning. Given a network G with only two-pin nets and a pair of nodes s and t to be separated, we introduce a replication graph and an $O(mn\log(n^2/m))$ algorithm for optimum partitioning with replication and without size constraints, where m and n denote the number of nets and the number of nodes in G, respectively. In VLSI designs, each partition has size constraints and the given network contains multiple-pin nets. A heuristic extension is adopted to construct replication graphs with multiple-pin nets. Then we use a directed Fiduccia-Mattheyses algorithm in the constructed replication graph to solve the replication cut problem with size constraints.**

I. Introduction

In VLSI circuit layout, a common problem is to partition the cells (gates, circuits, devices, etc.) into parts, each part occupying a separate chip such that the number of interchip connections is minimized. In a graph model, a circuit can be represented by a network G. The nodes in G represent the cells and the nets in G represents the connection between cells. Each node is associated with the size of its respective cell and each net is associated with the cost (number of connections) of its respective net. The classical two-way partitioning problem is to partition the nodes of G into two subsets no larger than a given size, so as to minimize the total cost of the nets cut. With the size constraint on each subset of nodes, this problem is known to be $\mathcal{NP}$-complete [7]. In the case of partitioning into two subsets without size constraints, we can derive an optimum solution using the max-flow min-cut approach [1]. However, the method addresses only the case where no replication of cells is allowed.

In practice, a given cell can be replicated and placed in two chips so as to reduce the number of interchip connections. For example, in Fig. 1(a), the Min-Cut partitioning separating cells S and T has a cut cost of 13, while replicating R results in a cut cost of 4 as shown in Fig. 1(b). In order to maintain the functional correctness of the duplicated circuit, each cell in the replicated circuit should collect the same input nets as that in the original circuit. For example, the R on right part of the cut in Fig. 1(b) should have input nets from S and from T. However, the nets from R to S are not replicated because S gets inputs from R on left part of the cut. This property of replication is utilized to reduce the number of interchip connections in partitioning.

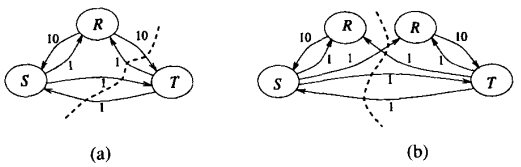


Fig. 1. Effect of replication. (a) The min-cut has a cut cost of 13 without replication. (b) Replicating R results a cut cost of 4.

In [4], Charney and Plato first propose the replication problem. Later [12] and [14] propose heuristic approaches. Kring and Newton [12] extend the Fiduccia and Mattheyses (FM) algorithm to allow nodes to be duplicated explicitly during partitioning. In a recent paper by Hwang and El Gamal [9], the replication problem is formulated as a problem to determine optimum replication sets for an existing partitioning. They find a partitioning (V_1, V_2) with no replication first. Given the obtained partitioning, the max-flow min-cut algorithm is used to identify a set of nodes such that replicating the set will minimize the crossing net count. In this restricted problem that V_1 and V_2 have to remain on their respective sides, the replication is optimal in the case with no size constraints. In the case with size constraints, if the resulted replication by the max-flow min-cut algorithm satisfies the constraint, it is kept as a solution. Otherwise, a directed FM heuristic which considers net directionality is applied to obtain a feasible solution. Their approach achieves a 21.3% reduction in crossing net count on average for the multiway partitioning, when compared with a recursive FM method.

In this paper, we target the general replication problem which is not restricted to any prior partitioning. Given a network G with only two-pin nets and a pair of nodes s and t to be separated, we present an optimum algorithm to solve the replication partitioning problem without size constraints. We first formulate the partitioning as a linear programming problem. The formulation leads to the construction of a novel replication graph. Optimality is achieved by applying the maximum flow algorithm on the replication graph. The running time of this algorithm is $O(mn\log(n^2/m))$ [8] where m and n is the number of nets and nodes, respectively.

For VLSI applications, however, each partition has a maximum size constraint. In addition, we should consider the case of networks containing multiple-pin nets. A heuristic extension is adopted to construct a replication graph for the case of networks with multiple-pin nets. Then we use a directed Fiduccia-Mattheyses method on the replication graph to derive a two-way replication partitioning of the original network.

Manuscript received December 29, 1993; revised July 15, 1994 and November 29, 1994. This paper was supported in part by Grants from the NSF project MIP-9315794. This paper was recommended by Associate Editor T. Yoshimura.

L.-T. Liu is with AT&T Bell Laboratories, Murray Hill, NJ 07974 USA.

M.-T. Kuo, C.K. Cheng, and T. C. Hu are with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093 USA.

IEEE Log Number 9409644.

The remainder of the paper is organized as follows. In Section II, we formulate the partitioning problem with replication allowed. In Section III, we concentrate on the case of networks containing only two-pin nets and no size constraints. We use linear programming to formulate the partitioning problem. Based on its dual program, we develop a replication graph which gives an optimum solution with replication allowed. In Section IV, we introduce an approach for constructing the replication graph of a network with multiple-pin nets and apply a directed Fiduccia-Mattheyses method on the constructed replication graph to solve the replication cut problem arising in real VLSI designs. A discussion on the experimental results is presented in Section V. In Section VI, we state the conclusions.

II. Problem Formulation

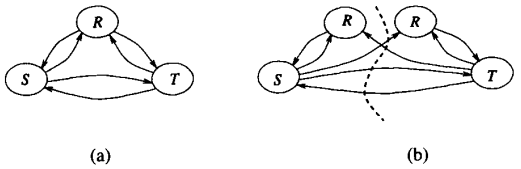
Given a circuit, we represent its netlist as a directed network $G = (V, E)$ where set V consists of nodes i $(i = 1, \ldots, n)$ with size s_i and set E consists of directed nets e_k $(k = 1, \ldots, m)$ with cost c_k denoting the number of connections in the net. A multiple-pin net e_k is characterized by (a_k, b_k) where $a_k \subset V$ are the source nodes of the net and $b_k \subset V$ are the sink nodes of the net. We assume that $|a_k \cup b_k| \geq 2$, $|a_k| \geq 1$ and $|b_k| \geq 1$. Usually, each net has one source node and multiple sink nodes. However, some nets may have multiple sources which share the same interconnect line. Furthermore, one node can be both source node and sink node of the same net. Therefore, a_k and b_k may have nonempty intersection.

For two disjoint node sets X and Y, we shall use $(X \rightarrow Y)$ to denote the directed cut set from X to Y. Therefore, $(X \rightarrow Y)$ contains all the nets $e_k = (a_k, b_k)$ such that X intersects source node set a_k and Y intersects sink node set b_k, i.e., $(X \rightarrow Y) = \{e_k \mid e_k = (a_k, b_k), a_k \cap X \neq \emptyset,\ b_k \cap Y \neq \emptyset\}$. We use function $c(X \rightarrow Y)$ to denote the total cost of the nets in $(X \rightarrow Y)$, i.e., $c(X \rightarrow Y) = \sum_{e_k \in (X \rightarrow Y)} c_k$. We use function $size(X)$ to denote the total size of the nodes in X, i.e., $size(X) = \sum_{i \in X} s_i$.

Partitioning with Replication: Given a network $G = (V, E)$ and three sets of nodes S, R and T such that $S \cap T = \emptyset$ and $R = V - S - T$ as shown in Fig. 2(a), we derive a partitioning with R being the set of replicated nodes as shown in Fig. 2(b). Each copy of R needs to collect a complete set of input signals in order to compute the function properly. Thus, the nets from S to R and from T to R are duplicated. However, the output signals of R can be obtained from either copy of R. For example, nets from R on the right side of the cut to S in Fig. 2(b) are not duplicated because S gets inputs from the R on the left side of cut. For the same reason, we do not replicate the nets from R on the left side of the cut to T. Given two disjoint sets S and T, let a *replication cut* (S, T) denote the cut set of a partitioning with $R = V - S - T$ being duplicated. From Fig. 2(b), we can see that *replication cut* (S, T) is the union of four directed cuts, that is,

$$(S, T) = (S \rightarrow T) \cup (T \rightarrow S) \cup (S \rightarrow R) \cup (T \rightarrow R).$$

Let z_1 and z_2 denote the size limits on the two partitioned subsets. We state the *Replication Cut Problem* as:



(a) (b)

Fig. 2. Replication cut problem. (a) The three sets of nodes S, R, and T. (b) The duplicated network with R being replicated.

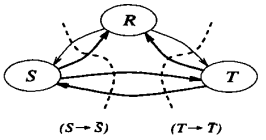


Fig. 3. Two directed cuts $(S \rightarrow \bar{S})$ and $(T \rightarrow \bar{T})$.

Given a directed network G and size limits z_1 and z_2, find a replication cut (S, T) with an objective

$$\min c(S, T) = c((S \rightarrow T) \cup (T \rightarrow S) \cup (S \rightarrow R) \cup (T \rightarrow R))$$

subject to the size constraints

$$\text{size}(S \cup R) \leq z_1, \quad \text{size}(T \cup R) \leq z_2,$$

and the feasible condition

$$S \cap T = \emptyset, \quad R = V - S - T.$$

Interpretation of the Cut Set: Suppose we rewrite the cut set in the format

$$\begin{aligned}(S, T) &= (S \rightarrow R) \cup (S \rightarrow T) \cup (T \rightarrow S) \cup (T \rightarrow R) \\ &= (S \rightarrow \bar{S}) \cup (T \rightarrow \bar{T})\end{aligned}$$

where $\bar{S}$ and $\bar{T}$ denote the complementary sets of S and T, respectively. The cut set becomes the union of $(S \rightarrow \bar{S})$ and $(T \rightarrow \bar{T})$. We can interpret the cut set of the replication cut (S, T) as two directed cuts on the original network G as shown in Fig. 3. Thus the mathematical problem is to find set S and set T with the objective

$$\min c((S \rightarrow \bar{S}) \cup (T \rightarrow \bar{T}))$$

subject to the constraints that $S \cap T = \emptyset$, $size(S \cup R) \leq z_1$ and $size(T \cup R) \leq z_2$, where $R = V - S - T$.

In the case with no size constraints, suppose we are given one node s to be in S and one node t to be in T, it appears that $\min c(S \rightarrow \bar{S})$ could be obtained by finding the maximum flow minimum cut from s to t and $\min c(T \rightarrow \bar{T})$ could be obtained by the maximum flow minimum cut from t to s. It seems that we can simply solve two maximum flow problems to generate an optimum solution of the replication cut problem without size constraints, using s and t as sources respectively.

Unfortunately, the minimum directed cut (obtained by maximum flow) from s and the minimum directed cut (obtained by maximum flow) from t can cross each other, i.e. $S \cap T \neq \emptyset$. For example, the network in Fig. 4 has a minimum directed cut $(S \rightarrow \bar{S}) = (\{s, a\}, \{b, c, t\})$ with cut cost one. And, the minimum directed cut $(T \rightarrow \bar{T}) = (\{t, a, b\}, \{c, s\})$ has cut cost three. Suppose we set $S = \{s, a\}$ and $T = \{t, a, b\}$. Then,

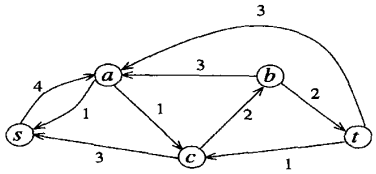


Fig. 4. A network G with five nodes and nine nets.

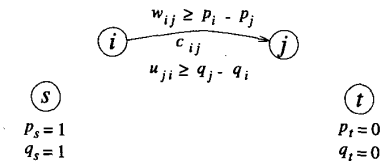


Fig. 5. Each node i is associated with potential p_i and q_i; each net (i, j) is associated with cost c_{ij}, potential difference w_{ij} and u_{ji}.

the two sets S and T are not disjoint, which means that the replication cut (S, T) is not a feasible solution.

III. Replication Cut Separating a Pair of Nodes With Two-Pin Net Model

In this section, we focus on the case of networks containing only two-pin nets and relax the size constraints. For two-pin nets, we denote the cost of net (i, j) from node i to node j by c_{ij}. The size constraints are replaced by the separation of a pair nodes s and t. In other words, the replication cut (S, T) is subject to

$$s \in S, t \in T \quad \text{and} \quad S \cap T = \emptyset.$$

In the following, we first use linear programming to formulate the replication cut problem. Then a dual formulation can be derived by linear programming transformation [13]. We show that the dual formulation corresponds to a network flow problem on a replication graph. In subsection B, we describe the construction of the replication graph and the algorithm to find the optimum solution. The complexity of the algorithm is analyzed in subsection C.

A. Primal Dual Formulation

In this subsection, we first introduce the notations in terms of a network flow problem. We then describe a linear programming formulation of the replication cut problem. Using Lagrangian multiplier, we transform the problem into a dual formulation, which will derive the proposed replication graph.

We adopt the linear programming formulation of network flow problem [1], [13], where each node is assigned a potential and a cut is represented by the difference of node potentials as shown in Fig. 5. With respect to the directed cut $(S \rightarrow \bar{S})$, we use w_{ij} to denote the potential difference between the cut from node i to node j. The potential of each node i is denoted by p_i. For nodes i in S, $p_i = 1$, and for nodes i in $\bar{S}$, $p_i = 0$. Thus all nets (i, j) in the cut set $(S \rightarrow \bar{S})$ have $w_{ij} = 1$. The remaining nets have $w_{ij} = 0$.

With respect to the directed cut $(T \rightarrow \bar{T})$, we use u_{ji} with a reversed subscript ji to denote the potential difference between the cut from node i to node j (Fig. 5). The potential of each node i is denoted by q_i. For nodes i in $\bar{T}$, $q_i = 1$, and for nodes i in T, $q_i = 0$. The potential difference u_{ji} has a reverse direction with net (i, j) because we set the potential on $\bar{T}$ side high and the potential on T side low. Thus, all nets (i, j) in the cut set $(T \rightarrow \bar{T})$ have $u_{ji} = 1$. The remaining nets have $u_{ji} = 0$.

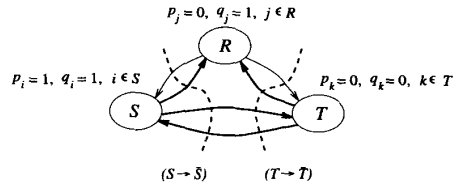


Fig. 6. The p potential and q potential of each node.

Primal Linear Programming Formulation: The problem is to minimize the total cost of crossing nets

$$\text{Obj: min} \sum_{(i,j)\in E} c_{ij} w_{ij} + \sum_{(j,i)\in E} c_{ji} u_{ij} \tag{1}$$

subject to

$$w_{ij} - p_i + p_j \geq 0 \quad \forall (i,j) \in E \tag{2}$$
$$u_{ij} - q_i + q_j \geq 0 \quad \forall (j,i) \in E \tag{3}$$
$$q_i - p_i \geq 0 \quad \forall i \in V, i \neq s, t \tag{4}$$
$$p_s = 1 \tag{5}$$
$$q_s = 1 \tag{6}$$
$$p_t = 0 \tag{7}$$
$$q_t = 0 \tag{8}$$
$$w_{ij}, u_{ji} \geq 0 \quad \forall (i,j) \in E. \tag{9}$$

To minimize objective function (1), the equality of constraint (2) holds, i.e., $w_{ij} = p_i - p_j$, if $p_i \geq p_j$, otherwise, $w_{ij} = 0$. Similarly, constraint (3) requires $u_{ij} = q_i - q_j$ if $q_i \geq q_j$, otherwise, $u_{ij} = 0$. Expression (4) demands potential q_i be not less than potential p_i for any node i in V. Since high potential p_i corresponds to set S, and high potential q_i corresponds to set $\bar{T}$, inequality (4) enforces S be a subset of $\bar{T}$. Consequently, the requirement that $S \cap T = \emptyset$ is satisfied.

Constraints (5)–(8) set the potentials of nodes s and t. Constraint (9) requires potential difference w_{ij} and u_{ij} be nonnegative. Fig. 6 shows one ideal potential configuration of the solution.

Dual Linear Programming Formulation: By assigning dual variables (Lagrangian multiplier) x_{ij} to inequality (2) with respect to each net, x'_{ij} to inequality (3), λ_i to inequality (4) with respect to node i, and a_s, b_s, a_t, b_t to inequalities (5)–(8), respectively, we have the dual formulation as follows

$$\text{Obj: max } a_s + b_s \tag{10}$$

subject to

$$x_{ij} \leq c_{ij} \quad \forall\ (i,j) \in E \tag{11}$$
$$x'_{ij} \leq c_{ji} \quad \forall\ (j,i) \in E \tag{12}$$

$$\sum_j -x_{ij} + x_{ji} - \lambda_i = 0 \quad \forall i \in V, i \neq s,t \tag{13}$$

$$\sum_j -x'_{ij} + x'_{ji} + \lambda_i = 0 \quad \forall i \in V, i \neq s,t \tag{14}$$

$$\sum_j -x_{sj} + x_{js} + a_s = 0 \tag{15}$$

$$\sum_j -x_{tj} + x_{jt} + a_t = 0 \tag{16}$$

$$\sum_j -x'_{sj} + x'_{js} + b_s = 0 \tag{17}$$

$$\sum_j -x'_{tj} + x'_{jt} + b_t = 0 \tag{18}$$

$$\lambda_i, x_{ij}, x'_{ji} \geq 0 \quad \forall i \in V, (i,j) \in E \tag{19}$$

$$a_s, a_t, b_s, b_t: \text{unrestricted.} \tag{20}$$

Inequalities (11) and (12) are derived with respect to each w_{ij} and u_{ij} respectively. Similarly, (13)–(18) are derived with respect to each p_i, q_i, p_s, p_t, q_s and q_t. Eqs. (13–18) hold because p_i, q_i, p_s, p_t, q_s and q_t are not restricted on sign in the primal formulation. Variables λ_i, x_{ij}, and x'_{ji} are required to be nonnegative in (19) because their corresponding expressions (2)–(4) are inequality constraints.

We can view the dual formulation as a network flow problem in $G = (V, E)$ by interpreting c_{ij} as the flow capacity, and x_{ij} as the flow of net (i,j). Constraint (11) requires that the flow x_{ij} be not larger than the flow capacity c_{ij} on each net (i,j). In (12), the set of nets are in a reversed direction and flow x'_{ij} is not larger than the capacity c_{ji} of net (j,i) in E. Corresponding to $G = (V, E)$, we use $G' = (V', E')$ to denote the reversed graph.

Constraint (13) has the total flow x_{ij} injected from node i into G be equal to $-\lambda_i$. On the other hand, constraint (14) has the total flow x'_{ij} injected from node i' into G' be equal to λ_i. Suppose we combine (13) and (14), we have

$$\sum_j -x_{ij} + x_{ji} = \lambda_i = \sum_j x'_{ij} - x'_{ji}. \tag{21}$$

This means that the amount of flow λ_i which emanates from node i in G enters its corresponding node i' in G'.

Constraints (15)–(18) indicate that a_s and b_s are the flow injections to node s in G and s' in its reversed network G'; a_t and b_t are the flow ejections from node t in G and t' in its reversed network G', respectively. Combining network G and G' together, we have the maximum total flow, $a_s + b_s$, be the optimum solution of the minimum replication cut problem.

B. The Optimum Partition

Given a network $G = (V, E)$ and a pair of nodes s and t to be separated, we formally state the construction of replication graph and take an example to describe it. We then apply the maximum flow algorithm on the constructed replication graph to derive an optimum replication cut. The optimality of the derived replication cut is proved by using a network flow approach.

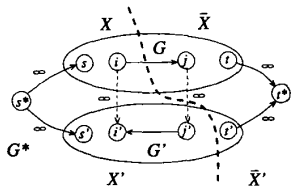


Fig. 7. The replication graph G^*.

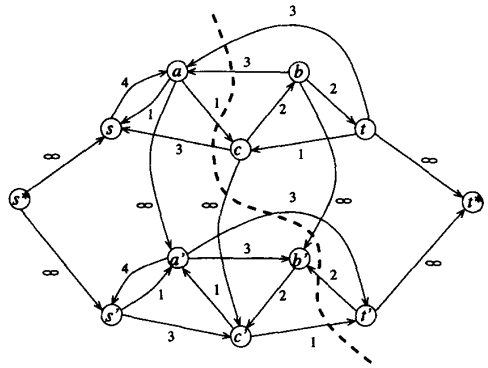


Fig. 8. The constructed replication graph of the network shown in Fig. 4.

Construction of Replication Graph: Given a network $G = (V, E)$ and nodes s and t, we construct another network $G' = (V', E')$ where $|V'| = |V|$ with each node i' in V' corresponding to a node i in V, and $|E'| = |E|$ with each directed net (j', i') in E' in the reverse direction of net (i,j) in E. We create super nodes s^* and t^* and nets (s^*, s), (s^*, s'), (t, t^*), and (t', t^*) with infinite capacity as shown in Fig. 7. From every node i in V except s and t, we add a directed net of infinite capacity to the corresponding node i' in V'. We refer to the combined network as G^*.

Polynomial-Time Algorithm: The optimum replication cut problem with respect to node pair s and t and without size constraints can be solved by a maximum-flow minimum-cut solution of the network G^* with s^* as the source and t^* as the sink of the flow [1]. Suppose the maximum-flow minimum-cut solution partitions V by cut $(X, \bar{X})$ with $s \in X$ and $t \in \bar{X}$ and partitions V' by cut $(X', \bar{X}')$ with $s' \in X'$ and $t' \in \bar{X}'$. Then a replication cut (S, T) of the original network with $S = X$, $T = \{i \mid i' \in \bar{X}'\}$ and $R = V - S - T$ is an optimum solution. Note that T is derived from the cut in node set V'. To simplify the notation, we shall use $(X, \bar{X}')$ to denote the derived replication cut of G.

Example: Given a network in Fig. 4, its replication graph G^* is constructed as shown in Fig. 8. The maximum-flow minimum-cut of G^* derives $(X, \bar{X}) = (\{s, a\}, \{b, c, t\})$ and $(X', \bar{X}') = (\{s', a', b', c'\}, \{t'\})$ with a flow amount, 5 (Fig. 8). Thus the sets $S = \{s, a\}$ and $T = \{t\}$ define an optimum replication cut (S, T) with $R = \{b, c\}$ and a cut cost equal to 5 (Fig. 9).

The following theorem states the optimality of the solution. We use a network flow approach to prove the theorem.

Theorem 1: The replication cut $(S, T) = (X, \bar{X}')$ derived from the replication graph G^* generates the minimum cut cost.

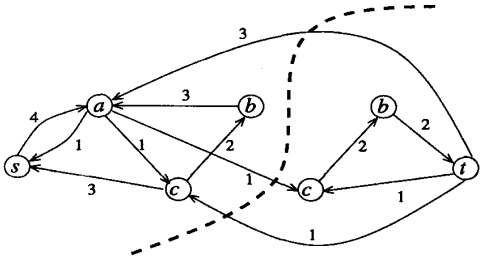


Fig. 9. The duplicated network of the network shown in Fig. 4.

Proof: First we claim that $S \cap T$ is an empty set. This claim can be proved by contradiction. Suppose the claim is not true. Let i be a node in $S \cap T$. In G^*, we have $i \in X$ and $i' \in \bar{X}'$. Then the maximum flow minimum cut of G^* will cut the net (i, i') connecting node i in V and i' in V'. Because net (i, i') is assigned an infinite capacity in G^*, we can thus augment the flow, which contradicts to the property of maximum flow minimum cut.

The above claim indicates that the derived replication cut (S, T) is a feasible solution. Suppose replication cut (A, B) is an optimum solution, we then have

$$c(X, \bar{X}') \geq c(A, B).$$

On the other hand, replication cut (A, B) corresponds to a partition of the replication graph G^* with a cut value $c(A, B)$ equal to the cut capacity on G^*. Because the maximum flow cannot be larger than the capacity of any cuts, we have

$$c(X, \bar{X}') \leq c(A, B).$$

From the above two inequalities, we conclude that

$$c(X, \bar{X}') = c(A, B). \qquad \square$$

C. Algorithm Complexity

Given a network G, because the replication graph G^* consists of the original network and its reversed network G', the construction of G^* takes $O(m + n)$ time, where m and n denote the numbers of nets and nodes of G, respectively,

The partition of the replication graph G^* can be performed by any maximum-flow minimum-cut algorithm. The fastest maximum flow algorithm [8] needs $O(mn \log(n^2/m))$ which dominates the complexity of the algorithm. Therefore, the time complexity of the algorithm is $O(mn \log(n^2/m))$.

IV. Replication Cut With Size Constraints and Multi-Pin Net Model

For VLSI applications, we release the constraint to separate nodes s and t, instead we set an upper limit on the total size of nodes in each partition. We allow that nets of network G contain multiple pins. Thus, the definition of the replication graph G^* needs to be extended for VLSI applications.

Fig. 10 shows the outline of the algorithm. We first construct the replication graph G^*. Then we extend the Fiduccia-Mattheyses (FM) algorithm to search for a directed cut with size constraints on G^*. The cut of G^* finds a heuristic solution of the replication cut.

Step 1: Construct replication graph G^*;

Step 2: Call directed Fiduccia-Mattheyses algorithm to obtain replication cut $(X, \bar{X}')$;

Step 3: Return $(X, \bar{X}')$;

Fig. 10. Algorithm for Directed FM on Replication Graph Partitioning (DFRGP).

A. Extension of Replication Graph G^*

For VLSI applications, the replication graph G^* of G is almost identical to the graph in Section III. The difference is that there is no assigned s and t in G. Therefore, there is no need to create s^* and t^* and their connecting nets in G^*. For multiple-pin nets $e_k = (a_k, b_k) \in E$, we swap the pair in E' of the reversed network G', i.e. $e_{k'} = (a_{k'}, b_{k'})$ with $a_{k'} = b_k$ and $b_{k'} = a_k$.

In practice, the infinite capacity of nets (i, i') connecting node i in V and i' in V' is replaced with cost equal to total net cost of E, i.e., $\sum_{e_k \in E} c_k$. Thus, the cost of nets (i, i') dominates the cost of the rest of nets.

Given a partitioning (V_1^*, V_2^*) of G^* with $V_1^* \cap V_2^* = \emptyset$ and $V_1^* \cup V_2^* = V \cup V'$, let partitioning (V_1^*, V_2^*) of G^* separate V and V' into $(X, \bar{X})$ and $(X', \bar{X}')$, respectively. We define the cut cost to accommodate multiple-pin net model and the size constraint using replication graph G^*.

Objective Function: When we count the nets in the replication cut, unlike a two-pin net, a multiple-pin net e_k in $(X \rightarrow \bar{X})$ may have its corresponding net $e_{k'}$ in $(X' \rightarrow \bar{X}')$. In order to count the cut cost of net e_k only once, we find the nets e_k in E corresponding to the nets $e_{k'}$ in the cut set $(X' \rightarrow \bar{X}')$ in E' and define the set

$$\begin{aligned} U(V_1^* \rightarrow V_2^*) = &\{e_k \mid e_k \in (X \rightarrow \bar{X})\} \\ &\cup \{e_k \mid e_{k'} \in (X' \rightarrow \bar{X}')\} \\ &\cup \{(i, i') \mid i \in V_1^*, i' \in V_2^*\} \end{aligned} \tag{22}$$

Therefore, the objective function is to minimize the cut cost

$$c(U(V_1^* \rightarrow V_2^*)). \tag{23}$$

Size Constraints: Because replication cut (S, T) has $S = X$ and $T = \bar{X}'$, we can derive that $S \cup R = X'$ and $T \cup R = \bar{X}$. Thus, the size constraints can be expressed as

$$\text{size}(X') \leq z_1 \tag{24}$$

and

$$\text{size}(\bar{X}) \leq z_2. \tag{25}$$

B. Directed FM Method Using Replication Graph (DFRG)

We extend FM to a directed FM to partition G^*. Our directed FM is very similar to [9]. However, [9] applies a directed FM method to the original graph to minimize a directed cut cost under size constraints. Our approach applies the directed FM to the proposed replication graph to minimize the replication cut cost.

We set a flag-array of boolean value (i.e., 0 or 1) with a dimension equal to the number of the nets in G^* to keep track

TABLE I
CHARACTERISTICS OF TEST CASES

Circuit	# cells	# nets	# iopads	# pins
C2670	1977	1995	64	5770
C3540	685	678	72	1961
C5315	936	1042	303	2655
C6288	2095	2069	64	6133
C7552	1183	1326	317	3124
Test02	1663	1789	69	8049
Test03	1607	1683	65	7529
Test04	1506	1674	36	7418
Test05	2595	2813	63	12400
Test06	1752	1710	69	7696
Test07	2379	2561	99	11775

of the cut cost $c(U(V_1^* \rightarrow V_2^*))$. Each element of the flag-array is used to indicate if the corresponding net is in the cut set of $(V_1^* \rightarrow V_2^*)$ or not. Note that both net e_k in E and its corresponding net $e_{k'}$ in E' can be in the cut set of $(V_1^* \rightarrow V_2^*)$.

Given a net, we can check if it is in the cut set of $(V_1^* \rightarrow V_2^*)$ or not by using the flag-array in O(1) time. In this sense, the time complexity of our modified FM algorithm on each iteration is still $O(p)$ where p is the number of pins in the network.

Feasible Solution: It is important that the replication cut (S, T) derived by directed FM algorithm satisfies the feasible condition, i.e., $S \cap T = \emptyset$. Suppose we start with a feasible partition, the following theorem states that the final result is also feasible.

Theorem 2: Given a feasible initial partition, the directed FM method always generates a feasible replication cut.

Proof: Suppose the solution is not feasible, i.e., $S \cap T \neq \emptyset$, the cut will include net (i, i'), $i \in S \cap T$, which will make the cut cost larger than the previous cut. Thus, the FM method will return the most recent feasible solution.

V. EXPERIMENTAL RESULTS

We use C2670, C3540, C5315, C6288, and C7552 from ISCAS85 and Test02, Test03, Test04, Test05, Test06, and Test07 from the Microelectronics Center of North Carolina (MCNC) as our test cases. For ISCAS85 test cases, the netlists are obtained after logic minimization. The characteristics of these test cases are listed in Table I. In [9], Hwang and El Gamal perform multiway partitioning. Thus, we do not compare our results with [9]. In [11], the netlists of ISCAS85 are minimized via the standard script MISII [2]. Our experiments on ISCAS85 follow the same methodology as [11]. However, because MISII has been revised, the minimized netlist is different. Therefore, we cannot make direct comparison with [12] either.

We compare our algorithm, Directed FM on Replication Graph Partitioning (DFRG), with the Ratio Cut (RC) [5] and Fiduccia-Mattheyses (FM) [6] algorithms. All algorithms are run on a single-processor SUN SPARC10 workstation under the C/UNIX environment. The results of FM and DFRG are chosen from the best of twenty runs each. Given an area overhead limit of $r\%$, each partition will have size constraint equal to $(50 + r/2)\%$ of the circuit size. For example, an area overhead limit 10% restricts the maximum size of each partition to 55% of the network size and node replication to 10% of the total size in the given network. We do experiments on two different area overhead limits 30% and 10% for ISCAS85 test cases. For MCNC test cases, we set three different area overhead limits 50, 20, and 11%.

TABLE II
COMPARISON OF RESULTS FOR THREE ALGORITHMS ON ISCAS85 TEST CASES WITH AN AREA OVERHEAD LIMIT 30%

	RC		FM		DFRG			Improvement	
Circuit	Cut	CPU	Cut	CPU	Cut	CPU	Area Overhead	RC	FM
C2670	15	21	30	51	10	68	6.67%	33%	67%
C3540	26	7	37	8	6	18	27.49%	77%	84%
C5315	17	11	23	16	1	24	22.94%	94%	96%
C6288	34	22	48	53	9	90	18.48%	74%	81%
C7552	6	11	9	18	1	25	7.50%	83%	89%

TABLE III
COMPARISON OF RESULTS FOR THREE ALGORITHMS ON ISCAS85 TEST CASES WITH AN AREA OVERHEAD LIMIT 10%

	RC		FM		DFRG			Improvement	
Circuit	Cut	CPU	Cut	CPU	Cut	CPU	Area Overhead	RC	FM
C2670	16	22	25	98	13	157	9.96%	19%	48%
C3540	35	7	40	12	18	23	9.36%	49%	55%
C5315	23	12	24	21	15	34	2.01%	35%	38%
C6288	35	23	61	99	10	152	9.87%	71%	84%
C7552	8	11	10	26	3	44	2.10%	63%	70%

Table II shows the results of our experiments for ISCAS85 test cases with an area overhead limit 30%. The data in subcolumns *Cut* and CPU represent the crossing net count and execution time, respectively. The unit for the CPU time is in seconds. The subcolumn, *Area Overhead*, shows the percentage of the total size replicated by DFRG. The last column shows the improvement on the crossing net count achieved by DFRG compared with RC and FM. When compared with RC, DFRG can achieve 33% to 94% with an average of 72% improvement on the number of crossing nets. Compared with FM, DFRG obtains 67% to 96% with an average of 83% reduction in crossing net count. In terms of area overhead, DFRG needs 6.67% to 27.49% with an average of 16.61%. For most test cases, DFRG can achieve over 80% reduction in the number of crossing nets by allowing replication, compared with the FM approach.

Table III shows the results of our experiments for ISCAS85 test cases with an area overhead limit 10%. When compared with RC, DFRG can achieve 19% to 71% with an average of 47% improvement on the number of crossing nets. When compared with FM, DFRG obtains 38% to 84% with an average of 59% reduction in crossing net count. DFRG needs 2.01% to 9.96% with an average of 6.66% area overhead. These experimental results demonstrate that we can replicate a small percentage of the network size to reduce crossing net counts dramatically. Let us take C7552 as an example. FM obtains a crossing net count 10, while DFRG generates a cut of 3 with 2.10% area overhead. In this case, our algorithm achieves a 70% reduction in the crossing net count.

Table IV shows the results of our experiments on MCNC test cases with an area overhead limit 50%. When compared with RC, our algorithm DFRG achieves 8% to 39% with

TABLE IV
COMPARISON OF RESULTS FOR THREE ALGORITHMS ON MCNC TEST CASES WITH AN AREA OVERHEAD 50%

	RC		FM		DFRG			Improvement	
Circuit	Cut	CPU	Cut	CPU	Cut	CPU	Area Overhead	RC	FM
Test02	42	24	47	55	33	77	37.20%	21%	30%
Test03	25	23	30	31	23	58	48.39%	8%	23%
Test04	41	22	44	40	25	68	19.69%	39%	43%
Test05	42	43	42	96	34	146	4.45%	19%	19%
Test06	23	27	25	25	17	75	33.90%	26%	32%
Test07	45	48	54	77	29	142	44.98%	36%	46%

TABLE V
COMPARISON OF RESULTS FOR THREE ALGORITHMS ON MCNC TEST CASES WITH AN AREA OVERHEAD LIMIT 20%

	RC		FM		DFRG			Improvement	
Circuit	Cut	CPU	Cut	CPU	Cut	CPU	Area Overhead	RC	FM
Test02	75	24	78	74	63	82	17.49%	16%	19%
Test03	44	23	52	48	38	64	4.59%	14%	27%
Test04	45	22	51	51	29	135	6.03%	36%	43%
Test05	59	42	62	136	49	270	4.19%	17%	21%
Test06	36	27	35	45	29	101	19.77%	19%	17%
Test07	83	49	76	119	41	355	18.46%	51%	46%

an average of 24% reduction in the number of crossing nets. When compared with FM, DFRG obtains 19% to 46% with an average of 32% reduction in the crossing net count. Take Test04 as an example. RC generates a partitioning with crossing net count of 41. FM produces a result with crossing net count of 44. The result of DFRG algorithm yields a crossing net count of 25 with 19.69% of the total size being duplicated in the partitioning. In Test04, DFRG achieves 43% reduction in the number of crossing nets, compared with FM.

Table V shows the results of our experiments with area overhead limit 20%. In terms of the crossing net count, DFRG achieves 14% to 51% with an average of 25% reduction compared with RC and 17% to 46% with an average of 29% reduction compared with FM. The average area overhead is 11.76%. For example, DFRG generates a partitioning with crossing net count 41 with an area overhead 18.64% on Test07. On the other hand, FM and RC get partitioning with crossing net count of 76 and 83, respectively. By duplicating 18.64% of the network size, we achieve 51% improvement on the number of crossing nets compared with RC. The experimental results for area overhead limit 11% are listed in Table VI. DFRG obtains 2% to 32% with an average of 19% reduction on the number of crossing nets comparing with RC. When compared with FM, DFRG achieves 10% to 45% with an average of 27% reduction in crossing net count. The area overhead is 9.10% on average. But DFRG obtains an average of 27% reduction in terms of crossing net count compared with FM. Even compared with better two-way partitioning algorithm RC, DFRG still achieves an average of 19% reduction.

Overall, among Tables IV, V, and VI, the cut decreases with area overhead limit; the improvement by DFRG over FM decreases from 32%, to 29% to 27%, and the overhead decreases from 31.44%, to 11.76% to 9.10% on average.

Execution Time: Given a network G, DFRG first constructs its replication graph G^*. Since G^* contains more nodes and nets than G, DFRG will take more CPU time than FM. For example, DFRG takes 355 s for Test 07 in Table V while FM needs only 119 s. However, we improve FM's result by 46%.

TABLE VI
COMPARISON OF RESULTS FOR THREE ALGORITHMS ON MCNC TEST CASES WITH AN AREA OVERHEAD LIMIT 11%

	RC		FM		DFRG			Improvement	
Circuit	Cut	CPU	Cut	CPU	Cut	CPU	Area Overhead	RC	FM
Test02	81	25	88	63	79	221	4.55%	2%	10%
Test03	52	24	71	54	44	97	10.84%	15%	38%
Test04	50	22	62	50	34	139	10.27%	32%	45%
Test05	62	43	66	125	51	385	10.57%	15%	23%
Test06	41	27	41	55	34	157	7.28%	17%	17%
Test07	88	49	91	115	60	497	11.00%	32%	34%

VI. CONCLUDING REMARKS

In this paper, we target the general replication problem which is not restricted to any prior partitioning. We present a polynomial-time algorithm to solve the replication cut problem with only two-pin nets and without size constraints. A directed Fiduccia–Mattheyses method is used to solve the problem with multiple-pin nets and size constraints. Experimental results show that we can trade area overhead for the reduction of crossing net count by the proposed replication cut.

ACKNOWLEDGMENT

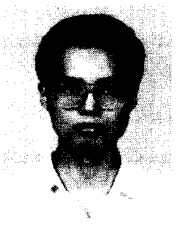
The authors would like to thank C. Kring and the reviewers for their helpful suggestions.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice Hall, 1993.
[2] 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, pp. 1062–1081, Nov. 1987.
[3] M. A. Breuer, "Min cut placement," *J. Design Automation and Fault Tolerant Computing*, vol. 1, no. 4, pp. 343–362, Oct. 1977.
[4] H. R. Charney and D. L. Plato, "Efficient partitioning of components," in *Proc. 5th Design Automation Wkshp.*, 1968, pp. 16–21.
[5] C.-K. Cheng and Y.-C. Wei, "An improved two-way partitioning algorithm with stable performance," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1502–1511, Dec. 1991.
[6] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. 19th ACM/IEEE Design Automation Conf.*, 1982, pp. 175–181.
[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman, 1979.
[8] A. W. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," *J. ACM*, vol. 35, no. 4, pp. 921–940, Oct. 1988.
[9] J. Hwang and A. El Gamal, "Optimal replication for min-cut partitioning," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1992, pp. 432–435.
[10] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291–307, Feb. 1970.
[11] C. Kring, personal communication, 1994.
[12] C. Kring and A. R. Newton, "A cell-replicating approach to mincut based circuit partitioning," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1991, pp. 2–5.
[13] D. G. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.
[14] 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. C-20, pp. 1455–1462, Dec. 1971.

Lung-Tien Liu received the B.S. and M.S. degrees in 1987 and 1989 from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree from the University of California at San Diego in 1994, all in computer science.

Since July 1994, he has been with AT&T Bell Laboratories, Murray Hill, NJ, as a technical staff member. His research interests include optimization algorithms for CAD, timing-driven placement, and circuit partitioning.



Ming-Ter Kuo received the B.S. and M.S. degrees from National Taiwan University in 1987 and 1989, respectively, and the M.S. degree in computer science from the University of Iowa in 1993.

Currently, he is a Ph.D. student in computer science and engineering at the University of California, San Diego. His research interests include circuit partitioning, finite state machine decomposition and optimization.

Chung-Kuan Cheng (S'82-M'84-SM'95) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, and the Ph.D. degree in electrical engineering and computer sciences from University of California, Berkeley in 1984.

From 1984 to 1986 he was a senior CAD engineer at Advanced Micro Devices Inc. In 1986, he joined the University of California, San Diego, where he is currently an Associate Professor in the Computer Science and Engineering Department. His research interests include network optimization and design automation on microelectronic circuits.

T. C. Hu received the B.S. degree in engineering from National Taiwan University in 1953 and the M.S. degree in engineering from the University of Illinois in 1956 and the Ph.D. degree in 1960.

Presently, he is a Professor of Computer Science and Engineering at the University of California, San Diego. He is the author of *Integer Programming and Network Flows* (translated into German, Russian, and Japanese) (Addison-Wesley). He is co-editor of *Mathematical Programming* with S.M. Robinson (Academic Press) and *Theory and Concepts of Circuit Layout* with E.S. Kuh. He also has more than 70 technical papers published. His current interests are computer-aided design and combinatorial algorithms.