Network Flow Based Multi-way Partitioning with Area and Pin Constraints

Huiqun Liu and D. F. Wong

Abstract—

Network flow is an excellent approach to finding min-cuts because of the celebrated max-flow min-cut theorem. However, for a long time, it was perceived as computationally expensive and deemed impractical for circuit partitioning. Only until recently, FBB [1,2] successfully applied network flow to two-way balanced partitioning and for the first time demonstrated that network flow was a viable approach to circuit partitioning. In this paper, we present FBB-MW, which is an extension of FBB, to solve the problem of multi-way partitioning with area and pin constraints. Experimental results show that FBB-MW outperforms previous approaches for multi-FPGA partitioning. In particular, although FBB-MW does not employ logic replication and logic re-synthesis, it still outperforms [5,6] which allow replication and re-synthesis for optimization.

I. Introduction

Circuit partitioning is a critical optimization in many subfields of VLSI CAD: any top-down hierarchical approach to system design must rely on some underlying partitioning techniques. The partitioning solutions greatly influence both the feasibility and quality of the automatic placement and routing algorithms.

Multi-way partitioning is becoming very important with the ever increasing system size and the popularity of hierarchical design. A target device such as an FPGA usually has an upper bound for both the area and the total number of I/O pins. For multiple-FPGA design such as in logic emulation systems, the objective for circuit partitioning is to minimize the total number of interconnecting nets while satisfying both the area and pin constraints. Traditional multi-way partitioning algorithms which only minimize the total cut nets are no longer applicable.

Recent research in multiple-FPGA systems has addressed objectives which minimize the number of cut nets between partitions subject to area and pin constraints. Woo and Kim [3] proposed a modification of the Fiduccia-Mattheyses (FM) method to perform k-way partitioning, which tries to minimize the total number of pins of all partitions while satisfying the given area and pin constraints. Kužnar and Brglez [4] proposed an algorithm to partition a given configurable logic block (CLB) level netlist into multiple types of FPGA devices to minimize total cost, where each FPGA type in a given library can have distinct price, size and pin capacity. Their method recursively applies a variant of FM bipartitioning which allows some uphill moves. In subsequent work, Kužnar and Brglez [5] allow CLBs to be replicated, *i.e.* they introduce functional replication to minimize the inter-device interconnection and total cost of devices. Kužnar and Brglez [6] proposed a greedy heuristic which combines logic re-synthesis with replication-based partitioning. Chou *et al.* [7] have proposed an algorithm to partition a circuit into a single type of FPGAs, such that the number of FPGAs is minimized. They use "local ratio-cut" clustering to reduce the circuit complexity, then derive a partition by using a set covering approach. Chan *et al.* [10] developed a spectral approach for multi-way partitioning into heterogeneous FPGAs. Huang and Kahng [11] proposed an algorithm which incorporates ordering, clustering and dynamic programming to achieve good partitioning results.

Network flow is an excellent approach to finding min-cuts because of the celebrated max-flow min-cut theorem [14]. However, for a long time, it was perceived as computationally expensive and deemed impractical for circuit partitioning. Only until recently, FBB [1,2] successfully applied network flow to two-way balanced partitioning and for the first time demonstrated that network flow was a viable approach to circuit partitioning. Later, [9] improved FBB by introducing node-selection heuristics based on linear placement of the nodes.

In this paper we present FBB-MW, which is an extension of FBB, to solve the problem of multi-way partitioning with area and pin constraints. We first give an improvement of FBB by finding the "most desirable" min-cut during every iteration of FBB. This is based on the observation that there are usually many min-cuts in the flow network after a maximum-flow computation. Utilizing the "most desirable" min-cut reduces the number of iterations in FBB which would in turn reduce final cut-size. We also improve the net modeling in FBB by distinguishing the multi-terminal and two-terminal nets. In FBB-MW, the techniques in FBB and its improvements are applied to multi-way partitioning. While FBB only minimizes the number of cut nets without taking into consideration of the total number of pins for each partitioned component, in order to satisfy the area and pin limits, we must consider both the primary I/O nodes and the interconnecting nets which will occupy the I/O pins. By a suitable network modeling of the external I/O nodes, we can minimize the total number of pins for one component by maximum flow computation. Experimental results show that FBB-MW outperforms previous multi-FPGA partitioning approaches [4,5,6,7,11,15]. The efficient implementation of

H. Liu and D. F. Wong are with the Department of Computer Sciences, University of Texas at Austin, TX 78712. Email: {hqliu, wong}@cs.utexas.edu. This work was partially supported by the Texas Advanced Research Program under Grant No. 003658288.

FBB-MW enables it to partition a circuit with more than $25K$ nodes within a few minutes of CPU time.

The organization of this paper is as follows. Section 2 outlines the FBB method and presents our improvement to it in two aspects: finding a desirable min-cut and improving the modeling of two-terminal nets. This forms the basis of our later discussion of the multi-way partitioning algorithms. In Section 3, we first give the formulation for the multi-way partitioning problem with area and pin constraints, then propose three network flow based algorithms which are extensions of FBB to multi-way partitioning. FBB-MW is a combination of algorithms 1 and 2 and yields better results. Section 4 shows the experimental results comparing FBB-MW with previous multi-FPGA partitioning algorithms.

II. Improvement of FBB

A. Overview of FBB

Network flow based partitioning method was once overlooked to be a practical partitioning method because of its relatively high complexity. Recently, [1,2] proposed FBB algorithm for flow based balanced circuit bipartitioning. FBB proposed a method for exactly modeling a netlist by a flow network, and a balanced bipartitioning heuristic based on a repeated max-flow min-cut computation. By proper net-modeling and employing incremental flow computation, FBB not only yields excellent partitioning results, but also is efficient in computation time.

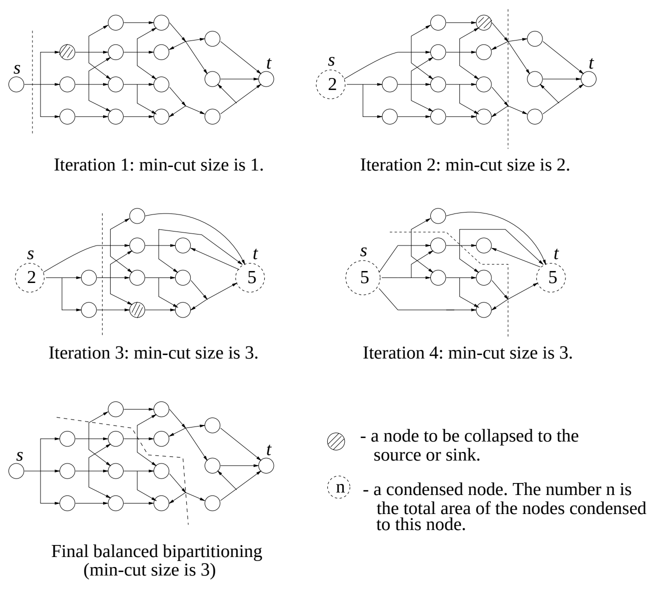

s
t
Iteration 1: min-cut size is 1.
s
2
t
Iteration 2: min-cut size is 2.
s
2
t
5
Iteration 3: min-cut size is 3.
s
5
t
5
Iteration 4: min-cut size is 3.
s
t
Final balanced bipartitioning (min-cut size is 3)
- a node to be collapsed to the source or sink.
n
- a condensed node. The number n is the total area of the nodes condensed to this node.

Fig. 1. Using FBB for balanced bipartitioning

A flow network $F = (V, E)$ is a directed graph in which each edge $e \in E$ has a capacity $c(e) \geq 0$. Two nodes s and t in V are specified: s is called the *source*, t is called the *sink*. An *s-t flow* in F is a real-valued function $f : E \rightarrow R$ such that (1) for each $e \in E$, $0 \leq f(e) \leq c(e)$, and (2) for each $u \in V - \{s, t\}$, the sum of the incoming flow into u is equal to the sum of the outgoing flow from u. An edge is *saturated* if $f(e) = c(e)$. The *value* of a flow f is defined as the sum of the flow outgoing from s, which is equal to the sum of flow incoming to t. A *maximum flow* (or *max-flow* for short) in F is a flow of maximum value from s to t.

An *s-t cut* $(X, \overline{X})$ of a flow network $F = (V, E)$ is a bipartition of V into X and $\overline{X}$ such that $s \in X$ and $t \in \overline{X}$. An edge whose starting node is in X and ending node in $\overline{X}$ is called a $forward\ edge$. An edge whose ending node is in X and starting node is in $\overline{X}$ is called a $backward\ edge$. The *capacity* of a cut $(X, \overline{X})$ is the sum of the capacities on the edges from X to $\overline{X}$. An *augmenting path* from u to v is a sequence of consecutive edges that can be used to push additional flow from u to v. The well-known max-flow min-cut theorem [14] says that the value of a maximum flow is equal to the capacity of a min-cut on the flow network.

FBB applies an efficient max-flow min-cut heuristic to repeatedly cut the network to meet the area constraint. After one max-flow computation, a min-cut partitions the flow network into two parts. When the area of the two parts are not balanced, another iteration of max-flow computation is applied until a balanced min-cut is found. The repeated max-flow min-cut process was implemented efficiently by using incremental flow computation. It is not necessary to do the max-flow computation from scratch in each iteration, only additional flow is added to saturate the bridging edges from iteration to iteration. It was proved in [1,2] that the repeated max-flow min-cut process takes the same asymptotic time complexity as that of one max-flow computation.

Figure 1 shows an example of using FBB for balanced bipartitioning. For simplicity, the net modeling is not shown in the figure. In each iteration, max-flow is computed and a min-cut is found. Then all the nodes on the smaller side of the min-cut are condensed to form one seed node and a new node from the other side is collapsed to this seed node, so that more flows can be pushed through the network. This process goes on until a balanced partition is found. FBB applies to both combinational and sequential circuits.

B. Improvement of FBB

B.1 Finding the most desirable min-cut

In each iteration of FBB, after obtaining the max-flow, FBB used $(X_s, \overline{X_s})$ as the min-cut, where $X_s = \{v | \exists$ an augmenting path from s to $v\}$. An important observation is that there usually exist more min-cuts in the flow network (as shown in Figure 2). Besides X_s, $(X_t, \overline{X_t})$ defines a min-cut that is closest to the sink where $\overline{X_t} = \{v | \exists$ an augmenting path from v to $t\}$ and there may exist more min-cuts in between X_s and X_t. It is easy to show that for any min-cut $(X, \overline{X})$, $X_s \subseteq X \subseteq X_t$.

One improvement to FBB is that after the max-flow computation in each iteration, we try to find the min-cut that cuts the network into subsets with area as close to the area limit as possible. We observe that when collapsing a node to the source (or sink) and then pushing additional flow, the min-cut size is monotonically increasing. By first exploring the existing min-cuts and finding one closest to the area limit, we can obtain a subset with a larger area without increasing the min-cut size.

A min-cut partitions a flow network with total area A into two parts: X and $\overline{X}$, where the source $s \in X$ and the sink $t \in \overline{X}$. Let $\tilde{A}$ be the area constraint (*e.g.* $\tilde{A}= 0.5A$ for balanced bipartitioning). Let $\delta = min(|\tilde{A}-area(X)|, |\tilde{A}-area(\overline{X})|)$ for a min-cut $(X, \overline{X})$. The value δ measures the deviation of the partition from the specified area limit. Among all the min-cuts in the flow network after one max-flow computation, the one with minimum δ is called *the most desirable min-cut*, which is a min-cut closest to the area limit $\tilde{A}$.

In Figure 2, min-cut $C_1 = (X_s, \overline{X_s})$ and $C_5 = (X_t, \overline{X_t})$. C_2, C_3, C_4 are other min-cuts in the flow network. For example, if the area limit $\tilde{A}$=10, then C_2 which partitions the network into two subsets with area 10 and 10, would be the most desirable min-cut. Also, if $\tilde{A}$=8, then C_4 which partitions the network into subsets of area 13 and 7 would be a min-cut that is closest to the area limit.

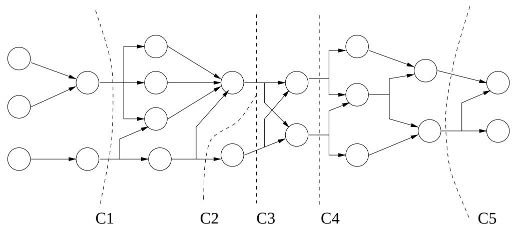


Fig. 2. There are multiple min-cuts in the flow network partitioning the network into subsets of different area.

After obtaining max-flow in the network, all the existing min-cuts partition the flow network F into non-overlapping subsets. We define a *min-cut graph* H which is derived from F as follows:

1. Each subset separated by the min-cuts in F is represented by a node in H.
2. If all edges from subset s_1 to s_2 in F are saturated and all edges from subset s_2 to s_1 in F have zero flow, then add an edge from node s_1 to node s_2 in H.
3. Each node s in H is associated with an area which is equal to the total area of all the nodes in the subset represented by s.

Figure 3 shows the min-cut graph H corresponding to the network in Figure 2. The five min-cuts in F (Figure 2) partition the network into six subsets, each of which is represented by a node in H (Figure 3).

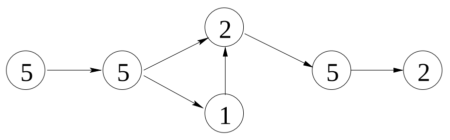


Fig. 3. The min-cut graph H corresponding to the network in Figure 2. The number inside each node is the total area of the subset represented by the node.

Lemma 1: *The min-cut graph H is a directed acyclic graph.*

Proof: We prove the lemma by contradiction. If there exists a loop in H, then for any cut on the loop, there are edges crossing the cut in opposite directions. Let C be the corresponding cut in F, then there are saturated edges (or edges with zero flow) crossing C in opposite directions. Thus C can not be a min-cut or part of a min-cut in F. So all the nodes on the loop can not be partitioned by any min-cut. This leads to a contradiction with the fact that the subsets in F corresponding to the nodes on the loop in H are separated by min-cuts. Therefore, H must be an acyclic graph. □

A *unidirectional cut* of H is defined as a cut that partitions H into $(Y, \overline{Y})$, such that all the cut edges are directed from nodes in Y to nodes in $\overline{Y}$. Note that the set of unidirectional cuts in Figure 4 corresponds to the set of min-cuts in Figure 2. This is true in general and we have the following lemma.

Lemma 2: *There is a 1-1 correspondence between the set of unidirectional cuts in H and the set of min-cuts in F.*

Proof: First we prove that any unidirectional cut in H corresponds to a min-cut in F. Let $(P, \overline{P})$ be a unidirectional cut in H, then by definition there are only edges going from P to $\overline{P}$. Let $(X, \overline{X})$ be the corresponding cut in F and let T_s be the subset of nodes in F represented by node s in H, then $X = \cup_{s \in P} T_s$ and $\overline{X} = \cup_{s \in \overline{P}} T_s$. By the definition of an edge in H and the fact that $(P, \overline{P})$ is unidirectional, the edges in F from X to $\overline{X}$ must be saturated and the edges from $\overline{X}$ to X must have zero flow. Therefore, $(X, \overline{X})$ is a min-cut of F. Conversely, let $(X, \overline{X})$ be a min-cut in F. Clearly, $(X, \overline{X})$ induces a cut $(P, \overline{P})$ in H. Since $(X, \overline{X})$ is a min-cut, all edges from X to $\overline{X}$ are saturated and all edges from $\overline{X}$ to X have zero flow. It then follows from the definition of an edge in H that all edges in $(P, \overline{P})$ are directed from P to $\overline{P}$, and hence $(P, \overline{P})$ is unidirectional. □

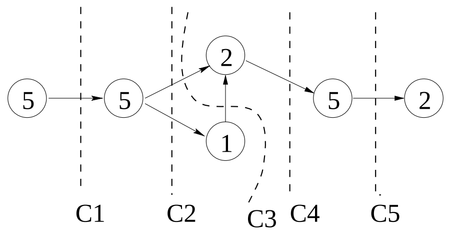


Fig. 4. The set of unidirectional cuts in the min-cut graph H

To find a most desirable min-cut, *i.e.* a min-cut which is closest to the area limit, we can first construct the min-cut graph H and then find a unidirectional cut which partitions the network into subsets with a desirable area. To speed up the computation time, instead of constructing the exact H, we use a greedy heuristic **DMC** to build a min-cut graph and find a unidirectional cut with area close to the area constraint.

```
Procedure DMC: finding a desirable min-cut
1. Let $X_0 = \{v | \exists$ an augmenting path from $v$ to $t\}$;
   Mark all nodes in $X_0$;
2. Let $X_1 = \{v | \exists$ an augmenting path from $s$ to $v\}$;
   Mark all nodes in $X_1$;
   $i = 2$;
3. Select an unmarked node $v$ incident to $X_j (1 \leq j < i)$;
   $X_i = \{u | \exists$ an augmenting path from $v$ to $u$
           and $u$ is unmarked$\}$;
   Mark all nodes in $X_i$;
4. $i = i + 1$;
   If all nodes are marked, then goto step 5; else goto step 3;
5. Select $k$ with $\sum_{i=1}^{k} area(X_i) \leq \tilde{A}$ and $max(\sum_{i=1}^{k} area(X_i))$;
   let $X = \cup_{i=1}^{k} X_i$ ;
   return $X$;
```

From steps 1 to 4 a min-cut graph is constructed, but it is not the exact H as defined above since some nodes in H may be merged into one node here. A simple strategy is given in step 5 to find a min-cut that has area close to the area limit. It can be easily proved that $X = \cup_{i=1}^{k} X_i$ is a unidirectional cut in the min-cut graph H and thus a min-cut in the flow network. More intelligent strategies can be used in step 5 for finding a desirable min-cut by searching the min-cut graph. In addition to checking the total area, the total number of pins for X can also be calculated and compared with the pin limit.

When the size of the network is much larger than the area limit, we do not need to construct H on the whole network. Instead, we can build a partial min-cut graph as follows. In step 5, if $\sum_{i \geq 1} area(X_i)$ exceeds a certain limit, we leave the rest of the nodes to be in one subset of the min-cut graph.

Note that FBB iteratively find min-cuts in order to satisfy the area limit, therefore procedure **DMC** has to be repeatedly called to find a bi-partition.

B.2 Net Modeling

To make the network flow based method suitable for circuit partitioning, [1,2] gave an exact net modeling of the hyperedges so that by max-flow min-cut computation, the min-cut found is the total number of cut nets. Here we further improve the net modeling in FBB by simplifying the modeling of the two-terminal nets. Let $G = (V, E)$ be a netlist, where V is a set of nodes and E is a set of nets which connect nodes in V. A flow network $G' = (V', E')$ can be constructed from G as follows (see Figure 5):

1. V' contains all the nodes in V.

2. For a multi-terminal net $n = (v_1, v_2, ...v_k)$ in G, add two nodes n_1, n_2 in V' and add a bridging edge (n_1, n_2) in E' with capacity 1. For each node $v \in \{v_1, v_2, ..., v_k\}$ incident on net n, add two edges (v, n_1) and (n_2, v) with capacity ∞ in E'.

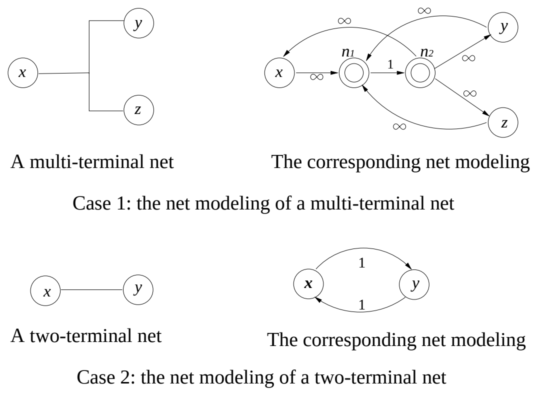

x
y
z
A multi-terminal net
x
n1
n2
y
z
∞
1
The corresponding net modeling
Case 1: the net modeling of a multi-terminal net
x
y
A two-terminal net
x
y
1
The corresponding net modeling
Case 2: the net modeling of a two-terminal net

Fig. 5. Net modeling

3. For a two-terminal net $n = (u, v)$ in G, add two bridging edges (v, u) and (u, v) in E', each with capacity 1.

4. For each node $v \in V$, its corresponding node in V' has the same area as in V. For a node in V' but not in V, assign its area as 0.

Note that FBB [1,2] uses the net modeling in step 2 above for both two-terminal and multi-terminal nets. Here we distinguish between a two-terminal net and a multi-terminal net, so that we do not need to add the extra two nodes and the extra edges for a two-terminal net. Since most of the nets in a circuit are two-terminal nets, this new net modeling significantly reduces the size of the resulting network and thus speed up the max-flow computation.

The capacity on each of the two bridging edges between the two nodes v_1, v_2 in a two-terminal net n is 1. If net n is cut such that $v_1 \in X'$ and $v_2 \in \overline{X'}$, then the bridging edge $v_1 \rightarrow v_2$ must be saturated with flow 1 and the bridging edge $v_2 \rightarrow v_1$ must have flow 0. Thus this net contributes one unit of flow in the maximum-flow and is counted as one cut net. Therefore the new modeling of two-terminal nets is correct.

Our multi-way partitioning algorithms in Section 3 are based on the above net modeling. Since the modeling for multi-terminal nets has been proved to be correct in [1,2], we have the following lemma.

Lemma 3: *Let $(X', \overline{X'})$ be a min-cut of capacity C in G' and $(X, \overline{X})$ be the corresponding cut in G, we have $(X, \overline{X})$ is a minimum net cut in G and the number of cut nets is equal to C.*

III. Multi-way partitioning

In this section, we introduce algorithm FBB-MW, an extension of FBB to multi-way circuit partitioning with area and pin constraints. For multi-way partitioning with area and pin constraints, partitioning heuristics which only minimize the total number of interconnections will no longer be applicable for the following two reasons. First, they can not guarantee that each partitioned component meet the pin limit as the cut nets may be distributed unevenly among the components even if the total is minimized. Secondly, besides the cut nets, the primary I/O nodes will also occupy the I/O pins and therefore should be taken into consideration. In addition, for multi-FPGA design, it is desirable to find a partition that uses as few FPGA devices as possible in order to reduce the total cost of the design. In this section, we will first give the problem formulation and then present our network flow based algorithms.

A. Problem Formulation

For a netlist $G = (V, E)$, V is a set of nodes with each node $v \in V$ associated with an area $area(v)$, E is a set of nets where a net is a subset of V. Given the upper bound for both the total area ($\tilde{A}$) and the number of pins ($\tilde{P}$) for one subset, the multi-way partitioning problem is to partition V into k non-overlapping subsets $V_1, V_2, ..., V_k$, such that

(1) $V = \cup_{i=1}^{k} V_i$;
(2) $area(V_i) \leq \tilde{A}$, $1 \leq i \leq k$;
(3) $pin(V_i) \leq \tilde{P}$, $i \leq i \leq k$;

with the objective of minimizing k and $\sum_{i=1}^{k} pin(V_i)$.

Each subset V_i is also called a component. Here $area(V_i) = \sum_{v \in V_i} area(v)$ and $pin(V_i)$ is the total number of pins for component V_i. The objective is to minimize the number of components and to minimize the total number of pins while each component must satisfy the specified area and pin constraints. Notice that the total pins for one component is comprised of pins for both the interconnecting nets among the components and the primary I/Os.

B. Algorithm 1

A direct extension of FBB to multi-way partitioning is to iteratively apply the max-flow min-cut process to find one component at a time that meets the area and pin constraint, until every node in G is assigned to a component. A *feasible component* V_i is a subset of V, which satisfies both the area and pin constraints, *i.e.* $area(V_i) \leq \tilde{A}$ and $pin(V_i) \leq \tilde{P}$. In order to partition the netlist into as few components as possible, it is desirable to find a large feasible component at a time.

A flow network $G' = (V', E')$ is first constructed from netlist $G = (V, E)$ by the net modeling method discussed in section 2.2.2. **FC** is an algorithm which repeatedly compute max-flow min-cuts in G' for finding a feasible component with area as large as possible.

```
    Procedure FC: finding a feasible component in $G'$.
1.  Select a source $s$ and a sink $t$; $F = \phi$;
2.  Compute max-flow in the flow network $G'$;
3.  Call procedure DMC to find a desirable min-cut
    $C = (X, \overline{X})$;
4.  If $C \geq \tilde{P}$, then return $F$;
    $assign(X, F)$; $assign(\overline{X}, F)$;
5.  if $area(X) < \tilde{A}$, then
        collapse nodes in $X$ to $s$;
        collapse to $s$ a node $v \in \overline{X}$ incident on $s$;
    else if $area(X) \geq \tilde{A}$, then
        collapse nodes in $\overline{X}$ to $t$;
        collapse to $t$ a node $v \in X$ incident on $t$;
6.  If all nodes are collapsed to $s$ or $t$, then
        return $F$;
    else goto step 2;
```

In step 2, the maximum flow is pushed through the network. Incremental flow computation is employed, as only additional flow is added to saturate the edges from iteration to iteration. Procedure **DMC** is called in step 3 to find a desirable min-cut. In step 4, F is used to store the best feasible subset that has been found so far. In function $assign(X, F)$, if $pin(X) \leq \tilde{P}$ and $area(F) < area(X) \leq \tilde{A}$, then F is replaced by X since X is a larger feasible subset than the previous F. The min-cut calculated in step 3 is the number of cut nets between X and $\overline{X}$, not including the primary I/Os and the nets which are connected to the earlier partitioned components. So in function $assign(X, F)$, we have to add the number of these external connections to the min-cut size to obtain the total number of pins. The nodes on one side of the min-cut are condensed to one seed node in step 5. Steps 2 to 5 are repeated to find the next desirable min-cut by pushing more flow through the network. If the min-cut size exceeds the pin limit or if all nodes have been collapsed either to s or t, then procedure **FC** terminates and F is returned as the largest feasible component that is found,

Similar to the time complexity analysis of FBB [1,2], we can show that it takes $O(|E|)$ time to find one augmenting path and the maximum number of augmenting paths is $O(|V|)$. Therefore the time complexity of finding a feasible component by **FC** is $O(|V||E|)$.

Algorithm 1 is designed for multi-way partitioning. First, a flow network G' is constructed from netlist G, then procedure **FC** is repeatedly called to find one feasible component at a time. After one component F is found, G' is modified to be the rest of the network (*i.e.* $G' = G' - F$). The flow on the edges in network G' is reset to zero before finding the next feasible component. The time complexity of algorithm 1 is $O(k|V||E|)$ with k being the number of partitioned components.

C. Algorithm 2

In procedure **FC** of algorithm 1, after the max-flow computation in each iteration, the resulting min-cut size measures the number of cut nets rather than the total number of pins for one component. Hence in order to meet the pin constraint, the number of external connections (including primary I/Os and the cut nets connected to previously partitioned components) should be added together with the min-cut size to obtain the total number of pins. The disadvantage is that during the max-flow computation, there is little control on how many external connections will be included in either side of the min-cut. The random distri-

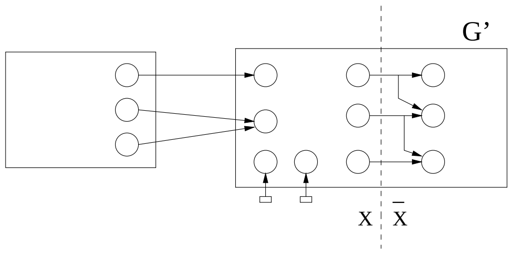


Fig. 6. The total number of pins for component X consists of three parts: PI/PO, cut nets between X and $\overline{X}$, and cut nets connected to other components

bution of these external I/Os sometimes makes it difficult to find a large feasible component while the min-cut size is still relatively small. Therefore it is important to model the I/O nodes properly.

Before we partition a flow network G', it has some external connections which come from two sources: (1) A primary I/O node, (2) A cut net with some nodes in G' and some nodes in other previously partitioned components. We refer to these two types of nodes as *external* I/O nodes (or I/O nodes) in G'. In the discussion below, we use the following notations: $pin(X)$ denotes the total number of pins for a subset X, $net(X, \overline{X})$ denotes the number of crossing nets from X to $\overline{X}$ and $io(X)$ denotes the number of external I/Os in subset X. It is easy to show that $pin(X) = net(X, \overline{X}) + io(X)$.

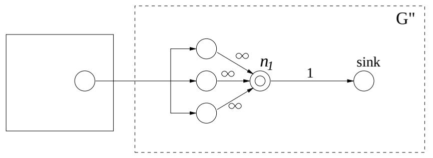
As shown in Figure 6, component X has eight I/O pins which are used by the three cut nets between X and $\overline{X}$ and the five I/Os. The five I/Os consist of two primary I/Os and three cut nets from a previously partitioned component. We model all the I/O nodes in G' to construct G'' as follows:

1. All nodes and edges in G' are in G''.
2. For a cut net with more than one node in G', add a virtual node n_1. Add an edge from each unassigned node in the net to n_1 with capacity ∞, then add a bridging edge from n_1 to the sink with capacity 1 (Figure 7(a)).
3. If a node v is a primary I/O node, then add a bridging edge from v to the sink, with capacity 1. For a cut net with only one node v in G', add a bridging edge from v to the sink with capacity 1 (Figure 7(b)).

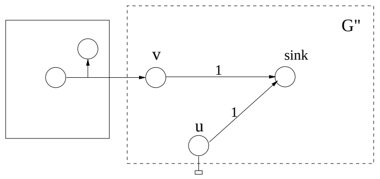
Figure 8 shows an example of a flow network before and after I/O modeling. With the I/O modeling, we can derive good properties as stated in Lemmas 4 and 5.

Lemma 4: *For a min-cut $(X, \overline{X})$ in G'', the cut size C is equal to the total number of pins for X.*

Proof: For any edge which is cut by the min-cut, it is either a cut net or a bridging edge to the sink for I/O modeling. Since the capacity on such an edge is 1, it is counted exactly once in the min-cut size. We have $C \leq net(X, \overline{X}) + io(X)$. On the other hand, if a net is cut, then it is counted as one in the min-cut size. If an I/O node is in X, then any bridging edge from this node to the sink must be cut and counted once in the min-cut



(a) If a cut net has more than one node in G', then it occupies one pin. Node n1 is added with a bridging edge to the sink with capacity 1. Add an edge from each of the unpartitioned node in this net to n1 with capacity ∞ .



(b) If a node is a primary I/O or if a cut net has only one node in G", then add a bridging edge from this node to the sink with capacity 1.

Fig. 7. Modeling of the I/O nodes

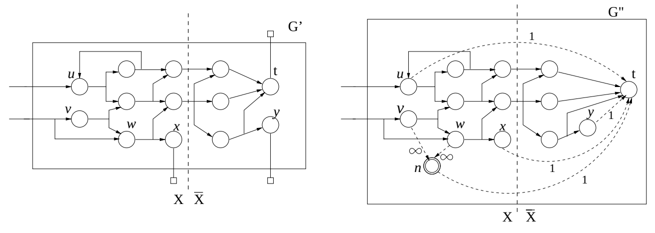


Fig. 8. Example of a network before and after I/O modeling

size. Therefore $net(X, \overline{X}) + io(X) \leq C$. From the above analysis, we have $C = net(X, \overline{X}) + io(X)$. Since $pin(X) = net(X, \overline{X}) + io(X)$, this leads to $C = pin(X)$. Therefore, the min-cut size C is equal to the total number of pins for X. □

As demonstrated in Figure 8, before the I/O modeling, the min-cut size for $(X, \overline{X})$ is two. Yet the total number of pins for X should be five because two additional I/O pins are used for the two cut nets interconnected to other partitioned components, and one I/O pin is occupied by the primary output node x. After the I/O modeling, the min-cut size for $(X, \overline{X})$ is five which is equal to the total number of pins for X.

Lemma 5 compares two min-cuts with the same cut size that cut the network into different area, and validates the benefit of using procedure **DMC** to find a desirable min-cut which has a large area as close to the area limit as possible.

Lemma 5: *If $(X_1, \overline{X_1})$ and $(X_2, \overline{X_2})$ are two min-cuts with the same cut size in G'' such that $X_1 \subseteq X_2$, then $pin(X_1) = pin(X_2)$, $net(X_1, \overline{X_1}) \geq net(X_2, \overline{X_2})$ and $pin(\overline{X_1}) \geq pin(\overline{X_2})$.*

Proof: If $(X_1, \overline{X_1})$ and $(X_2, \overline{X_2})$ are two min-cuts with the same cut size, then by Lemma 4, $pin(X_1) = pin(X_2)$. As $X_1 \subseteq X_2$, so $io(X_1) \leq io(X_2)$ and $io(\overline{X_1}) \geq io(\overline{X_2})$. It is true that $pin(X_1) = net(X_1, \overline{X_1}) + io(X_1)$ and $pin(X_2) = net(X_2, \overline{X_2}) + io(X_2)$, thus $net(X_1, \overline{X_1}) \geq net(X_2, \overline{X_2})$. Further, we have $net(X_1, \overline{X_1}) + io(\overline{X_1}) \geq net(X_2, \overline{X_2}) +$

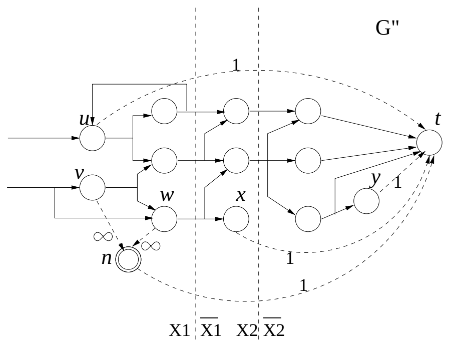


Fig. 9. Comparison of two min-cuts

$io(\overline{X_2})$, this leads to $pin(\overline{X_1}) \geq pin(\overline{X_2})$. □

By Lemma 5, for two min-cuts, the one with a larger area is better because it not only has a larger area, but also results in fewer number of cut nets and fewer occupied I/O pins in the rest of the network to be partitioned later. Figure 9 illustrates an example. With $X_1 \subseteq X_2$, both X_1 and X_2 have five pins, but there are three cut nets between X_1 and $\overline{X_1}$, and only two cut nets between X_2 and $\overline{X_2}$. Thus X_2 is a larger subset with fewer cut nets. $\overline{X_2}$ has four I/O pins (*i.e.* used by the two primary I/O nodes t, y and the two cut nets connected to X_2), and $\overline{X_1}$ has six I/O pins (*i.e.* used by the three primary I/O nodes t, x, y and the three cut nets connected to X_1). Therefore, $\overline{X_2}$ has a smaller area as well as a fewer number of pins than $\overline{X_1}$.

We designed algorithm 2 for multi-way partitioning with I/O modeling. Similar to algorithm 1, it iteratively calls procedure **FC** to find one large feasible component at a time. But procedure **FC** is modified as follows: after selecting the source and sink, G'' is constructed from G' by the I/O modeling process. Then max-flow computation is repeatedly applied on G'' to find a min-cut until either the area or the pin constraint is met. By Lemma 4, the min-cut size obtained by procedure **DMC** in each iteration equals to the total number of pins for X with the source $s \in X$. Each iteration of the max-flow min-cut process tries to minimize the total number of pins for X. By Lemma 5, procedure **DMC** selects a better min-cut which has fewer cut nets and occupies fewer I/Os in the rest of the network. The time complexity of Algorithm 2 is $O(|V||E|)$, since the I/O modeling process does not increase the complexity of the number of nodes or edges in the flow network.

D. Algorithm FBB-MW: the Merging of Algorithms 1 and 2

By I/O modeling, Algorithm 2 tries to minimize the total number of pins for each component. However, one shortcoming is that by adding extra bridging edges, more flows can be pushed in the network, which tends to increase the number of cut nets. To solve this problem, we designed the third algorithm FBB-MW, which is a combination of algorithms 1 and 2, for flow-based multi-way partitioning.

FBB-MW iteratively finds one feasible component at a time in the network. Two stages are involved to find a feasible component. In the first stage, algorithm 1 is applied which repeatedly calculates the max-flow to obtain min-cut. The min-cut size measures the total number of cut nets. Let X be the best feasible subset which is found at the end of algorithm 1. In the second stage, algorithm 2 is used to further increase the area of X. All the nodes in X are condensed to be the source node and G'' is constructed by I/O modeling in the reduced network $G' - X$. Repeated max-flow computation is then applied to find min-cut whose capacity is equal to the total number of pins for the component.

As algorithm 1 minimizes the number of cut nets but has no control on the distribution of I/O nodes, it may only find a small feasible component even if the net cut size is still small relative to the pin constraint. Algorithm 2 has a better control on the number of I/O nodes in component X. The I/O modeling guarantees that the total number of pins is minimized while satisfying the constraints. FBB-MW utilizes the benefit of both algorithms 1 and 2. Experiments shows that FBB-MW produces better results than algorithms 1 and 2, FBB-MW has time complexity $O(k|V||E|)$, which is the same as algorithms 1 and 2.

After the multi-way partitioning by algorithm FBB-MW, postprocessing is performed to further improve the partitioning result. We do a pairwise merge to remove small components in order to reduce the number of components and the number of cut nets.

IV. Experimental Results

We implemented FBB-MW in C language and experimented with two sets of netlists with a wide range of sizes from the MCNC Partitioning93 Benchmark. The first set of test cases contains flat netlists (Table 1) which are combinational and sequential circuits with the number of nodes ranging from 1778 to 25589. The circuits whose name starts with 'c' are combinational circuits and whose name starts with 's' are sequential circuits. The second set of test cases contains CLB-level netlists (Table 4) which are derived by mapping the flat netlists in Table 1 into CLBs (Configurable Logic Block) of XILINX XC2000 and XC3000 device families. The number of CLBs in the netlists ranges from 283 to 3956.

For the first set of test cases, we compared FBB-MW with the MW-part of TAPIR package[15], which is an efficiently and well implemented FM based algorithm for multi-way partitioning under predefined area and pin constraints. The MW-part of TAPIR works as follows. It maintains a set of infeasible components, initially with one single component containing the entire design. In each round, the largest infeasible component is selected from this set and bipartitioned using the FM heuristic. 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 area and pin constraints, they are merged instead of bipartitioned. The process continues until all components are feasible.

For circuits of different sizes, we tried different area and pin limits for the experiments. As shown in Tables 2 and 3, FBB-MW consistently yields better results than TAPIR in terms of the number of components, the total number of pins and cut nets. Since other multi-FPGA partitioning papers did not report partitioning results on these flat netlists in the MCNC Partitioning93 benchmark, we could not conduct the comparison with these algorithms here. However, FBB-MW is compared with these methods on the CLB-level netlists (shown in Tables 4,5,6).

TABLE I
Circuits in MCNC Partitioning93 Benchmark

Circuit Name	# of Nodes	# of Nets	# of I/O
c5315	1778	1655	301
c7552	2247	2140	313
c6288	2856	2824	64
s5378	3225	3176	88
s9234	6098	6076	45
s13207	9445	9324	156
s15850	11071	10984	105
s35932	19882	19560	359
s38417	25589	25483	138
s38584	22451	20719	294

TABLE II
Comparison with TAPIR in terms of the number of components (#comp.)

Circuit	Area Limit	Pin Limit	TAPIR #comp.	FBB-MW #comp.	FBB-MW Impr.%
c5315	1500	100	8	7	12.5
c7552	1500	100	5	5	0
c6288	1500	100	4	2	50
s5378	1500	100	8	6	25
s9234	1500	100	7	6	14
s15850	1500	100	13	10	23
s5378	3000	150	5	2	60
s15850	3000	150	7	5	28
s15850	3000	200	6	4	33
s15850	5000	200	4	3	25
s13207	5000	200	4	2	50
s35932	5000	200	8	7	12.5
s38417	5000	200	9	6	33
s35932	10000	250	5	3	40
s38417	10000	250	4	3	25
s38584	10000	250	5	3	40

As shown in Table 2, FBB-MW results in fewer number of components than MW-part of TAPIR under the same area and pin constraint. For some of the circuits such as C6288, FBB-MW only results in 2 components while the MW-part of TAPIR gets more. Table 3 shows that FBB-MW also results in fewer number of total pins and cut nets.

TABLE IV
CLB-level benchmark circuit characteristics

Circuit	Map to XC2000 families #CLBs	Map to XC2000 families #IOBs	Map to XC3000 families #CLBs	Map to XC3000 families #IOBs
c3540xc	373	72	283	72
c5315xc	535	301	377	301
c7552xc	611	313	489	313
c6288xc	833	64	833	64
s5378xc	500	86	381	86
s9234xc	565	43	454	43
s13207xc	1038	154	915	154
s15850xc	1013	102	842	102
s38417xc	2763	136	2221	156
s38584xc	3956	292	2904	292

TABLE V
Comparison of FBB-MW with three other algorithms. The FPGA area and pin constraints are 64 and 58 respectively for the XILINX XC2064 device. The area and pin constraints are 320 and 144 respectively for the XILINX XC3090 device.

Circuit	Partitioning into XC2064 devices			
	k-way.x[4]	SC[7]	WCDP[11]	FBB-MW
c3540xc	6	6	7	6
c5315xc	11	12	12	10
c7552xc	11	11	11	10
c6288xc	14	14	14	14
Total	42	43	44	40
Circuit	Partitioning into XC3090 devices			
	k-way.x[4]	SC[7]	WCDP[11]	FBB-MW
s15850xc	4	3	3	3
s13207xc	7	6	6	5
s38417xc	9	10	8	8
s38584xc	14	14	12	11
Total	34	33	29	27

For circuit s38417 under area limit 5000 and pin limit 200, FBB-MW results in more number of pins than TAPIR. This is because FBB-MW partitioned the circuit into 6 components and the nodes are more densely packed, while the FM-based method partitioned it into 9 components. However, FBB-MW still gets fewer number of cut nets in this case. For most of the other experiments, FBB-MW not only yields fewer number of components, but also uses fewer number of pins and cut nets.

Our efficient implementation of FBB-MW enables it to partition large benchmark circuits with reasonable running time. Table 2 also shows the average running time (CPU time) of FBB-MW on Intel Pentium-Pro of 200MHz, 32M

TABLE III

Comparison with TAPIR in terms of the total number of pins and cut nets

Circuit	Area Limit	Pin Limit	TAPIR #pins	TAPIR #nets	FBB-MW #pins	FBB-MW #nets	FBB-MW Imprv.% #pins	FBB-MW Imprv.% #nets	FBB-MW T_{run}(sec.)
c5315	1500	100	610	138	584	126	4.3	8.7	1.5
c7552	1500	100	432	55	430	54	0.0	0.02	4.2
c6288	1500	100	335	122	162	49	51.6	59.8	6.3
s5378	1500	100	663	254	567	221	14.5	12.9	7.9
s9234	1500	100	493	197	441	162	10.5	17.6	15.1
s15850	1500	100	869	321	857	287	1.4	10.6	53.8
s5378	3000	150	592	229	132	22	77.7	90.4	3.2
s15850	3000	150	669	250	604	238	9.7	4.8	36.1
s15850	3000	200	665	254	582	228	12.5	10.2	53.1
s15850	5000	200	562	214	417	157	25.8	26.6	35.7
s13207	5000	200	562	154	306	75	45.6	51.3	27.4
s35932	5000	200	1101	306	978	254	11.2	16.9	437.0
s38417	5000	200	760	280	847	244	-11.4	12.8	184.6
s35932	10000	250	957	270	404	89	57.8	67.0	40.5
s38417	10000	250	652	251	558	197	14.4	21.5	90.7
s38584	10000	250	960	311	570	134	40.6	56.9	70.5

memory under Linux environment. The FBB-MW algorithm is run ten times with different pairs of initial source and sink, and the best result is selected as the final partitioning result. For circuits such as C5315, C6288 and C7552, it averages around 5 seconds (CPU time) to find a multi-way partition with area limit 1500 and pin limit 100. For large circuit s38417 which has 25589 nodes, the average running time for multi-way partitioning under area limit 10000 and pin limit 250 is around 90 seconds (which is around 1.5 minutes) CPU time. It takes about 184 seconds (around 3.07 minutes) CPU time to partition s38417 under area limit 5000 and pin limit 200. The observation is that for the same circuit, it usually takes a longer running time when the area and pin limit becomes smaller. This is because with tighter constraints, fewer nodes can be packed in one component and therefore more components will be resulted and a longer running time is needed.

For the second set of test cases, we experimented on the CLB-level netlists from the MCNC Partitioning93 benchmark. Table 4 shows the number of CLBs and IOBs of these netlists when they are mapped to the XILINX XC2000 and XC3000 families. The number of CLBs and IOBs in a FPGA device forms the area and pin contraints for multi-FPGA partitioning.

We compared our results with previous multi-FPGA partitioning algorithms, including *k-way.x* [4], SC [7], WCDP [11], *r+p.0* [5] and PROP [6]. *k-way.x* is a recursive FM-based method for multi-FPGA partitioning. SC is a set covering algorithm for multi-FPGA partitioning in huge logic emulation systems. The WCDP algorithm incorporates ordering, clustering and dynamic programming for good partitioning result. *r+p.0* is an improvement of the *k-way.x* algorithm by using logic replication to reduce the number of interconnections. PROP is a heuristic which combines logic replication and logic re-synthesis in the recursive FM-based partitioning paradigm. The goal is to partition the netlist into a minimum number of homogeneous FPGA devices while satisfying the CLB and IOB constraints of the device. The experimental results in Tables 5 and 6 show that FBB-MW achieves better results than these previous approaches.

In Table 5, we compare FBB-MW with *k-way.x*, SC and WCDP approaches in terms of the number of FPGA devices used in the multi-way partition. Table 5 shows the partitioning of the combinational netlists into XILINX XC2064 devices whose area and pin constraints are 64 and 58 respectively, and the partitioning of the sequential netlists into XILINX XC3090 devices whose area and pin constraints are 320 and 144 respectively.

In Table 6, we compare FBB-MW with *r+p.0* and PROP, which are partitioners that employ logic replication and re-synthesis. The CLB-level netlists are partitioned into XILINX XC3020 and XC3042 devices. All results are obtained under a limit of 90% maximum logic utilization for each device. From the experiments, FBB-MW yields better results than *r+p.0* in terms of the number of FPGA devices used, even though *r+p.0* employs logic replication in the partitioning process to reduce the number of interconnections. Columns 3 and 4 show the results from PROP, where the *(p,o,p)* version uses logic re-synthesis to reduce the amount of logic during the FM-based partitioning process, and the *(p,r,o,p)* version uses both logic re-synthesis and logic replication during the partitioning. Though FBB-MW does not apply any logic optimization

TABLE VI

Comparison with partitioners that employ logic replication and logic re-synthesis. All results are obtained under 90% maximum logic utilization limit. The area and pin constraints are 64 and 64 respectively for XC3020 devices. The area and pin constraints are 144 and 96 for XC3042 devices. *r+p.0* allows logic replication, *(p,o,p)* allows logic re-synthesis and *(p,r,o,p)* allows both re-synthesis and replication. FBB-MW does not allow replication and re-synthesis.

	Partitioning into XC3020 devices			
		PROP[6]		
Circuit	*r+p.0* [5]	*(p,o,p)*	*(p,r,o,p)*	FBB-MW
c3540xc	6	6	6	6
c5315xc	8	8	8	8
c6288xc	16	12	12	15*
c7552xc	10	9	9	9*
s5378xc	10	11	9	9
s9234xc	10	9	9	8*
s13207xc	23	21	19	18
s15850xc	19	17	16	15*
s38417xc	48	44	44	41
s38584xc	60	60	56	54
Total	210	198	188	183

	Partitioning into XC3042 devices			
		PROP[6]		
Circuit	*r+p.0* [5]	*(p,o,p)*	*(p,r,o,p)*	FBB-MW
c3540xc	3	2	2	3*
c5315xc	5	4	4	4*
c6288xc	7	6	5	7*
c7552xc	4	5	4	4*
s5378xc	4	4	4	4
s9234xc	4	4	4	4*
s13207xc	10	9	8	9
s15850xc	9	8	7	8
s38417xc	20	20	19	18*
s38584xc	27	25	25	23*
Total	93	87	82	84

$*$ These partitioning results can not be improved because they are equal to the lower bound given by $\lceil \frac{\#\ total\ CLBs\ in\ netlist}{0.9\times(\#\ of\ CLBs\ per\ device)} \rceil$.

and replication techniques, it still produces comparable result with PROP. For partitioning netlist c6288xc into XC3042 devices, FBB-MW results in 15 devices, which already reaches the lower bound (*i.e.* $\lceil \frac{833}{0.9\times 64} \rceil$). PROP results in 12 devices, because the logic optimization greatly reduces the number of CLBs in the netlist (from 833 to around 669). It is reported in [6] that by logic re-synthesis, the number of CLBs can be reduced on average by 15%.

V. Conclusion

Network flow is an excellent approach to finding min-cuts because of the celebrated max-flow min-cut theorem. FBB [1,2] successfully applied network flow to two-way balanced partitioning and demonstrated that network flow was a viable approach to circuit partitioning.

In this paper, we first present improvements to FBB by finding the most desirable min-cut in order to make better utilization of the min-cuts in the network, and by improving the modeling of two-terminal nets. Next we present algorithm FBB-MW, which is an extension of FBB, for multi-way partitioning with area and pin constraints. From the experiments, FBB-MW outperforms previous approaches.

FBB-MW is efficiently implemented and the average running time on a large circuits with 25K nodes is within a few minutes. To further improve FBB-MW, our future research direction includes exploring strategies such as integrating FBB-MW with the replication algorithm in [17], and the merging and re-partitioning of components.

References

[1] Honghua Yang and D. F. Wong, *Efficient Network Flow Based Min-Cut Balanced Partitioning*, Proc. ICCAD 1994, pp50-55.

[2] Honghua Yang and D. F. Wong, *Efficient Network Flow Based Min-Cut Balanced Partitioning*, IEEE Transactions on Computer-Aided Design, Vol. 15, No. 12, 1996, pp1533-1540.

[3] N. S. Woo and J. Kim, *An Efficient Method of Partitioning Circuits for Multiple-FPGA Implementation*, 30th ACM/IEEE Design Automation Conference, pp202-207, Texas, June 1993.

[4] R. Kuznar, F. Brglez and K. Kozminski, *Cost Minimization of Partitions into Multiple Devices*, 30th ACM/IEEE Design Automation Conference, pp315-320, 1993.

[5] R. Kuznar, F. Brglez and B. Zajc, *A Unified Cost Model for Min-Cut Partitioning with Replication Applied to Optimization of Large Heterogeneous FPGA Partitions*, Euro-DAC'94, September 1994.

[6] Roman Kuznar and Franc Brglez, *PROP: A Recursive Paradigm for Area-Efficient and Performance Oriented Partitioning of Large FPGA Netlists*, International Conference on Computer Aided Design, pp644-649, 1995.

[7] N.C. Chou, L.T. Liu, C.K. Cheng, W.J. Dai and R. Lindelof, *Circuit Partitioning for Huge Logic Emulation Systems*, 31th ACM/IEEE Design Automation Conference, pp244-249, CA, June 1994.

[8] Roman Kuznar, Franc Brglez and Baldomir Zajc, *Multi-way Netlist Partitioning into Heterogeneous FPGAs and Minimization of Total Device Cost and Interconnect*, 31th ACM/IEEE Design Automation Conference, pp238-243, June 1994.

[9] Jianmin Li, John Lillis and Chung-Kuan Cheng, *Linear Decomposition Algorithm for VLSI Design Applications*, ICCAD'95, pp223-228.

[10] Pak K. Chan, Martin D.F Schlag and Jason Y. Zien, *Spectral-Based Multi-Way FPGA Partitioning*, FPGA'95, pp133-139, Monterey, CA.

[11] Dennis J.H Huang and Andrew B. Kahng, *Multi-Way System Partitioning into a Single Type or Multiple Types of FPGAs*, FPGA'95, pp140-145, Monterey, CA.

[12] Minshine Shih, Ernest S. Kuh and Ren-Song Tsay, *Performance-Driven System Partitioning on Multi-Chip Modules*, 29th ACM/IEEE Design Automation Conference, 1992, pp53-56.

[13] Kalapi Roy-Neogi and Carl Sechen, *Multiple FPGA Partitioning with Performance Optimization*, Internation Symposium on FPGA, pp146-152, 1995.

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

[15] L. James Hwang and Abbas El Gamal, *Min-Cut Replication in Partitioned Networks*, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol 14, No. 1, pp96-106, Jan. 1995.

[16] Laura A. Sanchis, *Multiple-Way Network Partitioning*, IEEE Trans. on Computers, VOL. 38, No.1, Jan. 1989.

[17] Honghua Yang and D.F Wong, *New algorithms for Min-Cut Replication in Partitioned Circuits*, ICCAD'95, pp216-222.

Huiqun Liu received the B.S. degree in computer science from Harbin Institute of Technology, China, in 1991, and M.S. degree in computer science from Tsinghua University, Beijing, China, in 1994. Currently she is pursuing a Ph.D. degree in computer science at the University of Texas at Austin.

Her main research interests include computer-aided design of VLSI circuits, FPGA design and computer architecture.

D. F. Wong received the B.Sc. degree in mathematics from the University of Toronto (Canada) and the M.S. degree in mathematics from the University of Illinois at Urbana-Champaign. He obtained the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 1987.

Dr. Wong is currently an Associate Professor of Computer Sciences at the University of Texas at Austin. His main research interest is CAD of VLSI. He has published more than 140 technical papers and has graduated 14 Ph.D. students in this area. He is a coauthor of "Simulated Annealing for VLSI Design" (Kluwer Academic Publishers, 1988).

Dr. Wong received best paper awards at DAC-86 and ICCD-95 for his work on floorplan design and FPGA routing, respectively. He is the technical program chair of the 1988 ACM International Symposium on Physical Design (ISPD-98). He has also served on the technical program committees of a number of other VLSI CAD conferences (e.g. ICCAD, EDTC, ISCAS, FPGA). He is an Editor of IEEE Transactions on Computers.