Multi-Way System Partitioning into a Single Type or Multiple Types of FPGAs*

Dennis J.-H. Huang and Andrew B. Kahng

UCLA Computer Science Department, Los Angeles, CA 90024-1596 USA
jenhsin@cs.ucla.edu, abk@cs.ucla.edu

Abstract

This paper considers the problem of partitioning a circuit into a collection of subcircuits, such that each subcircuit is feasible for some device from an FPGA library, and the total cost of devices is minimized. We propose a three-phase heuristic that uses ordering, clustering, and dynamic programming to achieve good solutions. Experimental comparisons are made with the previous methods of [4][9].

1 Introduction

Multi-way circuit partitioning has been studied extensively in VLSI CAD. Recent research in multiple-FPGA systems has addressed objectives which minimize the net cut between partitions subject to size constraints (#CLBs) and pin constraints (#IOBs or incident signal nets) on each partition. Woo and Kim [12] proposed a modification of the Fiduccia-Mattheyses (FM) [5] method to perform k-way partitioning, with k given, which tries to minimize the total number of pins of all partitions while satisfying given size and pin constraints. Kužnar et al. [9] proposed an algorithm to partition a given CLB-level netlist into multiple device types to minimize total device cost, where each device 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 et al. [10] allow CLBs to be duplicated, i.e., they introduce functional replication to minimize the total cost of devices and inter-device interconnect. Finally, Chou [4] et al. have proposed an algorithm to partition a circuit into instances of a single FPGA type, such that the number of FPGAs is minimized. They use "local ratio-cut" clustering to reduce the instance complexity, then derive a disjoint partition using a set covering approach, following the paradigm of Espresso II. Their algorithm significantly improves over recursive FM on large benchmarks with up to 160K gates and 90K nets.

In this paper, we consider the problem of partitioning a circuit such that each partition is feasible for some FPGA device type from a library of device types, and the total cost of devices is minimized. We propose a new heuristic which incorporates (i) ordering, (ii) clustering, and (iii) dynamic programming to achieve good solutions. Experimental results show that our heuristic can give substantial improvement over previous methods with comparable CPU costs, particularly for large problem instances. A related result shows that our formulation, which seeks a *disjoint* cover of the circuit by FPGAs, is general. Specifically, we extend a result stated in [4] and prove that an *overlapping* partitioning solution does not reduce total device cost or interconnect, i.e., given an overlapping solution we can always construct a non-overlapping solution (i.e., having disjoint partitions) with the same cost and the same or less I/O utilization.

2 Problem Formulation

A digital circuit can be represented as a hypergraph $G = (V_{io} \cup V_{in}, E)$, where V_{io} is the set of primary input and primary output nodes, V_{in} is the set of internal nodes, and E is the set of hyperedges. Each node $v \in V_{in}$ has a weight $w(v)$ associated with it (e.g., corresponding to area). A subcircuit C is the induced subhypergraph over some subset $V' \subseteq V_{in}$. C is called a *feasible FPGA* if the sum of the weights of its nodes is $w(C) \leq s$, and the number of nets that it cuts is $Pin(C) \leq p$, where s and p are respectively the node capacity and the pin capacity of an FPGA device. The FPGA partitioning implies an assignment of each node in V_{in} to at least one of a set of k subcircuits, such that each subcircuit is a feasible FPGA.

The simplest version of this problem is to partition a circuit into instances of a single type of FPGA device.

Single Type FPGA Partitioning: Given a circuit G, find a minimum number of disjoint feasible FPGAs to cover G.

The generalization of this problem [9] is to cover

*This work was partially supported by NSF MIP-9257982 and MIP-23740.

Device	CLBs(s)	IOBs(p)	cost	LB(l)	UB(u)
XC3020x-x	64	64	1.00	0.8	0.9
XC3030x-x	100	80	1.36	0.8	0.9
XC3042x-x	144	96	1.84	0.8	0.9
XC3064x-x	224	110	3.15	0.8	0.9
XC3090x-x	320	144	4.83	0.8	0.9

Table 1: Xilinx XC3000 device library [9].

G with feasible devices from an FPGA library, such that total device cost is minimized. Table 1 depicts an FPGA library corresponding to a specific device family (Xilinx XC3000), as summarized in [9].

Each device type D_j in the FPGA library has pin capacity p_j (IOBs), node capacity s_j (CLBs), and price c_j. Optionally a user may specify l_i and u_i to represent the lower and upper bounds on the utilization of CLBs, e.g., to enhance autoroutability. A subcircuit C_i is called *type-j feasible* if $Pin(C_i) \leq p_j$ and $l_j s_j \leq w(C_i) \leq u_i s_j$. A k-way partition is called *feasible* if each partition C_i, $i = 1, 2, \ldots, k$ is type-j feasible for some device D_j.

Multiple Type FPGA Partitioning: Given a circuit G, find disjoint feasible FPGAs to cover G with minimum total device cost.

3 Node Ordering, Clustering, and Dynamic Programming

In this section, we describe a three-phase heuristic algorithm for FPGA partitioning. First, we embed the given circuit into a linear ordering of nodes. Intuitively, in a good ordering "natural" clusters will occur as contiguous subsets of the ordering. Second, for large circuits we optionally use a clustering scheme to reduce the problem size. Third, we apply dynamic programming over the ordering to get an optimal *restricted partitioning*, i.e., subject to all clusters being contiguous in the ordering. This approach is similar to that of Alpert and Kahng [1]. We also use techniques from [2], which constructs an ordering by iteratively adding the node with highest *attraction* to the existing ordering.

3.1 Node Ordering

The first phase of our algorithm is node ordering. An ordering $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$ of a node set $V = \{v_1, v_2, \ldots, v_n\}$ is defined by a bijection $\pi : \{1 \ldots n\} \rightarrow \{1 \ldots n\}$. Node v_i is the j^{th} node in the ordering if $\pi(j) = i$. [2] presented various orderings, such as max-adjacency ordering, min-perimeter ordering, max absorption [11], and min scaled cost [3], each of which may be achieved by defining some "attraction" from unordered nodes to the set of all previously ordered nodes. Let $Nets(i) = \{e \in E | v_i \in e\}$ be the set of nets incident on v_i, and let S be the set of all previously ordered nodes, and for each unordered node v_i let $Attract(i)$ be the attraction from v_i to S. The ordering may be obtained by the following three steps:

1. **Initialization**: Randomly choose a node v_x, and set $\pi(1) = x, S = v_x$, and $index = 1$. For each $v_i \in V - S$, compute $Attract(i)$.
2. **Best Node**: If $V - S \neq \emptyset$ choose $v_x \in V - S$ with optimal $Attract(x)$, else exit.
3. **Update**: Increment $index$. Set $\pi(index) = x$, and $S = S \cup v_x$. Update $Attract(i)$ for each $v_i \in V - S$ and go to step 2.

The WINDOW algorithm in [2] used the scaled cost metric, with corresponding attraction function defined as follows :

$$Attract_{\mathrm{SC}}(i) = \sum_{e \in Nets(i)} \frac{|S \cap e|}{|e| - 1}$$

The scaled cost attraction leads to reasonable ordering decisions in most cases. However, sometimes a node is added which has high scaled cost attraction but which also has high degree, thus leading to more cut nets (see Figure 1). In addition, it only works for unit weight nodes. To prevent adding a node which has too strong connectivity to remaining unordered nodes, we introduce a combination of the scaled cost and *min perimeter* objectives. Let each net $e \in E$ have weight $w(e) = \sum_{v_i \in e} w(v_i)$ which is the sum of node weights in e. We use the WINDOW algorithm with the following attraction function to obtain our *Scaled Cost + Min Perimeter* (SCMP) ordering.

$$Attract_{\mathrm{SCMP}}(i) = \sum_{\{e \in Nets(i) \mid e \cap S \neq \emptyset\}} \frac{\sum_{u \in S \cap e} w(u) + w(v_i)}{w(e)} - \alpha \cdot \sum_{\{e \in Nets(i) \mid e \cap S = \emptyset\}} \frac{w(e) - w(v_i)}{w(e)}$$

The first term is the scaled cost attraction function modified to account for weighted nodes. The second term is the min perimeter attraction function also modified to account for weighted nodes. α is the relative weighting factor of the min perimeter attraction. In Figure 1, if $\alpha = \frac{1}{2}$, $Attract(u) = 1$ and $Attract(v) = 2 - \frac{1}{2}(\frac{2}{5}) = \frac{3}{4}$ in the SCMP ordering construction. Thus, SCMP would choose u instead of v. Note that our use of the ordering concept will allow an efficient dynamic programming solution (Section 3.3). Note also that our particular SCMP ordering is designed with respect to the stated problem formulation. For specific multi-FPGA system designs, other objectives may prevail, e.g., reflecting inter-FPGA routing architecture. In such cases, different attraction functions may be preferable, along with use of the *window size* and *tail size* parameters in [2].

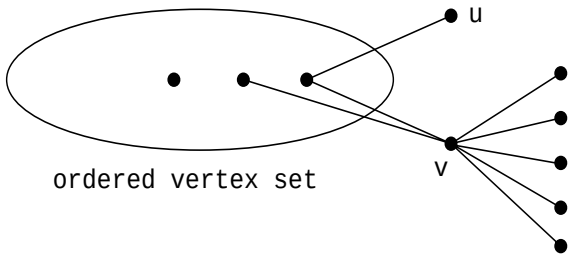

Figure 1: The scaled cost attraction function will choose node v to be added next into the ordered set S, thus leading to more cut nets.

3.2 Clustering

To partition large circuits, the use of clustering to reduce problem complexity is a widely adopted technique. Chou et al. [4] used *local ratio cut* clustering for such a pre-processing step. They iteratively extract a subcircuit C with desired cluster size from the original circuit G, and apply ratio cut partitioning to all nodes in C and all nodes in $G \setminus C$ which have connections to nodes in C. Their clustering technique yields very good results for FPGA partitioning.

We propose a clustering scheme based on the constructed linear ordering, again based on the intuition that a good ordering should preserve dense circuit structure in its contiguous subsets. Given a cluster size limit U, we start from the first node in the ordering, compute the scaled cost of all contiguous clusters starting from that node and having size $\leq U$, and greedily choose the cluster with optimal scaled cost. We then start from the next node, and repeat the above steps until the whole ordering is traversed. Figure 2 shows the algorithm template.

Clustering Algorithm
Input : Linear Ordering $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$ U $\equiv$ Upper cluster size bound
Output : Clustering solution $Clusters$
Vars : $C_{[i,j]} \equiv$ cluster $\{v_{\pi_i}, v_{\pi_{i+1}}, \ldots, v_{\pi j}\}$
$Clusters = \emptyset$ $start = 1$ **while** $(start < n)$ **do** Find cluster $C_{[start, start+j]}$ with optimal scaled cost for all $1 \leq j \leq U$ Let $best_j \equiv$ variable containing best index j $Clusters = Clusters \cup C_{[start, start+best_j]}$ $start = start + best_j + 1$ **return** $Clusters$

Figure 2: Clustering algorithm template.

3.3 Dynamic Programming

Given a node ordering, our dynamic programming (DP) phase finds the optimal FPGA covering cost for a contiguous subset of the ordering, based on the optimal covering costs of the smaller (contained) subsets of the ordering. For the Single Type FPGA Partitioning problem, let $nFPGA[i, size]$ be the minimum number of FPGAs needed to cover the node ordering from index i to index $i + size - 1$. Our DP is based on the following recurrence relation :

$$nFPGA[i, size] = 1 \qquad \text{if } C_{[i,i+size-1]} \text{ is feasible FPGA}$$

else,

$$nFPGA[i, size] = \min_{1 \leq x < size} (nFPGA[i, x] + nFPGA[i + x, size - x])$$

Similarly, for the Multiple Type FPGA Partitioning, let $mcost[i, size]$ be the minimum device cost to cover the node ordering from index i to index $i + size - 1$. The DP recurrence relation is as follows:

$$mcost[i, size] = \min \left(\min_{C_{[i,i+size-1]} \text{ is type-}j \text{ feasible}} c_j, \min_{1 \leq x < size} (mcost[i, x] + mcost[i + x, size - x]) \right)$$

The basic algorithm template for DP is shown in Figure 3. The complexity of this algorithm is $O(Un^2)$. Note that Alpert and Kahng's DP-RP algorithm [1] used a different dynamic programming recurrence, i.e., they derived a k-way partitioning from the best $(k-1)$-way partitioning solutions for all contiguous clusters.

3.4 Experimental Results

All algorithms and experiments were implemented in the UNIX/C environment using a Sun SPARCstation 10. All benchmarks were obtained from MCNC under the directory /pub/benchmark/Partitioning93. We also obtained three large industrial benchmarks from the authors of [4].

We omitted the clustering phase for the small benchmarks. For all experiments, we set $\alpha = 0.3$ in the WINDOW ordering phase. For the four relatively small benchmarks in [9], the FPGA constraint is based on the Xilinx XC2064 (i.e., 64 CLBs and 58 IOBs). Table 2 shows that our algorithm (denoted by WCDP) achieves comparable results with [9] and [4]. WCDP also performs very well in larger test cases. For the four larger benchmarks(s-series), the FPGA constraints are set to 320 CLBs and 144 IOBs. Table 3 shows that WCDP outperforms recursive FM and SC [4].

We also tested WCDP on three huge industrial benchmarks with up to $160K$ gates and $90K$ nets. WCDP achieves an average of 18% improvement over SC with reasonable (but longer) running time, as shown in Table 4.

For Multiple Type FPGA Partitioning, we tested all nine benchmarks studied in [9], using the same library. Our WCDP heuristic outperforms [9] by an average of 5.08%, as shown in Table 5. In [10], Kužnar et al. allow node replication during the partitioning; WCDP still obtains better results without using replication.

DP Algorithm for Single Type FPGA Partitioning

Input : Linear Ordering $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$
Output : the number of FPGAs required by the given circuit and partitioning solution
Vars : $C_{[i,j]} \equiv$ cluster $\{v_{\pi_i}, v_{\pi_{i+1}}, \ldots, v_{\pi j}\}$
nFPGA[i,j] $\equiv$ minimum number of FPGAs required by cluster $C_{[i,i+j-1]}$

```
for i = 1 to n do
    nFPGA[i,1] = 1;
for size = 2 to n do
    for i = 1 to n do
        if C[i,i+size-1] is feasible then
            nFPGA[i,size] = 1;
        else
            min_value = ∞
            for j = 1 to U do
                min_value = Min( min_value,
                                 nFPGA[i,j] + nFPGA[i+j,size-j])
            nFPGA[i,size] = min_value;
return  min   nFPGA[i, n]
      1≤i≤n
```

DP Algorithm for Multiple Type FPGA Partitioning

Input : Linear Ordering $\{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}\}$
U $\equiv$ max FPGA size in the library
Output : the minimum device cost required by the given circuit and partitioning solution
Vars : $C_{[i,j]} \equiv$ cluster $\{v_{\pi_i}, v_{\pi_{i+1}}, \ldots, v_{\pi j}\}$
mcost[i,j] $\equiv$ minimum device cost required by cluster $C_{[i,i+j-1]}$

```
for i = 1 to n do
    mcost[i,1] = minimum FPGA device cost;
for size = 2 to n do
    for i = 1 to n do
        if C[i,i+size-1] can be fitted into a device with minimum
                cost min_dev_cost then
            min_cost = min_dev_cost
            for j = 1 to U do
                min_cost = Min(min_cost, cost[i,j] + cost[i+j,size-j])
            cost[i,size] = min_cost;
return  min   cost[i, n]
      1≤i≤n
```

Figure 3: Dynamic programming for single- and multiple-type FPGA partitioning.

4 Node Overlapping

Our partitioning formulation is inherently based on an undirected circuit representation. Moreover, there is no intrinsic reason to seek a disjoint cover of the circuit by feasible FPGAs. One might thus consider the use of *node overlapping*, i.e., assigning a node to more than one partition, to reduce the number of pins in some partitions. (Figure 4 shows a non-overlapping solution; if v_1 and v_2 are overlapped by the two partitions, the number of pins in C_2 is reduced by 1.) However, we will show that node overlapping cannot reduce the cost of an FPGA partitioning. (This is a consequence of both the undirected formulation and the satisficing pin constraints.) Note that node overlapping is fundamentally different from the concept of node replication discussed in [8, 7]: node replication uses directional information and only duplicates *incoming* edges incident to the replicated nodes.

For the Single Type FPGA partitioning problem, Chou et al. [4] proposed the "FPGA complementary theorem", which states that k disjoint feasible FPGAs can be obtained from k overlapping feasible FPGAs. They gave the proof for the 2-way result, but did not give a formal proof for the k-way result. We now provide an analogous complementarity result for the k-way Multiple Type FPGA partitioning problem.

ckt	(CLBs, IOBs, nets, pins)	KBK93	SC	WCDP
c3540	(373, 72, 569, 1933)	6	6	7
c5315	(535, 301, 936, 3004)	11	12	12
c7552	(611, 313, 1057, 3318)	11	11	11
c6288	(833, 64, 1472, 3438)	14	14	14

Table 2: Comparison of WCDP and two other algorithms. KBK93 is the modified FM algorithm in [9], and SC is the Set Covering algorithm in [4]. The FPGA size constraint and pin constraint are 64 and 58 respectively, reflecting the XILINX XC2064 part.

ckt	(CLBs, IOBs, nets, pins)	RFM	SC	WCDP
s15850	(842, 102, 1265, 4977)	4	3	3
s13207	(915, 154, 1377, 5309)	7	6	6
s38417	(2221, 156, 3216, 13257)	12	10	8
s38584	(2904, 292, 3884, 17483)	17	14	12

Table 3: Comparison of the WCDP, recursive FM (RFM), and Set Covering [4] algorithms. The FPGA size constraint and pin constraint are 320 and 144 respectively.

Theorem 1 : k-Way Single Type FPGA Complementary Theorem (Chou et al. [4]). *Let $C_1, C_2, \ldots, C_k$ be k feasible FPGA candidates. Then there exists a permutation function $\pi(i), i \in 1, 2, \ldots, k$, such that*

1. $C_{\pi(1)}$,
2. $C_{\pi(2)} \setminus C_{\pi(1)}$,
3. $C_{\pi(3)} \setminus (C_{\pi(1)} \cup C_{\pi(2)})$,

$\vdots$

k. $C_{\pi(k)} \setminus (C_{\pi(1)} \cup C_{\pi(2)} \cup \ldots \cup C_{\pi(k-1)})$

are disjoint feasible FPGAs.

ckt	gates	pins	I/Os	nodes	nets
38K	38039	30043	120	7259	10102
49K	48992	62700	1129	19215	19734
160K	159054	245311	1885	89288	90029

	LRSC	WCDP	
ckt	#FPGAs (min.)	#FPGAs (min.)	% improv.
38K	37 (9)	33 (14)	10.81%
49K	87 (65)	77 (110)	11.49%
160K	186 (383)	128 (589)	31.38%

Table 4: Comparison of WCDP and SC [4] algorithms based on three huge industrial benchmarks. Top: Benchmark parameters. Bottom: partitioning results. The FPGA size constraint is 2,700 CLBs and the pin constraint is 184 IOBs. The numbers in parentheses giving running time in minutes on a Sun SPARC-10.

ckt	KBK93	WCDP	LB	KBZ94
c3540	5.52	4.51(18.30%)	4.51	4.56(18.30%)
c5315	7.03	7.04(-0.14%)	5.92	6.92(1.57%)
c6288	13.66	12.32(9.81%)	12.32	13.66(0%)
c7552	7.36	7.88(-7.06%)	7.36	7.36(0%)
s5378	6.67	6.67(0%)	5.52	6.19(7.20%)
s9234	7.98	6.83(14.42%)	6.83	7.98 (0%)
s13207	17.16	18.52(-7.92%)	13.40	18.12 (-5.59%)
s15850	14.80	13.55(8.45%)	12.35	14.97 (-1.14%)
s38584	51.83	45.60(12.03%)	45.50	51.19 (1.24%)
Impr. vs. KBK93	–	5.08%		2.40%

Table 5: Comparison of Multiple type FPGA Partitioning results. WCDP outperforms KBK93 [9] by 5.08%. LB = lower bound on solution cost obtained by integer programming in [9].

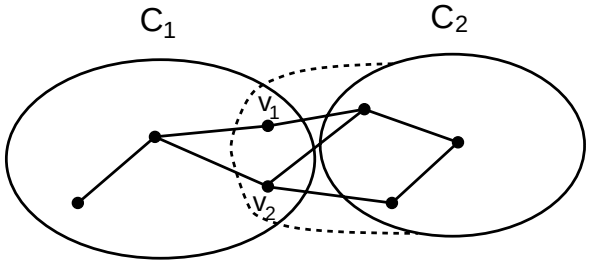

Figure 4: An example of node overlapping. When node v_1 and v_2 are overlapped (dotted line), the number of pins in C_2 is reduced by 1.

This theorem suggests that for Single Type FPGA partitioning, a solution without overlapping can be obtained from an overlapping solution. We now extend the result to Multiple Type FPGA partitioning.

Theorem 2 : 2-Way Multiple Type FPGA Complementary Theorem. *Let C_1 be a type-1 feasible FPGA, and C_2 be a type-2 feasible FPGA. Then at least one of the following is true:*

1. C_1 is type-1 feasible and $C_2 \setminus C_1$ is type-2 feasible.
2. C_2 is type-2 feasible and $C_1 \setminus C_2$ is type-1 feasible.

Proof : Let $C_1 = A \cup O$, $C_2 = B \cup O$, $C_1 \cap C_2 = O$, and R be the rest of the circuit (note that $O \equiv$ "overlap"). We consider the possible multi-pin nets. Let AB denote the number of nets which contain at least one pin in each of A and B, and let $AO(R \cup B)$ denote the number of nets which contain at least one pin in each of A, O, and $(R \cup B)$. Let p_1 and p_2 be the pin constraints for type-1 and type-2 FPGAs respectively. Suppose neither of the above holds, i.e., $A = C_1 \setminus C_2$ is not type-1 feasible and $B = C_2 \setminus C_1$ is not type-2 feasible. Since C_1 is type-1 feasible,

$$AO(R \cup B) + A(R \cup B) + O(R \cup B) \leq p_1 \quad (1)$$

and since A is not type-1 feasible,

$$AO(R \cup B) + AO + A(R \cup B) > p_1 \quad (2)$$

From (1) and (2), we have

$$AO > O(B \cup R) \Rightarrow AO > BO \quad (3)$$

Similarly, C_2 is type-2 feasible and B is not type-2 feasible, implying

$$BO > O(A \cup R) \Rightarrow BO > AO \quad (4)$$

This contradicts (3), and hence one of the two outcomes must hold. □

Theorem 3 : k-Way Multiple Type FPGA Complementary Theorem: *Let $C_1, C_2, \ldots, C_k$ be type-t_1, type-t_2, ..., type-t_k feasible FPGAs with pin constraints $p_{t_1}, p_{t_2}, \ldots, p_{t_k}$, respectively. There exists a permutation function $\pi(i), i \in 1, 2, \ldots, k$, such that :*

1. $C_{\pi(1)}$ is type-$t_{\pi(1)}$ feasible.
2. $C_{\pi(2)} \setminus C_{\pi(1)}$ is type-$t_{\pi(2)}$ feasible.
3. $C_{\pi(3)} \setminus (C_{\pi(1)} \cup C_{\pi(2)})$ is type-$t_{\pi(3)}$ feasible.
$\vdots$
k. $C_{\pi(k)} \setminus (C_{\pi(1)} \cup C_{\pi(2)} \cup \ldots \cup C_{\pi(k-1)})$ is type-$t_{\pi(k)}$ feasible FPGA.

Proof : We prove the result by induction. When $k = 2$, this theorem is the 2-Way Multiple Type FPGA Complementary Theorem. Assume the result holds for $k = j - 1$ partitions. We prove the result for $k = j$ as follows :

Let $C_i = N_i \cup O_i$, where N_i is the non-overlapping part of C_i and O_i is the part of C_i overlaps with other clusters. Let R be the set of I/O pads and let $G = C_1 \cup C_2 \cup \ldots \cup C_k \cup R$ be the whole circuit as shown in Figure 5. We claim that at least one of the N_i $(1 \leq i \leq j)$ is a feasible FPGA. Suppose none of the N_i are feasible. Then, since C_i is feasible, for all i we have (using the same notation as before)

$$N_i(G \setminus C_i) + O_i(G \setminus C_i) + N_i O_i(G \setminus C_i) \leq p_{t_i} \quad (5)$$

and since N_i is not feasible,

$$N_i(G \setminus C_i) + N_i O_i + N_i O_i(G \setminus C_i) > p_{t_i} \quad (6)$$

From (5) and (6), $N_i O_i > O_i(G \setminus C_i)$ for all i, whence

$$\sum_{i=1}^{j} N_i O_i > \sum_{i=1}^{j} O_i(G \setminus C_i) \quad (7)$$

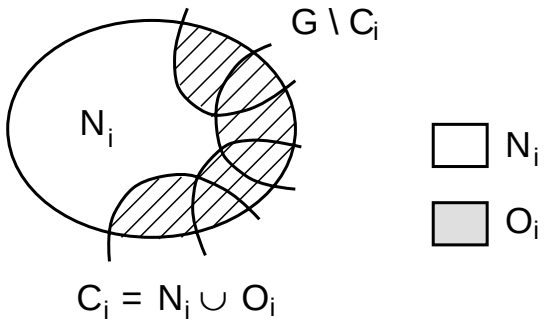

Figure 5: A feasible FPGA partition C_i with non-overlapping part N_i and overlapping part O_i.

All nets in N_iO_i (for all i) are disjoint since they must have pins in N_i. Every net in $\bigcup_i N_iO_i$ must appear at least once in $\bigcup_i O_i(G \setminus C_i)$ since the net in N_iO_i must have pins in N_i and have pins in O_i sharing with other clusters. For example, if the net in N_iO_i shares pins with clusters C_x and C_y, then this net will also be contained by $O_x(G \setminus C_x)$ and $O_y(G \setminus C_y)$. Thus, $\sum_{i=1}^{k} N_iO_i \leq \sum_{i=1}^{k} O_i(G \setminus C_i)$. This contradicts (7). Therefore, there exists at least one N_i which is a feasible FPGA. After removing one feasible N_i, by the induction hypothesis the remaining $j-1$ overlapping clusters can be rearranged to yield a non-overlapping partitioning. □

This theorem suggests that for a given overlapping solution of the multi-type FPGA partitioning problem, we can construct a non-overlapping solution with the same cost and the same (or less) total I/O utilization. The optimal solutions for overlapping and non-overlapping are the same.

5 Conclusion

We have presented the WCDP heuristic – composed of modified WINDOW ordering, clustering, and dynamic programming phases – for the Single Type and Multiple Type FPGA Partitioning problems. Our algorithm outperforms SC by 18% on huge benchmarks, and outperforms KBK93 by 5% for Multiple Type FPGA partitioning. Of supplementary interest is our extension of the FPGA complementary theorems in [4] to multiple-type FPGA partitioning. According to this result, and within our given problem formulation, we can always obtain a disjoint partitioning solution from an overlapping solution.

6 Acknowledgments

We are grateful to Dr. L. T. Liu for his valuable discussion and providing the translator of benchmarks [9] from MCNC.

References

[1] C. J. Alpert and A. B. Kahng, "Multi-Way Partitioning Via Spacefilling Curves and Dynamic Programming", 31st *ACM/IEEE Design Automation Conference*, 1994, pp. 652-657.

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

[3] P. K. Chan, M. D. F. Schlag and J. Y. Zien, "Spectral K-Way Ratio-Cut Partitioning and Clustering", *IEEE Trans. on CAD* 13(9), Sept. 1994, pp. 1088-1096.

[4] N.-C. Chou, L.-T. Liu, C.-K. Cheng, W.-J. Dai, and R. Lindelof, "Circuit Partitioning for Huge Logic Emulation Systems", 31st *ACM/IEEE Design Automation Conference*, 1994, pp. 244-249.

[5] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", 19th *ACM/IEEE Design Automation Conference*, 1982, pp. 175-181.

[6] L. Hagen and A. B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering", *IEEE Trans. on CAD* 11(9), Sept. 1992, pp. 1074-1085.

[7] J. Hwang and A. El Gamal, "Optimal Replication for Min-Cut Partitioning", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1992, pp. 432-435.

[8] C. Kring and A. R. Newton, "A Cell-Replication Appraoch to Mincut-Based Circuit Partitioning", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1991, pp. 2-5.

[9] R. Kužnar, F. Brglez, and K. Kozminski, "Cost Minimization of Partitions into Multiple Devices, 30th *ACM/IEEE Design Automation Conference*, 1993, pp. 315-320.

[10] R. Kužnar, F. Brglez, and B. Zajc, "Multi-way Netlist Partitioning into Heterogeneous FPGAs and Minimization of Total Device Cost and Interconnect", 31st *ACM/IEEE Design Automation Conference*, 1994, pp. 238-243.

[11] W. Sun and C. Sechen, "Efficient and Effective Placements for Very Large Circuits" *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, Nov. 1993, pp. 170-177.

[12] N.-S. Woo and J. Kim, "An Efficient Method of Partitioning Circuits for Multiple- FPGA Implementation", 30th *ACM/IEEE Design Automation Conference*, 1993, pp. 202-207.