システムＬＳＩ設計技術　94－7
（2000. 1. 11）

最小カットを用いて適切な部分回路を抽出するための効率的手法

畔上 謙吾, 高橋 篤司, 梶谷 洋司

東京工業大学 工学部 電気・電子工学科
〒 152-8552 東京都 目黒区 大岡山 2-12-1
Tel : (03)5734-3572, Fax : (03)5734-2902
E-mail : {azegami, atushi, kajitani}@ss.titech.ac.jp

あらまし　ハイパーグラフからミンカットグラフを生成する手法について述べる。ミンカットグラフは，その有向カットがハイパーグラフのミンカットに 1:1 で対応する。従来はこのグラフを高速に得るため、その正確さを犠牲にしてきた。本手法は正確さと高速性を同時に達成している。得られたミンカットグラフの有向カット数はその点数に対し指数オーダだが、実際はその点数が小さいため、全探索により実用的な時間内に適切な有向カットを求められることを実験的に示した。

キーワード : 回路分割，ハイパーグラフ分割，ネットワークフロー，ミンカット，集積回路

An Efficient Algorithm to Extract An Optimal Sub-Circuit by the Minimum Cut

Kengo Azegami, Atsushi Takahashi and Yoji Kajitani

Dept. of Electrical and Electronic Engrg., Tokyo Inst. of Tech.
Ookayama, Meguro, Tokyo, 152-8552 Japan
Tel : +81-3-5734-3572, Fax : +81-3-5734-2902
E-mail : {azegami, atushi, kajitani}@ss.titech.ac.jp

Abstract　We improve the algorithm to obtain the min-cut graph of a hyper-graph and show an application to the sub-network extraction problem. The min-cut graph is a directed acyclic graph whose directed cuts correspond one-to-one to the min-cuts of the hyper-graph. While the known approach trades the exactness of the min-cut graph for some speed improvement, we propose an algorithm which gives an exact one without substantial computation overhead. An exhaustive algorithm is proposed to find an optimal sub-circuit that is cut by a min-cut from the rest. By experiments with the industrial data, the proposing method showed a performance enough for practical use.

Key Words : circuit partition, hyper-graph partition, network flow, min-cut, integrated circuit design

1 Introduction

Partitioning is essential in various stages of hierarchal VLSI design. The problem formulations as well as its solutions have been studied extensively. Studies related to our subject are found in [1], [2], [9], [7], [14], and [15]. See [3] for an extensive survey. Since the problem is so diversitive, we start with some general framework with respect to problems, approaches and application environments.

In partitioning a given huge circuit into sub-circuits, three major indices are focused : 1. the cut size (number of edges interconnecting sub-circuits), 2. the number of sub-circuits, and 3. the balance (maximum difference of sizes) of sub-circuits. The problem formulation, approach and environment are stated in terms of them.

Typical problem formulations are such that "partition a given circuit into a certain number of sub-circuits minimizing the cut size" and that "partition the circuit under the constraint of a limit of cut size minimizing the number of sub-circuits". There are other combinations of three indices but they are too unrealistic to be considered in VLSI design.

The approaches are classified roughly in two ways, one based on the max-flow min-cut theorem and the other based on the greedy vertex exchange strategy (such as KL or FM methods).

While applications of partition in VLSI design are in two environments. One is in the placement based on the slice line structure. Since the circuit is embedded in two divided half zones, it is necessary to give a solution to :

(Slicing Problem) Find a partition into two with the minimum cut-size under the constraint on the number and balance of sub-circuits.

The other is in module design of pre-fabricated circuits such as FPGA or MCM architecture. Since each module is a look-up table or a cell in a library, the problem is described as :

(Module Problem) Find a partition of the circuit into the minimum number of sub-circuits under the constraint of the cut-size and balance of sub-circuits.

For the Slicing problem, the max-flow min-cut based approach has not been believed effective since the algorithm finds a difficulty handling hyper-edges, the balance is not controllable, and the computational cost is large. So, the exchange based approaches have been taking the place by the merits: it has no discrimination against hyper-edges, its computation cost can be any small (the algorithm could stop any time), and the balance can be arbitrarily defined (the initial partition defines the balance). The only problem, but fatal, is that the approximation to the exactness is not known.

For the Module problem, both approaches were considered not adequate. The exchange based one has no idea to minimize the number of sub-circuits. It is also true that the max-flow min-cut based approach complies no idea of balancing nor minimizing the number of sub-circuits.

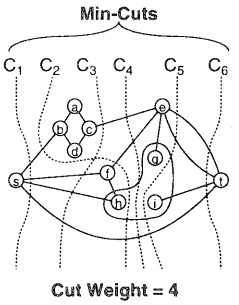

Figure 1: Hyper-Graph H and its s-t min-cuts of cut weight 4. An edge with two end vertices are drawn as a single line, while an edge with more than two end vertices are drawn as a curve.

The concern of this paper is in the Module problem. However, it is so intractable that we introduce an alternative which is described as follows.

(Extraction Problem) Given a circuit with two vertices assigned, extract a sub-circuit that includes one designated vertex inside and the other outside under the constraint of the size of the cut being minimum and that of the circuit within a specified range.

To approach this problem, the only algorithm so far proposed is to fix a maximum flow, find one minimum cut to split the vertices to S and T where the source is in S, and choose one vertex in T and find the flow-reachable vertex set T'. Then, the cut that separates $T \cup T'$ is another minimum cut. Select one vertex there, and continue. Not to miss any minimum cut, the choice shall be from all the vertices. Hence this meek algorithm needs n times of the whole graph traversal, which is not tolerable by its huge computation time.

To display all the minimum cuts, a useful data structure known by the name of min-cut graph [6] exists. It is a directed graph $M(H)$ defined for any hyper-graph H with a source and a sink such that all the directed minimum cuts of $M(H)$ correspond one-to-one to the minimum cuts of H. The existence of the min-cut graph for an ordinary graph is a known fact. For the hyper-graph H, as far as the authors know, it was first suggested by Liu and Wong in [6]. See Fig. 1 for a hyper-graph H. This graph has min-cuts $C_1, \ldots, C_6$ of cut weight 4. See Fig. 2 for its corresponding min-cut graph $M(H)$.

They [6] claim merits of the min-cut graph but mentioned also that the penalty is in the computational cost to obtain it since they assumed the meek algorithm mentioned above. And they proposed to a heuristic

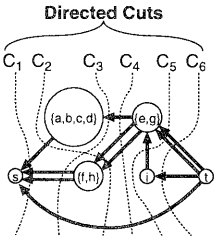

Figure 2: The min-cut graph $M(H)$ for H. Each vertex in $M(H)$ is labeled by a set of vertices in H surrounded by its min-cuts.

as a compromise. The first objective of this paper is to claim that the min-cut graph is easy to obtain by noting if we come to the idea that each strongly connected component of the flow graph with respect to the flow-reachability corresponds to a vertex of the min-cut graph.

Once $M(H)$ is obtained, we would use it to extract a sub-circuit from H. It is done by extracting one sink after another considering maximization of the evaluation. If it is done by exhaustive search, the variety will increase exponentially according to the number of new sinks born each time a vertex is extracted. We dare to apply the strategy faithfully to the industrial data which are consisting of thousands of vertices and the size constraint is around hundreds of vertices. The second objective of this paper is to show by experiments that this apparently brute-force will work in a reasonable computation time. It also revealed the merit of using the exact min-cut graph, unlike the study found in [6].

2 Flow-Graph and Flow-Block

Let $F = (V, E)$ be an ordinary flow-graph where V and E are the sets of vertices and edges, respectively. $(u, v) \in E$ is an edge with direction from u to v. Each edge e has an associated capacity $cap(e)$. Special vertices s and t are the source and sink, respectively.

Assume an s-t flow of F. Of each edge e, $flow(e)$ denotes the amount of the flow on the edge. An edge e is said *saturated* if $flow(e) = cap(e)$ and *zero-flow* if $flow(e) = 0$. A vertex p is *flow-reachable* from vertex v if there exist a flow-augmentable path from v to p. The maximal amount of flow (or its flow distribution on the edges) from s to t is referred to as a *maximum s-t flow*, or simply a *max-flow*. In a max-flow, there is no flow-augmentable path from s to t, that is, t is not flow-reachable from s. A max-flow can be computed by known algorithms, for instance, by the preflow-push method in $O(n^2m)$ [10], [11] where $n = |V|$ and $m = |E|$.

Let V_1 and V_2 be sets of vertices such that $s \in V_1$, $t \in V_2$, $V_1, V_2 \subset V$, $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$. The set of edges connecting a vertex in V_1 and a vertex in V_2 is called an s-t cut, denoted by $[V_1, V_2]$. An edge (u, v) in $[V_1, V_2]$ is called *forward* if $u \in V_1$ and $v \in V_2$, and *backward* if $u \in V_2$ and $v \in V_1$. The capacity of an s-t cut C, denoted as $cap(C)$, is the sum of capacities of all the forward-edges. An s-t cut with the minimum capacity is referred to a *minimum s-t cut*, or simply a *min-cut*.

Assume a max-flow. Let a *flow-block* be a set of vertices separated by the min-cuts. An equivalent definition is that two vertices are in the same flow-block if and only if they are mutually flow-reachable. Thus a flow-block is a maximal set of mutually flow-reachable vertices. That is, a flow-block is equivalent to the strongly connected component in terms of flow-reachability.

3 Constructing The Min-Cut : Our Proposal

We confirm that our main task is to (1) find all the flow-blocks, and then (2) browse them to find an appropriate min-cut. Our approach consists of the following four steps. The details will be given later.

Procedure 1. (Construction of $M(H)$: outline)

Input : Hyper-Graph H

Output : Min-Cut Graph $M(H)$

1. (Graph Transformation) Get the s-t flow-graph F by applying the Yang-Wong transformation to H.
2. (Max-Flow) Compute a max-flow in F.
3. (Finding Flow-Block Candidates) Obtain F' by applying *Flow-Reachability transformation* to all the *virtual edges* with their incident vertices.
4. (Strongly Connected Component) Find the strongly connected component in F'. Get min-cut graph $M(H)$ by contracting each of them into a single vertex.

□

3.1 Graph Transformation

We transform $H = (V_H, E_H)$ to a directed graph $F = (V_F, E_F)$ by applying *Yang-Wong Transformation* to all the hyper-edges. This transformation replaces a hyper-edge e_H with a structure, called the *local structure* of e_H. See Fig. 3 for an example where Yang-Wong Transformation is applied to a hyper-edge with three end vertices. The formal definition is as follows.

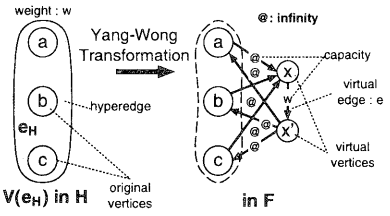

Figure 3: Yang-Wong Transformation : Local Structure

Definition 1 *(Yang-Wong Transformation of e_H with weight w : Step 1 in Procedure 1.)*

1. *Create a pair of vertices x and x'.*
2. *Create a directed edge $e_F = (x, x')$ with $cap(e_F) = w$.*
3. *For each end vertex v of e_H, create a pair of directed edges $e = (v, x)$ and $e' = (x', v)$ with $cap(e) = cap(e') = \infty$.*

□

V_F consists of two kinds of vertices, copies of V_H and the added ones. We refer to the copied ones as *original vertices*, the added ones as *virtual vertices*, and the added edges (x, x') as *virtual edges*.

An observation leads the following fact : In F, every vertex is contained in a directed cycle consisting of two infinite-capacity edges and one edge, which is the virtual edge, of capacity w. Then, any cut $C = [V_s, V_t]$ in F with finite capacity cannot contain a virtual edge as a backward edge. Assume C to be a min-cut. A virtual edge is contained in C if and only if at least one end-vertex belongs to V_s and at least one to V_t. Hence, $cap(C)$ in F is equal to the sum of the capacity of the hyper-edges belonging to the corresponding cut in H. In other words, the finite capacity cuts in F correspond one-to-one to the cuts of H such that they separate the original vertices in the same way. This is the property the Yang-Wong transformation is intending to achieve.

It must be mentioned that we could explain the Yang-Wong transformation by a series of known techniques: See Fig. 4. First, an undirected edge-capacity hyper-graph (A) is transformed to a vertex-capacity graph (B) with respect to the equivalence of cut capacity, where the cut is the one that splits the vertices. It is done by replacing the hyper-edge with a star graph, where the hyper-edge capacity w in H is transferred to the capacity of the center vertex x. Second, (B) is transformed to a vertex-capacity directed graph (C) by replacing each undirected edge by a pair of edges of opposite directions. Finally, C is transformed to an

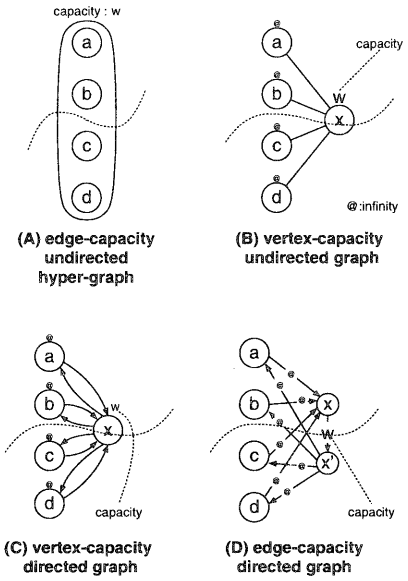

Figure 4: Principle of the Yang-Wong Transformation

edge-capacity directed graph (D) to a local structure as described. The cut here is the one that cuts the edges.

Each step is too well-known in network flows and graph theory. However, as far as the authors know, [4] and [5] were the first to bridge hyper-graph and directed graph in terms of flow. Moreover, including [4], [5] and [6], there have been no references that clarify explicitly the principles on how an edge-capacity hyper-graph can stepwise be transformed to an edge-capacity directed graph, as we did here.

3.2 Flow-Block Candidates

By applying the *flow-reachability transformation* to F, we can obtain a graph F' such that the flow-reachability of the vertices is embossed.

Procedure 2. (Flow-Reachability : $F \rightarrow F'$: Step 3 in Procedure 1.)

Input : Max-Flow computed Flow-Graph $F = (V_F, E_F)$

Output : Directed graph F' whose strongly connected components correspond to the flow-blocks.

1. Copy all the original vertices of F in F'
2. For each virtual edge e_F and its local structure
   (a) if e_F is saturated then
      i. let V_o, V_i and V_n be the sets of vertices in F' where, in F, from which the flow

comes in to e_F, goes out from, and neither, respectively.

ii. Pick two vertices v_i and v_o, each from V_i and V_o respectively

iii. If $|V_n| = 0$ then make an edge (v_i, v_o) in F'

iv. Else for each vertex $v_n \in V_n$, make edges (v_i, v_n) and (v_n, v_o) in F'

v. if $|V_i| > 1$ then connect all of the vertices in V_i by a ring (directed cycle)

vi. if $|V_o| > 1$ then connect all of the vertices in V_o by a ring

(b) Else

i. connect all of the vertices in V_n by a ring

□

Some examples are shown in Fig. 5. Procedure 2 is very tactical: It copies all the original vertices of H and creates edges that represent the flow-reachability in F with a maximum flow. The following facts hold.

Theorem 1

1. *A non-virtual vertex is reachable, i.e. a directed path exists, from the other non-virtual vertex in F if and only if the former is flow-reachable from the latter in F'.*

2. *Two non-virtual vertices belong to the same flow-blocks if and only if they belong to the same strongly connected component in F'.*

3. *The graph obtained from F' by contracting each strongly connected component into a single vertex is $M(H)$.*

□

For example, see Fig. 5. In Case A, any two of u_1, u_2, v_1, and v_3 are mutually flow-reachable in F and they are strongly connected in F'. Also in Case B, v_1 is flow-reachable to any other vertex in F, i.e., there is a directed path to any other vertex in F'.

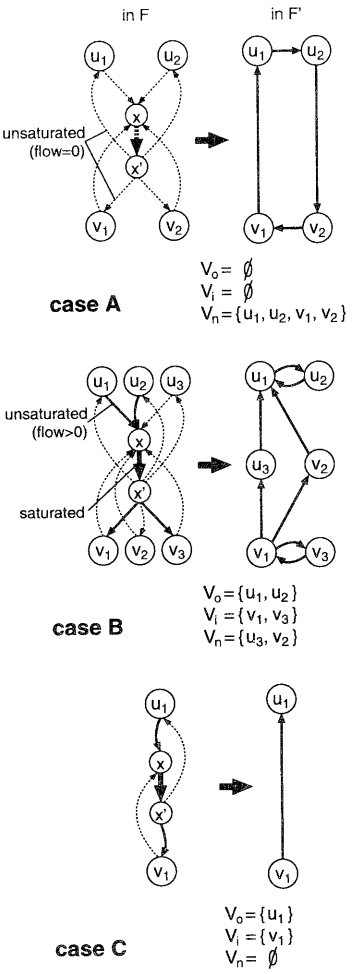

Figure 5: Flow-Reachability Transformation: Some Examples

3.3 Computational Complexity

Along Procedure 1, let us estimate the computational complexity of our approach in obtaining an exact min-cut graph.

Step 1 is possible by the search of the edges and vertices of constant times which needs $O(n+m)$ time. In Step 3, applying flow-reachability transformation can be done by one time graph search. Step 4, finding the strongly connected components and their contraction, are possible in $O(n+m)$ time as well, by the depth-first-search [12], [13]. Therefore, the total computational complexity of Steps 1, 3 and 4 is $O(n+m)$. Then Step 2, to fix a max-flow, will be dominant in the part to get $M(H)$ from H. One of the fastest algorithm works in $O(n^2m)$ [11].

The reason why we claim of the computation complexity after Step 2 is that this is the main part of the improvement in obtaining $M(H)$. In fact we should note that there has been no comparable proposal to get an exact $M(H)$.

4 Applications to Sub-Circuit Extraction

The way a practical circuit is modeled to the graph of the Extraction problem will be as follows. First of all,

two reference vertices shall be assigned as s and t with the purpose that t is to be included in the sub-circuit and s is not.

Once the problem instance is fixed, we apply our way of constructing the min-cut graph. Assume then that the min-cut graph is obtained. The exact solution is as follows. In the beginning, t is only a single vertex in $M(H)$ which has no outgoing edges. Extract t. Of all the new sinks born by the deletion of t, list all the combinations. Check the feasibility of each combination if the sum of weights of t and the chosen vertices is not violating the size constraint of the sub-circuit to be extracted. For each feasible combination, by deleting all the sinks, construct a DAG from $M(H)$. Thus we create a set of Extraction problems. Each time, the number increases by at most the number of created sinks.

See an example shown in Fig. 6 where $M(H)$ is given. The vertices represent subsets of vertices of hyper-graph H (shown in Fig. 1). The current cut is C_1 which extracts $\{s\}$. Deletion of s gives birth to two sinks v_2 and v_3. We have two combinations for further extraction $\{v_1, v_2\}$ or $\{v_1, v_3\}$. The former consists of 5 vertices and the latter of 3. So, if the constraint of the sub-circuit size is 4 or less, the former branching is bounded, i.e. pruned. The latter branches on to $\{v_1, v_2, v_3\}$ of weight 7, over weighted. Hence we conclude that $\{v_1, v_2\}$ is the only exactly optimal solution.

The problem in the above strategy is how to decide s and t. Here we had to introduce an expert heuristic: there are many I/O pins and we divide them into two, s-side and s-side according to the closeness between pins measured by the shortest path length. And created new vertices s and t as the grand source and sink connected to all the vertices of the s-side and t-side, respectively.

Reasons we think that the exhaustive search to work in the Extraction Problem in VLSI design comes from the belief that the resultant $M(H)$ is small for practically large circuits. This must be verified by experiments on the industrial data set that is chosen properly convincing enough to cover the variety in the actual design.

5 Min-cut Graphs in Experiments

The experiments were to ensure the following items on the properly chosen data set and constraint.

1. Smallness of $M(H)$
2. Smallness of computation time.
3. Merit over the the existing one that uses a heuristic min-cut graph (proposed in [6]).

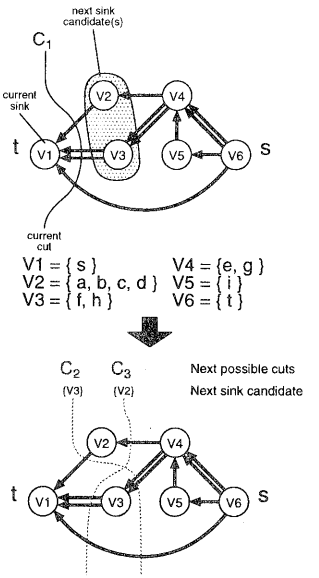

Figure 6: Proposing Method to Finding a Min-Cut in $M(H)$

The test program, implemented in C, for the experiments (a) reads a circuit net-list and constraints (I/O number and size limit), (b) create a hyper-graph from it, (c) apply Yang-Wong transformation, (d) partition the I/O terminals, (e) fix a max-flow, (f) apply flow-reachability transformation, (g) contract its strongly connected components into single vertices, and then (h) exhausts the directed cuts. It was tested on a 400MHz Pentium2 PC with 64MB of memory, under FreeBSD.

See Table 1 featuring the test cases, and Table 2 for the size of the largest and smallest library cells being used.

Library Cell Size		
max.	min.	variety of cells
1000	62	272

Table 2: Cells of the Library : largest and smallest

The test cases are categorized as follows.

A–D, G–O, Q–R : multi-media video decoder (Adder, DCT, Control sequencer, Interface logic)

E–F, P : micro-processor (Adder, Multiplier)

The first experiment took place to observe (a) the numbers of the vertices in $M(H)$ (# of flow-blocks), and (b) time to exhaustively search for a partition (directed cut in $M(H)$) whose size is the largest under the

Case	# cells	# nets	# I/O	fanouts max.	fanouts avr.	size
A	146	210	97	4	2.29	25187
B	294	422	193	4	2.30	48435
C	453	645	289	4	2.29	73365
D	589	845	385	4	2.31	96869
E	938	989	80	108	3.54	128920
F	1654	1704	79	157	3.46	218902
G	2180	2205	76	41	3.72	314768
H	2175	2250	138	329	3.46	290984
I	2232	2286	88	245	3.39	277722
J	2317	2446	189	57	3.16	294744
K	2573	2677	181	180	3.57	348780
L	2350	2493	203	509	3.27	310065
M	2527	2596	110	63	3.41	309402
N	2396	2418	40	486	3.16	280931
O	2254	2393	268	285	3.16	292056
P	183	254	84	26	2.56	31157
Q	2920	3020	175	557	3.27	366513
R	10530	11844	300	201	3.45	1469494

Table 1: Features of the Test Data

I/O number and the size constraints of 64 and 25000, respectively. These constraints were decided based on our experiences in successful utilization of $10K$ gate FPGA. For comparison, we also implemented the Liu-Wong's approach. See Table 3 for the result of the first experiment.

From the results, we can observe that the number of the flow-blocks are affected by the designs, however, in most of the cases, they are small compared to the number of the vertices in the hyper-graph, and the time required to exhaust for an appropriate min-cut is usually very short. Therefore, an exhaustive search is considered practical in real cases. In some cases, we observed that Liu-Wong's approach gave a partition whose size is less than the one obtained by our exact method. This is because their approach is a heuristic while ours is exhaustive. Liu and Wong paid for the reduction of computation cost by the exactness of $M(H)$, while we showed that exactness and small computation cost can be achieved at the same time.

6 Concluding Remarks

The use of the min-cut graph has been known very effective to extract a desired subcircuit when the constraint is the smallness of the cut. To find the min-cut graph of a hyper-graph has been believed to use a formidable computation cost. This paper showed that the preceding works had missed one simple fact, i.e. the flow-reachability of a directed flow graph can be equivalently transformed to the existence of a directed path.

After showing how to construct the min-cut graph, we also showed its application to the sub-circuit extraction problem for the industrial data and showed that even an exhaustive search works very well since the size of the min-cut graph is very small.

The only problem we had not discussed in detail is to give a way to define the source and sink. Our ad-hoc idea is just to group the I/O pins embedded in practical circuits. To give a reasonable way to define s and t is included in future works.

Acknowledgment

The authors would like to express their thanks to Dr. Shigetoshi Nakatake (Kita-Kyushu Univ.), Mr. Hiromasa Takahashi (Fujitsu Labs.), and Dr. Kaoru Kawamura (Fujitsu Labs.) for their valuable advices and support.

This work is a part of CAD21 project at Tokyo Institute of Technology. It is also partly financially supported by New Energy and Industrial Technology Development Organization (NEDO) #98C05-002-2

References

[1] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", Proceedings of the ACM/IEEE DAC 1982, p. 175-181

[2] Y. C. Wei and C. K. Cheng, "Ratio Cut Partitioning for Hierarchical Designs", IEEE Transaction on Computer-Aided Design, Vol. 10, No. 7, July 1991, p. 911-921

[3] C. J. Alpert and A. B. Kahng, "Recent Directions in Netlist Partitioning: A survey", Integration, the VLSI Journal, No. 19, 1995, p. 1-81

	Flow-Block			Ours			Liu-Wong		
Case	#	max-size	min-size	time [ms]	search #	subcircuit size	time [ms]	search #	subcircuit size
A	4	15218	62	10	3	19062	9	3	19062
B	4	33016	62	20	3	19062	12	3	19062
C	3	57867	94	30	2	15498	16	1	15404
D	3	81557	94	40	2	15312	18	1	15218
E	2	125912	3008	50	1	3008	40	1	3008
F	2	215894	3008	80	1	3008	61	1	3008
G	19	308166	62	1250	73728	8408	124	7	6602
H	16	286660	94	2200	16384	4324	111	7	282
I	7	273152	94	100	16	4570	100	4	94
J	4	291548	94	120	3	3196	125	2	282
K	22	343406	94	2183	36043	24327	1649	18	5374
L	8	293103	62	130	66	16962	26	2	12765
M	14	305271	62	140	551	4131	123	7	282
N	7	278863	94	130	34	2068	121	6	282
O	2	289048	3008	120	1	3008	89	1	3008
P	3	26289	125	10	2	4868	14	1	4743
Q	19	344461	94	870	2	22052	14	2	16357
R	14	1466486	94	760	4098	3008	360	5	94
(avr.)	8.4	279773	734	457.9	7274.4	9770.6	167.3	4	6940.4

Table 3: Number of flow-blocks and time to obtain an appropriate min-cut, under constraints: size $\leq$ 25000, I/O$\leq$ 64

[4] H. Yang and D. F. Wong, "Efficient Network Flow Based Mincut Balanced Partitioning", Proceedings of the ACM/IEEE ICCAD 1994, p. 50-55

[5] H. H. Yang and D. F. Wong, "Efficient Network Flow Based Min-Cut Balanced Partitioning", IEEE Transaction on Computer-Aided Design, Vol. 15, No. 12, December 1996, p. 1533-1540

[6] H. Liu and D.F. Wong, "Network-Flow-Based Multiway Partitioning with Area and Pin Constraints", IEEE Transaction on Computer-Aided Design, Vol. 17, No. 1, January 1998, p. 50-59

[7] H. Nagamochi, K. Nishimura and T. Ibaraki, "Computing All Small Cuts in an Undirected Network", SIAM Journal of Discrete Mathematics, Vol. 10, No. 3, August 1997, p. 469-481

[8] E. Ihler, D. Wagner and F. Wagner, "Modeling hypergraphs by graphs with the same mincut properties", Information Processing Letters, No. 45, March 1993, p. 171-175

[9] N. S. Woo and Jaeseok Kim, "An Efficient Method of Partitioning Circuits for Multiple-FPGA Implementations", Proceedings of DAC 1993, p. 202-207

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, "Introduction to Algorithms", M.I.T Press 1990

[11] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, "Network Flows: Theory, Algorithms and Applications", Prentice Hall International Inc. 1993

[12] A. Dolan and J. Aldous, "Networks and Algorithms: An Introductory approach", John Wiley & Sons. Ltd, 1993

[13] T. C. Hu, "Integer Programming and Network Flows", Addison-Wesley Publishing Company, Inc., 1970

[14] N. Togawa, M. Sato, and T. Ohtsuki, "A Circuit Partitioning Algorithm with Replication Capability for Multi-FPGA Systems", IEICE Trans. on Fundamentals, Vol. E78-A No. 12, Dec. 1995

[15] N. Togawa, M. Sato, and T. Ohtsuki, "A Performance-Oriented Circuit Partitioning Algorithm with Logic-Block Replication for Multi-FPGA Systems", IEEE APCCAS 1996, p. 294-297

[16] K. R. Azegami, A. Takahashi and Y. Kajitani, "Maxflow based Method for Enumerating Mincut Edges of Graph Modeled Logic Circuit", IEICE Technical Report VLD Vol. 98, No. 448, December 1998, p. 131-138

[17] K. R. Azegami, A. Takahashi and Y. Kajitani, "Enumerating The Min-Cuts for Applications to Graph Extraction under Size Constraints", Proceedings of the IEEE ISCAS 1999, p. VI.174-VI.177