PAPER

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

Kengo R. AZEGAMI[†*a)], **Atsushi TAKAHASHI**[†], *and* **Yoji KAJITANI**[†], *Regular Members*

SUMMARY 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, our proposed algorithm gives an exact one without substantial computation overhead. By using the exact min-cut graph, an exhaustive algorithm finds an optimal sub-circuit that is extracted by a min-cut from the circuit. 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 [3], [6], [10], [13], [14], and [16]. See [12] for an extensive survey. Since the problem is so diverse, 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 [3]).

While applications of partition in VLSI design are derived mainly from 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. 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 maximal sub-circuit that includes one designated vertex inside and the other outside under the constraint of the size of the cut being minimum and the size of the circuit within a specified range.

To approach this problem, the only algorithm so far proposed is to fix a maximum flow between the two designated vertices, one being the source and the other

Manuscript received December 10, 1999.
Manuscript revised September 6, 2000.

[†]The authors are with the Department of Electrical and Electronic Engineering, Tokyo Institute of Technology, Tokyo, 152-8552 Japan.
[*]Presently, with System LSI Development Laboratory of Fujitsu Laboratories LTD.
a) E-mail: azegami@lab.ss.titech.ac.jp

being the sink of the flow, and find the minimum cut nearest to the source (sink) which separates the flow-reachable set of vertices from the source (to the sink) and the rest. Other minimum cuts are found by choosing some vertices and finding the flow-reachable sets of vertices from them. Not to miss any minimum cut, the choice of the vertices shall be from all the combinations of the vertices. Even though some bounding techniques are applied, this meek algorithm needs innumerable times of the whole graph traversal, which is not tolerable by its huge computation time.

To display all the minimum cuts between two designated vertices, a useful data structure known by the name of min-cut graph [17] exists. It is a directed acyclic graph $M(H)$ defined for any hyper-graph H such that all of its directed cuts 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 [17]. 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 [17] 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. Therefore they proposed a heuristic to construct an approximated min-cut graph as a compromise, which in some cases lack preciseness in displaying the min-cuts.

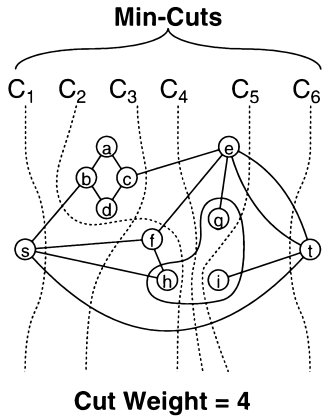

Fig. 1 Hyper-graph H and its s-t min-cuts of cut weight 4. An edge with two end vertices are drawn as a single line, while an edge with more than two end vertices are drawn as a curve.

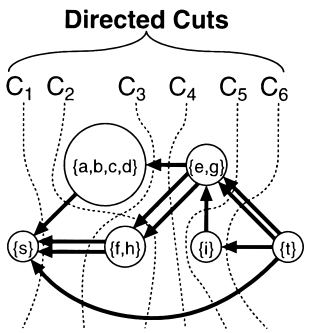

Fig. 2 The min-cut graph $M(H)$ for H. Each vertex in $M(H)$ is labeled by a set of vertices in H surrounded by its min-cuts.

The first objective of this paper is to give an empirical proof that the exactness of min-cut graph is easy to achieve by noting if we come to the idea that a vertex of the min-cut graph corresponds to a strongly connected component of the flow graph with respect to the flow-reachability. We can expect the algorithm to find better partitions by utilizing an exact min-cut graph. Once $M(H)$ is obtained, we would use it to extract a sub-circuit from H.

In worst case, the number of directed cuts in $M(H)$ is exponential to the number of its vertices. The second objective of this paper is to show by experiments that an exhaustive search for its desired directed cut will work in a reasonable computation time. It is done by extracting a vertex from $M(H)$, closest to the source, after another considering maximization of the evaluation. 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. This reveals an empirical fact that using the exact min-cut graph to extract a desired sub-circuit is practical enough for industrial use.

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 exists 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 flow-graph with 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 $O(n^2m)$ [5], [7], and Goldberg et al.'s method in $O(nm \log \frac{n^2}{m})$ [4], 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 $v \in V_1$. 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 maximal set of mutually flow-reachable vertices. An equivalent definition is that a flow-block is a strongly connected component in terms of flow-reachability. Since two vertices are in the same flow-block if and only if they are mutually flow-reachable, a flow-block is not separated by a min-cut, and any two flow-blocks can be separated by a min-cut.

3. Constructing Min-Cut Graph: 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.

In [17], an algorithm *DMC* (Desirable Min-Cut) is described. *DMC* is a heuristic which creates an approximated min-cut graph from a given hyper-graph. Since it is a heuristic, the graph obtained by it may not be precise and results in lack of disclosed min-cut.

Unlike the approach in *DMC*, we determine an exact min-cut graph. The approach consists of the following four steps. The details will be given later.

Procedure 1: (Construction of $M(H)$: Our Proposal)

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 all the hyper-edges in H.
2. (Max-Flow) Compute a max-flow in F.
3. (Flow-Block Candidates) Obtain F' by applying *Flow-Reachability transformation* to all the local structures in F.
4. (Strongly Connected Component) Find all the strongly connected components 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 [2], [11], [15] (Fig. 3). This transformation replaces a hyper-edge e_H with a structure, called the *local structure* of e_H. The formal definition is as follows.

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 $e_l = (x, x')$ with $cap(e_l) = w$.

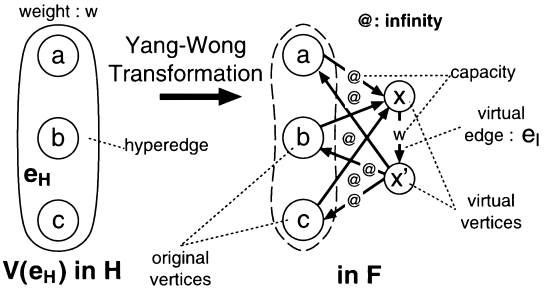

Fig. 3 Yang-Wong transformation to a hyper-edge of three end vertices to a local structure.

3. For each end vertex v of e_H, create a pair of directed edges $e = (v, x)$ and $e' = (x', v)$ with $cap(e) = cap(e') = \infty$.
4. delete e_H

□

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

This equivalent transformation could be interpreted as a series of three well-known equivalent transformations from the edge-capacity undirected hyper-graph to the vertex-capacity undirected graph, from it to the vertex capacity directed graph, and from it to the edge-capacity directed graph. But the authors would like to call it as the Yang-Wong transformation since, from the authors knowledge, Yang and Wong [11], [15] was the first to define it as a single operation, which has made the description very smart.

3.2 Flow-Block Candidates

After computing the max-flow, we transform F to a directed graph F' by applying the *flow-reachability transformation* to all the local structures before finding the strongly connected components. Yang-Wong transformation adds extra vertices and edges, i.e. virtual vertices, virtual edges and infinity-capacity edges. By applying the *flow-reachability transformation* to all the local structures in F, such extra components are removed and the flow-reachability relations between the original vertices are embossed.

Example given in Fig. 4 would be helpful to understand.

Definition 2: (Flow-Reachability Transformation of a local structure l: Step 3 in Procedure 1.)

1. let e_l be the virtual edge of l
2. if e_l is saturated then
   a. let V_o, V_i and V_n be the sets of original vertices in l, from which the flow comes in to e_l, goes out from, and neither, respectively.
   b. Pick two vertices v_i and v_o, each from V_i and V_o, respectively

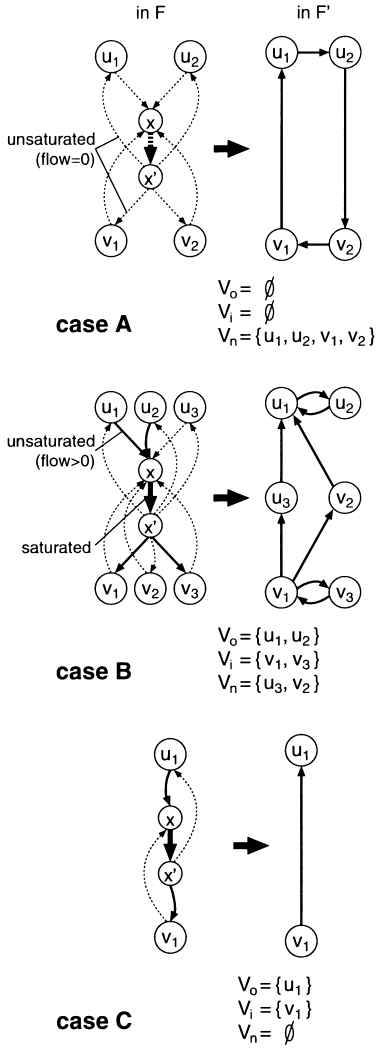

Fig. 4 Flow-reachability transformation: Some examples.

c. If $|V_n| = 0$ then create an edge (v_i, v_o)
d. Else for each vertex $v_n \in V_n$, create edges (v_i, v_n) and (v_n, v_o)
e. if $|V_i| > 1$ then connect all of the vertices in V_i by a ring (directed cycle)
f. if $|V_o| > 1$ then connect all of the vertices in V_o by a ring

3. Else
   a. connect all of the original vertices in l by a ring
4. delete e_l, all the virtual vertices and the infinity-capacity edges in l.

□

The following facts hold.

Theorem 1:

1. An original vertex is flow-reachable, i.e. a directed path exists, from the other original vertex in F if and only if the former is reachable from the latter in F'.
2. Two original 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. 4. In Case A, any two of u_1, u_2, v_1, and v_2 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'. Therefore, by contracting each of the strongly connected component in F' into a single vertex, $M(H)$ is obtained.

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 [1], [8]. 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 algorithms works in $O(nm \log \frac{n^2}{m})$ [4].

4. Applications to Sub-Circuit Extraction

The way a practical circuit is modeled to the graph of the Extraction Problem is as follows. First, two designated vertices are assigned as the source and sink of F intending that the source is to be included in the sub-circuit which we are concerned, and sink is not.

Once the problem instance is fixed, we apply our way of constructing the min-cut graph. Assume that the min-cut graph $M(H)$ is obtained. The exact solution is obtained as follows. In a DAG, define a leaf as a vertex in it with no outgoing edges. So, a vertex that corresponds to the source of the flow is the only leaf in $M(H)$. Extract it, and construct a new DAG by deleting it. Of all the new leaves born after the deletion, list all of its combinations of the leaves. Check the feasibility of each combination if the sum of weights of chosen leaves and the extracted leaves does not violate the size constraint of the sub-circuit to be extracted. For each feasible combination, by deleting all the leaves and the ancestors of the unchosen leaves, construct a new DAG for further extraction. Apply the mentioned procedure recursively while a new DAG is constructed.

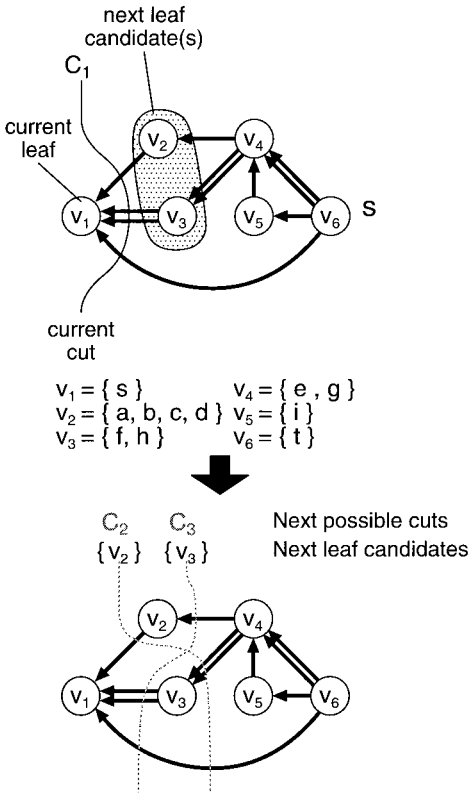

Fig. 5 Proposing method to finding a min-cut in $M(H)$.

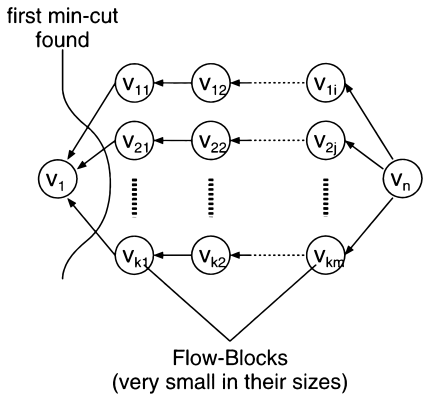

Fig. 6 Example of $M(H)$ where our approach takes a long time to find an optimal cut.

See an example shown in Fig. 5 where $M(H)$ is given. The vertices represent subsets of vertices of hyper-graph H (shown in Fig. 1). Suppose the given constraint of the sub-circuit size is 4. The current cut is C_1 which extracts v_1. Deletion of v_1 gives birth to two leaves v_2 and v_3. We now have three combinations for further extraction $\{v_2\}$, $\{v_3\}$ and $\{v_2, v_3\}$. We start the evaluation from the one with less number of vertices among the combinations. Suppose we chose $\{v_2\}$ as the first evaluation objective. The number of vertices in H for $\{v_2\}$ with $\{v_1\}$ is 5, and this choice is discarded. Since any combination of vertices including $\{v_2\}$ will violate the size constraint, any further evaluations including these vertices, for instance $\{v_2, v_3\}$, are bounded, i.e. pruned. Finally, $\{v_3\}$ is evaluated. The number of vertices in H for $\{v_3\}$ with $\{v_1\}$ is 3, and this combination is stored. A new DAG to be created for $\{v_3\}$ is empty since the ancestors of $\{v_2\}$ are deleted. Hence, we conclude that $\{v_1, v_3\}$ is the only exactly optimal solution.

Our approach has two problems. The first problem is how to decide the source and the sink of the flow. Here we had to introduce an expert heuristic: there are many I/O pins and we divide them into two, the source side and the sink side according to the closeness between pins measured by the shortest path length. Then create new vertices s and t as the grand source and sink connected to all the vertices of the source side and the sink side, respectively.

The second problem is that the computational complexity of our approach is basically exponential, and there are cases where our approach takes a long time to find a solution. See Fig. 6 for such an example.

The reason we still think that the exhaustive search to work efficiently in the Extraction Problem in VLSI design comes from the belief that

- the structure of $M(H)$ depends on the structure of the hyper-graph, but is small and simple in common for practically large circuits.

and

- case where $M(H)$ consists of innumerable small-sized flow-blocks is rare.

These expectations shall be empirically proved on properly chosen industrial data set.

5. Min-Cut Graphs in Experiments

The experiments are 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 an approximated min-cut graph (proposed in [17]).

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

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 experiment took place to observe (a) the numbers of the vertices in $M(H)$, and (b) time to search for a partition (directed cut in $M(H)$) whose size is the largest under the 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 experiment. The table describes:

- number of flow-blocks (#)
- size of the largest flow-block (max-size)
- size of the smallest flow-block (min-size)

of each test case, and

- time to find a desired min-cut (time[ms]), including the time to obtain a min-cut graph
- number of sub-circuits tested for feasibility (search #)
- size of the extracted sub-circuit (sub-circuit size)

for both ours and Liu-Wong's approaches of the corresponding test cases.

As we had expected prior to the experiments, the number of the flow-blocks are affected by the designs, however, they are small compared to the number of the vertices in the hyper-graph in most cases. For example in case A, 146 vertices in hyper-graph but only 4 vertices in min-cut graph.

Our approach exhaustively searches for a desired cut in $M(H)$, thus the number of searches can be large in some cases. However, the smallness and the simplicity of $M(H)$ overrides the inferiority in computation time of an exhaustive search. For instance, comparing case K to case L, the number of flow-blocks increases from 8 to 22, and the search # increases from 66 to 36043, while the computation time increases from 130 ms to only 2183 ms.

Comparing the time in finding a desired cut, Liu-Wong's approach is advantageous. However, it is only few hundreds of milli-seconds, which may be considered negligible. Similarly, comparing the size of the

Table 1 Features of the test data.

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 2 Cells of the library: Largest and smallest.

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

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

	Flow-Block			Ours			Liu-Wong		
Case	#	max-size	min-size	time [ms]	search #	sub-circuit size	time [ms]	search #	sub-circuit 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

extracted sub-circuit, our result is improved by 41% (9770.6 for ours and 6940.4 for Liu-Wong's). This is because our algorithm exhaustively searches for an exact optimal solution while Liu-Wong's searches heuristically, and in many cases, lose optimality.

From these observations, we can conclude that Liu and Wong paid for the reduction of computation cost by the exactness of $M(H)$, which results in disadvantage of the size of the extracted sub-circuit, while we showed that exactness and small computation cost can be achieved simultaneously in real industrial cases.

6. Concluding Remarks

The use of the min-cut graph has been known very effective to extract a desired sub-circuit 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 the source and the sink 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] T.C. Hu, Integer Programming and Network Flows, Addison-Wesley Publishing Company, Inc., 1970.
[2] E.L. Lawler, "Cutsets and partitions of hypergraphs," Networks, no.3, pp.275–285, 1973.
[3] C.M. Fiduccia and R.M. Mattheyses, "A linear time heuristic for improving network partitions," Proc. ACM/IEEE DAC, pp.175–181, 1982.
[4] A.V. Goldberg and R.E. Tarjan, "A new approach to the maximum flow problem," Proc. 18th Annual ACM Symposium on Theory of Computing, pp.136–146, 1986.
[5] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, M.I.T. Press, 1990.
[6] Y.C. Wei and C.K. Cheng, "Ratio cut partitioning for hierarchical designs," IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst., vol.10, no.7, pp.911–921, July 1991.
[7] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall International Inc., 1993.
[8] A. Dolan and J. Aldous, Networks and Algorithms: An Introductory approach, John Wiley & Sons, 1993.
[9] E. Ihler, D. Wagner, and F. Wagner, "Modeling hypergraphs by graphs with the same mincut properties," Information Processing Letters, no.45, pp.171–175, March 1993.
[10] N.S. Woo and J. Kim, "An efficient method of partitioning circuits for multiple-FPGA implementations," Proc. DAC, pp.202–207, 1993.
[11] H. Yang and D.F. Wong, "Efficient network flow based mincut balanced partitioning," Proc. ACM/IEEE ICCAD, pp.50–55, 1994.
[12] C.J. Alpert and A.B. Kahng, "Recent directions in netlist partitioning: A survey," Integration, the VLSI J., no.19, pp.1–81, 1995.
[13] N. Togawa, M. Sato, and T. Ohtsuki, "A circuit partitioning algorithm with replication capability for multi-FPGA systems," IEICE Trans. Fundamentals, vol.E78-A, no.12, Dec. 1995.
[14] N. Togawa, M. Sato, and T. Ohtsuki, "A performance-oriented circuit partitioning algorithm with logic-block replication for multi-FPGA systems," IEEE APCCAS, pp.294–297, 1996.
[15] H.H. Yang and D.F. Wong, "Efficient network flow based min-cut balanced partitioning," IEEE Trans. Comput.-Aided Des., Integrated Circuits & Syst., vol.15, no.12, pp.1533–1540, Dec. 1996.
[16] H. Nagamochi, K. Nishimura, and T. Ibaraki, "Computing all small cuts in an undirected network," SIAM J. Discrete Mathematics, vol.10, no.3, pp.469–481, Aug. 1997.
[17] H. Liu and D.F. Wong, "Network-flow-based multiway partitioning with area and pin constraints," IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst., vol.17, no.1, pp.50–59, Jan. 1998.
[18] K.R. Azegami, A. Takahashi, and Y. Kajitani, "Maxflow based method for enumerating mincut edges of graph modeled logic circuit," IEICE Technical Report, VLD98-116, 1998.
[19] K.R. Azegami, A. Takahashi, and Y. Kajitani, "Enumerating the min-cuts for applications to graph extraction under size constraints," Proc. IEEE ISCAS, pp.VI.174-VI.177, 1999.

Kengo R. Azegami received his B.E. and M.E. degrees from Nagaoka University of Technology, Niigata, Japan, all in electronic engineering, in 1990, and 1992, respectively. Received D.E. from Tokyo Institute of Technology in 2001. He is currently with System LSI Development Lab. of Fujitsu Labs. LTD. His research interests include applications of graph theory in VLSI circuit designs.

Atsushi Takahashi received his B.E., M.E., and D.E. degrees in electrical and electronic engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He had been with the Tokyo Institute of Technology as a research associate from 1991 to 1997 and has been an associate professor since 1997 in the Department of Electrical and Electronic Engineering. His research interests include in VLSI layout design and combinational algorithms. He is a member of the IPSJ and IEEE.

Yoji Kajitani received his B.E., M.E. and D.E. degrees from the Tokyo Institute of Technology, Tokyo, Japan, all in electrical engineering, in 1964, 1966 and 1969, respectively. He has been a professor in the Department of Electrical and Electronic Engineering at the Tokyo Institute of Technology since 1985, and has been a professor at the Japan Advanced Institute of Science and Technology from 1992 to 1995. His current research interests are in combinatorial algorithm applied to VLSI layout design. He was awarded IEEE Fellowship in 1992.