論文 / 著書情報
Article / Book Information

Title	An Improvement of Network-Flow Based Multi-Way Circuit Partitioning Algorithm
Authors	Kengo R. Azegami, Masato Inagi, Atsushi Takahashi, Yoji Kajitani
Citation	IEICE Trans. Fundamentals, Vol. E85-A, No. 3, pp. 655-663
Pub. date	2002, 3
URL	http://search.ieice.org/
Copyright	(c) 2002 Institute of Electronics, Information and Communication Engineers

PAPER

An Improvement of Network-Flow Based Multi-Way Circuit Partitioning Algorithm

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

SUMMARY In this paper, we propose an improved network-flow based multi-way circuit partitioning algorithm whose objective is to minimize the number of sub-circuits. It iteratively extracts a size-maximal feasible sub-circuit one at a time. In our approach, two devices are applied. One is in the use of an exact min-cut graph, and the other is in the idea of keeping the number of I/O pins of the residual circuit as small as possible after one-time extraction. We implemented our algorithm in C for experiments, and tested it with several industrial cases and MCNC benchmarks. Compared to the known approach, we observed more than 10% reduction in average of the sub-circuit number.

key words: *circuit partition, hyper-graph partition, network flow, min-cut, integrated circuit design*

1. Introduction

Partitioning a circuit into multiple sub-circuits under size and I/O pin number constraints (multi-way circuit partitioning [2], [4], [9]) is an important issue in VLSI designs. Multi-chip design (i.e. multi-FPGA [7]), the issue we focus in this paper, is one of its applications. Since the constraint by the I/O capacity of a real-world chip is severe than that by the the size capacity, partitioning a circuit to fit these constraints often results in low device usage. This increases the number of chips and leads to high product cost. Thus circuit partition for multi-chip design suffers from a conflicting requirement, large sub-circuit with less pins.

In this paper, we discuss a solution adopting the familiar max-flow min-cut theorem [1], [3], [5], [6].

One of the practical multi-way circuit partitioning algorithm using max-flow is discussed by Liu and Wong [11]. Their algorithm iteratively 'extracts' a size-maximal feasible sub-circuit one at a time. A sub-circuit is feasible if it satisfies both size and I/O pin number constraints.

Their objective is to minimize the number of sub-circuits by iteratively extracting a sub-circuit as large in size as possible. Their algorithm consists of four main parts, hyper-graph to flow-graph transformation, min-cut search, sub-circuit size enlargement and source s and sink t determination. In it, a hyper-graph that models a circuit is first transformed into a flow-graph for max-flow computation. Next, they randomly pick from the vertices of the flow-graph 10 pairs of source s and sink t. Then, for each pair of s and t, find a feasible sub-circuit as large in size as possible using procedure *DMC* (Desirable Min-Cut). If there remains room for such sub-circuit to the given constraint, the size of the sub-circuit is enlarged by modifying the cut and accommodating more circuit elements. Among the 10 sub-circuits, the one that is the largest in size is extracted. The algorithm repeats this process until all circuit elements are extracted.

In their algorithm, there are several problems that may increase the number of sub-circuits. First, it is in procedure *DMC*. In it, a min-cut that extracts a feasible sub-circuit as large in size as possible is searched for. However, there are cases where size-maximality is not achieved. Second, it is in their way of sub-circuit size enlargement in which some vertices are collapsed to s or t to modify the cut. However, they try only one combination of vertices, and often in practice, combinations that give better extraction results are discarded. Third, it is in their choice of s and t. Their approach does not pay much attention to this. Since the algorithm applies the max-flow min-cut theorem for circuit partitioning, the results strongly depend on s and t.

The aim of this paper is to propose an algorithm which partitions a circuit into as less number of sub-circuits as possible, while each sub-circuit satisfying the given pin number and size constraints. The proposed algorithm basically inherits the Liu-Wong's approach with improvements over their way that are essential in practice. The first improvement is in the use of procedure *OMC* (Optimal Min-Cut). Unlike their *DMC*, our *OMC*, based on [12]–[14] guarantees the size maximality. The second improvement is in our way of sub-circuit size enlargement. Being aware of the fact that the extraction result depends on the combinations of the vertices to collapse to s and t, our algorithm tries several combinations and then determines the one to use for the actual sub-circuit size enlargement. The third improvement is in the way we choose s and t. In our algorithm, an expert heuristic is adopted to determine them each time a sub-circuit is extracted.

Manuscript received November 22, 2000.
Manuscript revised August 17, 2001.
Final manuscript received November 19, 2001.

[†]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@flab.fujitsu.co.jp

We implemented both our and Liu-Wong's algorithm and compared the performance. We observed that our approach, though slower than Liu-Wong's, outputs partitioning results with significantly less sub-circuits.

2. The Mainstream of Our Approach

A circuit C is modeled by a hyper-graph $H = (V(H), E(H))$ where vertex set $V(H)$ represents the circuit elements and edge set $E(H)$ represents the nets. Each vertex is labeled by the size of its corresponding circuit element. Our problem is to obtain a partition of H using network flows. Algorithm *PART* in the following shows our way of multi-way circuit partitioning. Each of the underlined procedures will be detailed later.

Functions $size(V_i)$ and $io(V_i)$ return the size and number of I/O pins of the sub-circuit defined by $V_i \subseteq V(H)$, respectively. Also, a cut of a graph with vertex set V is shown as a pair of non-overlapping sets of vertices such as $[V_1, V_2]$ ($V_1, V_2 \neq \emptyset$, $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$).

Algorithm *PART*: (Our way of Multi-Way Circuit Partitioning)

Input: circuit C, I/O pin number constraint lim_{io}, size constraint lim_{size}.
Output: set $\mathcal{P}$ of sub-circuits.

1. obtain hyper-graph $H = (V(H), E(H))$ from C.
2. $\mathcal{P} \leftarrow \emptyset$.
3. obtain the initial flow-graph $F = (V(F), E(F))$ from H by:
   a. creating the source and sink vertices, s and t, in F.
   b. applying *Yang-Wong transformation* [8], [10] to each $e_H \in E(H)$.
4. $P \leftarrow \emptyset$.
5. pick a vertex v_s using *PICK*, and obtain a new flow-graph F from the previous one by adding an edge (s, v_s) of capacity ∞.
6. find an optimal min-cut $[V_1, V_2]$ in F by *OMC*.
7. if $size(V_1) < lim_{size}$ and $io(V_1) < lim_{io}$ then
   a. obtain V_1' by enlarging V_1 by *EXPAND*.
   b. if $BETTER(V_1', P)$ is *True* then
      i. $P \leftarrow V_1'$.
   c. go to step 5.
8. if $P = \emptyset$ then
   a. abort the algorithm: C has a circuit element that violates the given I/O pin number or size constraints.
9. else
   a. $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$.
10. delete the vertices and edges that correspond to P in H.
11. if $H \neq \emptyset$ then go to step 3.
12. end. □

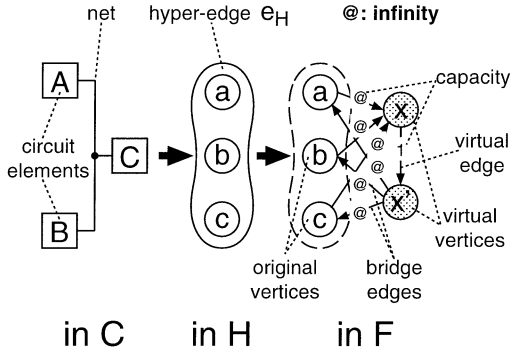


Fig. 1 Yang-Wong transformation applied to a hyper-edge with 3 end vertices representing a network without an I/O pin.

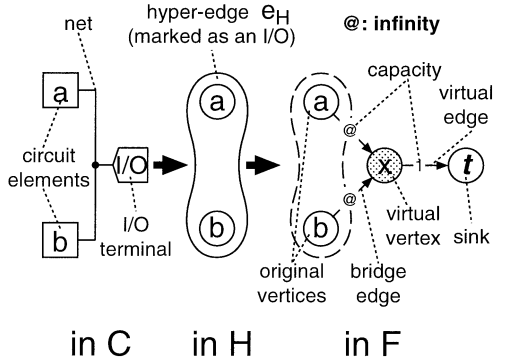


Fig. 2 Yang-Wong transformation applied to a hyper-edge with 2 end vertices representing a network with an I/O pin.

Algorithm *PART* searches for a set of cuts in C, one cut at a time, using a *flow-graph* $F = (V(F), E(F))$ obtained from H by adding to it the source s and sink t, applying the Yang-Wong transformation to each of the hyper-edges, and then adding some ∞ capacity edges from s to some vertices. See Definition 1 for the definition of the Yang-Wong transformation and Figs. 1 and 2 for the illustrative examples of two transformation cases. Definition 2 defines the resultant flow-graph.

Definition 1: (Yang-Wong transformation of $e_H \in E(H)$.)

1. let $V_{orig}(e_H)$ be the set of end vertices of e_H.
2. if e_H represents a circuit net without an I/O pin in C then (Fig. 1):
   a. create a pair of vertices x and x'.
   b. create a directed edge (x, x') with capacity 1.
   c. for each vertex $v \in V_{orig}(e_H)$, create a pair of ∞ capacity directed edges $e = (v, x)$ and $e' = (x', v)$.
3. else (Fig. 2):
   a. create a vertex x.

b. create a directed edge (x, t) with capacity 1.
c. for each vertex $v \in V_{orig}(e_H)$, create an ∞ capacity directed edge $e = (v, x)$.

4. remove e_H. □

It is known that the Yang-Wong transformation could be interpreted as a series of well known equivalent transformations [14].

Definition 2: (Flow-Graph F obtained from a hypergraph H)
Flow-graph $F = (V(F), E(F))$ is an edge-capacity directed graph where $V(F)$ denotes the set of vertices, and $E(F) \subseteq V(F) \times V(F)$ denotes the set of edges. $V(F)$ is a union of T, V_H, and V_V where each denotes the source s and sink t, set of vertices inherited from H (referred to as *original vertices*), and set of vertices added by the Yang-Wong transformation (referred to as *virtual vertices*), respectively. □

In the initially obtained flow-graph F after step 3, source s has no outgoing edges. A flow-graph that is actually used to search for cuts in C is obtained from this initial one by adding edges of capacity ∞ from s to a set of original vertices selected by *PICK*. The amount of s-t max-flow is equal to the I/O pin number of the sub-circuit to be extracted by the min-cut.

3. Details of Our Devices

There are several issues devised in our algorithm that are not found in the known flow-based multi-way partitioning algorithms such as in [11]. They are included in procedures *PICK*, *OMC*, *BETTER* and *EXPAND*, where each is detailed in the subsections that follow.

3.1 Procedure *PICK*

Both our and Liu-Wong's approaches adopt the max-flow min-cut theorem, thus it is obvious that the result depends on how s and t are decided. In Liu-Wong's approach, s and t are decided randomly. In fact, 10 random pairs of s and t are initially decided, then the sub-circuit extraction is done for each of them.

Since there are so many possible s-t combinations, simply and randomly deciding 10 pairs is not sufficient to obtain good partition results. Some sort of strategy that aims the reduction of the overall number of sub-circuits should exist instead of random decision.

In contrast to their approach, we applied a heuristic which is likely to extract a large sub-circuit. This heuristic determines a set of vertices, one vertex at a time, to be fixed to s by ∞ capacity edges. Note that it picks a vertex to be fixed to s. The set of vertices to be fixed to t are determined while applying the Yang-Wong transformation (see Definition 1 and Fig. 2). See below for its details.

Procedure 1. (Source s Selection *PICK*)
Input: a set of vertices $V_u \subseteq V(F)$ whose corresponding elements in C are connected to the I/O pins of the circuit, and a set of vertices $V_s \subseteq V_u$ whose elements are being fixed to s by ∞-capacity edges.
Output: a vertex $v_s \in V_u$ to be fixed to s.

1. if $|V_s| = 0$ then
   a. pick $v_s \in V_u$ with the highest degree of I/O-marked edges (see Fig. 2).
2. else
   a. pick $v_s \in V_u \setminus V_s$ whose value of the evaluation function $F_{eval}(v_s, V_s)$ is the largest.
3. return v_s. □

This heuristic, for the first time of vertex selection, picks a vertex with many I/O pins, and for the second time and later, picks a vertex according to the value of the evaluation function F_{eval}:

$$F_{eval}(v_s, V_s) = \frac{share(v_s, V_s) + 1}{dist(v_s, s)}$$

In the function, $share(v_s, V_s)$ describes the total number of shared nets that, in C, connect v_s and the vertices in V_s, while, $dist(v_s, s)$ describe the distance between v_s and s measured by the shortest path method in F.

A constant added to the numerator is to let the procedure output correct evaluation result even if there are no shared nets. Tie-breaking is done by further evaluating the amount of max-flow from each of the candidates to t. The one with the largest amount of flow is selected. This gives priority in picking a vertex which reduces the number of I/O pins of the sub-circuit to be extracted. Further tie breaking is done by random selection.

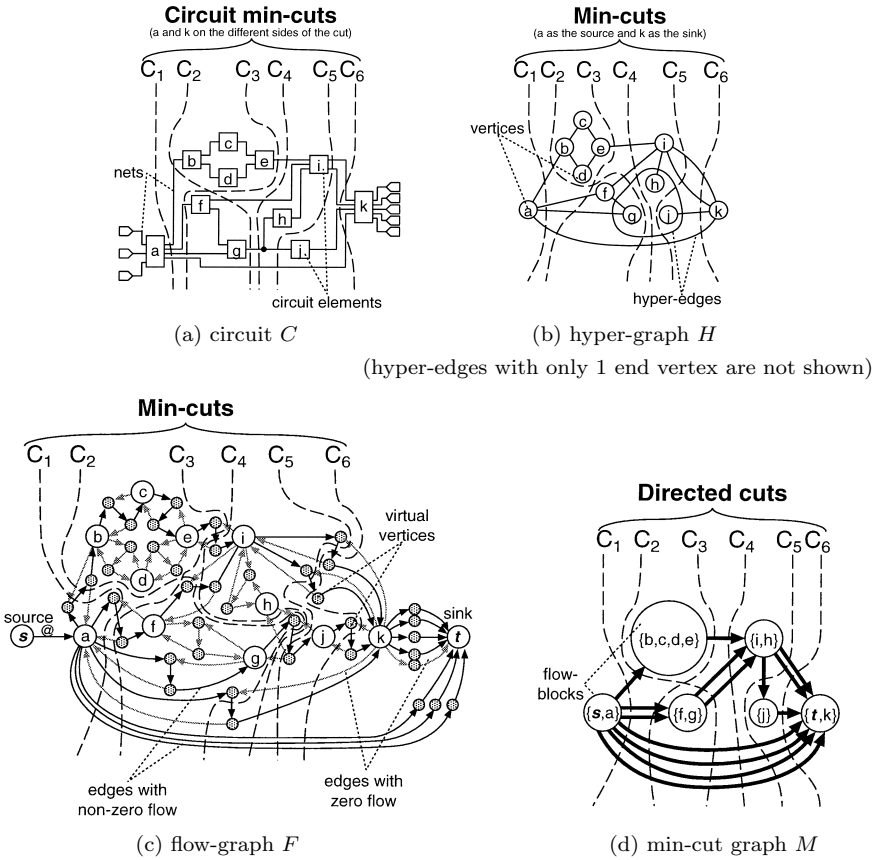
The set of vertices picked using *PICK* tends to extract a sub-circuit consisting of circuit elements tightly connected to each other (many circuit nets exist among the circuit elements). We consider this approach to be straight-forward in reducing the circuit nets between the extracted and residual circuits. The same device is also adopted in Procedure *BETTER* described in the later section.

3.2 Procedure *OMC*

In our approach, we search for an optimal min-cut by procedure *OMC*. The details of the differences between *OMC* and *DMC* are given later.

See below for its details. A vertex u is flow-reachable from another vertex v if there exist a flow augmentable path from v to u.

Procedure 2. (Optimal Min-Cut Search *OMC*)



(a) circuit C (b) hyper-graph H

(hyper-edges with only 1 end vertex are not shown)

(c) flow-graph F (d) min-cut graph M

Fig. 3 (a) circuit C, (b) hyper-graph H, (c) flow-graph F and (d) min-cut graph M. Described set of cuts have one-to-one correspondence to each other. Each of the cuts extracts a sub-circuit with 7 I/O pins.

Input: F with s and t, lim_{size}.
Output: optimal min-cut $[V_1, V_2]$.

1. compute s-t max-flow of F by Ford-Fulkerson method [3].
2. create a min-cut graph M from F by collapsing each set of mutually flow-reachable vertices into a single vertex, and then deleting all edges with zero flow.
3. by exhaustively searching for a directed cut in M, find a cut $[V_1, V_2]$ in F, where $s \in V_1$ and $t \in V_2$, that maximizes $size(V_1)$ under constraint lim_{size}.
4. return $[V_1, V_2]$. □

In procedure *OMC*, a directed acyclic graph M, called a *min-cut graph*, is created by collapsing each set of mutually flow-reachable vertices into a single vertex, and then removing all edges with zero flow. Such sets can be easily found by searching for the strongly connected components in terms of flow. See Definition 3 for its formal definition.

Definition 3: Min-cut graph M obtained from a flow-graph F)
Min-cut graph $M = (V(M), E(M))$ is a directed acyclic graph where $V(M)$ denotes the set of vertices, and $E(M) \subseteq V(M) \times V(M)$ denotes the set of edges. Each vertex in M, referred to as a *flow-block*, corresponds to a non-overlapping subset of $V(F)$. □

M has the following characteristic: let $e_M = (u_M, v_M)$ be an edge in M, and u_M, v_M correspond to $U_F, V_F \subseteq V(F)$, respectively. Then, any two vertices in U_F are mutually flow-reachable in F. At the same time, for any two vertices $u_F \in U_F$ and $v_F \in V_F$, u_F is flow-reachable from v_F in F but not vice versa.

There is a one-to-one correspondence between any directed cut in M, min-cut in F, and a min-cut in H. It is known that all the min-cuts in H are displayed as directed cuts in M. Thus min-cuts in H can be browsed by browsing the directed cuts in M. In M, any single flow-block is not separated by a min-cut, and any two flow-blocks are separated by a min-cut. These facts are proven in [14].

Take an example in Fig. 3. See a circuit C (Fig. 3(a)), its corresponding hyper-graph H (Fig. 3(b)), its corresponding flow-graph F (Fig. 3(c)), and its corresponding min-cut graph M (Fig. 3(d)).

It is common for both *DMC* [11] and our *OMC* in obtaining a min-cut graph and browsing it for a desirable cut. The differences are that *DMC* sacrifices the exactness for speed in obtaining the min-cut graph and

searching for a desirable cut, while *OMC* exploits the exactness in either process. The possible disadvantage of *OMC* to *DMC* is in the computational cost since the worst case estimation of the computational cost is apparently exponential to the number of vertices. Thus the range of *OMC* being available practically depends on the instances. The base we believe in *OMC* is an empirical proof in [14] that *OMC* works well in terms of speed for real circuits.

3.3 Procedure *EXPAND*

There are cases where there is a margin for a sub-circuit extracted by *OMC* to the given size and I/O pin number constraints. In exchange with the increase of I/O pin number, we can increase the size of the sub-circuit. In our algorithm, procedure *EXPAND* does this. See below for its details.

Procedure 3. (Sub-Circuit Enlargement *EXPAND*)

Input: F, lim_{io}, lim_{size}, set of vertices V_1 in F for enlargement.
Output: enlarged set of vertices V_{exp} in F.

1. $V_{exp} \leftarrow V_1$.
2. $R \leftarrow \emptyset$.
3. create a min-cut graph M from F (see Procedure 2 for the detail).
4. select a flow-block from $V(M)$ that contains a vertex connected by an edge to a vertex of V_{exp}. Let V_m be a set of vertices in F represented by such a flow-block.
5. Let V_m^{in} and V_m^{out} be subsets of V_m where the flow comes in and goes out, respectively (see illustrative Fig. 4. V_m^{in} and V_m^{out} are hatched).
6. create an edge $e_o = (v_m, t)$ of ∞ capacity for each $v_m \in V_m^{out}$. Let E be the set of created edges.
7. for each vertex $v_m \in V_m^{in}$, do the following:
   a. create an edge $e_i = (s, v_m)$ of ∞ capacity.
   b. find an optimal min-cut $[V_1', V_2']$ by procedure *OMC* (See illustrative Figs. 5 and 6. The created edges are labeled by '@').
   c. if $size(V_1') \leq lim_{size}$ and $io(V_1') \leq lim_{io}$ and $BETTER(V_1', V_{exp}) = True$ then
      i. $V_{exp} \leftarrow V_1'$.
      ii. $R \leftarrow E \cup \{e_i\}$.
   d. remove e_i.
8. remove the edges stored in E.
9. create an edge $e_i = (s, v_m)$ of ∞ capacity for each $v_m \in V_m^{in}$. Let E be the set of created edges.
10. for each vertex $v_m \in V_m^{out}$, do the following:
    a. create an edge $e_o = (v_m, t)$ of ∞ capacity.
    b. find an optimal min-cut $[V_1', V_2']$ by procedure *OMC*.
    c. if $size(V_1') \leq lim_{size}$ and $io(V_1') \leq lim_{io}$ and $BETTER(V_1', V_{exp}) = True$ then
       i. $V_{exp} \leftarrow V_1'$.
       ii. $R \leftarrow E \cup \{e_o\}$.
    d. remove e_o.
11. remove the edges stored in E.
12. if $R \neq \emptyset$ then
    a. retrieve the edges recorded in R and go to step2.
13. else
    a. return V_{exp}. □

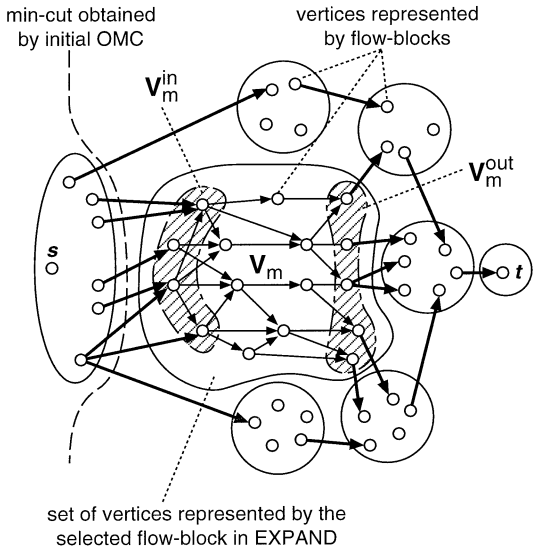


Fig. 4 Before applying *EXPAND*. Virtual vertices and bridge edges are abbreviated as single edges.

Procedure *EXPAND* searches for a new cut in F by fixing some of the vertices in V_m to either s or t. The vertices to be fixed to s and t are chosen from V_m^{in} and V_m^{out}, respectively.

See Fig. 4 for an example where the first min-cut is obtained, and V_m is selected for further partitioning. Figure 5 is an example where v_m^{in} is fixed to the s-side of the cut, and the next set of min-cuts to be browsed is obtained. Those min-cuts, again, will be browsed by procedure *OMC* for finding a desirable one. See V_m is partitioned into smaller sets of vertices. The capacity of any new min-cut is larger than the initial one (the number of I/O pins will increase), but the size of the sub-circuit to be extracted will be larger. Figure 6 is an example where some vertices are fixed to s and t sides to enlarge the sub-circuit to be extracted.

In Liu-Wong's sub-circuit enlargement approach, they collapse to s/t a randomly selected vertex incident to the t-side/s-side of the cut, respectively, to manage the size of the sub-circuit. After a random selection of a vertex to collapse to s or t, the side of the min-cut where it exists, either s or t, will be fixed without evaluating

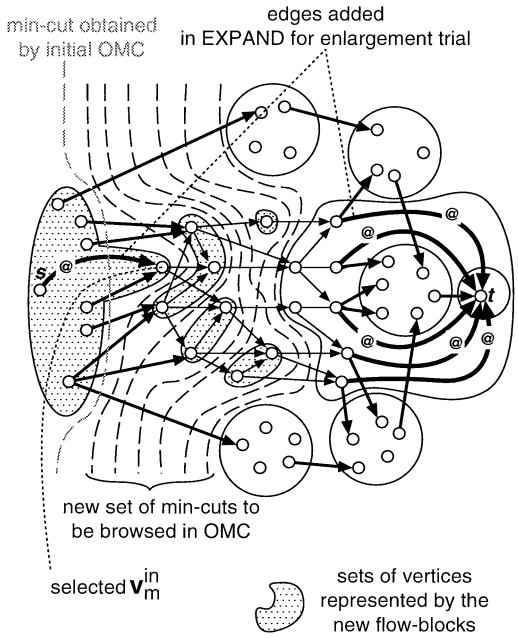


Fig. 5 *EXPAND* being applied to try sub-circuit size enlargement. $v_m \in V_m^{in}$ and all vertices in V_m^{out} are fixed to s/t by ∞-capacity edges to allow excess flow. Sets of vertices represented by the new set of flow-blocks and min-cuts to be browsed are displayed. These new min-cuts will also be exhaustively browsed using *OMC*. Other combinations of vertices will be evaluated after this.

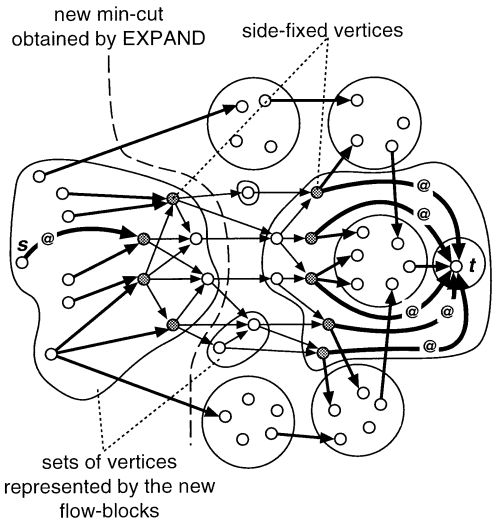


Fig. 6 A new sub-circuit obtained by *EXPAND* as an evaluation result of various combinations of vertices. Some of the vertices in V_m are merged with the sub-circuit extracted by the min-cut obtained by the initial *OMC*.

other vertices for selection. This creates a problem that other selection of vertices will not be evaluated during the entire process, and may miss better results. Furthermore, trials in enlargement of sub-circuits involve in min-cut searches, and Liu-Wong's approach uses their *DMC* for it. This also enhances the possibility to miss better partitioning results.

In our proposing algorithm, we first try sub-circuit enlargement with several vertices incident to the obtained min-cut, and then pick the one that maximizes the size of the sub-circuit to fix to either s or t. Additionally, we again use our exact procedure *OMC* to browse and search among the min-cuts for an optimal one. Therefore, our algorithm has more chances in obtaining better partitioning results.

3.4 Procedure *BETTER*

Another difference between Liu-Wong's and our approach is in how a sub-circuit to be extracted is managed. In our approach, we manage the sub-circuit to be extracted by procedure *BETTER*.

This procedure outputs *True* or *False* according to the I/O pin constraint being satisfied or not for use in the previous procedure *EXPAND* to decide whether we should enlarge the extraction portion when the capacity constraint is satisfied.

There are two types of I/O pins in an extracted sub-circuit, one from the original circuit and one newly created by cutting the nets. We will refer to them as primary I/O pins and secondary I/O pins, respectively. Extracting a sub-circuit with many secondary I/O pins increases the I/O pins of the residual circuit, and this usually results in the increase of the total number of sub-circuits.

Since our aim is to reduce the total number of sub-circuits, not only trying to maximize the size of the sub-circuit but also trying to minimize the the increase of the I/O pins of the residual circuit is important. Its minimization can be achieved by encouraging the extraction algorithm to extract a sub-circuit with many primary I/O pins.

See below for its details. Function $primIO(V_i)$ returns the number of primary I/O pins of the sub-circuit defined by a set of vertices V_i.

Procedure 4. (Sub-Circuit Suitability Check *BETTER*)

Input: sets of vertices V_1 and P for evaluation (V_1 to be compared with P).
Output: Boolean value *True* or *False*.

1. if $P = \emptyset$ then
   a. return *True*.
2. if $size(V_1) > size(P)$ or
   ($size(V_1) = size(P)$ and
   $primIO(V_1)/io(V_1) > primIO(P)/io(P)$) then
   a. return *True*.
3. else
   a. return *False*. □

Note that the sub-circuits in C that correspond to V_1 and P do not violate the given I/O pin number or size constraints thus their sizes and I/O pin numbers are not evaluated in the procedure.

4. Computational Complexity of Algorithm *PART*

Let us estimate the worst case computational complexity of our algorithm, both in terms of time and space, along algorithm *PART*.

Steps 2, 4, 7.b, 7.c, 11 and 12 need constant times. Steps 1 and 3, the Yang-Wong transformation, needs $O(|V(H)| + |E(H)|)$. Step 6, algorithm *OMC*, needs $O(|E(F)||E(H)| + 2^{|V(H)|})$. Step 7.a, algorithm *EXPAND*, needs $O(|E(F)||E(H)||V(H)|+|V(H)|2^{|V(H)|})$. Step 5, algorithm *PICK*, needs $O(|E(F)||E(H)|)$. Step 10, deleting vertices and edges from H, needs $O(|V(H)|+|E(H)|)$. Above steps are repeated $|V(H)|\times |V(H)|$ times in worst case (steps 7.c and 11). Note that the computational time complexity of a single max-flow computation in our case is $O(|E(F)||E(H)|)$. This is because, by Ford-Fulkerson method, a max-flow computation needs $O(|E| \times f)$ time where f denotes the amount of the flow from s to t [3]. However, it is clear that f never exceeds $|E(H)|$ in our case. In the Yang-Wong transformation, all vertices in H are inherited by F, and each of the hyper-edges in H is transformed to a pair of vertices, a directed edge of capacity 1, and some pairs of ∞ capacity directed edges in F. We can replace $|V(F)|$ and $|E(F)|$ by $|V(H)| + 2|E(H)|$ and $(1 + 2c)|E(H)|$, respectively. c is a constant that denotes the average number of end vertices on a single hyper-edge. Therefore, the entire computational time complexity of algorithm *PART* will be $O(|V(H)||E(H)|^3 + |V(H)||E(H)|2^{|V(H)|})$.

On the other hand, in [11], procedure *DMC* needs $O(|E(H)|^2)$, and their way of extraction portion enlargement needs $O(|V(H)||E(H)|^2)$, both under an assumption that the same max-flow computation algorithm and implementation of the Yang-Wong transformation are used. Then, these operations are applied for 10 random pairs of s and t to determine a single sub-circuit. In worst case, the number of sub-circuits will be $|V(H)|$. Therefore, the entire computational time complexity of algorithm in [11] will be $O(|V(H)|^2|E(H)|^2)$

The exponential term in our case shows the computational time complexity of exhaustively browsing the directed cuts in M. However, in real circuits, this term is empirically proven to be small [14]. Thus the main difference between Liu-Wong's and our approach is $|V(H)|$ that indicates the difference in the way s and t are selected.

The space required for both ways is proportional to the size of F, i.e., $O(|V(H)| + |E(H)|)$.

5. Experimental Results

We implemented our algorithm in C, and tested on a 400MHz Pentium2 PC with 64MB of memory, under FreeBSD. The test cases are chosen from the real industrial circuits and MCNC [16] Partition93 benchmark data set (Xilinx 2000 series version). See Tables 1 and 2 for their features and the variety of library cells used in the industrial cases.

The test cases are categorized as follows.

A,B,H: 32-bit ALU, Multiplier and Adder used in industrial microprocessor design.
C,D,E: Controller unit used in industrial Multimedia video decoder design.
F: Industrial 32-bit RISC embedded microprocessor.
G: Quantizer unit used in industrial Multimedia video decoder design.

Others, for instance, c7552, are MCNC benchmark test cases.

The experiment took place to observe the number of sub-circuits and time to obtain them. We also implemented the Liu-Wong's algorithm for comparison. For the industrial data set, the constraints were of size 10000 and I/O pin number of 50. These constraints were decided based on our experiences in successful utilization of 5 K gate FPGA. For the MCNC benchmarks, the constraints were of size 64 and I/O pin number of 58. These constraints were decided from the features of Xilinx 2064 device [15]. See Table 3 for the results. It describes the number of sub-circuits (# s.c.), and time (time[s]) to partition the test case circuits, by both ours and Liu-Wong's approaches. There are results of

Table 1 Summary of the test cases. Case A through H are industrial test cases, and the others are MCNC benchmarks.

Case/Name	Size	# Nets	# I/O
A	121400	989	80
B	211476	1704	125
C	38610	358	36
D	44648	426	77
E	138731	1237	150
F	66970	552	30
G	103914	979	234
H	14975	209	97
c7552	610	1056	267
c6288	833	1456	54
c3540	373	567	72
c5315	145	209	97
s1238	201	301	30

Table 2 Variety of the library cells used in the industrial cases: The largest and the smallest.

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

Table 3 Comparison of our (five cases) and Liu-Wong's algorithms in terms of the number of sub-circuits (# s.c.) and computation time to obtain them (time [s]). The size and I/O pin number constraints for the test cases are 10000 and 50 for industrial cases, and 64 and 58 for MCNC benchmark cases, respectively.

Case/ Name	Ours										Liu-Wong	
	PROPOSED		*NOEXPAND*		*FORCEDMC*		*NOBETTER*		*ST10*			
	# s.c.	time [s]	# s.c.	time [s]	# s.c.	time [s]	# s.c.	time [s]	# s.c.	time [s]	# s.c.	time [s]
A	23	747.1	26	744.8	28	829.4	27	743.7	26	743.7	24	303.1
B	35	2376.7	35	2380.9	38	2572.5	35	2381.6	35	2381.6	37	1728.3
C	5	34.0	6	33.9	7	48.8	6	33.7	7	24.8	6	26.6
D	7	38.5	9	38.4	10	56.8	8	42.5	10	25.8	7	34.0
E	23	1034.5	24	1032.1	24	628.0	24	504.0	28	67.9	30	863.1
F	14	497.1	15	496.2	15	484.4	14	498.9	18	496.3	15	170.0
G	16	300.3	16	251.6	16	336.1	16	251.4	19	38.4	17	110.0
H	2	1.7	2	1.7	2	1.9	2	1.8	4	1.8	3	5.5
c7552	15	258.5	17	67.3	17	299.9	15	235.0	25	60.3	19	86.2
c6288	16	147.1	17	76.8	18	148.6	17	140.9	20	95.5	18	145.0
c3540	11	202.5	12	200.8	14	226.5	12	201.9	14	244.6	12	339.1
c5315	14	134.9	14	134.7	14	134.7	14	134.7	17	77.5	16	69.9
s1238	7	15.9	7	15.9	10	15.7	10	15.2	8	123.2	8	157.8
average	14.5	445.3	15.4	421.2	16.4	444.9	15.4	398.9	17.8	327.8	16.3	310.7

five cases, *PROPOSED*, *NOEXPAND*, *FORCEDMC*, *NOBETTER* and *ST10*, shown for our approach. The following describes the details.

PROPOSED: using the proposed procedures (*PICK*, *OMC*, *BETTER* and *EXPAND*).
NOEXPAND: procedure *EXPAND* replaced by a sub-circuit size enlargement algorithm described in [11].
FORCEDMC: procedure *OMC* replaced by procedure *DMC* in [11].
NOBETTER: procedure *BETTER* modified to size-only evaluation as in [11] (i.e. the bigger, the better).
ST10: procedure *PICK* replaced by choosing 10 random pairs of s and t as in [11].

In most of the cases, our algorithm resulted in less sub-circuits (2 less sub-circuits compared to Liu-Wong's, in average). We observed that the proposed procedures (*OMC*, *EXPAND*, *BETTER* and *PICK*) are effective in reducing the number of sub-circuits. We can also study from the results of *FORCEDMC* and *ST10* that displaying the min-cuts, browsing the min-cuts, and choosing the source s and sink t should be done with care.

From this, we can see that our improvements in:

- finding a min-cut (procedure *OMC*),
- sub-circuit size enlargements by evaluating various sets of vertices combinations,
- s and t determination by a heuristic,

and

- an attempt to try to keep the number of I/O pins of the circuit as small as possible

contribute in reducing the overall sub-circuit numbers.

The next issue is its computation cost. In almost all cases, Liu-Wong's approach is advantageous. Their approach is about 30% faster than ours. This is mainly because our algorithm evaluates various combinations of vertices for sub-circuit size enlargement, on the other hand, Liu-Wong's approach evaluate only one combination that has been determined greedily. In case s1238, our algorithm worked faster against our expectation. This is because the variety of the combination of vertices in procedure *EXPAND* was small, and evaluating all of these required shorter time than trying the extraction 10 times with different s-t pairs.

6. Conclusion

We successfully improved the network-flow based multiway circuit partitioning algorithm by:

- extraction of an exactly size-optimal sub-circuit using an exact min-cut graph,
- enlargement of sub-circuit size by evaluating various sets of vertices combinations,

and

- attempt in keeping the number of I/O pins of the circuit as small as possible by encouraging the extraction algorithm to choose an extraction portion with many number of primary I/O pins.

Applications of our implementation to various industrial cases and benchmark circuits showed our advantage over Liu-Wong's in the number of sub-circuits. In average, our approach resulted in 2 sub-circuits less than that of Liu-Wong's approach using several industrial cases and MCNC benchmarks fairly large in their sizes.

The only problem is the computation cost. Partitioning test cases with hundreds of vertices require minutes. Improvements such as reducing the computation costs, taking into account the delays, and adopting replication techniques to further reduce the number of

sub-circuits, are included in our future work.

Acknowledgment

The authors would like to express their thanks to Prof. Shigetoshi Nakatake (University of Kita-Kyushu), Mr. Hiromasa Takahashi (Fujitsu Laboratories LTD.), and Dr. Kaoru Kawamura (Fujitsu Laboratories LTD.) for their valuable advice and support.

This work is a part of CAD21 project at Tokyo Institute of Technology.

References

[1] T.C. Hu, Integer Programming and Network Flows, Addison-Wesley Publishing Company, Inc., 1970.
[2] C.M. Fiduccia and R.M. Mattheyses, "A linear time heuristic for improving network partitions," Proc. ACM/IEEE DAC, pp.175–181, 1982.
[3] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, M.I.T Press 1990.
[4] 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.
[5] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall International Inc., 1993.
[6] A. Dolan and J. Aldous, Networks and Algorithms: An Introductory approach, John Wiley & Sons, 1993.
[7] N.S. Woo and J. Kim, "An efficient method of partitioning circuits for multiple-FPGA implementations," Proc. DAC, pp.202–207, 1993.
[8] H. Yang and D.F. Wong, "Efficient network flow based mincut balanced partitioning," Proc. ACM/IEEE ICCAD, pp.50–55, 1994.
[9] C.J. Alpert and A.B. Kahng, "Recent directions in netlist partitioning: A survey," Integration, the VLSI Journal, no.19, pp.1–81, 1995.
[10] 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.
[11] 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.
[12] K.R. Azegami, A. Takahashi, and Y. Kajitani, "Maxflow based method for enumerating mincut edges of graph modeled logic circuit," IEICE Technical Report, VLD98–448, Dec. 1998.
[13] 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.
[14] K.R. Azegami, A. Takahashi, and Y. Kajitani, "An efficient algorithm to extract an optimal sub-circuit by the minimum cut," IEICE Trans. Fundamentals, vol.E84-A, no.5, pp.1301–1308, May 2001.
[15] *http://www.xilinx.com*, Xilinx home page
[16] *http://www.cbl.ncsu.edu/benchmarks/*, The Benchmark Archives at CBL

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

Masato Inagi received his B.E. from the Tokyo Institute of Technology in electronic engineering in 2000. He is currently continuing his researches for the M.E. degree in the Department of Communications and Integrated Systems, Graduate School of Science and Engineering at the Tokyo Institute of Technology. His research interests include application of graph theory in VLSI circuit design.

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. He is currently with Department of Communications and Integrated Systems, Graduate School of Science and Engineering. His research interests are in VLSI layout design and combinational algorithms. He is a member of the Information Processing Society of Japan 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. He moved to the University of Kita-Kyushu in 2001. His current research interests are in combinatorial algorithm applied to VLSI layout design. He was awarded IEEE Fellowship in 1992.