IPSJ Transactions on System LSI Design Methodology Vol. 2 200–211 (Aug. 2009)

Regular Paper

Area Recovery under Depth Constraint for Technology Mapping for LUT-based FPGAs [⋆1]

Taiga Takata[†1] and Yusuke Matsunaga[†2]

This paper presents Cut Resubstitution; a heuristic algorithm for post-processing of technology mapping for LUT-based FPGAs to minimize area under depth constraint. The concept of Cut Resubstitution is iterating local transformation of an LUT network with considering actual area reduction without using Boolean matching. Cut Resubstitution iterates the following process. At first, Cut Resubstitution substitutes several LUTs in current network in such a way that another LUT is to be redundant. Then Cut Resubstitution eliminates the redundant LUT from network. Experimental results show that a simple depth-minimum mapper followed by Cut Resubstitution generates network whose area is $7\%, 7\%, 10\%$ smaller than that generated by DAOmap for maximum number of inputs of LUT $4, 5, 6$ on average. Our method is similar or slightly faster than DAOmap.

1. Introduction

Recently, designs using LUT (LookUp-Table) based FPGAs (Field Programmable Gate Array) are becoming popular. LUT-based FPGAs require common photomasks for multiple designs, while ASICs (Application Specific Integrated Circuits) require specific photomasks for individual design. Once an LUT-based FPGA is reconfigured for a design, it is available to use immediately. For these reasons, LUT-based FPGAs are often used for prototypes or manufactures required to be developed faster. Main drawbacks of LUT-based FPGAs are performance and power consumption. Thus, technology mapping for LUT-based FPGAs is required to generate high-quality network in short run-time.

Technology mapping for LUT-based FPGAs is a process to convert a given Boolean network into a functionally equivalent network comprised of K-input LUTs [⋆2]. Technology mapping to generate an area-minimal LUT network whose depth is minimum is well studied [1)–6),9),11),13)]. Area of an LUT network means the number of LUTs in the LUT network. Depth means the length of the longest path [⋆3]. Because a problem for area-minimization has been shown to be NP-hard [10)], predicting accurately which LUTs are the best for minimizing area in practical time is difficult. Thus, heuristics are likely to be necessary to generate LUT network whose depth and area are minimum. Recent methods take a two-step approach. At first, they generate a depth-minimum LUT network [⋆4] with considering area cost of each K-feasible cone [⋆5] in Boolean network. Area cost is a heuristic metric to indicate how good a K-feasible cone is for area. Then, a post-processing recovers area of LUT network with keeping the depth. Some methods [2),9),13)] recover area by iterating global transformation of LUT network. At first, they modify area cost for each K-feasible cone with using the structure of current LUT network. Then, they generate new depth-minimum LUT network with using the area cost. Because the area cost does not have direct relation with actual area, they can fail to find good LUT network. Other methods [6),8),11)] recover area by iterating local transformation of LUT network. They iteratively extract a large cone in LUT network, and remap it with fewer LUTs with using Boolean matching. They are likely to be not suitable for large designs because methods using Boolean matching tend to consume significantly long run-time [⋆6].

This paper presents a heuristic algorithm for post-processing of technology mapping; Cut Resubstitution. The concept of Cut Resubstitution is iterating local transformation of LUT network without using Boolean matching. Cut Resubstitution generates a local optimum solution because it does not consider

†1 Graduate School of Information Science and Electrical Engineering, Kyushu University
†2 Faculty of Information Science and Electrical Engineering, Kyushu University
⋆1 This work is based on the presentation in the 13th Asia South Pacific Design Automation Conference, Korea, January 2008 [12)].

⋆2 In this paper, K-input LUT is denoted by LUT, and the network comprised of K-input LUTs is denoted by LUT network.
⋆3 Most of existing literatures on technology mapping use depth to refer to delay.
⋆4 The problem for depth-minimization can be solved optimally in polynomial time using a dynamic programming [3),7)].
⋆5 A K-feasible cone is a cone with K-input. Technology mapping is often treated as a problem to cover given Boolean network with K-feasible cones.
⋆6 For example, the most recent method based on Boolean matching [6)] consumes 69 minutes of run-time for a circuit "des" in MCNC benchmarks. For several larger circuits, it is experimentally confirmed to consume several thousands of minutes in Ref. 6).

area cost, but does actual area to be reduced. Cut Resubstitution runs fast because it uses only structures of Boolean network and LUT network without using Boolean matching. For details, Cut Resubstitution iterates the following process. At first, Cut Resubstitution extracts a LUT l and its fanout LUTs. Cut Resubstitution substitutes the fanout LUTs with the same number of LUTs in such a way that the LUT l is to be redundant. This substitution is executed with keeping the depth of LUT network. Then Cut Resubstitution eliminates the redundant LUT l. Experiments to compare Cut Resubstitution and one of the *state of the arts* depth-minimum mapper; DAOmap [2] are performed. LUT networks given for Cut Resubstitution are generated by a simple depth-minimum mapper [⋆1]. Cut Resubstitution has generated LUT network whose area is $7\%, 7\%, 10\%$ smaller on average in the case of $K = 4, 5, 6$ than that generated by DAOmap respectively. The depth of LUT network generated by Cut Resubstitution and that by DAOmap are same. The run-time of the depth-minimum mapper combined with Cut Resubstitution is only a few tens of seconds even for large circuits as ITC'99 benchmarks, which is similar or slightly shorter than that of DAOmap.

The rest of this paper is organized as follows. Section 2 presents some basic definitions and formulation. Cut Resubstitution is presented in Section 3. Section 4 presents overall technology mapping combined with Cut Resubstitution. Section 5 presents experimental results. Section 6 concludes this paper.

2. Preliminaries

The inputs of technology mapping are a DAG which is called **subject graph** and a natural number K. For each node v in subject graph, there is a constraint where the number of inputs is up to K [⋆2]. The natural number K corresponds to the maximum number of inputs of LUTs. The output of technology mapping is a network whose nodes represent K-input LUTs. This network is called **LUT network**.

⋆1 Cut Resubstitution is not applicable to an LUT network generated by DAOmap in a simple way because Cut Resubstitution needs an information which K-feasible cones are used for covering a given Boolean network to generate the LUT network.

⋆2 This constraint guarantees that there is at least one LUT network derived by a subject graph.

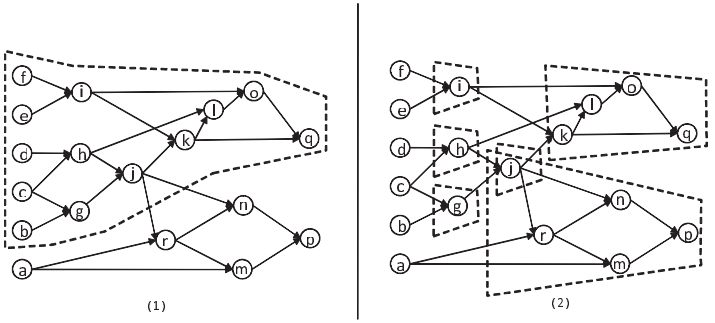

Fig. 1 (1) The transitive fanin graph of q. (2) An example of a realizable set.

A node of a subject graph (V, E) represents a Boolean function and has up to K inputs. If a node $i \in V$ is an input of a node $j \in V$, there is an edge $(i, j) \in E$. The **fanin** of a node v, denoted by $FI(v)$, is the set of immediate predecessors of v. The fanin of v is defined by $FI(v) = \{u \mid \exists(u, v) \in E\}$. The **fanout** of a node v, denoted by $FO(v)$, is the set of immediate successors of v. The fanout of v is defined by $FO(v) = \{w \mid \exists(v, w) \in E\}$. A node v where $FI(v) = \phi$ is called a **primary input**. A node v where $FO(v) = \phi$ is called a **primary output**. PI and PO denote the set of primary inputs and the set of primary outputs respectively. The **transitive fanin** of a node v is the set of all nodes which lie on all paths from any PI to v. More exactly, a transitive fanin of v, denoted by $TFI(v)$, is defined by the following expression.

$$TFI(v) = \{v\} \cup \bigcup_{u \in FI(v)} TFI(u)$$

The **transitive fanin graph** of a node v is the subgraph induced by $TFI(v)$, denoted by $TFIG(v)$. **Figure 1** (1) is the example of a subject graph and a transitive fanin graph. The circles and arrows represent nodes and edges in the subject graph respectively. The dashed line in Fig. 1 (1) illustrates the transitive fanin graph of node q. The **transitive fanout** of a node v is the set of all nodes which lie on all paths from v to any PO. More exactly, a transitive fanout of v, denoted by $TFO(v)$, is defined by the following expression.

$$TFO(v) = \{v\} \cup \bigcup_{u \in FO(v)} TFO(u)$$

A separator s for $TFIG(v)$ is a set of nodes where any path from any primary input to v includes one or more nodes in s. For example, nodes $\{g, h, i, j\}$ in subject graph in Fig. 1 (1) is a separator for $TFIG(q)$. A minimal separator s for $TFIG(v)$ is a separator for $TFIG(v)$ which does not contain any other separator for $TFIG(v)$. For example, nodes $\{g, h, i\}$ in Fig. 1 (1) is a minimal separator for $TFIG(q)$. $\{g, h, i, j\}$ is not minimal separator for $TFIG(q)$ because $\{g, h, i, j\}$ contains the separator $\{g, h, i\}$. A **cut** (s, v) is a pair of node v and set of nodes s where s is a minimal separator for $TFIG(v)$. For a node v, the cut $(v, \{v\})$ is called the *trivial* cut of v. For a cut (v, s), v is called the **root** of the cut, denoted by $RT((v, s))$. For a cut (v, s), s is called the **leaf** of the cut, denoted by $LEAF((v, s))$. For a cut c, $|LEAF(c)|$ is called the *cut-size* of c. A ***K*-feasible cut** c is a cut where $|LEAF(c)|$ is up to K. Because only K-feasible cuts are considered, a K-feasible cut is simply called as a cut in the rest of this paper. $\Phi_K(v)$ denotes the set of all of cuts whose roots are v. For a set of cuts C and a cut $c \in C$, the **cut fanin** $CFI(c, C)$ and **cut fanout** $CFO(c, C)$ are defined by the following equations.

$CFI(c, C) = \{c' | c' \in C, RT(c') \in LEAF(c)\}$

$CFO(c, C) = \{c' | c' \in C, RT(c) \in LEAF(c')\}$

For a feasible cut c, the **feasible cone** $KFC(c)$ is the subgraph induced by the nodes between $RT(c)$ and $LEAF(c)$. A K-feasible cone is exactly defined as the subgraph induced by the set of nodes $V_{interv}(RT(c), LEAF(c))$, where $V_{interv}(v, V)$ is derived by the following equation.

$$V_{interv}(v, V) = \{v\} \cup \bigcup_{u \in FI(v) - V} V_{interv}(u, V)$$

For example, $(q, \{g, h, i\})$ in Fig. 1 (1) is a 3-feasible cut at q. 3-feasible cone of $(q, \{g, h, i\})$ is a subgraph induced by the nodes $\{j, k, l, o, q\}$. For the K-feasible cone $KFC(c)$, the root of c is also called the root of $KFC(c)$, and denoted by $RT(KFC(c))$. In above case, the leaf $LEAF(c)$ is called the inputs of K-feasible cone $KFC(c)$, and denoted by $INPUT(KFC(c))$.

For a K-feasible cone C, $INPUT(C)$ is up to K. Thus, a K-input LUT can implement the Boolean function of any K-feasible cone. If a K-input LUT L implement the Boolean function of a K-feasible cone $KFC(c)$, the output signal of L corresponds to $RT(KFC(c))$, i.e. $RT(c)$, and the input signals of L correspond to $INPUT(KFC(c))$, i.e. $LEAF(c)$. If a set S of cuts meets below three conditions, S is called as the **realizable set**.

- $\forall i \in PO, (\exists c \in S, i = RT(c)) \vee i \in PI$
- $\forall c \in S,\ \forall i \in LEAF(c), (\exists c' \in S, i = RT(c')) \vee i \in PI$
- There is no *trivial* cut in S.

An LUT network can be generated from a realizable set S by below operations.

- For each primary input v in the subject graph, generate a primary input which corresponds to v in the LUT network.
- For each cut c in S, generate an LUT which implements the Boolean function of $KFC(c)$.
- For each $c \in S$, for each $i \in CFI(c, S)$, generate the edge (b, a) where a is the LUT which implements the function of $KFC(c)$ and b is the LUT which implements the function of $KFC(i)$.

The technology mapping problem can be defined as DAG covering problem which is the problem to find a realizable set of cuts.

For a node L in an LUT network, the **level** of L is the length of the longest path from any primary input to L. For a realizable set S and a cut $c \in S$, the **cut level** $LEV(c, S)$ denotes the level of LUT which implement the function of $KFC(c)$. $LEV(c, S)$ can be calculated by the following equation.

$$LEV(c, S) = \begin{cases} \max_{c' \in CFI(c,S)} LEV(c', S) + 1 & (CFI(c, S) \neq \phi) \\ 1 & \text{otherwise} \end{cases}$$

The **depth** of an LUT network is the longest path from any primary input to any primary output. The depth of an LUT network is equal to the largest level in the LUT network. For a realizable set S, the depth $D(S)$ is calculated by the following equation.

$$D(S) = \max_{c \in S}(LEV(c, S))$$

The **area** of an LUT network is the number of nodes in LUT network. For a

realizable set S, the area is calculated by $|S|$. The technology mapping problem to generate an area-minimum LUT network under depth constraint can be defined as the problem to find a realizable set S where $|S|$ is the minimum and $D(S)$ is equal or under the depth constraint d. For a realizable set S and a cut $c \in S$, the **cut required level** $RLV(c, S)$ denotes the level of LUT which implement the function of $KFC(c)$ required to make the depth minimum. $RLV(c, S)$ can be calculated by the following equation.

$$RLV(c,S) = \begin{cases} \min_{c' \in CFO(c,S)} RLV(c', S) - 1 & (CFO(c,S) \neq \phi) \\ d & \text{otherwise} \end{cases} \quad (1)$$

For example, Fig. 1 (2) shows a realizable set where $K = 3$. For each cut c in the realizable set, $KFC(c)$ is illustrated by the dashed trapezoid. $LEV(g, \{c, d\})$, $LEV(h, \{c, d\})$ and $LEV(i, \{e, f\})$ are 1. $LEV(j, \{g, h\})$ and $LEV(p, \{a, g, h\})$ are 2. $LEV(q, \{h, i, j\})$ is 3. Thus, $D(S)$ is 3 in Fig. 1 (2).

3. Cut Resubstitution; a Method for Area Recovery under Detph Constraint

Technology mapping algorithm often consists of the following four phases.

(1) Cut Enumeration
(2) Cut Ranking
(3) Covering
(4) Post-processing

Cut enumeration enumerates cuts in given subject graph. Cut ranking examines how good each cut is for depth and area in a topological order from PI to PO. Covering picks a good cut for each node in a reverse topological order from PO to PI, and generate a depth-minimum LUT network. Post-processing recovers the area with keeping the depth.

Proposed method is a heuristic post-processing which is called as Cut Resubstitution. **Figure 2** (1) is a motivational example for Cut Resubstitution for $K = 3$. Cuts $(w, \{q, v\})$ and $(x, \{s, v, q\})$ are selected as a part of realizable set. If cuts $(w, \{q, y\})$ and $(x, \{s, y\})$ are selected as shown in Fig. 2 (2), cuts $(v, \{r, s\})$ would not be needed. Substituting $(w, \{q, v\})$ and $(x, \{s, v, q\})$ with $(w, \{q, y\})$ and $(x, \{s, y\})$ makes $(v, \{r, s\})$ be redundant without changing the area and the depth. Then, if $(v, \{r, s\})$ is removed from current realizable set, $(r, \{q, u, p\})$ can also be removed because it is also redundant. Therefore, area reduction is 2 in this example.

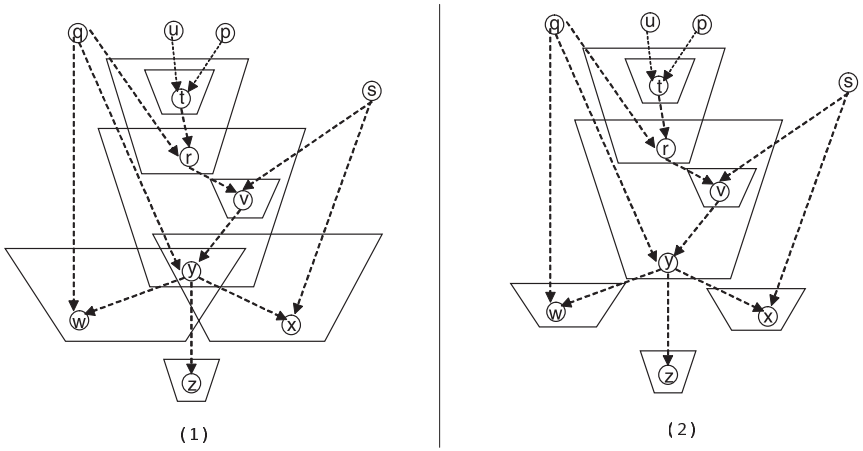

Fig. 2 A motivational example for Cut Resubstituion.

Cut Resubstitution recovers the area of an LUT network under a depth constraint [⋆1]. Inputs of Cut Resubstitution are a subject graph, a depth constraint, all of cuts and a realizable set. Cut Resubstitution generates a realizable set whose size is local optimum based on iterative elimination of cuts. Cut Resubstitution iterates the following process. At first, Cut Resubstitution identifies potentially redundant cuts in current realizable set S. A potentially redundant cut is such a cut c which can be redundant with substituting $CFO(c, S)$. Then, potentially redundant cuts are ranked with using a metric *gain* that denotes how good a cut is to be eliminated for area reduction. Then, for such a potentially redundant cut c_{best} whose gain is the maximum, Cut Resubstitution substitutes $CFO(c_{best}, S)$, and eliminates c_{best}. If the substitution of $CFO(c_{best}, S)$ or elimination of c_{best} make other cuts to be redundant, Cut Resubstitution also

⋆1 The depth of an LUT network must be equal or smaller than the depth constraint to apply Cut Resubstitution.

```
Cut Resubstitution( (V, E), C_all, S_0, d ){
  C_R := RCE(C_all, S_0);
  S_i := S_0;
  while (1) {
      CBEC( (V, E), C_R, S_i, d );
      S := CE( S_i, C_R );
      if (S = S_i) {
          break;
      }
      S_i := S;
  }
  return S;
}
```

Fig. 3 Pseudo code for Cut Resubstitution.

eliminates them.

Figure 3 provides the pseudo code of Cut Resubstitution. In Fig. 3, (V, E) represents a subject graph. C_{all} represents the set of all cuts enumerated in (V, E). S_0 represents given realizable set. C_R represents the sets of cuts which are replaceable for each cut in S_0. A replaceable cut for a cut c means a candidate for substitution of c. S_i and S represent realizable sets of cuts. d represents a depth constraint. RCE enumerates replaceable cuts for each cut in S_0. CBEC examines each cut c in S_i whether c is a potentially redundant cut. Potentially redundant cuts are ranked with using *gain*. For a potentially redundant cut c whose *gain* is the maximum, CE substitutes cuts in $CFO(c, S_i)$ with other cuts in C_R, and eliminates c and other redundant cuts from S_i. Then the size of new realizable set S is at least one smaller than that of S_i. If there is no potentially redundant cut in S_i, the process of Cut Resubstitution finishes.

3.1 Enumeration of Replaceable Cuts

At this phase, the set of all replaceable cuts are enumerated. A cut $c' \in C_{all}$ is a replaceable cut of $c \in S_0$ $(c \neq c')$ if c' meets the following conditions.

$$RT(c) = RT(c') \wedge (\forall i \in LEAF(c'), \exists c'' \in S_0, RT(c'') = i) \quad (2)$$

3.2 Enumeration of Potentially Redundant Cuts and Ranking

At this phase, each cut $c \in S_i$ is examined whether c is a potentially redundant cut. Each potentially redundant cut is ranked by an indicator *gain*. The *gain* of cut c means an estimated number of cuts to be redundant if c is redundant.

For a cut $c \in S_i$, a substitutable set of cuts is defined. A substitutable set C for $c \in S_i$ is a subset of C_R which meets the following three conditions. At first, there is only a cut $c'' \in C$ which meets the condition $RT(c'') = RT(c')$ for each $c' \in CFO(c, S_i)$. At second, $SW(c, C, S_i)$ is a realizable set, where $SW(c, C, S_i)$ denotes a set which is obtained by removing $CFO(c, S_i)$ from S_i, and adding C to S_i. Finally, $D(SW(c, C, S_i))$ must be equal or smaller than d. For example in Fig. 2 (1), $\{(w, \{q, y\}), (x, \{y, s\})\}$ is a substitutable set for $(v, \{r, s\})$.

A cut $c \in S_i$ is a potentially redundant cut if there is a substitutable set for c. Each cut $c \in S_i$ is checked whether there is a substitutable set for c. But examining all combinations to select a repleceable cut for each $c' \in CFO(c, S_i)$ may consume large run-time because the number of all combinations can increase exponentially with $|CFO(c, S_i)|$. For example, if the number of replaceable cuts for each $c \in CFO(c, S_i)$ are the same with s, and if $|CFO(c, S_i)| = t$, then the number of all combinations is s^t. To avoid this problem, a heuristic technique is introduced to check whether there is a substitutable set of $c \in S_i$. At first, each $c' \in CFO(c, S_i)$ is sorted in such a way that roots of cuts are in topological order from PI to PO, and copy S_i to S'_i. Then, each $c' \in CFO(c, S_i)$ is checked whether c' can be substituted with any replaceable cut of c' in the sorted order. If c' and a replaceable cut c'' meet the following condition (3) (4), c' can be substituted with c''.

$$RT(c') = RT(c'') \wedge (\forall j \in LEAF(c''), (j \in PI \vee (j \neq RT(c) \wedge (\exists p \in S'_i, RT(p) = j)))) \quad (3)$$

$$\max_{j \in CFI(c'', S'_i)} LEV(j, S'_i) + 1 \leq RLV(c', S'_i)) \quad (4)$$

If the condition (3) is met, the set of cuts obtained with substituting c' in S'_i with c'' is a realizable set. The condition (4) is for holding the depth constraint. If c' can be substituted by c'', update $S'_i = SW(c', \{c''\}, S'_i)$. If there is a cut which can substitute c' for each $c' \in CFO(c, S_i)$, c is determined as a potentially redundant cut.

An example in **Fig. 4** shows how to check whether there is a substitutable set of c. Current realizable set S'_i is illustrated in the solid trapezoids. $FI(q)$, $FI(t)$, $FO(x)$, $FO(z)$ are omitted in Fig. 4. The pair of numbers (α, β) beside a node v means the pair of cut level and cut required level for cuts whose roots are v. In this example, $CFO(c, S_i)$ is $\{(x, \{q, t, s\}), (y, \{q, t, s\})\}$. At first, $CFO(c, S_i)$

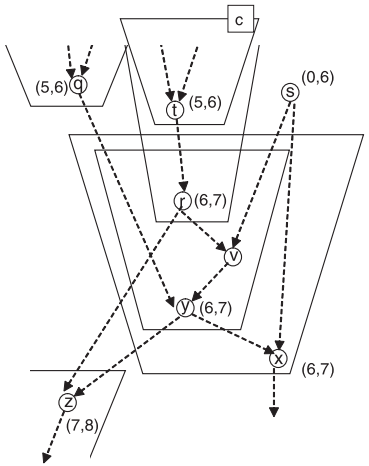

Fig. 4 Checking a substitutable set for a cut c.

are sorted in such a way that roots of cuts are in topological order from PI to PO, and the order $((y,\{q,t,s\}),(x,\{q,t,s\}))$ is obtained. Then, $(y,\{q,t,s\})$ is checked whether $(y,\{q,t,s\})$ can be substituted with any replaceable cut of $(y,\{q,t,s\})$. Let c' and c'' denote $(y,\{q,t,s\})$ and $(y,\{q,r,s\})$. c'' is a reaplaceable cut of c'. $s \in LEAF(c'')$ is in PI. $q \in LEAF(c'') \neq RT(c)$ and there is a cut whose root is q. $r \in LEAF(c'')$ is similar to q. Thus, c' and c'' meets Eq. (3). $\max_{j\in CFI(c'')} LEV(j,S_i') + 1$ is 7. $RLV(c',S_i')$ is 7. Thus, c' and c'' meets Eq. (4). Then, S_i' is updated with substituting c' with c''. Next, $(x,\{q,t,s\})$ is checked in a similar way. The replaceable cuts of $(x,\{q,t,s\})$ are $(x,\{q,r,s\})$ and $(x,\{y,s\})$. Although the pair of $(x,\{q,t,s\})$ and $(x,\{y,s\})$ meets Eq. (3), it does not meet Eq. (4). That is because $LEV((y,r,s),S_i')$ is 7, and cut level for $(x,\{y,s\})$ is 8, while the cut required level for $(x,\{y,s\})$ is 7. On the other hand, the pair of $(x,\{q,t,s\})$ and $(x,\{q,r,s\})$ meets Eqs. (3) and (4). Therefore, $\{(y,\{q,r,s\}),(x,\{q,r,s\})\}$ is a substitutable set of c.

In Eq. (4), $\max_{j\in CFI(c'',S_i')} LEV(j,S_i') + 1$ is the same with the cut level for c'' after substituting $CFO(c,S_i)$. That is because a cut to substitute each $j \in CFO(c,S_i)$ is decided in such an order that roots of cuts are in a topological order from PI to PO, and cut fanin of c'' have already been decided. On the other hand, $RLV(c',S_i')$ in Eq. (4) might be different from cut required level for c'' after substituting $CFO(c,S_i)$, because the cuts to substitute $CFO(c',S_i')$ have not decided yet. Thus, Cut Resubstitution might fail to find some few substitutable sets. But it is guaranteed that Cut Resubstitution does not substitute $CFO(c,S_i)$ with other cuts in such a way that violates the depth constraint. That is because $RLV(c',S_i')$ is the same with cut required level for c'' after substituting $CFO(c,S_i)$ if the following condition for c' is met.

$$(TFO(RT(c')) - \{RT(c')\}) \cap \left(\bigcup_{j\in CFO(c,S_i)} RT(j) \right) = \phi$$

A cut which meets the above condition is called a rearmost cut in $CFO(c,S_i)$. There are one or more rearmost cuts in $CFO(c,S_i)$. For example in Fig. 4, $(x,\{q,t,s\})$ is the rearmost cut in $CFO(c,S_i)$. Because $RLV(c',S_i')$ is accurate cut required level for c'' after substituting a rearmost c' with c'', if a pair of c' and c'' meets Eq. (4), substituting $CFO(c,S_i)$ with the current set of cuts does not violate the depth constraint. Cut Resubstitution avoids to examine all combinations to select a replaceable cut for each $c' \in CFO(c,S_i)$ with employing the above heuristic approach [⋆1].

For each cut $c \in S_i$, *gain* is calculated. The cut whose *gain* is the maximum among potentially redundant cuts is called *best-gain* cut. One of *best-gain* cuts is recorded. The *gain* $GAIN(c,S_i)$ for each $c \in S_i$ is calculated by the following equation in topological order from PI to PO in linear time [⋆2].

$$GAIN(c,S_i) = 1 + \sum_{j\in CFI(c,S_i)} GAIN'(c,j,S_i)$$

$$GAIN'(c,j,S_i) = \begin{cases} GAIN(j,S_i) & (CFO(j,S_i) = \{c\}) \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

For example in Fig. 2 (1), $(v,\{s,r\})$ is a potentially redundant cut. If $(v,\{s,r\})$

⋆1 For all curcuits used in experiments in this paper, there is no difference for the area of an LUT network between using the above heuristic approach and using exact substitutable sets.

⋆2 The computing complexity for *gain* for each cut $c \in S_i$ is $O(|LEAF(c)|)$. The computing complexity for *gain* of all cuts in S_i is $O(|S_i|)$ because $|LEAF(c)| \leq K$.

is eliminated, $(r, \{q, t\})$ can also be eliminated because $(r, \{q, t\})$ becomes redundant. Thus, the *gain* of $(v, \{s, r\})$ is 2.

3.3 Elimination of the Best Potentially Redundant Cut

At this phase, the *best-gain* cut c is eliminated. At first, S_i is copied to S. If there is no potentially redundant cut found in the previous phase, return S to finish Cut Resubstitution. Otherwise, each cut in $CFO(c, S_i)$ is deleted from S. For each $c \in CFO(c, S_i)$, the substitutional cut calculated in the immediately preceding phase is added to S. Then, each cut c' in S where $CFO(c', S) = \phi$ is deleted from S. The set S is returned.

The set of the replaceable cuts C_R is not needed to be updated after elimination of the *best-gain* cut. For a cut $c \notin C_R$, $LEAF(c)$ contains a node v which meets the condition $\forall s \in S_0, v \neq RT(s)$. In above case, if and only if a cut s whose root is v is added to S_0 for each $v \in LEAF(c)$ where $\forall s \in S_0, v \neq RT(s)$, c becomes a replaceable cut for a cut $s' \in S_0 + \{s\}$ where $RT(c) = RT(s')$. No cut c whose root is v where $\forall s \in S_0, v \neq RT(s)$ is added to current realizable set S in any step of Cut Resubstitution. Thus, any cut which is not a replaceable cut for any cut in current realizable set never become a replaceable cut. On the other hand, there is a case where a replaceable cut c for a cut in current realizable set S becomes no-replaceable for any cut in next realizable set S'. But such cut c is not used to substitute any cut in S' because c does not meet Eq. (3). Therefore, the set of the replaceable cuts C_R is not needed to be updated after elimination of the *best-gain* cut.

4. Overall Technology Mapping combined with Cut Resubstituion

Overall Technology mapping combined with Cut Resubstitution consists of cut enumeration, cut ranking, covering and Cut Resubstitution. In this section, cut enumeration, cut ranking and covering are described. The algorithms described in this section are classic. They are introduced to make clear the overall technology mapping and help to understand the experiments following this section.

4.1 Cut Enumeration

All of cuts are enumerated with using an existing technique shown in Ref. 5). The computing complexity for calculating each $\Phi_K(v)$ is proportional to the size of Cartesian products of $\Phi_K(u_1)$ and $\Phi_K(u_2)$, where $FI(v) = \{u_1, u_2\}$. If K is $4, 5$ or 6 [⋆1], $\Phi_K(v)$ for each node v is not so large in most cases. Thus, the run-time of the cut enumeration is practical as far as K is up to 6.

4.2 Cut Ranking

Cut Ranking examines how good each cut is for such objectives as depth and area. Depth-cost and area-cost of each cut are calculated. Depth-cost of a cut c means the minimum cut level in cut levels $LEV(c, S)$ for all of realizable sets S which include c. Depth-cost can be calculated exactly in polynomial time in topological order from PI to PO based on dynamic programming [3]. For a cut c at v, the depth-cost $DC(c)$ is calculated by the following expression (6), where $BD(v)$ denotes the minimum depth-cost of all cuts at v. $BD(v)$ of each PI is 0.

$$DC(c) = \max_{i \in LEAF(c)} BD(i) + 1 \tag{6}$$

$$BD(i) = \begin{cases} 0 & (i \in PI) \\ \min_{j \in \Phi_K(i)} DC(j) & \text{otherwise} \end{cases}$$

Area-cost of a cut c means the minimum size of realizable set S for $TFIG(v)$ where S includes c. Because area minimization for DAG covering problem has been shown to be NP-hard [10], it is likely that there is no efficient way to compute area-cost accurately. The difficulty of area estimation before covering is mainly due to the existence of nodes with multiple fanout and their reconvergence. For example in Fig. 1 (1), it is difficult to make a decision during cut ranking whether the nodes i, h, j, k, r would be better to be duplicated in covering. The node j may be covered by a single LUT. On the other hand, duplicating j in covering, j may be covered by $2 \sim 3$ LUTs. In Fig. 1 (2), j is covered by 2 LUTs.

Several existing algorithms [1),2),5),9),13)] calculate approximate area-cost based on a heuristic technique; *area flow*. The key idea of *area flow* is to distribute the area-cost of each node to its fanout nodes with taking into account the effect of input sharing. For a cut c, area-cost $AC_{af}(c)$ based on *area flow* is calculated by Eq. (7), where $BA(v)$ denotes the minimum area-cost of all cuts of v and $U(c)$ denotes the area contributed by c itself [⋆2].

⋆1 4, 5 and 6 are popular numbers for the maximum number of inputs of LUT in commercial FPGAs.
⋆2 $U(c)$ is usually 1.

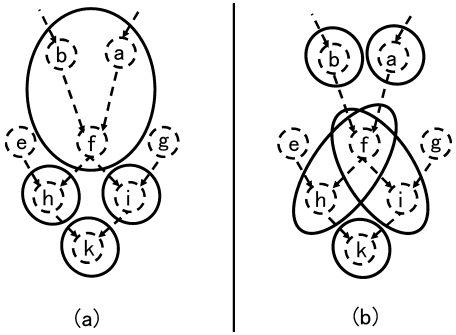

Fig. 5 An example of two realizable sets.

$$AC_{af}(c) = U(c) + \sum_{i \in LEAF(c)} BA(i)/|FO(i)| \quad (7)$$

$$BA(i) = \begin{cases} 0 & (i \in PI) \\ \min_{j \in \Phi_K(i)} AC(j) & \text{otherwise} \end{cases} \quad (8)$$

Equation (7) is based on the heuristics dividing $BA(v)$ by $|FO(v)|$ to avoid summing up $BA(v)$ redundantly.

Area-cost based on *area flow* is accurate if there is no duplication of any node. But if not so, area-cost based on *area flow* can be inaccurate. **Figure 5** shows an example of two realizable sets. Graphs by dashed lines illustrate subject graphs, and circles in solid lines illustrate the K-feasible cones corresponding to a cuts in the realizable set. The predecessors of a and b are omitted, and assume that $BA(i)$ for each $i \in FI(a) \cup FI(b)$ is 0. In Fig. 5 (a), the area for $TFIG(f)$ is 1, and both areas estimated for $TFIG(h)$ and that for $TFIG(i)$ are 3/2. The area estimated for $TFIG(k)$ is 4. Therefore the area based on *area flow* is accurate in Fig. 5 (a). In Fig. 5 (b), both areas estimated for $TFIG(h)$ and that for $TFIG(i)$ are 3. The area estimated for $TFIG(k)$ is 6. Therefore the area estimated based on *area flow* is inaccurate because the correct area is 5 in Fig. 5 (b).

weighted area flow is another heuristic which is implemented in Magus[⋆1]. The key idea of *weighted area flow* is to avoid that an area-cost is redundantly summed along the reconvergent paths with considering the effect of fanouts from nodes in K-feasible cone $KFC(c)$ to other nodes not in $KFC(c)$. Area-cost $AC_{waf}(c)$ of cut c is calculated by Eq. (9).

$$AC_{waf}(c) = 1 + \sum_{i \in LEAF(c)} BA(i) \times NW(i, v) \quad (9)$$

$NW(i, v)$ is an inverse number of times that mean how many the area for $TFIG(i)$ distributed to v is summed up redundantly. The weight $w((v, w))$ for an edge (v, w) is defined as $w((v, w)) = 1/|FO(v)|$. $NW(i, v)$ for a pair of node v and $i \in TFI(v)$ is calculated by summing up each product $w((j, v)) \times NW(i, j)$ for each $j \in FI(v)$ where j is in any path from i to v and $NW(i, i)$ is 1. Area estimated based on *weighted area flow* is accurate in Fig. 5 (b). $w((e, h))$, $w((b, f))$ and $w((a, f))$ are 1, and $w((f, h))$ is 1/2. $NW(b, h)$ and $NW(a, h)$ are both 1/2. $NW(e, h)$ is 1. The area estimated for $TFIG(h)$ is 2. Then, $NW(f, i)$ is 1/2, and $NW(b, i)$, $NW(a, i)$ are both 1/2. $NW(g, i)$ is 1. The area estimated for $TFIG(i)$ is 2. Then, the area estimated for $TFIG(k)$ is 5. Therefore, the area-cost based on *weighted area flow* is correct in Fig. 5 (b).

4.3 Covering

Covering selects the best cuts to cover each node. A best cut is the cut whose area-cost is the minimum where the depth-cost is not up to the cut required level. Covering is executed as follows. At first, the set of cut $S = \phi$ is prepared. At each $o \in PO$, a best cut c is selected and added to S. Next, the best cut of each node in $LEAF(c)$ is selected and added to S[⋆2]. Above process executed iteratively in reverse topological order from PO to PI.

5. Experimental Results

Experiments are performed to compare Cut Resubstitution with one of the *state of the arts* depth-minimum mapper; DAOmap [2]. DAOmap generates a depth-minimum LUT network with using area cost based on *area flow*, and recovers area by iterating global transformation of LUT network with modifying

⋆1 Magus is a Logic Synthesis system developed in Kyushu University

⋆2 The set S may be not realizable set yet. But the cut require level $RLV(c, S)$ can be calculated by the equation (1) because all nodes in $FO(RT(c))$ has been covered and $CFO(c, S)$ has been determinate.

area cost. Cut Resubstitution has been implemented using C++ within the Magus. Subject graphs are generated by decomposing each node of networks on MCNC benchmark set and ITC'99 benchmark set to nodes whose number of inputs are up to 2. The maximum number of inputs of LUT K is assumed to be $4, 5$ or 6. A machine whose CPU is Intel Xeon 3.00 GHz and memory size is 16 GB is used.

The initial LUT networks given for Cut Resubstitution are generated by FMap and WMap those are simple depth-minimum mappers implemented in Magus [⋆1]. FMap and WMap are based on the method described in section 4 to generate LUT network. FMap calculates area-cost based on *area flow*. WMap calculates area-cost based on *weighted area flow*. The depth constraints given for Cut Resubstitution are the depths of LUT networks derived by FMap and WMap. The combination of FMap and Cut Resubstitution is called as FRmap, and the combination of WMap and Cut Resubstitution is called as WRmap. If given subject graph and K are the same, the depths generated by DAOmap and FRmap and WRmap are the same because they are guaranteed to generate a depth-minimum LUT network. Area of LUT network and run-time for each algorithm are evaluated.

Table 1, **Table 3** and **Table 5** show the results on MCNC benchmark set in $K = 4, 5, 6$, respectively. **Table 2**, **Table 4** and **Table 6** show the results on ITC'99 benchmark set in $K = 4, 5, 6$, respectively. The results for small benchmarks are omitted. DAOmap cannot generate an LUT network for "des" in MCNC benchmark set in the case of $K = 6$ in 4 days. Thus, the result for "des" in the case of $K = 6$ is also omitted. "D" in tables means DAOmap. "F" and "FR" means FMap and FRmap, respectively. "W" and "WR" means WMap and WRmap, respectively. "AVG" means average.

Area of LUT network generated by FRmap is $5\%, 7\%, 8\%$ smaller for $K = 4, 5, 6$ respectively than that generated by DAOmap on average. Area of LUT network generated by WRmap is $7\%, 7\%, 10\%$ smaller for $K = 4, 5, 6$ respectively than that generated by DAOmap on average. For all of the circuits but att15 and att16 for $K = 6$, FRmap generated LUT networks with less area than those generated by DAOmap. For all of the circuits but C6288 for $K = 5$, WRmap generated LUT networks with less area than those generated by DAOmap. Cut Resubstitution generates better LUT network than DAOmap, because Cut Resubstitution performs local transformations with considering actual area reduc-

Table 1 The results for MCNC benchmark set in the case of $K = 4$.

	Area							Run Time (sec)		
Name	D	F	FR	FR/D	W	WR	WR/D	D	FR	WR
C5315	580	588	568	98%	590	568	98%	0.2	0.3	0.3
C6288	554	676	537	97%	605	508	92%	0.5	0.9	0.7
C7552	773	791	747	97%	779	731	95%	0.3	0.4	0.4
att10	1190	1163	1161	98%	1230	1161	98%	0.2	0.3	0.3
att15	868	794	791	91%	801	790	91%	0.2	0.2	0.2
att16	560	554	548	98%	574	541	97%	0.1	0.2	0.2
att21	5481	5346	5346	98%	5348	5346	98%	3.0	1.2	1.2
att6	547	520	518	95%	544	528	97%	0.1	0.2	0.2
att8	858	832	832	97%	832	832	97%	0.2	0.2	0.2
des	2807	2623	2615	93%	2702	2649	94%	0.9	0.7	0.7
rot	520	496	489	94%	503	486	93%	0.1	0.1	0.1
vda	547	520	518	95%	544	528	97%	0.1	0.2	0.1
AVG				96%			95%	0.5	0.4	0.4

Table 2 The results for ITC'99 benchmark set in the case of $K = 4$.

	Area							Run Time (sec)		
Name	D	F	FR	FR/D	W	WR	WR/D	D	FR	WR
b14	3097	3033	2945	95%	3248	2394	77%	1.1	1.2	1.2
b14_1	2289	2169	2089	91%	2261	2076	91%	0.7	0.8	0.8
b15	3586	3571	3393	95%	3715	3355	94%	1.0	1.1	1.1
b15_1	4260	4185	4110	96%	4298	4026	95%	1.5	1.3	1.3
b17	11796	11548	11139	94%	12139	10995	93%	3.7	3.7	3.6
b17_1	13032	12676	12447	96%	13075	12228	94%	4.6	4.1	4.1
b20	6255	6049	5874	94%	6321	5776	92%	2.2	2.4	2.4
b20_1	4468	4313	4161	93%	4484	4067	91%	1.5	1.7	1.7
b21	6437	6200	6043	94%	6428	5932	92%	2.3	2.5	2.5
b21_1	4586	4430	4283	93%	4614	4228	92%	1.5	1.7	1.7
b22	9291	8944	8735	94%	9413	8588	92%	3.4	3.6	3.6
b22_1	6802	6560	6362	94%	6859	6229	92%	2.3	2.6	2.6
AVG				94%			91%	2.2	2.2	2.2

⋆1 The experiments do not include applying Cut Resubstitution to LUT networks generated by DAOmap. That is because the realizable set used to generate an LUT network in DAOmap is not known. If functionally equivalent points between each node in an LUT network and nodes in a subject graph are specified, the realizable set can be derived with using the functionally equivalent points and the set of all cuts.

Table 3 The results for MCNC benchmark set in the case of $K = 5$.

	Area							Run Time (sec)		
Name	D	F	FR	FR/D	W	WR	WR/D	D	FR	WR
C5315	438	445	428	98%	448	425	97%	0.5	0.5	0.5
C6288	560	659	540	96%	894	632	113%	2.1	4.0	3.1
C7552	634	652	605	95%	662	608	96%	1.0	1.0	1.2
att10	935	886	884	95%	932	882	94%	0.5	0.4	0.4
att15	651	523	522	80%	533	522	80%	0.4	0.3	0.4
att16	436	437	428	98%	439	404	93%	0.3	0.3	0.3
att21	4391	4207	4207	96%	4204	4202	96%	5.1	1.4	1.4
att6	455	424	424	93%	455	422	93%	0.2	0.2	0.3
att8	619	598	598	97%	598	598	97%	0.2	0.2	0.2
des	2079	1975	1967	95%	2140	1971	95%	1.5	1.3	1.3
rot	385	359	358	93%	379	359	93%	0.2	0.2	0.2
vda	455	424	424	93%	455	422	93%	0.2	0.2	0.2
AVG				94%			95%	1.0	0.8	0.8

Table 4 The results for ITC'99 benchmark set in the case of $K = 5$.

	Area							Run Time (sec)		
Name	D	F	FR	FR/D	W	WR	WR/D	D	FR	WR
b14	2265	2260	2163	95%	2390	2097	93%	3.0	3.2	3.1
b14_1	1749	1721	1653	95%	1824	1633	93%	1.9	2.1	2.1
b15	2944	3021	2788	95%	3175	2693	91%	2.4	3.1	3.0
b15_1	3571	3339	3264	91%	3438	3145	88%	4.4	3.1	3.3
b17	9457	9410	8993	95%	10206	8726	92%	9.9	9.9	9.7
b17_1	10747	10250	10018	93%	10576	9676	90%	12.9	9.8	10.4
b20	4932	4562	4366	89%	4825	4276	87%	6.5	6.8	6.8
b20_1	3437	3275	3138	91%	3486	3087	90%	4.3	4.8	4.9
b21	5069	4788	4560	90%	5067	4499	89%	6.6	7.0	7.0
b21_1	3553	3415	3247	91%	3605	3219	91%	4.3	5.0	5.0
b22	6860	6740	6489	95%	7151	6310	92%	9.2	10.1	10.1
b22_1	5084	5029	4815	95%	5366	4731	93%	6.4	7.5	7.5
AVG				93%			91%	6.0	6.0	6.1

Table 5 The results for MCNC benchmark set in the case of $K = 6$.

	Area							Run Time (sec)		
Name	D	F	FR	FR/D	W	WR	WR/D	D	FR	WR
C5315	315	325	312	99%	320	305	97%	1.9	1.9	2.0
C6288	568	741	523	92%	718	504	89%	16.8	21.4	18.3
C7552	510	521	478	94%	523	449	88%	4.6	3.8	4.6
att10	837	765	759	91%	823	773	92%	1.3	0.8	0.8
att15	490	538	513	105%	486	460	94%	1.2	0.6	0.6
att16	352	358	355	101%	369	339	96%	0.6	0.4	0.4
att21	3900	3683	3679	94%	3679	3678	94%	8.1	2.0	2.0
att6	396	359	353	89%	372	363	92%	0.6	0.5	0.5
att8	525	480	480	91%	480	480	91%	0.3	0.3	0.3
rot	326	310	297	91%	315	295	90%	0.3	0.3	0.4
vda	396	359	353	89%	372	363	92%	0.6	0.4	0.4
AVG				94%			93%	3.3	2.7	2.5

Table 6 The results for ITC'99 benchmark set in the case of $K = 6$.

	Area							Run Time (sec)		
Name	D	F	FR	FR/D	W	WR	WR/D	D	FR	WR
b14	1925	1797	1706	89%	1966	1705	89%	14.3	12.4	12.4
b14_1	1445	1487	1365	94%	1520	1331	92%	9.3	8.5	7.8
b15	2423	2437	2178	90%	2576	2144	88%	11.2	12.8	12.5
b15_1	3173	2724	2590	82%	2838	2569	81%	20.4	11.5	15.6
b17	8127	7937	7304	90%	8457	7127	88%	45.6	39.7	42.0
b17_1	9319	8354	7958	85%	8681	7894	85%	57.8	35.9	48.4
b20	3887	3839	3568	92%	4089	3476	89%	31.6	29.9	29.4
b20_1	2938	2844	2650	90%	3001	2590	88%	22.0	20.6	20.7
b21	3980	3983	3698	93%	4131	3566	90%	33.3	30.8	30.1
b21_1	2979	2908	2701	91%	3078	2656	89%	23.0	21.7	21.7
b22	5795	5670	5316	92%	6003	5113	88%	48.1	45.7	44.3
b22_1	4386	4339	4040	92%	4585	3939	90%	32.9	32.8	31.8
AVG				90%			88%	29.1	25.2	26.4

tion while DAOmap performs only global transformations with considering area cost. For LUT network generated based on area cost, local transformations with considering actual area reduction is likely to work well to reduce area. Cut Resubstitution has been experimentally found to work better in the case of large K. Cut Resubstitution reduced $3\%, 3\%, 6\%$ area of LUT network generated by FMap on average for $K = 4, 5, 6$. Cut Resubstitution reduced $7\%, 9\%, 10\%$ area of LUT network generated by WMap on average for $K = 4, 5, 6$. WMap is considered to fit together with Cut Resubstitution better than FMap in most case. For example in Table 6, WMap generated LUT networks with more area than those generated by DAOmap or FMap for almost all of the circuits. Cut Resubstitution reduced 13% area of LUT network generated by WMap on averages, and the generated LUT network has 12% less area than that generated by DAOmap and 2% less area than that generated by FRmap on average. Furthermore, Cut Resubstitution with WMap works better for large circuit. The networks in ITC'99 benchmark set tend to be larger compared to the networks in MCNC benchmark

set. For LUT networks generated by WMap, the average rate of area reduction of Cut Resubstitution in ITC'99 benchmark set is $6\%, 4\%, 6\%$ larger than that in MCNC benchmark set for $K = 4, 5, 6$, respectively. On the other hand, for FMap, there is no great difference in the average rate of area reduction between MCNC benchmark set and ITC'99 benchmark set. Both the run-time of FRmap and that of WRmap are only a few tens of seconds even for large circuits as ITC'99 benchmarks, which is similar or slightly shorter than that of DAOmap.

6. Conclusions

In this paper Cut Resubstitution which is a heuristic post-processing of technology mapping to minimize area under depth constraint is proposed. Cut Resubstitution generates a local optimum solution because it considers not area cost but actual area to be reduced. Cut Resubstitution runs fast because it uses only structures of Boolean network and LUT network without using Boolean matching. For details, Cut Resubstitution iterates the following process. At first, Cut Resubstitution substitutes several LUTs in current network in such a way that another LUT is to be redundant. Then Cut Resubstitution eliminates the redundant LUT from network. Experimental results show that Cut Resubstitution fits together with a mapper using area-cost based on *weighted area flow* better than that using area-cost based on *area flow*. A simple depth-minimum mapper using *weighted area flow* followed by Cut Resubstitution generated network whose average area is $7\%, 7\%, 10\%$ smaller than DAOmap [2] for $K = 4, 5, 6$ respectively. The depth of LUT network generated by Cut Resubstitution and that by DAOmap are the same if given Boolean network and K are the same. The run-time of Cut Resubstitution is similar or slightly shorter than that of DAOmap. In conclusion, Cut Resubstitution is efficient to reduce area of LUT network generated with considering only area cost.

Our future work is to find initial LUT network which work well with Cut Resubstitution. Examinations of area of LUT networks generated by Cut Resubstitution combined with DAOmap or other algorithms are necessary.

Acknowledgments This work has been partially supported by CREST-DVLSI of JST and Grant-in-Aid for Scientific Research (B) (20300020) of Ministry of Education, Culture, Sports, Science and Technology (MEXT). We are grateful for their support.

References

1) Chatterjee, S., Mishchenko, A. and Brayton, R.: Factor cuts, *Proc. ICCAD '06*, pp.143–150 (2006).
2) Chen, D. and Cong, J.: DAOmap: A depth-optimal area optimization mapping algorithm for FPGA designs, *Proc. ICCAD '04*, pp.752–759 (2004).
3) Cong, J. and Ding, Y.: FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs, *IEEE Trans. CAD*, Vol.13, pp.1–12 (1994).
4) Cong, J. and Ding, Y.: On area/depth trade-off in LUT-based FPGA technology mapping, *IEEE Trans. VLSI Systems*, Vol.2, pp.213–218 (1994).
5) Cong, J., Wu, C. and Ding, Y.: Cut ranking and pruning: Enabling a general and efficient FPGA mapping solution, *Proc. FPGA '99*, pp.29–35, ACM Press (1999).
6) Hu, Y., Shih, V., Majumdar, R. and He, L.: FPGA Area Reduction by Multi-Output Function Based Sequential Resynthesis, *Proc. 45th Design Automation Conference*, pp.24–29 (2008).
7) Kukimoto, Y., Brayton, R.K. and Sawkar, P.: Delay-Optimal Technology Mapping by DAG Covering, *Proc. 35th Design Automation Conference*, pp.348–351 (1998).
8) Ling, A., Singh, D.P. and Brown, S.D.: FPGA technology mapping: A study of optimality, *Proc. 42nd Design Automation Conference*, pp.427–432 (2005).
9) Mishchenko, A., Cho, S., Chatterjee, S. and Brayton, R.: Combinational and sequential mapping with priority cuts, *Proc. ICCAD '07*, pp.354–361 (2007).
10) Rudell, R.: Logic synthesis for VLSI design, PhD Thesis, University of California, Berkeley (1989).
11) Safarpour, S., Veneris, A., Baeckler, G. and Yuan, R.: Efficient SAT-based Boolean Matching for FPGA Technology Mapping, *Proc. 43rd Design Automation Conference*, pp.466–471 (2006).
12) Takata, T. and Matsunaga, Y.: Area recovery under depth constraint by Cut Substitution for technology mapping for LUT-based FPGAs, *Proc. ASP-DAC '08*, pp.144–147 (2008).
13) Teslenko, M. and Dubrova, E.: Hermes: LUT FPGA technology mapping algorithm for area minimization with optimum depth, *Proc. ICCAD '04*, pp.748–751 (2004).

(Received November 17, 2008)
(Revised February 20, 2009)
(Accepted April 13, 2009)
(Released August 14, 2009)

(Recommended by Associate Editor: *Kiyoharu Hamaguchi*)

Taiga Takata received the B.E. and M.E. degrees from Kyushu University, Japan in 2005 and 2007 respectively. He is currently a Ph.D. candidate at the Graduate School of Information Science and Electrical Engineering, Kyushu University. His current research interests include CAD algorithms and design methodology for VLSI. He is a member of IPSJ.

Yusuke Matsunaga received the B.E., M.E. and Ph.D. degrees in Electronics and Communications Engineering from Waseda University, Tokyo, Japan, in 1985, 1987 and 1997, respectively. He joined Fujitsu Laboratories in Kawasaki, Japan, in 1987 and he has been involved in research and development of the CAD for digital systems. From October 1991 to November 1992, he has been a visiting Industrial Fellow at the University of California, Berkeley, in the department of Electrical Engineering and Computer Sciences. In 2001, he joined the faculty at Kyushu University. He is currently an associate professor of Department of Computer Science and Communication Engineering. His research interest includes logic synthesis, formal verification, high-level synthesis and automatic test patterns generation. He is a member of IEICE, IEEE, ACM and IPSJ.