A Power-aware Post-processing under depth constraint for LUT-based FPGA Technology Mapping

Takata, Taiga
Graduate School of Information Science and Electrical Engineering, Kyushu University

Matsunaga, Yusuke
Faculty of Information Science and Electrical Engineering, Kyushu University

A Power-aware Post-processing under depth constraint for LUT-based FPGA Technology Mapping

Taiga Takata

Graduate School of Information Science
and Electrical Engineering
Kyushu University
e-mail: taiga@c.csce.kyushu-u.ac.jp

Yusuke Matsunaga

Faculty of Information Science
and Electrical Engineering
Kyushu University
e-mail: matsunaga@c.csce.kyushu-u.ac.jp

Abstract— It is difficult for LUT-based FPGA technology mapping to generate a power-minimal K-input LUT network with minimum depth at one time because a problem for power-minimization was shown to be NP-hard[5]. A problem for area-minimization is also NP-hard, and area-aware algorithms[7][8][9][10] recover area after generating a depth-minimum network. On the other hand, existing power-aware algorithms[11][12][13][14] finish with generating a depth-minimum network whose power is small. There might be room for power improvement for the generated network. This paper presents a post-processing to minimize power under depth constraint for LUT-based FPGA technology mapping. The proposed algorithm is a power-aware application of Cut Resubstitution[6] which is a post-processing to minimize area under depth constraint. Experimental results show that a simple depth-minimum mapper followed by the proposed algorithm generates networks whose total switching activity is 21% smaller and dynamic power is 10% smaller than that generated by Emap[13] for the input size of LUT $K = 4$. The results also show that total switching activity is associated with dynamic power. The proposed method generated networks whose total switching activity is 24%, 27% smaller than that generated by Emap on average for $K = 5, 6$, respectively. The proposed algorithm runs in practical run-time for all the case.

I. 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.

Power consumed in a circuit consists of dynamic power and static power. Dynamic power is the power consumed with signal transitions. Static power is the power consumed independently of signal transitions. Dynamic power is affected significantly by a result of technology mapping for LUT-based FPGAs, while static power is little-affected. Based on the above discussion, only dynamic power is focused by power-aware technology mapping.

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[1]. Technology mapping algorithm for LUT-based FPGAs to minimize dynamic power and depth[2] has been well studied [11][12][13][14]. Depth means the length of the longest path. Existing power-aware algorithms typically reduce dynamic power by minimizing total switching activity[11][12][14]. Total switching activity means the sum of switching activities of LUTs, and it is denoted by TSA in the rest of this paper.

Because a problem to minimize TSA is known to be NP-hard, heuristics are likely to be necessary to generate LUT network whose depth and TSA are minimum. It is difficult to generate a TSA-minimal LUT network with minimum depth at one time, because it is difficult to predict accurately which LUTs are the best for minimizing TSA in practical time. A problem for area-minimization is also NP-hard, and area-aware algorithms[7][8][9][10] recover area after generating a depth-minimum network. On the other hand, existing power-aware algorithms[11][12][13][14] finish with generating a depth-minimum network whose TSA is small. There might be room for power improvement for the generated network.

This paper presents a post-processing to minimize TSA under depth constraint for LUT-based FPGA technology mapping. The proposed algorithm iteratively eliminates LUTs to minimize TSA with keeping depth constraint. The

[1]In this paper, K-input LUT is denoted by LUT, and the network comprised of K-input LUTs is denoted by LUT network.

[2]Most of existing literatures on technology mapping use depth to refer to delay.

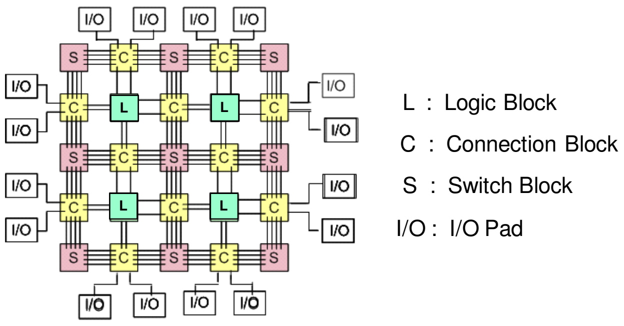

Fig. 1. a schematic diagram of island style architecture

proposed algorithm is a power-aware application of Cut Resubstitution[6] which is a post-processing to minimize area under depth constraint. Cut Resubstitution iteratively eliminates LUTs greedily with using an indicator to denote how big a gain is obtained with eliminating an LUT, however, the indicator is not accurate. The proposed algorithm employs more accurate indicator than that employed in Cut Resubstitution. Experimental results show that a simple depth-minimum mapper followed by the proposed algorithm generates networks whose total switching activity is 21% smaller and dynamic power is 10% smaller than that generated by Emap[13] for the input size of LUT $K = 4$. The results also show that total switching activity is associated with dynamic power. The proposed method generated networks whose total switching activity is 24%, 27% smaller than that generated by Emap on average for $K = 5, 6$, respectively. The run-time of the proposed algorithm is practical for all the case.

The rest of this paper is organized as follows. Section 2 introduce the power model with reviewing the relation of switching activity and dynamic power. Definitions are introduced in Section 3. Section 4 presents the proposed algorithm. Experimental results are shown in Section 5, and this paper is concluded in Section 6.

II. Dynamic Power Estimation for LUT-based FPGA Technology Mapping

A. Dynamic Power consumed in an LUT-based FPGA

A schematic diagram of island-style architecture that is popular for LUT-based FPGAs is shown in figure1. An logic block consists of a number of LUTs. There are also a buffer j for each LUT i where the input of j is the output signal of i. A switch block or a connection block are consists of path transistors and buffers.

Dynamic power P_{dyn} consumed in an LUT-based FPGA is calculated with the following equation.

$$P_{DYN} = P_{SIG} + P_{SHT}$$

P_{SIG} denotes the switching power consumed by charging and discharging capacitance of LUTs, buffers, and inverters. P_{SHT} denotes the power come up with short-circuit current in buffers and inverters. P_{DYN} is often dominated by P_{SIG}. P_{SIG} is calculated by the following equation.

$$P_{SIG} = P_L + P_R$$

P_L denotes the total switching power consumed in logic blocks. The switching power in a logic block is the power consumed with the capacitance of each LUT in the logic block. P_R denotes the total switching power consumed in switch blocks and connection blocks. The switching power in a switch block or a connection block is the power consumed with the capacitance of each buffers in it. P_L and P_R are calculated by the following equations.

$$\begin{aligned} P_L &= \sum_{i \in LBs} (\sum_{j \in LUTs(i)} (P_{lut}(j))) \\ P_R &= \sum_{i \in SBs \cup CBs} (\sum_{j \in BUF(i)} (P_{buf}(j))) \end{aligned}$$

LBs denotes the set of all the logic blocks. $LUTs(i)$ denotes the set of all the LUTs in a logic block i. $P_{lut}(i)$ denotes the switching power consumed in an LUT i. SBs and CBs denotes the set of all the switch blocks and the set of all the connection blocks, respectively. $BUF(i)$ denotes the set of all the buffers in a switch blocks or in a connection blocks. $P_{buf}(i)$ denotes the switching power consumed in a buffer i. $P_{lut}(i)$ and $P_{buf}(i)$ are calculated by the following equations.

$$\begin{aligned} P_{lut}(i) &= \frac{1}{2} \cdot C_{OUT}(j) \cdot V_{dd}^2 \cdot s(i) \cdot f_{clk} \\ &\quad + \frac{1}{2} \sum_{j \in IN_{LUT}(i)} (C_{IN}(i) \cdot V_{dd}^2 \cdot s(j) \cdot f_{clk}) \\ P_{buf}(i) &= \frac{1}{2} \cdot C_{BUF}(i) \cdot V_{dd}^2 \cdot s(IN_{BUF}(i)) \cdot f_{clk} \end{aligned}$$

$C_{IN}(i)$ and $C_{OUT}(i)$ denotes the capacitance of each input and the capacitance of the buffer who is the immediate successor of an LUT i in the logic block, respectively. V_{dd} denotes the supply voltage. $s(i)$ denotes the switching activity of an LUT i or a buffer i. $IN_{LUT}(i)$ denotes the set of LUTs whose output signal incomes to an input of an LUT i. $C_{BUF}(i)$ denotes the sum of the capacities of the input and the capacities of the output of a buffer i. $IN_{buf}(i)$ denotes the LUT whose output signal incomes to the input of a buffer i.

B. Switching Activity and Dynamic Power

Because $C_{IN}(i)$ is often much smaller than $C_{OUT}(i)$, P_{SIG} can be calculate by the following equation.

$$\begin{aligned} P_{SIG} &= \frac{1}{2} V_{dd}^2 \cdot f_{clk} \cdot \sum_{i \in LBs} (\sum_{j \in LUTs(i)} (s(j) \cdot \\ &\quad (C_{OUT}(j) + \sum_{k \in BUF_{FO}(j)} C_{BUF}(k)))) \end{aligned}$$

$BUF_{FO}(i)$ denotes the set of buffers in switch blocks and connection blocks whose input signal depends on the LUT i.

$BUF_{FO}(i)$ is not available during technology mapping, because $BUF_{FO}(i)$ is obtained by the placement and routing after technology mapping. Assuming ideal placement and routing, $|BUF_{FO}(i)|$ is considered as the same for each i. Because $|BUF_{FO}(i)|$ is the same for each LUT i, $\sum_{j \in LUTs(i)}(s(j) \cdot (C_{OUT}(j) + \sum_{k \in BUF_{FO}(j)} C_{BUF}(k)))$ is constant C for each LUT i. Thus, P_{SIG} can be calculated with the following equation.

$$P_{SIG} = \frac{1}{2} V_{dd}^2 \cdot f_{clk} \cdot C \cdot \sum_{i \in LBs} (\sum_{j \in LUTs(i)} (s(j))$$

P_{SIG} can be minimized with minimizing $\sum_{i \in LBs}(\sum_{j \in LUTs(i)}(s(j))$.

III. 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[3]. Switching activity is associated with each node in subject graph. 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**.

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 2 (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 figure 2 (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.

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

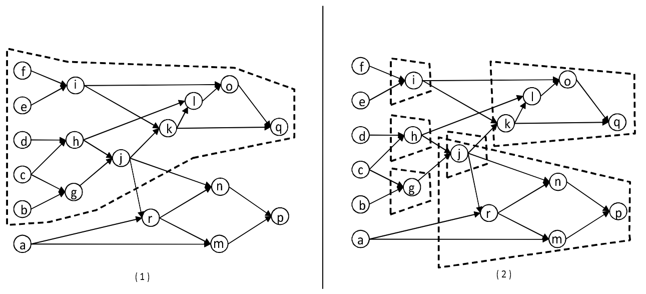

Fig. 2. (1) The transitive fanin graph of q (2) An example of a realizable set

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 figure 2 (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 figure 2 (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.

$$\begin{aligned} CFI(c, C) &= \{c' | c' \in C, RT(c') \in LEAF(c)\} \\ CFO(c, C) &= \{c' | c' \in C, RT(c) \in LEAF(c')\} \end{aligned}$$

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 figure 2 (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 $INPUT(KFC(c))$.

[3]This constraint guarantees that there is at least one LUT network derived by a subject graph.

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.

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

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 **TSA** of an LUT network is the sum of switching activities of LUTs in LUT network. For a node v, the switching activity for v is denoted by $s(v)$. For a realizable set S, the TSA for S is calculated by $TSA(S) = \sum_{l \in S} s(RT(l))$. The technology mapping problem to generate an TSA-minimum LUT network under depth constraint can be defined as the problem to find a realizable set S where $TSA(S)$ is 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.

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

For example, figure 2 (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 figure 2 (2).

IV. The proposed method

The proposed algorithm recovers the TSA of an LUT network under a depth constraint. Inputs of the algorithm are a subject graph, switching activities for each node in the subject graph, a depth constraint, all of cuts and a realizable set. The algorithm generates a realizable set whose TSA is local optimum based on iterative elimination of cuts with depth constraint.

A. A Revision of Cut Resubstitution for Total Switching Activity Minimization

The proposed algorithm is based on Cut Resubstitution[6]. 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. For such a potentially redundant cut c_{best} whose gain is maximum, each cut c in $CFO(c_{best}, S)$ is substituted to other cut c' whose root is the same with $RT(c)$ and whose input does not include c_{best}. Then, Cut Resubstitution 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 eliminates them.

Substituting each cut c in $CFO(c_{best}, S)$ with other cut c' whose root is the same with $RT(c)$ does not affect $TSA(S) = \sum_{l \in S} s(RT(l))$, because $\bigcup_{l \in S} RT(l)$ does not change. Then, TSA is reduced in $\sum_{l \in DEL} s(RT(l))$ where DEL denotes a set of cuts eliminated by Cut Resubstitution. Thus, the main difference of the proposed algorithm and Cut Resubstitution is $gain$. $gain$ for a cut c in Cut Resubstitution denotes the number of cuts expected to be eliminated if c is eliminated. On the other hand, $gain$ for a cut c in the proposed algorithm denotes the sum of $s(c')$ for c' in a set of cuts expected to be eliminated if c is eliminated.

$gain$ $GAIN_n$ obtained by naively replacing area in $gain$ for Cut Resubstitution with switching activity is the following, where S denotes current realizable set.

$$GAIN_n(c, S) = s(c) + \sum_{j \in CFI(c,S)} GAIN'(c, j, S)$$

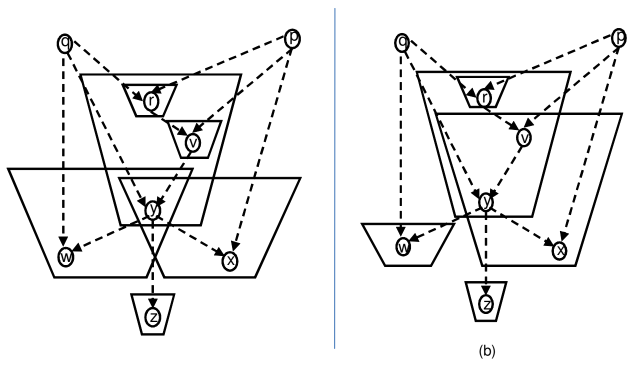

Fig. 3. An example of an elimination of excessive cut c_v (K=3)

```
Cut Substitution( $(V, E), C_{all}, S_0$ ){
    $S_i := S_0$;
    while (1) {
        $C_{rsb}$ := SubstCutEnumeration($C_{all}$, $S_i$);
        BestResubCalculation( $N$, $C_{rsb}$, $S_i$ );
        $S$ := CutElimination( $S_i$, $C_{rsb}$ );
        if ($S == S_i$) {
            break;
        }
        $S_i := S$;
    }
    return $S_i$;
}
```

Fig. 4. Pseudo code for the proposed algorithm

$$GAIN'(c, j, S) = \begin{cases} GAIN_n(j, S) & (CFO(j, S) = \{c\}) \\ 0 & otherwise \end{cases}$$

$GAIN_n$ is calculated by recursively checking whether $|CFO(i, S)|$ for each fanin i of a target cut c is 1 or not, because i can be redundant if c is eliminated.

The actual decrement of TSA might be smaller than the value of $GAIN_n(c, S)$. If $CFO(c', S) = \{c\}$, c' may be eliminated and if c is eliminated. However, if a cut c' in $CFO(c)$ is substituted with other cut c'' whose leaf include $RT(c')$, c' cannot be eliminated. Figure 3 shows an example. Figure 3(a) represents the current realizable set. Cut Resubstitution might substitute cuts in such a way illustrated by figure 3(b), where the cut with root v is the best cut. In this case, cut c whose root is r cannot be eliminated because the cut c' whose root is x and leaf is $\{q, v, p\}$ is substituted with the cut c'' which has r in $LEAF(c'')$. If the cut whose leaf is $\{y, p\}$ is used, c could be eliminated. To handle this problem, the algorithm is modified to decide a best cut c with considering the best combination of cuts to substitute each cut in $CFO(c, S)$.

B. Algorithm Description

The proposed algorithm tries to find the most effective combination of cuts to eliminate each cut $c \in S$ to reduce TSA. The algorithm uses the cuts whose leaf does not include any cut in the fanout free cone of a cut c for substitution to eliminate c. The fanout free cone of a cut c is the maximum set of cuts which have possible to be eliminated if c is eliminated.

Figure 4 provides the pseudo code of the proposed algorithm. In figure 4, (V, E) represents a subject graph. C_{all} represents a set of all the cuts. S, S_0, and S_i represent realizable sets of cuts. C_{rsb} represents a set of cuts. SubstCutEnumeration in figure 4 denotes a step to enumerate candidates for substitution of each cut $c \in S_i$. BestResubCalculation denotes a step to calculate a set of substitutions for $CFO(c, S_i)$ of each cut $c \in S_i$, and the accurate gain of c. CutElimination denotes a step to eliminate the best cut to be eliminated.

Enumeration of candidates for substitution At this step, a set of cuts C_{rsb} is held, where C_{rsb} represents a set of candidates for substitution of each cut $c \in S_i$. A cut $c' \in C_{all}$ which meets the following condition for each cut $c \in S_i$ are enumerated.

$$\forall i \in LEAF(c'), \exists c'' \in C_i, RT(c'') = i, RT(c) = RT(c')$$

Calculation of substitutions and gain At this step, a best set of candidates for substituting $CFO(c, S_i)$ and the accurate gain of c are calculated for each cut $c \in S_i$.

At first, the fanout free cone of c is checked. The fanout free cone for a cut c is the maximum set of cuts those have a possibility to be eliminated if c is eliminated. The fanout free cone for a cut c, denoted by $FFC(c)$, is the set of cut that defined recursively by the following condition.

- c is in $FFC(c)$.
- If $CFO(c', S_i) \subseteq FFC(c)$, c' is also in $FFC(c)$.

The fanout free cone $FFC(c)$ for a cut c is held.

For a cut c, a best candidate cut $c'' \in C_{rsb}$ is calculated for each $c' \in CFO(c, S_i)$. The best candidate cut c'' for c and $c' \in CFO(c, S_i)$ is a cut which meets the following condition.

- $RT(c')$ is the same with $RT(c'')$
- $RT(c)$ is not in $LEAF(c'')$
- $LEV(c'', S_i \cap \{c''\} - \{c'\})$ is not larger than $RLV(c', S_i)$
- $PENALTY(c, c'')$ is the minimum in the set of cuts $RTED(c', C_{all})$

where

$$\begin{aligned} RTED(c', C_{all}) &= \{t | t \neq c', RT(t) = RT(c'), \\ &\quad t \in C_{all}\} \\ PENALTY(c, c'') &= \sum_{i \in FFC\_IN(c'', FFC(c))} s(RT(i)) \\ FFC\_IN(c'', FFC(c)) &= (\bigcup_{i \in LEAF(c'')} FFC(i)) \\ &\quad \bigcap FFC(c) \end{aligned}$$

$RTED(c)$ denotes a set of cuts whose root is the same as that of c and not in S_i. $FFC\_IN(c'', FFC(c))$ denotes a set of cuts which cannot be eliminated with c at the same time if c'' is used to substitute c'. If the best candidate cut is c'' where $FFC\_IN(c'', FFC(c)) \neq \phi$, $FFC(c)$ is updated with $FFC(c) - FFC\_IN(c'', FFC(c))$.

For each cut $c \in S_i$, a set of the best candidates, $C_{best}(c)$, is held. The accurate gain that calculated by $\sum_{i \in FFC(c)} s(RT(i))$ is also held after the set of the best candidates is calculated.

Cut Elimination At this step, a cut $c \in S_i$ with the largest accurate gain is eliminated. Each cut in $CFO(c)$ is deleted from S_i. Each cut in $C_{best}(c)$ is added to S_i. Then each cut in $FFC(c)$ is also deleted from S_i.

V. EXPERIMENTAL RESULTS

The proposed algorithm has been implemented as a program using C++ within Magus Logic Synthesis system in Kyushu University. Experiments are executed to compare the proposed algorithm with one of the state-of-the-art, Emap[13] on a machine whose CPU is Intel Xeon $3.00GHz$ and the memory size is $16GB$. 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 switching activity of each node in a subject graph is calculated by logic simulation. The number of random input vectors used in logic simulation is $60,000$ for each circuit. The maximum number of inputs of LUT K is assumed to be $4, 5$ or 6. In order to estimate actual dynamic power of LUT networks generated by the proposed algorithm and Emap, a gate-level power estimator, PowerPlay Analyzer from Altera's Quartus II software is used. The experimental flow to estimate dynamic power of an LUT network is as following. At first the LUT network in Berkeley logic interchange format (BLIF) is converted to Verilog Quartus Mapping (VQM) format. Several options are set to restrict Quartus II not to modify the LUT network. A target device is assumed as Cyclone III device family. Quartus_map, quartus_fit, quartus_asm are executed for placement and routing. The random input vectors used in logic simulation are converted to Vector Waveform Format (VWF) form. Quartus_sim is executed to calculate switching activity with the VWF files, and obtained a Switching Activity File (SAF). PowerPlay Analyzer with SAF file is executed to obtain dynamic power. Then, dynamic power for the LUT network is obtained. Dynamic power of each LUT network for $K = 5$ and $K = 6$ is not obtained because the maximum number of inputs of LUTs in Cyclone III device family is 4.

Table I and Table II show dynamic power of circuits in MCNC benchmark set and ITC'99 benchmark set, respectively. The results for small circuits are omitted. Dynamic power for some circuits in ITC'99 benchmark set cannot be obtained because those have so many primary inputs and primary outputs and they cannot loaded in Cyclone III. "CutResub" means the proposed algorithm. "Comp" means the ratios of

TABLE I
DYNAMIC POWER OF CIRCUITS IN MCNC BENCHMARK SET IN THE CASE OF $K = 4$

	Emap(mW)	CutResub(mW)	Comp
9symml	1.76	1.68	95%
C1355	5.53	5.40	98%
C1908	7.25	7.00	97%
C2670	8.03	6.66	83%
C3540	9.36	8.99	96%
C432	2.14	1.68	79%
C499	5.53	4.71	85%
C5315	19.40	18.37	95%
C6288	29.36	21.89	75%
C7552	32.96	33.04	100%
C880	2.80	2.63	94%
apex6	5.35	5.71	107%
apex7	2.94	2.56	87%
att12	1.25	1.17	94%
att13	5.03	4.49	89%
att14	3.62	3.12	86%
att15	13.78	11.54	84%
att16	13.45	12.53	93%
att18	1.57	1.30	83%
att20	4.35	3.52	81%
att21	50.74	41.69	82%
att22	9.50	9.33	98%
att23	5.50	4.58	83%
att3	1.35	1.33	99%
att4	1.83	1.87	102%
att6	5.74	5.05	88%
att7	2.21	1.93	87%
att8	9.85	9.06	92%
b9	0.97	0.95	98%
des	65.23	66.09	101%
f51m	2.06	1.64	80%
rot	6.16	6.03	98%
too_large	50.82	42.72	84%
vda	5.74	5.05	88%
z4ml	1.57	1.57	100%
Average			91%

dynamic power of LUT network generated by the proposed algorithm and that generated by Emap. Dynamic power of LUT network generated by the proposed algorithm is 90% compared to that generated by Emap on average. Figure 5 shows a relation between total switching activity and dynamic power. Each dot in Figure 5 represents the total switching activity and dynamic power of a circuit generated by the proposed algorithm or Emap. The vertical axis denotes dynamic power, and the horizontal axis denotes total switching activity. This results show that total switching activity is associated with the actual dynamic power.

Table III shows total switching activity of each LUT network generated by the proposed algorithm and Emap for $K = 4, K = 5, K = 6$ for circuits in ITC'99 benchmark set. The results for MCNC benchmark set are omitted. "Sw" denotes total switching activity, and "Time" denotes run-time of the program. "Comp" denotes rations of total switching activity generated by the proposed algorithm and that generated by Emap. The proposed algorithm generates LUT networks whose total switching activity is 79%, 76%, 73% compared to LUT networks generated by Emap for $K = 4$, $K = 5$, $K = 6$ on average. The proposed algorithm runs in practical run-time for all the case.

TABLE III
THE TOTAL SWITCHING ACTIVITY OF CIRCUITS IN ITC '99 BENCHMARK SET FOR $K = 4$, $K = 5$, $K = 6$

	K=4					K=5					K=6				
	Emap		CutResub			Emap		CutResub			Emap		CutResub		
	Sw	Time	Sw	Time	Comp	Sw	Time	Sw	Time	Comp	Sw	Time	Sw	Time	Comp
b03	26.56	0.01	22.51	0.01	85%	20.80	0.01	16.92	0.01	81%	20.38	0.04	15.23	0.01	75%
b04	91.83	0.17	77.39	0.05	84%	76.92	2.23	64.57	0.17	84%	76.13	59.35	62.43	0.85	82%
b07	68.31	0.00	54.45	0.04	80%	54.92	0.28	43.02	0.09	78%	52.14	3.06	42.81	0.19	82%
b08	23.55	0.04	16.49	0.01	70%	21.20	0.08	16.04	0.02	76%	17.27	1.51	13.18	0.05	76%
b09	21.90	0.02	19.19	0.00	88%	18.18	0.02	15.53	0.01	85%	17.46	0.08	15.21	0.01	87%
b10	27.43	0.02	20.64	0.01	75%	22.45	0.04	16.18	0.01	72%	19.24	0.28	12.65	0.02	66%
b11	88.18	0.18	58.70	0.05	67%	76.95	1.39	56.26	0.09	73%	67.90	22.46	36.02	0.28	53%
b12	125.08	0.18	104.26	0.09	83%	113.08	1.23	94.51	0.13	84%	102.28	16.19	83.68	0.30	82%
b13	43.04	0.02	35.07	0.01	81%	37.40	0.02	28.40	0.01	76%	36.31	0.04	27.22	0.01	75%
b14	1212.96	2.01	981.32	6.51	81%	959.89	24.27	726.34	6.33	76%	851.77	440.60	578.03	18.24	68%
b14_1	862.47	1.32	711.30	2.60	82%	720.79	17.32	553.30	3.56	77%	641.01	332.09	448.31	10.84	70%
b15	1097.55	1.38	686.03	5.10	63%	814.18	18.16	577.74	8.46	71%	746.22	397.29	497.28	20.07	67%
b15_1	1026.99	2.31	639.74	4.27	62%	779.17	50.42	478.54	6.29	61%	713.65	3191.40	433.67	18.64	61%
b17	2660.25	5.38	1823.36	85.48	69%	2031.47	72.75	1442.73	119.46	71%	1793.13	2688.28	1253.10	160.50	70%
b17_1	2726.68	7.34	1767.41	52.17	65%	2096.16	161.40	1299.91	56.48	62%	1850.61	9964.96	1150.59	115.22	62%
b20	2430.42	4.16	2023.26	26.49	83%	1918.71	49.82	1496.43	27.02	78%	1678.51	923.32	1200.93	54.02	72%
b20_1	1749.11	3.14	1488.60	12.13	85%	1436.41	42.90	1117.19	12.50	78%	1267.70	633.99	949.84	31.96	75%
b21	2475.54	4.28	2086.38	29.37	84%	1977.10	53.12	1582.53	36.90	80%	1708.26	983.86	1231.74	54.71	72%
b21_1	1783.82	3.14	1526.98	13.98	86%	1450.20	44.55	1149.82	13.22	79%	1267.08	965.28	957.04	34.53	76%
b22	3601.80	6.04	3006.28	58.87	83%	2782.25	75.35	2194.49	52.70	79%	2481.36	1453.11	1774.23	105.09	72%
b22_1	2656.76	4.70	2248.54	28.33	85%	2122.74	64.66	1690.66	26.90	80%	1883.03	1368.75	1434.00	59.73	76%
Average					78%					76%					72%

TABLE II
DYNAMIC POWER OF THE CIRCUITS IN ITC'99 BENCHMARK SET IN THE CASE OF K=4

	Emap	CutResub	Comp
b03	1.47	1.42	97%
b04	5.97	5.73	96%
b07	5.07	4.30	85%
b08	1.27	0.77	61%
b09	1.47	1.25	85%
b10	1.61	1.80	112%
b11	6.66	5.12	77%
b12	9.41	8.34	89%
b13	2.06	1.98	96%
Average			88%

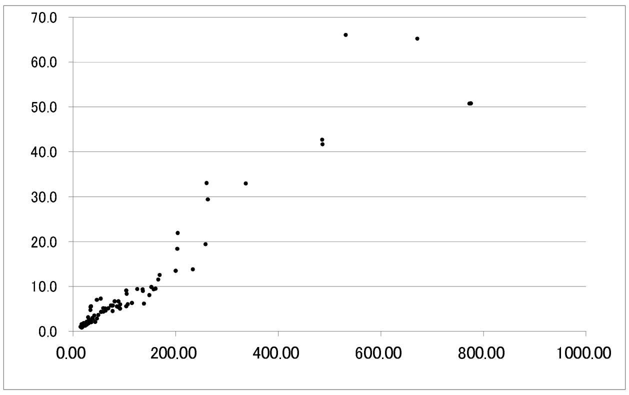

Fig. 5. The relation between switching activity and dynamic power (K=4)

VI. CONCLUSIONS

In this paper, we present a post-processing for power-aware technology mapping for LUT-based FPGAs. The proposed algorithm reduces total switching activity of LUT networks under depth constraint. Compared to a previous state-of-the-art power-aware technology mapping algorithm, Emap, the proposed method is 21% better for total switching activity, and 10% better for dynamic power on average if K=4. The results also show that total switching activity is associated with actual dynamic power. The proposed algorithm is 24%, 27% better for total switching activity compared to Emap on average if K=5, K=6, respectively. The run-time of the proposed algorithm is practical for all the case.

Future works includes a study for a post-processing to minimize dynamic power with considering glitch. Actual switching activity can be larger than that calculated by logic simulation because of the glitch. Dynamic power consumed by glitch can be a significant portion of dynamic power. With modifying the switching activity of each LUT during technology mapping, it might be possible to reduce more dynamic power than a method only use switching activity calculated statically.

VII. 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] Richard L.Rudell, "Logic synthesis for VLSI design," Ph.D.thesis, University of California, Berkeley, 1989.

[2] J.Cong, Y.Ding, "On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping," IEEE Transactions on Very Large Scale Integration Systems, June, 1994.

[3] D.Chen, J.Cong, "DAOmap: A Depth-optimal Area Optimization Mapping Algorithm for FPGA Designs," IEEE International Conference on Computer Aided Design, 2004.

[4] Maxim Teslenko, Elena Dubrova, "Hermes: LUT FPGA Technology Mapping Algorithm for Area Minimization with Optimal Depth," IEEE International Conference on Computer Aided Design, 2004.

[5] Rudell R, "Logic synthesis for VLSI design," PhD Thesis, University of California, Berkeley, 1989.

[6] Taiga Takata, Yusuke Matsunaga, "Area Recovery under Depth Constraint by Cut Substitution for Technology Mapping for LUT-Based FPGAs," Asia and South Pacific Design Automation Conference, Feb, 2008.

[7] D.Chen and J.Cong, "Daomap: A depth-optimal area optimization mapping algorithm for fpga designs," in Proc. ICCAD '04, 2004.

[8] Andrew Ling, Deshanand P. Singh, Stephen D. Brown, "FPGA Technology Mapping: A Study of Optimality," Design Automation Conference, 2005.

[9] Sean Safarpour, Andreas Veneris, Gregg Baeckler, Richard Yuan, "Efficient SAT-based Boolean Matching for FPGA Technology Mapping," Design Automation Conference, 2006.

[10] Yu Hu, Victor Shih, Rupak Majumdar, Lei He, "FPGA Area Reduction by Multi-Output Function Based Sequential Resynthesis," Design Automation Conference, 2008.

[11] Z. H. Wang et al, "Power Minimization in LUT-based FPGA Technology Mapping," Asia and South Pacific Design Automation Conference, 2001.

[12] H. Li, W. Mak, and S. Katkoori, "Efficient LUT-based FPGA Technology Mapping for Power Minimization," Asia and South Pacific Design Automation Conference, 2003.

[13] J. Lamoureux and S. J. E. Wilton, "On the Interaction between Power-Aware CAD Algorithms for FPGAs," International Conference on Computer-Aided Design, 2003.

[14] Lei Cheng, Deming Chen, Martin D. F. Wong, "GlitchMap: An FPGA Technology Mapper for Low Power Considering Glitches," Design Automation Conference, 2007.