# Technology Mapping for FPGAs with Embedded Memory Blocks

Jason Cong and Songjie Xu
Department of Computer Science
University of California, Los Angeles, CA 90024
{cong, sxu}@cs.ucla.edu

## Abstract

Modern field programmable gate arrays (FPGAs) provide embedded memory blocks (EMBs) to be used as on-chip memories. In this paper, we explore the possibility of using EMBs to implement logic functions when they are not used as on-chip memory. We propose a general technology mapping problem for FPGAs with EMBs for area and delay minimization and develop an efficient algorithm based on the concepts of Maximum Fanout Free Cone (MFFC) [3] and Maximum Fanout Free Subgraph (MFFS) [7], named EMB_Pack, which minimizes the area after or before technology mapping by using EMBs while maintaining the circuit delay. We have tested EMB_Pack on MCNC benchmarks on Altera's FLEX10K device family [1]. The experimental results show that compared with the original mapped circuits generated from CutMap [5] without using EMBs, EMB_Pack as postprocessing can further reduce up to 10% of the area on the mapped circuits while maintaining the layout delay by making efficient use of available EMB resources. Compared with CutMap-e without using EMBs, EMB_Pack as pre-mapping processing followed by CutMap-e can reduce 6% of the area while maintaining the circuit optimal delay.

## 1 Introduction

Modern FPGAs provide in a single device both logic array for general logic functions and embedded memory blocks (EMBs) for efficient implementation of on-chip memory and specialized logic functions. Figure 1 shows the device block diagram of Altera's FLEX10K device family, which is the first FPGAs to contain embedded memory blocks. In a single FPGA chip from FLEX10K device family, there are not only a logic array of 4-LUTs, but an embedded memory array with a series of EMBs[1] as well. When implementing memory functions, each EMB provides 2,048 bits, which can be used to create RAM, ROM, FIFO functions, or dual-port RAM. When used as RAM, each EMB can be configured in any of the following sizes: $256 \times 8$, $512 \times 4$, $1,024 \times 2$, or $2,048 \times 1$. Therefore, each EMB can implement any single 11-input 1-output, 10-input 2-output, 9-input 4-output, or 8-input 8-output logic function. The large capacity of EMBs enables designers to implement complex functions in one logic level without the routing delays associated with a subnetwork of LUTs. For example, a single EMB can implement a $4 \times 4$ multiplier with eight inputs and eight outputs. On the other hand, due to its large capacity, an EMB usually has larger delay than an LUT. For example, the delay of an EMB in FLEX10K device family is $3 \sim 4$ times that of a normal LUT. An interesting question is how to make efficient use of these EMBs to minimize circuit area and/or delay during technology mapping. In general, there are two difficulties associated with the technology mapping for FPGAs with EMBs. One difficulty is how to map networks into devices with a large number of $K$-LUTs and limited number of EMBs of much larger capacity and delay for overall circuit area and delay minimization. The other difficulty is due to the configuration flexibility of EMBs, which requires the mapping algorithms to consider extracting multiple-output logic blocks from the network.

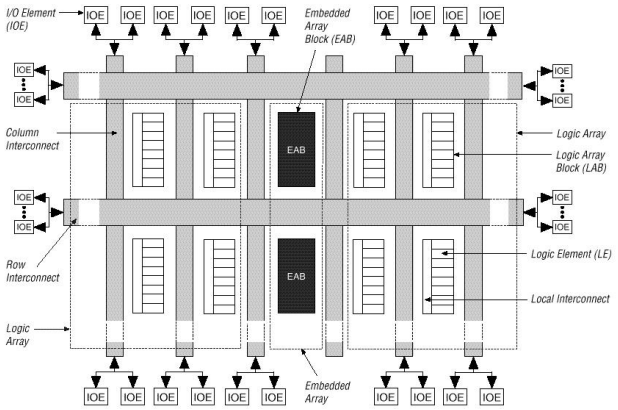


Figure 1: FLEX10K Device Block diagram.

[1]In Altera FLEX10K device family, *LUT* is called *logic element (LE)* and *embedded memory block (EMB)* is called *embedded array block (EAB)*. The *number of EMBs* contained in a single chip from FLEX10K device family varies from 3 to 12 according to the device size.

In the past few years, intensive studies have been done on technology mapping for LUT-based FPGAs. The previous LUT-based FPGA mapping algorithms can be roughly divided into four classes, area minimization, delay minimization, simultaneous area and delay minimization and routability optimization. A comprehensive survey of existing FPGA mapping algorithms is available in [6]. However, none of the existing mapping algorithms are able to deal with FPGAs with EMBs. In [12], an approach for technology mapping into heterogeneous LUT-based FPGAs was presented. Their architecture assumes a mixture of $p$-input LUTs and $s$-input LUTs with a fixed ratio of $r$ in one FPGA chip, which is quite different from FPGAs with a limited number of large EMBs which allow multiple configurations.

In this paper, we propose a general technology mapping problem for FPGAs with EMBs and focus our algorithm on the area minimization while maintaining the circuit delay. Our algorithm, named EMB_Pack, uses the concepts of Maximum Fanout Free Cones (MFFCs) [3] and Maximum Fanout Free Subgraphs (MFFSs) [7] and uses EMBs to minimize the area after or before technology mapping while maintaining the circuit delay. Although the description and the experimental results in this paper are based on Altera FLEX10K device family, our algorithm is general to various kinds of LUT-based FPGAs with embedded memory arrays.

The remainder of this paper is organized as follows. Section 2 gives the problem formulation and preliminaries. Section 3 presents the MFFC and MFFS based area minimization algorithm for FPGAs with embedded memory blocks as post-mapping processing. Section 4 presents the application of EMB_Pack as pre-mapping processing. Section 5 explores the integration of technology mapping with EMB using for area and delay minimization. Experimental results and comparative study are presented in Section 6. Section 7 concludes the paper.

## 2 Problem Formulation and Preliminaries

A Boolean network can be represented as a directed acyclic graph (DAG) where each node represents a logic gate[2], and a directed edge ($i$, $j$) exists if the output of gate $i$ is an input of gate $j$. A primary input (PI) node has no incoming edge and a primary output (PO) node has no outgoing edge. We use $input(v)$ to denote the set of nodes which are fanins of gate $v$, and $output(v)$ to denote the set of nodes which are fanouts of gate $v$. Given a subgraph $H$ of the Boolean network, $input(H)$ denotes the set of *distinct* nodes outside $H$ which supply inputs to the gates in $H$. Node $u$ is the *transitive fanin* of node $v$ if there is a path from $u$ to $v$. Similarly, node $u$ is the *transitive fanout* of node $v$ if there is a path from $v$ to $u$. The *level* of a node $v$ is the length of the longest path from any PI node to $v$. The level of a PI node is zero. The *depth* of a network is the largest node level in the network. A boolean network is *K-bounded* if $|input(v)| \leq K$ for each node $v$ in the network.

For a node $v$ in the network, a *fanin cone*[3] *at* $v$, denoted $C_v$, is a subgraph consisting of $v$ and its predecessors[4] such that any path connecting a node in $C_v$ and $v$ lies entirely in $C_v$. We call $v$ the root of $C_v$. $C_v$ is $K$-$feasible$ if $|input(C_v)| \leq K$. For a node $v$ in the network, a *fanout cone*[5] *at* $v$, denoted $D_v$, is a subgraph consisting of $v$ and its transitive fanouts such that any path connecting $v$ and a node in $D_v$ lies entirely in $D_v$. We call $v$ the root of $D_v$. For a set of nodes $S$ in the network, a *joint fanin cone at* $S$, denoted $C(S)$, is a subgraph consisting of $S$ and all $C_u$ ($u \in S$). We call $S$ the root set of $C(S)$. $C(S)$ is $K$-$feasible$ if $|input(C(S))| \leq K$. A *fanout-free cone (FFC) of* $v$, denoted $FFC_v$, is a fanin cone of $v$ such that for any node $u \neq v$ in $FFC_v$, $output(u) \subseteq FFC_v$.

We assume that an FPGA with EMBs consists of an array of $K$-LUTs and $r$ EMBs which can be configured to implement any function of $\alpha_1$-input $\beta_1$-output, $\alpha_2$-input $\beta_2$-output, ..., or $\alpha_c$-input $\beta_c$-output. The delay ratio of a $K$-LUT *vs.* an EMB is $d_1 : d_2$ ($d_1 < d_2$).

The area of a mapped circuit is measured by the number of $K$-LUTs used in the FPGA. As our goal is to use uncommitted EMBs (not used as on-chip memory) to reduce the number of $K$-LUTs, we will not count the EMBs used into area. The delay of a circuit is measured in terms of both its logic depth and the final circuit delay after placement and routing.

Given a *combinational* circuit described as a Boolean network $N$, the technology mapping problem for FPGAs with EMBs is to transform $N$ to an equivalent logic block boolean network $N'$, where each logic block is either a $K$-LUT or an EMB of $\alpha_1$-input $\beta_1$-output, $\alpha_2$-input $\beta_2$-output, ..., or $\alpha_c$-input $\beta_c$-output, and the number of EMBs in $N'$ is no more than $r$, such that the circuit area and/or delay is minimized.

Instead of solving this general technology mapping problem for FPGAs with EMBs directly, this paper explores three approaches to minimize the circuit area and delay by using EMBs. Section 3 presents an MFFC and MFFS based algorithm, named EMB_Pack, which minimizes the area after technology mapping by using EMBs while maintaining the circuit delay. Section 4 presents the application of EMB_Pack in pre-mapping processing as the second approach. Section 5 looks into the integration of technology mapping with EMB using for area and delay minimization.

## 3 EMB_Pack Algorithm

In this section, we study the following technology mapping problem for FPGAs with EMBs.

**Problem 1** *Given a* **mapped** *combinational K-LUT network N and the FPGA with EMBs whose architecture is*

[2] In the rest of the paper, *gate* and *node* are used interchangeably for Boolean networks.

[3] In this paper *fanin cone* is the same as *predecessor cone* or *transitive fanin cone*.

[4] $u$ is a predecessor of $v$ if there is a directed path from $u$ to $v$.

[5] In this paper *fanout cone* is the same as *transitive fanout cone*.

*described in Section 2, make use of the available EMBs and their various configurations to minimize the area while maintaining the circuit delay.*

The input to Problem 1 is a $K$-LUT netlist constructed by a technology mapper such as FlowMap [4] or CutMap [5]. The postprocessing aims at further minimizing the area by efficiently using available EMB resources, and at the same time maintaining the delay of the original mapped circuit. Figure 2 uses an example to illustrate this procedure, where we can see that after postprocessing a part of the circuit (shadowed area) can be put in one EMB with nine inputs and four outputs if delay is not a concern.

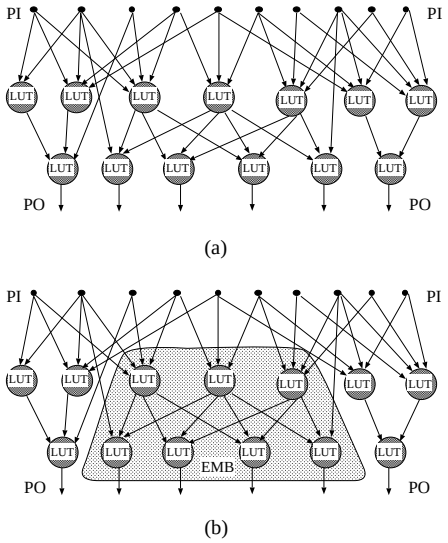


Figure 2: (a) Network after technology mapping ($K = 4$); (b) after postprocessing, part of the network (shadowed area) can be put in one 9-input 4-output EMB.

## 3.1 Preliminaries

The EMB_Pack algorithm uses the concepts of MFFC [3] and MFFS [7] extensively. In this subsection, we state their definitions and important properties.

**Definition 1 (MFFC)** *In a directed acyclic graph which represents a combinational circuit, The* maximum fanout free cone (MFFC) of $v$, *denoted $MFFC_v$, is an FFC of $v$ such that for any non-PI node $w$, if output$(w) \subseteq MFFC_v$, then $w \in MFFC_v$.*

Figure 3(a) shows the MFFC of each node (the smallest rectangular area) in a network. The MFFC of each node is unique, and any FFC of $v$ is contained in $MFFC_v$. According to [3], MFFC has two important properties: (1) If $w \in MFFC_v$, then $MFFC_w \subseteq MFFC_v$; (2) Two $MFFCs$ are either disjoint or one must contain another. Moreover, it was shown in [3] that a general Boolean network can be decomposed into a set of disjoint MFFCs such that the optimal duplication free mapping for the entire network can be carried out in each MFFC independently.

**Definition 2 (FFS and MFFS)** *In a directed acyclic graph which represents a combinational circuit, for a given node set $S$, A fanout-free subgraph of $S$, denoted $FFS(S)$, is a joint fanin cone of $S$ such that for any node $u \notin S$ in $FFS(S)$, output$(u) \subseteq FFS(S)$. The maximum fanout free cone of $S$, denoted $MFFS(S)$, is an FFS of $S$ such that for any non-PI node $w$, if output$(w) \subseteq MFFS(S)$, then $w \in MFFS(S)$. $S$ is called the root set of $MFFS(S)$, and the nodes in $S$ are the root nodes of $MFFS(S)$. An MFFS is said to be a $K$-input $M$-output MFFS if input$(MFFS(S)) = K$ and $|S| = M$.*

Apparently in combinational circuits, single output MFFS is equivalent to MFFC. From Definition 2 we can see that if $u \in FFS(S)$, every path from $u$ to any PO must pass at least one node in $S$ in the circuit. In addition, we can show some interesting and important properties of MFFSs,

**Lemma 1** *For $MFFS(S)$, where $S$ is the root set,*

$$MFFS(S) \supseteq \bigcup_{v \in S} MFFC_v.$$

The nodes in $(MFFS(S) - \bigcup_{v \in S} MFFC_v)$ are called *MFFS-specific* nodes. Figure 3(b) shows the MFFS of root set $\{P, Q\}$ with $R$ as its *MFFS-specific* node. The whole circuit depicted in Figure 3(a) or (b) is an 8-input 3-output MFFS and then can be implemented in one 8-input 3-output EMB.

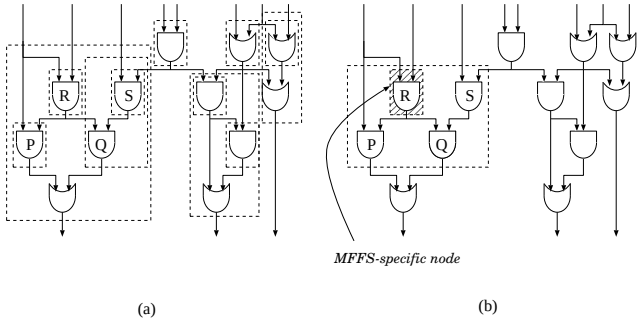


Figure 3: (a) MFFCs for all nodes; (b) $MFFS(\{P, Q\})$.

**Lemma 2** *If $u \in MFFS(S)$ and $u \notin S$, $v$ is the predecessor of some root node $r \in S$, $v$ is also the transitive fanout of $u$, and along any path between $u$ and $v$ there is no root node $\in S$, then $v \in MFFS(S)$.*

**Lemma 3** *If two MFFSs are not disjoint, one MFFS must contain at least one root node of the other MFFS and its MFFC.*

Proofs of these three lemmas are omitted due to page limitation.

### 3.2 Overview of EMB_Pack

EMB_Pack works on the mapped $K$-LUT network and is applicable to any LUT-based FPGA with EMBs. Given a mapped network, which in fact is a $K$-bounded netlist with each node as a $K$-LUT, EMB_Pack tries to appropriately *group* as many $K$-LUTs as possible into each EMB by calculating MFFCs and MFFSs in the network. We choose MFFCs and MFFSs as the candidates of EMB implementations because the nodes covered by one MFFC or MFFS do not have fanouts outside this MFFC or MFFS. Therefore, they can be implemented in one EMB without duplication, which is beneficial for area minimization. The algorithm runs in three phases: (1) select a set of MFFCs and MFFSs in the network for possible EMB implementations (Subsection 3.3); (2) those MFFCs and MFFSs with infeasible input size are cut to cater to the EMB configurations (Subsection 3.4); (3) EMBs are selected from the EMB candidates obtained from phase (1) and (2) (Subsection 3.5). The third phase also guarantees that the circuit delay will not increase, compared with the mapped input circuit.

### 3.3 MFFC and MFFS computation

Given one node $v$, according to Definition 1 we can compute $MFFC_v$ by the following procedure: (i) initialize $MFFC_v = \{v\}$; (ii) construct $MFFC_v$ by repeatedly including a transitive fanin $u$ of $v$ as soon as $output(u) \subseteq MFFC_v$ and stop when no such predecessor exists. For a combinational circuit, the time complexity of computing $MFFC(v)$ is $O(m_1)$, where $m_1$ is the number of edges in $MFFC(v)$. The same way can be used to compute $MFFS(S)$, where $S$ is the root set: (i) initialize $MFFS(S) = S$; (ii) construct $MFFS(S)$ by repeatedly including a transitive fanin $u$ of some node in $S$ as soon as $output(u) \subseteq MFFS(S)$ and stop when no such predecessor exists. The time complexity of computing $MFFS(S)$ in a combinational circuit is $O(m_2)$, where $m_2$ is the number of edges in $MFFS(S)$. For a network of size $n$, there are $n$ different MFFCs since each node has its unique MFFC. However, there are $\binom{n}{M}$ different MFFSs with root set size equal to $M$ (called an $M$-root set). As we can not afford to enumerate all the MFFSs in the network and consider EMB implementations from them, an important problem is how to choose a *good* root set $S$ as the basis to compute MFFS. Intuitively, a root set $S$ is *good* for computing MFFS if (i) the nodes in $S$ are *close* to one another; (ii) using $S$ as the root set results in as many *MFFS-specific* nodes as possible. Criterion (i) is a natural concern for routing, while satifying the second criterion will produce larger MFFS with smaller input size. We shall show later on how our heuristic algorithm tries to meet these two criteria.

Since we cannot afford to enumerate all $O(n^M)$ $M$-root sets and choose the *good* ones, even for $M = 5$, we developed a heuristic algorithm, named Root_Select, to find the *good* root sets efficiently. We first introduce some definitions.

**Definition 3** *The pth level transitive fanout set of node v, denoted $TR_p(v)$, is a set of transitive fanouts of v and the length of one of the paths between v and any node u in $TR_p(v)$ is p.*

Figure 4 shows the 2nd level transitive fanout set of node $v$ to be $\{n1, n2, n3, n4, n5\}$.

**Definition 4** *The two-level FFS of root set S, denoted $FFS_2(S)$, is an $FFS(S)$ where for any node $u \in (FFS_2(S) - S)$, $output(u) \subseteq S$.*

Figure 4 shows $FFS_2(TR_2(v))$ to be $\{n1, n2, n3, n4, n5, m1, m2, m3, m4, m5, m6\}$.

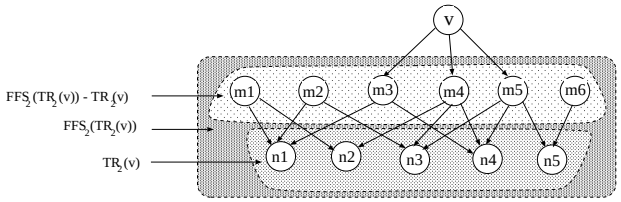


Figure 4: *$TR_2(v)$, the 2nd level transitive fanout set* of $v$, is $\{n1, n2, n3, n4, n5\}$, and $FFS_2(TR_2(v))$, the *two-level FFS* of $TR_2(v)$, is $\{n1, n2, n3, n4, n5, m1, m2, m3, m4, m5, m6\}$.

In order to select a root set with nodes close to each other, the Root_Select algorithm first calculates the $p$th level transitive fanout set $TR_p(v)$ for each node $v \in N$, where $p$ is an input parameter to EMB_Pack and is usually set to be 2. Afterwards, a *two-level FFS*, $FFS_2(TR_p(v))$, is computed, and a hypergraph $H$ is constructed. The *good* root sets are chosen from $H$ as hypergraph clusters (Definition 5) with large weights.

Given $TR_2(v)$ and $FFS_2(TR_2(v))$, the Root_Select algorithm first constructs a hypergraph $H(V, HE)$ as follows: (i) $V = TR_2(v)$; (ii) for each node $u \in (FFS_2(TR_2(v)) - TR_2(v))$, a hyperedge $e$ ($e = output(u)$) is inserted into $H$. Figure 5(b) shows how the hypergraph is constructed based on the $FFS_2(TR_2(v))$ depicted in Figure 5(a). $FFS_2(TR_2(v))$ can be taken as the union of two node sets, $S = TR_2(v) = \{n1, n2, n3, n4, n5\}$ and $T = FFS_2(TR_2(v)) - TR_2(v) = \{m1, m2, m3, m4, m5, m6\}$. For each node $m \in T$, a hyperedge $e$ ($e = output(m)$) is inserted into $H$. For example, for $m1$, there is a corresponding hyperedge $\{n1, n2\}$ in $H$. Now we introduce the concept of *cluster* in hypergraph $H$.

**Definition 5** *Given a hypergraph $H(V, HE)$, each subset of hyperedges $CE \subseteq HE$ forms a cluster $C(CV, CE)$ of $H$ with $CV = \bigcup_{e' \in CE} e'$. The weight of cluster $C(CV, CE)$ is the sum of the cardinalities of all the hyperedges in $C$ over the total possible number of hyperedges inside $C$, which is*

$$W(C) = \frac{\sum_{(e' \in CE)} |e'|}{2^{|CV|} - 1} \quad (1)$$

In Figure 5(b), hyperedges $\{n1, n2\}$ and $\{n1, n4\}$ form a cluster $C(CV, CE)$ with $CV = \{n1, n2, n4\}$ and $CE =$

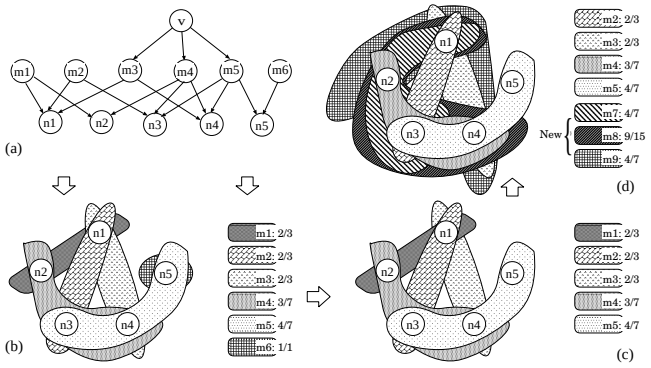


Figure 5: (a) Copy of Figure 4; (b) hypergraph initialization with those fractions as the hyperedges' weights; (c) hypergraph update after maximum hyperedge $m6$ was selected and taken away from the graph; (d) hypergraph update after maximum hyperedge $m1$ was selected and taken away from the graph.

$\{\{n1, n2\}, \{n1, n4\}\}$. The node set $CV$ of each cluster $C$ in $H$ represents a root set $S = CV$. For the initial $H$ with $m$ hyperedges, there exist $2^m - 1$ clusters. Clearly, the larger the cluster weight, the more *MFFS-specific* nodes will be introduced by the node set of this cluster compared with the cluster size, and the closer the nodes in this cluster will be to one another. In order to get *good* root sets, we pick up those clusters with weight no less than a pre-defined threshold $t$. This problem is solved by a heuristic algorithm named RN_HyperG_Select, which selects the clusters in $H$ with weight no less than $t$ in descending order. After constructing the initial $H$, RN_HyperG_Select assigns an initial weight to each hyperedge $e \in H$ as

$$W(e) = \frac{\sum_{(e' \in HE) \cap (e' \subseteq e)} |e'|}{2^{|e|} - 1} \tag{2}$$

which adds up the cardinality of every hyperedge $e' \subseteq e$ together and then divides it by the potential number of hyperedges included in $e$ (see Figure 5(b)). It then removes one hyperedge $e$ with the maximum weight in $H$ at each step, adds it to the selected cluster array if $W(e) \geq t$, and adds in some new hyperedges to $H$ as follows: for every hyperedge $e'$ which has overlap with $e$, add a new hyperedge $e_{new}$ ($e_{new} = e \cup e'$), as long as $e_{new}$ does not exist, to $H$ with weight

$$W(e_{new}) = \frac{\sum_{(e'' \in HE) \cap (e'' \subseteq e_{new})} |e''|}{2^{|e_{new}|} - 1} \tag{3}$$

Each $e_{new}$ represents the node set of a cluster in the original $H$, and $W(e_{new})$ is the same as the weight of the cluster $e_{new}$ represents. Based on this procedure, RN_HyperG_Select recursively selects and takes away the hyperedge with maximum weight from $H$, and inserts some new hyperedges into $H$, till all the hyperedges remained in $H$ have weight less than $t$. Although currently we cannot give the formal proof for the following statement, it is always observed in practice that $W(e_{new}) \leq W(e)$, where $e_{new}$ is a hyperedge inserted to $H$ when $e$ is removed from $H$. Therefore, practically the maximum hyperedge weight in the updated $H$ will not increase and the clusters in the initial hypergraph $H$ can be selected in non-ascending order by the RN_HyperG_Select algorithm efficiently.

The RN_HyperG_Select algortihm is summarized in Figure 6. Figure 5(c) and (d) illustrate the procedure of hypergraph cluster selecting. After taking away a maximum cluster $C$ from $H$, a MFFS will be computed with the root set of $CV$.

---

**Algorithm** RN_HyperG_Select
1 Set the initial weight $W(e)$ for each hyperedge $e$ in $H$ according to Eqn. 2;
2 Select the hyperedge $e$ in $H$ with maximum weight $W(e)$;
3 **while** ( $W(e) > t$ ) **do**
4 Put cluster $C$ with node set $CV = e$ at the end of the selected cluster array;
5 Take $e$ away from $H$ and update $H$ by adding in some new hyperedges according to Eqn. 3;
6 Select the hyperedge $e$ in updated $H$ with maximum weight $W(e)$;
7 **endwhile**
**end-algorithm**;

---

Figure 6: Pseudocode of the thresholded hypergraph cluster selecting algorithm.

## 3.4 MFFC and MFFS Cutting

After searching and computing MFFSs[6] in $N$, two sets of MFFSs are generated, *feasible MFFS set* and *infeasible MFFS set*. In the *feasible MFFS set*, MFFSs have input and output sizes which are feasible for at least one of the EMB configurations. However, each MFFS in the *infeasible MFFS set* needs to be cut to cater to the EMB configurations. Since our goal is to minimize the circuit area with EMBs, we want each EMB to cover as many $K$-LUTs as possible. Therefore, we formulate this problem as Problem 2.

**Problem 2** *Given $MFFS(S)$, compute the maximum volumn $K'$-feasible cut of $MFFS(S)$, where $K'$ is determined by $|S|$ to match the feasible EMB configurations.*

Currently, there is no polynomial time algorithms which can solve this problem optimally. In [4], a heuristic is proposed with time complexity of $O(K'^3 \cdot m)$, where $m$ is the number of edges in the subgraph to be cut. Although this approach guarantees *local optimality*, its time complexity is prohibitive since we may have a large number of MFFSs to be cut and $K'$ in our problem can be as large as 11 for Altera FLEX10K device. In order to solve this problem more efficiently, we propose a different heuristic algorithm. The

[6] Since $MFFC$ is the same as single output $MFFS$, we uniformly use $MFFS$ in the rest of the paper to make the description simpler.

basic idea is to repeatedly remove the node in $MFFS(S)$ with the maximum reduction on the MFFS input size, till this MFFS becomes *feasible* or no such node can be taken from $MFFS(S)$.

### 3.5 EMB selection from EMB candidates

Through MFFS computation and *infeasible MFFS* cutting, a number of *EMB candidates* including MFFSs and FFSs (cut MFFSs) are generated. All the *EMB candidates* are feasible to be EMB implementations. Since there are only $r$ EMBs available, we need to figure out how to pick up $r$ EMB implementations from those *EMB candidates* to minimize the area and at the same time maintain the circuit delay. According to Lemma 3, if two MFFSs are not disjoint, one MFFS must contain at least one root node of the other MFFS and its MFFC, which implies that if two MFFSs are not disjoint, their overlap could be a large area compared with their individual MFFS sizes. Based on this, overlap is not allowed among the MFFSs selected as EMB implementations. On the other hand, we want to maintain circuit delay after using EMBs as the delay ratio of $K$-LUT *vs.* EMB is $d_1 : d_2$ ($d_1 < d_2$). We propose a greedy algorithm which minimizes the circuit area by picking up $r$ EMBs from all EMB candidates computed by Root_Select while maintaining the circuit delay. In this algorithm, $cost(f)$ is computed for each EMB candidate $f$. We use $|f|$ to indicate the number of $K$-LUTs covered by EMB candidate $f$, use $Y$ to specify the set of EMB candidates, and $overlap(f)$ to specify the number of EMB candidates in $Y$ which have overlap with $f$. Clearly, the larger $|f|$ is, the smaller $cost(f)$ should be. Additionally, the more overlap exists between $f$ and other EMB candidates in $Y$, the larger $cost(f)$ should be since once $f$ is selected as EMB implementation, all the EMB candidates which have overlap with $f$ will be deleted from $Y$ and discarded from EMB selection. Considering these two factors, $cost(f)$ ($f \in Y$) is defined as follows,

$$cost(f) = \frac{\alpha \cdot overlap(f) + \beta \cdot \frac{\sum_{u \in Y} |u|}{|Y|}}{|f|} \qquad (4)$$

where $\alpha$ and $\beta$ are two input parameters to EMB_Pack. We usually set $\alpha$ to be 0.1 and $\beta$ to be 0.9. The algorithm works in the following steps: (1) Calculate $cost(f)$ for each $f \in Y$; (2) if $|Y| > 0$ and there are still available EMBs, select and delete $f$ with the least $cost(f)$ from $Y$; (3) Check if implementing $f$ as EMB will increase the *circuit delay*, if yes, go back to (2); (4) delete those EMB candidates in $Y$ which have overlap with $f$ and update $cost(u)$ ($u \in Y$), implement $f$ by one EMB, then go back to (2).

In step (3), the *circuit delay* is *normalized* because we take the delay ratio between LUT and EMB into consideration, and along one path from PI to PO, the delay of one LUT is $d_1$, while the delay of one EMB is $d_2$. *Normalized* delay is much more accurate than logic depth which takes every node as LUT, although some of them are already implemented as EMB outputs. Before EMB selection, a *slack* for every node in $N$ is computed which indicates the maximum delay that can be added to this single node to maintain the delay of the whole circuit. When we check in step (3) whether implementing $f$ as EMB will increase the circuit delay, a new circuit delay is calculated with the assumption that $f$ is implemented as an EMB. If the new delay is less than or equal to the original one, $f$ is accepted as EMB implementation and the slack value for each node in $N$ is updated for the next checking.

In summary, given a $K$-bounded *mapped* network, EMB_Pack first computes a set of MFFSs from those *good* root sets selected by the Root_Select algorithm. Then the MFFSs with infeasible input size are cut to cater to the EMB configurations. At the end, several MFFSs or FFSs are selected from the EMB candidates for EMB implementations.

## 4 Application of EMB_Pack as Pre-mapping Processing

In addition to post-mapping processing, EMB_Pack can also be used in pre-mapping processing to identify EMB implementations before technology mapping. Our pre-mapping processing aims at minimizing the area by using the available EMB resources while maintaining the *optimal* delay of the original circuit. The *optimal* delay of $N$ is its minimum logic depth if $N$ is mapped into a $K$-LUT network and can be obtained in the same way as that in FlowMap [4] and CutMap [5]. EMB_Pack is used before technology mapping to group as many gates as possible into the available EMBs to minimize the area while maintaining the *optimal* delay. First, the *optimal* delay is calculated through flow based labeling [4], and a *slack* is computed for each node which indicates the maximum delay that can be added to this single node to maintain the *optimal* delay of the whole circuit. After that, the EMB candidates are generated by MFFC and MFFS computation (Subsection 3.3) and MFFS cutting (Subsection 3.4). In the last step (Subsection 3.5), EMB_Pack selects EMB implementations by computing the cost of each EMB candidate and checking whether using an EMB candidate will increase the *optimal* circuit delay.

After the application of EMB_Pack as pre-mapping processing on network $N$ to choose EMB implementations, network $N'$ is generated where some nodes are implemented as EMB outputs. Thereafter, technology mapping will be performed by FlowMap [4] or CutMap [5] on $N'$ to map the remained circuit (the set of nodes which are not implemented as EMBs) into $K$-LUTs. However, since some nodes in $N'$ are already implemented as EMB outputs, it is necessary for the labeling procedure of the mapping algorithm to set the delay of an EMB to be $\frac{d_2}{d_1}$ while that of an LUT to be 1 and not to group the EMB outputs into $K$-LUTs. The optimal delay of $N'$ is preserved by our successive LUT technology mapping procedure, and it is observed in practice that the optimal delay of the original network $N$ is also preserved by

the EMB preprocessing and the subsequent LUT mapping.

## 5 EMB Selection During Technology Mapping

In the former two sections, we studied how to use EMBs after or before technology mapping to minimize the circuit area while maintaining the circuit delay. In this section, we propose the following technology mapping problem for FPGAs with EMBs and explore one possible approach to minimize the circuit delay by using $K$-LUTs and available EMB resources.

**Problem 3** *Given a $K$-bounded combinational network $N$ and the FPGA with EMBs whose architecture is described in Section 2, make use of the available EMBs and their various configurations during LUT technology mapping to minimize circuit area and/or delay.*

The input to Problem 3 is a *$K$-bounded* netlist, where each node represents a gate. We have obtained the following interesting result: assuming that there are an unlimited number of EMBs, and that the delay ratio of a $K$-LUT *vs.* an EMB is $d_1 : d_2$ ($d_1 < d_2$), we have a polynomial-time optimal mapping algorithm which makes use of $K$-LUTs and EMBs to minimize circuit delay. However, after technology mapping, the number of EMBs used may exceed $r$ and some EMB implementations need to be converted back to $K$-LUTs to meet the EMB resource limitation while minimizing the delay increase. We are currently in the process of implementing this approach.

## 6 Experimental Results

We have implemented the EMB_Pack algorithm with C language on SUN SPARC Workstation and integrated it into the RASP system developed at UCLA [8]. We designed a flow (see Figure 7) for EMB_Pack to work with Altera's FPGA development system — MAX+PLUSII 8.1. For EMB_Pack as post-mapping processing, the flow starts from reading in the netlist, performing technology mapping followed by EMB_Pack, which generates a special *eqn* file with EMB information specified in the comment section. Then, RASP converts this *eqn* file to *tdf* (Text Description Format) file as the input to MAX+PLUSII 8.1 and finally goes through MAX+PLUSII 8.1 for placement and routing. Procedures I and II in Figure 7 may interchange for EMB_Pack as pre-mapping processing. For Altera FLEX10K device family, $K$ is equal to 4, the number of available EMBs $r$ varies from 3 to 12 according to the chip size, and $d_1 : d_2 \approx 1 : 3$ or $1 : 4$. Depending on different logic funcitons, each EMB can be configured as an 11-input 1-output, 10-input 2-ouput, 9-input 4-output, or 8-input 8-ouput logic block.

Table 1 shows the comparison results of CutMap [5] and CutMap followed by EMB_Pack on MCNC benchmarks.

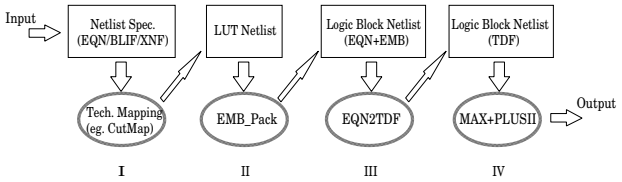


Figure 7: EMB_Pack Working Flow Overview

CutMap is an improvement over FlowMap [4], which considers area minimization during delay optimization. CutMap was run with $K = 4$. In EMB_Pack testing, the available number of EMBs for each circuit is determined by the smallest FLEX10K device into which this circuit can be fitted. We first map circuits with CutMap and then use EMB_Pack to locate the EMB implementations. EMB_Pack guarantees that the *normalized* logic delay will not increase upon the original mapped circuit. The experimental results show that compared with the original mapped circuits generated from CutMap without using EMBs, EMB_Pack with efficient use of available EMB resources can further reduce up to 10% of the area on the mapped circuits and at the same time maintain the *normalized* logic delay such that the final delay after layout remains the same.

Table 2 shows the comparison results between CutMap-e and EMB_Pack followed by CutMap-e. CutMap-e is a variation of CutMap which uses cut enumeration other than flow computation for area minimization while preserving the optimal depth. The experimental results show that compared with CutMap-e without using EMBs, EMB_Pack as pre-mapping processing followed by CutMap-e can reduce 6% of the area while maintaining the circuit optimal delay such that the layout delay remains the same.

In Table 1 and Table 2, the available number of EMBs for each circuit is determined by the smallest FLEX10K device into which this circuit can be fitted, but in order to maintain the circuit delay, EMB_Pack may not use all available EMBs. For example, in Table 1, the circuit *duke*2 does not use any EMB after EMB_Pack although it has 3 EMBs available. When delay is not a concern, EMB_Pack will exhaust almost all the available EMBs. In order to understand how the number of EMBs available will affect on the LUT area reduction without circuit delay consideration, we designed a test using EMB_Pack as post-mapping processing with gradually increasing number of available EMBs, including the *extreme* case where the number of EMBs available for each circuit is assumed to be unlimited. The minimum number of LUTs covered by an EMB is set to be 4. The results are summarized in Table 3, from which Figure 8 is drawn to show the tendency of LUT area reduction with increasing number of EMBs available. For the set of MCNC benchmark circuits we have, there are no further significant LUT area reduction for having more than 15 EMBs. Such experiment may be used for architecture study of FPGAs with EMBs.

| Circuits | CutMap | | | | CutMap+EMB_Pack | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #LUT | Depth | CPU (s) | Delay (ns) | #LUT | #C_LUT | #EMB | Depth | CPU (s) | Delay (ns) |
| C5315 | 673 | 9 | 108.2 | 55.6 | 625 | 48 | 3 | 9 | 2.2 | 59.9 |
| C6288 | 555 | 25 | 2176.8 | 108.0 | 533 | 22 | 3 | 25 | 196.0 | 102.4 |
| C7552 | 850 | 8 | 184.1 | 74.4 | 794 | 56 | 5 | 8 | 6.1 | 67.7 |
| C880 | 142 | 13 | 15.2 | 56.8 | 114 | 28 | 3 | 13 | 0.4 | 66.8 |
| 9sym | 90 | 6 | 1.8 | 32.0 | 0 | 90 | 1 | 3 | 0 | 25.5 |
| 9symml | 94 | 6 | 1.7 | 31.1 | 0 | 94 | 1 | 3 | 0.1 | 25.5 |
| alu2 | 192 | 11 | 16.2 | 52.3 | 174 | 18 | 3 | 11 | 1.2 | 61.1 |
| alu4 | 326 | 13 | 54.0 | 62.5 | 304 | 22 | 3 | 13 | 3.0 | 67.2 |
| apex6 | 300 | 6 | 6.4 | 36.7 | 279 | 21 | 3 | 6 | 0.7 | 44.6 |
| apex7 | 88 | 5 | 1.8 | 33.2 | 74 | 14 | 3 | 5 | 0.3 | 34.6 |
| des | 1367 | 6 | 49.7 | 73.6 | 1329 | 38 | 5 | 6 | 10.7 | 77.6 |
| duke2 | 199 | 5 | 5.1 | 36.5 | 199 | 0 | 0 | 5 | 0.6 | 34.0 |
| i10 | 1166 | 13 | 322.1 | 161.4 | 1082 | 84 | 8 | 13 | 10.2 | 163.6 |
| pair | 641 | 7 | 33.6 | 54.0 | 571 | 70 | 3 | 7 | 1.2 | 47.6 |
| rd84 | 83 | 5 | 1.7 | 29.5 | 0 | 83 | 1 | 3 | 0.1 | 25.5 |
| TOTAL | 6766 | 138 | 2978.4 | 897.6 | 6078 | 688 | 45 | 130 | 232.8 | 903.6 |
| Ar_Mean | 451.07 | 9.20 | 198.56 | 59.84 | 405.20 | 45.87 | 3 | 8.67 | 15.52 | 60.24 |
| Ar_Ratio | 1 | 1 | 1 | 1 | -10.17% | N/A | N/A | -5.80% | +7.82% | +0.67% |
| Ge_Mean | 294.07 | 8.18 | 21.47 | 52.82 | 0 | 0 | 0 | 7.21 | 0 | 52.30 |
| Ge_Ratio | 1 | 1 | 1 | 1 | N/A | N/A | N/A | -11.88% | N/A | -0.98% |

Table 1: Comparison between CutMap and CutMap followed by EMB_Pack
"#LUT" is the number of LUTs used. "#C_LUT" is the number of LUTs covered by EMBs. "#EMB" is the number of EMBs used.
"Depth" is the normalized logic delay which assumes the delay ratio of an LUT *vs.* an EMB to be 1 : 3.
"Delay" is the delay after placement and routing reported by MAX+PLUSII 8.1.
"Ar_Mean" and "Ge_Mean" represent *arithmetic mean* and *geometric mean* respectively.
"Ar_Ratio" and "Ge_Ratio" are ratios for *arithmetic mean* and *geometric mean*.

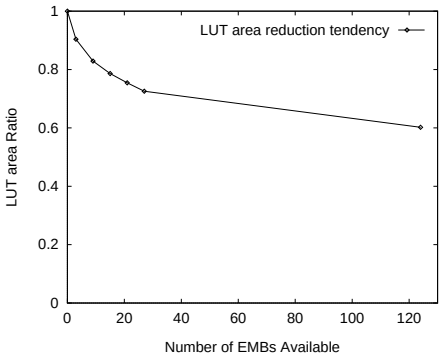


Figure 8: The LUT area reduction with increasing EMB numbers

## 7 Conclusions

In this paper, we proposed a general technology mapping problem for FPGAs with EMBs and presented an Maximum Fanout Free Cone (MFFC) [3] and Maximum Fanout Free Subgraph (MFFS) [7] based post-processing algorithm — EMB_Pack which minimizes the area after or before technology mapping by using EMBs while maintaining the circuit delay. We also proposed a delay minimization algorithm for FPGAs with EMBs which integrates technology mapping with EMB implementation together.

It remains an open problem of how to *optimally* map circuits to an FPGA with a *limited* number of LUTs and EMBs to minimize circuit area and delay. In the near future, we shall test our delay minimization algorithm for FPGAs with EMBs and gather more information about how to integrate LUT technology mapping with EMB utilization to minimize delay and/or area. Moreover, in order to minimize the circuit area with EMBs, it is important to find a large subcircuit which can be covered by an EMB. Such subcircuits are most likely being used to implement structural logic. However, it is difficult to locate this kind of subcircuit after technology independent synthesis and technology dependent mapping due to aggressive logic optimization. One possibility is to extract structural logic for EMB implementation before logic synthesis. We shall look into this approach in our future research.

## 8 Acknowledgments

We would like to give our sincere thanks to David Mendel from Altera Corp. for providing details of the FLEX10K architectures and Tim Southgate for his support of this project. We would also like to thank Yean-Yow Hwang, John Peck and Chang Wu for their helpful discussions. This work is partially supported by Altera Corp. under the California MICRO Program and National Science Foundation Young

| Circuits | CutMap-e | | | | EMB_Pack + CutMap-e | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #LUT | Depth | CPU (s) | Delay (ns) | #LUT | #C_GATE | #EMB | Depth | CPU (s) | Delay (ns) |
| C5315 | 711 | 9 | 1.2 | 55.7 | 711 | 0 | 0 | 9 | 76.7 | 55.7 |
| C6288 | 616 | 25 | 1.0 | 108.2 | 616 | 0 | 0 | 25 | 49.5 | 105.2 |
| C7552 | 826 | 8 | 1.4 | 76.2 | 771 | 428 | 9 | 8 | 156.9 | 75.8 |
| C880 | 148 | 13 | 0.2 | 61.9 | 120 | 60 | 3 | 13 | 2.8 | 55.7 |
| 9sym | 87 | 6 | 0.1 | 32.2 | 0 | 204 | 1 | 3 | 0.2 | 28.9 |
| 9symml | 93 | 6 | 0.1 | 31.9 | 0 | 199 | 1 | 3 | 0.2 | 28.9 |
| alu2 | 182 | 11 | 0.2 | 57.0 | 175 | 27 | 2 | 11 | 10.0 | 57.5 |
| alu4 | 305 | 13 | 0.3 | 65.7 | 299 | 45 | 3 | 13 | 22.5 | 75.0 |
| apex6 | 307 | 6 | 0.4 | 33.9 | 307 | 0 | 0 | 6 | 4.4 | 42.3 |
| apex7 | 91 | 5 | 0.1 | 29.0 | 91 | 0 | 0 | 5 | 0.5 | 30.7 |
| des | 1402 | 6 | 2.2 | 74.7 | 1402 | 0 | 0 | 6 | 806.9 | 71.9 |
| duke2 | 190 | 5 | 0.2 | 35.2 | 190 | 0 | 0 | 5 | 6.5 | 39.1 |
| i10 | 1126 | 13 | 1.6 | 131.3 | 1100 | 168 | 10 | 13 | 288.7 | 130.30 |
| pair | 618 | 7 | 1.0 | 47.7 | 618 | 0 | 0 | 7 | 50.2 | 47.7 |
| rd84 | 81 | 5 | 0.1 | 28.6 | 0 | 199 | 1 | 4 | 0.3 | 30.0 |
| TOTAL | 6783 | 138 | 10.1 | 869.2 | 6400 | 1330 | 30 | 131 | 1476.5 | 874.7 |
| Ar_Mean | 452.20 | 9.20 | 0.67 | 57.95 | 426.67 | 88.67 | 2 | 8.73 | 98.43 | 58.31 |
| Ar_Ratio | 1 | 1 | 1 | 1 | -5.65% | N/A | N/A | -5.07% | +14518% | +0.63% |
| Ge_Mean | 293.03 | 8.18 | 0.38 | 51.72 | 0 | 0 | 0 | 7.35 | 9.83 | 52.34 |
| Ge_Ratio | 1 | 1 | 1 | 1 | N/A | N/A | N/A | -10.17% | +2463% | +1.20% |

Table 2: Comparison between CutMap and EMB_Pack followed by CutMap-e
"#C_GATE" is the number of gates in the orginal unmapped circuit covered by EMBs.
"Depth" is the normalized logic delay which assumes the delay ratio of an LUT *vs.* an EMB to be 1 : 3 or 1 : 4.

Investigator Award MIP9357582.

## References


[1] Altera, *"Programmable Logic Devices Data Book",* Altera Corp., San Jose, CA, 1996.

[2] N. Bhat, D. Hill, *"Routable Technology Mapping for FPGAs",* First International ACM/SIGDA Workshop on Field Programmable Gate Arrays, Feb. 1992, pp. 143-148.

[3] J. Cong, Y. Ding, *"On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping",* Proc. 30th ACM/IEEE Design Automation Conference, pp. 213-218, June 1993, 213-218.

[4] J. Cong, Y. Ding, *"FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs",* IEEE Transactions on Computer-Aided Design, Feb. 1994, Vol. 13, No. 1, pp. 1-12.

[5] J. Cong, Y. Hwang, *"Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping",* UCLA Computer Science Dept. Tech. Report CSD-950001, January 1995.

[6] J. Cong, Y. Ding, *"Tutorial and Survey Paper — Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays",* ACM Transactions on Design Automation of Electronic Systems, Vol. 1, No. 2, April 1996, pp. 145-204.

[7] J. Cong, P. Li, S.-K. Lim, T. Shibuya, D. Xu, *"Large Scale Circuit Partitioning With Loose/Stable Net Removal and Signal Flow Based Hierachical Clustering",* Proc. IEEE International Conference on Computer-Aided Design, Nov. 1997, to appear.

[8] Jason Cong, John Peck, Yuzheng Ding, *"RASP: A General Logic Synthesis System for SRAM-based FPGAs",* Proc. ACM 4th International Symposium on FPGA, Feb. 1996, pp. 137-143.

[9] R. J. Francis, J. Rose, K. Chung, *"Chortle: A Technology Mapping Program for Lookup Table-Based Field Programmable Gate Arrays",* Proc. 27th ACM/IEEE Design Automation Conference, June 1990, pp. 613-619.

[10] R. J. Francis, J. Rose, Z. Vranesic, *"Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs",* Proc. 28th ACM/IEEE Design Automation Conference, June 1991, pp. 613-619.

[11] R. J. Francis, J. Rose, Z. Vranesic, *"Technology Mapping for Delay Optimization of Lookup Table-Based FPGAs",* MCNC Logic Synthesis Workshop, 1991.

[12] J. He, J. Rose, *"Technology Mapping for Heterogeneous FPGAs",* Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Feb. 1994.

[13] R. Murgai, Y. Nishizaki, N. Shenoy, R. Brayton, A. Sangiovanni-Vincentelli, *"Logic Synthesis Algorithms for Programmable Gate Arrays",* Proc. 27th ACM/IEEE Design Automation Conference, 1990, pp. 620-625.

[14] R. Murgai, N. Shenoy, R. K. Brayton, A. Sangiovanni-Vincentelli, *"Performance Directed Synthesis for Table Look Up Programmable Gate Arrays",* Proc. IEEE International Conference on Computer-Aided Design, Nov. 1991, pp. 572-575.

[15] R. Murgai, N. Shenoy, R. K. Brayton, A. Sangiovanni-Vincentelli, *"Improved Logic Synthesis Algorithms for Table Look Up Architectures",* Proc. IEEE International Conference on Computer-Aided Design, Nov. 1991, pp. 564-567.

[16] M. Schlag, J. Kong, P. K. Chan, *"Routability-Driven Technology Mapping for Lookup Table-Based FPGAs",* Proc. 1992 IEEE International Conference on Computer Design, Oct. 1992, pp. 86-90.

| Circuits | CutMap (No EMB) | | CutMap+EMB_Pack ($EMB\ number = 3$) | | | CutMap+EMB_Pack ($EMB\ number = 9$) | | | CutMap+EMB_Pack ($EMB\ number = 15$) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | #LUT | Depth | #LUT | #EMB | Depth | #LUT | #EMB | Depth | #LUT | #EMB | Depth |
| C5315 | 673 | 9 | 625 | 3 | 9 | 572 | 9 | 9 | 534 | 15 | 9 |
| C6288 | 555 | 25 | 532 | 3 | 25 | 494 | 9 | 24 | 482 | 12 | 23 |
| C7552 | 850 | 8 | 813 | 3 | 8 | 751 | 9 | 8 | 700 | 15 | 8 |
| C880 | 142 | 13 | 114 | 3 | 13 | 74 | 8 | 11 | 74 | 8 | 11 |
| 9sym | 90 | 6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 9symml | 94 | 6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| alu2 | 192 | 11 | 171 | 3 | 11 | 139 | 9 | 11 | 139 | 9 | 11 |
| alu4 | 326 | 13 | 302 | 3 | 13 | 268 | 9 | 13 | 260 | 11 | 13 |
| apex6 | 300 | 6 | 278 | 3 | 6 | 241 | 9 | 6 | 211 | 15 | 6 |
| apex7 | 88 | 5 | 69 | 3 | 5 | 50 | 7 | 5 | 50 | 7 | 5 |
| des | 1367 | 6 | 1339 | 3 | 6 | 1294 | 9 | 6 | 1255 | 15 | 6 |
| duke2 | 199 | 5 | 175 | 3 | 5 | 138 | 9 | 5 | 112 | 15 | 5 |
| i10 | 1166 | 13 | 1133 | 3 | 13 | 1073 | 9 | 13 | 1025 | 15 | 13 |
| pair | 641 | 7 | 565 | 3 | 7 | 514 | 9 | 7 | 478 | 15 | 7 |
| rd84 | 83 | 5 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| TOTAL | 6766 | 138 | 6116 | 39 | 124 | 5608 | 108 | 121 | 5320 | 155 | 120 |
| Ar_Mean | 451.07 | 9.20 | 407.73 | 2.60 | 8.27 | 373.87 | 7.20 | 8.07 | 354.67 | 10.33 | 8.00 |
| Ar_Ratio | 1 | 1 | -9.61% | N/A | -10.04% | -17.11% | N/A | -12.32% | -21.37% | N/A | -13.04% |
| Ge_Mean | 294.07 | 8.18 | 0 | 2.41 | 5.79 | 0 | 5.66 | 5.71 | 0 | 7.42 | 5.69 |
| Ge_Ratio | 1 | 1 | N/A | N/A | -29.26% | N/A | N/A | -30.24% | N/A | N/A | -30.43% |

| Circuits | CutMap+EMB_Pack ($EMB\ number = 21$) | | | CutMap+EMB_Pack ($EMB\ number = 27$) | | | CutMap+EMB_Pack ($EMB\ number = \infty$) | | |
|---|---|---|---|---|---|---|---|---|---|
| | #LUT | #EMB | Depth | #LUT | #EMB | Depth | #LUT | #EMB | Depth |
| C5315 | 502 | 21 | 9 | 472 | 27 | 9 | 373 | 51 | 9 |
| C6288 | 482 | 12 | 23 | 482 | 12 | 23 | 482 | 12 | 23 |
| C7552 | 656 | 21 | 8 | 620 | 27 | 8 | 563 | 39 | 8 |
| C880 | 74 | 8 | 11 | 74 | 8 | 11 | 74 | 8 | 11 |
| 9sym | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 9symml | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| alu2 | 139 | 9 | 11 | 139 | 9 | 11 | 139 | 9 | 11 |
| alu4 | 260 | 11 | 13 | 260 | 11 | 13 | 260 | 11 | 13 |
| apex6 | 181 | 21 | 6 | 151 | 27 | 6 | 131 | 32 | 5 |
| apex7 | 50 | 7 | 5 | 50 | 7 | 5 | 50 | 7 | 5 |
| des | 1219 | 21 | 6 | 1187 | 27 | 6 | 758 | 130 | 6 |
| duke2 | 112 | 15 | 5 | 112 | 15 | 5 | 112 | 15 | 5 |
| i10 | 984 | 21 | 13 | 948 | 27 | 13 | 793 | 58 | 13 |
| pair | 445 | 21 | 7 | 415 | 27 | 7 | 338 | 44 | 7 |
| rd84 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| TOTAL | 5104 | 191 | 120 | 4910 | 227 | 120 | 4073 | 419 | 119 |
| Ar_Mean | 340.27 | 12.73 | 8.00 | 327.33 | 15.13 | 8.00 | 271.53 | 27.93 | 7.93 |
| Ar_Ratio | -24.56% | N/A | -13.04% | -27.43% | N/A | -13.04% | -39.80% | N/A | -13.77% |
| Ge_Mean | 0 | 8.49 | 5.69 | 0 | 9.39 | 5.69 | 0 | 12.25 | 5.62 |
| Ge_Ratio | N/A | N/A | -30.43% | N/A | N/A | -30.43% | N/A | N/A | -31.27% |

Table 3: EMB_Pack testing with gradually increased number of EMBs available
"Depth" is the logic depth which assumes the delay ratio of an LUT *vs.* an EMB to be 1 : 1.