Technology Mapping Issues for an FPGA with Lookup Tables and PLA-like Blocks

Alireza Kaviani
Xilinx Inc., 2100 Logic Drive,
San Jose CA95124
alireza.kaviani@xilinx.com

Stephen Brown
University of Toronto, 10 Kings College Rd.,
Toronto, Canada M5S 3G4
brown@eecg.toronto.edu

Abstract

In this paper we present new technology mapping algorithms for use in a programmable logic device (PLD) that contains both lookup tables (LUTs) and PLA-like blocks. The technology mapping algorithms partially collapse circuits to reduce either area or depth, and pack the circuits into a minimum number of LUTs and PLA-like blocks. Since no other technology mapping algorithm for this problem has been previously published, we cannot compare our approach to others. Instead, to illustrate the importance of this problem we use our algorithms to investigate the benefits provided by a PLD architecture with both LUTs and PLA-like blocks compared to a traditional LUT-based FPGA. The experimental results indicate that our mixed PLD architecture is more area-efficient than LUT-based FPGAs by up to 29%, or more depth-efficient by up to 75%.[1]

1 Introduction

The design and implementation of digital circuits has been strongly influenced over the past several years by rapid developments in the sophistication of two types of programmable logic devices: Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). Commercially available FPGAs consist of a two-dimensional array of logic blocks, either based on LookUp Tables (LUTs), or an arrangement of multiplexers. CPLDs are composed of a collection of larger-grained blocks that is each similar to a Programmable Logic Array (PLA). A programmable device that includes both LUTs and PLA-like blocks provides advantages over existing devices [6, 7]; we refer to this new architecture as a Hybrid Field Programmable Architecture (HFPA).

Having different logic resources on the same chip complicates the technology mapping and raises the issue of providing an appropriate mixture of LUTs and PLA-like blocks in the new architecture. This paper deals with these issues by presenting new algorithms for mapping circuits to an HFPA and providing experimental results to suggest an appropriate mixture of logic resources in the new architecture. The goal of the HFPA technology mapper is to map circuits to LUTs and PLA-like blocks such that either area or speed is minimized. We consider the number of PLA-like blocks and LUTs in a mapped circuit as an approximate measure of the chip area of the circuit. Also, the maximum number of PLA-like blocks and LUTs that the signal needs to traverse from the inputs to the outputs of a combinational circuit is considered as an approximate measure for the speed of the circuit. This provides an estimate of the expected benefits of the new architecture compared to LUT-based FPGAs.

1. This work was done at the University of Toronto.

The recent *APEX* family from Altera [2] provides a combination of LUTs and PLA-like blocks on the same chip. The APEX architecture contains embedded system blocks that can be configured to function as PLAs or as memory blocks. The technology mapping techniques described in this paper can be used for both HFPA and APEX. However, the PLA-like blocks in these two architectures differ in both size and delay; therefore the potential advantages that are presented in this paper can not be applied to the APEX architecture. We discuss the background and related work in Section 2. The technology mapping algorithms are presented in Section 3, and our experimental results are given in Section 4.

2 Background and Related Work

A number of technology mapping algorithms for LUT-based FPGAs have been developed. *FlowMap*[3] is a technology mapper that optimizes depth by maximizing the flow in a network. There are also several algorithms whose initial goal is minimizing the number of LUTs. For example, [5] introduces *Level-map,* which is a technology mapping heuristic to minimize the number of LUTs. This paper also proves that the problem of obtaining minimum count for LUTs, with greater than four inputs, is NP-complete. A thorough survey of technology mapping algorithms for LUT-based FPGAs can be found in [4].

Mapping circuits to CPLDs is potentially more straightforward than technology mapping for FPGAs. One synthesis issue for CPLDs is effective two-level logic minimization, which reduces the number of product terms (Pterms) in the sum-of-products (SOP) form of a function. This is appealing for PLA-like blocks in CPLDs, because they can accommodate a limited number of Pterms, while their number of inputs can be large (typically < 40). *DDMAP* is a mapper developed by Kouloheris [8] to investigate area-efficiency of a proposed PLD based on PLA-like blocks. DDMAP first uses a LUT technology mapper to create a circuit of K-bounded nodes, where K is the number of inputs to the PLA-like block. Then the mapper packs the nodes into PLA-like blocks with multiple outputs, using a first fit decreasing algorithm. A recent study [1] presents a better approach and introduces a new heuristic for technology mapping to CPLDs. This algorithm performs the mapping in three steps: optimal tree mapping, partial collapsing, and bin packing. The first step involves partitioning of the circuit into a forest of fanout-free trees. Then the nodes in each tree are partially collapsed into their successors based on criteria such as the size or fanout of the node. Finally,

case	hsize(n)	condition
(I)	$\frac{I}{\text{LUT\_I}}$	$I \le \text{LUT\_I}$
(II)	$\frac{I \cdot Af}{\text{PALB\_I}} + \frac{P \cdot Af}{\text{PALB\_P}} - \frac{I \cdot P \cdot Af}{\text{PALB\_I} \cdot \text{PALB\_P}}$	$\text{LUT\_I} < I \le \text{PALB\_I}$ and $P \le \text{PALB\_P}$
(III)	$\frac{K \cdot I \cdot Af}{\text{PALB\_I}} + \frac{P \cdot Af}{\text{PALB\_P}} - \frac{I \cdot P \cdot Af}{\text{PALB\_I} \cdot \text{PALB\_P}}$	$\text{LUT\_I} < I \le \text{PALB\_I}$ and $P \le K \cdot \text{PALB\_P}$
(IV)	$1 + \frac{I \cdot Af}{\text{PALB\_I}}$	$\text{PALB\_I} < I$ and $P \le \text{PALB\_P}$
(V)	$1 + \frac{I \cdot P \cdot Af}{\text{PALB\_I} \cdot \text{PALB\_P}}$	$\text{PALB\_I} < I$ and $\text{PALB\_P} < P$

TABLE 1 . Definition of hsize for the node n.

the mapper packs circuit nodes into PLA-like blocks available in the target architecture, using a maximum shared input bin packing algorithm.

Technology mapping algorithms for traditional PLDs with only one kind of logic resource can not be applied to the HFPA directly. We present the algorithms for mapping to the HFPA in the next section. Although these algorithms do not depend on a specific size of PLA-like blocks or LUTs, we need to specify the size of these logic resources to be able to provide experimental results in Section 4. We have shown in [6, 7] that a PLA-like block with 16 inputs and 10 Pterms and 3 outputs is a reasonable choice for the HFPA. We refer to this PLA-like block as Programmable Array Logic Block (PALB). The HFPA also contains 4-input LUTs (4-LUTs). The details of this HFPA can be found in [6].

3 Technology Mapping Algorithms for the HFPA

This section summarizes the heuristic techniques used for each of the main steps in our technology mapping tool. There are a number of terms related to the HFPA architecture that are often used in this section:

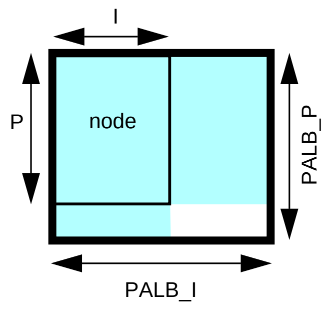
- **LUT_I** is the number of inputs to a LUT (4 in our case),
- **PALB_I** is the number of inputs to a PALB (16 in our case),
- **PALB_P** is the number of Pterms in a PALB (10 in our case),
- **I** is the fanin of a node in a circuit,
- **P** is the number of Pterms in the SOP representation of a node in a circuit.

3.1 Partial Collapsing

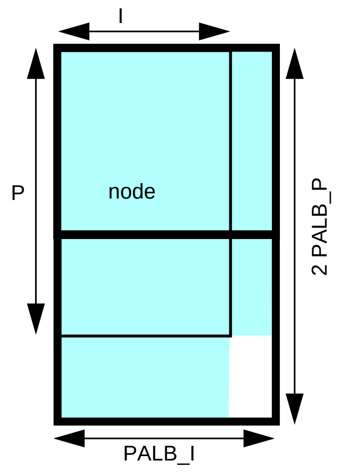
PLDs that contain PLA-like blocks with high fanin require controlled collapsing as part of their technology mapping. Collapsing is useful to reduce both the depth and area of the circuits. We consider two kinds of partial collapsing in this section: one to improve the performance of a circuit and the other to decrease its area.

Before we explain the algorithms for partial collapsing, we need to define the *hybrid size* (*hsize*) of a node, which is used as a cost function. Hsize is an estimate of the silicon area that a node occupies, where the unit area is the equivalent silicon area of a LUT and its associated routing wires and switches. The hybrid size of primary inputs (PIs) or primary outputs (POs) is defined to be zero and the hybrid size of internal nodes depends on their fanin (I) and number of Pterms (P) according to the definitions given in Table 1. The definition of the hybrid size includes five cases, as denoted in the table, where *n* is a node, *K* is an integer, and *Af* is the area factor by which the area of a PALB is larger than the area of a LUT.

The estimation of size (which is pessimistic for high-fanin but optimistic for low-fanin nodes) prevents the partial collapser from eliminating low-fanin nodes unless there is significant gain in doing so. An estimate of the size of a node is *pessimistic* when it is higher than the final chip area that will be taken by that node. Case (I) in the table estimates the size of low-fanin nodes. This is an optimistic approximation because after the final mapping there are some nodes with fanin less than LUT_I that consume one whole LUT. The second case in the table corresponds to a node that fits in a PALB. In this case the hybrid size of the node is the shaded area in



a) $\text{LUT\_I} < I \le \text{PALB\_I}$ and $P \le \text{PALB\_P}$



b) $\text{LUT\_I} < I \le \text{PALB\_I}$ and $P \le 2 \cdot \text{PALB\_P}$

Figure 1. The hybrid size of a node.

```
Lawler_clustering_algorithm {
/* LABELING */
        foreach node $n$ visited in topological from inputs to outputs (breadth-first) {
            if $n = PI$ then $L = 0$ else $L = \mathrm{MAX}(label(u))$ , $\forall u \in FaninSet(n)$
            if $hsize(n) + hsize(FaninSet(n)) \le H$ then $label(n) = L$ else $label(n) = L + 1$ }
/* RELABELING */
        foreach node $n$ visited in topological order from outputs to inputs {
            while $(n \ne PI) \wedge (n \ne PO) \wedge (n \notin \{\text{nodes on the critical path}\})$ increase label(n) }
/* CLUSTERING */
        foreach node $n$ visited in topological order from outputs to inputs {
            if $n = PO$ then $L = \infty$ and create a new cluster $C$ where $root(C) = n$
            else $L = \mathrm{MAX}(label(u))$ , $\forall u \in FanoutSet(n)$
            if $label(n) < L$ {
            create a new cluster $C = \{n\} \cup \{u \in FaninSet(n), \text{and } label(u) = label(n)\}$
            $root(C) = n$ }
        }
/* COLLAPSING */
        foreach cluster $C$ {
            collapse all nodes in $C$ into root(c) }
} /* end */
```

Figure 2. Lawler's algorithm for a given hybrid size H.

Figure 1 (a). The size estimation in case (II) is pessimistic because it ignores shared inputs in the high-fanin nodes. The hybrid size of a node that does not fit in a PALB, but whose number of inputs is less than or equal to PALB_I is estimated in case (III), where *K* is the number of PALBs needed to implement the node. The shaded area in Figure 1 (b) illustrates the hybrid size of high fanin nodes for case (III) with *K*=2. Finally, the last two cases in Table 1 correspond to the high-fanin nodes whose number of inputs is larger than PALB_I. The added '1' in cases (IV) and (V) in the table accounts for the LUT that might be needed to OR the PALBs implementing a large high-fanin node.

3.1.1 Collapsing for Depth Reduction

A simple way to improve the performance of a circuit is to collapse it into two levels of logic. Unfortunately, this technique is not applicable for a large class of circuits because the area penalty is too large for total collapsing to be practical. However, it is often possible to collapse the circuit partially in order to reduce delay at a more moderate cost in area. In this subsection we describe an algorithm, originally introduced by Lawler et al. in [9], with minor modifications. The algorithm minimizes the number of clusters a signal has to traverse from inputs to outputs, where each cluster is constrained. The cluster constraint can be the number of nodes in the cluster, the number of inputs entering the cluster, or in general any other function of the cluster structure.

An application of Lawler's clustering algorithm for delay optimization is presented in [11]. This implementation of the algorithm uses the number of nodes in the cluster as the constraint. The network should be decomposed into simple 2-input gates before applying the algorithm, to give some meaning to the number of nodes in the cluster as a capacity constraint. An implementation of this algorithm is publicly available as part of SIS [10], but it is not directly applicable to the HFPA because after logic optimization the network might have nodes of any size. Therefore we modified the cluster constraint in the algorithm implemented in SIS to use the hybrid size (hsize) of the cluster, which is defined as the sum of the hybrid sizes of all the nodes in a cluster. The heuristic algorithm, which has four steps, is outlined in Figure 2.

The first step of the algorithm, called *labeling,* uses dynamic programming. Then, in the second step the algorithm relabels all the nodes to increase the label of a node whenever possible without increasing the size of a cluster. This causes the clusters to break into smaller pieces along paths that are not critical in order to reduce the area penalty. Relabeling reduces the hybrid size of the circuits by about 9% on average with no adverse effect on depth; this agrees with the 8% area recovery that is reported in [11] using the same method. Relabeling also helps the balance of LUTs and PALBs by preventing unnecessary collapsing that leads to creation of high-fanin nodes, as is described in the next section. The third step of the algorithm places nodes with the same label that fanout to the same *root* node into one cluster, and the fourth step collapses each cluster into its root node.

3.1.2 Collapsing for Area Reduction

The advantages of HFPA are not limited to increasing the performance of the circuits by reducing the depth. In some cases low-fanin nodes can be collapsed into a high-fanin node that takes less silicon area if implemented in PLA-like blocks. Figure 3 summarizes the heuristic algorithm that we use when collapsing a circuit to reduce its area. The algorithm is integrated into the SIS tool, for

```
HFAarea_collapsing_algorithm {
        foreach node $n$ visited in from inputs to outputs (breadth-first) {
            if $(n \ne PI) \wedge (n \ne PO)$ then {
            $\cos t(n) = hsize(n) + hsize(FanoutSet(n)) - hsize((FanoutSet)^{after})$
            /* $(FanoutSet)^{after}$ is the set of fanout nodes of $n$ after $n$ is collapsed into them */
            if $(\cos t(n) < GivenCost) \wedge (pterms((FnnoutSet)^{after}) \le GivenP)$ then eliminate $n$
            }
            else continue
        }
} /* end */
```

Figure 3. Partial collapsing for area reduction.

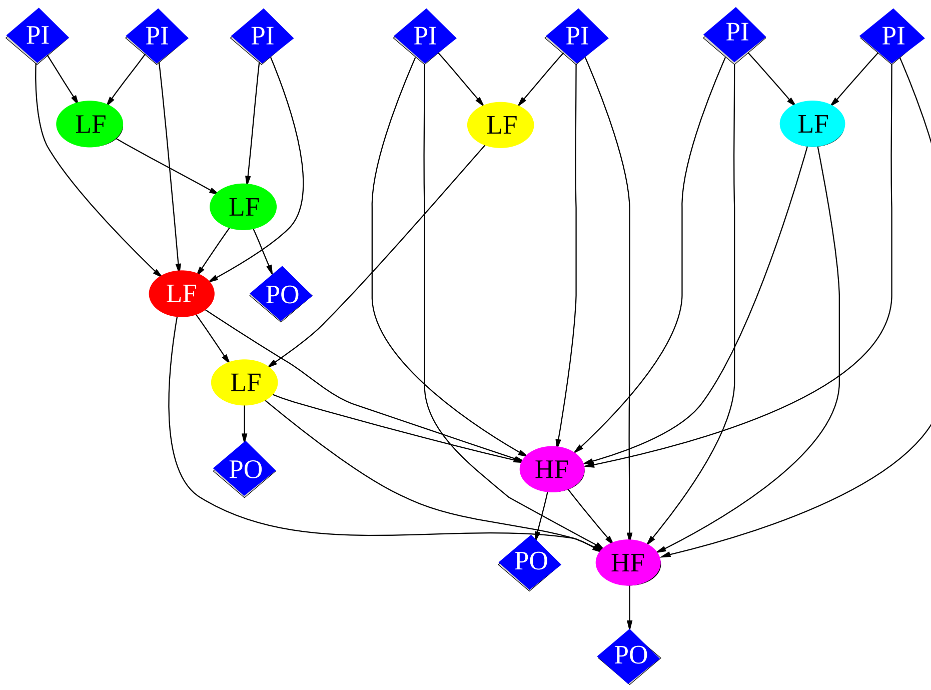


Figure 4. Example of a circuit.

convenience. The heuristic program accepts two parameters, *GivenCost* and *GivenP*, and then examines each node in the network to assign a cost to the node. The cost of a node is the increase in the hybrid size of the network if the node is eliminated by collapsing into its fanouts. If the cost of elimination of a node is greater than *GivenCost*, or the number of Pterms in any new nodes created by elimination is greater than *GivenP* the node will not be eliminated.

The implementation of our partial collapser is similar to the algorithm involved using the command *eliminate* in SIS. The cost function used in the eliminate algorithm is the *literal count* of a node. Literal count minimization, which is the main cost function in many of the algorithms in SIS, is an appropriate cost function for mapping the circuits to gate arrays. However, the literal count of nodes that fit in LUTs or PALBs does not affect the final silicon area of the mapped circuit. We compared our partial collapser to the eliminate algorithm for 10 benchmark circuits; our collapser results in 8% less area and 11% less depth on average. Another important advantage of the HFPA collapser over the eliminate algorithm is the amount of control on the collapsing, which is important for balancing the number of PALBs and LUTs, as explained later in this paper. Even the minimum possible amount of collapsing using eliminate (*eliminate -1)* results in too much collapsing for 4 out of 10 circuits that are considered.

3.2 Preparing and Packing

This section explains the final step of technology mapping, in which all the nodes in the network are packed into a minimal number of PALBs and LUTs. Before packing, we must prepare the circuit to assure that each node fits in either a LUT or a PALB. Some nodes created by logic synthesis or partial collapsing will be too large to fit into a PLA-like block, but they can be effectively partitioned. We partition a large node that does not fit in a PALB by clustering its Pterms into groups, in such a way that the Pterms in each group share the largest possible number of inputs [6].

The packing algorithm divides the nodes into two groups: high-fanin (HF) and low-fanin (LF). The number of inputs to high-fanin nodes is greater than LUT_I. The packing algorithm consists of two main parts: *fitting* high-fanin nodes into the minimum number of PALBs and *merging* low-fanin nodes into the minimum number of LUTs. The first part, which deals with high fanin nodes, sorts the nodes based on the multiplication of their number of inputs and Pterms (I×P). After sorting the high-fanin nodes in the descending order of (I×P), we fit them into the minimum number of PALBs using a *bin packing* algorithm. The choice of (I×P) is an intuitive decision based on the notion that nodes with higher (I×P) are more suitable for PALBs. The bin packing algorithm tries to place the nodes with the highest number of shared inputs into the same PALBs.

The second part of the packing operation involves merging two or more LF nodes into one node that can be implemented in a LUT. Figure 4 shows a circuit example that contains 6 low fanin and 2 high fanin nodes. The low fanin nodes with the same shade can be merged together. This problem is similar to technology mapping for traditional LUT-based FPGAs. We implemented an algorithm that traverses the network from primary inputs to primary outputs and merges the low-fanin nodes into their fanouts whenever possible to reduce the number of LUTs. If there is more than one choice for merging, the heuristic algorithm makes a decision based on the number of inputs and fanouts of the nodes. This algorithm is similar to Level-map [5] with the difference that we use only one fanout factor. Level-map tries several fanout factors for each circuit and selects the one with the best results. Changing this factor will not significantly affect the results in the HFPA because LF nodes are often connected to HF nodes and if an LF node fans out to an HF node it should be implemented regardless of its other fanouts. For the same reason the merging step has little effect on the final area of the circuits.

An optional step at the end of the packing phase considers all of the PALBs with unused outputs and unused Pterms and adds the LF

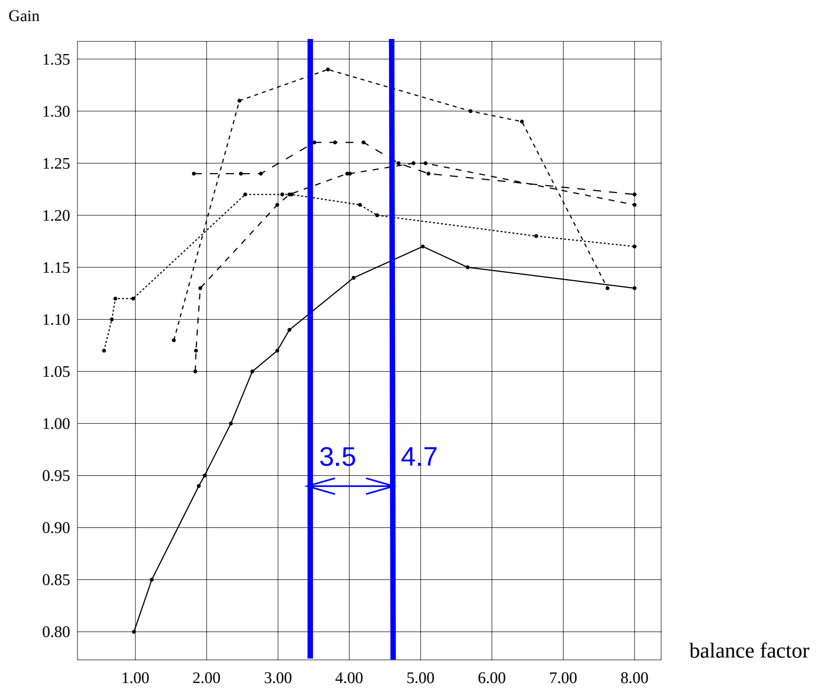


Figure 5. Balance curves for large benchmark circuits.

nodes to them until they are full. Also, at this stage it is possible to move some LF nodes in LUTs to the PALBs or decompose some of the HF nodes in PALBs and move them to LUTs. This will help to obtain a desirable balance of the number of PALBs and LUTs. Note that the best high fanin candidates to be decomposed and implemented in LUTs are at the end of the sorted list that is created for HF nodes. We added a routine to the SIS environment that traverses all the nodes in the network and decomposes the high fanin nodes with low (IXP) into LUTs[1]. The command for this new routine accepts an argument determining if the easily decomposable nodes on the critical path should remain intact. This command does not affect the final area significantly but changes the mixture of PALBs and LUTs. We discuss this issue more in the next section.

4 Experimental Results

Since no other algorithms for mapping circuits to an HFPA have been published, it is impossible to compare our techniques to others for the purpose of measuring the quality of the results. As an alternative we postulate that a technology mapper for an HFPA serves a useful purpose if it allows the HFPA to implement circuits in a more efficient manner than other types of PLDs. Accordingly, in this section we use our algorithms to demonstrate the advantages of the HFPA compared to traditional LUT-based FPGAs.

We consider the technology mapping results separately for both area and depth. Since the implementation results in an HFPA are affected by the available mixture of LUTs and PAL-like resources, we discuss this issue first.

4.1 Mixture of PALBs and LUTs

The mixture of PALBs and 4-LUTs in an HFPA that provides the best results depends on the properties of the circuit being implemented. For the results presented here we need to choose a *balance factor*, which is the ratio of the number of LUTs to the number of PALBs in an HFPA. Our algorithm can be used for any balance factor by changing the *GivenCost* and *GivenP* parameters in Section 3.1.2. The silicon area benefit, or *gain*, of an HFPA is defined as the ratio of the area when the circuit is mapped to a LUT-based FPGA over the area when the circuit is mapped to the HFPA. A *balance curve* shows how the area gain changes with respect to the balance factor for a specific circuit. We produced balance curves for a set of 15 MCNC benchmark circuits; a representative sample of five curves is given in Figure 5. The left extreme of each Bell-shape curve corresponds to a target architecture with only PALBs (balance factor=0) and the right extreme of each curve represents mapping to a LUT-based FPGA (balance factor = ∞).

We measured the balance range for each curve in which the gain is within 5% of its maximum and then calculated the intersection of those balance ranges for all the circuits. Figure 5 shows that the resulting range for these balance factor is between 3.5 and 4.7. An alternative approach for finding the best single mixture is to consider a meta circuit comprising all 15 circuits and measure the balance factor that corresponds to minimum silicon area. This alternative method results in a balance factor of four, which is approximately in the middle of the balance range shown in Figure 5. Therefore we can conclude that a mixture of PALBs and LUTs that supports a balance factor close to four is a reasonable choice. The balance factor of four implies allocation of the same silicon area to LUTs and PALBs, because a PALB occupies roughly the same area as four 4-LUTs [7]. In the remainder of this section we assume an HFPA with a balance factor of 4.

4.2 Comparison

Table 2 compares the results of implementing the MCNC benchmark circuits in both 4-LUT FPGAs and the HFPA. Our experimental procedure involves first reading the benchmark circuits

1. Simple AND/OR gates are examples of high fanin nodes with low (IXP) that can be easily implemented in LUTs.

circuit	Area: HFPA (area, depth)	balance factor	4-LUT (area, depth)	% higher (area, depth)	Depth: HFPA (area, depth)	balance factor	4-LUT (area, depth)	% higher (area, depth)
s1423	148, 17	4.1	154, 21	4%, 24%	187, 8	4.1	185, 16	-1%, 100%
frg2	243, 6	4.1	324, 8	33%, 33%	265, 5	4	324, 8	22%, 60%
x3	198, 4	3.9	282, 7	42%, 75%	207, 3	4	265, 6	28%, 100%
dalu	214, 10	3.9	357, 13	67%, 30%	358, 6	3.8	440, 12	23%, 100%
sbc	209, 8	4	266, 8	27%, 0%	229, 3	3.9	286, 7	25%, 133%
cps	421, 10	3.9	520, 8	24%, -20%	490, 6	4.2	520, 8	6%, 33%
s1488	165, 6	3.9	219, 7	33%, 17%	207, 5	4	219, 7	6%, 40%
scf	258, 6	3.8	300, 6	16%, -14%	279, 5	4	300, 6	8%, 20%
apex2	726, 13	4.1	905, 16	25%, 23%	912, 8	4	957, 15	5%, 88%
alu4	562, 10	4.1	666,10	19%, 0%	683, 6	3.9	666,10	-2%, 67%
Avg:		4		29%, 17%		4		12%, 75%

TABLE 2 . Estimate of the gain.

from their original EDIF into Synopsys Design Compiler, which is used to perform technology independent optimization. Synopsys *FPGA Compiler* is then used to map the circuits to 4-LUTs providing the results that are listed in the "4-LUT" column of Table 2 under the heading "Area". To compare the HFPA with LUT-based FPGAs the circuits are mapped using a similar optimization script. Then the circuits are converted to BLIF and partially collapsed for area using the algorithm explained in Subsection 3.1.2. The final area and depth results are presented in column "HFPA" of the table. The total area results are calculated in terms of 4-LUTs. We have made two assumptions based on our layout results presented in [6, 7]: 1) the area of a PALB and its routing equals the area of four 4-LUTs and their routing, and 2) the logic delay of a PALB is equal to the logic delay of a LUT. The results in the table show that the LUT-based FPGAs occupy 29% more silicon area than HFPA on average. Also, average depth of the resulting circuits is 17% higher when the target architecture contains only LUTs.

At this point we focus on the depth advantages of the new architecture. Reducing the depth of the circuits decreases not only the total delay due to blocks on the critical path, but also the number of switches needed to route those blocks, which in turn adds to the improvement in the speed-performance. We modify the Synopsys optimization script used for results in the left section of Table 2 and remap the circuits to reduce the delay. This change of the script should reduce the depth of the mapped circuits, though it does not guarantee it. The right section of Table 2 presents the data with the results of FPGA Compiler in column "4-LUT". [1] To minimize the depth of the circuits mapped to the HFPA we use the depth partial collapser explained in 3.1.1 before running our area collapser. After optimization by Synopsys, we collapse the circuits to a given depth and then reduce their area. To find the appropriate depth of each circuit that does not cause unacceptable area penalty we consider the graphs that present the increase in hybrid size of a circuit caused by reducing depth [6]. Each circuit is then collapsed to the depth that corresponds to the approximate knee of its corresponding graph. Table 2 shows that the depth of the circuits mapped to LUT-based FPGAs is 75% higher than their depth when mapped to the HFPA using this methodology. The HFPA still maintains an area gain of 12% on average.

1. In cases that the Synopsys script change did not reduce the depth, we conservatively used the original 4-LUT count listed on the left side of the table.

Finally, we investigate the distribution of the nodes on the above circuits' critical paths for each mapping option. On average, 66% of the nodes on all critical paths of each circuit are low-fanin, which are implemented in 4-LUTs. The remaining 34% of the nodes are high-fanin and are implemented in the PALBs. Although, the number of nodes on the critical paths increases significantly when the mapping goal is changed from area to depth, the above distribution is approximately held the same. Despite this high number of low-fanin nodes, 70% of the critical path delay (excluding the routing delay) is due to high-fanin nodes, on average. This observation emphasizes the importance of the speed of the PLA-like block. As a rule of thumb, the delay of the PLA-like block in an arbitrary hybrid architecture should be less than three or four times the LUT delay to obtain any speed gain. According to the same results, 13% of the nodes on the critical paths have fanin equal to five or six; these nodes contribute to roughly 20% of the critical path delay. Some commercial LUT-based architectures such as *Virtex* [12] contain additional circuitry for implementing nodes with five or six inputs. These architectures can provide results that are somewhat better than our reference LUT-based architecture.

5 Concluding Remarks

This paper has presented a new technology mapping algorithm for architectures that contain both LUTs and PLA-like blocks. We treat partial collapsing as a separate optimization step, by introducing two algorithms for reducing either the area or the depth of the circuit. We then pack all the high-fanin nodes into the minimum number of PALBs and merge the low-fanin nodes into the minimum number of LUTs.

To illustrate its usefulness, we used the HFPA mapper to show the advantages of the hybrid architecture compared to LUT-based FPGAs. According to our results, the area and depth gain of the HFPA are 29% and 17% on average. We also presented the results for another mapping trade-off with which HFPA provides for 12% and 75% area and depth gain, respectively. The mixture of PALBs and LUTs that provides the highest gain for an HFPA varies for different circuits. However, the HFPA provides some gain within the balance range of 3.5 to 4.7 for the majority of the benchmark circuits. We showed that it is reasonable if the hybrid architecture supports a balance factor of four, which implies roughly the same chip area for both PALBs and LUTs.

6 References

[1] J. Anderson and S. Brown, “Technology Mapping for large Complex PLDs,” Proceeding of Design Automation Conference, June 1998,

[2] F. Heile, A. Leaver, “Hybrid Product Term and LUT-based Architectures Using Embedded Memory Blocks,” Proceedings of the 1999 Symposium on FPGAs, pp. 13-16.

[3] J. Cong and Y. Ding, “FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs,” IEEE trans. on CAD of integrated circuits and systems, January 1994.

[4] J. Cong and Y. Ding, “Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays,” ACM trans. on Design Automation of Electronic Systems, Vol. 1, No. 2, April 1996, pp. 145-204.

[5] A.H. Farrahi and M. Sarrafzadeh, “Complexity of the Lookup-Table Minimization Problem for FPGA Technology Mapping,” IEEE trans. on computer-aided design of integrated circuits and systems, Nov. 1994.

[6] A. Kaviani, “Novel Architectures and Synthesis Methods for High Capacity Field Programmable Devices,” Ph.D. dissertation 1999

[7] A. Kaviani, S. Brown, “The Hybrid Field Programmable Architecture” IEEE design and test, April-June 1999.

[8] J. L. Kouloheris and A. El Gamal, “PLA-based FPGA Area vs. Cell Granularity,” Proceedings of the 1992 Custom Integrated Circuits Conference, pp. 4.31-4.3.4.

[9] E. L. Lawler, K. L. Levitt, and J. Turner, “Module Clustering to Minimize Delay in Digital Networks,” IEEE trans. on Computers, Jan. 1969, pp. 47-57.

[10] E. M. Sentovich et al., “SIS: A System for Sequential Circuit Synthesis,” Electronics Research Laboratory, Memorandum No. UCB/ERL M92/41.

[11] H. Touati, H. Savoj, R. Brayton, “Delay optimization of Combinational Logic Circuits by Clustering and Partial Collapsing,” IEEE Conference on Computer-Aided Design, 1991, pp. 188-191.

[12] Xilinx data book.