# FPGA Technology Mapping
# Using Logic Blocks with Independent LUTs

M. Korupolu, K.K. Lee, D.F. Wong

**Abstract**

The logic blocks of a lookup table (LUT) based FPGA consist of one or more LUTs, possibly of different sizes. The problem of mapping a given combinational circuit to an FPGA with different sizes of LUTs, using as few of these logic blocks as possible, is an important and non-trivial extension of the traditional LUT based technology mapping problem. In this paper, we focus on technology mapping where the target logic blocks (CLBs) consist of several *independent* LUTs of two different sizes. We use the term ICLBs for such logic blocks. The Actel ES6500 family is an example of a class of commercially available ICLBs.

In this paper, we give an $O(n^3)$ time exact algorithm for mapping a given tree network using the minimum number of target ICLBs, where $n$ is the number of nodes in the tree network. The only previous known approach for tree-based mapping to ICLBs was a bin packing based heuristic without any guarantees on the quality of the solution. Moreover the running time of this heuristic is $\Theta(n^{d+1})$, where $d$ is the maximum indegree of any node. Thus our algorithm represents an improvement over this heuristic in terms of run time, and the quality of the solution. For general networks, an effective strategy is to break it into trees and combine them. We also give an $O(n^3)$ exact algorithm for combining the optimal solutions to these trees, under the condition that LUTs do not go across trees.

We also show how to extend the method to solve mapping onto CLBs that can be configured into different ICLBs. The XC4000E is a commercially available CLB that has at least two configurations that are ICLBs.

## 1 Introduction

FPGAs are becoming increasingly popular due to the short design cycle and the low cost they offer. Current FPGAs fall into two main types based on their logic block structure: lookup table (LUT) based and multiplexor based. A $k$-input lookup table ($k$-LUT) is capable of implementing