Technology Mapping into General Programmable Cells

Alan Mishchenko Robert Brayton
Department of EECS, UC Berkeley

Wenyi Feng Jonathan Greene
Microsemi Corporation, SOC Products Group

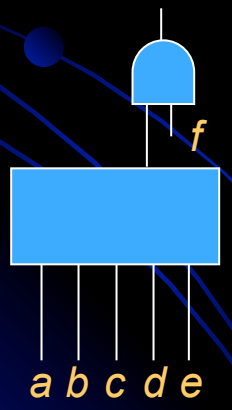
Overview

- Boolean matching is used in synthesis for FPGAs
  - To map into current programmable cells
  - To research future cell architectures
- Boolean matching methods are often specialized, slow, and require manual effort
- We proposed a new Boolean matcher, which is
  - General (works for many types of cells)
  - Fast (due to the use of concurrency)
  - Automatic (does not require manual tuning)
- Experiments show it is useful in practice

Boolean Matching

- Given a programmable cell and a Boolean functions, find if a cell can be programmed to implement a given function

Programmable cell



Function that can be implemented:

F = AND(a, b, c, d, e, f)

Function that cannot be implemented:

F = XOR(a, b, c, d, e, f)

Small Practical Functions

- Classifications of Boolean functions
  - Random functions
  - Classes of specialized functions
    - Symmetric, unate, etc
- Logic synthesis and technology mapping deal with
  - Functions appearing in the designs
  - Functions having small support (up to 16 variables)
- We call them small practical functions (SPFs)
- We concentrate on SPFs in this work

The Proposed Matching Flow

- Pre-computation
  - Collect SPFs appearing in the design(s)
  - Input and pre-process cell description
  - Perform Boolean matching concurrently for all SPFs and save data into a file
- During mapping
  - Perform additional Boolean matching step
  - Allow only matchable functions to be used

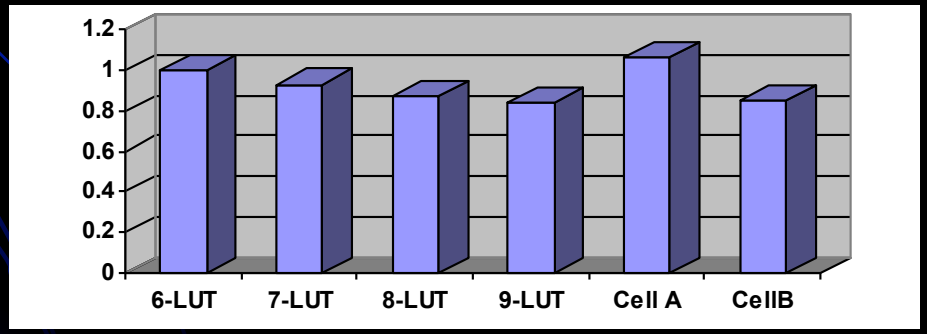
Programmable Cells Used

Cell A

Cell B

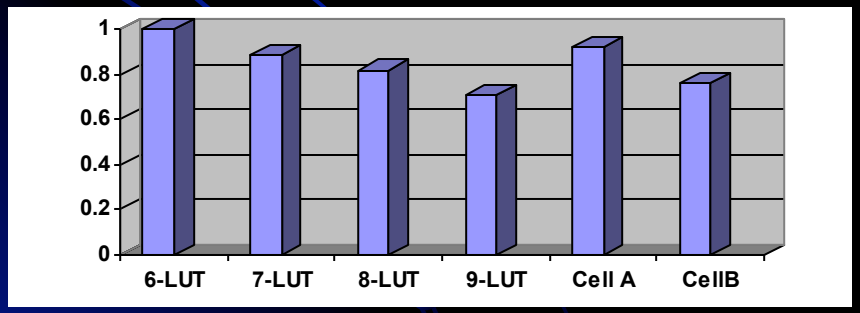
Area Comparison

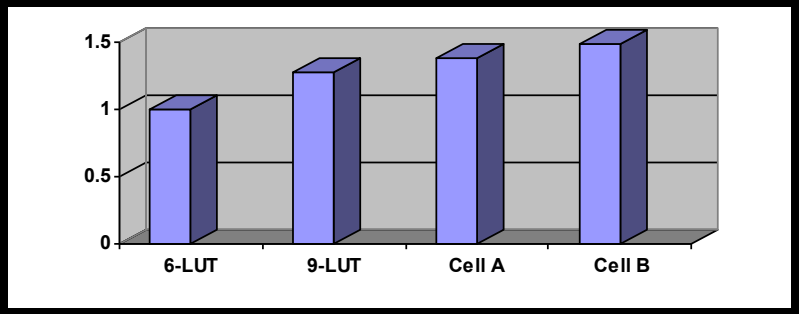
Design	Area (number of instances)					
	6-LUT	7-LUT	8-LUT	9-LUT	Cell A	Cell B
01						
02						
03						
04						
05						
06						
07						
08						
09						
10						
Geomean	1.000	0.929	0.870	0.841	1.065	0.852



Delay and Runtime Comparison

Delay (logic depth in terms of instances)						Runtime (seconds)			
6-LUT	7-LUT	8-LUT	9-LUT	Cell A	Cell B	6-LUT	9-LUT	Cell A	Cell B
1.000	0.885	0.820	0.707	0.919	0.759	1.000	1.282	1.381	1.487





Conclusions

- Introduced Boolean matching problem
- Described a mapping flow with matching
- Experimented with two programmable cells and reviewed the tradeoffs

- Future work will focus on
  - Improving implementation
  - Extending to standard cells
  - Use in technology-independent synthesis

Abstract

- Field-Programmable Gate Arrays (FPGA) implement logic functions using programmable cells, such as K-input lookup-tables (K-LUTs). A K-LUT can implement any Boolean function with K inputs and one output. Methods for mapping into K-LUTs are extensively researched and widely used. Recently, cells other than K LUTs have been explored, for example, those composed of several LUTs, and combinations of LUTs and gates. Known methods for mapping into these cells are specialized and complicated, requiring a substantial effort to evaluate such custom cell architectures. This paper presents a general approach to efficiently map into single-output K-input cells containing LUTs, MUXes, and other elementary gates. Cells with K up to 16 inputs can be handled. The mapper is fully automated and takes a logic network and a symbolic description of a programmable cell, and produces an optimized network composed of instances of the given cell. Past work on delay/area optimization during mapping is applicable and leads to good quality of results.