Impact of FPGA Architecture on Resource Sharing in High-Level Synthesis

Stefan Hadjis1, Andrew Canis1, Jason Anderson1, Jongsok Choi1, Kevin Nam1, Stephen Brown1, and Tomasz Czajkowski‡

1ECE Department, University of Toronto, Toronto, ON, Canada, ‡ Altera Toronto Technology Centre, Toronto, ON, Canada

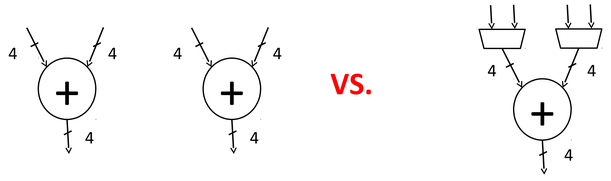
Introduction

- **Resource sharing** is an area optimization in high-level synthesis (HLS) in which a single hardware functional unit is shared by multiple operations
- How should resource sharing be adapted for different logic element architectures?
- **LegUp** open-source HLS tool, targets **Cyclone II** (4-LUTs) vs. **Stratix IV** (ALUTs)

www.legup.org

Example: 4-bit Adder

A sample C program performs two additions. Two hardware implementations exist:

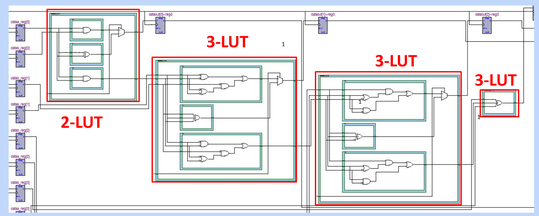


METHOD 1: Not Sharing – Two separate hardware adders

METHOD 2: Sharing – A single adder shared among both additions

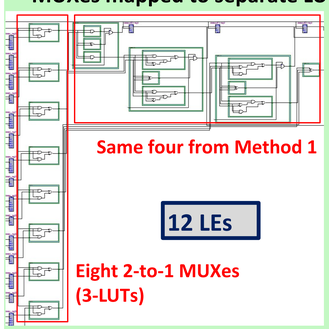
How does Quartus map these to LUTs on Cyclone II and Stratix IV?

METHOD 1: Cyclone II and Stratix IV



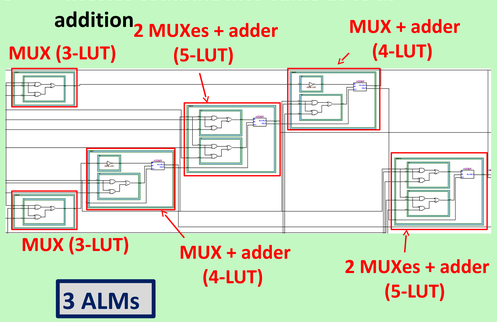
METHOD 2: Cyclone II

- **MUXes mapped to separate LUTs**



Stratix IV

- **MUXes combine into same LUTs as addition**



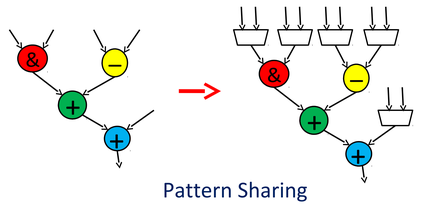
Single Operator Sharing Results

Which operators reduce area when shared?

Cyclone II	Stratix IV
Dividers	Dividers
Modulus	Modulus
Multipliers	Multipliers
Barrel Shifters	Barrel Shifters
	Add/Subtract
	Bitwise (OR, XOR, AND)

Pattern Sharing Algorithm

- Main area reduction comes from sharing **patterns** of smaller operators
- Patterns are **Data Flow Graphs** with a single root "output" node

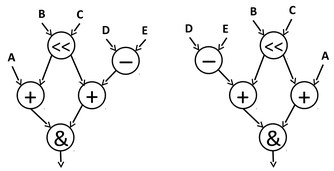


1 Discover Computational Patterns in the Software Program

- Accomplished by walking LegUp's DFG

2 Group together equivalent patterns

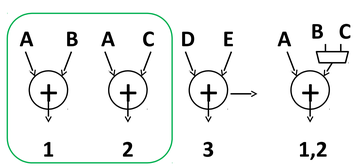
- Patterns are sorted by isomorphic equivalence to consider commutativity



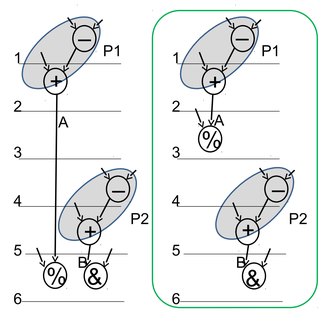
Two equivalent graphs due to commutativity

3 Pairing Patterns for Sharing

- Pair patterns to be implemented using the same hardware. Consider:
  1. Operator bitwidths
  2. Variable lifetimes
  3. Shared input variables



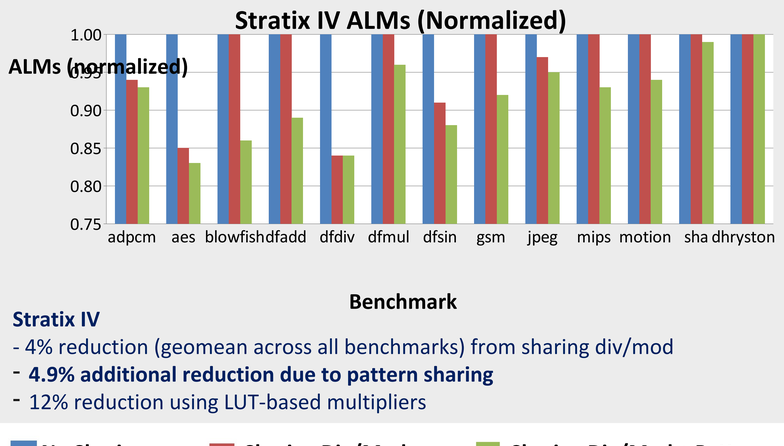
Shared Input Variables (Reduces Area)



Independent Variable Lifetimes

Pattern Sharing Results

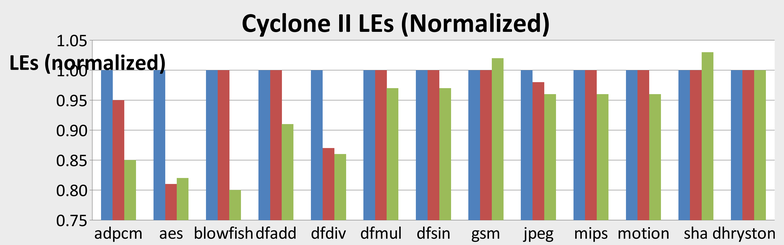
- While sharing larger patterns produces greater area reduction, the major factor is **LUT underutilization**
  - Allows MUXes to be incorporated into the same LUTs as the operator
  - **Registers** prevent an efficient mapping of operators into LUTs
- Geomean reduction in Fmax due to Pattern Sharing is 4% across all benchmarks
- Below are area results for 13 benchmarks using **three increasing levels of sharing**



Stratix IV

- 4% reduction (geomean across all benchmarks) from sharing div/mod
- **4.9% additional reduction due to pattern sharing**
- 12% reduction using LUT-based multipliers

No Sharing **Sharing Div/Mod** **Sharing Div/Mod + Patterns**



Cyclone II

- 3% reduction (geomean across all benchmarks) from sharing div/mod
- **4.2% additional reduction due to pattern sharing**
- 16% reduction using LUT-based multipliers

Summary

- Logic element architecture significantly impacts resource sharing
  - >10% area reduction in some circuits
- Future work: altering scheduling phase of HLS to favor the creation of patterns to provide further sharing opportunities