Efficient Synthesis of Compressor Trees on FPGAs

Hadi Parandeh-Afshar\textsuperscript{1,2}
Philip Brisk\textsuperscript{2}
Paolo Ienne\textsuperscript{2}

1: University of Tehran, ECE Department
2: EPFL, School of Computer and Communication Sciences
Outline

- State of the Art: FPGAs
- Motivation
- Generalized Parallel Counters
- Mapping Heuristic
- Experimental Results
- Conclusion
Outline

- State of the Art: FPGAs
- Motivation
- Generalized Parallel Counters
- Mapping Heuristic
- Experimental Results
- Conclusion
## FPGA vs. ASIC

<table>
<thead>
<tr>
<th></th>
<th>ASIC</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance</strong></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Area Utilization</strong></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Power Consumption</strong></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Flexibility</strong></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Time-to-Market</strong></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
FPGA Arithmetic Features

- Poor Performance for Arithmetic Operations Compared to ASIC
- IP Cores
  - High Routing Costs
  - Limited Flexibility; 18-bit Adder/Multiplier
- Full Adder Implemented in CLB Structure
  - Fast Carry-Chain (Xilinx and Altera)
    - Reduces Routing Delay
  - Cannot Use Compressor Trees to Add $k>2$ Values
    - Wallace/Dadda/3-Greedy
Outline

- State of the Art: FPGAs
- Motivation
- Generalized Parallel Counters
- Mapping Heuristic
- Experimental Results
- Conclusion
Motivation

- Arithmetic dominated circuits such as multimedia and signal processing
  - Multi-input ADD is inherently a frequent operation
  - ADD and MULT are dominant operations
  - Using [Verma-Ienne] transformations to integrate disperse operations
  - Accelerating multi-input additions
Verma-lenne Transformation
[ICCAD ’04]

\[
\text{SEL} = \text{SEL} + \text{SEL} + 0
\]

\[
\text{SEL} = \text{SEL} + \text{SEL} + 0
\]

\[
\text{SEL} = \text{SEL} + \text{SEL} + 0
\]

\[
\text{SEL} = \text{SEL} + \text{SEL} + 0
\]

\[
\sum \text{Compressor Tree}
\]

ADPCM
Outline

- State of the Art: FPGAs
- Motivation
- Generalized Parallel Counters
- Mapping Heuristic
- Experimental Results
- Conclusion
Counters

Count # of Input Bits Set to 1
Output # as a Binary Value

\[ n = \lceil \log_2(m+1) \rceil \]

Counters You Know
2:2 – Half Adder
3:2 – Full Adder
(Carry-Save Adder)

The correct building block for computing sums of k>2 numbers
Better than LUTs!
Generalized Parallel Counters (GPCs)

- A counter that can sum bits having different ranks
- **Representation:**
  
  \[(K_{N-2}, K_{N-1}, \ldots, K_0; S)\]

  - IO Constraints: M and N are input and output constraints
    \[
    \sum_{i=0}^{N-2} K_i \leq M \\
    \sum_{i=0}^{N-2} K_i 2^i \leq 2^N - 1
    \]

- **Examples**
  - (3, 3; 4) GPC
  - (5, 5; 4) GPC
GPC Implementation

- Using basic gates
- Using an $m:n$ counter can implement a GPC by connecting all input bits of rank $i$ to $2i$ inputs of the counter
- Using $k$-LUTs for a $k$-input GPCs
  - Current FPGAs have bigger LUTs
  - i.e. Three 6-LUTs are required for a $M=6$, $N=3$ GPC
Outline

- State of the Art: FPGAs
- Motivation
- Generalized Parallel Counters
- Mapping Heuristic
- Experimental Results
- Conclusion
Mapping Heuristic

● How to:
  ● Specify GPC configurations
  ● Connect GPCs

● Definitions
  ● Primitive GPCs (i.e. (1, 3; 3), (2, 3; 3))
  ● Covering GPCs (i.e. (2, 3; 3))
  ● Compression ratio
  ● Unreasonable GPCs (i.e. (1, 2; 3))
Mapping Heuristic

- Algorithm Objective:
  - Reducing logic levels and GPCs
- Algorithm Inputs:
  - M, N: Input/Output constraints
  - K: Number of final adder rows
  - A set of bits to be summed
- Algorithm contains 7 steps
Mapping Heuristics: Steps

- Steps 1, 2, 3 and 7 are executed once.
- Steps 4-6 occur inside a loop that generates the compressor tree.
- Steps:
  - Step1: Extracting covering GPCs.
  - Step2: Extracting primitive GPCs.
  - Step3: Sorting GPC w.r.t. compression ratio
  - Step4: Covering columns using above GPCs
  - Step5: Connecting generated GPCs to previously generated GPCs
  - Step6: Generating output bits of generated GPCs
  - Step7: Final addition, if number remained rows is less than K.
Mapping Heuristic: Step 4

- Covers all of the columns of the current logic level of the compressor tree with GPCs
- Finding base column: The tallest column
- Forward search
- Backward search
- Example
Outline

● State of the Art: FPGAs
● Motivation
● Generalized Parallel Counters
● Mapping Heuristic
● Experimental Results
● Conclusion
Experimental Methodology

- Altera Stratix-II
  - 90nm CMOS Technology
- For Multi-Input Addition Ops
  - ADD – Adder Tree
    - Ternary Adders in Stratix-II
  - 3GD – Compressor Tree
    - 3 input LUTs
  - GPCs – Compressor Tree, M=6 and N=3
    - 6 input LUTs, 2 in each ALM
Experimental results (Delay)

27% on average GPC is faster than ADD

![Graph showing delay (ns) for different operations with ADD, 3GD, and GPC]
Experimental results (Area)

5% increase in ALMs usage for GPC compared to ADD

![Graph showing area (ALM) comparison between ADD, 3GD, and GPC for various tasks such as adpcm, add2Q, fir3, G72x_2, mac, MotionEst, m12x12, m16x16, RQQBQ, RGYBY, samul, and Average.]
Experimental results (DSPs)

Resource Utilization for Adder Trees Using DSPs

- adpcm
- add2Q
- fir3
- G72x_2
- mac
- ME
- m12x12
- m16x16
- RGGQBQ
- RGYBY
- samul

DSPs
ALMs

January 22, 2008
Outline

- State of the Art: FPGAs
- Motivation
- Generalized Parallel Counters
- Mapping Heuristic
- Experimental Results
- Conclusion
Conclusion

- GPCs are flexible constructs for mapping compressor trees
- GPCs map to LUTs well
- Compressor trees built from GPCs do map onto FPGAs better than adder trees