Product-Term Based Synthesizable Embedded Programmable Logic Cores

Andy Yan,
Dr. Steven Wilton
SoC Research Lab, University of British Columbia
Vancouver, BC Canada

Programmable IP in SoC Design

Embedded Programmable Logic:
• Functionality specified through hardware configuration

Processor:
• Functionality specified using software

Fixed Logic:
• Functionality fixed at design time
• Little post-fab flexibility
“Hard” Programmable IP Flow

- Conventional approach
  - “Hard” FPGA layout provided by vendors

- Our approach
  - Synthesizable Programmable Logic Core (PLC)
  - “Soft”: HDL used to describe a PLC architecture, NOT to describe a particular user circuit
  - Synthesis required to translate RTL to gates
Soft Programmable Logic Cores

- Advantages
  - Easy to integrate, reduces design time
  - Very flexible, can create the exact required core
  - Easy to migrate to smaller technologies

- Disadvantages
  - Inefficient compared to hard cores

- Our thought
  - Makes sense if you only want a small core (a few hundred gates, perhaps)
  - e.g. next state logic in state machine

“Soft” Programmable IP Flow
Our Contribution

- Previous architecture design
  - Gradual Architecture [Kafafi et al., FPGA `03]
  - Basic logic element: Lookup-tables (LUT)

- Propose new architectural family
  - Basic logic element: Product-term array block
  - 35% density improvement
  - 72% speed improvement

Basic Logic Elements

Lookup-Table (LUT)  Product-term Block (PTB)

- i inputs
- p product-terms
- Single Output
- k Select
- o outputs
Architectural Requirements

- Area and delay minimization
  - Large capacity product-term blocks (PTB) and shallow core depth
- Simple placement and routing
  - “Full connectivity” routing fabric
- Flexible and scalable architecture
  - Architecture parameter definitions and optimizations

Synthesizable PTB Architecture

- Product-term blocks (PTBs) arranged in several levels
- Unidirectional signal flow to avoid combinational loops in un-programmed fabric
- Outputs of PTBs in one level can only be connected to inputs in subsequent levels
- 2 interconnect strategies:
  - Rectangular and Triangular PTB architecture
Detailed View of Interconnect Fabric

- Very flexible, not restrictive
- Easy P&R tools
- We can do better though

Parameter Optimization

Two Parameter Classes:

- High-Level Parameters
  - Specified by SoC core user / VLSI designer
  - Used to identify a specific core in a programmable library

- Low-Level Parameters
  - Not specified by SoC core user / VLSI designer
  - Used to describe specific characteristics of library
  - Determined through architectural experimentation
Architectural Parameters

<table>
<thead>
<tr>
<th>High-Level Parameters</th>
<th>Low-Level Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Primary Inputs Pins</td>
<td>Number of Inputs per PTB (I)</td>
</tr>
<tr>
<td>Number of Primary Output Pins</td>
<td>Number of Product-terms per PTB (p)</td>
</tr>
<tr>
<td>Number of Product-term blocks (PTBs)</td>
<td>Number of Outputs per PTB (o)</td>
</tr>
<tr>
<td>Ratio of PTBs in Neighboring Levels (r, α)</td>
<td>Ratio of PTBs in Neighboring Levels (r, α)</td>
</tr>
</tbody>
</table>

Low-Level Parameter Optimization

Rectangular PTB Architecture

\[ r = \text{ratio of width to height} \]
**Rectangular PTB Architecture Area**

![Area vs. r (ratio of width to height) graph](image)

**Rectangular PTB Architecture Depth**

![Depth of Circuit vs. r (ratio of width to height) graph](image)
Rectangular PTB Architecture

Low-Level Parameter Optimization

Triangular PTB Architecture

\( \alpha = 0.33 \), 0.5, 0.66, 0.75

\( \alpha = \) number of PTB drop-off factor

• 19% improvement in area-delay
Other Low-Level Parameters

Product-Term Block Parameters:
- input \( i = 12 \)
- product-term \( p = 9 \) or \( 18 \) depending on size of circuit
- output \( o = 3 \)

Comparison to LUT-based Architecture
- 35% area improvement, 72% delay improvement
- Gains mainly from larger circuits (more than 50 equivalent 4-LUTs)
- Factors:
  - PTB-based architecture has larger and fewer logic blocks
  - PTB-based architecture routing fabric simpler and depth of core shallower
Comparison to LUT-based Architecture

**Area (μm² x 1000)**

- # of LUTs per side
- # of PTBs

<table>
<thead>
<tr>
<th># of PTBs</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td># of LUTs per side</td>
<td>0</td>
<td>500</td>
<td>1000</td>
<td>1500</td>
<td>2000</td>
<td>2500</td>
</tr>
</tbody>
</table>

**Comparison to LUT-based Architecture**

**Delay (ns)**

- # of PTBs
- # of LUTs per side

<table>
<thead>
<tr>
<th># of PTBs</th>
<th>0</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td># of LUTs per side</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

**Graphs:**

- LUT Architecture
- PTB Architecture
Summary

- Presented a product-term based synthesizable programmable logic device
- Investigated effects of various architectural parameters
- Optimal product-term block (PTB) parameters:
  - input $i = 10$
  - product-term $p = 9$ or $18$ depending on size of circuit
  - output $o = 3$

Summary

- Compared product-term architecture to lookup-table based device
- Overall, 35% smaller and 72% faster
- Primarily due to reduction in amount of circuitry needed to route signals