HETRIS: Adaptive Floorplanning for Heterogeneous FPGAs

Kevin E. Murray and Vaughn Betz
Overview

• Heterogeneous FPGA Floorplanner
  • Dynamically adapts to targeted FPGA Architecture
  • 15.6x faster than prior work
  • Open Source

• Investigate nature of heterogeneous FPGA floorplanning

• First evaluation of a heterogeneous FPGA floorplanner on realistic benchmarks and architectures
  • Comparison to a commercial tool
Increasing FPGA Design Size

![FPGA Size and SPECInt Over Time Graph]

- **Largest FPGA**
- **Largest Monolithic FPGA**
- **SPECint**

Year:
- 1998
- 2000
- 2002
- 2004
- 2006
- 2008
- 2010
- 2012
- 2014

Normalized Value (1998):
- 50
- 100
- 150
- 200
- 250
- 300
- 350
- 400
- 450
Increasing FPGA Design Size

Need new approaches for scalable design implementation
Floorplanning
Floorplanning
Floorplanning
Floorplanning
Floorplanning
Floorplanning

- Divide-and-conquer design implementation
- Solve smaller sub-problems (potentially in parallel)
- Re-use existing CAD tools and algorithms
- Improved team-based design
- Required for Partial Reconfiguration
HETRIS: Heterogeneous Region Implementation System
Hetris: Overview

- Generate
- Move
- Realize
- Floorplan
- Evaluate

- Slicing Tree
- Irreducible Realization Lists
  [Cheng & Wong 2006]
- Area & Wirelength Costs
Hetris: Overview

Simulated Annealing

Generate Move

Realize Floorplan

Evaluate

- Slicing Tree
- Irreducible Realization Lists
  [Cheng & Wong 2006]
- Area & Wirelength Costs
Slicing Tree Moves
Slicing Tree Moves

Exchange 3 & 2
Slicing Tree Moves

Exchange 3 & 2

Rotate at c
Slicing Tree Moves

Exchange 3 & 2

Rotate at c

Exchange c & 3
Handling Heterogeneity: Irreducible Realization Lists

- Unique to every location on the FPGA
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans
Realizing Slicing Trees

• Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
• Realizations at root encode full floorplans
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans
Realizing Slicing Trees

• Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]

• Realizations at root encode full floorplans
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans
Realizing Slicing Trees

• Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
• Realizations at root encode full floorplans
Realizing Slicing Trees

- Recursively calculate shapes at each node in the tree [Cheng & Wong 2006]
- Realizations at root encode full floorplans
Algorithmic Enhancements

![Diagram of a tree structure with nodes labeled a, b, c, d, e, f, g, h1, h2, h3, h4, v1, v2, v3, v4. The tree is rooted at node a with branches b and c. The nodes d, e, f, and g are children of b and c respectively. The nodes h1, h2, h3, and h4 are children of d, e, f, and g respectively. There is a separate table with columns labeled 2, 3, 4.]}
Algorithmic Enhancements

Exchange 3 & 4

Diagram showing the exchange of nodes in a tree structure.
Algorithmic Enhancements

Exchange 3 & 4

Common sub-trees

[Diagram showing tree structures and exchange of nodes]
Algorithmic Enhancements

Memoization

• Save intermediate results
• Re-use instead of re-calculating

Common sub-trees
Algorithmic Enhancements

Memoization

- Save intermediate results
- Re-use instead of re-calculating

Lazy Evaluation

- Calculate leaf shapes as needed to avoid wasted work
- Important for non-tileable FPGAs
Impact of Algorithmic Enhancements

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Speed-Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>1.0x</td>
</tr>
<tr>
<td>Memoization</td>
<td>2.3x</td>
</tr>
<tr>
<td>Lazy Evaluation</td>
<td>5.4x</td>
</tr>
<tr>
<td>Memoization &amp; Lazy Evaluation</td>
<td>15.6x</td>
</tr>
</tbody>
</table>

- Titan Benchmarks: 90K – 550K primitives
- **Average run-time**: 9 minutes @ 32 partitions
Floorplan Legality
How to ensure legal solution?

• Impractical to forbid illegal solutions
• Cost penalty: Floorplan area outside the device
Split Cost Penalty

- Use separate cost penalties for horizontal and vertical legality
Legal Solution

![Diagram of legal solution]
Search Space

Tall & Narrow

- Width Limit
- Height Limit
- Minimum Area
Search Space

Tall & Narrow

Short & Wide
Adaptive Legality

- Need robust cost penalty
- Dynamically adapt penalty based on legal acceptance rate
- Stall the anneal until legality achieved
Experimental Results
Experimental Setup

- **Benchmarks**: Titan (90K - 550K primitives)
- **Architecture**: Stratix IV-like
- **Partitioner**: Metis
- **Packer**: VPR
- **Floorplanner**: Hetris
- **Area and Wirelength Optimization**
Floorplan Area and Number of Partitions

![Graph showing the relationship between normalized floor area and number of partitions. The x-axis represents the number of partitions on a logarithmic scale, and the y-axis represents the normalized floor area also on a logarithmic scale. The graph shows a trend of increasing normalized floor area with increasing number of partitions.](image-url)
Floorplan Area and Number of Partitions

A moderate number of partitions (up to 32) yield reasonable overheads.
Comparison with Quartus II

- Scalable benchmark (Cascaded FIR filters)
- Limited by DSP blocks on EP4SGX230 device
- Consider both 1-FIR and 2-FIR instances per partition

<table>
<thead>
<tr>
<th>Automated Design Flow</th>
<th>Max. FIR Inst. 1-FIR</th>
<th>Max. FIR Inst. 2-FIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quartus II</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>Hetris Default</td>
<td>38</td>
<td>44</td>
</tr>
<tr>
<td>Hetris High-Effort</td>
<td>39</td>
<td>44</td>
</tr>
</tbody>
</table>
Conclusion and Future Work
Conclusion

- Hetris open source FPGA floorplanning tool
- Algorithmic enhancements yielding 15.6x speed-up
- Adaptive optimization techniques to robustly handle legality
- First evaluation of FPGA floorplanning using realistic benchmarks and architectures
Future Work

Hetris

- Further algorithmic enhancements
- Timing-driven optimization
- Support for non-rectangular shapes

Design Flow

- Improved automated design partitioning
- Full post-place & route evaluation of floorplanning
Thanks!

Questions?
Email: kmurray@eecg.utoronto.ca

HETRIS Release:

uoft.me/hetris