RippleFPGA: A Routability-Driven Placement for Large-Scale Heterogeneous FPGAs

Chak-Wa Pui, Gengjie Chen, Wing-Kai Chow, Ka-Chun Lam, Jian Kuang, Peishan Tu, Hang Zhang, Evangeline F. Y. Young, Bei Yu
CSE Department, Chinese University of Hong Kong, Hong Kong
Speaker: Jordan, Chak-Wa Pui
Outline

• Background
• Problem Formulation
• Algorithms
• Experimental Results
Introduction

• As the scale of FPGA grows rapidly, routability becomes a major problem in FPGA placement

• The complex architecture of heterogeneous FPGAs yields more sophisticated placement techniques
Previous Works

• Three major categories
  • Simulated annealing based, e.g. VPR
  • Partitioning-based, e.g. [1]
  • Analytical approach, e.g. [2][3]

• Limitations of previous works
  • Very few of recent works considering routability
  • Previous works mainly consider routability in packing
  • Most of previous works on heterogeneous FPGAs pack logic elements into CLB and seldom change them after packing

Contributions

• A framework for heterogeneous FPGA flat placement
• Several methods are proposed to reduce routing congestion
  • Partitioning
  • Multi-stage congestion-driven global placement
  • Alignment-aware detailed placement
Problem Formulation

• Routability-driven FPGA placement
  • Given the netlist and architecture of an FPGA
  • Minimize: routed wirelength measured by VIVADO
  • Subject to: each logic element has no overlap, no violation to the architecture specific legalization rules
Overview of Our Algorithm

- Partitioning
- Packing
- Routability-Driven Global Placement
- Legalization
- Routability-Driven Detailed Placement
Overview of Our Algorithm

• Partitioning
• Packing
• Routability-Driven Global Placement
• Legalization
• Routability-Driven Detailed Placement
Partitioning

• Motivation
  • Unbalance between width and height of the chip
  • Cannot be resolved by spreading but by reallocating

Comparison of different methods in solving congestion
Partitioning

• Solution:
  • Partition the circuit into sub-circuits using recursive bi-partitioning
    • Cluster size less than 25% of #cells, cut size less than 5% of #net
  • Reallocate the cells across the chip as sparse as we can
    • Maintain relative order of clusters and cells inside the same cluster
    • Give more space for the cells in spreading while not increase the HPWL too much
Partitioning

• Effect on real test case

(a) w/o partitioning

(b) with partitioning

Comparison of spreading result w/o and with partitioning
Overview of Our Algorithm

• Partitioning
• Packing
• Routability-Driven Global Placement
• Legalization
• Routability-Driven Detailed Placement
Packing

1. Short global placement
2. Forming basic logic elements (BLEs) that consist of only one LUT and at least one FFs
3. Let the remaining LUTs and FFs be BLEs of itself only
4. Merging two BLEs into one if their LUTs have many connections

Different type of basic logic elements (BLEs)
Packing

• Use maximum weight matching in stage2, weight proportional to distance, only connected LUTs and FFs have edges

• In stage3, let the remaining LUTs and FFs be BLEs of itself only

How we do packing in stage2,3
Packing

- Use maximum weight matching in stage 4, weight proportional to distance and connections between the LUTs of the vertices

How we do packing in stage 4
Overview of Our Algorithm

• Partitioning
• Packing
• Routability-Driven Global Placement
• Legalization
• Routability-Driven Detailed Placement
Global Placement

• Quadratic placement based on Ripple
• Three-stage Optimization
  • First two stages, optimize wirelength, fix DSP/RAM to their legal position after stage 1
    • Legalizing DSP/RAM disturbs the global placement result a lot
  
  ![Same displacement, difference in HPWL](image1)
  ![Large displacement](image2)

• The third stage optimize routability using inflation (Accumulative)
Global Placement

• Routing congestion estimation
  • Probabilistic model
  • Consider both bounding box and HPWL

\[ \text{Cong}_{sti} = \sum_{m \in N_i} \frac{W_m \cdot \text{HPWL}_m}{\text{#G-Cells covered by net } m} \]

\[ W_m \cdot \text{HPWL}_m = 6 \]
\[ \#\text{G-Cell} = 6 \]

\[ W_m \cdot \text{HPWL}_m = 5 \]
\[ \#\text{G-Cell} = 3 \]
Routing congestion estimation

Comparison of the routing congestion estimation obtained by VIVADO and us
Overview of Our Algorithm

• Partitioning
• Packing
• Routability-Driven Global Placement
• Legalization
• Routability-Driven Detailed Placement
Legalization

• Greedy window-based cell by cell legalization process
  • Start from a small window
  • Sites inside a window are consider to have same displacement
  • Sort the sites by an objective function (HPWL)
  • If cannot be legalized, slowly increase the window size until it’s legalized
  • Keep BLEs intact unless cannot be legalized
Overview of Our Algorithm

• Partitioning
• Packing
• Routability-Driven Global Placement
• Legalization
• Routability-Driven Detailed Placement
Detailed Placement

• Move to optimal region to optimize HPWL
  • In both BLE level and CLB level
  • In CLB level, if the site is occupied, swap the cells if the HPWL does not increase
  • In BLE level, the BLE can only be moved to a slice if it won’t violate legalization rules
Detailed Placement

• If already in optimal region, move to site to optimize alignment (BLE level).
  • Compute the score of each site by assuming the cell is moved to there and get the alignment score by considering all related nets
  • Sort the candidate sites by their alignment score, try to move the cell to a site with smaller score

Comparison of alignment of different placement

(a) score=5  (b) score=4  (c) score=3  (d) score=2  (e) score=1
## Experimental Result

<table>
<thead>
<tr>
<th>Design</th>
<th>Ours</th>
<th>1st Place</th>
<th>2nd Place</th>
<th>3rd Place</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WL</td>
<td>Time(s)</td>
<td>Normalized WL</td>
<td>WL</td>
</tr>
<tr>
<td>FPGA-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>362563</td>
<td>74</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>FPGA-2</td>
<td>677563</td>
<td>167</td>
<td>1</td>
<td>677877</td>
</tr>
<tr>
<td>FPGA-3</td>
<td>3617466</td>
<td>1037</td>
<td>1</td>
<td>3223042</td>
</tr>
<tr>
<td>FPGA-4</td>
<td>6037293</td>
<td>621</td>
<td>1</td>
<td>5628519</td>
</tr>
<tr>
<td>FPGA-5</td>
<td>10455204</td>
<td>1012</td>
<td>1</td>
<td>10264769</td>
</tr>
<tr>
<td>FPGA-6</td>
<td>6960037</td>
<td>2772</td>
<td>1</td>
<td>6330179</td>
</tr>
<tr>
<td>FPGA-7</td>
<td>10248020</td>
<td>2170</td>
<td>1</td>
<td>10236827</td>
</tr>
<tr>
<td>FPGA-8</td>
<td>8874454</td>
<td>1426</td>
<td>1</td>
<td>8384338</td>
</tr>
<tr>
<td>FPGA-9</td>
<td>12954350</td>
<td>2683</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>FPGA-10</td>
<td>8564363</td>
<td>5555</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>FPGA-11</td>
<td>11226088</td>
<td>3636</td>
<td>1</td>
<td>11091383</td>
</tr>
<tr>
<td>FPGA-12</td>
<td>8928528</td>
<td>9748</td>
<td>1</td>
<td>9021768</td>
</tr>
</tbody>
</table>

Comparison of wirelength and runtime of our placer and the top3 winners
## Experimental Result

<table>
<thead>
<tr>
<th>Design</th>
<th>Raw</th>
<th>With congestion-driven GP</th>
<th>With partitioning</th>
<th>With Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA-1</td>
<td>362563</td>
<td>1.000</td>
<td>364143</td>
<td>1.004</td>
</tr>
<tr>
<td>FPGA-2</td>
<td>681418</td>
<td>1.000</td>
<td>677563</td>
<td>0.994</td>
</tr>
<tr>
<td>FPGA-3</td>
<td>4027586</td>
<td>1.000</td>
<td>3999517</td>
<td>0.993</td>
</tr>
<tr>
<td>FPGA-4</td>
<td>6037293</td>
<td>1.000</td>
<td>6087199</td>
<td>1.008</td>
</tr>
<tr>
<td>FPGA-5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FPGA-6</td>
<td>7801736</td>
<td>1.000</td>
<td>7723476</td>
<td>0.990</td>
</tr>
<tr>
<td>FPGA-7</td>
<td>10248020</td>
<td>1.000</td>
<td>10615672</td>
<td>1.036</td>
</tr>
<tr>
<td>FPGA-8</td>
<td>9171179</td>
<td>1.000</td>
<td>9392039</td>
<td>1.024</td>
</tr>
<tr>
<td>FPGA-9</td>
<td>12954350</td>
<td>1.000</td>
<td>13437554</td>
<td>-</td>
</tr>
<tr>
<td>FPGA-10</td>
<td>-</td>
<td>-</td>
<td>10372369</td>
<td>-</td>
</tr>
<tr>
<td>FPGA-11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FPGA-12</td>
<td>-</td>
<td>-</td>
<td>10286583</td>
<td>-</td>
</tr>
</tbody>
</table>

Comparison of wirelength of different method used in our placer
Thanks