Overview

• Steps in multi-FPGA software

• Bipartitioning

• Logic Replication

• Partition Ordering

• Theoretical limits of multi-FPGA systems.
Multi-FPGA Software

- Missing high-level synthesis
- Global placement and routing similar to intra-device CAD
System-level Constraints

- Even though general solutions are desirable, system specific issues must be considered.
- For many systems, designs are created independently of the system.
- Software efficiency determines performance and usability.
Bipartitioning

- Perhaps biggest problem in multi-FPGA design is partitioning

- Partitioner must deal with logic and pin constraints.

- Could simultaneously attempt partitioning across all devices. Even “simple” algorithms are $O(n^3)$

- Better to recursively bipartition circuit.
KLFM Partitioning

- Identify nodes to swap to reduce overall cut size
- Lock moved nodes
- Algorithm continues until no un-locked node can be moved without violating size constraints
KLFM Partitioning

Create initial partitioning;
While cutsize is reduced {
    While valid moves exist {
        Use bucket data structures to find unlocked node in each
        partition that most improves/least degrades cutsize when
        moved to other partition;
        Move whichever of the two nodes most improves/least degrades
        cutsize while not exceeding partition size bounds;
        Lock moved node;
        Update nets connected to moved nodes, and nodes connected to
        these nets;
    } endwhile;
    Backtrack to the point with minimum cutsize in move series just
    completed;
    Unlock all nodes;
} endwhile;

Figure 1. The Fiduccia-Mattheyses variant of the Kernighan-Lin algorithm.

- Key issue is implementing node costs in lists that can be easily
  accessed and updated.
- Many extensions to consider to speed up overall optimization
- Reasonably easy to implement in software
Partition Preprocessing: Clustering

- Identify bin size
- Choose a seed block (node)
- Identify node with highest connectivity to join cluster
- Terminate when cluster size met.

In practical terms, a cluster size of 4 works best.
Clustering

- Technology mapping before partitioning is typically ineffective since frequently area is secondary to interconnect

- Frequently bipartitioning continues after unclustering as well.

  ![Diagram](image.png)

  - This allows for additional fine-grain moves.
Initial Partition Creation

- KLFM primarily designed to operate on fixed-sized partitions.

- Several approaches exist to distribute nodes between the two partitions
  - Random -> assign $\frac{1}{2}$ to each
  - Breadth-first -> select a node, select the next node attached to it
  - Depth-first -> similar to B.F. except get all attached nodes
Partition Creation Results

<table>
<thead>
<tr>
<th>Mapping</th>
<th>Random</th>
<th>Seeded</th>
<th>Breadth-first</th>
<th>Depth-first</th>
</tr>
</thead>
<tbody>
<tr>
<td>s38417</td>
<td>57</td>
<td>69</td>
<td>57</td>
<td>65</td>
</tr>
<tr>
<td>s38584</td>
<td>54</td>
<td>87</td>
<td>56</td>
<td>54</td>
</tr>
<tr>
<td>s35932</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>industry3</td>
<td>427</td>
<td>477</td>
<td>427</td>
<td>427</td>
</tr>
<tr>
<td>industry2</td>
<td>181</td>
<td>181</td>
<td>181</td>
<td>181</td>
</tr>
<tr>
<td>s15850</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>s13207</td>
<td>73</td>
<td>75</td>
<td>80</td>
<td>74</td>
</tr>
<tr>
<td>biomed</td>
<td>83</td>
<td>105</td>
<td>104</td>
<td>102</td>
</tr>
<tr>
<td>s9234</td>
<td>52</td>
<td>68</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>s5378</td>
<td>68</td>
<td>79</td>
<td>80</td>
<td>78</td>
</tr>
<tr>
<td>Geom. Mean</td>
<td>82.4</td>
<td>95.3</td>
<td>86.7</td>
<td>86.5</td>
</tr>
<tr>
<td>Time</td>
<td>499.5</td>
<td>511.4</td>
<td>498.6</td>
<td>502.1</td>
</tr>
</tbody>
</table>

- Suprisingly random appears to be the best
- For the largest designs, results similar
- For smaller designs, variance across designs
- Seeded ->start from an empty partition and apply KLFM
Higher-level Gains

- Effectively look-ahead to try to anticipate next move

- Look-ahead of 3 considered best tradeoff
Partition Size Variation

- Most bipartitions must be balanced so that full FPGA utilization may be achieved.
- Frequently application designers do not create circuits that are evenly balanced.
Logic Replication

• Attempt to reduce cutset by replicating logic.
• Every input of original cell must also input the replicated cell.
• Replication can either be integrated into the partitioning process or used as a post-process technique.

Figure 1. An example of node replication. Node A is replicated into A1 and A2, yielding a gain of 1.
Example: Kring-Newton Replication

- Introduce a new state to partitioning
  - Node can exist in separate locations
- Possible node moves include gain/reduce, replication, and unreplication
- Positive unreplication moves must be taken before any other moves
- Gradient technique-only allow replication when cut-size changes by more than 10%
Kring-Newton Results

<table>
<thead>
<tr>
<th>Bench.</th>
<th>Straw</th>
<th>Basic</th>
<th>KN</th>
<th>HLG</th>
<th>KN</th>
<th>Clust.</th>
<th>KN</th>
<th>Grad.</th>
</tr>
</thead>
<tbody>
<tr>
<td>s5378</td>
<td>60</td>
<td>41</td>
<td>39</td>
<td>48</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s9234</td>
<td>45</td>
<td>30</td>
<td>32</td>
<td>33</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s13207</td>
<td>67</td>
<td>53</td>
<td>48</td>
<td>49</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s15850</td>
<td>52</td>
<td>43</td>
<td>43</td>
<td>39</td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6288</td>
<td>50</td>
<td>33</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>biomed</td>
<td>161</td>
<td>136</td>
<td>137</td>
<td>126</td>
<td>130</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prim2</td>
<td>133</td>
<td>114</td>
<td>114</td>
<td>110</td>
<td>114</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>struct</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s35932</td>
<td>46</td>
<td>39</td>
<td>39</td>
<td>28</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s38584</td>
<td>50</td>
<td>35</td>
<td>35</td>
<td>46</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s38417</td>
<td>55</td>
<td>51</td>
<td>49</td>
<td>47</td>
<td>49</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>60.8</td>
<td>48.2</td>
<td>47.7</td>
<td>47.6</td>
<td>44.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>17.4</td>
<td>17.3</td>
<td>17.6</td>
<td>17.1</td>
<td>19.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rep %</td>
<td>0.0%</td>
<td>4.2%</td>
<td>4.5%</td>
<td>6.2%</td>
<td>3.5%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Results indicate 20% improvement in cut size with 5% increase in logic node count.
- Minimal increase in computation time
Functional Replication

- Applied to tech-mapped Xilinx blocks.

- Outputs in CLBs split into two CLBs

- Only inputs needed by both CLBs split across partitions.
Replication Summary

- Tech mapping before partitioning shown to be ineffective (again)
- Kring-Newton simple but effective
- Overall summary of bipartitioning
  - Use random initial placement
  - Bandwidth clustering
  - High-order gain of 3 and Kring-Newton to achieve best results
Logic Partition Ordering

- Simply bipartitioning not enough. Knowing what to partition is important.
- One approach -> locate critical point of expected wires/available wires and partition here first.
- Example above shows alternating horizontal and vertical cuts.
Terminal Propogation

• Even though bipartitioning occurs with a fixed set of nodes, previously cut nodes may play a factor.
• Consider recursive cut. Need to use “anchors” to guide partitioning.
Splash 2

- 68 connections most FPGAs, only 35 between A-7
- More balanced with even schedule
- Somewhat unimportant due to Splash programming style.
Are Meshes Really Realistic?

- The number of wires leaving a partition grows with Rent’s Rule
  \[ P = K G^B \]
- Perimeter grows as \( G^{0.5} \) but unfortunately most circuits grow at \( G^B \)
  where \( B > 0.5 \)
- Effectively devices highly pin limited
- What does this mean for meshes?
Summary

- Multi-FPGA system software requires many steps.

- Bipartitioning has been the subject of much research.

- Surprisingly, simple approaches to initializing partitions and replicating logic is most effective.

- Pin limitations pose a problem -> address this issue in next class.