ESE535: Electronic Design Automation

Day 15: March 13, 2013
High Level Synthesis II
Dataflow Graph Sharing

Today

Sharing
• Dataflow subgraph
  – Pattern identification
  – Pattern selection

Flow Review

Additional Concerns?
What are we still not satisfied with?
• Parallelism in hyperblock
  – Especially if memory sequentialized
  • Disambiguate memories?
  • Allow multiple memory banks?
• Only one hyperblock active at a time
  – Share hardware between blocks?
• Data only used from one side of mux
  – Share hardware between sides?
• Most logic in hyperblock idle?
  – Couldn’t we pipeline execution?

Preclass

• Common subgraphs?
• How would we like to share?
  – If trying to avoid slowdown
  – If willing to make area-time tradeoffs?

Subgraph Sharing

• Can potentially share identical subgraphs
• Can share similar subgraphs
Evaluating Subgraph Sharing

• What do we have to do to share subgraphs?

• When is it worthwhile?
  – How big does graph need to be?
  – How much overhead to share?

Example

• Muxes on inputs to an adder
  – Probably bigger than just having two adders
  – 2(Amux) + Aadd > 2(Aadd)

• Muxes on input to multiplier
  – Probably smaller than two multipliers
  – 2(Amux+Ampy) < 2(Ampy)

Extreme Case

• If ignored multiplexing overhead, what would we get?

VLIW Extreme

• Sketch
  – Each basic block requires a set of operators to achieve minimum path length
  – Union sets over all basic blocks
  – Build VLIW with that operator set

• Why unsatisfying?

Favorable Subgraphs

• Particularly beneficial when I/O into subgraph small
  – Overhead for muxing proportional to inputs

Approach

• Find candidate, reusable subgraphs
• Select a cover set of subgraphs
• Assign original graph to subgraphs
  – Assess benefits of sharing
• Patch together subgraphs with control and multiplexing
Find Subgraphs

• How might we find the set of candidate subgraphs?

Finding Subgraphs

• Keep set of subgraphs of size k
• Create subgraphs of size k+1 from subgraphs of size k
  – By adding a neighboring node
    • Maybe several such expansions for each k-subgraph
• Careful: can end up with exponential subgraphs

Optimization

• Canonicalize subgraphs so recognize when encounter same subgraph again
  – Keep set of subgraphs small

Optimization

• Compute candidate graph patterns during subgraph generation
  – Each subgraph may become a candidate
  – Keep track of subgraphs that might match with candidate subgraphs
  – As add subgraph, compare it with candidate patterns and add to list if “close” enough
  – At end of a given graph size, prune out patterns with too few potential matches

Close enough?

• Conceptually: not too expensive to use the candidate pattern
• Concretely: compute a distance metric between graph and pattern
  – Minimum cost of edits to morph one graph into another
    • E.g. relabel nodes, remove nodes
  – Want to capture potential cost of adding muxes and control

[Cong & Jiang / FPGA 2008]
Cover Subgraphs

• What’s our goal?

Cover Goal

• Minimize area
  \[ \sum_{p} A(p) + \sum_{BB} A_{use}(p \in P) \]

• Minimum added latency
  – Delay of BB covered by p in P

Cover Subgraph

• Given a proposed set of pattern graphs, how can we cover?

Greedy Cover Subgraph

• How might we cover greedily?

• Select most beneficial pattern
• Assign it to the stuff it covers
  – Add logic to share accommodate
  – Remove those as things that need to be covered
• Repeat until all covered or no benefit
Most Beneficial Pattern

- How would we define pattern benefit?

Beneficial Pattern

- Area
  \[ \frac{N \times (\text{mux}(io) + \text{mux}(\text{internal})) + \text{area}(P)}{N \times \text{mux}(io) + \text{area}(P)} \]

- Latency
  \[ \frac{|P|}{\text{latency}(P)} \]

Pattern and Graph Statistics

<table>
<thead>
<tr>
<th>Size</th>
<th>#Subgraph</th>
<th>#Pattern</th>
<th>#Inst</th>
<th>#Calc</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>62</td>
<td>3</td>
<td>62</td>
<td>0.96</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>12</td>
<td>108</td>
<td>1.08</td>
</tr>
<tr>
<td>4</td>
<td>195</td>
<td>20</td>
<td>161</td>
<td>1.48</td>
</tr>
<tr>
<td>5</td>
<td>366</td>
<td>26</td>
<td>248</td>
<td>1.49</td>
</tr>
<tr>
<td>6</td>
<td>701</td>
<td>35</td>
<td>404</td>
<td>1.9</td>
</tr>
<tr>
<td>7</td>
<td>1357</td>
<td>58</td>
<td>579</td>
<td>2.6</td>
</tr>
<tr>
<td>8</td>
<td>2533</td>
<td>76</td>
<td>714</td>
<td>3.18</td>
</tr>
<tr>
<td>9</td>
<td>4517</td>
<td>86</td>
<td>762</td>
<td>3.82</td>
</tr>
<tr>
<td>10</td>
<td>7800</td>
<td>94</td>
<td>793</td>
<td>4.43</td>
</tr>
<tr>
<td>11</td>
<td>13112</td>
<td>101</td>
<td>668</td>
<td>7.04</td>
</tr>
<tr>
<td>12</td>
<td>21365</td>
<td>73</td>
<td>348</td>
<td>7.89</td>
</tr>
<tr>
<td>13</td>
<td>33316</td>
<td>32</td>
<td>87</td>
<td>5.03</td>
</tr>
<tr>
<td>14</td>
<td>49040</td>
<td>3</td>
<td>6</td>
<td>1.7</td>
</tr>
</tbody>
</table>

Big Ideas:

- Sharing
- Estimation
- Techniques
  - Graph Matching
  - Covering
  - Greedy

Admin

- Assignment 5a due Monday
- Reading for Monday online