Overview

• How Cost Scales
• How Performance Scales
• How Performance/Cost Scales
• Recommendations for future work
Cost Model

- Based on Area
  - Included: clusters, internal cluster switch, cluster communication, microcontroller
  - Not included: SRF, memory system, and other interfaces

- Units
  - Roughly in terms of mm$^2$
  - Scaled so that cost of Imagine is 100
Imagine

This is the area considered in the cost model.
Basic Formula

- Cost = NumThreads \times (\text{NumClusters} \times \text{ClusterArea} + \text{ClusterCommSwitchArea} + \text{microControllerArea})
  - \text{ClusterArea} = f(A_F, \text{InternalClusterSwitchSize})
  - 1 \text{ ScratchPad and CommUnit per 5 other ALUs}
Graph of Cost Model

Cost Model

DLP: Num Clusters

ILP: NumALUs per Cluster

- Imagine
Graph of Total ALUs / Cost

Cost Model (TotalALUs/Cost)

DLP: Num Clusters

ILP: NumALUs per Cluster

Imagine
InternalClusterSwitchSize

From *Register Organization for Media Processing*

- SwitchSize = \((p_eN + 2N)(p_eN + N) \times w^2 \times b^2\)
- \(N = \text{Number of functional units}\)
- \(p_e = \frac{1}{4} \quad \text{Number of external ports per ALU}\)
- \(b = 32 \quad \text{Data width}\)
- \(w = \text{wire pitch (typically 0.5 – 2\text{um})}\)
- SwitchSize = \(N^2/100\)
- Switch can overlap part of the ALU area
MicroControllerArea

- MicroControllerArea = MemoryArea + DecoderAreaPerFunctionalUnit * NumFunctionalUnitsPerCluster

- How the MicroController Scales
  - **DLP** -- MicroControllerArea is constant
  - **ILP** -- As instructions get wider we need fewer of them. Memory area remains constant but the decoder area grows.
  - **TLP** -- Each thread requires 1 MicroController
How Performance Scales

- ILP: our cluster configurations
- Kernel and overall performance
- ILP expansion and uc size
- Kernel classification
- We extrapolated to scale DLP, TLP
Our Cluster Configurations

- **Wimp:** 1 ADD, 1 MUL, 1 DIV ... 78
- **Tin:** 3 ADD, 1 MUL, 2 DIV ... 100
- **Gold:** 3 ADD, 2 MUL, 1 DIV ... 100
- **Straw:** 6 ADD, 4 MUL, 2 DIV ... 181
- **Stud:** 12 ADD, 8 MUL, 4 DIV ... 352
Kernel Performance

- Speedup of straw/gold = 1.75x, stud/gold = 2.88x

- Kernel performance/area: straw and gold fare the best
Overall Performance

- Include the StreamC overhead and things are very different...

- Stud and straw speedup now 1.2-1.3x

- Wimp is the performance/area champ

- StreamC pipelining would probably help
What is the cost of ILP?

- Is it uc size?
  Actually, if kernel is well-matched to HW and we don’t unroll excessively, NO.

- But the control part of the microcontroller (25% of area) does grow.
Kernel classification

- In our code, we saw:
  - Wimpy kernels
  - FU-limited kernels
  - Zero-communication kernels

- In provided code, we saw:
  - Comm-limited kernels
  - Bigger kernels
How Performance/Cost Scales

- Estimate for performance based on:
  - Scheduled kernels (uCode file)
  - Number of cycles per kernel invocation
    \((\text{num\_loop\_instr} \times \text{num\_iter}) + \text{num\_non\_loop\_instr}\)
  - Number of kernel invocations
- JPEG: extracted DLP and TLP performance
- Performance/Cost = \(K / (\text{cycles} \times \text{cost})\)
JPEG: DLP Results

JPEG DLP: Performance / Cost

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>M</th>
<th>D</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wimp8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>70</td>
</tr>
<tr>
<td>Tin8</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>Gold8</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Straw8</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>181</td>
</tr>
<tr>
<td>Stud8</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>352</td>
</tr>
</tbody>
</table>
JPEG: TLP Results (1)

- Ran entire JPEG encode process on different 8x8 portions of the image
- Kept total number of clusters constant while adding TEU’s
- Not useful to pipeline our implementation, since DCT dominates
JPEG TLP: performance / cost for gold

Number of TEU's

perf/cost

8 total clusters
16 total clusters
32 total clusters
Cost vs. Number of Clusters

**DLP: cost for gold**

- Cost on the y-axis
- Number of clusters on the x-axis

**JPEG DLP: Execution Time for gold**

- Cycles on the y-axis
- Number of clusters on the x-axis

**JPEG DLP: performance/cost for gold**

- Performance/cost on the y-axis
Recommendations

- Automate *.md file generation
- Have compiler convert DIV to MULT
- Continue kernel classification
- Explore TLP with different execution units
- Collect data on more applications and coding styles
- Scaling model for the SRF
<table>
<thead>
<tr>
<th></th>
<th>perf</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wimp</td>
<td>1.14</td>
<td>.82</td>
</tr>
<tr>
<td>Tin</td>
<td>.86</td>
<td>.54</td>
</tr>
<tr>
<td>Gold</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Straw</td>
<td>.72</td>
<td>.97</td>
</tr>
<tr>
<td>Stud</td>
<td>.38</td>
<td>.81</td>
</tr>
</tbody>
</table>

(overall) (kernel code)
This kernel from mpeg is typical:

<table>
<thead>
<tr>
<th>Name</th>
<th>Cycles (p(1))</th>
<th>uc size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wimp</td>
<td>126</td>
<td>58125</td>
</tr>
<tr>
<td>Tin</td>
<td>52</td>
<td>37284</td>
</tr>
<tr>
<td>Gold</td>
<td>43</td>
<td>34790</td>
</tr>
<tr>
<td>Straw</td>
<td>22</td>
<td>33840</td>
</tr>
<tr>
<td>Stud</td>
<td>16</td>
<td>47418</td>
</tr>
</tbody>
</table>
### Extracting cost/perf

<table>
<thead>
<tr>
<th>kernel</th>
<th>cycles/loop</th>
<th>nonloop cycles</th>
<th>unroll</th>
<th>loop count</th>
<th>cycles/invoke</th>
<th>total cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>gen_idx</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>21</td>
</tr>
<tr>
<td>dct</td>
<td>60</td>
<td>36</td>
<td>2</td>
<td>4</td>
<td>256</td>
<td>276</td>
</tr>
<tr>
<td>quantize</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>61</td>
</tr>
<tr>
<td>rle1</td>
<td>19</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>158</td>
</tr>
<tr>
<td>rle2</td>
<td>14</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>4</td>
<td>117</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>config</th>
<th>cycles</th>
<th>perf/cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>gold8</td>
<td>8</td>
<td>100</td>
</tr>
<tr>
<td>cost</td>
<td>18021</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>kernel</th>
<th>invoke</th>
<th>cycles</th>
<th>cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>gen_idx</td>
<td>1</td>
<td>21</td>
<td>18021</td>
</tr>
<tr>
<td>dct</td>
<td>64</td>
<td>17664</td>
<td>0</td>
</tr>
<tr>
<td>quantize</td>
<td>1</td>
<td>61</td>
<td>0</td>
</tr>
<tr>
<td>rle1</td>
<td>1</td>
<td>158</td>
<td>0</td>
</tr>
<tr>
<td>rle2</td>
<td>1</td>
<td>117</td>
<td>0</td>
</tr>
</tbody>
</table>

---

72084
Estimating Wire Pitch

- 0.18um process
  - Metal pitch ranges from 0.64 – 1.6 um/wire
  - Assume some room for power & ground
  - Also some room to shield for noise
  - 2 um/wire seems reasonable

- Accounting for wire pitch
  - SwitchSize = \(2.81N^2 \times 2^2 \times 32^2 / 10^6 = 0.012N^2\)
  - This formula can be simplified to \(N^2/100\)
ClusterArea

- ALU area and Switch can overlap
  - Introduced an overlap factor $O_F$
  - $O_F$ indicates how wire limited the ALU area is and how much of the switch can overlap this area
  - $O_F$ was set to 0.75

- ClusterArea
  - $\text{ClusterArea} = \text{MAX}(A_F, O_F A_{iCSw}) + (1-O_F) A_{iCSw}$
  - Note that $A_F$ is larger than $O_F A_{iCSw}$ when $N_F < 100$