Multi-Level Cache Hierarchy Evaluation for Programmable Media Processors

Jason Fritts
Assistant Professor
Department of Computer Science
Co-Author: Prof. Wayne Wolf

Overview

- Why Programmable Media Processors?
- Evaluation Environment
- Cache Memory Hierarchy Evaluation
   — preliminary investigation of memory hierarchy for media processing
- Conclusions
- Future Research
Multimedia Applications

- Wide range of applications
  - Communication
    - video conferencing
    - World Wide Web
    - digital/video libraries
    - videophones
  - Entertainment
    - video/computer games
    - movies
    - animation
  - Computer Vision
    - image understanding
    - surveillance
    - tracking
  - Education
    - interactive learning
    - virtual classrooms
  - Art and Architecture

Multimedia is primarily a communication media

Future of Multimedia

Multimedia industry evolves with processor performance.

Multimedia is moving towards advanced representations
Multimedia Processing Solutions

- **Application-specific processors**
  - high performance at low cost
  - very limited flexibility

- **Multimedia extensions to general-purpose processors**
  - good programmability at little added cost
  - some speedup for SIMD parallelism

- **Current “programmable” media processors**
  - good performance
    - specialized hardware
    - subword parallelism
    - ILP
  - good programmability (w/ special programming libraries)
  - moderate frequency

Expectations for Future Media Processors

- Greater Throughput
- Larger On-Chip Memory Hierarchies
- Increased Architecture Regularity
Evaluation Environment

MediaBench Benchmark Suite

- Developed at UCLA

- Excellent combination of applications
  - video: MPEG-2
  - audio: ADPCM coder
  - graphics: Mesa
  - image: JPEG, EPIC, Ghostscript
  - security: PGP, Pegwit
  - speech: GSM, G.721, Rasta

- Augmented for greater representation of future multimedia
  - MPEG-4 object-oriented video
  - H.263 very-low bitrate video
IMPACT Environment

- Aggressive ILP research compiler
  - Three levels of optimizations
    - Classical
    - Superscalar
    - Hyperblock
  - classical optimizations only
  - adds loop unrolling and superblock formation
  - adds hyperblock optimization

- Architecture-independent evaluation
  - large, generic instruction set
  - retargetable back-end

- Performance analysis tools
  - profiling
  - simulation for superscalar and VLIW architectures

Cache Memory Hierarchy Evaluation
Architecture Evaluation

- Variety of Memory Hierarchy Options
  - Cache vs. Memory
  - Automatic Prefetching vs. Software Prefetching
  - Streaming Memory vs. DMA Prefetching
  - Organization of hierarchy?

- Related Work

Base Architecture Model

- Architecture model
  - 8-issue VLIW media processor
  - operation latencies targeting 500 MHz to 1 GHz processor frequency
  - 64 integer and floating-point registers
  - pipeline: 1 fetch, 2 decode, 1 write back, variable execute stages

- L1 Cache
  - 16 KB direct-mapped L1 instruction cache w/ 256 byte lines
  - 32 KB direct-mapped L1 data cache w/ 64 byte lines
    - non-blocking w/ 8-entry miss buffer
    - no-write allocate w/ 8-entry write buffer
  - currently no streaming memory support

- On-Chip L2 Cache
  - 256 KB 4-way set associate w/ 64 byte lines
    - non-blocking w/ 8-entry miss buffer
    - write allocate w/ 8-entry write buffer

- External Memory
  - 4:1 Processor to external bus frequency ratio

<table>
<thead>
<tr>
<th>Cache</th>
<th>Miss Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 I-Cache</td>
<td>20</td>
</tr>
<tr>
<td>L1 D-Cache</td>
<td>15</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>50</td>
</tr>
</tbody>
</table>
L1 Cache

- **Results from earlier workload evaluation:**
  - i-cache working set size: < 8 KB
  - i-cache spatial locality: 84.8% locality within 256 bytes
  - d-cache working set size: < 32 KB
  - d-cache spatial locality: 60.8% locality within 128 bytes


- **No streaming memory support**
  - to be evaluated in future work

L2 Cache Evaluation

- **Cache size**
  - regression over cache sizes from 128 KB to 1 MB
  - base cache size is 256 KB
  - 0.5% avg. performance increase from doubling cache size
    - ~7% difference for unepic and mpeg4dec

- **Access latency**
  - regression over access latencies of 8, 15, 30, 60 cycles
  - base access latency is 15 cycles
  - 5.6% avg. performance decrease from doubling access latency
    - ~35% difference for pegwitdec and pegwiten (large working set size)
    - ~16% difference for mpeg2dec
  - attributable to increasing memory access latency
**L2 Cache**

**Line Size Evaluation**

- **Line size**
  - regression over line sizes from 32 to 512 bytes
  - base line size is 64 bytes
  - 10% avg. performance decrease from doubling line size
    - 1.5-3.5% degradation for speech and security media
    - 32-37% degradation for image, audio, and graphics
  - degradation attributable to increased latency for longer lines

- **Graph** showing instructions per cycle (IPC) vs. L2 Cache Line Size (# bytes)

**External Memory**

**Latency Evaluation**

- **Latency**
  - regression over memory latencies from 25 to 400 bus cycles
  - base line size is 50 bytes
  - 20% avg. performance decrease from doubling memory latency
    - minimal degradation for speech and security media
    - 59-77% degradation for image, audio, and graphics

- **Graph** showing instructions per cycle (IPC) vs. Memory Latency (# cycles to access 64-byte line)
External Memory
Bandwidth Evaluation

- **Bandwidth**
  - regression over system bus width of 4 to 32 bytes
  - base system bus width is 8 bytes
  - 6% avg. performance increase from doubling system bus width
    - 0.6 - 2.7% increase for speech, security, and encoding benchmarks
    - 7.5 - 13.9% increase for decoding and graphics benchmarks

Correlation Between External Memory Latency and Bandwidth Experiments

- **Latency Experiment**
  - increasing memory latency decreases memory bandwidth

- **Bandwidth Experiment**
  - increasing memory bandwidth decreases transfer latency

- **Simultaneously Evaluate Latency and Bandwidth**
  - consider only high bandwidth benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Avg. Latency Degradation (%)</th>
<th>Avg. Bandwidth Degradation (%)</th>
<th>Bandwidth (L, M, H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cjpeg</td>
<td>68.1</td>
<td>11.3</td>
<td>M</td>
</tr>
<tr>
<td>gs</td>
<td>66.8</td>
<td>15.4</td>
<td>M</td>
</tr>
<tr>
<td>gsmencode</td>
<td>3.6</td>
<td>0.4</td>
<td>L</td>
</tr>
<tr>
<td>H264dec</td>
<td>99.1</td>
<td>30.8</td>
<td>H</td>
</tr>
<tr>
<td>mjpegdec</td>
<td>75.6</td>
<td>13.1</td>
<td>H</td>
</tr>
<tr>
<td>mpeg2dec</td>
<td>25.3</td>
<td>2.8</td>
<td>L</td>
</tr>
<tr>
<td>pipewidthdec</td>
<td>25.1</td>
<td>3.0</td>
<td>L</td>
</tr>
<tr>
<td>rawaudio</td>
<td>108.1</td>
<td>22.3</td>
<td>H</td>
</tr>
<tr>
<td>texgen</td>
<td>53.3</td>
<td>6.1</td>
<td>M</td>
</tr>
<tr>
<td>unepc</td>
<td>88.1</td>
<td>21.5</td>
<td>H</td>
</tr>
</tbody>
</table>
Conclusions

- L2 cache has little impact on performance
  - useful for storing state during context switches

- External memory latency => primary memory problem
  - Streaming data structures will help alleviate this

- External memory bandwidth => secondary problem

Future Work

- Multi-Level Prefetch Hierarchy
  - automatic prefetching structures primarily researched at L1-level
  - desire automatic prefetching without saturating bandwidth
  - possible solution:
    - conservative prefetch unit on-chip
    - aggressive prefetch unit off-chip

- Streaming Data Out
  - automated prefetching techniques primarily support streaming data IN
  - examine characteristics of streaming data out
  - modify streaming memory structures to support both input and output
  - example:
    - write buffers already similar to streaming memory buffers for output data
    - modify to predict output stride and fetch (allocate) memory lines as appropriate