Memory Access Optimization for Ultra High Definition Video Decoding

Ping Chao       Youn-Long Lin

Department of Computer Science
National Tsing Hua University
Hsin-Chu, Taiwan 300
ylin@cs.nthu.edu.tw
Agenda

• Introduction
• Memory Access Efficiency Metrics
• Motivational Example
• Elastic Cache with Fast Prefetching
• Optimized Segment Sequence Generation
• Experimental Results
• Conclusion
Agenda

• Introduction
• Memory Access Efficiency Metrics
• Motivational Example
• Elastic Cache with Fast Prefetching
• SSG Optimization
• Experimental Results
• Conclusion
Memory Access Requirement During Video Decoding
Motion Compensation Steps

1. Calculating motion vector
2. Accessing reference frame pixels
3. Interpolation & Weighted prediction
Challenges and Solutions (1/2)

• Access redundancy
  – Straightforward access pattern:
    \{1,\ldots,8\}, \{10,\ldots,17\}, \{20,\ldots,27\}, \{9,\ldots,18\}, \{19,\ldots,28\}, \{29,\ldots,38\}.
    \(#\text{Accesses: 54}\)
  – Solution: **Data reuse (Cache)**
    \{1,\ldots,8\}, \{10,\ldots,17\}, \{20,\ldots,27\}, \{9\}, \{18\}, \{19\}, \{28\}, \{29,\ldots,38\}.
    \(#\text{Accesses: 38}\)
Challenges and Solutions (2/2)

- Inefficient external memory access
  - Cache access pattern:
    \{1,\ldots,8\}, \{10,\ldots,17\}, \{20,\ldots,27\}, \{9\}, \{18\}, \{19\}, \{28\}, \{29,\ldots,38\}.  
    (#DMA transactions: 8)
  - Solution: **Access rearrangement**
    \{1,\ldots,8\}, \{9,\ldots,18\}, \{19,\ldots,28\}, \{29,\ldots,38\}.  
    (#DMA transactions: 4)
 Agenda

- Introduction
- Memory Access Efficiency Metrics
- Motivational Example
- Elastic Cache with Fast Prefetching
- Optimized Segment Sequence Generation
- Experimental Results
- Conclusion
Memory Access Efficiency Metrics (1/4)

• Data reuse efficiency:

\[
\text{Data Reuse Efficiency (DRE)} = \left( \frac{\sum (\text{Effective Data Count})}{\sum (\text{Data Transfer Count})} \right) \times 100\%
\]

• Example:
  - Assume DRE = 100% in the cache approach
  - For Straightforward access:
    DRE = \( \frac{38}{54} \times 100\% = 70\% \)
Memory Access Efficiency Metrics (2/4)

- **Data transfer efficiency** *(can only be measured in actual environment)*:

\[
\text{Data Transfer Efficiency (DTE)} = \frac{\sum (\text{Data Transfer Count})}{\sum (\text{Occupied Cycles})} \times 100\%
\]

- **Estimated data transfer efficiency** *(Assume that each DMA transaction induces nearly-constant latency)*:

\[
\text{EDTE} = \frac{\sum (\text{Data Transfer Count})}{\sum (\text{Data Transfer Count}) + \left( \sum (\text{DMA Operations Count}) \times \frac{\sum (\text{Expected Access Latency})}{\text{Expected Access Latency}} \right)} \times 100\%
\]
Memory Access Efficiency Metrics (3/4)

• Estimated data transfer efficiency
  – Example:
    Assume that expected access latency = 13 cycles*
    • EDTE for cache:
      \[
      \frac{38}{38 + (8 \times 13)} \times 100\% = 26\%
      \]
    • EDTE for rearranged cache access pattern:
      \[
      \frac{38}{38 + (4 \times 13)} \times 100\% = 42\%
      \]

* Expected access latency = 13 cycles
  (Using DDR3, CL + tRP + tRCD = 27, 27/2 = 13 for command rate to data rate translation)
Memory Access Efficiency Metrics (4/4)

- Goal: To Maximize DRE*EDTE

![Graph showing Memory Access Efficiency Metrics with various cache sizes and DRE*EDTE values. The graph illustrates that higher DRE*EDTE values indicate better efficiency.]
Agenda

- Introduction
- Memory Access Efficiency Metrics
- Motivational Example
- Elastic Cache with Fast Prefetching
- Optimized Segment Sequence Generation
- Experimental Results
- Conclusion
Motivational Example (1/3)

• Assumption:
  – Basic access unit is an 8x1-pixel block
  – Each approach comes with its own optimized reference frame layout
Motivational Example (2/3)

a. Straightforward Access

b. Cache (8x4-pixel block cache line)

c. Cache (16x8-pixel block cache line)

d. Rearranged access

MPSOC, June 2010
Motivational Example (3/3)

<table>
<thead>
<tr>
<th></th>
<th># Data Transfers</th>
<th># DMA Operations</th>
<th>DRE</th>
<th>EDTE</th>
<th>DRE* EDTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>233</td>
<td>21</td>
<td>63%</td>
<td>46%</td>
<td>29%</td>
</tr>
<tr>
<td>b.</td>
<td>168</td>
<td>21</td>
<td>88%</td>
<td>38%</td>
<td>33%</td>
</tr>
<tr>
<td>c.</td>
<td>224</td>
<td>10</td>
<td>66%</td>
<td>63%</td>
<td>42%</td>
</tr>
<tr>
<td>d.</td>
<td>164</td>
<td>5</td>
<td>90%</td>
<td>71%</td>
<td>64%</td>
</tr>
</tbody>
</table>

Expected access latency = 13 cycles
(Using DDR3, CL + tRP + tRCD = 27, 27/2 = 13 for command rate to data rate translation)
Agenda

• Introduction
• Memory Access Efficiency Metrics
• Motivational Example
• Elastic Cache with Fast Prefetching
• Optimized Segment Sequence Generation
• Experimental Results
• Conclusion
Proposed Elastic Cache

- Based on Scratch Pad Memory (SPM)
- Come with variable-sized lines, adaptable to fit certain group of data
- Need run-time SPM allocation
  - *Variable Entry-Size Cyclic Queue* allocation
  - No garbage collection
Proposed Elastic Cache (Cont.)

• The most effective way to optimize for rearranged accesses
  – Fully utilized scratch pad memory
  – Small data look-up overhead

• Replacement via *Segment Sequence Generation (SSG)*
  – Allocation order follows the generated sequence
  – Replacement depends on the allocation order
Elastic Cache Data Look-up

Physical Address

<table>
<thead>
<tr>
<th>Physical Starting/Ending Address</th>
<th>SPM Starting Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_A end_A</td>
<td>0x21</td>
</tr>
<tr>
<td>start_B end_B</td>
<td>0x5b</td>
</tr>
<tr>
<td>start_C end_C</td>
<td>0x4b</td>
</tr>
</tbody>
</table>

Start_B 0x5b

Load 0x21 0x4b 0x5b

SPM A B C D E

MPSOC, June 2010

20
Elastic Cache Data Look-up (Cont.)

- Perform only one look-up operation for each group of accesses to the same segment
Opportunity for Access Rearrangement & Prefetching

- Reschedule Motion Vector (MV) producer and consumer, which issues memory accesses

---

<table>
<thead>
<tr>
<th>MV Producer</th>
<th>MV Data</th>
<th>MV Consumer</th>
<th>MV Producer</th>
<th>MV Data</th>
<th>MV Consumer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB1</td>
<td>MB1</td>
<td>MB1</td>
<td>MB1</td>
<td>MB1</td>
<td>MB1</td>
</tr>
<tr>
<td>MB2</td>
<td>MB2</td>
<td>MB2</td>
<td>MB2</td>
<td>MB2</td>
<td>MB2</td>
</tr>
<tr>
<td>MB3</td>
<td>MB3</td>
<td>MB3</td>
<td>MB3</td>
<td>MB3</td>
<td>MB3</td>
</tr>
</tbody>
</table>

MB1～4

MB2～5

MB3～6

Analyze $N$ MBs for access rearrangement, $N = U - 1 = 4$
Illustrative Example

<table>
<thead>
<tr>
<th>MV Producer</th>
<th>MV Data</th>
<th>Prefetch</th>
<th>MV Consumer</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB1</td>
<td>MB1~3</td>
<td>MB1</td>
<td>A</td>
</tr>
<tr>
<td>MB2</td>
<td>MB1~4</td>
<td>MB1</td>
<td>B</td>
</tr>
<tr>
<td>MB3</td>
<td>MB2~5</td>
<td>MB2</td>
<td>C</td>
</tr>
<tr>
<td>MB4</td>
<td>MB3~6</td>
<td>MB3</td>
<td>D</td>
</tr>
<tr>
<td>MB5</td>
<td></td>
<td>MB1</td>
<td></td>
</tr>
<tr>
<td>MB6</td>
<td></td>
<td>MB2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MB3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MB4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MB4</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

- Introduction
- Memory Access Efficiency Metrics
- Motivational Example
- Elastic Cache with Fast Prefetching
  - Optimized Segment Sequence Generation
- Experimental Results
- Conclusion
Optimizing Segment Sequence Generation (SSG)

- **Input**
  - Scattered segments translated from MVs

- **Output**
  - A segment sequence
    - Generated by segment merging
      - Minimize the number of segments and their total area
    - With sequence order optimized for replacement
Segment Merging

- Merge the segments with
  - Overlapping
    - Redundant accesses
  - Narrow gap
    - The gap between two segments ($n$ access units) is smaller than the expected access latency ($m$ cycles), $n < m$

$$\sum \frac{(\text{Effective Data})}{(\text{Data Transfer})} \times \frac{\sum (\text{Data Transfer})}{\sum (\text{Occupied Cycles})} = +n +n -m$$
Two Levels of Segment Merging

- Level 1: Intra-MB Segment Merging
- Level 2: Inter-MB Segment Merging
SSG Block Diagram

Level 1 Segment Merging

Intra-MB Segment Merging

Level 2 Segment Merging

MB-to-Segment link Information:

MB m+2 → MB m+1 → MB m

Inter-MB Segment Merging & Order Adjusting

Transitional Segment Sequence

Segment Shift-out Control

Input

→ : Shifting

→ : Adjusting

Output

MPSOC, June 2010

28
SSG Timing Diagram

Level 1

Level 2:
Merge $Sn$ with $S$

Output segment sub-sequence

$S = \text{Transitional segment sequence}$

$Sn = \text{Sub-sequence of } S \text{ to be used by MB } n$
1. Initial condition:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Seg. MB</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>D(B_t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B'</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>B'</td>
<td>D'</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>54</td>
</tr>
<tr>
<td>C</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>64</td>
</tr>
</tbody>
</table>

Total segment area = 160

- ●: Required Segment
- ○: Latent Segment

SSG Illustrative Example (1/5)

SPM Size Limit M = 92
SSG Illustrative Example (2/5)

### Merge e with B’:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Seg. MB</th>
<th>A</th>
<th>B'</th>
<th>D'</th>
<th>C</th>
<th>e</th>
<th>f</th>
<th>D(B_t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>B_1</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>e</td>
<td>f</td>
<td>64</td>
</tr>
<tr>
<td>B_2</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>B_3</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td></td>
<td>66</td>
</tr>
</tbody>
</table>

- ●: Required Segment
- ○: Latent Segment

**Total segment area = 166**

**SPM Size Limit M = 92**
3. Remove a latent segment from B₃:

Total segment area = 138

SPM Size Limit M = 92
SSG Illustrative Example (4/5)

4. Merge f with D':

- Required Segment
- Latent Segment

Total segment area = 128

SPM Size Limit M = 92

MPSOC, June 2010
SSG Illustrative Example (5/5)

5. Remove all latent segments from $B_3$:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Seg.</th>
<th>MB</th>
<th>B</th>
<th>B</th>
<th>B</th>
<th>D</th>
<th>D($B_1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\Box$</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\Box$</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$90$</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$80$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$60$</td>
</tr>
</tbody>
</table>

Total segment area = $122$

$\Box$: Required Segment
$\circ$: Latent Segment

SPM Size Limit $M = 92$
Opportunity of Prefetching

- If $B_t$ occupies almost the whole SPM, $B_{t+1}$ will have no space for prefetching.
- Check segment sub-sequence of both $B_t$ and $B_{t+1}$
  - Empirical SPM size for high prefetching success rate.

![Graph showing the relationship between SPM size and prefetching success rate.](image)
Agenda

- Introduction
- Memory Access Efficiency Metrics
- Motivational Example
- Elastic Cache with Fast Prefetching
- Optimized Segment Sequence Generation
- Experimental Results
- Conclusion
Experimental Results

• In terms of
  – Computational overhead
  – Performance (Data Transfer & Reuse Efficiency)

• Test video sequences
  – Encoded by JM
  – Resolution: Full HD (1920x1080) and Quad Full HD (3840x2160)
  – Number of pictures: 101 for FHD, 16 for QFHD
  – GOP: IBBBP…
  – Search window size: 128x128 for FHD, 256x256 for QFHD
### Table Look-up Overhead

<table>
<thead>
<tr>
<th>Cache Size</th>
<th># of Look Up Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache 8x2</td>
<td>rushhour: 45, bluesky: 35, pedestrian: 35</td>
</tr>
<tr>
<td>Cache 8x4</td>
<td>rushhour: 40, riverbed: 40, pedestrian: 35</td>
</tr>
<tr>
<td>Cache 8x8</td>
<td>rushhour: 40, riverbed: 40, pedestrian: 35</td>
</tr>
<tr>
<td>Cache 16x8</td>
<td>rushhour: 35, bluesky: 30, pedestrian: 30</td>
</tr>
<tr>
<td>Cache 16x16</td>
<td>rushhour: 30, riverbed: 25, pedestrian: 25</td>
</tr>
<tr>
<td>Cache 32x16</td>
<td>rushhour: 25, bluesky: 20, pedestrian: 20</td>
</tr>
<tr>
<td>Cache 32x32</td>
<td>rushhour: 20, riverbed: 15, pedestrian: 15</td>
</tr>
</tbody>
</table>

- rushhour
- bluesky
- pedestrian
- QFHD_rushhour
- QFHD_pedestrian

![Bar Chart showing look-up overhead for different cache sizes and scenarios.]
Table Look-up Overhead (cont.)

- # Operations per look-up
  - Depends on the number of activated lines / MBs
  - Average number = 2.39
SSG Computation Overhead

- Depends on the length of the transitional segment sequence
- Distribution diagram shows the more MBs buffered the larger the overhead
Performance of the Proposed Elastic Cache

- **DRE*EDTE** under different parameters
  - \( N \): the number of buffered MBs
  - \( M \): the size of the SPM
Comparison with Conventional Cache
Conventional Cache’s Trade-off between DRE and EDTE

![Graph showing the trade-off between DRE and EDTE for different cache configurations and scenarios.](image)

MPSOC, June 2010
Memory Access Efficiency Summary

Higher DRE * EDTE is better
Efficiency of the Proposed SSG

- The proposed fast algorithm does not suffer from performance loss
Agenda

- Introduction
- Memory Access Efficiency Metrics
- Motivational Example
- Elastic Cache with Fast Prefetching
- Optimized Segment Sequence Generation
- Experimental Results
- Conclusion
Conclusion

• Proposed *Elastic Cache* goes beyond the trade-off between DRE and DTE optimizations by a conventional cache

• Elastic cache requires low overhead in data look-up

• Proposed a fast replacement algorithm that produces nearly-optimal solution
Thank You!