Optimizing Discrete Wavelet Transform on the Cell Broadband Engine

Seunghwa Kang  David A. Bader
Key Contributions

- We design an efficient data decomposition scheme to achieve high performance with affordable programming complexity.
- We introduce multiple Cell/B.E. and DWT specific optimization issues and solutions.
- Our implementation achieves 34 and 56 times speedup over one PPE performance, and 4.7 and 3.7 times speedup over the cutting edge multicore processor (AMD Barcelona), for lossless and lossy DWT, respectively.
Presentation Outline

- Discrete Wavelet Transform
- Cell Broadband Engine architecture
  - Comparison with the traditional multicore processor
  - Impact in performance and programmability
- Optimization Strategies
  - Previous work
  - Data decomposition scheme
  - Real number representation
  - Loop interleaving
  - Fine-grain data transfer control
- Performance Evaluation
  - Comparison with the AMD Barcelona
- Conclusions
Presentation Outline

• **Discrete Wavelet Transform**
  • Cell Broadband Engine architecture
    - Comparison with the traditional multicore processor
    - Impact in performance and programmability

• **Optimization Strategies**
  - Previous work
  - Data decomposition scheme
  - Real number representation
  - Loop interleaving
  - Fine-grain data transfer control

• **Performance Evaluation**
  - Comparison with the AMD Barcelona

• **Conclusions**
Discrete Wavelet Transform (in JPEG2000)

- Decompose an image in both vertical and horizontal direction to the sub-bands representing the coarse and detail part while preserving space information
Discrete Wavelet Transform (in JPEG2000)

- Vertical filtering followed by horizontal filtering
- Highly parallel but bandwidth intensive
- Distinct memory access pattern becomes a problem
- Adopt Jasper [Adams2005] as a baseline code
Presentation Outline

• Discrete Wavelet Transform

• Cell Broadband Engine architecture
  - Comparison with the traditional multicore processor
  - Impact in performance and programmability

• Optimization Strategies
  - Previous work
  - Data decomposition scheme
  - Real number representation
  - Loop interleaving
  - Fine-grain data transfer control

• Performance Evaluation
  - Comparison with the AMD Barcelona

• Conclusions
Cell/B.E. vs Traditional Multi-core Processor

- **In-order**
- No dynamic branch prediction
- SIMD only
  => Small and simple core

- **Out-of-order**
- Dynamic branch prediction
- Scalar + SIMD
  => Large and complex core
Cell/B.E. vs Traditional Multi-core Processor

- **Isolated** constant latency LS access
- **Software controlled** DMA data transfer between LS and main memory

- Every memory access is cache coherent
- **Hardware controlled** data transfer
Cell/B.E. Architecture - Performance

- More cores within power and transistor budget
- Invest the larger fraction of the die area for actual computation
- Highly scalable memory architecture
- Enable fine-grain data transfer control
- Efficient vectorization is even more important (No scalar unit)
Cell/B.E. Architecture - Programmability

- Software (mostly programmer up to date) controlled data transfer
- Limited LS size
- Manual vectorization
- Manual branch hint, loop unrolling, etc.
- Efficient DMA data transfer requires cache line alignment and transfer size needs to be a multiple of cache line size.
- Vectorization (SIMD) requires 16 byte alignment and vector size needs to be 16 byte.

=> Challenging to deal with misaligned data !!!
Cell/B.E. Architecture - Programmability

Satisfies alignment and size requirements

```
for( i = 0 ; i < n ; i++ ) {
  a[i] = b[i] + c[i]
}
```

No guarantee in alignment and size

```
v_a = ( vector int* )a;
v_b = ( vector int* )b;
v_c = ( vector int* )c;
for( i = 0 ; i < n_c / 4 ; i++ ) {
  v_a[i] = v_add( v_b[i], v_c[i] )
}
//n_c: a constant multiple of 4
```

```
n_head = ( 16 – ( ( unsigned int )a % 16 ) / 4;
n_head = n_head % 4;
n_body = (n – n_head) / 4;
for( i = 0 ; i < n_head ; i++ ) {
  a[i] = b[i] + c[i];
}
v_a = ( vector int* )( a + n_head );
v_b = ( vector int* )( b + n_head );
v_c = ( vector int* )( c + n_head );
for( i = 0 ; i < n_body ; i++ ) {
  v_a[i] = v_add( v_b[i], v_c[i] );
}
a = ( int* )( v_a + n_body );
b = ( int* )( v_b + n_body );
c = ( int* )( v_c + n_body );
for( i = 0 ; i < n_tail ; i++ ) {
  a[i] = b[i] + c[i];
}
```

=> Even more complex if a, b, and c are misaligned!!!
Presentation Outline

• Discrete Wavelet Transform
• Cell Broadband Engine architecture
  - Comparison with the traditional multicore processor
  - Impact in performance and programmability

• Optimization Strategies
  - Previous work
  - Data Decomposition Scheme
  - Real Number Representation
  - Loop Interleaving
  - Fine-grain Data Transfer Control

• Performance Evaluation
  - Comparison with the AMD Barcelona

• Conclusions
Previous work

• Column grouping [Chaver2002] to enhance cache behavior in vertical filtering

• Muta et al. [Muta2007] optimized convolution based (require up to 2 times more operations than lifting based approach) DWT for Cell/B.E.
  - High single SPE performance
  - Does not scale above 1 SPE
Data Decomposition Scheme

- 2-D array width
  - A multiple of the cache line size
  - Remainder
  - Distributed to the SPEs
  - Processed by the PPE

- 2-D array height
  - A multiple of the cache line size
  - A unit of data transfer and computation
  - A unit of data distribution to the processing elements

- Cache line aligned
Data Decomposition Scheme

- Satisfies the alignment and size requirements for efficient DMA data transfer and vectorization.
- Fixed LS space requirements regardless of an input image size.
- Constant loop count.
Vectorization – Real number representation

- Jasper adopts fixed point representation
  - Replace floating point arithmetic with fixed point arithmetic
  - Not a good choice for Cell/B.E.

<table>
<thead>
<tr>
<th>Inst.</th>
<th>Latency (SPE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpyh $5, $3, $4</td>
<td>7 cycles</td>
</tr>
<tr>
<td>mpyh $2, $4, $3</td>
<td>7 cycles</td>
</tr>
<tr>
<td>mpyu $4, $3, $4</td>
<td>7 cycles</td>
</tr>
<tr>
<td>a $3, $5, $2</td>
<td>2 cycles</td>
</tr>
<tr>
<td>a $3, $3, $4</td>
<td>6 cycles</td>
</tr>
<tr>
<td>fm $3, $3, $4</td>
<td></td>
</tr>
</tbody>
</table>
Loop Interleaving

- In a naïve approach, a single vertical filtering involves 3 or 6 times data transfer
- Bandwidth becomes a bottleneck
- Interleave splitting, lifting, and optional scaling steps

Does not fit into the LS
Loop Interleaving

- First interleave multiple lifting steps
- Then, merge splitting step with the interleaved lifting step

- Use temporary main memory buffer for the upper half
Fine-grain Data Transfer Control

- Initially, we copy data from the buffer after the interleaved loop is finished
- Yet, we can start it just after low2 and high2 are read
- Cell/B.E.’s software controlled DMA data transfer enables this
Presentation Outline

• Discrete Wavelet Transform
• Cell Broadband Engine architecture
  - Comparison with the traditional multicore processor
  - Impact in performance and programmability
• Optimization Strategies
  - Previous work
  - Data decomposition scheme
  - Real number representation
  - Loop interleaving
  - Fine-grain data transfer control
• Performance Evaluation
  - Comparison with the AMD Barcelona
• Conclusions
Performance Evaluation

* 3800 X 2600 color image, 5 resolution levels
* Execution time and scalability up to 2 Cell/B.E. chips (IBM QS20)
Performance Evaluation – Comparison with x86 Architecture

- One 3.2 GHz Cell/B.E. chip (IBM QS20)
- One 2.0 GHz AMD Barcelona chip (AMD Quad-core Opteron 8350)

<table>
<thead>
<tr>
<th>Parallelization</th>
<th>OpenMP based parallelization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vectorization</td>
<td>Auto-vectorization with compiler directives</td>
</tr>
<tr>
<td>Real Number Representation</td>
<td>Identical to the Cell/B.E. case</td>
</tr>
<tr>
<td>Loop Interleaving</td>
<td>Identical to the Cell/B.E. case</td>
</tr>
<tr>
<td>Run-time profile feedback</td>
<td>Compile with run-time profile feedback</td>
</tr>
</tbody>
</table>

* Optimization for the Barcelona
Presentation Outline

- Discrete Wavelet Transform
- Cell Broadband Engine architecture
  - Comparison with the traditional multicore processor
  - Impact in performance and programmability
- Optimization Strategies
  - Previous work
  - Data decomposition scheme
  - Real number representation
  - Loop interleaving
  - Fine-grain data transfer control
- Performance Evaluation
  - Comparison with the AMD Barcelona
- Conclusions
Conclusions

• Cell/B.E. has a great potential to speed-up parallel workloads but requires judicious implementation

• We design an efficient data decomposition scheme to achieve high performance with affordable programming complexity

• Our implementation demonstrates 34 and 56 times speedup over one PPE, and 4.7 and 3.7 times speedup over the AMD Barcelona processor with one Cell/B.E. chip

• Cell/B.E. can also be used as an accelerator in combination with the traditional microprocessor
Acknowledgment of Support

David A. Bader
