Implementation of 2D FFT and Image Filtering on Cell BE.

CSE260
WI 2008
Slavik Bryksin & Tingfan Wu
{vbryksin,t3wu}@cs.ucsd.edu
Fast Image Filtering

- Convolution is computational intensive: \( O( N^2M^2) \)

\[
\text{Gabor Filter} = \text{Conv}(\text{NxN}, \text{MxM})
\]

- Solved by doing FFT: \( O( (N+M) \log(N+M)) \)

\[
= \text{IFFT}(\text{FFT}(\text{NxN}) \cdot \text{FFT}(\text{MxM}))
\]
System Overview

Basic Operations: 2DFFT, Complex-Product, Complex-ABS
About FFT

- Survey of existing cell 2D FFT implementations

<table>
<thead>
<tr>
<th>Implementations</th>
<th>1D</th>
<th>2D</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFTW</td>
<td>Parallel / multi-SPU</td>
<td>PPU only</td>
</tr>
<tr>
<td>FFTC</td>
<td>Parallel / multi-SPU</td>
<td>N/A</td>
</tr>
<tr>
<td>FFT-CellSDK</td>
<td>Serial / SPU or PPU</td>
<td>Serial / SPU only (input 64x64 max)</td>
</tr>
</tbody>
</table>

- 2DFFT = 1D-FFT, Transpose, 1DFFT, Transpose
Generic SPU API Framework

- Remote Procedure Call (RPC) Pattern
- Basic Op: 1DFFT, Transpose, Complex-Product, Complex-ABS

![Diagram of SPU API Framework]

- **PPU**
  - Application: Filter
    1. FFT(img, w, h)
    2. Transpose()

- **SPU (Resident)**
  - SPE_Dispatcher
    1. DMA_in arguments
    2. Manage double buffer
    3. Invoke spu modules
      - Switch(Op){
        - case FFT:
          - Dot Product
          - Transpose
          - FFT
          - Compute and DMAout

- **Image**
- **Overlapped DMAs**
- **Double Buffers**
Divide Independent Jobs

- Parallel Processing
- Image could not fit into a local Store
1D FFT Speedup
Optimization Strategies

- **SIMD**
  - All basic operations are SIMDized
  - Big performance boost 5-15 speedup

- **DMA**
  - Using optimized transfer size
  - Use list-DMA for transpose

- **Double Buffering**
Transpose Mechanism (SIMD)

16bytes

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
</tr>
<tr>
<td>l</td>
<td>J</td>
<td>k</td>
<td>L</td>
</tr>
<tr>
<td>m</td>
<td>n</td>
<td>o</td>
<td>p</td>
</tr>
</tbody>
</table>

ae = spu_shuffle(ab, ef, mask1)
bf = spu_shuffle(ab, ef, mask2)
Transposte Mechanism (DMA List)

- Why DMA list
  - Wrap up 2048 transfer in 1 request
  - Save the overhead to program DMA controller repeatedly.

- DMA list
  - Argument: src/dst base addr, offsets, transfer sizes
  - SIMDized list generation.
Matrix Transpose Timing results
Double buffering

![Diagram of Double Buffer DMA and Single Buffer DMA compared](image)
1D FFT Timing results
SPE DMA Overlap(1SPE), Ticks

Computation Begin  Computation end

DMA Begin  DMA end

DMA Begin  DMA end

Time, ticks

x 10^6
SPE DMA Overlap (16 SPEs), Ticks

Computation Begin

Computation end

DMA Issue

DMA end

Buffer 0

Buffer 1
SPE DMA Overlap (1SPE), Ticks
2D FFT Timing results

Total SPE runtime, usec

- Single buffer, no SIMD
- Double buffer, SIMD
Optimization Story

1. No Opt: DMA+Computation contention
2. Double Buffering: Computation contention
3. SIMD: DMA Contention (again!)
4. Pack basic operations into one DMA transfer.
Final Result

- Implemented 2D FFT and Image Filtering on Cell BE
- Designed SPU RPC Pattern
- Fast: 357 fps on Cell BE, (256x256 image, complex)
  64.9fps on P4 3.2G
  46.7fps on Mac G5
- Generic
  - Image Size: 32x32 to 2048x2048
  - Allow Complex Filters (eg. Gabor Filters)
References:

5. CellSDK Example Library. file:///opt/ibm/cell-sdk/prototype/src/lib/  
7. FFTW http://www.fftw.org/cell/cellblade/  