A Performance Model of Fast 2D-DCT Parallel JPEG Encoding Using CUDA GPU and SMP-Architecture

Mohammed K. Shatnawi
Hussein A. Shatnawi

University of Ottawa
SAU University
Presentation Outline

- Introduction
- Parallel JPEG Implementation
- Performance Evaluation
- Experimental Results
- Conclusion
Introduction

• Multi-core Processor
• NVIDIA video card with GPU
• CUDA
• Main Features of GPU CUDA
• SESC
Introduction

• Comparing JPEG algorithm implementation on SESC with GPU
Parallel JPEG Implementation

• JPEG steps:
  – Convert from RGB to YC CBCR
  – Discrete cosine transform (DCT)
  – Quantization
  – Encoding
Parallel JPEG Implementation

• Running JPEG on CUDA GPU and SESC
  – Small image
  – Big image
Fast 2D-DCT

Parallel block transposes in the FCT
Parallel Model Analysis

• Processing JPEG among available PUs and the main memory
Cross-Architectural Design

- Guarantee of cross-compatibility on GPU and SMD systems.
- Provide cross portability across different hardware
Cross-Architectural Design

- Proposed Algorithm stages for SMP and GPU

Fast DCT
Fast DCT
Fast DCT

Split to blocks

Y Convert RGB to YCbCr
Thread pid=n

Blocks=MxN/64

8x8 blocks

Cb Cr

Y Cr

Thread pid=n

PIDs

0 1 2 NC

Pid*W/NC

W

24bit BMP

24bit JPEG
Performance Evaluation

• Optimized GPU and out-of-order optimized SMP architecture
  – System Specifications
  – Evaluation Metrics
## System Specifications

<table>
<thead>
<tr>
<th></th>
<th><strong>SMD</strong></th>
<th><strong>GPU</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>Symmetric</td>
<td>Symmetric grid</td>
</tr>
<tr>
<td><strong>Technology</strong></td>
<td>70nm</td>
<td>40nm</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>256-bit shared</td>
<td>128-bit global</td>
</tr>
<tr>
<td><strong>Processor clock</strong></td>
<td>5GHz</td>
<td>1.4GHz</td>
</tr>
<tr>
<td><strong>int, fp registers</strong></td>
<td>128-bit</td>
<td>64-bit</td>
</tr>
<tr>
<td><strong>NP, N</strong></td>
<td>[1-32]</td>
<td>96</td>
</tr>
</tbody>
</table>
Evaluation Metrics

- Speedup: the improvement of parallel code in the range of $[0, N]$ or $[0, NP]$
- Efficiency: the average speedup of each PU in range of $[0, 1]$
- Efficiency cutoff point: the highest number of $(N$ or $NP)$ at which $\eta > 50\%$.
- Utilization: the percentage of available utilized resources $U = m_i/NP$ or $U = m_i/NP$
Evaluation Metrics

- Scalability: how $\eta$ maintains constant when input size and (N or NP) increase.

- **Amdahl’s law ($S_{max}$):**
  \[ S_{max} = \frac{1}{f_s + \frac{f_p}{N}} \]

- **Karp-Flatt:** define the sequential fraction $f_s$. The less the value $f_s$ the better the performance. Let $P = N \ or \ NP$, then:
  \[ f_s = \frac{P - S}{S(P - 1)} \]
Experimental Results

The theoretical $f_p$ values for 4 BMP images
Encoding times and $T_{i+1}$ values in SMP and GPU simulation
The efficiency, ECP and the optimal number of processors
Runtime speedup for 4 24-bit BMP images on multiple NP in SMP architecture
## GPU SIMULATION RESULT

<table>
<thead>
<tr>
<th></th>
<th>24-bit BMP images</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>~369KB 512x240</td>
</tr>
<tr>
<td>$T_S$</td>
<td>64</td>
</tr>
<tr>
<td>$T_P$</td>
<td>7.500934</td>
</tr>
<tr>
<td>$T_{i+1}$</td>
<td>32</td>
</tr>
<tr>
<td>$S$</td>
<td>8.53</td>
</tr>
<tr>
<td>$\eta$</td>
<td>8.89%</td>
</tr>
<tr>
<td>$ECP$</td>
<td>1</td>
</tr>
<tr>
<td>$f_p$</td>
<td>89.21%</td>
</tr>
</tbody>
</table>
Conclusion

• Two efficient design for JPEG encoding for 24-bit BMP images were presented
• Significant speedup in both parallel CPU and GPU execution
• CPU performed the proposed JPEG algorithm much better