Algorithm and Architecture Optimizations for Large Size Two Dimensional Discrete Fourier Transform

Overview

Large size 2D Fast Fourier Transform

- Used in image processing, scientific computing
- 2D FFT algorithm is factorization of DFT matrix
  \( y = DFT_m \cdot x \), \( DFT_m = [e^{-2\pi j k/n}]_{k,l} \)
- FFT algorithm is factorization of DFT matrix

- 2D FFT operates on 2D data, e.g. images
- \( DFT_{2 \times 2} = (DFT_2 \otimes I_2) (I_2 \otimes DFT_2) \)

2D-FFT algorithms

- Row column algorithm:
  \( DFT_{m \times m} = (DFT_m \otimes I_m) (I_m \otimes DFT_m) \)

Memory access pattern and achieved bandwidth

- Have large pipelined DRAM access pattern
- Does not exploit DRAM row-buffer locality
- Results in low memory bandwidth utilization
- Memory bandwidth becomes the bottleneck for achieving high performance
- Effective bandwidth orchestration is required for:
  - Performance
  - Bandwidth Efficiency
  - Power Efficiency

1024-by-1024 double precision 2D-FFT

- Raw performance (Gflop/s):
  - Intel Core i7 960 (CPU): 7.6
  - Nvidia GTX 480 (GPU): 82.1
- Performance to bandwidth ratio (Gflop/s):
  - Intel Core i7 960 (CPU): 1.90
  - Nvidia GTX 480 (GPU): 2.76
- Performance to power consumption ratio (Gflop/s):
  - Intel Core i7 960 (CPU): 1.56
  - Nvidia GTX 480 (GPU): 4.83

Background

DFT is matrix-vector multiplication

\[ y = DFT_m \cdot x, \quad DFT_m = \left[ e^{-2\pi j k/n} \right]_{k,l} \]

FFT algorithm is factorization of DFT matrix

\[ DFT_m = (DFT_2 \otimes I_2) (I_2 \otimes DFT_2) \]

2D-FFT operates on 2D data, e.g. images

\[ DFT_{2 \times 2} = (DFT_2 \otimes I_2) (I_2 \otimes DFT_2) \]

Solution: Algorithm and Architecture

Restructured algorithm

- Linear data mapping in DRAM causes row-columnwise access
- Use 2D tiled data mapping where each tile is mapped to a DRAM row
- Restructure the algorithm given 2D data mapping

\[ DFT_{m \times m} = \prod_{l=0}^{m-1} (I_m \otimes DFT_m) (I_m \otimes DFT_m) \]

- Data is accessed as tiles, not row and column-wise
- Row-buffer misses are minimized!

From algorithm to hardware

- Double-buffering:
  - Overlapped computation and I/O
  - All modules keep busy
- Matching throughput to memory bandwidth:
  - Achieved via fine-grained control over datapath parallelism
  - Results in balanced design
- Ensuring continuous dataflow:
  - Buffers are used to smooth the flow of data.

Evaluation

Target application:
- Double-buffering complex 2D-FFT
- Data sizes up to 2,048-by-2,048

Target platforms:

- Core i7 960 (CPU)
- GTX 480 (GPU)
- Altera DE4 (FPGA)
- Intel Core i7 960 (CPU)
- Nvidia GTX 480 (GPU)
- Altera DE4 (FPGA)

<table>
<thead>
<tr>
<th>DRAM Type</th>
<th>DDR3</th>
<th>DDR5</th>
<th>DDR2</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Memory Channels</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Memory BW (GB/s)</td>
<td>25.6</td>
<td>177.4</td>
<td>12</td>
</tr>
<tr>
<td>On-chip Memory (MB)</td>
<td>8</td>
<td>1.69</td>
<td>2.55</td>
</tr>
<tr>
<td>Pinout (I/Os)</td>
<td>3,200</td>
<td>4,491</td>
<td>280</td>
</tr>
<tr>
<td># of Cores</td>
<td>6</td>
<td>480</td>
<td>N/A</td>
</tr>
<tr>
<td>Technology Node (nm)</td>
<td>45</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Application Infrastructure</td>
<td>Spiral</td>
<td>CUDAX 4.0</td>
<td>Spiral/Verilog</td>
</tr>
</tbody>
</table>

Raw performance:

2D-FFT Raw performance (double precision) Performance (Gflop/s) vs. Problem size

Bandwidth Efficiency:

2D-FFT Bandwidth Efficiency (double precision) Bandwidth normalized performance (Gflop/s/Gb/s) vs. Problem size

Power Efficiency:

2D-FFT Power Efficiency (double precision) Power normalized performance (Gflop/s/Watt) vs. Problem size

The authors acknowledge the support of the C252 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.