GPU Accelerated Decoding of High Performance Error Correcting Codes

Andrew D. Copeland, Nicholas B. Chang, and Stephen Leung

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.
• Introduction
  – Wireless Communications
  – Performance vs. Complexity
  – Candidate Implementations

• Low Density Parity Check (LDPC) Codes
• NVIDIA GPU Architecture
• GPU Implementation
• Results
• Summary
Wireless Communications

Generic Communication System

Data → Coding → Modulation → Demodulation → Decoding → Data

- Channel
  - Multipath
  - Noise
  - Interference
Wireless Communications

Generic Communication System

Our Focus

Data → Coding → Modulation → Demodulation → Decoding → Data

- Channel
  - Multipath
  - Noise
  - Interference

MIT Lincoln Laboratory
HPEC_GPU_DECODE-4
ADC 10/1/2009
Wireless Communications

Generic Communication System

Coding ➔ Modulation ➔ Demodulation ➔ Decoding

Channel
• Multipath
• Noise
• Interference

Simple Error Correcting Code

01011 ➔ Coding ➔ 01011
01011 ➔ Coding ➔ 01011
01011 ➔ Coding ➔ 01011
Wireless Communications

Generic Communication System

Data → Coding → Modulation → Demodulation → Decoding → Data

Channel
- Multipath
- Noise
- Interference

Simple Error Correcting Code

Coding 01011
01011 → Coding → 01011
01011

High Performance Codes
- Allow for larger link distance and/or higher data rate
- Superior performance comes with significant increase in complexity

Link Range

Off the shelf (Low Complexity)
Custom (High Complexity)

MIT Lincoln Laboratory

HPEC_GPU_DECODE-6
ADC 10/1/2009
Performance vs. Complexity

- Increased performance requires larger computational cost
- Off the shelf systems require 5dB more power to close link than custom system
- Best custom design only require 0.5dB more power than theoretical minimum
Performance vs. Complexity

- Increased performance requires larger computational cost
- Off the shelf systems require 5dB more power to close link than custom system
- Best custom design only require 0.5dB more power than theoretical minimum
Candidate Implementations

- Half second burst of data
  - Decode complexity 37 TFLOPS
  - Memory transferred during decode 44 TB

<table>
<thead>
<tr>
<th>Implementation Time (beyond development)</th>
<th>PC</th>
<th>9x FPGA</th>
<th>ASIC</th>
<th>4x GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance (decode time)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Candidate Implementations

- Half second burst of data
  - Decode complexity 37 TFLOPS
  - Memory transferred during decode 44 TB

<table>
<thead>
<tr>
<th></th>
<th>PC</th>
<th>9x FPGA</th>
<th>ASIC</th>
<th>4x GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation Time (beyond development)</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance (decode time)</td>
<td>9 days</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Cost</td>
<td>≈ $2,000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Candidate Implementations

- **Half second burst of data**
  - Decode complexity 37 TFLOPS
  - Memory transferred during decode 44 TB

<table>
<thead>
<tr>
<th></th>
<th>PC</th>
<th>9x FPGA</th>
<th>ASIC</th>
<th>4x GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>-</td>
<td>1 person year (est.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (beyond</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>development)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>9 days</td>
<td>5.1 minutes (est.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(decode time)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Cost</td>
<td>≈ $2,000</td>
<td>≈ $90,000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Candidate Implementations

- **Half second burst of data**
  - Decode complexity 37 TFLOPS
  - Memory transferred during decode 44 TB

<table>
<thead>
<tr>
<th>Implementation</th>
<th>PC</th>
<th>9x FPGA</th>
<th>ASIC</th>
<th>4x GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (beyond development)</td>
<td>-</td>
<td>1 person year (est.)</td>
<td>1.5 person years (est.) + 1 year to Fab.</td>
<td></td>
</tr>
<tr>
<td>Performance (decode time)</td>
<td>9 days</td>
<td>5.1 minutes (est.)</td>
<td>0.5s (if possible)</td>
<td></td>
</tr>
<tr>
<td>Hardware Cost</td>
<td>$2,000</td>
<td>$90,000</td>
<td>$1 million</td>
<td></td>
</tr>
</tbody>
</table>
Candidate Implementations

- Half second burst of data
  - Decode complexity 37 TFLOPS
  - Memory transferred during decode 44 TB

<table>
<thead>
<tr>
<th></th>
<th>PC</th>
<th>9x FPGA</th>
<th>ASIC</th>
<th>4x GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation Time</td>
<td>-</td>
<td>1 person year (est.)</td>
<td>1.5 person years (est.) +1 year to Fab.</td>
<td>?</td>
</tr>
<tr>
<td>(beyond development)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance (decode time)</td>
<td>9 days</td>
<td>5.1 minutes (est.)</td>
<td>0.5s (if possible)</td>
<td>?</td>
</tr>
<tr>
<td>Hardware Cost</td>
<td>≈ $2,000</td>
<td>≈ $90,000</td>
<td>≈ $1 million</td>
<td>≈ $5,000 (includes PC cost)</td>
</tr>
</tbody>
</table>
• Introduction

• **Low Density Parity Check (LDPC) Codes**
  – Decoding LDPC GF(256) Codes
  – Algorithmic Demands of LDPC Decoder

• NVIDIA GPU Architecture
• GPU Implementation
• Results
• Summary
Low Density Parity Check (LDPC) Codes

LDPC Codes

- Proposed by Gallager (1962)
  - codes were binary
- Not used until mid 1990s
- GF(256) Davey and McKay (1998)
  - symbols in [0, 1, 2, …, 255]
- Parity check matrix defines graph
  - Symbols connected to checks nodes
- Has error state feedback
  - Syndrome = 0 (no errors)
  - Potential to re-transmit

LDPC GF(256) Parity-Check Matrix

\[
N_{\text{Checks}} \times N_{\text{Symbols}}
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 52 & 0 & 174 & 0 & 0 & 0 & 50 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 213 & 0 & 0 & 0 & 0 & 97 & 0 \\
0 & 0 & 119 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 78 & 0 & 0 & 0 & 0 & 218 \\
0 & 0 & 107 & 0 & 0 & 0 & 0 & 129 & 0 & 0 & 174 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 172 & 214 & 0 & 0 & 0 & 0 & 78 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 216 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 39 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 49 & 0 & 139 & 0 & 0 & 0 & 0 & 0 \\
191 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 152 \\
0 & 0 & 0 & 134 & 0 & 0 & 97 & 0 & 0 & 0 & 0 & 0 & 0 & 220 & 0 & 0 \\
0 & 0 & 238 & 0 & 0 & 0 & 0 & 110 & 0 & 0 & 0 & 178 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
C_1 \\
\vdots \\
C_n \\
\vdots \\
C_{15}
\end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}
\]

Syndrome

Codeword

Tanner Graph

\[\text{Tanner Graph} - \text{Syndrome} = 0 (\text{no errors}) - \text{Potential to re-transmit}\]
Decoding LDPC GF(256) Codes

- Solved using LDPC Sum-Product Decoding Algorithm
  - A belief propagation algorithm
- Iterative Algorithm
  - Number of iterations depends on SNR

Tanner Graph

- ML Probabilities
- Decoded symbols
- Syndrome
- Codeword

GFD GF(256) Codes

- Solved using LDPC Sum-Product Decoding Algorithm
  - A belief propagation algorithm
- Iterative Algorithm
  - Number of iterations depends on SNR
Algorithmic Demands of LDPC Decoder

Computational Complexity

• Decoding of a entire sequence 37 TFLOPS
  – Single frame 922 MFLOPS
  – Check update 95% of total

Memory Requirements

• Data moved between device memory and multiprocessors 44.2 TB
  – Single codeword 1.1 GB
• Data moved within Walsh Transform and permutations (part of check update) 177 TB
  – Single codeword 4.4 GB

Decoder requires architectures with high computational and memory bandwidths
Outline

• Introduction
• Low Density Parity Check (LDPC) Codes
  • NVIDIA GPU Architecture
    – Dispatching Parallel Jobs
• GPU Implementation
• Results
• Summary
NVIDIA GPU Architecture

- GPU contains N multiprocessors
  - Each Performs Single Instruction Multiple Thread (SIMT) operations
- Constant and Texture caches speed up certain memory access patterns
- NVIDIA GTX 295 (2 GPUs)
  - N = 30 multiprocessors
  - M = 8 scalar processors
  - Shared memory 16KB
  - Device memory 896MB
  - Up to 512 threads on each multiprocessor
  - Capable of 930 GFLOPs
  - Capable of 112 GB/sec
Dispatching Parallel Jobs

• Program decomposed into blocks of threads
  – Thread blocks are executed on individual multiprocessors
  – Threads within blocks coordinate through shared memory
  – Threads optimized for coalesced memory access

Grid

• Calling parallel jobs (kernels) within C function is simple
• Scheduling is transparent and managed by GPU and API

```
grid.x = M; threads.x = N;
kernel_call<<<grid, threads>>>(variables);
```

Thread blocks execute independently of one another
Threads allow faster computation and better use of memory bandwidth
Threads work together using shared memory
Outline

• Introduction
• Low Density Parity Check (LDPC) Codes
• NVIDIA GPU Architecture
  • GPU Implementation
    – Check Update Implementation
• Results
• Summary
GPU Implementation of LDPC Decoder

• Quad GPU Workstation
  – 2 NVIDIA GTX 295s
  – 3.20GHz Intel Core i7 965
  – 12GB of 1330MHz DDR3 memory
  – Costs $5000

• Software written using NVIDIA’s Compute Unified Device Architecture (CUDA)
• Replaced Matlab functions with CUDA MEX functions
• Utilized static variables
  – GPU Memory allocated first time function is called
  – Constants used to decode many codewords only loaded once
• Codewords distributed across the 4 GPUs
• Thread blocks assigned to each check independently of all other checks
• $Q_{ij}$ and $R_{ij}$ are 256 element vectors
  – Each thread is assigned to element of $\text{GF}(256)$
  – Coalesced memory reads
  – Threads coordinate using shared memory

```python
grid.x = num_checks;
threads.x = 256;
check_update<<<grid, threads>>>(Q, R);
```

Structure of algorithm exploitable on GPU
Outline

• Introduction
• Low Density Parity Check (LDPC) Codes
• NVIDIA GPU Architecture
• GPU Implementation

• Results
• Summary
## Candidate Implementations

<table>
<thead>
<tr>
<th></th>
<th>PC</th>
<th>9x FPGA</th>
<th>ASIC</th>
<th>4x GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Implementation</strong></td>
<td>-</td>
<td>1 person</td>
<td>1.5 person</td>
<td>?</td>
</tr>
<tr>
<td>Time (beyond</td>
<td>year (est.)</td>
<td>years (est.)</td>
<td>+1 year to</td>
<td>?</td>
</tr>
<tr>
<td>development)</td>
<td></td>
<td>+1 year to</td>
<td>Fab.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>(time to decode half</td>
<td></td>
<td></td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>second of data)</td>
<td>9 days</td>
<td>5.1 minutes</td>
<td>0.5s</td>
<td>?</td>
</tr>
<tr>
<td>(if possible)</td>
<td></td>
<td></td>
<td>(if possible)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hardware Cost</strong></td>
<td>≈ $2,000</td>
<td>≈ $90,000</td>
<td>≈ $1 million</td>
<td>≈ $5,000</td>
</tr>
<tr>
<td>(includes PC cost)</td>
<td></td>
<td></td>
<td></td>
<td>(includes PC cost)</td>
</tr>
</tbody>
</table>
## Candidate Implementations

<table>
<thead>
<tr>
<th></th>
<th>PC</th>
<th>9x FPGA</th>
<th>ASIC</th>
<th>4x GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>-</td>
<td>1 person year (est.)</td>
<td>1.5 person years (est.)</td>
<td>6 person weeks</td>
</tr>
<tr>
<td>Time (beyond</td>
<td></td>
<td></td>
<td>+1 year to Fab.</td>
<td></td>
</tr>
<tr>
<td>development)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>9 days</td>
<td>5.1 minutes</td>
<td>0.5s (if possible)</td>
<td>5.6 minutes</td>
</tr>
<tr>
<td>(time to decode half</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>second of data)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Cost</td>
<td>≈ $2,000</td>
<td>≈ $90,000</td>
<td>≈ $1 million</td>
<td>≈ $5,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(includes PC cost)</td>
</tr>
</tbody>
</table>
Implementation Results

- Decoded test data in 5.6 minutes instead of 9 days
  - A 2336x speedup

GPU implementation supports analysis, testing, and demonstration activities
Implementation Results

- Decoded test data in 5.6 minutes instead of 9 days
  - A 2336x speedup
- Hard coded version runs fixed number of iterations
  - Likely used on FPGA or ASIC versions

GPU implementation supports analysis, testing, and demonstration activities

Decode Time

- GPU
- Matlab
- Hard Coded

SNR (dB)

Decode Time (Minutes)
Summary

• GPU architecture can provide significant performance improvement with limited hardware cost and implementation effort

• GPU implementation is a good fit for algorithms with parallel steps that require systems with large computational and memory bandwidths
  – LDPC GF(256) decoding is a well-suited algorithm

• Implementation allows reasonable algorithm development cycle
  – New algorithms can be tested and demonstrated in minutes instead of days
Experimental Setup

Test System Parameters

• Data rate 1.3 Gb/s
• Half second burst
  – 40,000 codewords (6000 symbols)
• Parity check matrix 4000 checks × 6000 symbols
  – Corresponds to rate 1/3 code
• Performance numbers based on 15 iterations of decoding algorithm