Parallelization Techniques for Implementing Trellis Algorithms on Graphics Processors

Qi Zheng*, Yajing Chen*, Ronald Dreslinski*, Chaitali Chakrabarti†, Achilleas Anastasopoulos*, Scott Mahlke*, Trevor Mudge*

*University of Michigan, Ann Arbor
†Arizona State University, Tempe

ISCAS 13’
May 21, 2013
Trellis algorithm

- Trellis is widely used in coding theory
  - Progression of symbols within a code
  - Representation of the state transitions of a finite state machine
Treillis algorithm

- Treillis is widely used in coding theory
  - Progression of symbols within a code
  - Representation of the state transitions of a finite state machine
- Treillis algorithm
  - The processing described by the value propagation in a trellis

![Trellis Diagram]

input=0
input=1

stage k-1  |  stage k  |  stage k+1
state 0    |  state 1  |  state 0
state 1    |  state 2  |  state 1
state 2    |  state 3  |  state 2
state 3    |  state 4  |  state 3
state 4    |  state 5  |  state 4
state 5    |  state 6  |  state 5
state 6    |  state 7  |  state 6
state 7    |           |  state 7
Trellis algorithm

- Broad scope of uses
  - Viterbi/BCJR/Baum-Welch
  - Communication system/Data compression/Speech recognition
  - Play important roles

Runtime breakdown of the baseband in LTE uplink

- Turbo decoder: 71%
- Others: 29%
GPU—Graphics processing unit

- High Throughput:
  - GFLOPS/TFLOPS-level peak throughput

- High Efficiency

<table>
<thead>
<tr>
<th>Processor</th>
<th>GFLOP/dollar</th>
<th>GFLOP/watt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nvidia GeForce GTX680</td>
<td>6.192</td>
<td>15.848</td>
</tr>
<tr>
<td>Intel Xeon E7-8837</td>
<td>0.037</td>
<td>0.656</td>
</tr>
<tr>
<td>Intel Itanium 8350</td>
<td>0.007</td>
<td>0.150</td>
</tr>
</tbody>
</table>

- Programming Support
  - OpenCL
  - CUDA
GPU Performance Challenge

- Sources of GPU underutilization
  - Thread inadequacy
  - Pipeline stall

- Thread inadequacy
  - ≥ 1000 cores on a commercial GPU

- Pipeline stall
  - Long memory access latency
    - L2 cache/External memory
  - Using multithreading to hide pipeline stall
Contribution of the paper

- Previous work
  - Mapped the Turbo decoder on a GPU
  - Study the throughput and BER of the implementation

- Our work
  - Generalize the parallelization schemes to the implementation of trellis algorithms on a GPU
  - Explore additional schemes not in previous works: forward-backward and branch-metric parallelism
  - Study the implementation tradeoffs between throughput, processing latency and BER
  - Show different combinations of parallelization schemes for different system requirements
Outline

- Motivation and background
  - Trellis algorithm
  - GPU
- Parallelization schemes
  - Packet-level
  - Subblock-level
  - Trellis-level
- Implementation tradeoffs
- Conclusion
Packet-level Parallelism

- Process multiple packets
- \#Threads = \#packets
- Long processing latency
  - Especially for the 1st packet
Subblock-level Parallelism

- #Threads = #subblocks
- Increases the output error rate
- Recovery scheme to fix performance loss
  - Training sequence (TS)
  - Next iteration initialization (NII)
  - Need additional computations
Trellis-level Parallelism

- State-level parallelism
  - \#Threads = \#states of a stage
Trellis-level Parallelism

- State-level parallelism
- Branch-metric parallelization
  - \#Threads = \#branches

```
thread 0
thread 2

\vdots

thread 1
thread 3

\vdots

thread 14
thread 15

stage k
stage k+1
```
Trellis-level Parallelism

- State-level parallelism (SL)
- Branch-metric parallelization (BM)
- Forward-backward parallelization (FB)
  - #Threads = 2
## Summary

- **Total number of threads**
  \[
  N_{\text{thread}} = N_{\text{packet}} \cdot N_{\text{subblock}} \cdot T_{\text{trellis}}
  \]

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Throughput</th>
<th>Latency</th>
<th>Bit Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet-level</td>
<td>Better</td>
<td>Worse</td>
<td>No Change</td>
</tr>
<tr>
<td>Subblock-level</td>
<td>Better</td>
<td>No Change</td>
<td>Worse</td>
</tr>
<tr>
<td>Trellis-level</td>
<td>Better</td>
<td>No Change</td>
<td>No Change</td>
</tr>
<tr>
<td>Subblock+NII</td>
<td>Worse</td>
<td>No Change</td>
<td>Better</td>
</tr>
<tr>
<td>Subblock+TS</td>
<td>Worse</td>
<td>No Change</td>
<td>Better</td>
</tr>
</tbody>
</table>
Outline

- Motivation and background
  - Trellis algorithm
  - GPU

- Parallelization schemes
  - Packet-level
  - Subblock-level
  - Trellis-level

- Implementation tradeoffs

- Conclusion
Experiment setup

- Nvidia GeForce GTX470
  - 14 Streaming Multiprocessors (SM)
  - 448 Streaming Processors (SP)
  - 64KB L1 cache + shared memory per SM
  - 768KB L2 cache per GPU
  - 2GB DRAM

- LTE Turbo decoder
  - Codeword size: 6144
  - Code rate: 1/3
  - Iteration num: 5
Throughput vs. Latency

- More packets → higher throughput, and longer latency.
- Trellis-level parallelism improves throughput without affecting the latency.

- \#subblock = 1
- BER ≤ 10^{-5} with SNR = 1dB

Higher is better
Throughput vs. BER

- SNR requirement presented are the lowest values to achieve $10^{-5}$ BER.
- One packet

- More subblocks $\rightarrow$ higher throughput, but higher SNR requirement.
- Longer TS $\rightarrow$ lower SNR requirement, but lower throughput.
- NII+TS-4 achieves the best tradeoff.
## Implementation tradeoff

<table>
<thead>
<tr>
<th>Parallelization Schemes</th>
<th>TH* (Mbps)</th>
<th>WPL* (ms)</th>
<th>SNR* (dB)</th>
<th>BER*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trellis-level</td>
<td>Subblock Num</td>
<td>Packet Num</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>512</td>
<td>1</td>
<td>4.26</td>
<td>1.44</td>
</tr>
<tr>
<td>SL^+</td>
<td>512</td>
<td>1</td>
<td>20.49</td>
<td>0.55</td>
</tr>
<tr>
<td>SL</td>
<td>256</td>
<td>2</td>
<td>21.09</td>
<td>1.07</td>
</tr>
<tr>
<td>SL,FB^+</td>
<td>256</td>
<td>1</td>
<td>19.65</td>
<td>0.56</td>
</tr>
<tr>
<td>SL,FB</td>
<td>128</td>
<td>10</td>
<td>29.00</td>
<td>4.58</td>
</tr>
</tbody>
</table>

*SL = State-level parallelism, FB = Forward-backward parallelism
*TH = Throughput, WPL = Worst-case Packet Latency
*SNR requirement presented are the lowest values to achieve 10^{-5} BER.
*BER is the bit error rate when SNR = 1 dB.

Different combinations of parallelization schemes can satisfy different system requirements.
## Implementation tradeoff

<table>
<thead>
<tr>
<th>Parallelization Schemes</th>
<th>Subblock Num</th>
<th>Packet Num</th>
<th>TH* (Mbps)</th>
<th>WPL* (ms)</th>
<th>SNR* (dB)</th>
<th>BER*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trellis-level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>512</td>
<td>1</td>
<td>4.26</td>
<td>1.44</td>
<td>1.7</td>
<td>1.6x10^{-3}</td>
</tr>
<tr>
<td>SL^+</td>
<td>512</td>
<td>1</td>
<td>20.49</td>
<td>0.55</td>
<td>1.7</td>
<td>1.6x10^{-3}</td>
</tr>
<tr>
<td>SL</td>
<td>256</td>
<td>2</td>
<td>21.09</td>
<td>1.07</td>
<td>1.3</td>
<td>4.1x10^{-4}</td>
</tr>
<tr>
<td>SL,FB^+</td>
<td>256</td>
<td>1</td>
<td>19.65</td>
<td>0.56</td>
<td>1.3</td>
<td>4.1x10^{-4}</td>
</tr>
<tr>
<td>SL,FB</td>
<td>128</td>
<td>10</td>
<td>29.00</td>
<td>4.58</td>
<td>1.1</td>
<td>2.0x10^{-4}</td>
</tr>
</tbody>
</table>

*SL = State-level parallelism, FB = Forward-backward parallelism
*TH = Throughput, WPL = Worst-case Packet Latency
*SNR requirement presented are the lowest values to achieve $10^{-5}$ BER.
*BER is the bit error rate when SNR = 1 dB.

- Different combinations of parallelization schemes can satisfy different system requirements.
### Implementation tradeoff

<table>
<thead>
<tr>
<th>Parallelization Schemes</th>
<th>TH* (Mbps)</th>
<th>WPL* (ms)</th>
<th>SNR* (dB)</th>
<th>BER*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trellis-level</td>
<td>Subblock Num</td>
<td>Packet Num</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>512</td>
<td>1</td>
<td>4.26</td>
<td>1.44</td>
</tr>
<tr>
<td>SL^+</td>
<td>512</td>
<td>1</td>
<td>20.49</td>
<td>0.55</td>
</tr>
<tr>
<td>SL</td>
<td>256</td>
<td>2</td>
<td>21.09</td>
<td>1.07</td>
</tr>
<tr>
<td>SL,FB^+</td>
<td>256</td>
<td>1</td>
<td>19.65</td>
<td>0.56</td>
</tr>
<tr>
<td>SL,FB</td>
<td>128</td>
<td>10</td>
<td>29.00</td>
<td>4.58</td>
</tr>
</tbody>
</table>

*SL = State-level parallelism, FB = Forward-backward parallelism

*TH = Throughput, WPL = Worst-case Packet Latency

*SNR requirement presented are the lowest values to achieve $10^{-5}$ BER.

*BER is the bit error rate when SNR = 1 dB.

- Different combinations of parallelization schemes can satisfy different system requirements
Outline

- Motivation and background
  - Trellis algorithm
  - GPU
- Parallelization schemes
  - Packet-level
  - Subblock-level
  - Trellis-level
- Implementation tradeoffs
- Conclusion
Conclusion

- Implement different parallelization schemes of trellis algorithms on a GPU

- Discuss the implementation tradeoffs between throughput, processing latency and BER

- Different combinations of parallelization schemes can satisfy different system requirements
Thanks!

Any questions?
Backup
Next iteration initiation

Sub-block 0 \[\alpha_0\] \[\beta_1\] Sub-block 1 \[\alpha_1\] \[\beta_2\] \ldots \ldots \beta_{p-1} \[\alpha_{p-2}\] Sub-block P-1
Turbo decoder performance on GPGPU

- GPGPU Utilization

\[ N_{thread} = N_{codeword} \cdot N_{sub\text{-}block} \cdot \text{Thread}_{sub\text{-}block} \]

- Throughput

\[ THR_{Dec} = \frac{N_{codeword}}{T_{decoding}} \]

- Decoding latency

\[ t = t_{buf} + t_{decode} = \frac{N_{codeword} \cdot R \cdot K}{THR_{phy}} + K \cdot T_{decoding} \]
Treillis algorithm

- Example—Turbo codes in LTE

In this work, we study different parallelization techniques for implementing trellis algorithms on Graphics Processors (GPUs). We focus on the trade-offs between throughput, latency, and bit error rate.

The trellis is a graph representation of the state transitions of a finite state machine. Each column represents a unit of the trellis, and each node represents a possible state. The states are connected by branches, which represent possible transitions depending on the input to the system.

Example—Turbo codes in LTE

- We consider parallelization schemes at the packet-level, subblock-level, and trellis-level to increase the number of threads in a system.
- Trade-offs between throughput, latency, and bit error rate are evaluated.
- Turbo decoder is implemented on an NVIDIA GTX470 GPU.
- At the trellis-level, we consider state-level, forward-direction, and reverse-direction parallelism.
- Implementations of trellis-based algorithms have been proposed for GPUs, which provide high computational dependencies per dollar.
- In spite of their raw compute power, GPUs are very attractive for their programmability and throughput.

We consider parallelization schemes at the packet-level, subblock-level, and trellis-level to increase the number of threads in a system. These algorithms are used in many systems such as speech, signal processing, and error correction coding.

Figure 28 depicts the progression of symbols within a code. There are two main branches: forward and backward.

In the forward directions, the forward metric of a state is computed as the sum of the forward metric of states and the branch metric corresponding to the transition.

In the backward directions, the backward metric of a state is computed as the sum of the backward metric of states and the branch metric corresponding to the transition.

Example:

- For input = 0, the forward metric of state 0 is computed as the sum of the forward metric of states 1 and 2.
- For input = 1, the forward metric of state 0 is computed as the sum of the forward metric of states 3 and 4.

Result:

- For input = 0, the forward metric of state 0 is computed as the sum of the forward metric of states 1 and 2.
- For input = 1, the forward metric of state 0 is computed as the sum of the forward metric of states 3 and 4.

The final result is the computation of the forward and backward metrics to obtain the most likely path through the trellis.
Subblock-level Parallelism

- Increases the output error rate
- Recovery scheme to fix performance loss
  - Training sequence (TS)
  - Next iteration initialization (NII)
  - Additional computation