Åbo Akademi University

An Efficient GPU-based LDPC Decoder for Long Codewords

Stefan Grönroos
Kristian Nybom
Jerker Björkqvist

Åbo Akademi University

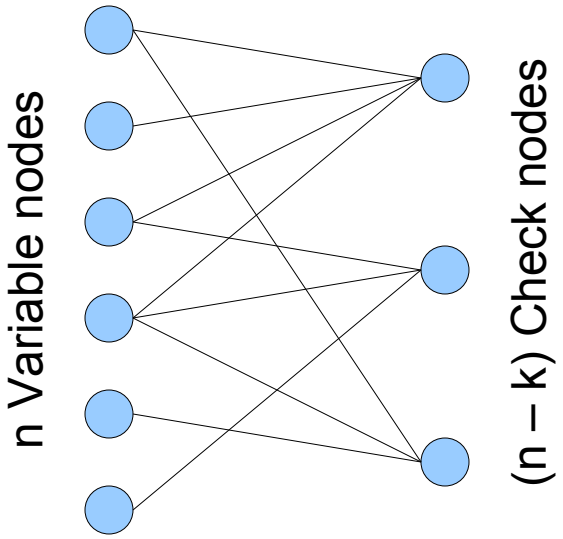
Background

- Working on software real-time DVB-T2 implementation for general purpose computers
- DVB-T2, DVB-C2, DVB-S2 standards use LDPC codes as part of FEC scheme
  - Very long codewords: 16200 or 64800 bits
  - One of the most complex operations in the signal processing chain
  - DVB-T2 requires up to ~61 Mbps decoder throughput
- Our CPU implementation not even close to realtime capable
- Thus we turned to GPUs
  - More specifically NVIDIAs CUDA framework

LDPC Decoding

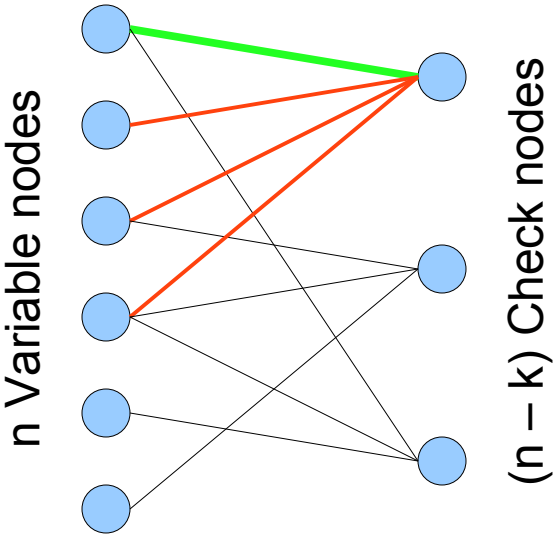
- LDPC Code can be described by:
  - **H** matrix
  - Corresponding bipartite graph
- n-bit codeword
  - k data bits
  - (n-k) parity bits

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

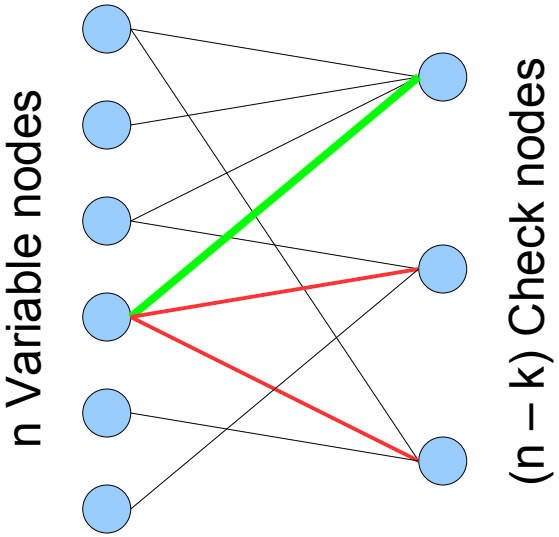


Iterative message passing

- Each edge in graph holds message between check- and variable nodes
- Check node update:

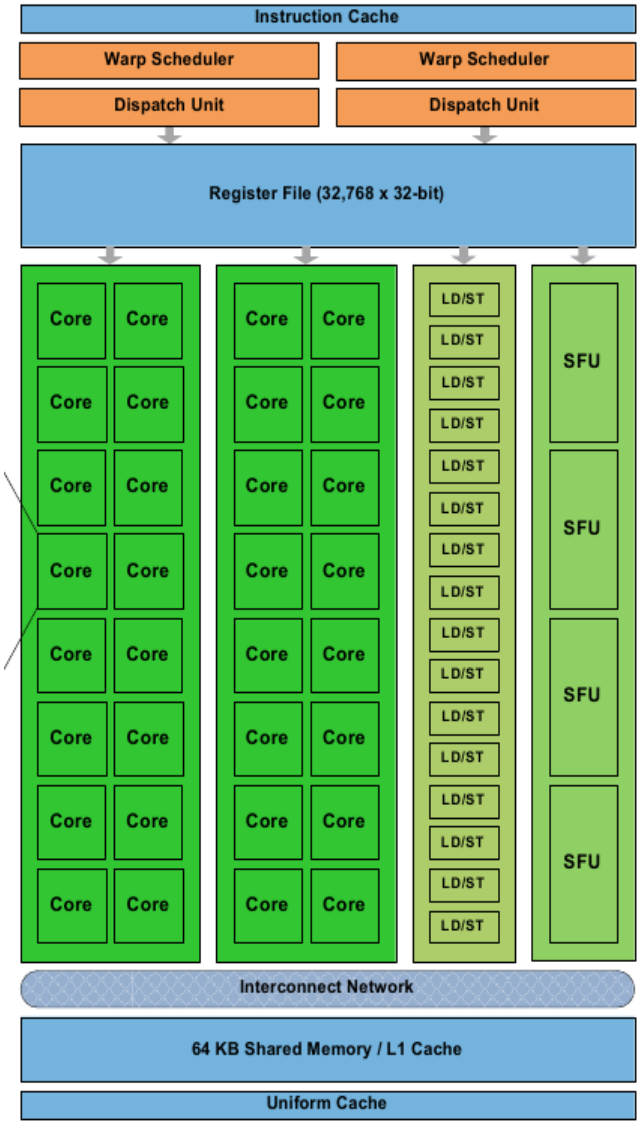


- Variable node update:



Hardware Setup

- NVIDIA GeForce GTX 570
- Based on NVIDIA Fermi architecture
- 15 Streaming Multiprocessors
  - 32 cores per SM
- Thread *warp*:
  - Group of 32 consecutive threads
  - The same instruction is run for a half-warp (16 threads) at a time on 16 cores of an SM



Fermi Streaming Multiprocessor (SM)
Source: NVIDIA

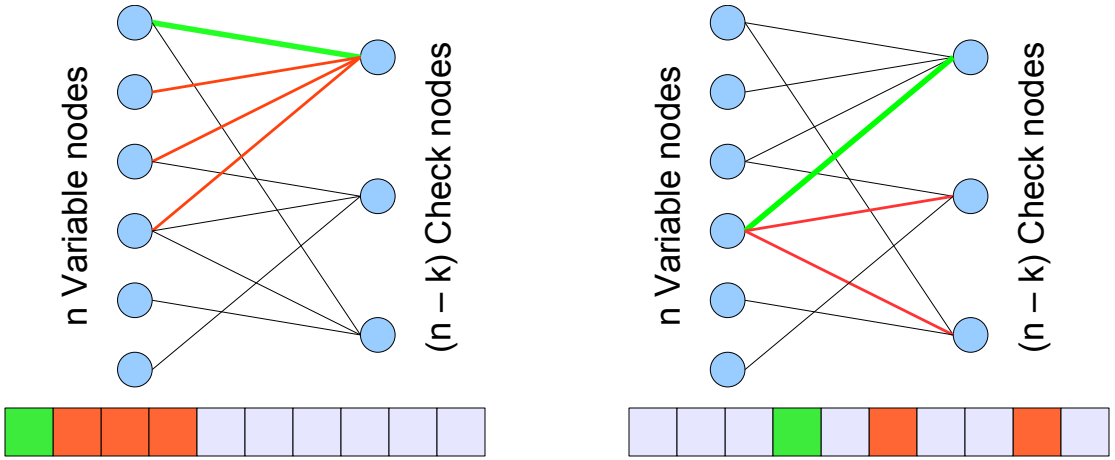
Åbo Akademi University

GPU Memory Accesses

- Access to the large global memory is very slow on the GPU
- Global memory accesses are processed per warp (32 threads)
- If the threads of a warp access 32 aligned consecutive 32-byte words, we get full memory *coalescence*
  - Only one memory request for 128 bytes is made, and memory bus is fully utilized
  - Very low bus utilization if memory accesses are scattered within a warp!

Decoder memory accesses

- If we decode one codeword at a time:
  - Either check node update or variable node update memory accesses scattered
- Solution: Decode several codewords in parallel
  - Efficient memory accesses
  - Increases parallelism



Åbo Akademi University

Our LDPC Decoder approach

- Two main kernels (functions). Iterated alternately.
  - Check node update
  - Variable node update
- 8-bit fixed-point representation for messages
  - Messages for same edge for all codewords stored consecutively in memory
- We decode 128 codewords in parallel
- Each thread updates the outgoing messages from one check/variable node for 4 different codewords
  - A warp processes the same updates for all 128 codewords (32 threads x 4 codewords).
  - Result: 128-byte message reads/writes to global memory

Performance

- Good memory access patterns
  - Solution is now instruction bound
- No shared (”scratchpad”) memory used, just 48KB L1 cache.
  - Allows larger number of active threads
- Throughput:
  - Codeword length: 64800 bits
  - Code rate ½ (32400 information bits, 32400 parity bits)

20 iterations	30 iterations	50 iterations
163 Mbps	112 Mbps	69 Mbps

Åbo Akademi University

Conclusions

- Real-time LDPC decoding for DVB-T2, DVB-S2, DVB-C2 possible on a modern GPU
- Some capacity left on GPU for other complex tasks, such as QAM constellation demapper
  - Future work

Åbo Akademi University

Thank you for listening!

Questions?