A Bit-Serial Approximate Min-Sum LDPC Decoder

and FPGA Implementation

Ahmad Darabiha, Anthony Chan Carusone, Frank R. Kschischang

University of Toronto, Toronto, Canada
ISCAS, Kos, Greece
May 2006
Outline

• Introduction to LDPC codes/decoders
• Proposed techniques
  • Bit-serial message passing
  • Approximate Min-Sum decoding
• FPGA implementation
• Conclusion
Introduction

• Low-Density Parity-Check (LDPC) codes
  • A sub-class of Error Control Codes (ECC)
  • Perform better than Turbo and Reed-Solomon codes
  • Approach Shannon limit

• LDPC codes are adopted for
  • IEEE 802.3 10Gbit Ethernet standard
    • Input BER > 10^{-3}
    • Output BER < 10^{-13}
    • Code rate 0.84
  • DVB-S2 Digital Video Broadcast standard
LDPC Codes: Structure

- $D_c = 4$
- $D_v = 2$
- $M = 5$ check nodes
- $N = 10$ variable nodes

- Each bit participates in $D_v$ parity checks
- Each check consists of $D_c$ bits
- Good LDPC codes are long (large $N$) and with a random-like graph

$H_{MxN} =$

\[
\begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1
\end{pmatrix}
\]
Min-Sum LDPC Decoding

- A form of iterative message passing decoding

- Messages in Log-Likelihood Ratio (LLR)
  - \( \log \left( \frac{P(x=0)}{P(x=1)} \right) \)

- Each iteration does two updates:
  - Check node update
    \[
    c_m = \text{chk}(v_1, \ldots, v_{m-1})
    = (\text{sgn}(v_1) \ldots \text{sgn}(v_{m-1})) \min(|v_1|, \ldots, |v_{m-1}|)
    \]

  - Variable node update
    \[
    v_n = \text{var}(c_1, c_2, \ldots, c_{n-1})
    = c_1 + c_2 + \ldots + c_{n-1}
    \]
LDPC Decoders: Architecture

• We use fully-parallel architecture
  • Allows high throughput
  • Graph is directly mapped to hardware

• Major challenge:
  • Complex interconnection
    • Wire capacitance
      - Power dissipation
      - Wire delay
    • Routing congestion
      - Larger area

• We propose a *bit-serial* scheme to reduce interconnections
**Bit-Serial Message-Passing**

- **bit-serial scheme**
  - transfers an *n*-bit message in *n* clock cycles over a *single* wire
  - is Ideal for Min-Sum decoding
    - reduces the number of wires => reduced routing congestion
    - both *Min* and *Sum* functions are naturally bit-serial
  - facilitates efficient gear shift decoding

---

**Bit-parallel**

**Bit-serial**

![Diagram showing bit-parallel and bit-serial message-passing](image)
Min-Sum Approximation
Approximation to Min-Sum Decoding

- In original Min-Sum each check node needs to find first and second minimum

- We approximate Min-Sum:
  - Variable node the same as conventional Min-Sum
  - Check node update:

\[ c_m = \text{check}(v_1, ..., v_m) = (\text{sgn}(v_1), ..., \text{sgn}(v_{m-1})) \text{Min}(|v_1|, ..., |v_m|) \]

- In Approximate Min-Sum only the first minimum needs to be found
  - reduces the check node logic by about 50%
Under full-precision computations, conventional Min-Sum and approximate Min-Sum perform closely.
With quantized calculations, there is more than 0.5dB difference in performance between conventional and approximate Min-Sum.
Approximate Min-Sum + Correction

• We add a correction factor to the check update rule to reduce the gap between conventional Min-Sum and approximate Min-Sum

• Variable node unchanged

• Check node:

  • Define $M = \min(|v_1|, \ldots, |v_{m-1}|, |v_m|)$
  • Define $T = \text{No. of identical absolute minimums}$

  • $c_m = \text{check}(v_1, v_2, \ldots, v_m)$

  \[
  \begin{cases}
  (\text{sgn}(v_1) \ldots \text{sgn}(v_{m-1})) \ M + 1 & \text{if } (V_m = M \ & T = 1),
  \\
  (\text{sgn}(v_1) \ldots \text{sgn}(v_{m-1})) \ M & \text{otherwise}
  \end{cases}
  \]
Approximate Min-Sum + Correction: Performance

By adding the correction factor, performance loss is reduced from 0.6 dB to less than 0.1 dB
FPGA implementation
FPGA Min-Sum Decoder

• Decoder built on FPGA (Stratix EP1S80)

• Fully parallel architecture

• Approximate Min-Sum decoding with correction

• Bit-serial message passing

• A regular-(4,15) (480, 355) LDPC code constructed using progressive edge growth (PEG) algorithm
• Output sign is calculated separately using XOR gates
Variable Node Architecture

• Inputs arrive bit-serially, LSB first
• Addition and subtractions are performed bit-parallel
• For larger word lengths, a serial approach will be more efficient
## Decoder Spec

<table>
<thead>
<tr>
<th></th>
<th>This work</th>
<th>L. Yang et al. TCAS 2006</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FPGA device</strong></td>
<td>Stratix EP1S80</td>
<td>Xilinx XC2V8000</td>
</tr>
<tr>
<td><strong>Logic</strong></td>
<td>66,000 LUTs</td>
<td>53,000 LUTs</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>0</td>
<td>102 blockRAMs</td>
</tr>
<tr>
<td><strong>Code type</strong></td>
<td>PEG_LDPC (4,15) regular</td>
<td>multi-rate codes</td>
</tr>
<tr>
<td><strong>Code rate</strong></td>
<td>0.74</td>
<td>½, 5/8, 7/8</td>
</tr>
<tr>
<td><strong>Code length</strong></td>
<td>480</td>
<td>9000</td>
</tr>
<tr>
<td><strong>Decoding algorithm</strong></td>
<td>Modified Min-Sum</td>
<td>Min-Sum</td>
</tr>
<tr>
<td><strong>iterations/frame</strong></td>
<td>15</td>
<td>24</td>
</tr>
<tr>
<td><strong>Clock frequency</strong></td>
<td>61 MHz</td>
<td>100 MHz</td>
</tr>
<tr>
<td><strong>Throughput</strong></td>
<td>650 Mbit/sec</td>
<td>Up to 40 Mbit/sec</td>
</tr>
<tr>
<td><strong>Quantization</strong></td>
<td>3 bit</td>
<td>-</td>
</tr>
</tbody>
</table>
Conclusion

• Bit-serial message passing
  • An efficient approach for implementing fully-parallel LDPC decoders
  • Reduces interconnection complexity
  • Suitable for Min-Sum decoding

• Modified Min-Sum decoding with the correction factor demonstrates high BER performance with 48% saving in check node logic