On the VLSI Design of High-Performance LDPC Decoders

Mohammad M. Mansour
Electrical and Computer Engineering Dept.
American University of Beirut
Beirut, Lebanon
Email: mmansour@aub.edu.lb

“LDPC @ Work” Workshop
University of Pisa, Italy
Oct. 6, 2006
Outline

- Motivation
- Class of SPCM codes
- Efficient Decoding of LDPC Codes
- Algorithmic Optimizations:
  - Performance: Parallelized Turbo-Decoding Algorithm
  - Low Power: Saturated Decoding Algorithm
- Message Computation Kernels
- Scalable SPCM Decoder Architectures
- LDPC Decoder Chip
- Conclusions
Complexity of Iterative Decoders

- Decoder throughput vs. Code length

![Diagram showing the trade-offs between decoding throughput, complexity, and block length for short and long codes with parallel and serial architectures.]

Focus of this talk: High-throughput low-power decoders of long turbo-like codes
Algorithmic vs. Architectural Aspects

**Algorithmic Performance**
- Error correcting capability approaching capacity

**Algorithmic Complexity**
- Irregular code structure
- Decent code/graph properties: distance, girth, diameter
- Long block-length
- Many decoding iterations

**Algorithmic Flexibility**
- Code rate
- Block length
- Puncturing
- Modulation
- Interleaving patterns

**VLSI Performance**
- High decoding throughput approaching Gbps

**VLSI Complexity**
- Parallel processing
- Parallel interleaving
- Moderate memory overhead
- Moderate power consumption
- Moderate decoder gate count

**VLSI Flexibility**
- Programmable architectures
- Scalable architectures
- Reusable cores

Coding-theoretic & Algorithmic Aspects

VLSI & Architectural Aspects
Sparse Parity-Check Matrix (SPCM) Codes

- Linear block codes defined by a sparse parity-check matrix
- These include:
  - Original Gallager codes
  - Regular LDPC codes
  - Irregular LDPC codes
  - Regular Repeat-Accumulate Codes
  - Irregular Repeat-Accumulate Codes
  - More general Turbo-Like Codes
  - Digital Fountain Codes (LT, Tornado, Raptor)
  - ...
  - ...
  - Any code that can be decoded *efficiently* using the sub-optimal message-passing algorithm...
SPCM Code: Irregular Repeat-Accumulate Code

- Simple repetition codes concatenated with memory-1 convolutional codes through an interleaver.
  - Advantage: Linear encoding complexity

Block length: $n+k$

Code rate: $gk/(gk+m)$
Matrix Description of IRA Codes

\[
H = \begin{bmatrix}
H^i_{n \times k} & H^p_{n \times n}
\end{bmatrix}
\]

\(H^i_{n \times k}\) randomly distributed 1's depending on permuter

\(H^p_{n \times n}\) (parity)

\(m\) randomly distributed 1's depending on permuter

(zeros)
Tanner Graph of RA Codes
Decoding Problem

- Equivalent to mapping Tanner graph into hardware
- **Full Parallel mapping**: Allocate a Bit Function Unit for each bit-node, a Check Function Unit for each check-node, connect through interconnection network
- **Advantages**: High-throughput
- **Disadvantages**:
  - Complex interconnect limiting scalability and clock speed
  - Prohibitive number of function units
  - Power consumption: Uses registers to store messages; high switching activity on large capacitive interconnect wires
  - Practical only for relatively short codes
- **Optimizations**: Bit-serial architectures
Implementation Challenges

- Interconnect bottleneck restricts efficient implementation.
- Ring placement topology has average interconnect length $O(\sqrt{n})$
  - 52.5 mm$^2$ area for a only 1K code in 0.18 μm CMOS!
  - 50% utilization, 3mm-interconnect, routing congestion restricts clock speed.
Interconnect Bottleneck of Parallel Architectures

- Based on post-layout data for a regular (3,6)-LDPC code in 0.18\(\mu\)m, 1.8V CMOS technology

![Graph showing interconnect length and decoder area vs. system size](image)
Serial LDPC Decoder Architectures

- Fold bit-message and check-message computations onto a small set of BFUs and CFUs.
- Store all messages in centralized memory, and communicate through read/write networks.
Design Methodology

I- Architecture-Aware LDPC codes
- Reduced interconnect complexity
- Graph properties matched to the requirements of iterative decoding

II- Turbo-Decoding Algorithm
- Faster convergence
- Increased memory efficiency

Optimizations:
- Performance: Parallelized turbo-decoding algorithm
- Low Power: Saturated Turbo-Decoding Algorithm
- Area: Optimized message computations, shorter bit precision, lookup-table free

III- Code-Programmable, Rate-Tunable
- Efficient, scalable, dynamic routing networks
- Parallel processing using multiple MPU’s
- Distributed message memory

IV- Parameterized Decoder Core
- Power-delay-area optimization
- Geometric parameterization
- Power-rail scaling
- Low-level transistor sizing
Turbo-Decoding Message-Passing Algorithm

- Four steps:
  1. Read extrinsic messages
  2. Subtract extrinsics from total sum of messages

(a) Read extrinsic messages and sum of all vertical messages. $I_3$ corresponds to the grey shaded rectangles.

(b) Subtract extrinsic messages from total sum. The difference $\lambda$ is saved and used as intrinsic messages.
Turbo-Decoding Algorithm

- **Four steps:**
  3. Decode using SISO algorithm
  4. Update total sums

(c) Decode parity-check equation using new prior messages as input.

(d) Decoded messages are saved as extrinsic messages. The total sum of vertical messages is updated by adding these messages to the old saved difference.
Convergence Speed

Standard algorithm

TDMP algorithm

Trajectory of iterative decoding using two-phase schedule at 0.5dB

40 Iterations

Trajectory of iterative decoding using turbo-decoding schedule at 0.5dB

20 Iterations
Mutual Information Transfer
Advantages of Turbo-Decoding Algorithm

- 2X improvement in convergence speed
- Improvement in error performance
  - Order of magnitude in BER at relatively high SNR
- Significant reduction in memory overhead
  - 43.75% saving for DVB-S2 LDPC codes
Optimizations for Performance

- Matrices with non-overlapping bands

Messages computed in parallel

Non-overlapping rows

Non-overlapping rows

Non-overlapping rows

Non-overlapping rows
Optimizations for Performance

- Simplifying the interleaver:
  - Factor bands in $H$ into permutation matrices
  - Interleaver can be realized via programmable hardware
  - Structure called architecture-aware

- Code design: Many explicit construction methods yield powerful SPCM codes (e.g. Ramanujan graphs)
Further Optimizations: Twin Turbo-Decoding

- Twin Turbo-Decoding Algorithm:
  - Process two bands in parallel, performing two passes over same band per iteration
  - Reorder rows to avoid collisions
- First sub-iteration:
Twin Turbo-Decoding (cont’d)

- Second sub-iteration:
Twin Turbo-Decoding (cont’d)

- Third sub-iteration:

```
Compute Messages in parallel
Non-overlapping rows

Compute Messages in parallel
Non-overlapping rows

Non-overlapping rows
```
Twin Turbo-Decoding (cont’d)

- Fourth sub-iteration:

```
Compute Messages in parallel
Non-overlapping rows
Compute Messages in parallel
Non-overlapping rows
Compute Messages in parallel
Non-overlapping rows
```
Convergence Rate

- Significant improvement in convergence rate over Gallager’s algorithm
Low Power Optimization: Saturated Decoding Algorithm

- Idea based on decoding Fountain codes on BEC
Saturated Decoding Algorithm

- Slice all variable nodes whose sum of messages exceeds a threshold.
- Update neighboring check nodes
- Remove edges from graph
**Savings in Computations**

- Significant savings in computations at high SNR

![Graph showing savings in computations](chart.png)
Reduced-Complexity Message Computation

- Employ a simple function to compute check messages:

\[
Q(x, y) = \max(x, y) + \max(5/8 - |x - y|/4, 0) - \max(x + y, 0) - \max(5/8 - |x + y|/4, 0)
\]

\[
\approx \log(e^x + e^y) - \log(e^{x+y} + 1).
\]

\[
\approx \text{sgn}(xy) \cdot \psi^{-1}(\psi(|x|) + \psi(|y|))
\]

- Properties:

(i) \(Q(x, y) = Q(y, x)\)

(ii) \(Q(-x, -y) = Q(x, y)\)

(iii) \(Q(x, 0) = Q(0, y) = Q(0, 0) = 0\)

(iv) \(Q(x, -y) = Q(-x, y) = -Q(x, y)\)

(v) \(Q(|x|, |y|) \geq 0\)

(vi) \(Q(x, y) = Q(\text{sgn}(x) \cdot |x|, \text{sgn}(y) \cdot |y|) = \text{sgn}(xy) \cdot Q(|x|, |y|)\)

(vii) \(\text{sgn}(xy) \cdot \psi^{-1}(\psi(|x|) + \psi(|y|)) = Q(x, y)\)

(viii) \(\text{sgn}(xyz) \cdot \psi^{-1}(\psi(|x|) + \psi(|y|) + \psi(|z|)) = Q(Q(x, y), z)\)

(ix) \(\text{sgn}(\prod_{i=1}^{c} x_i) \cdot \psi^{-1}(\sum_{i=1}^{c} \psi(|x_i|)) = Q(\cdots (Q(Q(x_1, x_2, x_3), \cdots, x_c), \cdots, x_c)\)
Error Performance

- Ideal
- Using proposed $Q(x,y)$
- Using $q_c(x,y)$, $c=0.5$

BER vs. $E_b/N_0$ [dB]

Using Q-function

Ideal
Message Computation Architecture

- Q-function block

![Diagram of Message Computation Architecture with Q-function block](image-url)
To process a parity-check equation with \( c \) check nodes:

\[
\text{sgn}(\prod_{i=1}^{c} x_i) \cdot \psi^{-1}\left(\sum_{i=1}^{c} \psi(|x_i|)\right) = Q(\cdots(Q(Q(x_1, x_2), x_3), \cdots), x_c)
\]
SISO Decoder: Parallel Dataflow

\[ \Lambda_j = Q_{[j]}(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6), \quad j = 1, \ldots, 6 \]
SPCM Decoder Architecture
LDPC Decoder Chip

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datapath</td>
<td>4-bit</td>
</tr>
<tr>
<td>Code length</td>
<td>2048</td>
</tr>
<tr>
<td>Code rate</td>
<td>0.5-0.875</td>
</tr>
<tr>
<td>Frequency</td>
<td>125 MHz</td>
</tr>
<tr>
<td>Throughput</td>
<td>1.6 Gps</td>
</tr>
<tr>
<td>IO Pins</td>
<td>76</td>
</tr>
<tr>
<td>Area</td>
<td>14.3 mm²</td>
</tr>
<tr>
<td>Power</td>
<td>787 mW</td>
</tr>
<tr>
<td>Technology</td>
<td>0.18 um TSMC</td>
</tr>
<tr>
<td>Foundry</td>
<td>MOSIS</td>
</tr>
</tbody>
</table>
Chip Power and Area Distribution

(a) Power distribution

(b) Area distribution
Conclusions

- High-throughput decoders for LONG SPCM codes cannot be implemented efficiently using fully parallel architectures
  - Interconnect bottleneck
  - Large footprint

- Desirable features:
  - High throughput
  - Low memory overhead
  - Support for long codes
  - Flexibility

- Changes required span
  - Code design: Structured (architecture-aware) codes
  - Decoding algorithm: TDMP + Q-function
  - Decoder architecture: Programmable architecture