Design issues on the parallel implementation of versatile, high-speed iterative decoders

Sergio Benedetto
(benedetto@polito.it)
CERCOM - Dipartimento di Elettronica
Politecnico di Torino

Tainan, November 26, 2009
Acknowledgments

This presentation summarizes a body of work performed by the communications group at Politecnico di Torino. In particular, I would like to thank the contributions of Guido Montorsi, Libero Dinoi and Alberto Tarable.
Summary of the presentation

- Introduction and motivations
- Algorithms and architectures for high-speed parallel iterative decoders
- The problem of collisions in memory access: code-dependent and code-independent solutions
- Code rate versatility: PCCC, SCCC, LDPC codes
- Interleaver versatility
- Merging parallelism and versatility
Summary of the presentation

- Introduction and motivations tainan_sergio_aux.ppt
- Algorithms and architectures for high-speed parallel iterative decoders
- The problem of collisions in memory access: code-dependent and code-independent solutions
- Code rate versatility: PCCC, SCCC, LDPC codes
- Interleaver versatility
- Merging parallelism and versatility
Introduction and motivations

- After the invention of turbo codes (ICC93, Geneva), turbo-like codes and the resurrected LDPC)codes have seen many applications in system communication standards (CCSDS, UMTS, DVB-RCS, DVB-S2, DVB-T2, IEEE 802.16e, etc.)

- Both classes of codes are matching very well the requirements of "good" codes according to Shannon:
  - a high degree of randomness
  - A co-decoding complexity roughly linear with the code word length

- As such, they approach the capacity limits as close as no other code in the past
Introduction and motivations

- A good deal of work has been devoted to the efficient implementation of iterative decoders for turbo-like and LDPC codes, using DSP, FPGA, and ASICS.

- Recently, with the ever increasing demand in the transmission rate, and the varying channel conditions in many system applications, emphasis has been shifted to:
  - **Very high-speed implementation**, which poses new problems and requires a careful code, algorithms, and architectural design.
  - **Versatility of the code-modulation scheme**.
Introduction and motivations

High-speed decoding (1 Gbit/s and beyond):

**Applications** :
- Magnetic recording
- Optical fiber communication (100 Gbits/s!)
- WPAN (IEEE 802.15.3c working group)

**Requirements** :
- Parallel algorithms and architectures
- Collision-free memory access
- Similar for turbo-like and LDPC decoders
Versatile co-decoders:

- **Versatility in terms of:**
  - Information block size, in order to satisfy different latency requirements or data rates (e.g., UMTS standard, DVBS-2,…)
  - Code rate/ spectral efficiency

- **Requirements:**
  - Puncturable codes (to yield different rates)
  - Wide size range of good interleavers for turbo-like codes (to yield different information or code word length)
  - Easily expandable/prunable H matrices for LDPC codes (to yield different information or code word length)
Turbo-like co-decoders

Parallel concatenated convolutional code

Serially concatenated convolutional code

Iterative decoder
LDPC co-decoders

Variable nodes

Check nodes
Summary of the presentation

- Introduction and motivations
- **Algorithms and architectures for high-speed parallel iterative decoders**
- The problem of collisions in memory access: code-dependent and code-independent solutions
- Code rate versatility: PCCC, SCCC, LDPC codes
- Interleaver versatility
- Merging parallelism and versatility
Parallel architectures and algorithms

We assume block encoding and decoding:

- Required in most cases by the frame structure of the system
- Obvious for LDPC codes, which are inherently block codes
- Requires *trellis termination* for turbo-like convolutional codes
Block decoder structure (LDPC, PCCC, SCCC)

Throughput

\[ \frac{R_{\text{data/sec}}}{N/\text{sec}} \]

Complexity \( C = C_a + C_b \)
Memory \( M = N + N_i \)

From the channel A/D

\[ N \]

LLR memory

Processor

A

Internal memory (EXT) size \( N_i \)

B

\[ K \]
Block decoders data dependency

- The two processors must operate sequentially on the internal data
Block decoder structure: PCCC

Throughput

Bit/sec

Data/sec

From the channel A/D

LLR
memory

SISO upper

SISO lower

Data/sec

Internal shared memory size $K$
Block decoder structure: SCCC

**Throughput**

**Data/sec**

From the channel A/D

N

LLR memory

SISO Inner

SISO Outer

Internal shared memory size $N=K/r_o$

Processors

Throughput

**bit/sec**
Block decoder structure: LDPC

Throughput

edge density: average variable degree

Internal memory size $N$
High-speed architectures

- Trivial solution: functional replication
  
  $L$ decoders

- It is often assumed in the complexity evaluation of turbo decoders
  - The memory dominates the overall complexity when the block size is large

- It was never considered for the implementation of LDPC
General parallel decoder architecture: a cleverer solution

Throughput: $LR$
Complexity: $LC$
Memory: $M$

We assume the same number of processors for decoders 1 and 2.
Parallel decoder architecture: PCCC

Diagram showing two sets of SISO processors labeled A 1, A L, B 1, and B L, connected by LLR modules and an EXT module. The diagram also indicates two sets of processors, upper and lower, with arrows indicating the flow of information between them.
Parallel decoder architecture: SCCC
Parallel decoder architecture: LDPC

Check node processors

CNP 1

CNP L

Variable node processors

VNP 1

VNP L

EXT

LLR memory

......
Problem for turbo-like decoders

- The parallel architecture requires that parallel processors do not show data dependency
  - For LDPC this is true for processors on check nodes and variable nodes, so that parallelism can be increased up to the number of check nodes and variable nodes
  - For turbo decoder this is not possible in principle

\[ \text{Dependency graph of turbo decoders} \]

\[ \text{Initialization of Forward - backward recursions} \]
The weak dependency of one window with respect to the adjacent windows can be broken without affecting the performance, provided that the window is not too small.
Simulation results: SCCC decoder with delayed initialization

- SCCC decoder
- 4-state constituent encoders
- variable window size
Summary of the presentation

- Introduction and motivations
- Algorithms and architectures for high-speed parallel iterative decoders
- The problem of collisions in memory access: code-dependent and code-independent solutions
- Code rate versatility: PCCC, SCCC, LDPC codes
- Interleaver versatility
- Merging parallelism and versatility
The memory collision problem

The two accesses to the internal memory follow different orders because of the permutation!
The memory collision problem

- Each of the $L=N/w$ processors of a SISO decoder reads and writes sequentially from $L$ memory banks.
- At each time instant, $L$ accesses to the memory have to be dealt with.
- When two (or more) processors access the same memory bank at the same time instant we have a *collision*.
The memory collision problem

Collisions can be avoided:

1. Using very high-speed memory and/or additional hardware

1. Constraining the interleaver/code design in order to avoid them

1. With permutation decomposition, which applies to any given permutation
Solution I: High speed memory

- **High speed memory**: When collisions occur, some additional hardware can solve the problem by serializing the memory access:

  - It requires memory with speed higher than the processor speed or implies a loss of throughput
  - When the parallelism is very high the solution becomes unfeasible without reducing the overall decoder speed
Solution II: Joint code/decoder design

- **Proper designing of the code/interleaver**: By designing the interleaver/code targeting a specific hardware architecture, i.e. parallelism, it is possible to avoid collisions:
  - The code/interleaver must be matched to the architecture
  - Changing the parallelism degree may require a change of the code/interleaver
  - The constraints imposed to the code may lead to performance losses (e.g., smaller interleaver spread)
  - It does not apply in general to existing code standards
Constrained design of interleavers for PCCC and SCCC

- Constraint on values of the elements of the interleaver
  - $L$, number of processors
  - $N$, interleaver size

Mathematical expression:

\[ \forall i \in \left[ 0; \frac{N}{L} - 1 \right], \forall j, k \in [0; L - 1], j \neq k \]

\[ \left[ \Pi \left( i + \frac{N}{L} j \right) / (N/L) \right] \neq \left[ \Pi \left( i + \frac{N}{L} k \right) / (N/L) \right] \]
Parallelism and LDPC code design

- Joint code-decoder design (structured matrix[1])
  - Macro-matrix, expansion factor
  - Building blocks: permuted versions of the identity matrix
  - Very popular solution (DVB-S2, etc…)
  - Compact description of the parity-check matrix (the macro-matrix and permutations of building blocks, further improvement if only shifted identity matrices are allowed)

LDPC Parallel decoder

- Each column (row) of a macro-column (macro-row) assigned to a different processor

- Representation of the code in terms of:
  - Temporal permutation (address)
  - Spatial permutations (memory bank)

- Easy addressing rule:
  - j\text{th} unit, k\text{th} time instant
  - Address \( \square(k) \), bank \( (\square(k)+j)_{\text{mod}L} \)
Solution III: Permutation decomposition

- **Permutation decomposition**: It has been proved [1] that, for any desired parallelism $L$, it is possible to find a collision-free decomposition of any given permutation as follows

$$L \quad \text{permutations} \quad \frac{N}{L} \text{elements}$$

$$\text{sequence of } \frac{N}{L} \text{permutations on } M \text{ elements}$$

A comparison between solutions II and III:

- **Solution II**
  - **Pros**: Optimal in terms of throughput increase; no overhead in terms of memory/complexity
  - **Cons**: Cannot be applied to code defined by standards; it can be difficult to obtain good codes for very high values of parallelism

- **Solution III**
  - **Pros**: Universal for every code and every parallelism degree
  - **Cons**: Penalty in terms of the description of the permutation decomposition needed to route the extrinsic information
LDPC and turbo decoder admit very similar parallel architectures.

A collision-free memory access scheme can be realized without any particular attention to the structure of the parity-check matrix or the interleaver.

This permits the high-speed parallel implementation of existing code standards and/or carefully optimized codes/interleaver.

This approach, however, requires two \( L \)-way multiplexers, where \( L \) is the degree of parallelism.

Special code construction or interleaver structures may further simplify the routing problem.
Summary of the presentation

- Introduction and motivations
- Algorithms and architectures for high-speed parallel iterative decoders
- The problem of collisions in memory access: code-dependent and code-independent solutions
- **Code rate versatility**: PCCC, SCCC, LDPC codes
- Interleaver versatility
- Merging parallelism and versatility
Code-rate and block size versatility

- **Code-rate versatility**
  - PCCC puncturing
  - SCCC puncturing and/or interleaver size modification
  - LDPC puncturing or extension of a minimum set of distinct encoders for LDPC codes

- **Block size versatility**
  - PCCC and SCCC interleaver size modification
  - LDPC Shortening or different expansion factors of the macro-matrix
Code-rate: PCCC & SCCCs

- **PCCC**
  - Parity-check and/or systematic bits are punctured according to trade-off strategies between convergence threshold and error-floor performance

- **SCCC**
  - See next example
LDPC Codes

- Class of codes considered:
  - Dual-diagonal pattern [1]
  - Easy encoding
  - Very popular (DVB-S2, e-IRA, IEEE 802.16e, …)

- Code design technique, specific for the dual-diagonal pattern.
  - Constraints [2] on
    - Degree distribution
    - Cycle structure


Versatility

- Code-rate
  - Puncturing (high rates)
    - Alternate pattern
  - Extension (low rates)
    - Deterministic rule
    - Optimal spacing properties
    - and same parallelism degree

- Block length
  - Shortening
Puncturing

- Punctured bits are similar to erasures
- The role of stopping sets and erasures:
  - Definition of stopping sets
  - BP decoding: error-free decoding is impossible if the bits of a stopping set are erased
- No stopping set is erased if there are no parity-check equations affected by more than one erasure
  - Very easy condition to enforce for this class of LDPC
  - Puncturing rule based on alternate pattern
  - From a code with rate $R_c$ up to rate $(2R_c/(1+R_c))$
- Five codes to cover the range $[0.20;0.89]$
Shortening

- Well-known technique

- How to **choose the deleted columns** (2 options):
  - Approximation of the optimal degree distribution for the desired rate
  - Deletion of low-degree systematic columns (improvement of the distance spectrum)

- In **parallel decoders** it is better to delete the same number of columns for each processor (to keep the workload balanced)
Extension

- Very easy extension rule: each row is split into two new rows

- Double the number of rows: from rate $R_c$ to $(R_c/(2-R_c))$

- CN degree distributions highly concentrated around mean value

- Preserves code properties

- The decoder can be configured according to the compact description of the original code

- Same parallelism degree
Results

- The (1200,600) code was designed for $L=30$

- Four (600,300) codes are obtained with different strategies:
  - Exact parameters
  - Puncturing
  - Extension
  - Shortening and puncturing

- Comparison with [3]

---

Code versatility: example of SCCC design

- Design a coding/modulation scheme capable of working as close as possible to the capacity limits in a wide range of spectral efficiencies (say from 1 to 5) with a granularity of roughly 1 dB in the required SNR from one spectral efficiency to the next one.

- Propose solutions that can be implemented at the maximum channel symbol rate of 200 Mbaud, which corresponds for the highest spectral efficiency, to an information rate greater than 1 Gbit/s.

- To compare candidate solutions belonging to the classes of turbo-like codes (parallel and serial) and LDPC.
The design objectives

- Owing to the great flexibility requested to the code, *ad-hoc solutions* aiming at a joint optimisation of the code/modulation pair *have to be discarded* for obvious complexity reasons (a different code for every distinct spectral efficiency)

- The *investigation* has to be *limited* thus to *pragmatic* schemes designed according to the optimisation of one binary encoder to be coupled with all modulations in order to achieve the desired broad range of spectral efficiencies.
The SCCC scheme

- It is based on the serial concatenation of an outer 4-state systematic recursive rate $\frac{1}{2}$ encoder punctured to rate $\frac{2}{3}$, a fixed interleaver independent from the code rate, and an inner 4-state systematic recursive rate $\frac{1}{2}$ encoder with suitable puncturing to obtain the desired rate.
- The code design involves choosing the puncturing patterns matching the desired rate.
Puncturing the systematic bits

- The systematic bits of the inner encoder correspond to the code word generated by the outer encoder, so that the puncturing pattern on these bits has been designed to maximise the free distance of the outer encoder (this is known to maximise the *interleaving gain*), and take into account that puncturing occurs after interleaving.

- Puncturing is performed according to a puncturing pattern periodic with period 200 trellis steps, which correspond in our case to 300 outer coded bits.

- We used a slightly sub-optimal, yet manageable, searching algorithm that works incrementally, in a rate-compatible fashion, so that the punctured positions for a given outer rate are also punctured for all higher rates.
The SCCC rationale

- The main feature of the SCCC scheme consists in moving the puncturing operations at the SCCC output.
- This way, the information exchanged by the two SISOs during the iterative decoding procedure refer to the cascade of a truly rate 2/3 encoder with a truly rate ½ encoder.
- The interleaver gain, which is larger for lower outer code rates, is then kept significant also in the case of heavy puncturing of the outer code; this has a great impact in keeping the error floor limited to very low bit (and frame) error probabilities.
- A very evident advantage of the new scheme is its simplicity, owing to the use of two equal 4-state encoders.
- Having the same encoder replicated twice also helps in the parallel architecture implementation needed to achieve high data rates.
The set of Adaptive coding-modulation (ACM) formats

- The modulation schemes vary from QPSK, to 8-PSK, 16-QAM, 32-QAM and 64-QAM
- The maximum spectral efficiency is $h=5.4$, corresponding to a rate 9/10 code associated with a 64-QAM modulation,
- An ACM format is defined for each 1 dB step on the $E_s/N_0$ scale with the unconstrained capacity formula

$$ SNR_{\text{max}} \leq 2^{5.4} \cdot 1 \cdot 16 \cdot 15 \cdot 15 \text{[dB]} $$

$$ i_i \leq 10 \log_{10} \left( 1 \cdot 10^{(SNR_{\text{max}} \cdot i_i) / 10} \right) $$
The set of Adaptive coding-modulation (ACM) formats

- This approach permits to provide always the maximum available spectral efficiency of 5.4
- The minimum spectral efficiency that can be obtained is 0.666 (1/3 code with QPSK). However, the 1 dB step leads to a minimum spectral efficiency of 0.7254
- The total number of available spectral efficiencies is then 19
- To each spectral efficiency we associate the modulation scheme with the lowest cardinality, provided that the rate of the encoder be smaller or equal to 9/10

\[
R_i \quad \frac{K_i}{8,100} \quad m_j \quad \frac{n_j}{m_j} \quad \frac{9}{10}
\]
Simulated performance of the 27 ACM formats for satellite communications
FER 256-QAM for point-to-point wireless communications

- SCCC with variable rate
- Information block size $k=12,000$
- Code/interleaver design with a parallelism degree of 120
- Data rate: 1 Gbit/s
256-QAM for point-to-point wireless communications: comparison with inf-theoretical bound

<table>
<thead>
<tr>
<th>Rate</th>
<th>Capacity</th>
<th>Coded</th>
<th>Δ@1e-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/8</td>
<td>13,4</td>
<td>14,9</td>
<td>1,5</td>
</tr>
<tr>
<td>8/9</td>
<td>13,8</td>
<td>15,3</td>
<td>1,5</td>
</tr>
<tr>
<td>9/10</td>
<td>14,4</td>
<td>15,6</td>
<td>1,2</td>
</tr>
<tr>
<td>10/11</td>
<td>14,4</td>
<td>15,8</td>
<td>1,4</td>
</tr>
<tr>
<td>11/12</td>
<td>14,7</td>
<td>16,1</td>
<td>1,4</td>
</tr>
<tr>
<td>13/14</td>
<td>15,1</td>
<td>16,6</td>
<td>1,5</td>
</tr>
<tr>
<td>15/16</td>
<td>15,4</td>
<td>16,9</td>
<td>1,5</td>
</tr>
<tr>
<td>19/20</td>
<td>15,9</td>
<td>17,6</td>
<td>1,7</td>
</tr>
<tr>
<td>24/25</td>
<td>16,4</td>
<td>18,2</td>
<td>1,8</td>
</tr>
<tr>
<td>32/33</td>
<td>16,9</td>
<td>19,4</td>
<td>2,5</td>
</tr>
</tbody>
</table>
SCCC for optical communications: BER performance

- SCCC with rate 0.833
- Information block size $k = 15,296$
- Code/interleaver design with a parallelism degree of 300
- Data rate: 100 Gbit/s
SCCC for optical communications: Comparison with information-theoretical bounds

![Graph showing comparison between SCCC and information-theoretical bounds for different code block sizes. The graph includes lines for SCCC long and SCCC short with different rates, and a note specifying Soft BER = 10^{-15}.]
Summary of the presentation

● Introduction and motivations
● Algorithms and architectures for high-speed parallel iterative decoders
● The problem of collisions in memory access: code-dependent and code-independent solutions
● Code rate versatility: PCCC, SCCC, LDPC codes
● Interleaver versatility
● Merging parallelism and versatility
Variable-size S-random interleavers [4]

● Goals:
  ○ Being able to generate on-the-fly all the shortened versions of a large interleaver, whenever it is required to support different interleaver sizes
  ○ Only the largest one is stored (memory savings)

● Constraints:
  ○ Keeping good spread properties in the whole range of desired interleaver sizes
  ○ Easy and low-complexity pruning (shortening) algorithm

Interleaver design technique

- **Incremental** design: a new element is added at each step of the algorithm
- The new elements are chosen on the basis of an **optimization**:
  - Meaning of **optimum**: number of new elements required for an increased spread factor
- Spread **violations** can be **broken** by inserting a new element
  - Analysis of spread violations
    - Positions where to insert new elements
    - Value of the new elements
- On even steps a new element is appended at the end of the permutation and its value is optimized
- On odd steps a new element with maximum value is inserted and its position is optimized
Spread properties

- Optimal spread properties; very fast algorithm
Performance

- 8-state PCCC (UMTS), $R_c = 1/2$, $N=5120$
- The prunable interleaver obtained from $N=640$
Features of the decoder

- **Case study:**
  - 3GPP2 standard (12 interl.)
  - Sum of block lengths: 62,712
  - Storage requirements: 13,820 (77.96% saving)
  - Parallelism degree up to 15

- **Different types of flexibility:**
  - Code-rate (puncturing)
  - Block size (pruning)
  - Parallel degree (R parameter)
謝謝您

HOWDY!