Optimisation conjointe de codes LDPC et de leurs architectures de décodage et mise en œuvre sur FPGA

Thèse présentée devant l’INSA de Rennes en vue de l’obtention du doctorat d’Électronique

Jean-Baptiste Doré

26 Octobre 2007 à 10H00 – Amphithéâtre de FT R&D Cesson-Sévigné
France Telecom R&D Units

- Broadband Wireless Access
  - Innovative Radio Interface (RESA/BWA/IRI)
  - Broadcasting network Cooperation and radio access Mobility (RESA/BWA/BCM)

Supervisors

- Marie-Hélène Hamon – R&D engineer at France Telecom R&D
- Pénard Pierre – R&D engineer at France Telecom R&D
- Ramesh Pyndiah – ENST Bretagne

Contexts

- PRICE - Internal project
  - Prospective Research for Infrastructure and Communication Enhancement
- VERITY - Internal project
  - Validation and Evaluation of Research studies in digital Systems
Outline

- Introduction
- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion
Introduction

- Context
- LDPC codes
- Decoding LDPC codes
- Encoding LDPC codes
- Codes construction
Digital communications

- High data rates
  - Base assumption is now Hundreds of Mbit/s
  - 1-10 Gbit/s in the future?

- Low complexity implementation
  - Small component size
  - Low power consumption
  - Low cost

Physical layer

- Forward Error Correction Scheme
  - Close to the theoretical limit
Background history

- **Introduction**
  - Structured LDPC codes
  - Decoding architectures for LDPC decoders
  - FPGA implementation of LDPC coder/decoder
  - Conclusion

- **Context**
  - LDPC codes
  - Decoding LDPC codes
  - Encoding LDPC codes
  - Codes construction
Background history

- 1948: Shannon
- 1955: Elias
- 1962: Gallager
- 1969: Viterbi, Tanner
- 1981: Tanner codes

**Theoretical limit**

- **BER**
- **SNR**

- **> 3dB**

Introduction

- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion

Context

- LDPC codes
- Decoding LDPC codes
- Encoding LDPC codes
- Codes construction

**Context**

- LDPC codes
- Decoding LDPC codes
- Encoding LDPC codes
- Codes construction
### Background history

#### 1948 - 1955
- **Shannon**
- Convolutional codes
- LDPC codes

#### 1955 - 1962
- **Elias**

#### 1962 - 1969
- **Gallager**
- Convolutional codes
- Viterbi algorithm

#### 1969 - 1981
- **Viterbi**
- **Tanner**
- Tanner codes

#### 1981 - 1993
- **Turbo-Codes**
- Iterative principle

- **Berrou & Glavieux**

- **Breakthrough**
- SNR
- BER
- Theoretical limit
- > 3dB

---

#### Introduction
- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion

#### Context
- LDPC codes
- Decoding LDPC codes
- Encoding LDPC codes
- Codes construction

---

#### Structure

- Introduction
- Context
- Background history
- Breakthrough

---

#### Timeline
- 1948
- 1955
- 1962
- 1969
- 1981
- 1993

---

#### Key Figures
- Shannon
- Elias
- Gallager
- Viterbi
- Tanner
- Berrou & Glavieux
### Background history

#### 1948-1981
- **Shannon**
- Convolutional codes
- LDPC codes
- Viterbi algorithm
- Tanner codes

#### 1969-1993
- Gallager
- Convolutional codes
- LDPC codes
- Viterbi algorithm
- Tanner codes

#### 1993-2000
- Gallager
- Convolutional codes
- LDPC codes
- Viterbi algorithm
- Tanner codes

### Breakthrough
- **Turbo-Codes**
- *Iterative principle*

### Context
- LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Encoding LDPC codes
- Codes construction

### Introduction
- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion

### Time line
- 1948
- 1955
- 1962
- 1969
- 1981
- 1993
- 1995
- 1998
- 2000

#### Key Figures
- Shannon
- Elias
- Gallager
- Viterbi
- Tanner
- MacKay

#### Performance Metrics
- SNR
- BER
- Theoretical limit
- Error floor
- Threshold effect
- <0.01 dB
Definitions

- **LDPC codes**
  - Low Density Parity Check codes
  - Parity check constraints, \( M \) parity equations and \( N \) bits

\[
H x^t = 0^t
\]

- **Modeling**
  - Matrix representation

\[
H = \begin{bmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1
\end{bmatrix}
\]

- **Graphical definition**
  - Bipartite graph (Tanner graph)
Belief Propagation (BP) Algorithm

- Graph based algorithm
- Computation of messages which are propagated along the edges
  - Exchange of extrinsic information
- Optimal decoding
  - No cycle into the code graph
Problematic of encoding

- **Encoding LDPC codes**
  - Unconstraint parity check matrices
    - Encoding through the generator matrices $G$

  \[ GH^t = 0 \quad x = cG \]
**Problematic of encoding**

**Encoding LDPC codes**

- Unconstraint parity check matrices
  - Encoding through the generator matrices $G$

- Constraint parity check matrices
  - Quasi-cyclic codes
  - Upper/Lower triangular matrices
    - Unconstraint
    - Strictly or not dual-diagonal structure

\[ H = [H_s \ H_p] \rightarrow [H_s \ H_p] \begin{bmatrix} c^t \\ p^t \end{bmatrix} = 0^t \rightarrow H_{pp}^t = H_{sc}^t \]

\[ p^t = H_{p}^{-1}H_{sc}^t \]
Objective:

- Define the position of all the non null elements into the parity check matrix
  - Degree distribution optimization (EXIT Chart, Density Evolution)
Objective:
- Define the position of all the non null elements into the parity check matrix
  - Degree distribution optimization (EXIT Chart, Density Evolution)

Unconstraint construction
- Pseudo random construction
- Progressive Edge-Growth (PEG) algorithm [Hu01]
Design of LDPC codes

- Objective:
  - Define the position of all the non null elements into the parity check matrix
  - Degree distribution optimization (EXIT Chart, Density Evolution)

- Unconstraint construction
  - Pseudo random construction
  - Progressive Edge-Growth (PEG) algorithm

- Structured LDPC codes
  - Dual-diagonal structure (RA and IRA codes)
  - Protograph based codes
    - Quasi-Cyclic codes
  - Etc..
How to define an efficient coding system using LDPC codes?

- Structured LDPC codes family
- Study the link between architectures and codes design
- Optimize jointly codes and architectures

A joint definition of the codes and the encoding/decoding methods is highly recommended
Structured LDPC codes

- Structured LDPC codes design
- Codes analysis
- Decoding structured LDPC codes
- Conclusions
Motivations

- Constraints on family of LDPC codes

  - Good codes have **strictly concentrated** CN degree distribution [Chung01]
    \[
    \rho(x) = \rho_i x^{i-1} + (1 - \rho_i) x^i
    \]
  
  - Richardson *et al.* design rules about **degree 2 variables nodes** [Richardson01,03]

  - **dual-diagonal structure for** $H$

  - Simple characterization
    - Protograph based codes

  - **Parity check matrices designed from permutation matrices**
Definition of the code considered

- The parity check matrix \( H \) \((M \times N)\) is divided into two sub matrices \( H_s \) \((M \times K)\) and \( H_p \) \((M \times M)\)

\[
H = [H_s \ H_p]
\]

- \( H_p \) is defined to be a dual-diagonal matrix
  - Stability condition

\[
\tilde{\lambda}_2 \geq 1 - R
\]

- No short cycles involving only degree 2 variable nodes
Matrix $H_s$ of size $M \times K$ is constructed with both:

- Circularly shifted identity matrices of size $z \times z$
  - Notation: $I_\delta$, $\delta \geq 0$, is a right shifted identity matrix by $\delta$ positions (modulo $z$)

  $$I_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- Null matrices of size $z \times z$
  - Notation: $I_\delta$, $\delta < 0$, is a null matrix

$H_s$ can be defined by a $(m \times k)$ block matrix

- Simple characterization

$$H_s = \begin{bmatrix} I_{\delta(0,0)} & I_{\delta(0,1)} & \cdots & I_{\delta(0,k-1)} \\ I_{\delta(1,0)} & I_{\delta(1,1)} & \cdots & I_{\delta(1,k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ I_{\delta(m-1,0)} & I_{\delta(m-1,1)} & \cdots & I_{\delta(m-1,k-1)} \end{bmatrix}$$
Matrix $H_p$ of size $M \times M$ is a dual-diagonal matrix

- Avoid low weight codeword requires a new definition of $H_p$

$$H_p = \begin{bmatrix}
I & I & I \\
I & I & \cdots \\
I & \cdots & I \\
0 & I & I
\end{bmatrix}$$

$$I_x = \begin{bmatrix}
0 & 0 & 0 & 0 \\
1 & 0 & \cdots & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

Quasi-Cyclic Irregular Repeat Accumulate Codes (QC IRA) [Tanner99]

$$p^t = H_p^{-1}H_s c^t$$
Distances properties
- Which kind of configurations are critical for performance?

Cycles properties
- How to detect cycles into the code graph?
- What is the role of short cycles on decoder behavior?

Definition of a design algorithm for the family of codes studied
Main results

- Based on Return To Zero properties of the dual-diagonal part of $H$
  - Accumulator code

- Bound on minimal distance
  - Influence the choice of parameter $m$ and the smallest variable node degree $q$
  - $d_{\text{min}} \leq 2 + mq$

- Rules on permutation coefficients
  - Weight-Spectrum of the codes can be constrained
  - Avoid the generation of low weight codeword from low weight information word

Equi-repartition of permutation coefficients on $[0,z-1]$ into a column of $H_s$ but not strictly…
Detection of cycle and enumeration of the distribution

Geometrical approach

\[
\begin{bmatrix}
I_{\delta A_0} & I_{\delta A_3} \\
I_{\delta A_1} & I_{\delta A_2}
\end{bmatrix}
\]

\[A_0 \xrightarrow{V} A_1 \xrightarrow{H} A_2 \xrightarrow{V} A_3 \xrightarrow{H} A_0\]
Algorithm for code design

- **Problematic**: Find the unknown coefficient which maximizes the cycle length and guarantees a minimal cycle length (Target Cycle Length-TCL).

\[
\delta_4 \rightarrow ? \begin{bmatrix}
I_0 & I_0 & \bullet & I_0 & I'_{1} \\
I_6 & I_\delta_4 & \bullet & I_0 & I_0 \\
I_3 & \bullet & \bullet & I_0 & I_0
\end{bmatrix}
\]

- Maximizes the cycle length
- Guarantees a minimal cycle length (Target Cycle Length-TCL)

- **Application to the description of a design algorithm**
  - Incremental construction of the code
  - PEG like algorithm
    - Based on protograph representation of the code
Additional constraints

- **Improve design algorithm**
  - Target Cycle Length (TCL) depends on variable node degree
    - $\Leftrightarrow$ ACE (Approximate Cycle Extrinsic message) [Tian03]
  - Avoid low weight codeword and pseudo-codeword (Trapping-set)
    - Better minimal distance
    - Better behavior of the BP decoder
LDPC codes decoding algorithm

- No a priori information on the code structure
  - BP with flooding scheduling

- When the structure of the code is known
  - Explore other decoding strategies

Example: Codes defined by a protograph

- Layered BP decoding
What’s about the dual-diagonal structure properties?
- “Isolate trellis-like sub graphs and locally applying the MAP algorithm is a good scheduling” [Forney01]

Modeling of the considered codes
- Consider the decoder as the dual of the encoder

Association with layer decoding concept
Turbo Layered BP: Scheduling

\[ H_{SC}^t = H_{PP}^t \]
Turbo Layered BP: Scheduling

\[ H_{sc}^t = H_{pp}^t \]
Turbo Layered BP: Scheduling

Illustration of the sequencing
Turbo Layered BP: Scheduling

$$H_{sc}^t = H_{pp}^t$$

Layered 1

Layered 2

Layered 3
Simulation results

- Comparison LBP/TLBP

Rules on permutation coefficients to reach the best possible convergence
Definition of structured LDPC codes

- Good performance
- Simple encoding (linear time)
- Simple characterization

QC IRA codes
Definition of structured LDPC codes

- Good performance
- Simple encoding (linear time)
- Simple characterization

Analysis of code properties

- Distance properties
- Cycles properties
Definition of structured LDPC codes
- Good performance
- Simple encoding (linear time)
- Simple characterization

Analysis of code properties
- Distance properties
- Cycles properties

Studies on the decoding of structured LDPC codes
- A priori information on code structure is exploited at the decoder side

QC IRA codes
Codes design algorithm
Turbo Layered BP
Decoding architectures for LDPC decoders

- Conception flow
- Architectures for LBP decoding algorithm
- Architectures for TLBP decoding algorithm
- Conclusions
Framework

- Methodology
  - “Cross stage” design flow

  ![Diagram](image.png)

  - Codes design
  - Decoding Algorithm
  - Layered BP
  - Turbo Layered BP

Introduction
- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion

Conception flow
- Architectures for LBP decoding algorithm
- Architectures for TLBP decoding algorithm
- Conclusions
- Methodology
  - “Cross stage” design flow

- Framework

- Conception flow
  - Architectures for LBP decoding algorithm
  - Architectures for TLBP decoding algorithm
  - Conclusions

- Introduction
  - Structured LDPC codes
    - Decoding architectures for LDPC decoders
    - FPGA implementation of LDPC coder/decoder
  - Conclusion

- Structured LDPC codes
  - Decoding architectures for LDPC decoders
  - FPGA implementation of LDPC coder/decoder
  - Conclusion

- Architectures
  - Conception flow
    - Architectures for LBP decoding algorithm
    - Architectures for TLBP decoding algorithm
  - Conclusions
Methodology

“Cross stage” design flow

- Codes design
- Decoding Algorithm
  - Layered BP
  - Turbo Layered BP
- Architectures
  - Optimization of complexity
  - Optimization of processors activity
  - Optimization of data rate
Methodology

“Cross stage” design flow

A joint design of both code and decoder architecture is highly recommended for the design of an efficient system.
**Problematic:** Maximize the activity of processors?

- Layered BP with serial architecture for CNP

- $z$ is the size of a shifted identity matrix
- $n_p$ is the number of processors working in parallel
Problematic: Maximize the activity of processors?
### Configurations studied

- **$n_p = \max z$**
  - Already studied in the literature
    - WiMAX LDPC codes $R=1/2$ and $2/3$
  - But
    - Very complex for large $z$
    - Not very efficient when $z$ is not constant

- **$n_p < \max z$**
  - Motivations
    - Optimize the activity of processor
    - Target a complexity
  - Goals
    - Look for design rules on permutation coefficients in order to keep the Layered BP properties
Various **sequencing** have been studied

- **Constraints** on the code design

The decoding of a window can start if all the most up-to-date extrinsic information are available
Various **sequencing** have been studied

- Serial scheduling (pipelined or not)
Various sequencing have been studied

- Serial scheduling (pipelined or not)
- Parallel scheduling (pipelined or not)
Various sequencing have been studied
- Serial scheduling (pipelined or not)
- Parallel scheduling (pipelined or not)

Definition of an efficient multi-rate decoder

![Graph showing check node degree vs. coding rate with good codes shaded in red]
Genericity problematic

Exploit the structure of the parity check matrix
- Properties of the dual-diagonal structure
Parallel architecture for SPC processors

- $J_0$ messages are processed in parallel

$\Rightarrow J_0$ ones per rows of $H_s$
Proposed solution

- Parallel architecture for SPC processors
  - $J_0$ messages are processed in parallel

  $\Rightarrow J_0$ ones per rows of $H_s$

  $J_0 = 2$

  \[
  H = \begin{bmatrix}
  1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
  1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
  0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 
  \end{bmatrix}
  \]

  $H = \begin{bmatrix} H_s & H_p \end{bmatrix}$
### Proposed solution

- **Parallel architecture for SPC processors**
  - $J_0$ messages are processed in parallel

\[ J_0 \text{ ones per rows of } H_s \]

\[
J_0 = 2
\]

\[
H = \begin{bmatrix}
1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

\[
H_{eq} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{bmatrix}
\]

\[
H = \begin{bmatrix}
H_s & H_p
\end{bmatrix}
\]
Proposed solution

- Parallel architecture for SPC processors
  - $J_0$ messages are processed in parallel

  \[ H = \begin{bmatrix}
  0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
  1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
  0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 
\end{bmatrix} \]

  \[ H_{eq} = \begin{bmatrix}
  1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
  1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
  0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
  0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 
\end{bmatrix} \]

- Extension of the dual-diagonal part
  - Window decoding of the trellis (serial oriented)
    - Memory size proportional to the size of the window

- Efficient method for TLBP algorithm
  - Very flexible scheme if $J_0$ is well designed
"Architecture driven" approach
- Joint design of code and decoder architectures
"Architecture driven" approach
- Joint design of code and decoder architectures

Architectures for Layered BP decoding algorithm
- Modeling of the CNP processors
- Study the case of \( n_p < \text{max } z \)
- Some open issues
  - Flexible permutation network (barrel shifter)
"Architecture driven" approach
- Joint design of code and decoder architectures

Architectures for Layered BP decoding algorithm
- Modeling of the CNP processors
- Study the case of $n_p < \text{max } z$
- Some open issues
  - Flexible permutation network (barrel shifter)

Architectures for Turbo Layered BP decoding algorithm
- Various sequencing have been described
- Problematic of multi-rate decoder
FPGA implementation of LDPC coder/decoder

- Implementation options
- Quantization
- Complexity considerations
- Simulation results
- Conclusion
Turbo Layered BP algorithm
  - With and without pipeline

- 2 decoding processors
  - $p = 2$
  - Duplication of the buffers
Turbo Layered BP algorithm

- With and without pipeline

- 2 decoding processors
  - $p = 2$
  - Duplication of the buffers

- Double input memories
  - Optimize the processor activity
  - Optimize the decoding throughput

- Memory banks organization
  - Avoid simultaneous access
  - Exploit code structure
Problematic

- Continuous to discrete domain
  - Influence the performance
    - Lower granularity
    - Introduction of erasures
  - Influence the complexity of the decoder
    - Size of the memories
    - Size of internal data path
    - Complexity of the basic operators (+, -, < …)

- What is the good trade off between performance and complexity?
Problematic

- Continuous to discrete domain
  - Influence the performance
    - Lower granularity
    - Introduction of erasures
  - Influence the complexity of the decoder
    - Size of the memories
    - Size of internal data path
    - Complexity of the basic operators (+, -, < …)

- What is the good trade off between performance and complexity?
  - Input data
    - Quantization on 4 bits is a good trade off
  - Internal data path
    - Various methods have been studied
FPGA integration

Introduction
Structured LDPC codes
Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
Conclusion

Implementation options
- Quantization
- Complexity considerations
Simulation results
Conclusions

- FIFO yp 18%
- Decoding processors 52%
- Memory Banks 30%
- Memory

ALTERA STRATIX EP1S80 - C6

Implementation options
- Quantization
- Complexity considerations
Simulation results
Conclusions
FPGA integration

- FIFO yp: 18%
- Memory Banks: 30%
- Decoding processors: 52%
- ALU: 6%
- Buffers: 7%

Memory

- dcv: 81%

Introduction
Structured LDPC codes
Decoding architectures for LDPC decoders
• FPGA implementation of LDPC coder/decoder
Conclusion

Implementation options
Quantization
• Complexity considerations
Simulation results
Conclusions

ALtera Stratix
EP1S80 - C6
FPGA integration

**Memory**
- Decoding processors: 52%
- Memory Banks: 30%
- FIFO: 18%

**Logic cells**
- Decoding processors: 50%
- Memory Banks: 43%
- FIFO: 3%
- Control: 4%

**ALtera Stratix EP1S80 - C6**

Introduction
Structured LDPC codes
Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder

Conclusion
Implementation options
Quantization
- Complexity considerations
Simulation results
Conclusions

Implementation options
Quantization
- Complexity considerations
Simulation results
Conclusions

**Simulation results**

- Input
- Output

- SPC Forward
- SPC Backward
- Max
- Min
- PA
- SPC Forward
- SPC Backward
- Logic
FPGA integration

Memory

- Decoding processors: 52%
- Memory Banks: 30%
- Control: 4%
- FIFO yp: 18%

Logic cells

- Decoding processors: 50%
- Memory Banks: 43%
- FIFO yp: 3%

Drawback of a double input buffer

ALTERA STRATIX EP1S80 - C6

Implementation options
- Complexity considerations

Simulation results
Conclusions
Simulation context

- FPGA Hardware simulation chain
  - ALTERA Stratix EPS80-C6

- Source
  - PRBS 20

- AWGN Channel
  - Box Muller algorithm

- LDPC decoder/coder
  - 4 loaded codes
Simulation context

- FPGA Hardware simulation chain
  - ALTERA Stratix EPS80-C6
Simulation context

- FPGA Hardware simulation chain
  - ALTERA Stratix EPS80-C6
Simulation context

- FPGA Hardware simulation chain
  - ALTERA Stratix EPS80-C6

- QNX real time OS
  - Automatic measures
  - Real time performance curves
Applications

- **Broadcast context**
  - Large block size (≈16 kbits)

- **Parameters**
  - AWGN, QPSK
  - $Q_c = 4$ bits (+/-7)
  - 15 it TLBP

![Graph showing BER and FER](image-url)
Applications

- **Broadcast context**
  - Large block size (≈16 kbits)

- **Parameters**
  - AWGN, QPSK
  - $Q_c = 4$ bits (+/-7)
  - 15 it TLBP

No early error floor

- **Validation of both**
  - Code design algorithm
  - Quantization strategy
Objectives

- Try to do a fair comparison
  - 8 states duo binary Turbo-codes
  - LDPC codes studied

Difficulties of comparisons

- Usually context are different
  - Coding size, coding rate, coding structure, Target performance

- Implementation choices
  - Architectures, Quantizations
  - FPGA (Altera or Virtex), mm² on x µm for ASIC

Proposed scheme

- Similar context (Valentinno)
Duo binary Turbo-Codes vs LDPC

Performance
- DBTC
  - 10 iterations
  - Max Log Map
- LDPC
  - 20 iterations
  - TLBP

Context
- AWGN, QPSK
- $Q_c = 4$ bits (+/-7)

Graph: Performance comparison between DBTC and LDPC with $K = 1800$ bits, $R = 1/2$, QPSK, showing a performance difference of 0.2 - 0.4 dB.
Duo binary Turbo-Codes vs LDPC

Performance
- **DBTC**
  - 10 iterations
  - Max Log Map
- **LDPC**
  - 20 iterations
  - TLBP

Context
- **AWGN, QPSK**
- **$Q_c = 4 \text{ bits (}/+/-7)$$
Introduction
Structured LDPC codes
Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder

Conclusion
Implementation options
Quantization
Complexity considerations
- Simulation results

Duo binary Turbo-Codes vs LDPC

- Implementation aspects
  - FPGA – ALTERA Stratix EP1S80F C6

<table>
<thead>
<tr>
<th>Logic cells</th>
<th>Memory (ko)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1000 2x</td>
<td>5503</td>
</tr>
</tbody>
</table>

LDPC studied

<table>
<thead>
<tr>
<th>Logic cells</th>
<th>Memory (ko)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline</td>
<td>7320</td>
</tr>
<tr>
<td>Without Pipeline</td>
<td>6967</td>
</tr>
</tbody>
</table>

x 1.33 x 1.34
Implementation aspects

- FPGA – ALTERA Stratix EP1S80F C6

<table>
<thead>
<tr>
<th>Logic cells</th>
<th>Memory (ko)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1000 2x</td>
<td>5503</td>
</tr>
<tr>
<td></td>
<td>8.147</td>
</tr>
</tbody>
</table>

Duo binary TC

<table>
<thead>
<tr>
<th>Logic cells</th>
<th>Memory (ko)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline</td>
<td>7320</td>
</tr>
<tr>
<td>Without Pipeline</td>
<td>6967</td>
</tr>
<tr>
<td></td>
<td>10.91</td>
</tr>
<tr>
<td></td>
<td>10.53</td>
</tr>
</tbody>
</table>

Decoding throughput

- At same data rates

\[ \text{it}_{LDPC} \approx \text{it}_{DTC} \]

Without pipeline

\[ \text{it}_{LDPC} \approx 2 \text{it}_{DTC} \]

Pipeline

x 1.33

x 1.34
Performance

- Duo binary TC have a very good decoding threshold
  - Even for small size
- Proposed LDPC outperforms TC at low error rate
Duo binary Turbo-Codes vs LDPC

- Performance
  - **Duo binary TC** have a very **good decoding threshold**
    - Even for small size
    - ! **For large coding size, the gap is reduced**
  - **Proposed LDPC** outperforms TC at **low error rate**
    - ! **It is possible to design DBTC with better behavior at low error rate (3D TC,16 states)**
Duo binary Turbo-Codes vs LDPC

- **Performance**
  - Duo binary TC have a very **good decoding threshold**
    - Even for small size
    - For large coding size, the gap is reduced
  - Proposed LDPC outperforms TC at **low error rate**
    - It is possible to design DBTC with better behavior at low error rate (3D TC, 16 states)

- **Complexity**
  - DBTC outperforms LDPC codes studied
Duo binary Turbo-Codes vs LDPC

- **Performance**
  - Duo binary TC have a very good decoding **threshold**
    - Even for small size
    - For large coding size, the gap is reduced
  - Proposed LDPC outperforms TC at **low error rate**
    - It is possible to design DBTC with better behavior at low error rate (3D TC,16 states)

- **Complexity**
  - DBTC outperforms LDPC codes studied
  - However
    - Maturity of the work and architectures
Performance

- Duo binary TC have a very **good decoding threshold**
  - Even for small size
  - ![Warning] For large coding size, the gap is reduced

- Proposed LDPC outperforms TC at **low error rate**
  - ![Warning] It is possible to design DBTC with better behavior at low error rate (3D TC,16 states)

Complexity

- DBTC outperforms LDPC codes studied
- ![Warning] however
  - Maturity of the work and architectures

---

A FEC technology should be considered into a global system, according to the target application
Integration of the decoders into a FPGA

- Definition of the computational units
- Proof of concept: FPGA integration
Integration of the decoders into a FPGA
- Definition of the computational units
- Proof of concept: FPGA integration

Study of the quantization effects
- Influence of the quantization of channel observations
- Quantization of internal data path
Integration of the decoders into a FPGA
- Definition of the computational units
- Proof of concept: FPGA integration

Study of the quantization effects
- Influence of the quantization of channel observations
- Quantization of internal data path

Applications
- Analysis of architectures proposed for different contexts
- Comparison with duo binary Turbo-Codes
- Low error rate behavior: Track and analyze
Conclusions

- General conclusion
- Future prospects
- Discussion
Analysis of QC IRA codes

- Constraints on permutation coefficients

- Definition of a new algorithm for the design of codes

- Joint studies on code design and decoding algorithm definition
Contributions

- Analysis of QC IRA codes

- Decoding architectures for LDPC codes
  - Layered BP algorithm
  - Turbo Layered BP algorithm
Contributions

- Analysis of QC IRA codes

- Decoding architectures for LDPC codes

- FPGA implementation of LDPC decoders
  - Study of quantization effects
  - Definition of computational units
  - Complexity analysis of the proposed architectures
Contributions

- Analysis of QC IRA codes
- Decoding architectures for LDPC codes
- FPGA implementation of LDPC decoders

Disseminations
- 7 international conferences
- 1 journal submission (under review)
- 5 patents
Extension of the work

- Integration of the hardware decoder into a realistic context
  - Realistic channel
  - Integration into a whole communication system

- Turbo Layered BP decoding
  - Application to other parity check matrix structures
    - Modified dual-diagonal matrix (WiMax)
Perspectives

- **Extension of the work**
  - Integration of the hardware decoder into a realistic context
    - Realistic channel
    - Integration into a whole communication system
  - Turbo Layered BP decoding
    - Application to other parity check matrix structures
      - Modified dual-diagonal matrix (WiMax)

- **Theoretical aspects**
  - Analysis the behavior of the sequencing proposed
  - How to improve the convergence threshold of the codes?
    - Practical aspect (Finite length)
Perspectives

■ Extension of the work
  ■ Integration of the hardware decoder into a realistic context
    • Realistic channel
    • Integration into a whole communication system
  ■ Turbo Layered BP decoding
    • Application to other parity check matrix structures
      – Modified dual-diagonal matrix (WiMax)

■ Theoretical aspects
  ■ Analysis the behavior of the sequencing proposed
  ■ How to improve the convergence threshold of the codes?
    • Practical aspect (Finite length)

■ Implementation issues
  ■ Explore the problematic of flexible ultra-parallelized LDPC decoder
    • Very high throughput decoding
Questions and answers

Optimisation conjointe de codes LDPC et de leurs architectures de décodage et mise en œuvre sur FPGA

Jean-Baptiste Doré


Annexes
Outline

- Simplification of BP
- Illustration of the design of codes
- Performance example: effects of TCL
- BP/LBP/TLBP
- CNP modeling: case of Min-Sum approximation
Practical implementation of BP algorithm

- Approximation of the function $f(x)$
- Limit the number of different edges messages

<table>
<thead>
<tr>
<th>Decoding Algorithm</th>
<th>Variable node update rule</th>
<th>Check node update rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>$m_v^i = y_v^0 + \sum_{v' \notin V_v/v} m_{v'v}^{i-1}$</td>
<td>$</td>
</tr>
<tr>
<td>BP-Based/Min-Sum</td>
<td>idem BP</td>
<td>$</td>
</tr>
<tr>
<td>Offset Min-Sum</td>
<td>idem BP</td>
<td>$</td>
</tr>
<tr>
<td>Normalized Min-Sum</td>
<td>idem BP</td>
<td>$</td>
</tr>
<tr>
<td>λ-min</td>
<td>idem BP</td>
<td>$</td>
</tr>
<tr>
<td>A-min*</td>
<td>idem BP</td>
<td>if $v = \arg\min_{v' \in V_v/v}</td>
</tr>
<tr>
<td>APP - check</td>
<td>idem BP</td>
<td>$</td>
</tr>
<tr>
<td>APP-variable</td>
<td>$m_v^i = y_v^0 + \sum_{v' \in V_v} m_{v'v}^{i-1}$</td>
<td>$</td>
</tr>
</tbody>
</table>
Description of the algorithm
- Target: > 6 length cycle
- $z = 8$
- Mask considered:

$$H = \begin{bmatrix}
I_{\delta_0} & I_{\delta_3} & I_{\delta_6} & I_0 & I'_1 \\
I_{\delta_1} & I_{\delta_4} & I_{\delta_7} & I_0 & I_0 \\
I_{\delta_2} & I_{\delta_5} & I_{\delta_8} & I_0 & I_0 \\
\end{bmatrix}$$

Incremental construction

<table>
<thead>
<tr>
<th>Step</th>
<th>Matrix to analyse</th>
<th>Forbidden coefficient</th>
<th>Choice</th>
</tr>
</thead>
</table>
| 1    | $\begin{bmatrix} I_{\delta_0} & \cdot & \cdot & I_0 & I'_1 \\
\cdot & \cdot & \cdot & I_0 & I_0 \\
\cdot & \cdot & \cdot & I_0 & I_0 \end{bmatrix}$ | \emptyset | $\delta_0 = 0$ |
Algorithm for codes design

$$H = \begin{bmatrix} I_{\delta_0} & I_{\delta_3} & I_{\delta_6} & I_0 & I_1 \\ I_{\delta_1} & I_{\delta_4} & I_{\delta_7} & I_0 & I_0 \\ I_{\delta_2} & I_{\delta_5} & I_{\delta_8} & I_0 & I_0 \end{bmatrix}$$

<table>
<thead>
<tr>
<th>Step</th>
<th>Matrix to analyse</th>
<th>Forbidden coefficient</th>
<th>Choice</th>
</tr>
</thead>
</table>
| 1    | \[
\begin{bmatrix}
I_{\delta_0} & \star & \star & I_0 & I_1 \\
\star & \star & \star & I_0 & I_0 \\
\star & \star & \star & I_0 & I_0
\end{bmatrix}
\] | $\emptyset$ | $\delta_0 = 0$ |
| 2    | \[
\begin{bmatrix}
I_0 & \star & \star & I_0 & I_1 \\
\star & \star & \star & I_0 & I_0 \\
\star & \star & \star & I_0 & I_0
\end{bmatrix}
\] | $[0]$ | $\delta_1 = 6$ |
Algorithm for codes design

\[ H = \begin{bmatrix}
I_{\delta_0} & I_{\delta_3} & I_{\delta_6} & I_0 & I_1 \\
I_{\delta_1} & I_{\delta_4} & I_{\delta_7} & I_0 & I_0 \\
I_{\delta_2} & I_{\delta_5} & I_{\delta_8} & I_0 & I_0
\end{bmatrix} \]

<table>
<thead>
<tr>
<th>Step</th>
<th>Matrix to analyse</th>
<th>Forbidden coefficient</th>
<th>Choice</th>
</tr>
</thead>
</table>
| 1    | \[ \begin{bmatrix}
I_{\delta_0} & \cdot & \cdot & I_0 & I_1 \\
\cdot & \cdot & \cdot & I_0 & I_0 \\
\cdot & \cdot & \cdot & I_0 & I_0
\end{bmatrix} \] | \(\emptyset\) | \(\delta_0 = 0\) |
| 2    | \[ \begin{bmatrix}
I_0 & \cdot & \cdot & I_0 & I_1 \\
I_{\delta_1} & \cdot & \cdot & I_0 & I_0 \\
\cdot & \cdot & \cdot & I_0 & I_0
\end{bmatrix} \] | \([0]\) | \(\delta_1 = 6\) |
| 3    | \[ \begin{bmatrix}
I_0 & \cdot & \cdot & I_0 & I_1 \\
I_6 & \cdot & \cdot & I_0 & I_0 \\
I_{\delta_2} & \cdot & \cdot & I_0 & I_0
\end{bmatrix} \] | \([7, 6]\) | \(\delta_2 = 3\) |

Expansion of the protograph to detect the configurations
### Algorithm for codes design

$$H = \begin{bmatrix} I_{\delta_0} & I_{\delta_3} & I_{\delta_6} & I_0 & I'_1 \\ I_{\delta_1} & I_{\delta_4} & I_{\delta_7} & I_0 & I_0 \\ I_{\delta_2} & I_{\delta_5} & I_{\delta_8} & I_0 & I_0 \end{bmatrix}$$

<table>
<thead>
<tr>
<th>Step</th>
<th>Matrix to analyse</th>
<th>Forbidden coefficient</th>
<th>Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\begin{bmatrix} I_{\delta_0} &amp; \cdot &amp; \cdot &amp; I_0 &amp; I_0 &amp; I'_1 \ \cdot &amp; \cdot &amp; \cdot &amp; I_0 &amp; I_0 &amp; I_0 \end{bmatrix}$</td>
<td>$\emptyset$</td>
<td>$\delta_0 = 0$</td>
</tr>
<tr>
<td>2</td>
<td>$\begin{bmatrix} I_0 &amp; \cdot &amp; \cdot &amp; I_0 &amp; I_0 &amp; I'<em>1 \ I</em>{\delta_1} &amp; \cdot &amp; \cdot &amp; I_0 &amp; I_0 &amp; I_0 \end{bmatrix}$</td>
<td>$[0]$</td>
<td>$\delta_1 = 6$</td>
</tr>
<tr>
<td>3</td>
<td>$\begin{bmatrix} I_0 &amp; \cdot &amp; \cdot &amp; I_0 &amp; I_0 &amp; I'<em>1 \ I</em>{\delta_2} &amp; \cdot &amp; \cdot &amp; I_0 &amp; I_0 &amp; I_0 \end{bmatrix}$</td>
<td>$[7, 6]$</td>
<td>$\delta_2 = 3$</td>
</tr>
<tr>
<td>4</td>
<td>$\begin{bmatrix} I_0 &amp; I_{\delta_2} &amp; \cdot &amp; I_0 &amp; I_0 &amp; I'_1 \ I_6 &amp; \cdot &amp; \cdot &amp; I_0 &amp; I_0 &amp; I_0 \ I_3 &amp; \cdot &amp; \cdot &amp; I_0 &amp; I_0 &amp; I_0 \end{bmatrix}$</td>
<td>$\emptyset$</td>
<td>$\delta_3 = 0$</td>
</tr>
</tbody>
</table>
Algorithm for codes design

\[ H = \begin{bmatrix}
I_{\delta_0} & I_{\delta_3} & I_{\delta_6} & I_0 & I_1 \\
I_{\delta_1} & I_{\delta_4} & I_{\delta_7} & I_0 & I_0 \\
I_{\delta_2} & I_{\delta_5} & I_{\delta_8} & I_0 & I_0 \\
\end{bmatrix} \]

<table>
<thead>
<tr>
<th>Step</th>
<th>Matrix to analyse</th>
<th>Forbidden coefficient</th>
<th>Choice</th>
</tr>
</thead>
</table>
| 1    | \[
\begin{bmatrix}
I_{\delta_0} & \cdot & \cdot & I_0 & I_0 & I_1' \\
\cdot & \cdot & \cdot & I_0 & I_0 & I_0 \\
\cdot & \cdot & \cdot & \cdot & I_0 & I_0 \\
\end{bmatrix}
\] | \(\emptyset\) | \(\delta_0 = 0\) |
| 2    | \[
\begin{bmatrix}
I_0 & \cdot & \cdot & I_0 & I_0 & I_1' \\
I_{\delta_1} & \cdot & \cdot & I_0 & I_0 & I_0 \\
\cdot & \cdot & \cdot & \cdot & I_0 & I_0 \\
\end{bmatrix}
\] | \([0]\) | \(\delta_1 = 6\) |
| 3    | \[
\begin{bmatrix}
I_0 & \cdot & \cdot & I_0 & I_0 & I_1' \\
I_{\delta_2} & \cdot & \cdot & I_0 & I_0 & I_0 \\
I_6 & \cdot & \cdot & I_0 & I_0 & I_0 \\
\end{bmatrix}
\] | \([7, 6]\) | \(\delta_2 = 3\) |
| 4    | \[
\begin{bmatrix}
I_0 & I_{\delta_3} & \cdot & I_0 & I_0 & I_1' \\
I_6 & \cdot & \cdot & I_0 & I_0 & I_0 \\
I_3 & \cdot & \cdot & I_0 & I_0 & I_0 \\
\end{bmatrix}
\] | \(\emptyset\) | \(\delta_3 = 0\) |
| 5    | \[
\begin{bmatrix}
I_0 & I_{\delta_3} & \cdot & I_0 & I_0 & I_1' \\
I_6 & \cdot & \cdot & I_0 & I_0 & I_0 \\
I_3 & \cdot & \cdot & I_0 & I_0 & I_0 \\
\end{bmatrix}
\] | \([0, 6]\) | \(\delta_4 = 7\) |
Algorithm for codes design

\[ H = \begin{bmatrix} I_{\delta_0} & I_{\delta_3} & I_{\delta_6} & I_0 & I'_1 \\ I_{\delta_1} & I_{\delta_4} & I_{\delta_7} & I_0 & I_0 \\ I_{\delta_2} & I_{\delta_5} & I_{\delta_8} & I_0 & I_0 \end{bmatrix} \]

<table>
<thead>
<tr>
<th>Step</th>
<th>Matrix to analyse</th>
<th>Forbidden coefficient</th>
<th>Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[ \begin{bmatrix} I_{\delta_0} &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 &amp; I'_1 \ \bullet &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \ \bullet &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \end{bmatrix} ]</td>
<td>\emptyset</td>
<td>( \delta_0 = 0 )</td>
</tr>
<tr>
<td>2</td>
<td>[ \begin{bmatrix} I_0 &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 &amp; I'<em>1 \ I</em>{\delta_1} &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \ \bullet &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \end{bmatrix} ]</td>
<td>([0])</td>
<td>( \delta_1 = 6 )</td>
</tr>
<tr>
<td>3</td>
<td>[ \begin{bmatrix} I_0 &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 &amp; I'<em>1 \ I_6 &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \ I</em>{\delta_2} &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \end{bmatrix} ]</td>
<td>([7, 6])</td>
<td>( \delta_2 = 3 )</td>
</tr>
<tr>
<td>4</td>
<td>[ \begin{bmatrix} I_0 &amp; I_{\delta_3} &amp; \bullet &amp; I_0 &amp; I_0 &amp; I'_1 \ I_6 &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \ I_3 &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \end{bmatrix} ]</td>
<td>\emptyset</td>
<td>( \delta_3 = 0 )</td>
</tr>
<tr>
<td>5</td>
<td>[ \begin{bmatrix} I_0 &amp; I_0 &amp; \bullet &amp; I_0 &amp; I_0 &amp; I'<em>1 \ I_6 &amp; I</em>{\delta_4} &amp; \bullet &amp; I_0 &amp; I_0 \ I_3 &amp; \bullet &amp; \bullet &amp; I_0 &amp; I_0 \end{bmatrix} ]</td>
<td>([0, 6])</td>
<td>( \delta_4 = 7 )</td>
</tr>
<tr>
<td>\vdots</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>[ \begin{bmatrix} I_0 &amp; I_0 &amp; I_0 &amp; I_0 &amp; I'_1 \ I_6 &amp; I_7 &amp; I_3 &amp; I_0 &amp; I_0 \ I_3 &amp; I_1 &amp; I_0 &amp; I_0 &amp; I_0 \end{bmatrix} ]</td>
<td>([3, 0, 1, 5, 7])</td>
<td>( \delta_8 = 6 )</td>
</tr>
</tbody>
</table>
Algorithm parameterization: some results

- Avoid low length cycle involving low degree variable nodes
Simulation results

- Algorithm parameterization: some results
  - Avoid low length cycle involving low degree variable nodes

![Graphs showing BER and BLER performance](image-url)
Comparison BP/LBP

Good convergence for LBP
15-25 iterations
Comparison LBP/TLBP

Good convergence for TLBP
10-20 iterations
Serial implementation of CNP Processors

- $d_c$ cycles to compute $m_{vc}$

\[
m_{vc} = y_v + \sum_{e \in C_v} m_{e'v} - m_{cv}
\]

\[
m_{vc} = \frac{A_v - m_{cv}}{}\]

- $d_c$ cycles to compute $A_v$

\[
A_v = y_v + \sum_{e \in C_{v/c}} m_{e'v} + m_{cv}
\]

\[
A_v = \frac{m_{vc}}{} + m_{cv}
\]
About CNP processors: Modeling for Min-Sum algorithm (II)

\[ D = p \frac{RN}{M(2d_c + \epsilon)it} f_{clk} \]
About CNP processors: Modeling for Min-Sum algorithm (II)

\[ D = p \frac{RN}{M (2d_c + \epsilon) IT f_{clk}} \]
About CNP processors: Modeling for Min-Sum algorithm (II)

\[ D = p \frac{RN}{M(2d_c + \epsilon)it} f_{clk} \]
About CNP processors: Modeling for Min-Sum algorithm (II)

\[ D = p \frac{RN}{M(2d_c + \epsilon)it} f_{clk} \]
About CNP processors: Modeling for Min-Sum algorithm (II)

\[ D = p \frac{RN}{M(2d_c + \epsilon)t} f_{clk} \]
Min-Sum algorithm

- Comparison of the input $m_{vc}$
  - Two smallest messages

- It can be done during the forward step
  - The backward step is not required

- but...
  - $d_c$ cycles are required for the computation of $m_{vc}$
  - At least $d_c$ cycles are required for the computation of the min (pipeline)
  - $d_c$ cycles are required to re-estimate $A_v$
Min-Sum algorithm

- The proposed model is valid but parameter $\epsilon$ depends on the algorithm

$$\epsilon_{BP} > \epsilon_{MS}$$

$$D = p \frac{RN}{M(2d_c + \epsilon)it}f_{dk}$$