



# 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

## Foreword

- 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 **R**esearch for **I**nfrastructure and **C**ommunication **E**nhancement
  - VERITY - Internal project
    - Validation and **E**valuation of **R**esearch studies in dig**I**Tal s**Y**stems

## 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

**• 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



## Motivations

### ■ 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

• **Introduction**

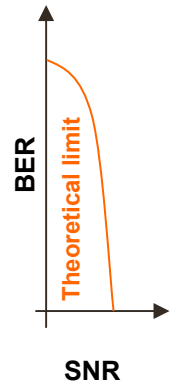
- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion

• **Context**

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



## Background history



**Shannon**



• **Introduction**

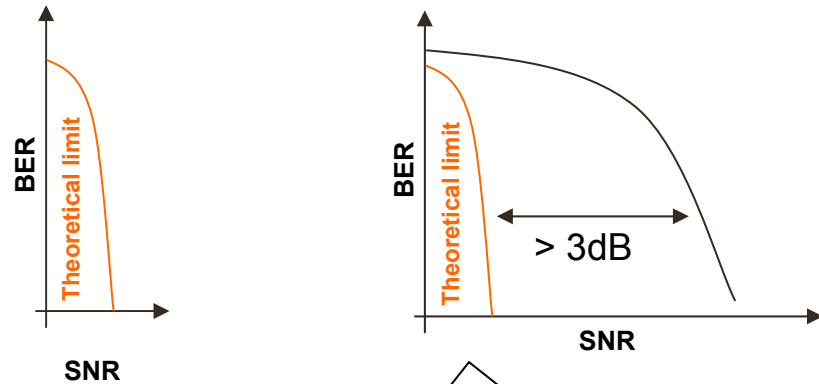
- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion

• **Context**

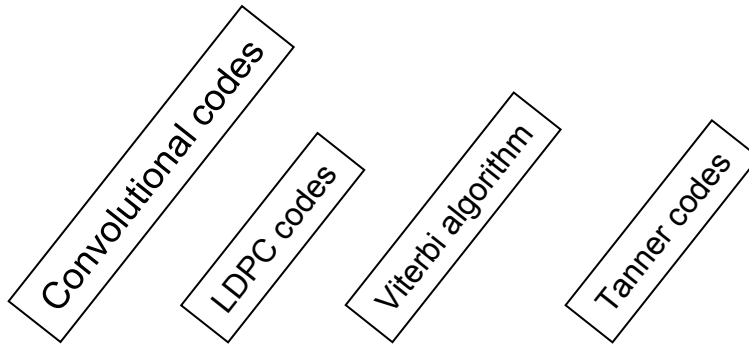
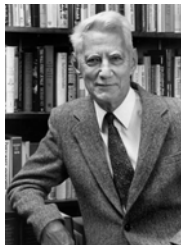
- LDPC codes
- Decoding LDPC codes
- Encoding LDPC codes
- Codes construction



# Background history



Shannon



1955      1962      1969      1981

1948



Elias



Gallager



Viterbi



Tanner

time

• Introduction

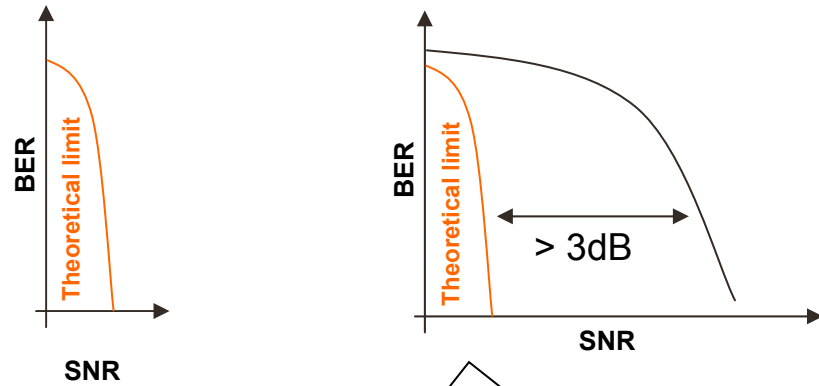
- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion

• Context

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

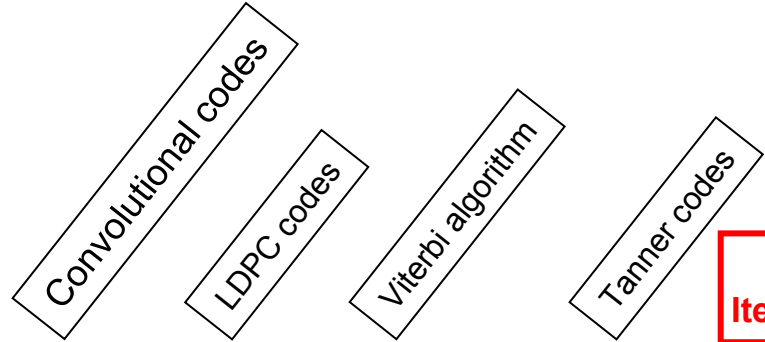
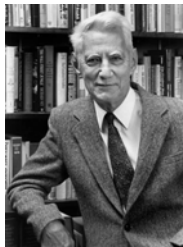


# Background history



Breakthrough

Shannon



Turbo-Codes  
Iterative principle



Elias



Gallager



Viterbi



Tanner



Berrou & Glavieux



• Introduction

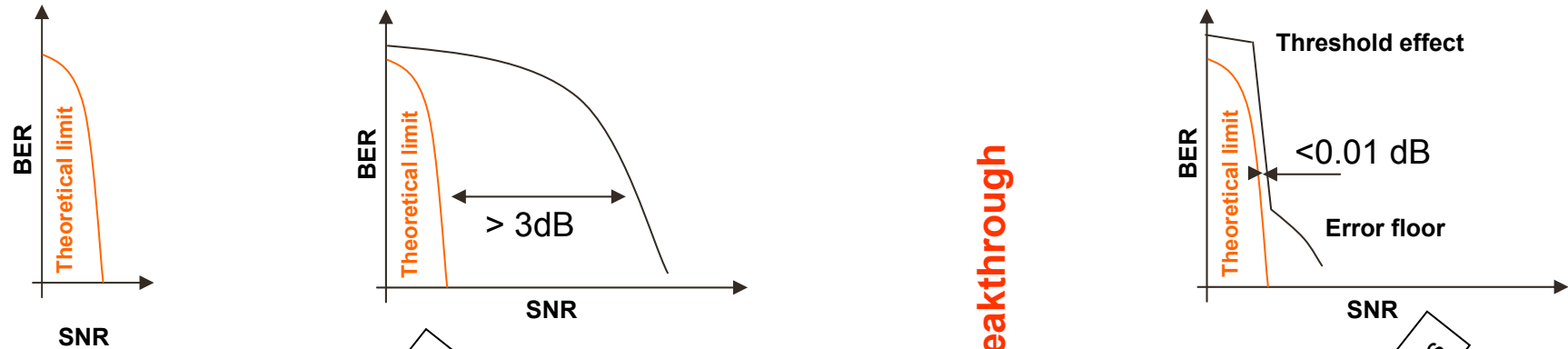
- Structured LDPC codes
- Decoding architectures for LDPC decoders
- FPGA implementation of LDPC coder/decoder
- Conclusion

• Context

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

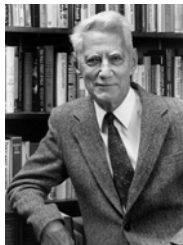


# Background history



**Breakthrough**

Shannon



Convolutional codes

LDPC codes

Viterbi algorithm

Tanner codes

**Turbo-Codes  
Iterative principle**

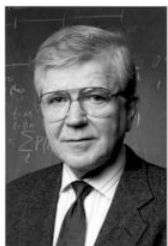
Re-discovery of LDPC codes

Irregular LDPC codes

Duo binary Turbo-Codes



Elias



Gallager



Viterbi



Tanner



Berrou & Glavieux



MacKay



## Definitions

### ■ LDPC codes

- Low Density Parity Check codes
- Parity check constraints,  $M$  parity equations and  $N$  bits

$$\mathbf{H}\underline{x}^t = \underline{0}^t$$

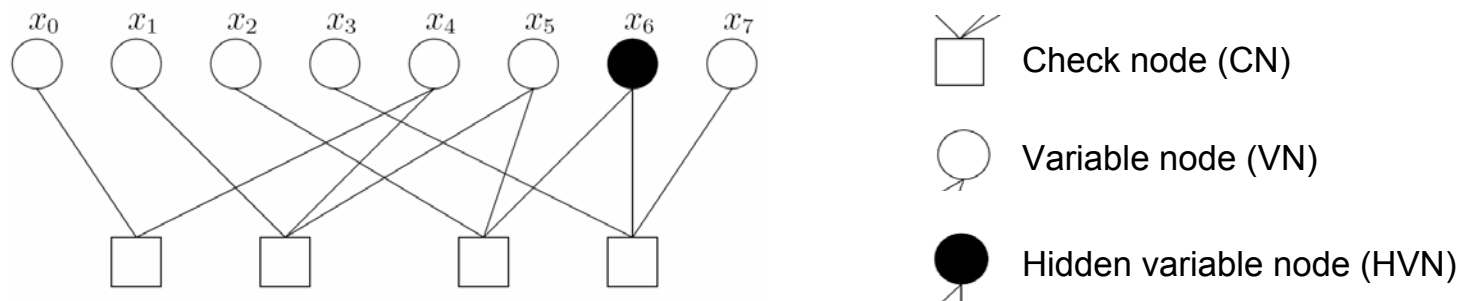
### ■ Modeling

- Matrix representation

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

- Graphical definition

- Bipartite graph (Tanner graph)

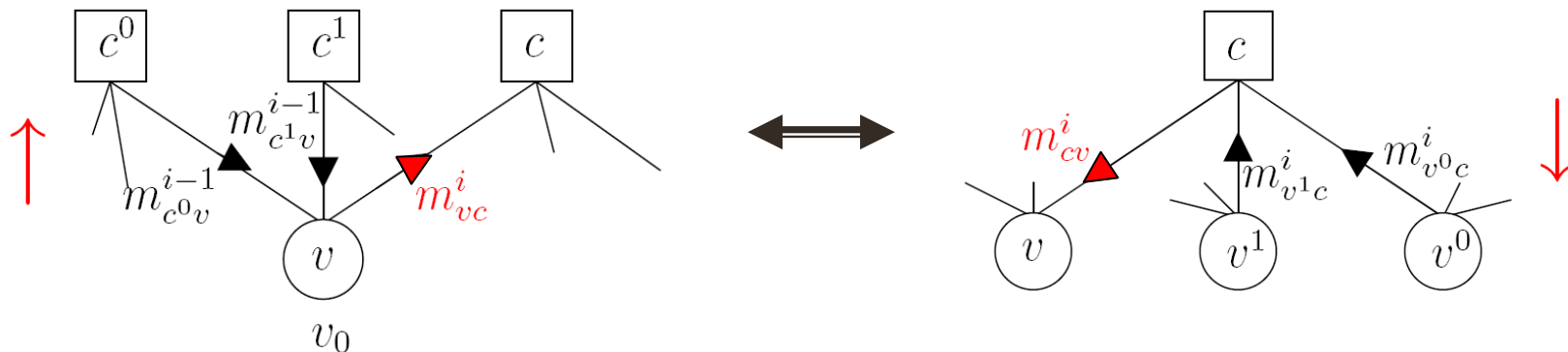




## Decoding algorithm

### ■ 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



- **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



## Problematic of encoding

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

$$\mathbf{GH}^t = \mathbf{0} \quad \underline{x} = \underline{c}\mathbf{G}$$



## Problematic of encoding

### ■ Encoding LDPC codes

- Unconstraint parity check matrices
  - Encoding through the generator matrices  $\mathbf{G}$
- Constraint parity check matrices
  - Quasi-cyclic codes



- Upper/Lower triangular matrices
  - Unconstraint
  - Strictly or not dual-diagonal structure

Simple encoding in a linear time

$$\mathbf{H} = [\mathbf{H}_s \ \mathbf{H}_p] \quad \longrightarrow \quad [\mathbf{H}_s \ \mathbf{H}_p] \begin{bmatrix} \underline{c}^t \\ \underline{p}^t \end{bmatrix} = \underline{0}^t \quad \longrightarrow \quad \underline{p}^t = \mathbf{H}_p^{-1} \mathbf{H}_s \underline{c}^t$$

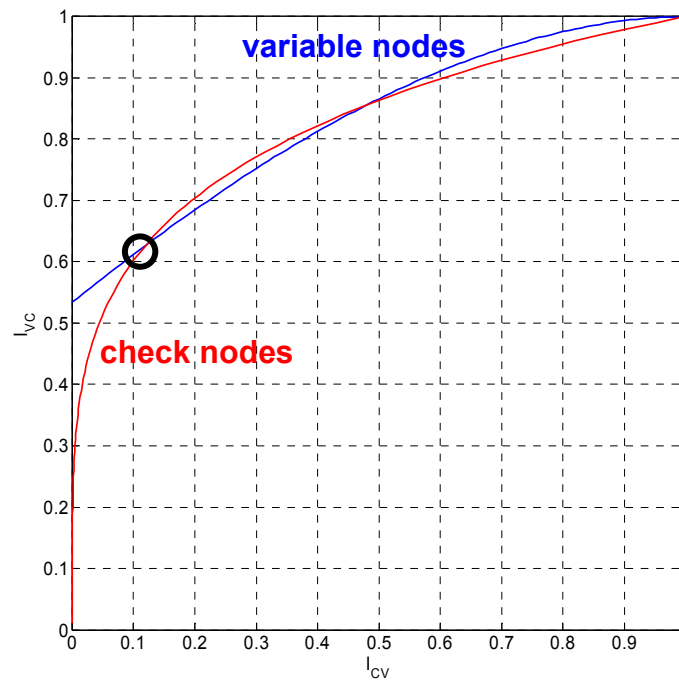
$$\mathbf{H}_p \underline{p}^t = \mathbf{H}_s \underline{c}^t$$



## Design of LDPC codes

### ■ Objective:

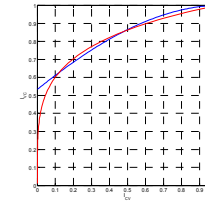
- Define the position of all the non null elements into the parity check matrix
  - Degree distribution optimization (EXIT Chart, Density Evolution)



## Design of LDPC codes

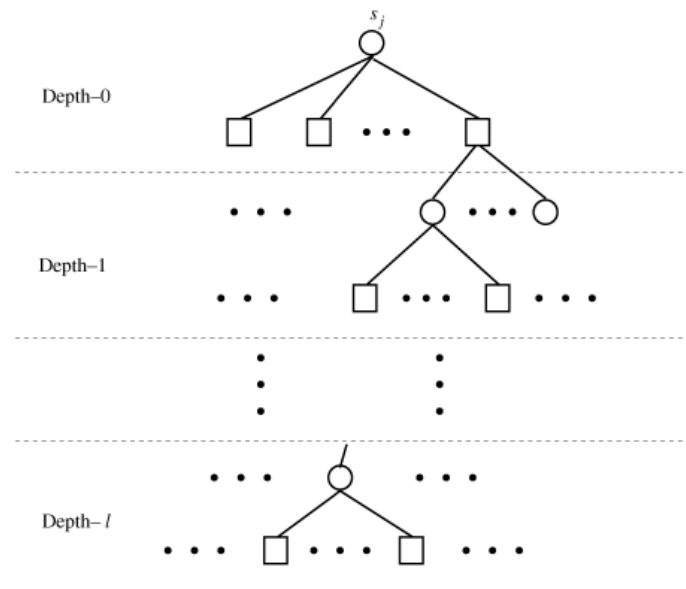
- Objective:

- 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 [Hu01]

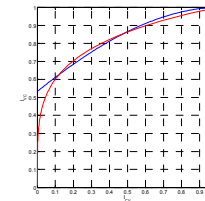


## Design of LDPC codes

- Objective:

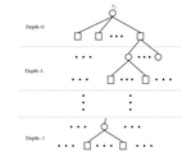
- Define the position of all the non null elements into the parity check matrix

- Degree distribution optimization (EXIT Chart, Density Evolution)



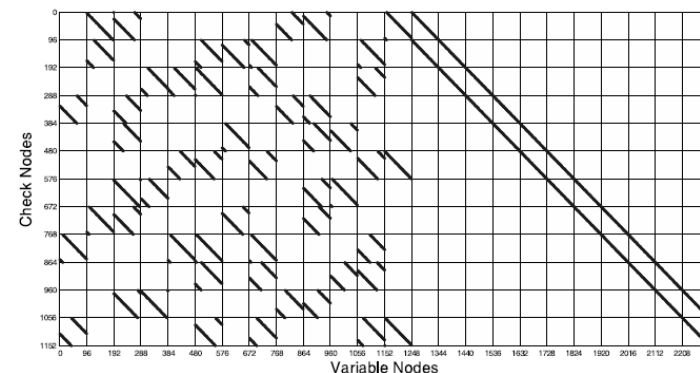
- Unconstraint construction

- 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..







## Problematic

- 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





## Characterization

- Matrix  $\mathbf{H}_s$  of size  $M \times K$  is constructed with both:

- Circularly shifted identity matrices of size  $z \times z$

- Notation:  $\mathbf{I}_\delta$ ,  $\delta \geq 0$ , is a right shifted identity matrix by  $\delta$  positions (modulo  $z$ )

$$\mathbf{I}_1 = \begin{bmatrix} 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 \end{bmatrix}$$

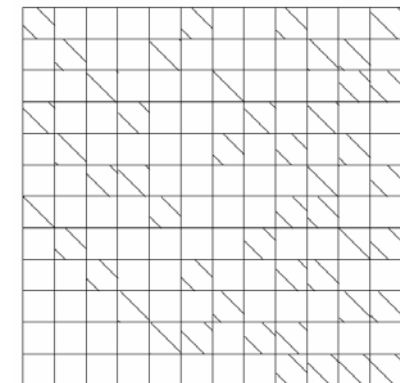
- Null matrices of size  $z \times z$

- Notation:  $\mathbf{I}_\delta$ ,  $\delta < 0$ , is a null matrix

- $\mathbf{H}_s$  can be defined by a  $(m \times k)$  block matrix

- Simple characterization

$$\mathbf{H}_s = \begin{bmatrix} \mathbf{I}_{\delta(0,0)} & \mathbf{I}_{\delta(0,1)} & \cdots & \mathbf{I}_{\delta(0,k-1)} \\ \mathbf{I}_{\delta(1,0)} & \mathbf{I}_{\delta(1,1)} & \cdots & \mathbf{I}_{\delta(1,k-1)} \\ \vdots & \vdots & \mathbf{I}_{\delta(i,j)} & \vdots \\ \mathbf{I}_{\delta(m-1,0)} & \mathbf{I}_{\delta(m-1,1)} & \cdots & \mathbf{I}_{\delta(m-1,k-1)} \end{bmatrix}$$







## Framework

- 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



## Distances properties

### ■ Main results

- Based on Return To Zero properties of the dual-diagonal part of  $\mathbf{H}$ 
  - Accumulator code
- Bound on minimal distance
  - Influence the choice of parameter  $m$  and the smallest variable node degree  $q$

$$d_{\min} \leq 2 + mq$$

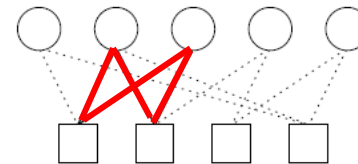
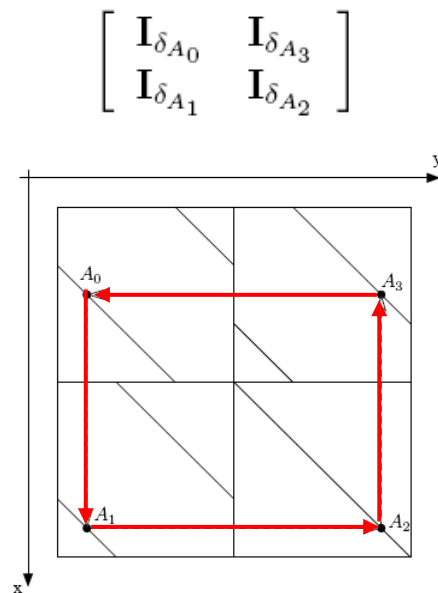
- Rules on permutation coefficients
  - Weight-Spectrum of the codes can be constrained
  - Avoid the generation of low weight codeword from low weight information word

Equi-repartition of permutation coefficients on  $[0, z-1]$  into a column of  $\mathbf{H}_s$   
but not strictly...



## Cycles detection

- Detection of cycle and enumeration of the distribution
  - Geometrical approach



$$A_0 \xrightarrow{V} A_1 \xrightarrow{H} A_2 \xrightarrow{V} A_3 \xrightarrow{H} A_0$$



## Algorithm for code design

- **Problematic:** Find the unknown coefficient which

$$\delta_4 \rightarrow ? \quad \begin{bmatrix} \mathbf{I}_0 & \mathbf{I}_0 & \bullet & \mathbf{I}_0 & & \mathbf{I}'_1 \\ \mathbf{I}_6 & \mathbf{I}_{\delta_4} & \bullet & \mathbf{I}_0 & \mathbf{I}_0 & \\ \mathbf{I}_3 & \bullet & \bullet & & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$$

- 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

- **Structured LDPC codes**

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

- **Codes analysis**

Decoding Structured LDPC codes  
 Conclusions



## Additional constraints

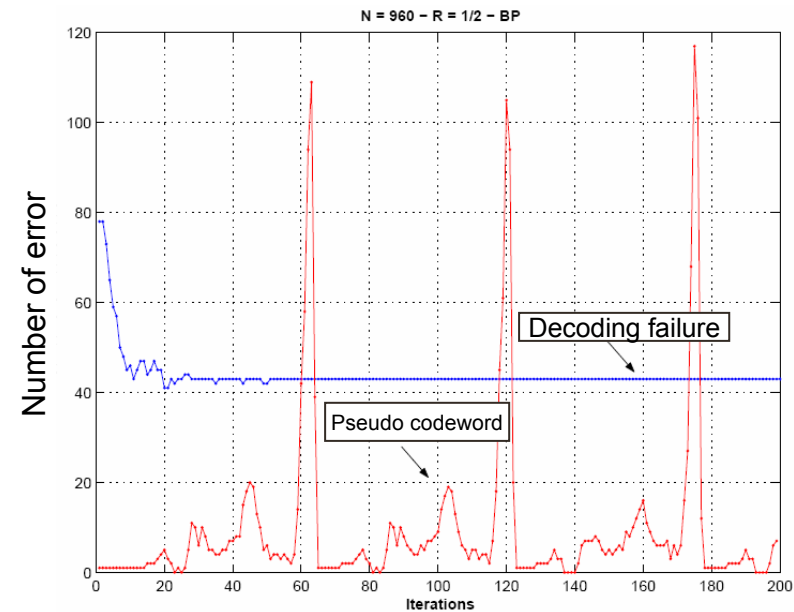
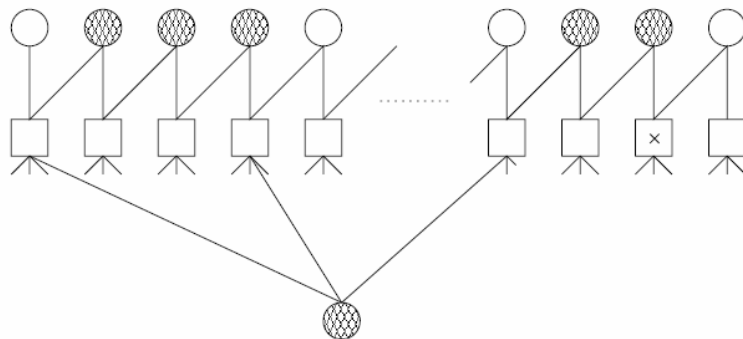
### ■ Improve design algorithm

#### ■ Target Cycle Length (TCL) depends on variable node degree

- ⇔ ACE (Approximate Cycle Extrinsic message) [Tian03]

#### ■ Avoid low weight codeword and pseudo-codeword (Trapping-set)

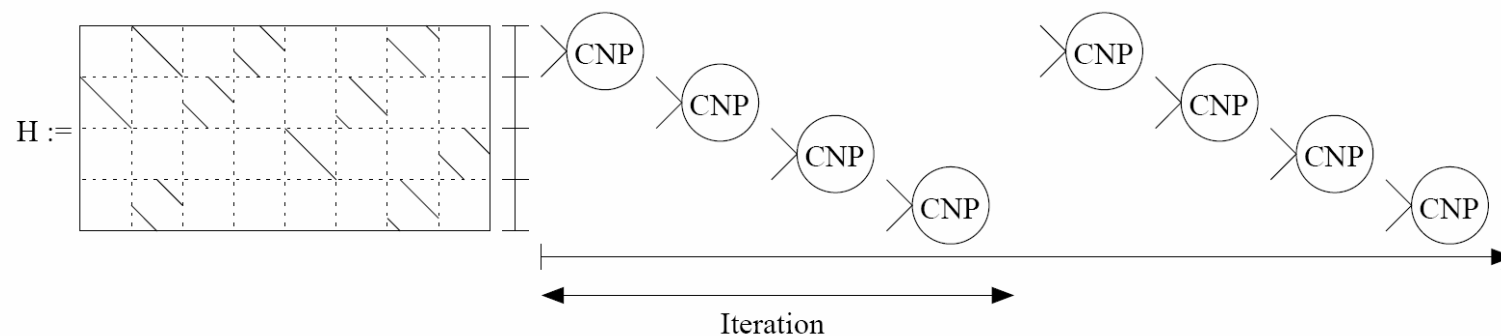
- Better minimal distance
- Better behavior of the BP decoder





## State of art

- **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



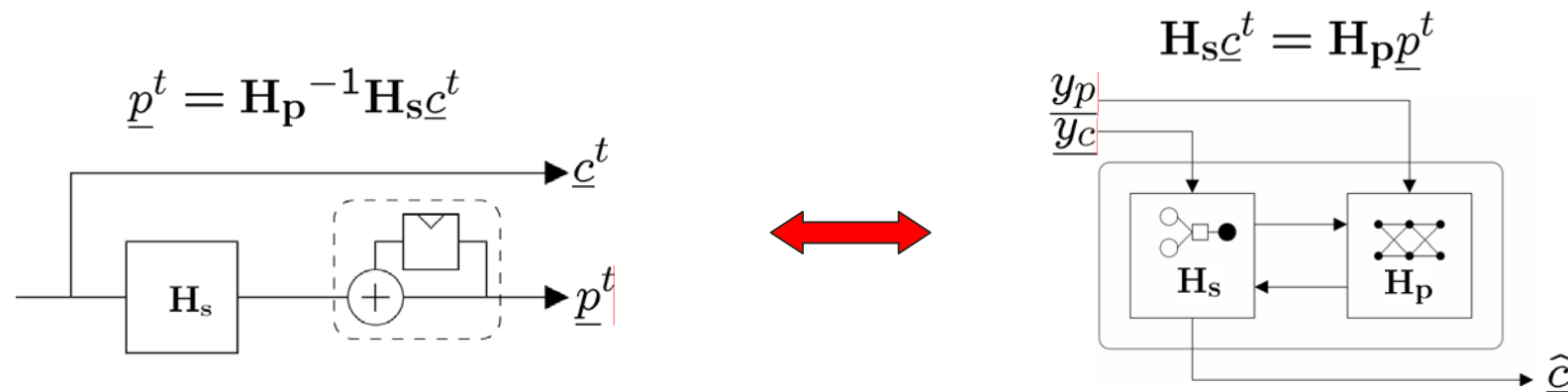


## Main idea

- 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

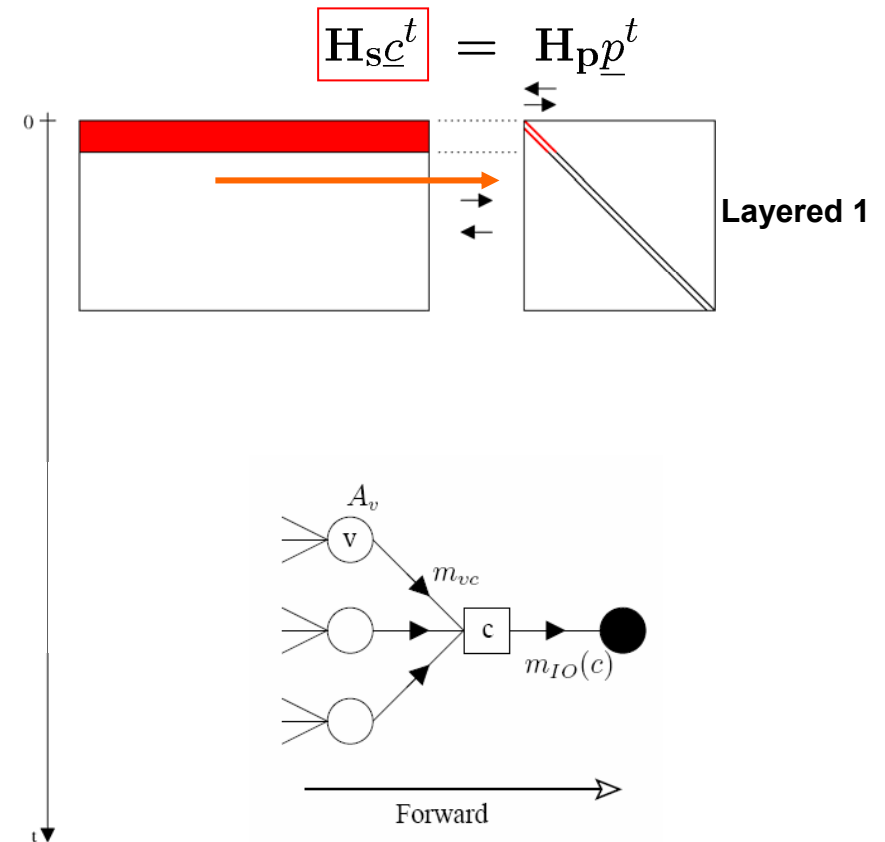
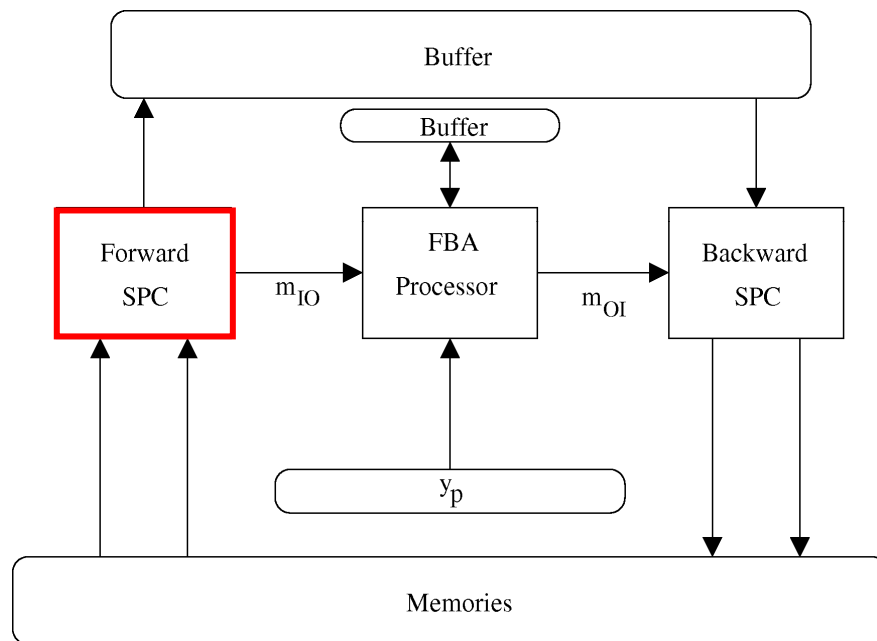
- Consider the decoder as the **dual** of the encoder



- Association with **layer** decoding concept

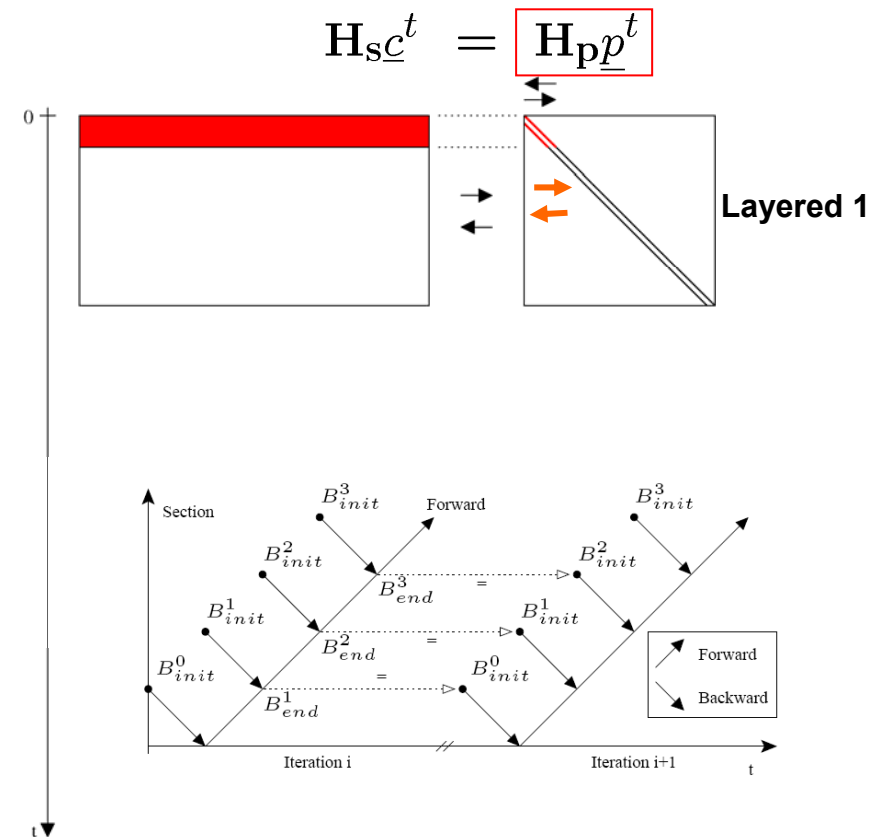
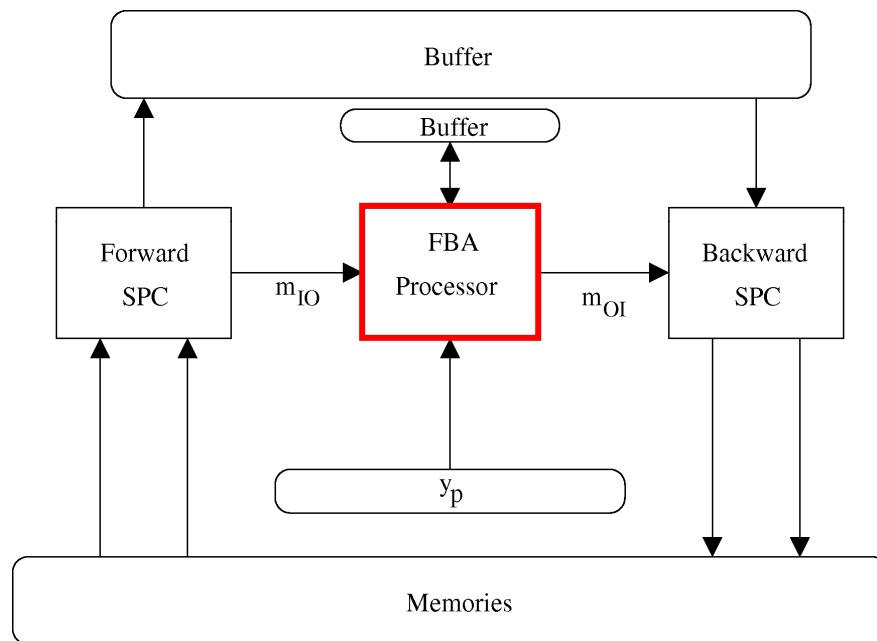
## Illustration of the sequencing

### ■ Turbo Layered BP: Scheduling



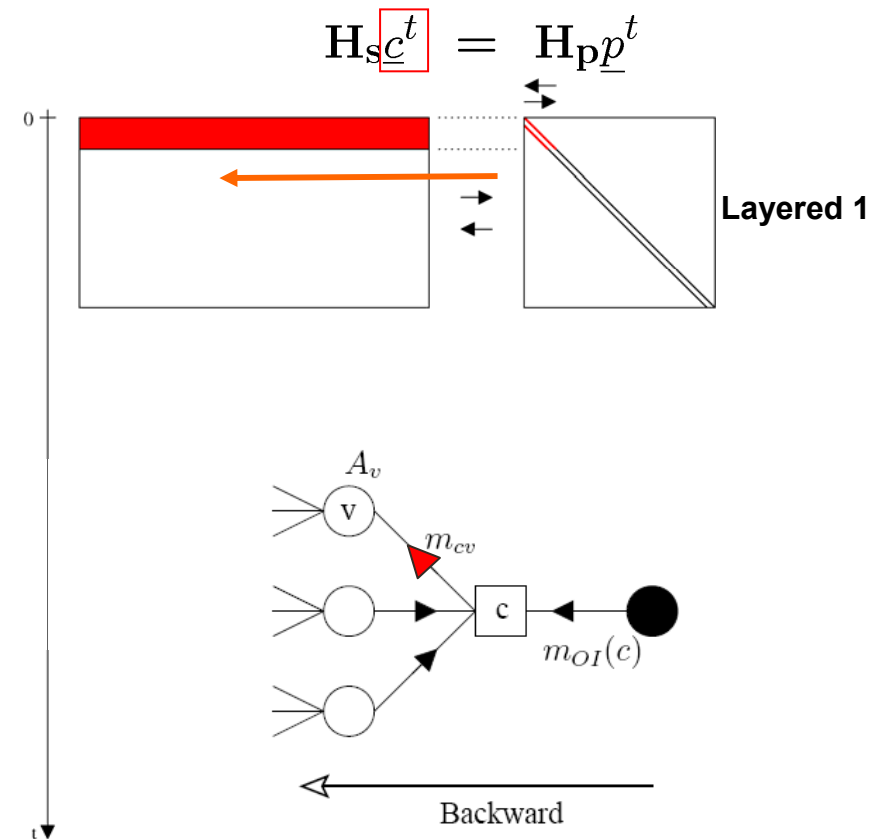
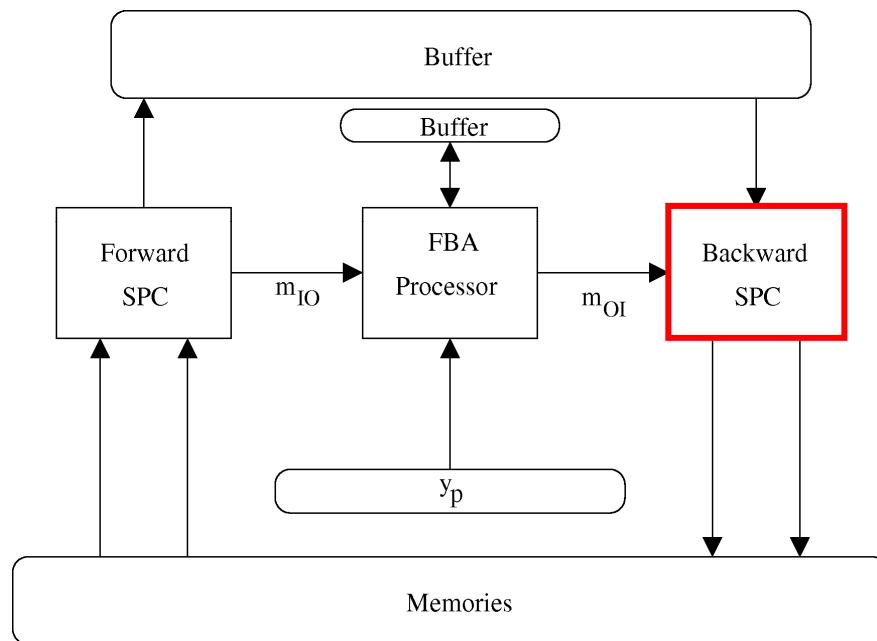
## Illustration of the sequencing

### ■ Turbo Layered BP: Scheduling



## Illustration of the sequencing

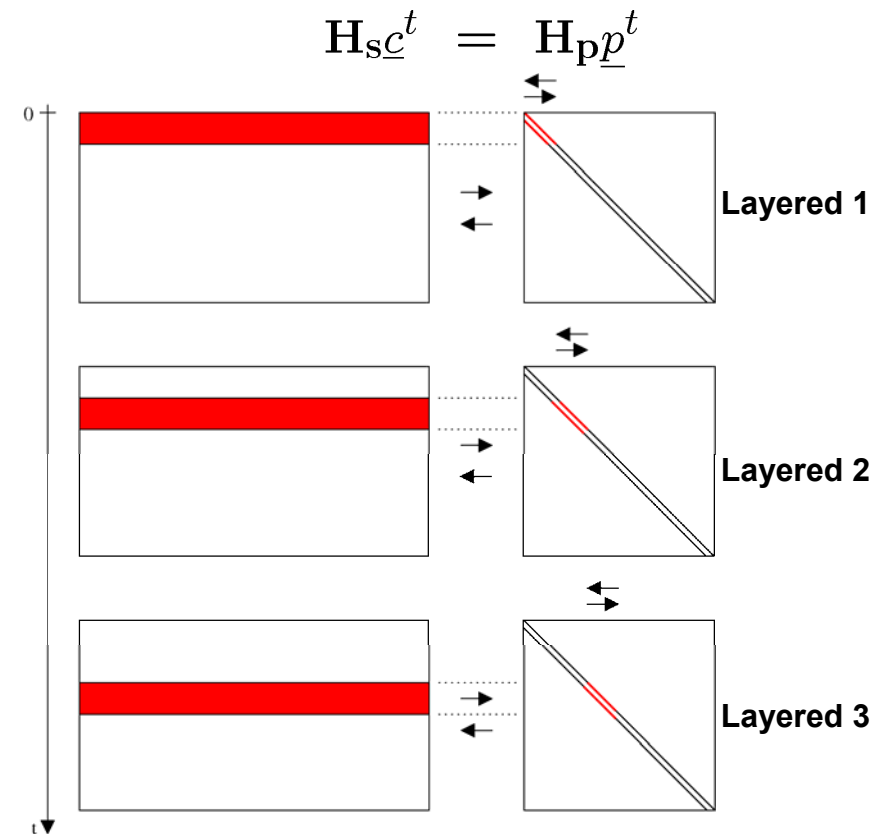
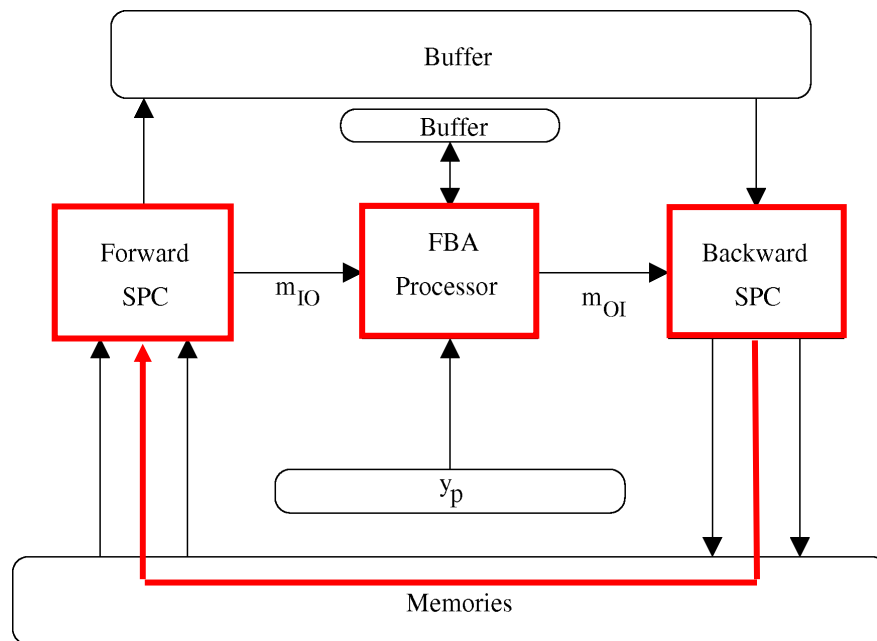
### ■ Turbo Layered BP: Scheduling





## Illustration of the sequencing

### ■ Turbo Layered BP: Scheduling



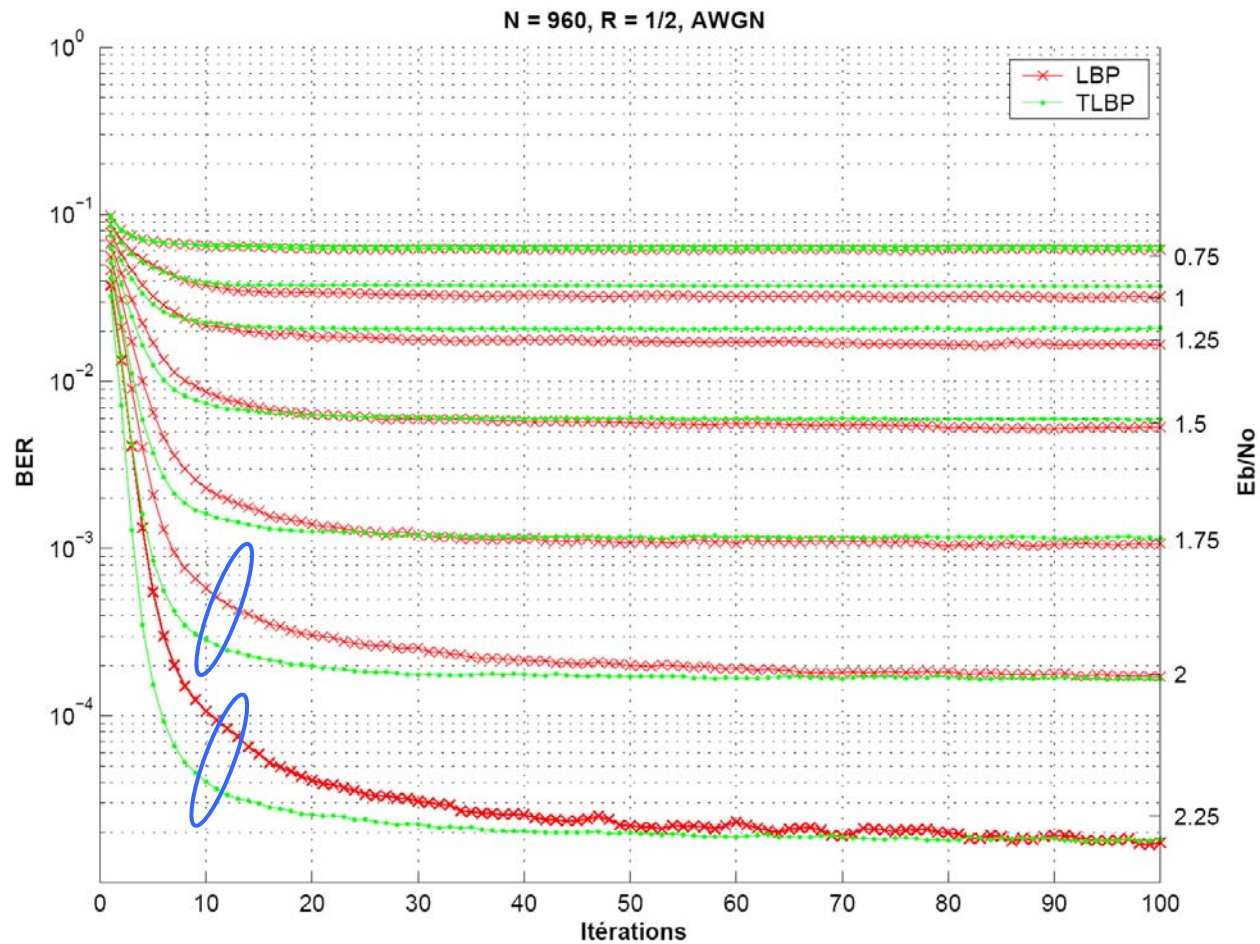
• **Structured LDPC codes**

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

▪ **Decoding Structured LDPC codes**

## Simulation results

### ■ Comparison LBP/TLBP



Rules on permutation coefficients to reach the best possible convergence



## Synthesis

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

QC IRA codes



## Synthesis

### ■ Definition of structured LDPC codes

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

QC IRA codes

### ■ Analysis of code properties

- Distance properties
- Cycles properties

Codes design algorithm



## Synthesis

### ■ Definition of structured LDPC codes

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

QC IRA codes

### ■ Analysis of code properties

- Distance properties
- Cycles properties

Codes design algorithm

### ■ Studies on the decoding of structured LDPC codes

- A priori information on code structure is exploited at the decoder side

Turbo Layered BP



# Decoding architectures for LDPC decoders

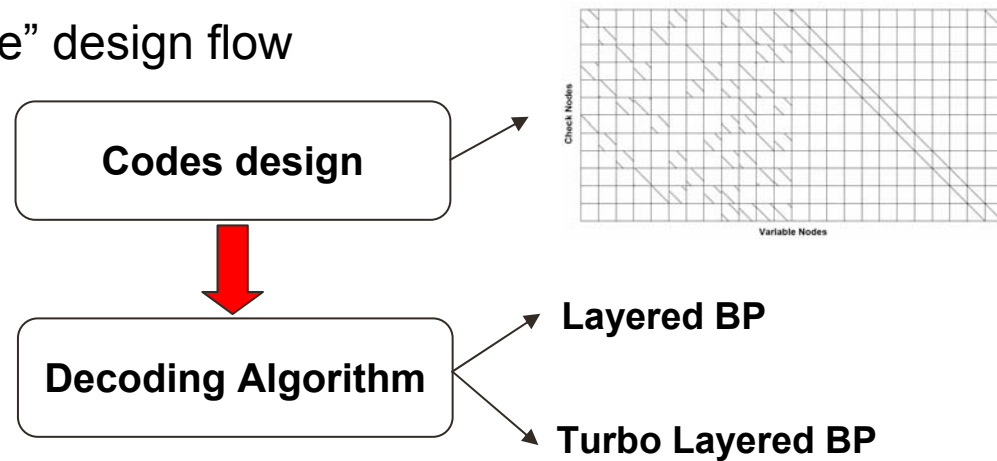
- Conception flow
- Architectures for LBP decoding algorithm
- Architectures for TLBP decoding algorithm
- Conclusions



## Framework

### ■ Methodology

#### ■ “Cross stage” design flow

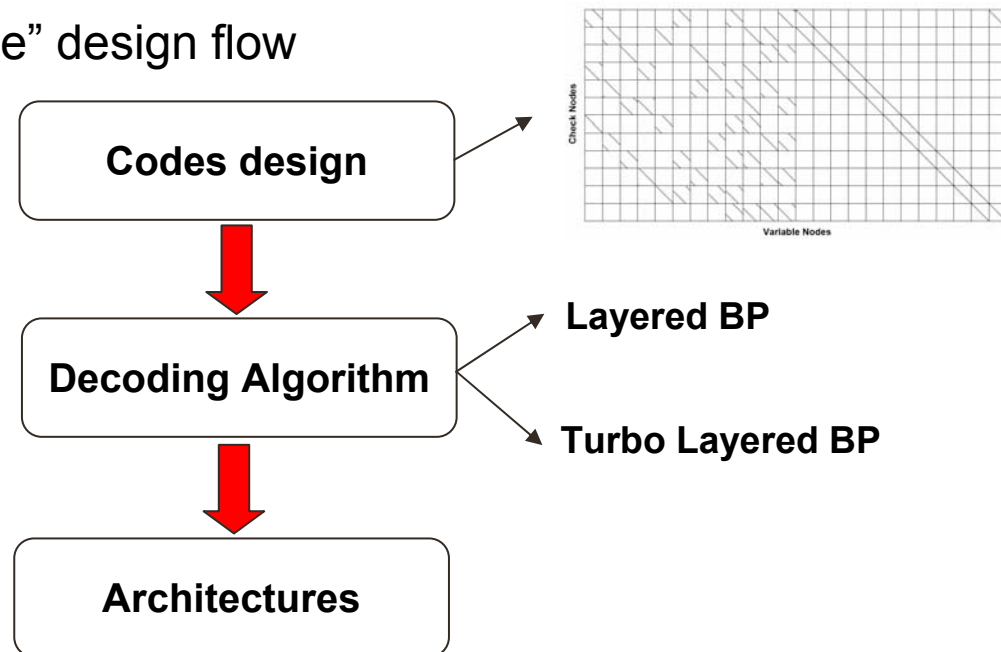




## Framework

### ■ Methodology

#### ■ “Cross stage” design flow



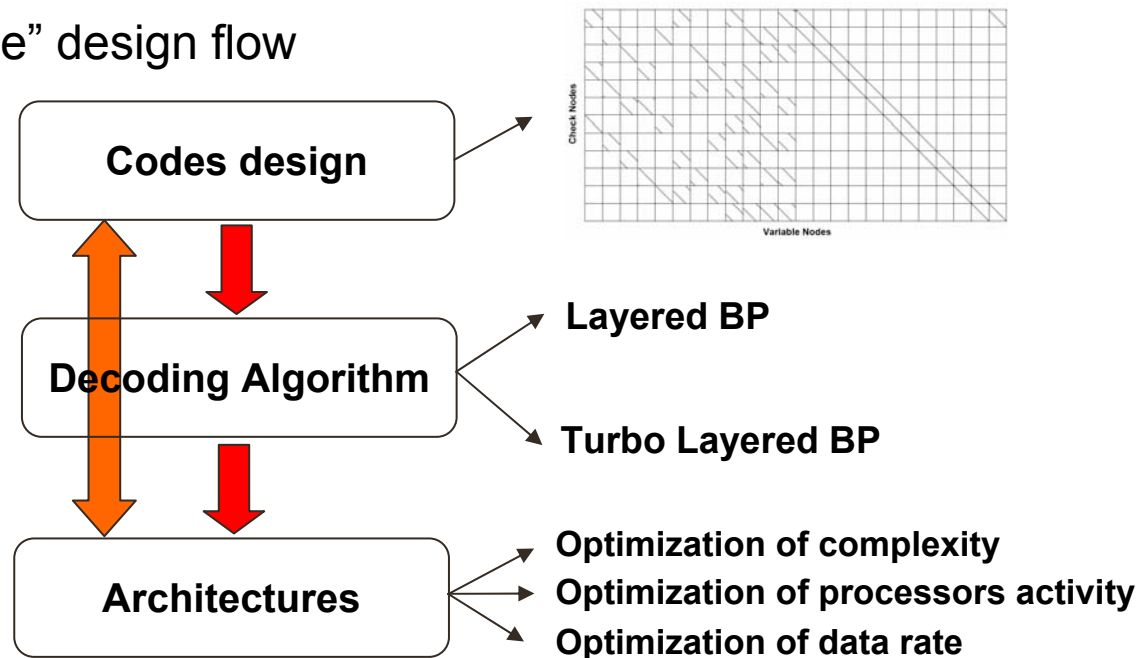




## Framework

### ■ Methodology

#### ■ “Cross stage” design flow

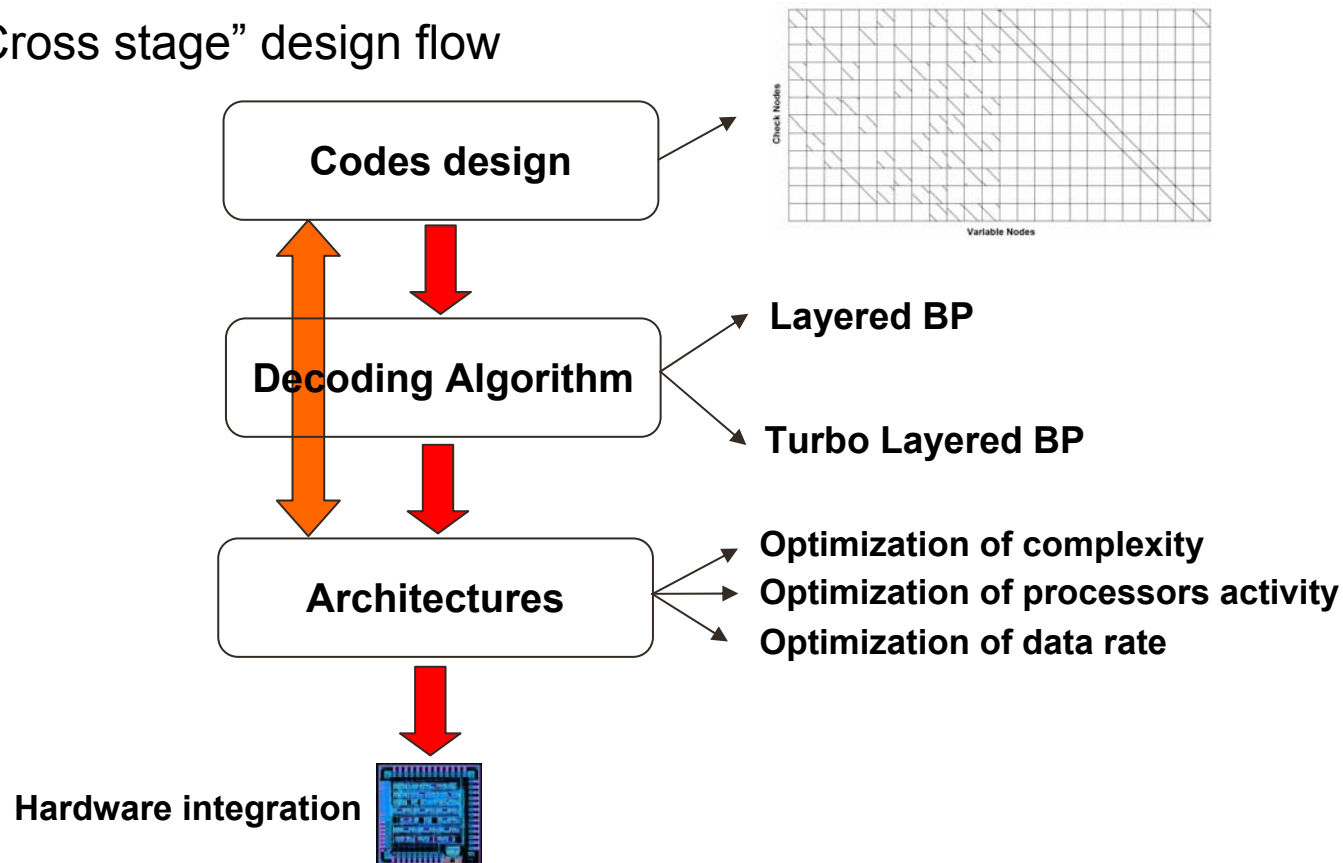




## Framework

### ■ Methodology

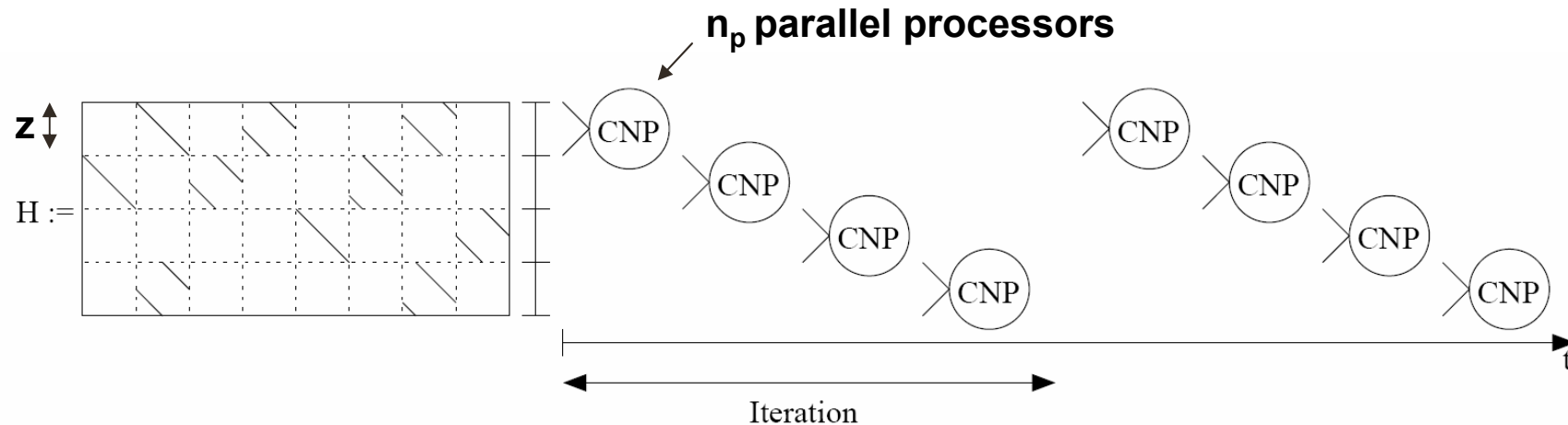
#### ■ “Cross stage” design flow



A **joint** design of both **code** and **decoder architecture** is highly recommended for the design of an efficient system

## Efficiency of the decoder

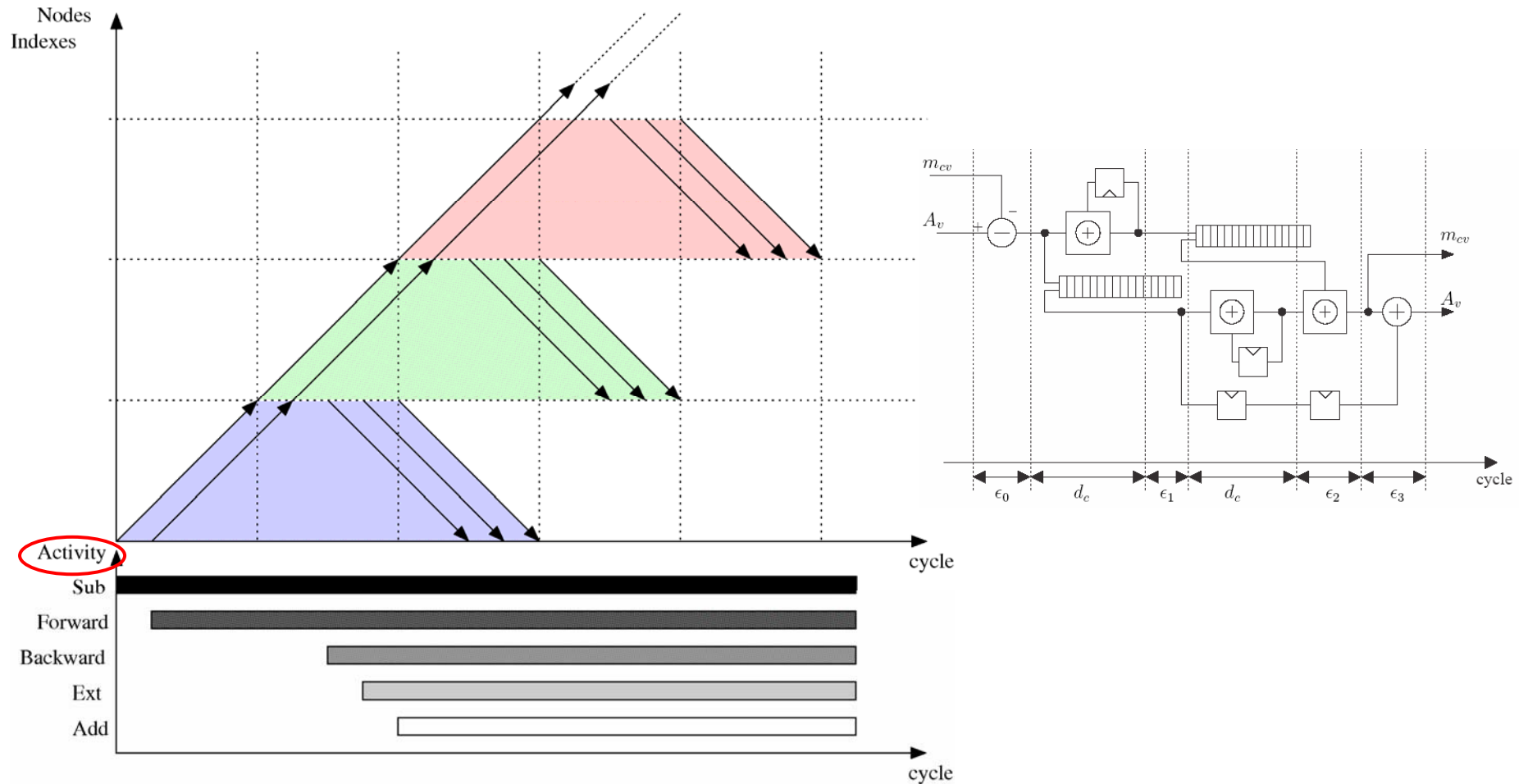
- **Problematic:** Maximize the activity of processors?
  - Layered BP with serial architecture for CNP



- $z$  is the size of a shifted identity matrix
- $n_p$  is the number of processors working in parallel

## Efficiency of the decoder

### ■ **Problematic:** Maximize the activity of processors?





## Configurations studied

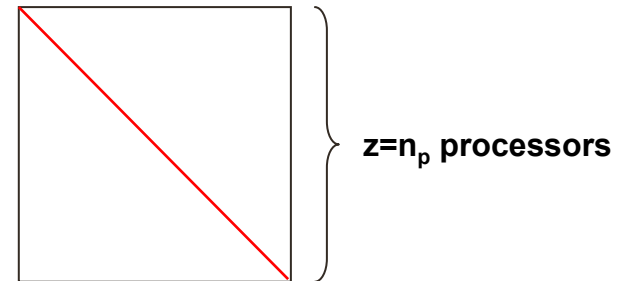
### ■ $n_p = \max z$

#### ■ 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



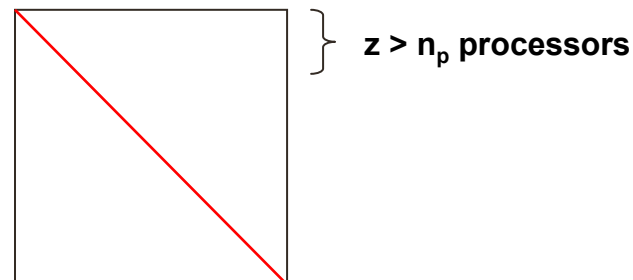
### ■ $n_p < \max z$

#### ■ 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





## Turbo Layered BP: Summary of the work

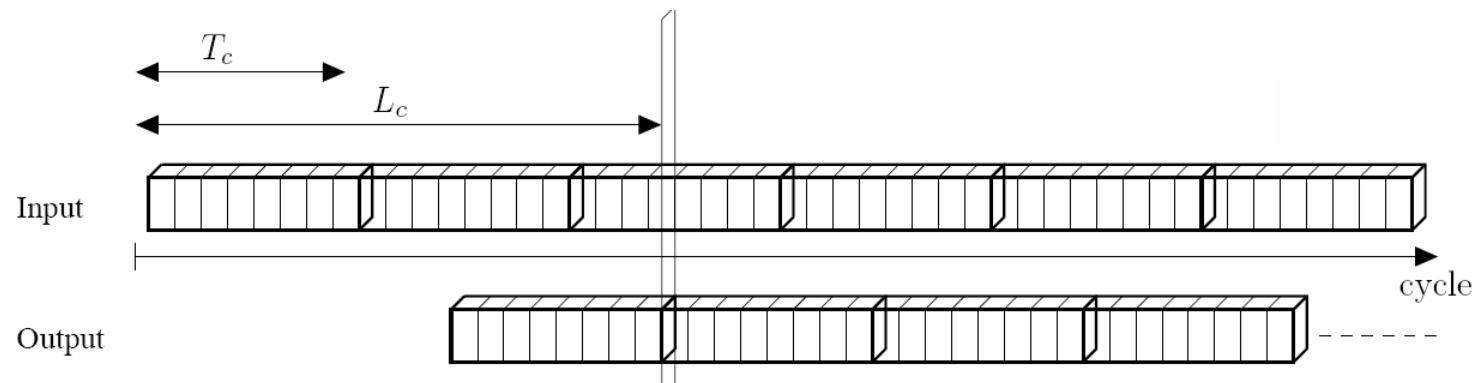
- Various **sequencing** have been studied
  - **Constraints** on the code design

The decoding of a window can start if all the most up-to-date extrinsic information are available



## Turbo Layered BP: Summary of the work

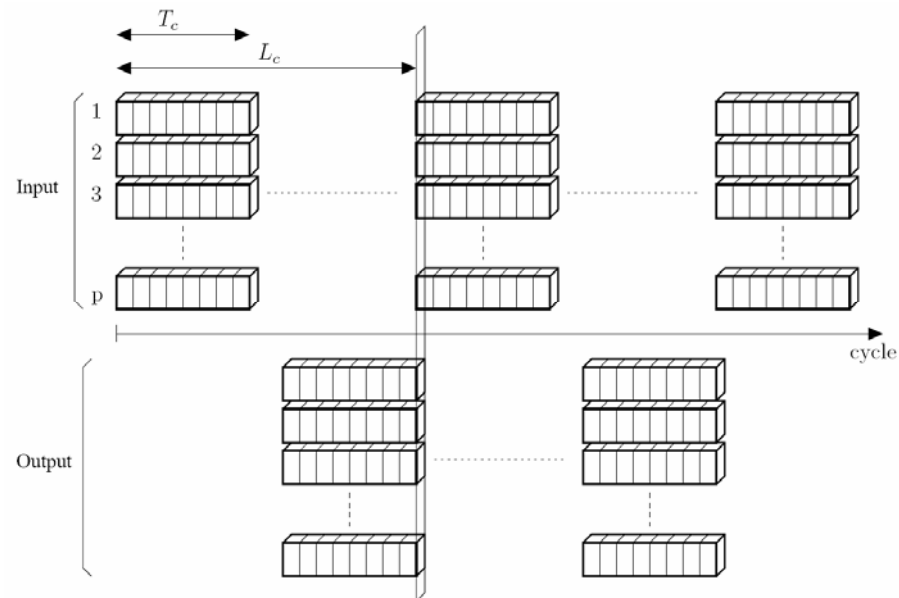
- Various **sequencing** have been studied
  - Serial scheduling (pipelined or not)





## Turbo Layered BP: Summary of the work

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

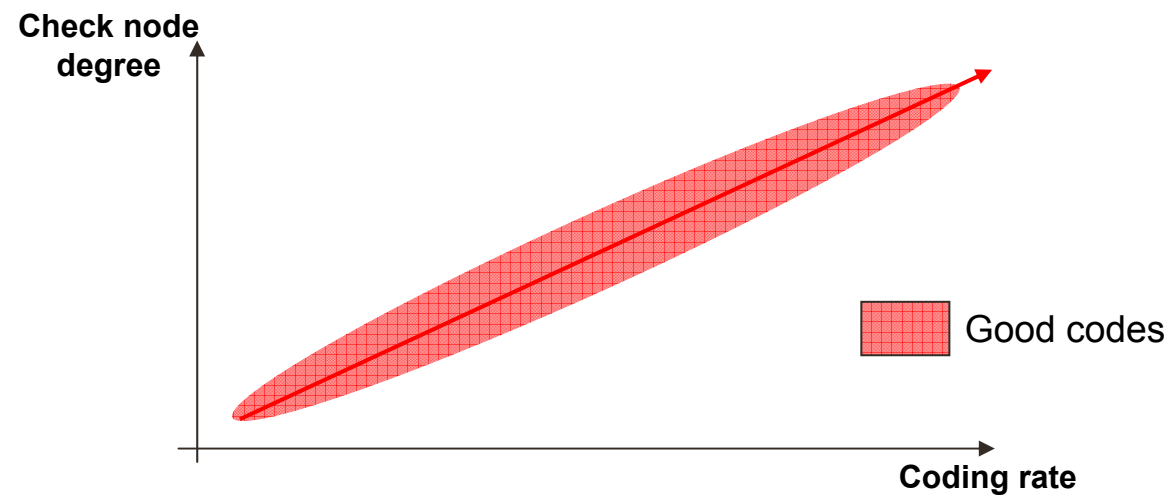




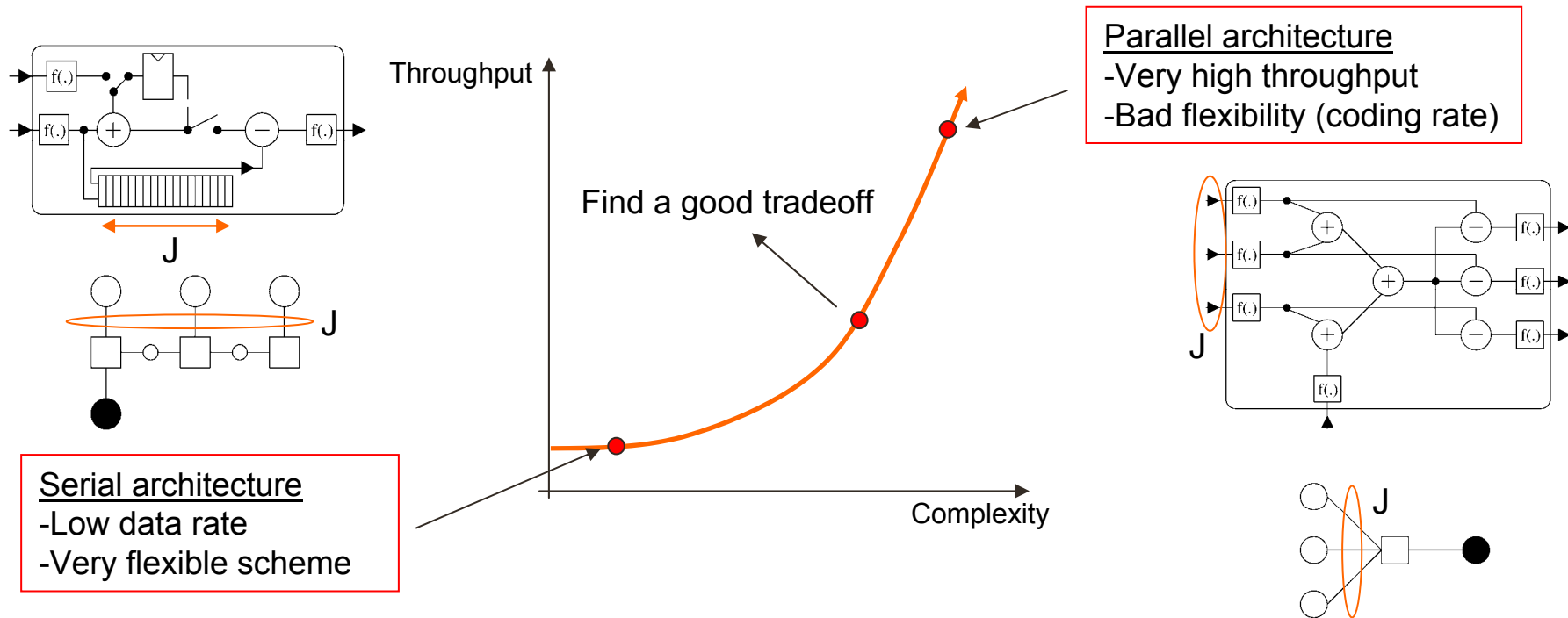


## Turbo Layered BP: Summary of the work

- Various **sequencing** have been studied
  - Serial scheduling (pipelined or not)
  - Parallel scheduling (pipelined or not)
- Definition of an efficient **multi-rate** decoder



## Genericity problematic



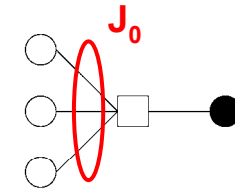
- **Exploit** the structure of the parity check matrix
  - Properties of the dual-diagonal structure



## Proposed solution

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

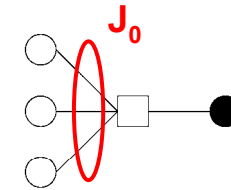
⇒  $J_0$  ones per rows of  $H_s$



## Proposed solution

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

➔  $J_0$  ones per rows of  $H_s$



$$J_0=2$$

$$H = \left[ \begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{array} \right]$$

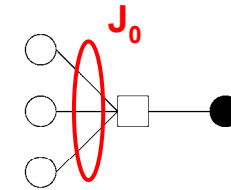
$$H = [H_s \ H_p]$$

## Proposed solution

### ■ Parallel architecture for SPC processors

- $J_0$  messages are processed in parallel

⇒  $J_0$  ones per rows of  $H_s$



$J_0=2$

$$\mathbf{H} = \left[ \begin{array}{cccc|cccc}
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0
 \end{array} \right]$$

$$\mathbf{H}_{eq} = \left\{ \oplus \begin{array}{cccc|cccc}
 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1
 \end{array} \right\}$$

$$\mathbf{H} = \left[ \mathbf{H}_s \quad \mathbf{H}_p \right]$$



## Proposed solution

### ■ Parallel architecture for SPC processors

- $J_0$  messages are processed in parallel

$$\mathbf{H} = \left[ \begin{array}{cccc|cccc}
 \textcircled{1} & 0 & \textcircled{1} & 0 & \textcircled{1} & 0 & \textcircled{1} & 0 & 1 & 0 & 0 & 0 \\
 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1
 \end{array} \right]$$

$$\mathbf{H}_{\text{eq}} = \left\{ \oplus \left[ \begin{array}{cccc|cccc}
 \textcircled{1} & 0 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & \textcircled{1} & 0 & \textcircled{1} & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & \textcircled{1} & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & \textcircled{1} & 1 & 0 & 0 & 0 \\
 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \textcircled{1} & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 1 & 0 \\
 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \textcircled{1} & 0 \\
 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 1
 \end{array} \right\}$$

### ■ 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



## Synthesis

- "Architecture driven" approach
  - Joint design of code and decoder architectures



## Synthesis

- "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  $n_p < \max z$
  - Some open issues
    - Flexible permutation network (barrel shifter)





## Synthesis

- "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  $n_p < \max z$
  - 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



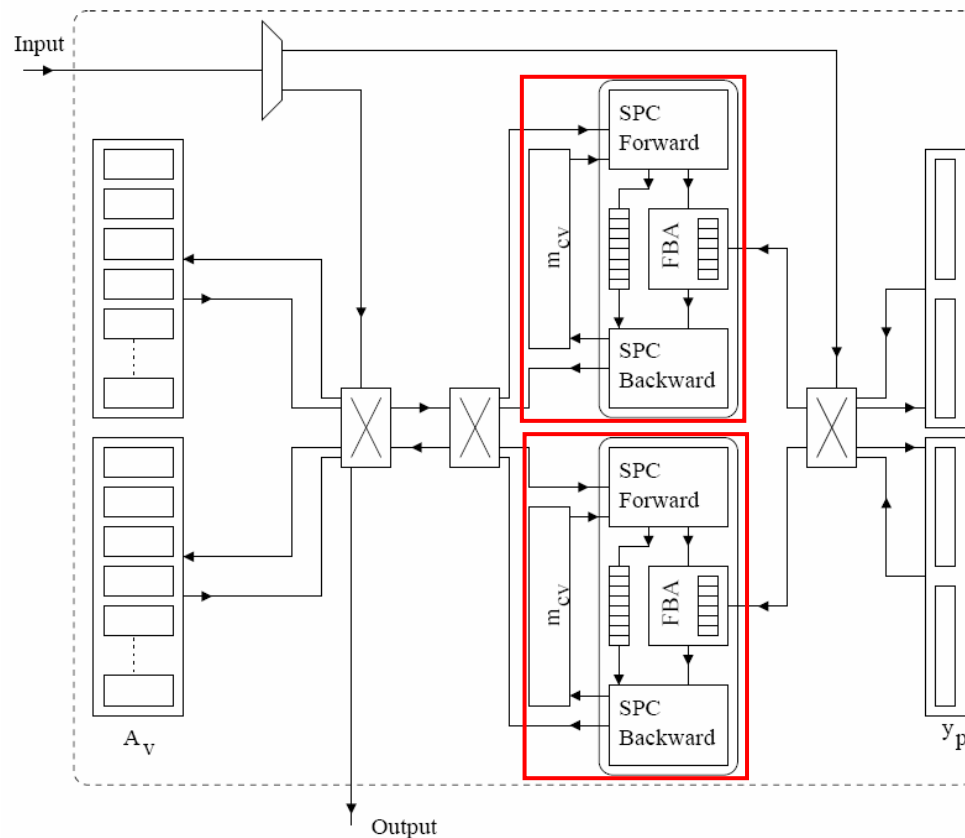
# FPGA implementation of LDPC coder/decoder

- Implementation options
- Quantization
- Complexity considerations
- Simulation results
- Conclusion



## Options

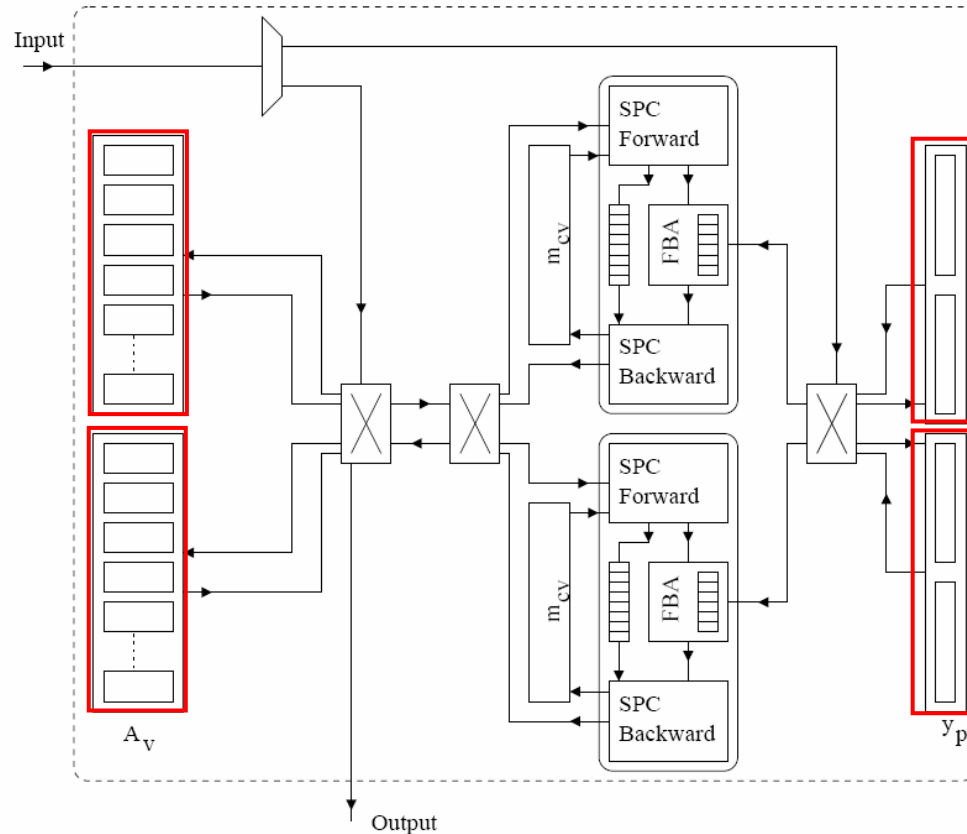
- Turbo Layered BP algorithm
  - With and without pipeline



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

## Options

- 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

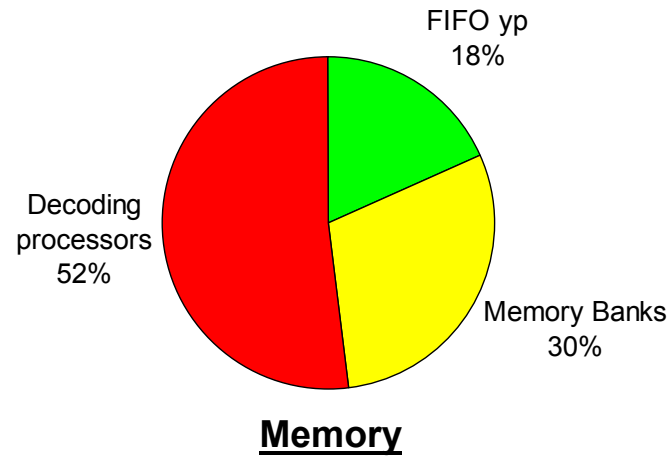
- Continuous to discrete domain
  - Influence the performance
    - Lower granularity
    - Introduction of erasures
  - Influence the complexity of the decoder
    - Size of the memories
    - Size of internal data path
    - Complexity of the basic operators (+, -, < ...)
  
- What is the good **trade off** between **performance** and **complexity**?



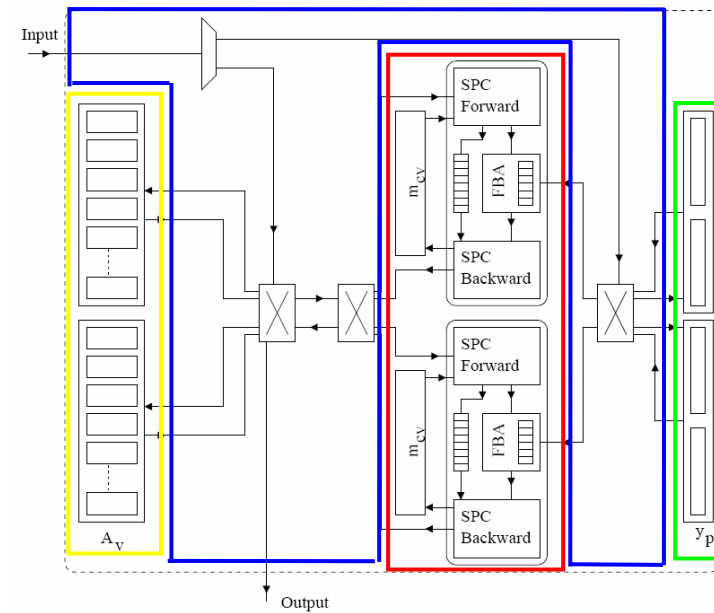
## Problematic

- 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

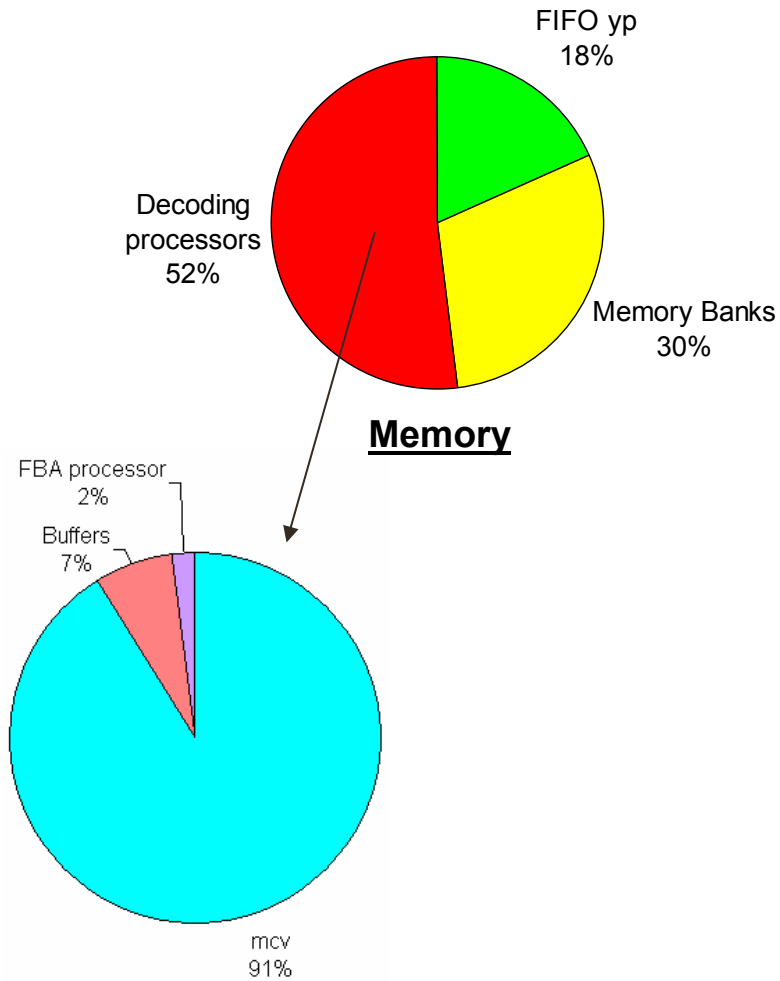
# FPGA integration



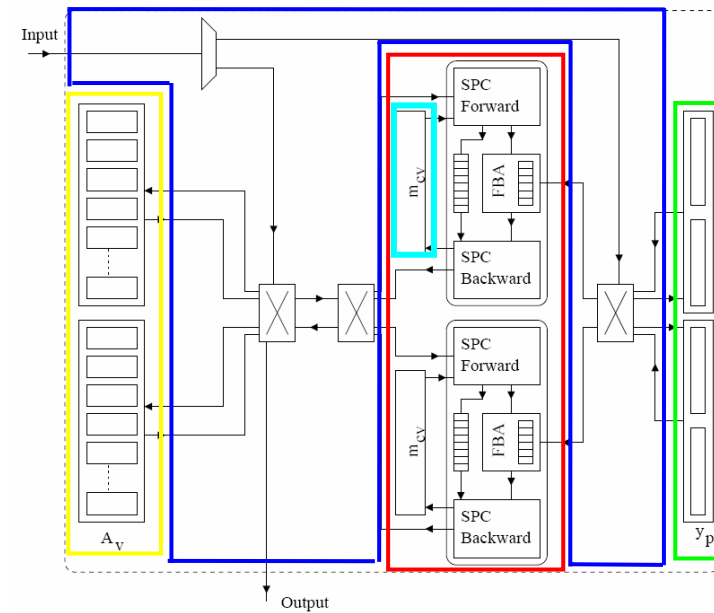
**ALTERA STRATIX  
 EP1S80 - C6**



# FPGA integration

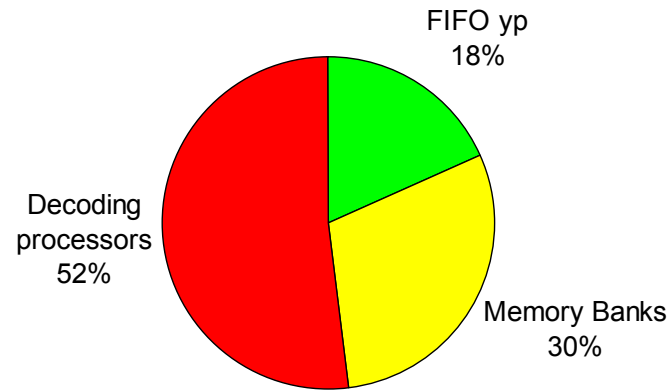


**ALTERA STRATIX  
 EP1S80 - C6**

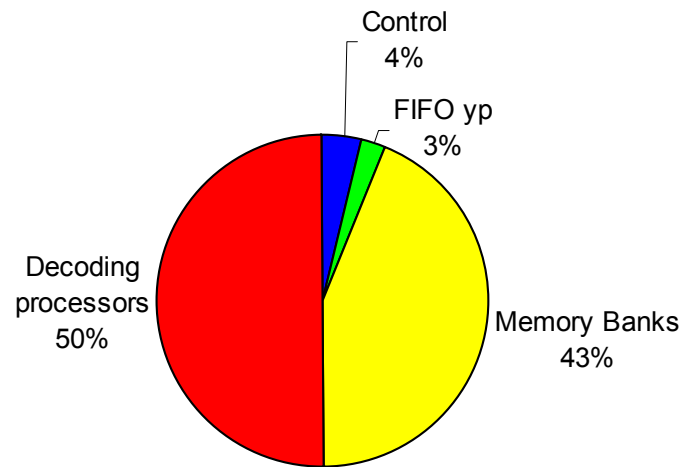




# FPGA integration



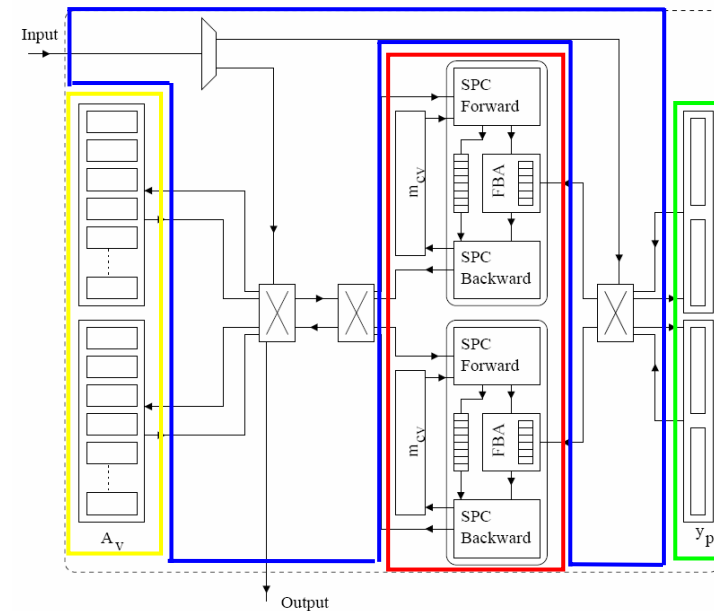
## Memory



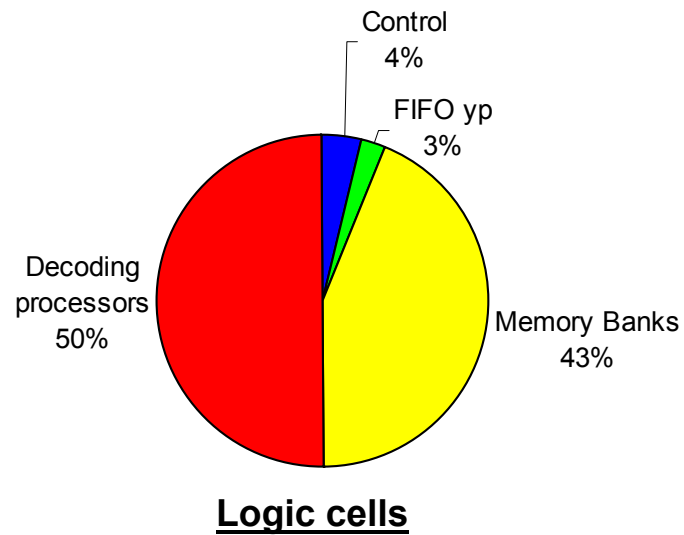
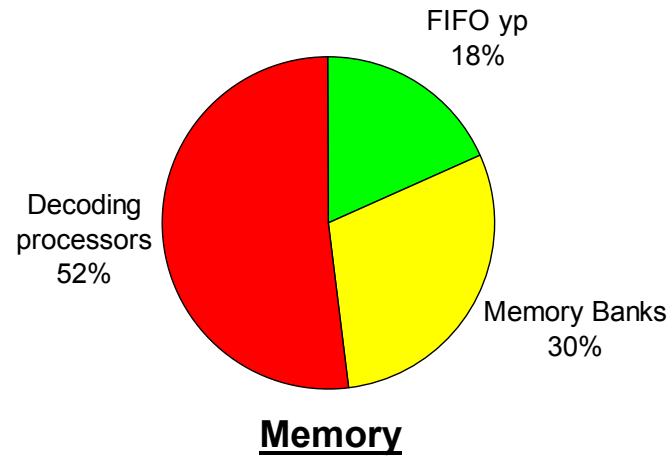
## Logic cells



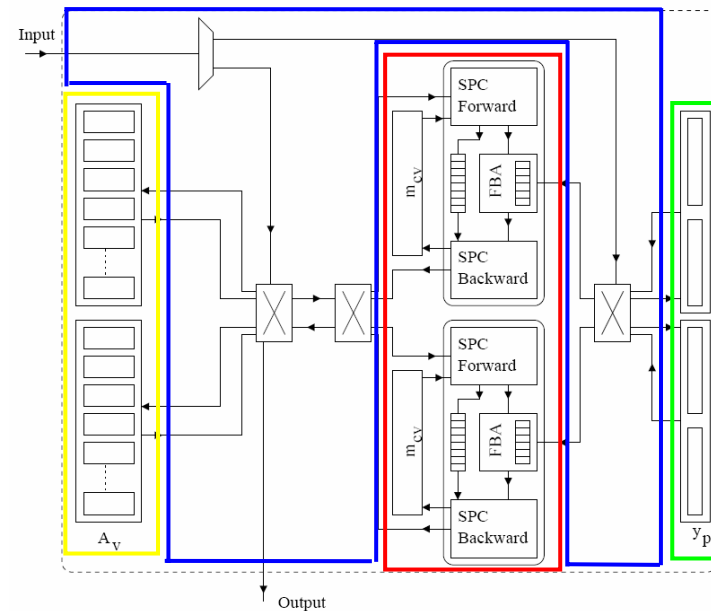
**ALTERA STRATIX  
 EP1S80 - C6**



# FPGA integration



**ALTERA STRATIX  
 EP1S80 - C6**

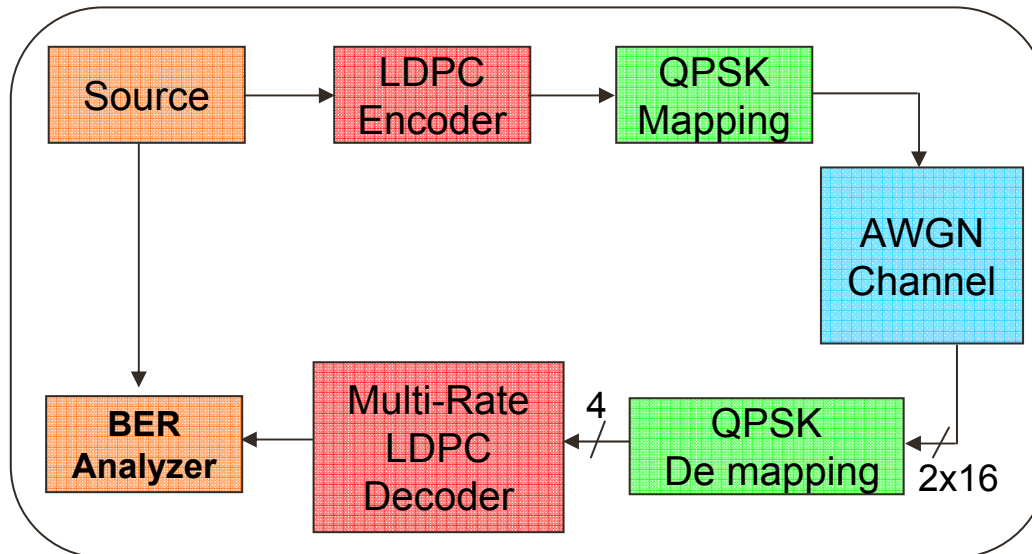


**Drawback of a double input buffer**



## Simulation context

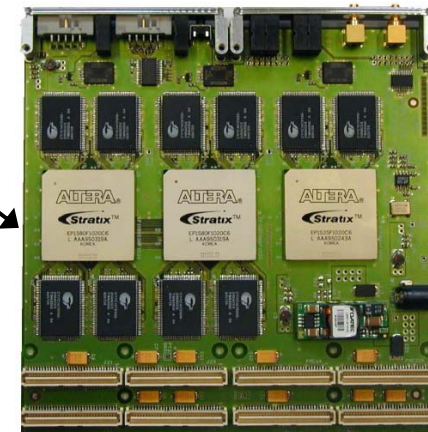
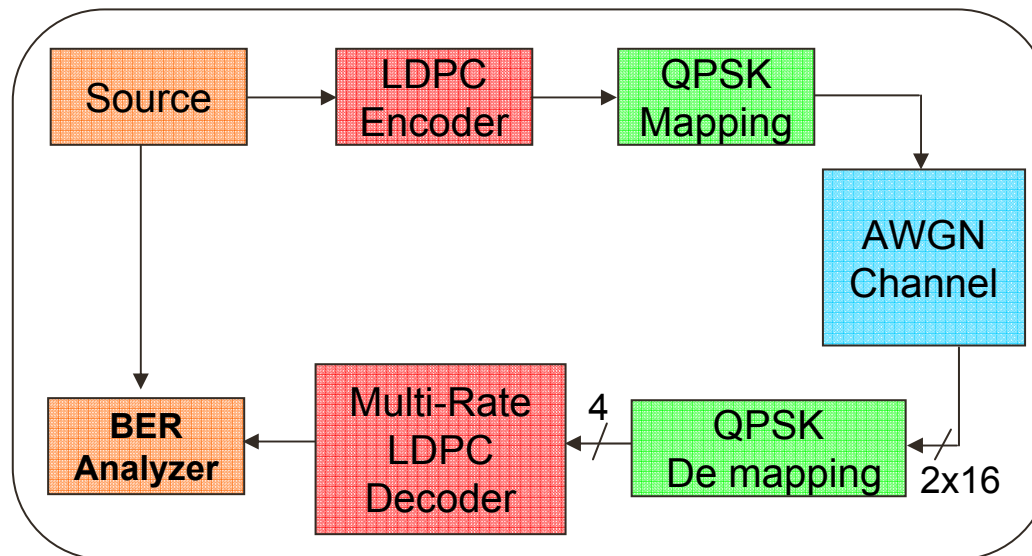
- **FPGA Hardware simulation chain**
  - ALTERA Stratix EPS80-C6



- **Source**
  - PRBS 20
- **AWGN Channel**
  - Box Muller algorithm
- **LDPC decoder/coder**
  - 4 loaded codes

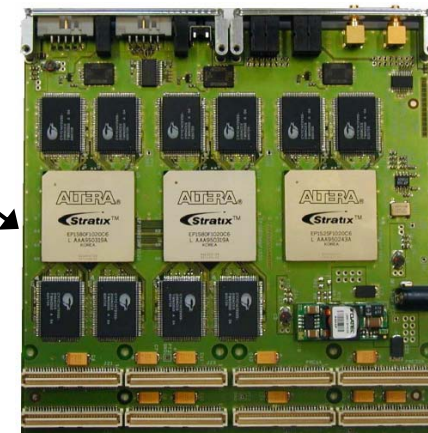
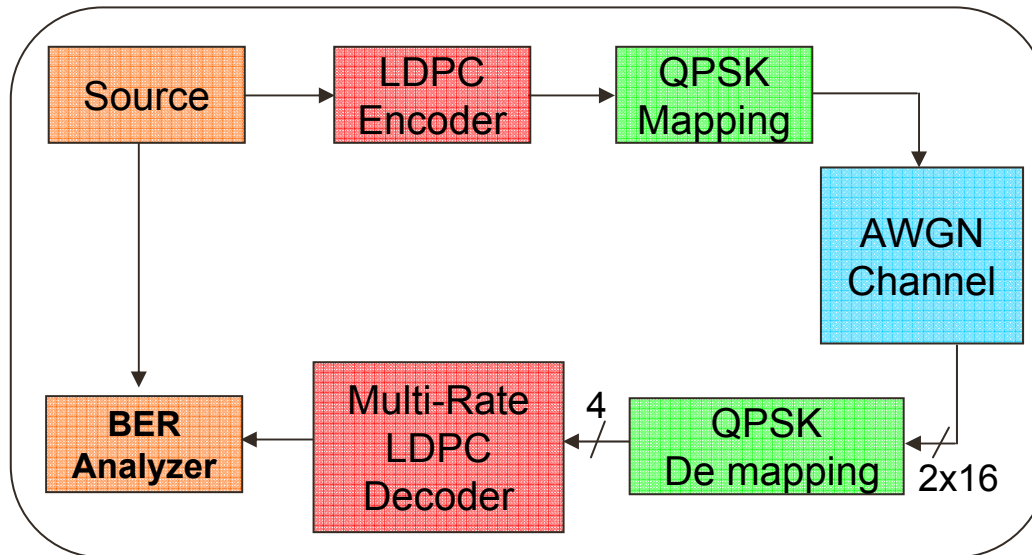
## Simulation context

- **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

**LDPC Evaluation**

**Configurations**

**Platform**  
 Starting boards detection ...  
 PLDA agent detected  
 CPCISYS-T1 card detected with LDPC application

Boards Detection  LDPC application

**Parameters**

Code: 1 Packet size: 768  
 Nb of iter.: 10 Rate: 1/2

**Messages**  
 Reg = 232 -- Eb/N0cor = 3.01 dB  
 Reg = 226 -- Eb/N0cor = 3.24 dB

Validation  Done

START STOP

EXIT

**Results**

**Transmission Analysis**

Data rate: 7.258 Mbit/s Bit error rate(1s): 0.00e+00  
 Reset Synchronized

Bit errors: 3 BER: 3.98e-08  
 Packet errors: 1 PER: 1.03e-06

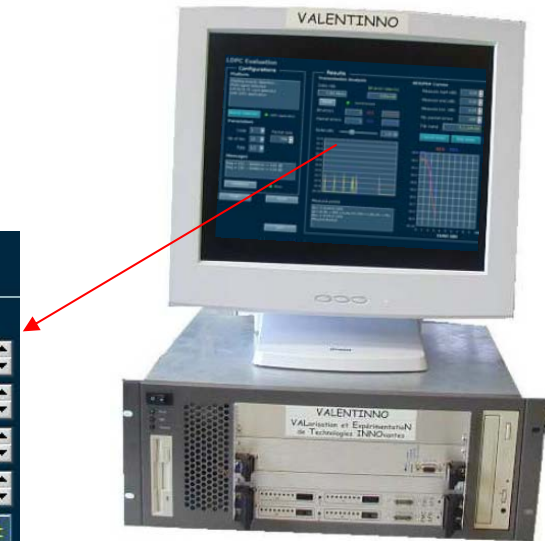
Eb/N0 (dB): 3.25 dB

**BER/PER Curves**

Measure start (dB): 0.00  
 Measure end (dB): 5.00  
 Measure incr. (dB): 0.25  
 Nb. packet errors: 100  
 File name: R\_1\_10it.dat

Launch meas. Stop meas.

**Measure points**  
 Nov 9 10:46:41 2006  
 @ 3.24 dB -> BER = 4.20e-09 | PER = 1.08e-06 -> Thu  
 Nov 9 10:50:22 2006  
 Measure finished



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

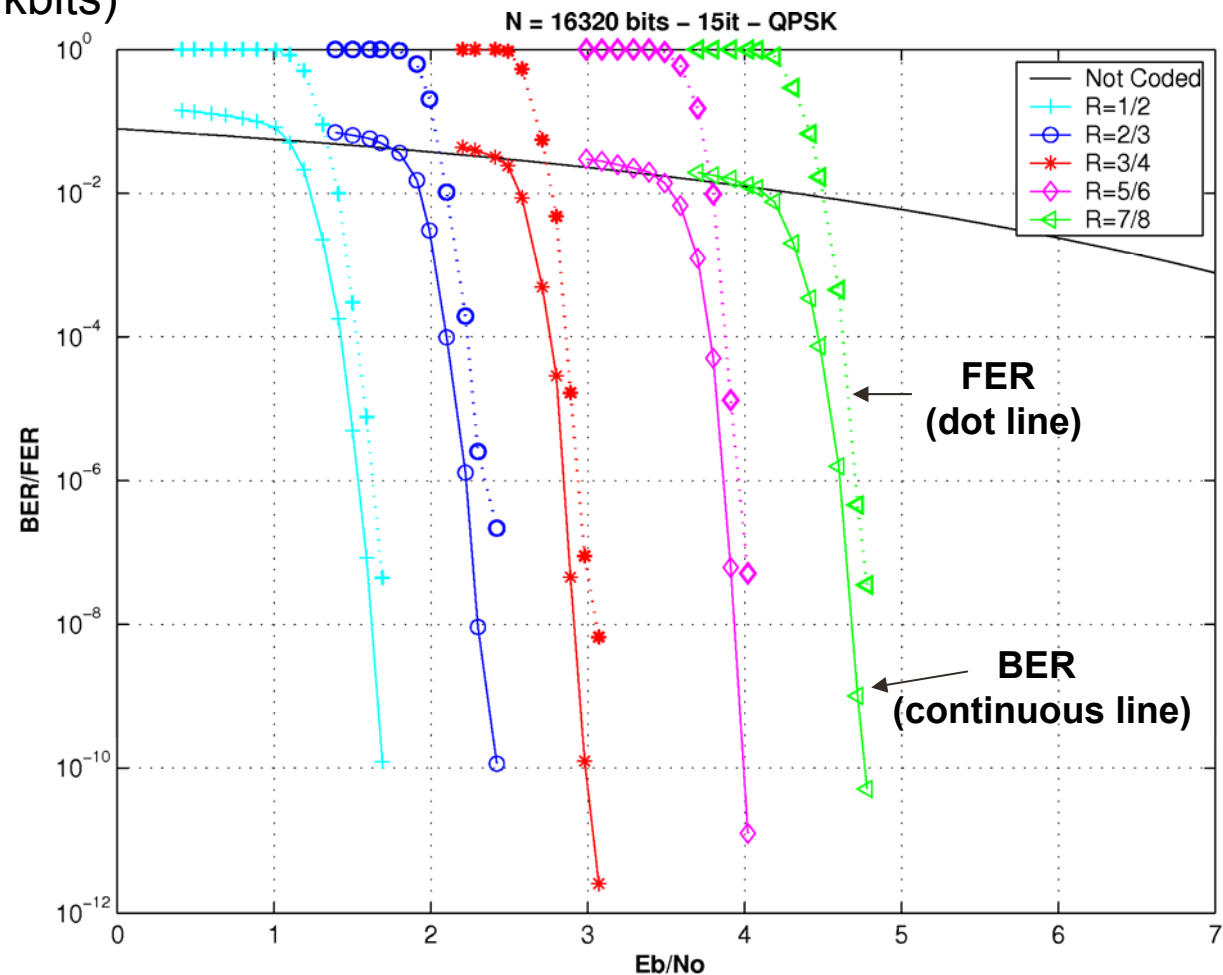
## Applications

### ■ Broadcast context

- Large block size ( $\approx 16$  kbits)

### ■ Parameters

- AWGN, QPSK
- $Q_c = 4$  bits ( $\pm 7$ )
- **15** it TLBP





## Applications

### ■ Broadcast context

- Large block size ( $\approx 16$  kbits)

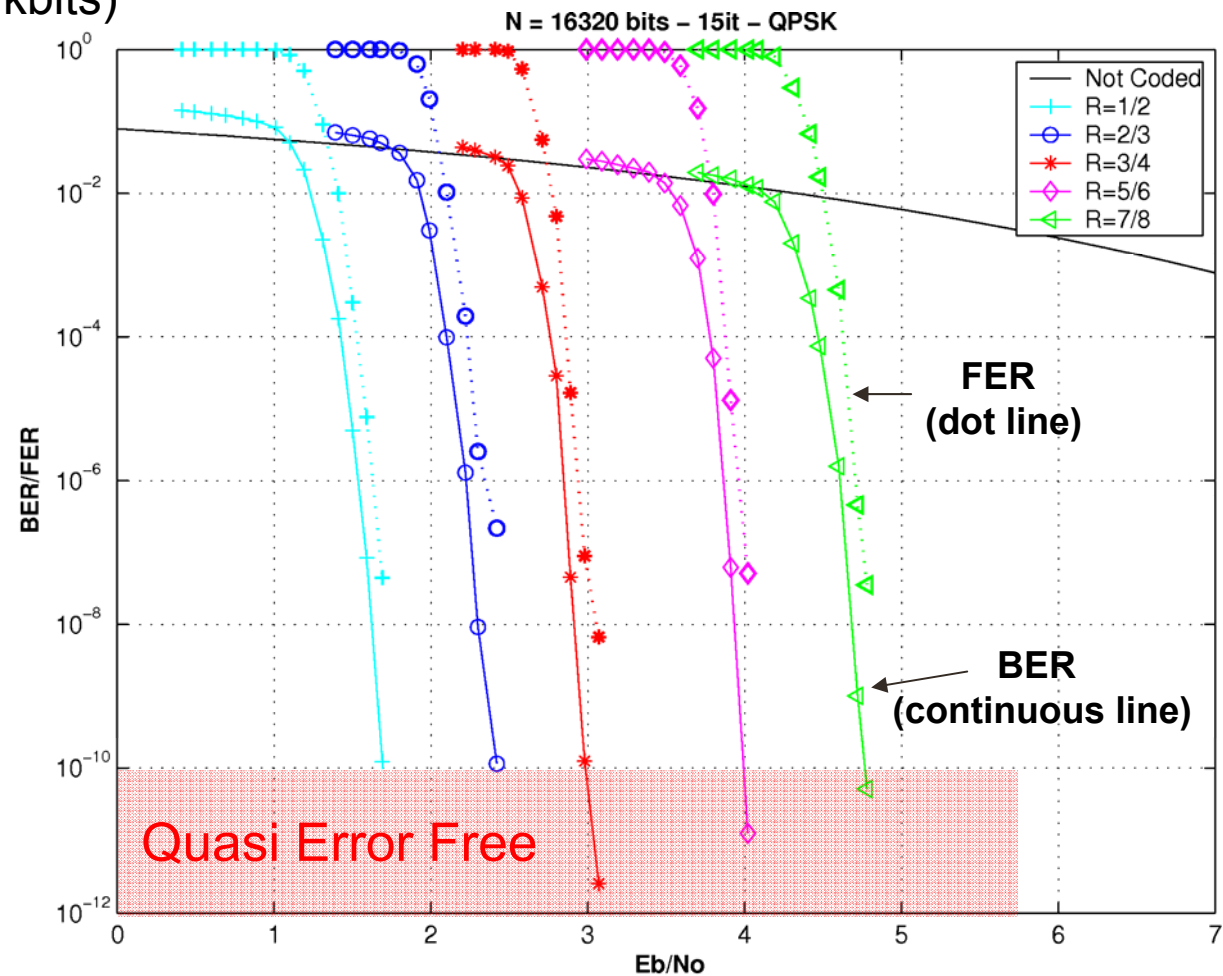
### ■ Parameters

- AWGN, QPSK
- $Q_c = 4$  bits ( $\pm 7$ )
- **15** it TLBP

No early error floor

### ■ Validation of both

- Code design algorithm
- Quantization strategy







## Duo binary Turbo-Codes vs LDPC

### ■ 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), mm<sup>2</sup> on x μm for ASIC

### ■ Proposed scheme

- Similar context (Valentinno)

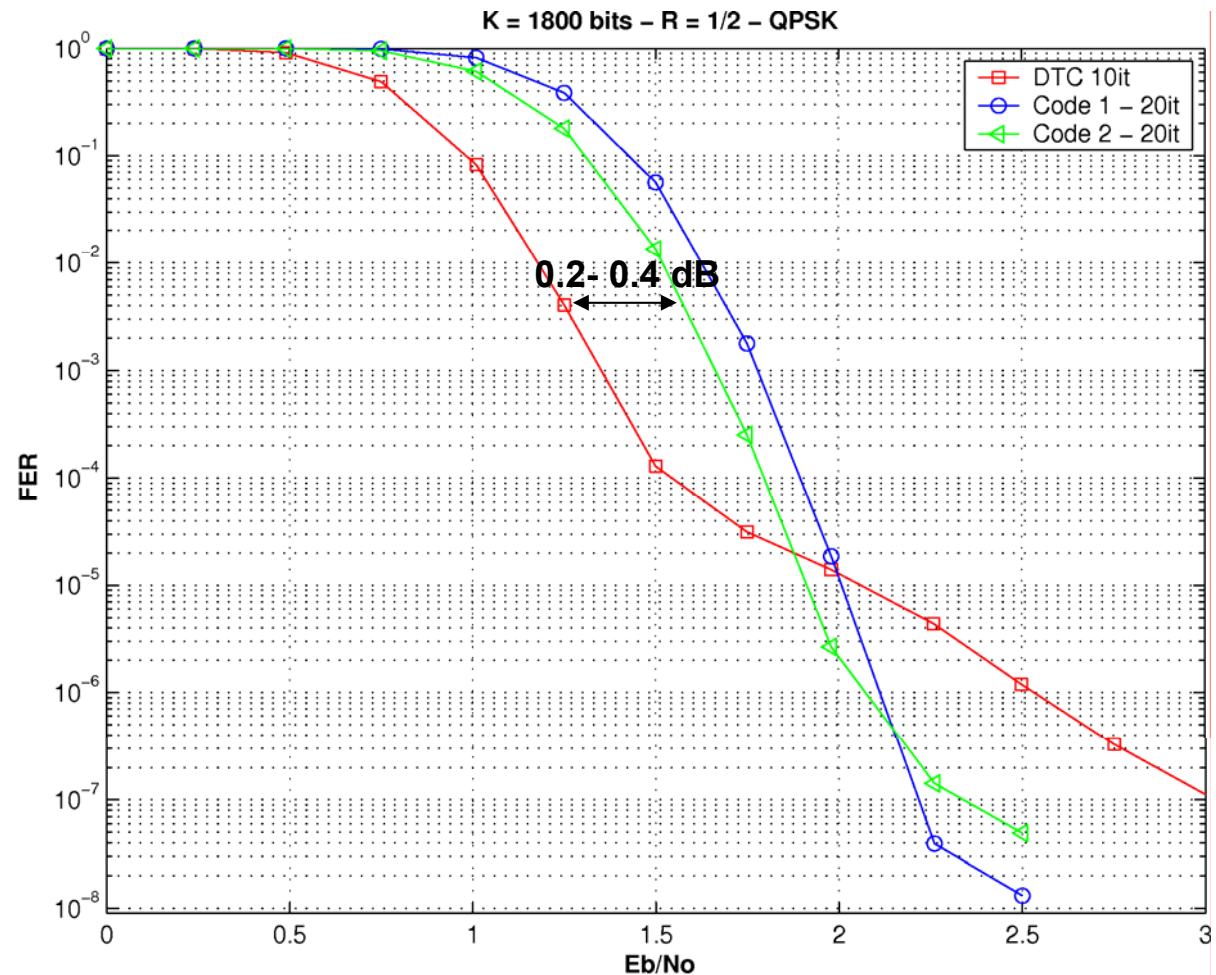
# Duo binary Turbo-Codes vs LDPC

## ■ Performance

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

## ■ Context

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



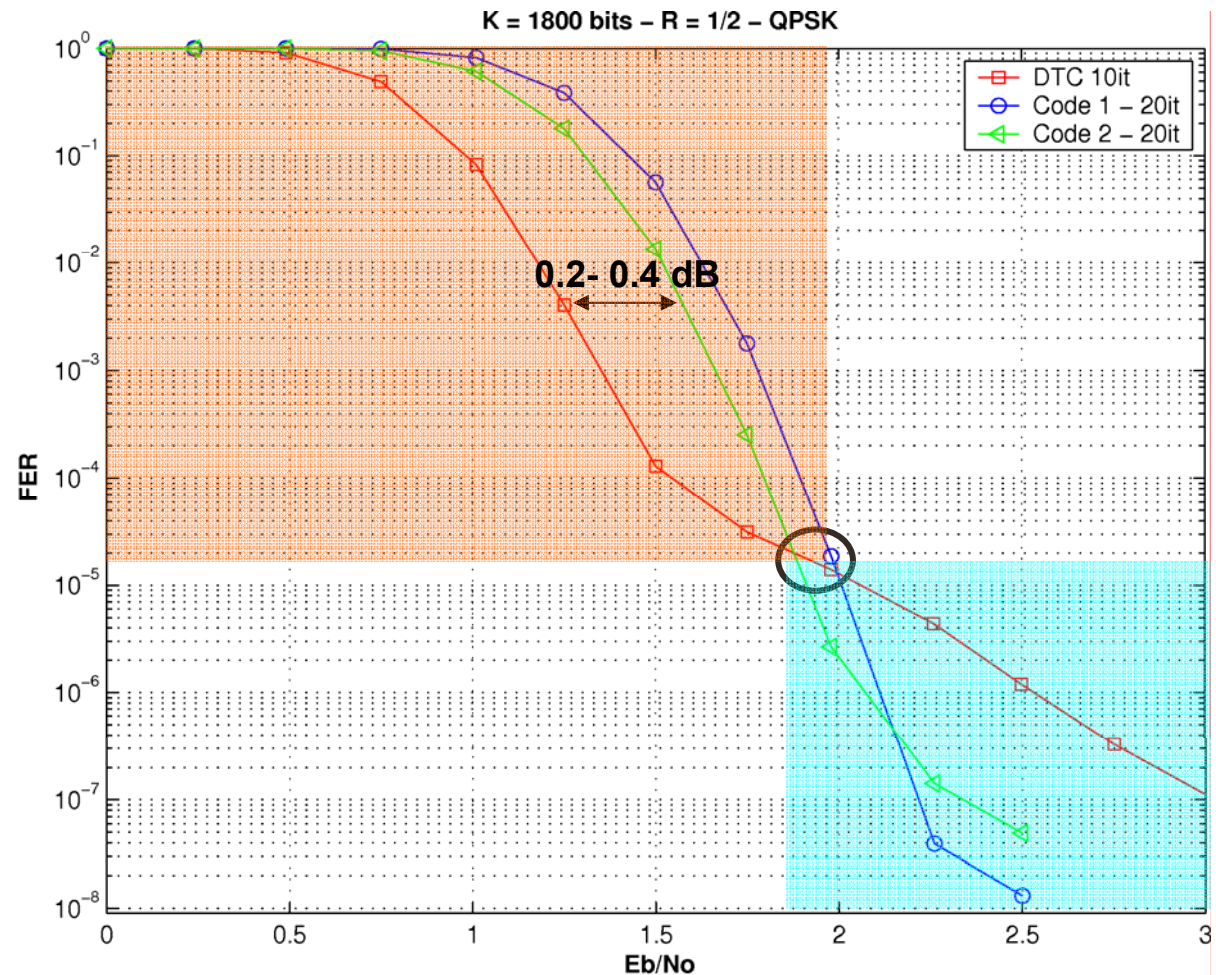
# Duo binary Turbo-Codes vs LDPC

## ■ Performance

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

## ■ Context

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



## Duo binary Turbo-Codes vs LDPC

### ■ Implementation aspects

#### ■ FPGA – ALTERA Stratix EP1S80F C6

	Logic cells	Memory (ko)	
TC1000 2x	5503	8.147	↔ Duo binary TC
LDPC studied ↔	Pipeline	7320	↔
	Without Pipeline	6967	
			x 1.33
			x 1.34



## Duo binary Turbo-Codes vs LDPC

### ■ Implementation aspects

#### ■ FPGA – ALTERA Stratix EP1S80F C6

TC1000 2x	Logic cells	Memory (ko)	↔ Duo binary TC
	5503	8.147	

LDPC studied ↔	Pipeline	Logic cells	Memory (ko)
	Without Pipeline	7320	10.91
		6967	10.53

**x 1.33**
**x 1.34**

#### ■ Decoding throughput

- At same data rates

$$it_{LDPC} \simeq it_{DTC} \quad \text{Without pipeline}$$

$$it_{LDPC} \simeq 2 it_{DTC} \quad \text{pipeline}$$





## Duo binary Turbo-Codes vs LDPC



### ■ 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




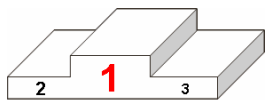
## Duo binary Turbo-Codes vs LDPC

### ■ 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



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



## Synthesis

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



## Synthesis

- 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



## Synthesis

- 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



## Contributions

- 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



## 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)



## 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)

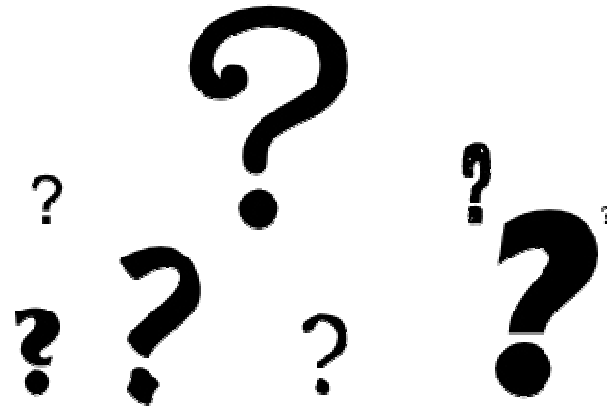


## Perspectives

- **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

## Bibliography

- [Tanner99] - R. M. Tanner, “On quasi-cyclic repeat-accumulate codes”, in *Proc. of the 37<sup>th</sup> Allerton Conference*, 1999.
- [Forney01] - G. D. Forney, “Codes on graphs : Normal realizations”, *IEEE Transactions on Information Theory*, Feb 2001.
- [Chung01] - S.-Y. Chung, T. Richardson, and R. Urbanke, “Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation”, *IEEE Transactions on Information Theory*, vol. 47, Feb 2001.
- [Hu01] - X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, “Progressive edge-growth tanner graphs”, *IEEE Global Telecommunications Conference*, vol. 2, Nov 2001.
- [Richardson01] - T. Richardson, A. Shokrollahi, and R. Urbanke, “Design of capacity approaching irregular low-density parity-check codes”, *IEEE Transactions on Information Theory*, vol. 47, Feb 2001.
- [Tian03] - T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, “Construction of irregular LDPC codes with low error floors”, *IEEE International Conference on Communications*, June 2003.
- [Richardson03] - T. Richardson, “Error floors of LDPC codes”, *41st Annual Allerton Conference on Communications, Control, and Computing*, Oct 2003.



# Annexes



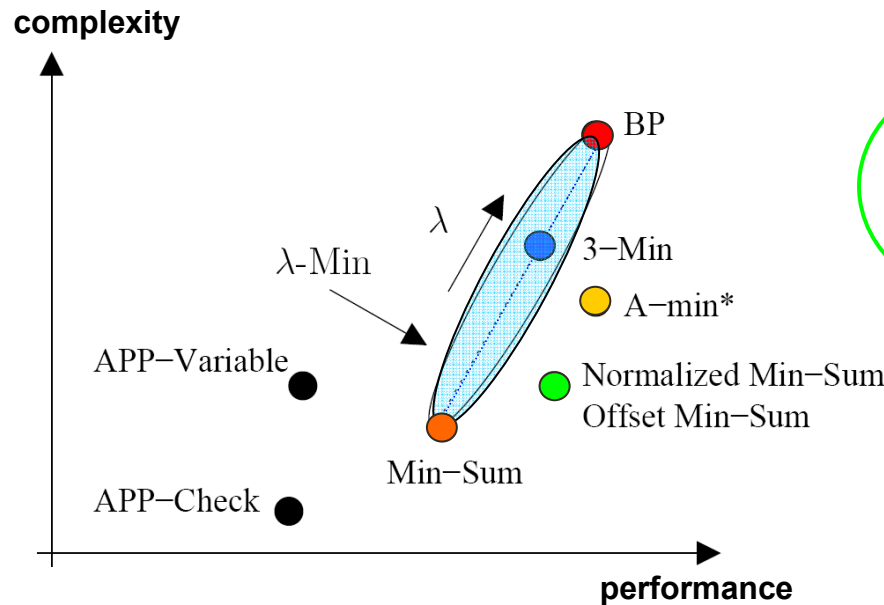


## Outline

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

◀ Simplification of BP algorithm

- Practical implementation of BP algorithm
  - Approximation of the function  $f(x)$
  - Limit the number of different edges messages



Decoding Algorithm	Variable node update rule	Check node update rule
● BP	$m_{vc}^i = y_v^0 + \sum_{c' \in C_v/c} m_{c'v}^{i-1}$	$ m_{cv}^i  = f \left( \sum_{v' \in V_c/v} f( m_{v'c}^i ) \right)$
BP-Based/Min-Sum	idem BP	$ m_{cv}^i  = \min_{v' \in V_c/v}  m_{v'c}^i $
Offset Min-Sum	idem BP	$ m_{cv}^i  = \min_{v' \in V_c/v}  m_{v'c}^i  + \beta$
Normalized Min-Sum	idem BP	$ m_{cv}^i  = \alpha \min_{v' \in V_c/v}  m_{v'c}^i $
● λ-min	idem BP	$ m_{cv}^i  = f \left( \sum_{v' \in V_c/v} f( m_{v'c}^i ) \right)$
● A-min*	idem BP	if $v = \arg \min_{v' \in V_c}  m_{v'c}^i $ $ m_{cv}^i  = f \left( \sum_{v' \in V_c/v} f( m_{v'c}^i ) \right)$ else $ m_{cv}^i  = f \left( \sum_{v' \in V_c} f( m_{v'c}^i ) \right)$
APP - check	idem BP	$ m_{cv}^i  = f \left( \sum_{v' \in V_c} f( m_{v'c}^i ) \right)$
APP-variable	$m_{vc}^i = y_v^0 + \sum_{c' \in C_v} m_{c'v}^{i-1}$	$ m_{cv}^i  = f \left( \sum_{v' \in V_c/v} f( m_{v'c}^i ) \right)$



## ■ Description of the algorithm

- Target: > 6 length cycle
- $z = 8$
- Mask considered:

$$H = \begin{bmatrix} \mathbf{I}_{\delta_0} & \mathbf{I}_{\delta_3} & \mathbf{I}_{\delta_6} & \mathbf{I}_0 & & \mathbf{I}'_1 \\ \mathbf{I}_{\delta_1} & \mathbf{I}_{\delta_4} & \mathbf{I}_{\delta_7} & \mathbf{I}_0 & \mathbf{I}_0 & \\ \mathbf{I}_{\delta_2} & \mathbf{I}_{\delta_5} & \mathbf{I}_{\delta_8} & & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$$

## ■ Incremental construction

Step	Matrix to analyse	Forbidden co-efficient	Choice
1	$\begin{bmatrix} \mathbf{I}_{\delta_0} & \bullet & \bullet & \mathbf{I}_0 & & \mathbf{I}'_1 \\ \bullet & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 & \\ \bullet & \bullet & \bullet & & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$\emptyset$	$\delta_0 = 0$

## Algorithm for codes design

Step	Matrix to analyse	Forbidden coefficient	Choice
1	$\begin{bmatrix} \mathbf{I}_{\delta_0} & \bullet & \bullet & \mathbf{I}_0 & & \mathbf{I}'_1 \\ \bullet & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 & \\ \bullet & \bullet & \bullet & & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$\emptyset$	$\delta_0 = 0$
2	$\begin{bmatrix} \mathbf{I}_0 & \bullet & \bullet & \mathbf{I}_0 & & \mathbf{I}'_1 \\ \mathbf{I}_{\delta_1} & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 & \\ \bullet & \bullet & \bullet & & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$[0]$	$\delta_1 = 6$

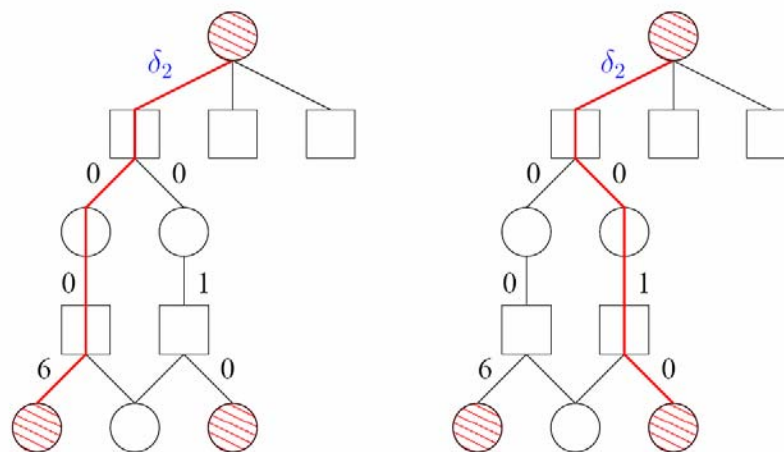
$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{\delta_0} & \mathbf{I}_{\delta_3} & \mathbf{I}_{\delta_6} & \mathbf{I}_0 & & \mathbf{I}'_1 \\ \mathbf{I}_{\delta_1} & \mathbf{I}_{\delta_4} & \mathbf{I}_{\delta_7} & \mathbf{I}_0 & \mathbf{I}_0 & \\ \mathbf{I}_{\delta_2} & \mathbf{I}_{\delta_5} & \mathbf{I}_{\delta_8} & & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$$

Algorithm for codes design

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

Step	Matrix to analyse	Forbidden coefficient	Choice
1	$\begin{bmatrix} I_{\delta_0} & \bullet & \bullet & I_0 & I'_1 \\ \bullet & \bullet & \bullet & I_0 & I_0 \\ \bullet & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$\emptyset$	$\delta_0 = 0$
2	$\begin{bmatrix} I_0 & \bullet & \bullet & I_0 & I'_1 \\ I_{\delta_1} & \bullet & \bullet & I_0 & I_0 \\ \bullet & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$[0]$	$\delta_1 = 6$
3	$\begin{bmatrix} I_0 & \bullet & \bullet & I_0 & I'_1 \\ I_6 & \bullet & \bullet & I_0 & I_0 \\ I_{\delta_2} & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$[7, 6]$	$\delta_2 = 3$

Expansion of the protograph to detect the configurations





# Algorithm for codes design

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

Step	Matrix to analyse	Forbidden coefficient	Choice
1	$\begin{bmatrix} I_{\delta_0} & \bullet & \bullet & I_0 & I'_1 \\ \bullet & \bullet & \bullet & I_0 & I_0 \\ \bullet & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$\emptyset$	$\delta_0 = 0$
2	$\begin{bmatrix} I_0 & \bullet & \bullet & I_0 & I'_1 \\ I_{\delta_1} & \bullet & \bullet & I_0 & I_0 \\ \bullet & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$[0]$	$\delta_1 = 6$
3	$\begin{bmatrix} I_0 & \bullet & \bullet & I_0 & I'_1 \\ I_6 & \bullet & \bullet & I_0 & I_0 \\ I_{\delta_2} & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$[7, 6]$	$\delta_2 = 3$
4	$\begin{bmatrix} I_0 & I_{\delta_3} & \bullet & I_0 & I'_1 \\ I_6 & \bullet & \bullet & I_0 & I_0 \\ I_3 & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$\emptyset$	$\delta_3 = 0$



# Algorithm for codes design

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

Step	Matrix to analyse	Forbidden coefficient	Choice
1	$\begin{bmatrix} I_{\delta_0} & \bullet & \bullet & I_0 & I'_1 \\ \bullet & \bullet & \bullet & I_0 & I_0 \\ \bullet & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$\emptyset$	$\delta_0 = 0$
2	$\begin{bmatrix} I_0 & \bullet & \bullet & I_0 & I'_1 \\ I_{\delta_1} & \bullet & \bullet & I_0 & I_0 \\ \bullet & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$[0]$	$\delta_1 = 6$
3	$\begin{bmatrix} I_0 & \bullet & \bullet & I_0 & I'_1 \\ I_6 & \bullet & \bullet & I_0 & I_0 \\ I_{\delta_2} & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$[7, 6]$	$\delta_2 = 3$
4	$\begin{bmatrix} I_0 & I_{\delta_3} & \bullet & I_0 & I'_1 \\ I_6 & \bullet & \bullet & I_0 & I_0 \\ I_3 & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$\emptyset$	$\delta_3 = 0$
5	$\begin{bmatrix} I_0 & I_0 & \bullet & I_0 & I'_1 \\ I_6 & I_{\delta_4} & \bullet & I_0 & I_0 \\ I_3 & \bullet & \bullet & I_0 & I_0 \end{bmatrix}$	$[0, 6]$	$\delta_4 = 7$



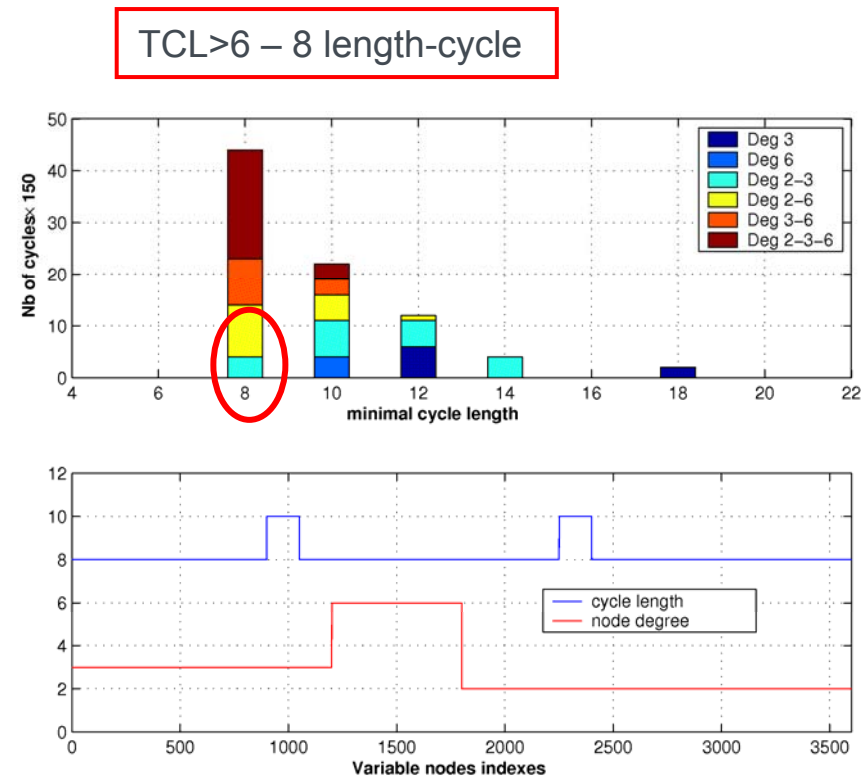
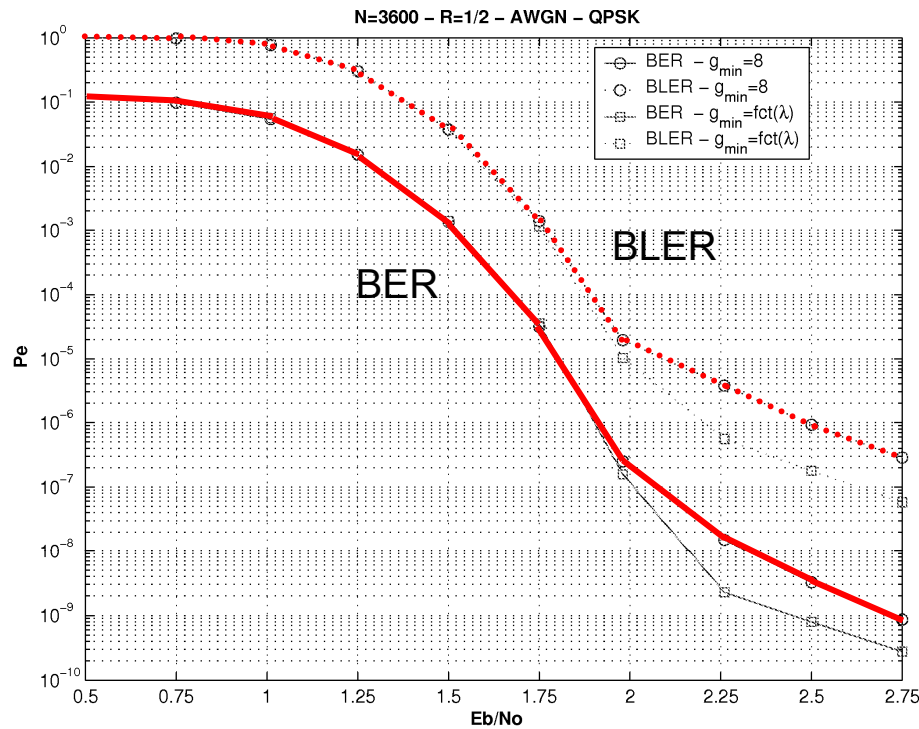
## Algorithm for codes design

$$H = \begin{bmatrix} \mathbf{I}_{\delta_0} & \mathbf{I}_{\delta_3} & \mathbf{I}_{\delta_6} & \mathbf{I}_0 & \mathbf{I}'_1 \\ \mathbf{I}_{\delta_1} & \mathbf{I}_{\delta_4} & \mathbf{I}_{\delta_7} & \mathbf{I}_0 & \mathbf{I}_0 \\ \mathbf{I}_{\delta_2} & \mathbf{I}_{\delta_5} & \mathbf{I}_{\delta_8} & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$$

Step	Matrix to analyse	Forbidden coefficient	Choice
1	$\begin{bmatrix} \mathbf{I}_{\delta_0} & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}'_1 \\ \bullet & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \\ \bullet & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$\emptyset$	$\delta_0 = 0$
2	$\begin{bmatrix} \mathbf{I}_0 & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}'_1 \\ \mathbf{I}_{\delta_1} & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \\ \bullet & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$[0]$	$\delta_1 = 6$
3	$\begin{bmatrix} \mathbf{I}_0 & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}'_1 \\ \mathbf{I}_6 & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \\ \mathbf{I}_{\delta_2} & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$[7, 6]$	$\delta_2 = 3$
4	$\begin{bmatrix} \mathbf{I}_0 & \mathbf{I}_{\delta_3} & \bullet & \mathbf{I}_0 & \mathbf{I}'_1 \\ \mathbf{I}_6 & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \\ \mathbf{I}_3 & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$\emptyset$	$\delta_3 = 0$
5	$\begin{bmatrix} \mathbf{I}_0 & \mathbf{I}_0 & \bullet & \mathbf{I}_0 & \mathbf{I}'_1 \\ \mathbf{I}_6 & \mathbf{I}_{\delta_4} & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \\ \mathbf{I}_3 & \bullet & \bullet & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$[0, 6]$	$\delta_4 = 7$
$\vdots$			
9	$\begin{bmatrix} \mathbf{I}_0 & \mathbf{I}_0 & \mathbf{I}_0 & \mathbf{I}_0 & \mathbf{I}'_1 \\ \mathbf{I}_6 & \mathbf{I}_7 & \mathbf{I}_3 & \mathbf{I}_0 & \mathbf{I}_0 \\ \mathbf{I}_3 & \mathbf{I}_1 & \mathbf{I}_{\delta_8} & \mathbf{I}_0 & \mathbf{I}_0 \end{bmatrix}$	$[3, 0, 1, 5, 7]$	$\delta_8 = 6$

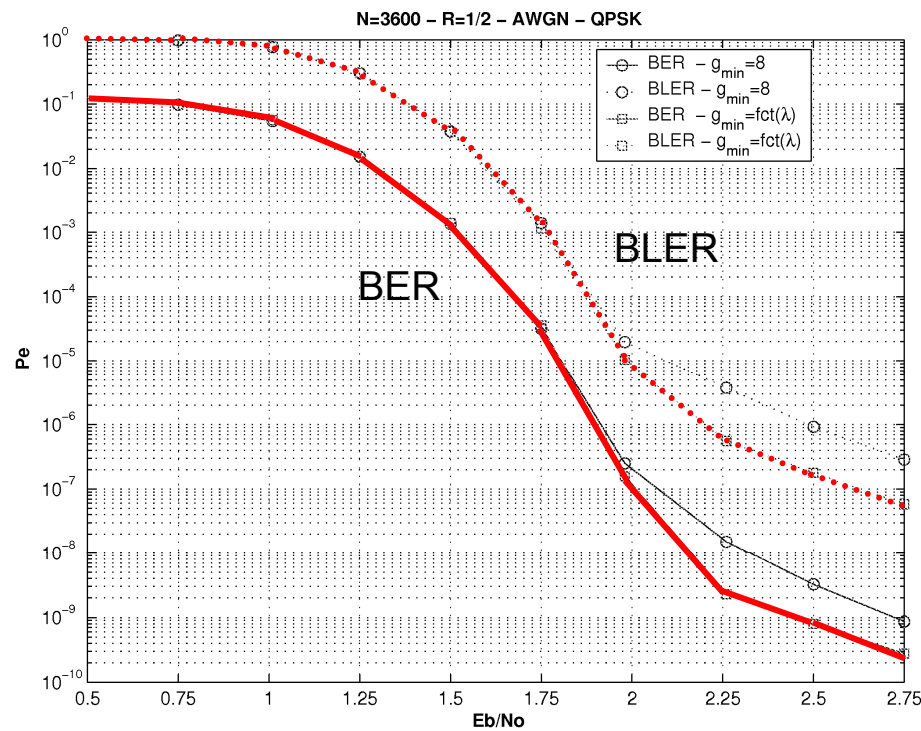


- Algorithm parameterization: some results
  - Avoid low length cycle involving low degree variable nodes

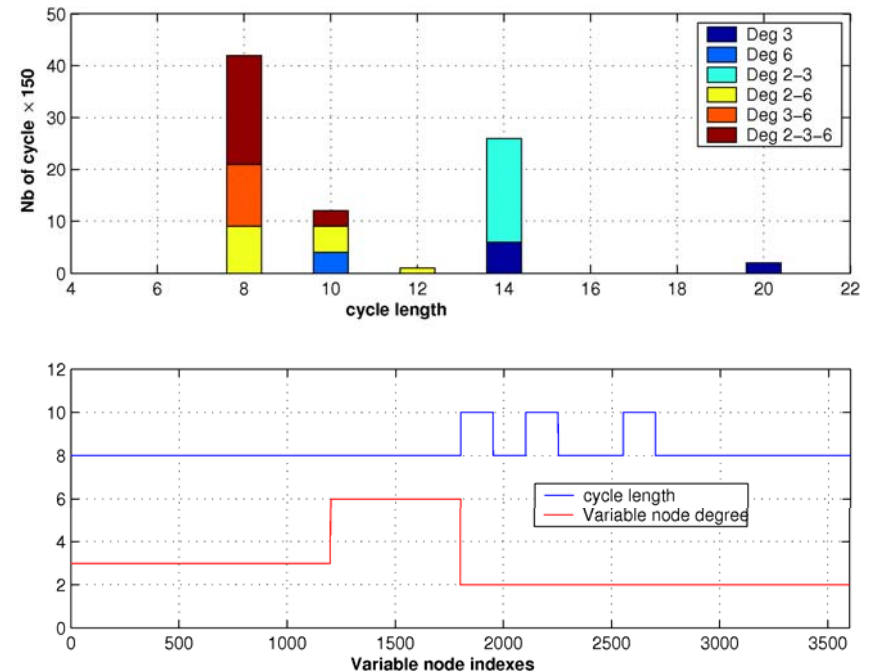


## Algorithm parameterization: some results

- Avoid low length cycle involving low degree variable nodes

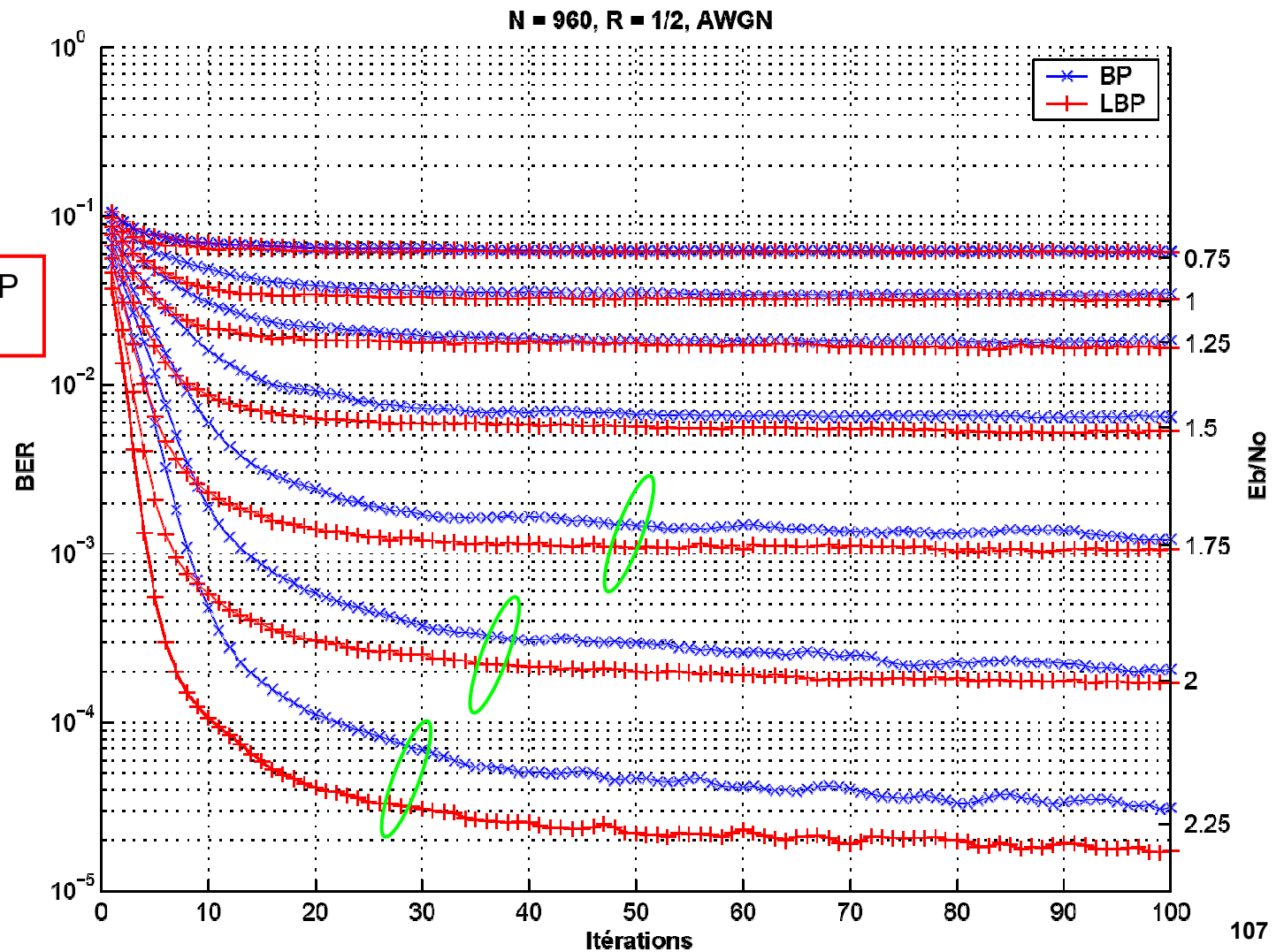


TCL depends on variable node degree



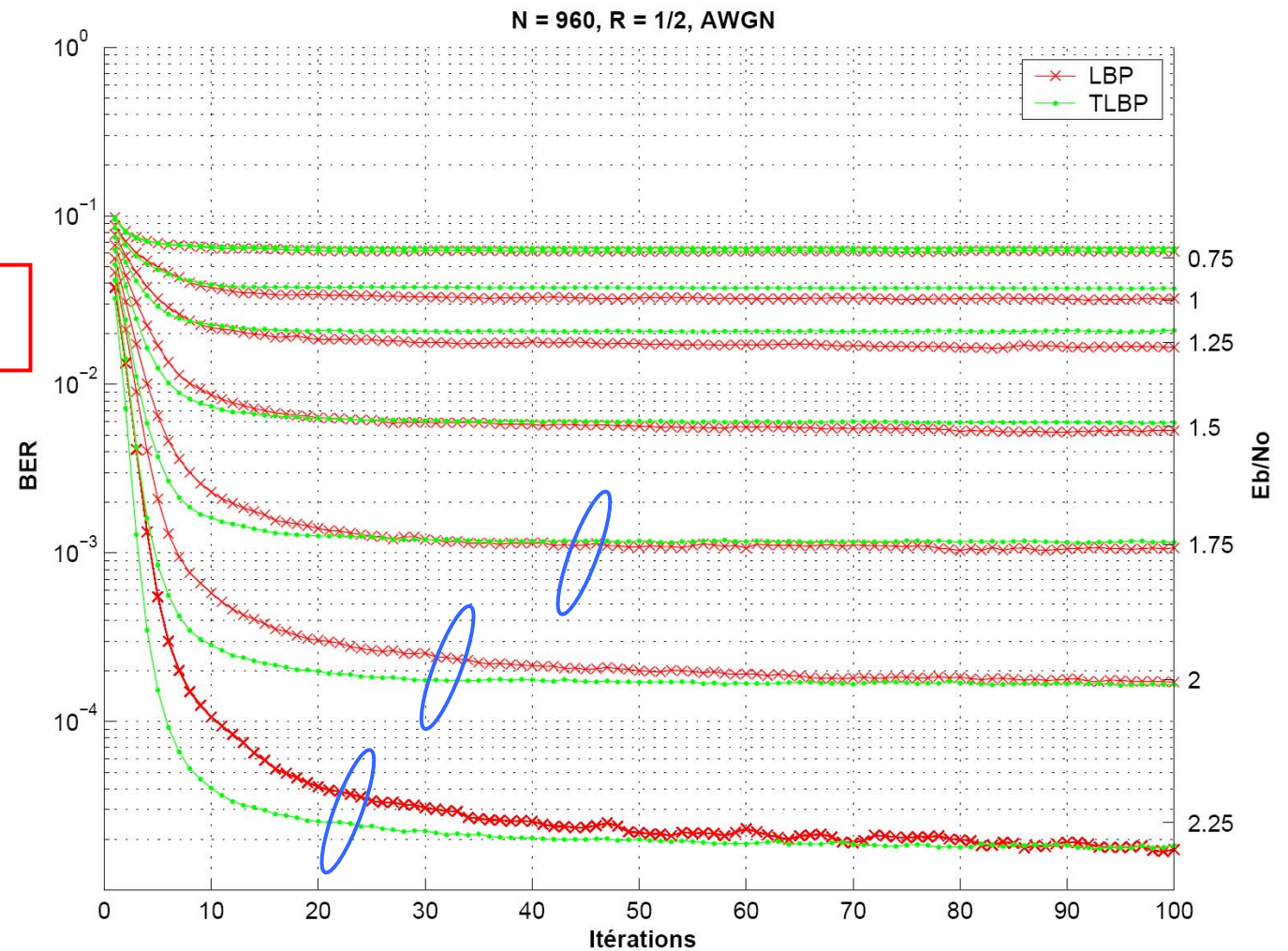
■ Comparison BP/LBP

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  $m_{vc}$

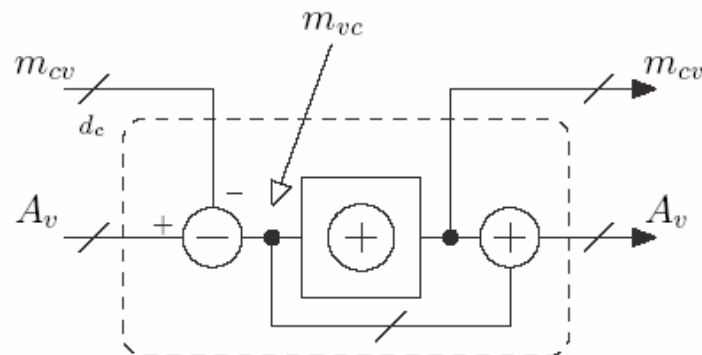
$$m_{vc} = y_v + \sum_{c' \in C_v} m_{c'v} - m_{cv}$$

$$m_{vc} = \underbrace{\hspace{10em}}_{A_v} - m_{cv}$$

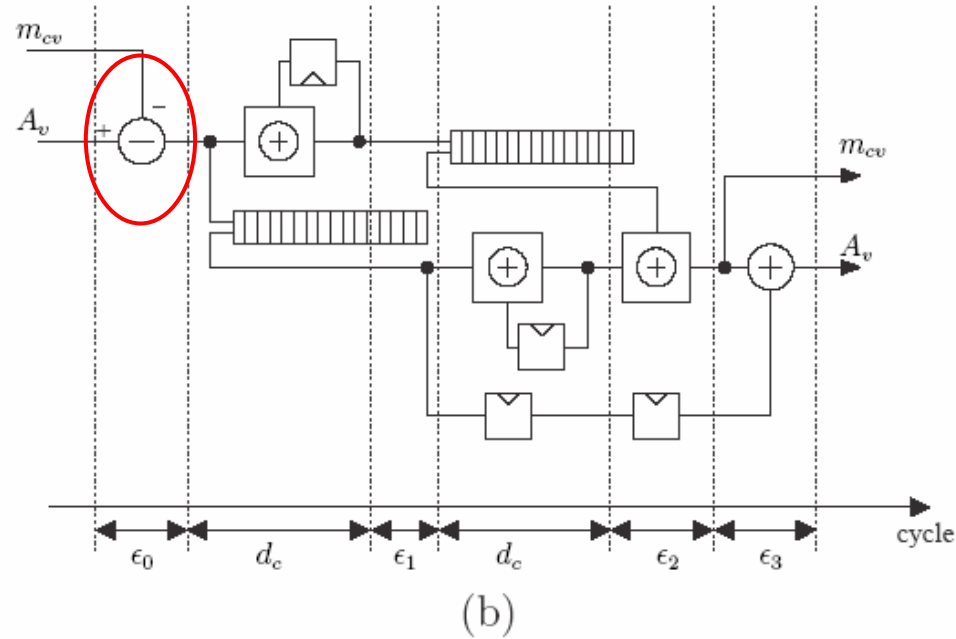
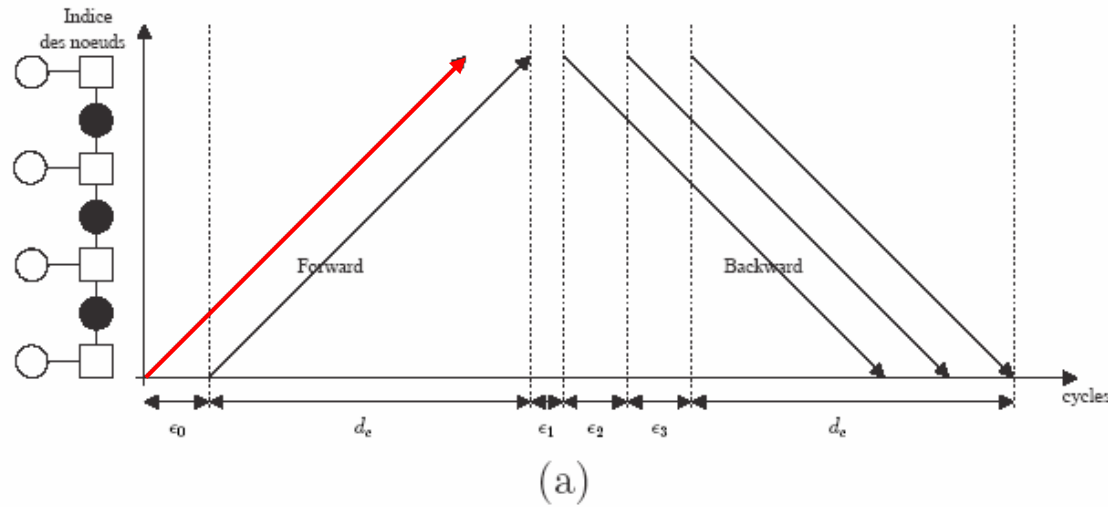
- $d_c$  cycles to compute  $A_v$

$$A_v = y_v + \sum_{c' \in C_v/c} m_{c'v} + m_{cv}$$

$$A_v = \underbrace{\hspace{10em}}_{m_{vc}} + m_{cv}$$

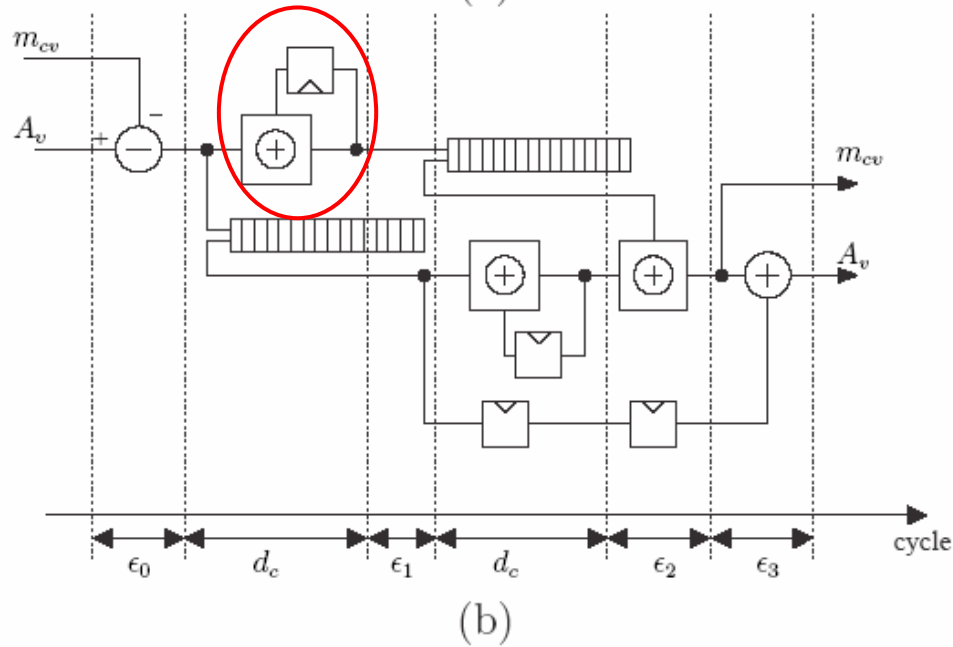
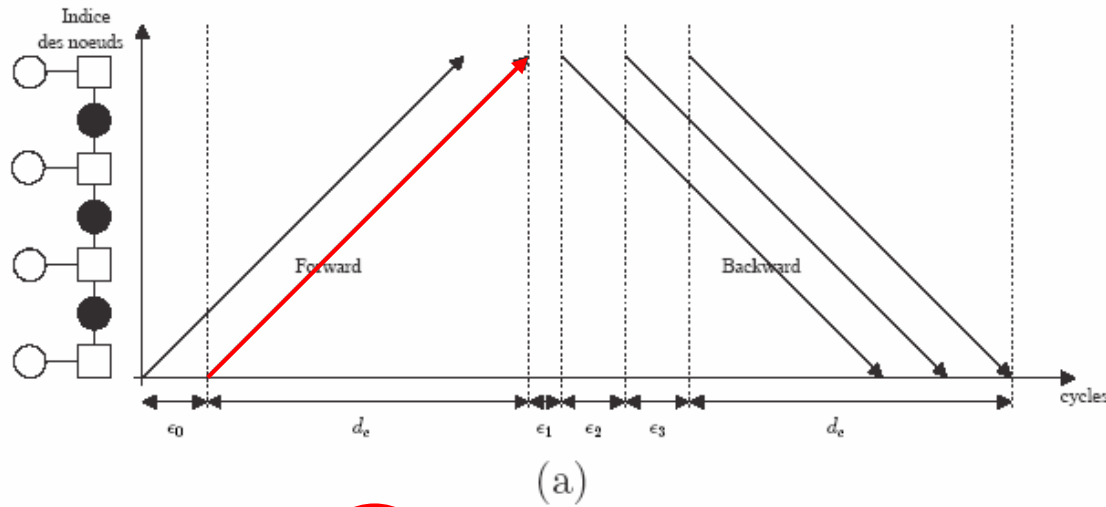


◀ About CNP processors: Modeling for Min-Sum algorithm (II)



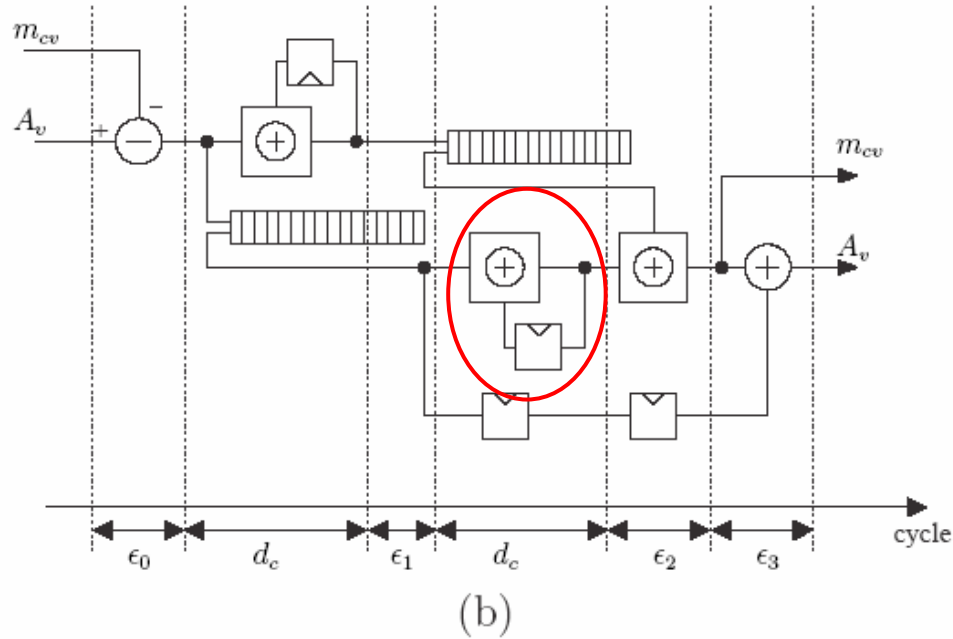
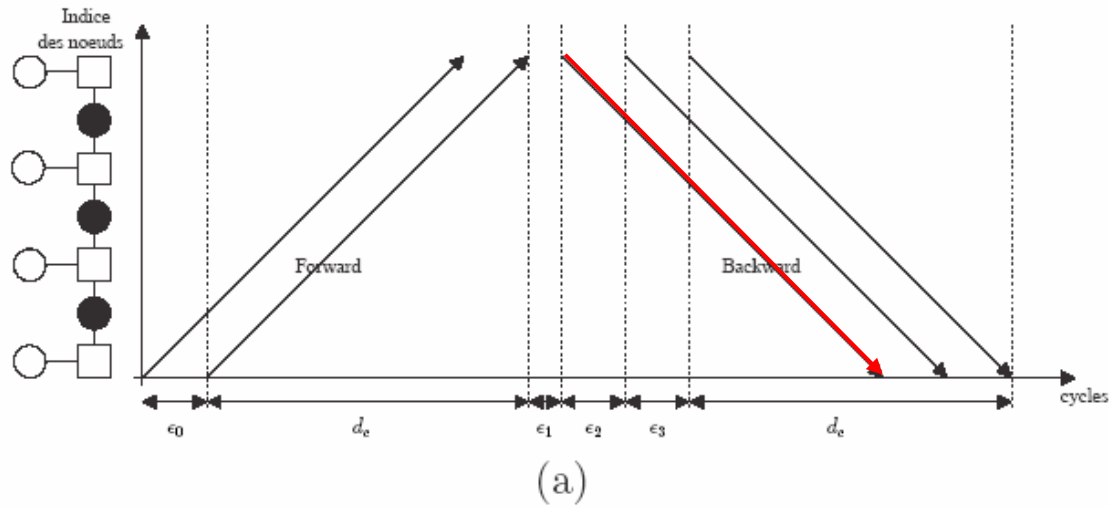
$$D = p \frac{RN}{M(2d_c + \epsilon)it} f_{clk}$$

# About CNP processors: Modeling for Min-Sum algorithm (II)



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

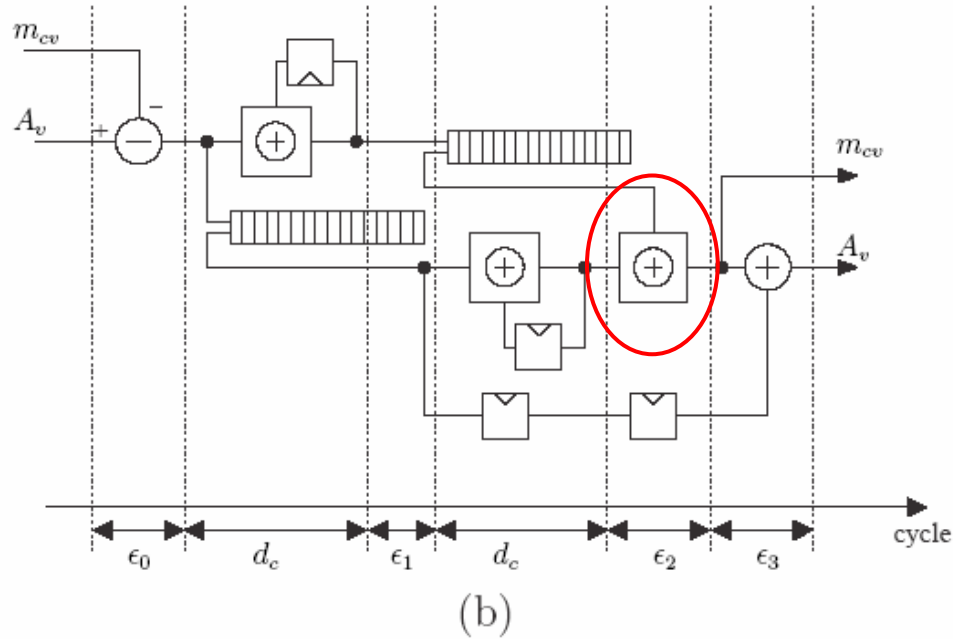
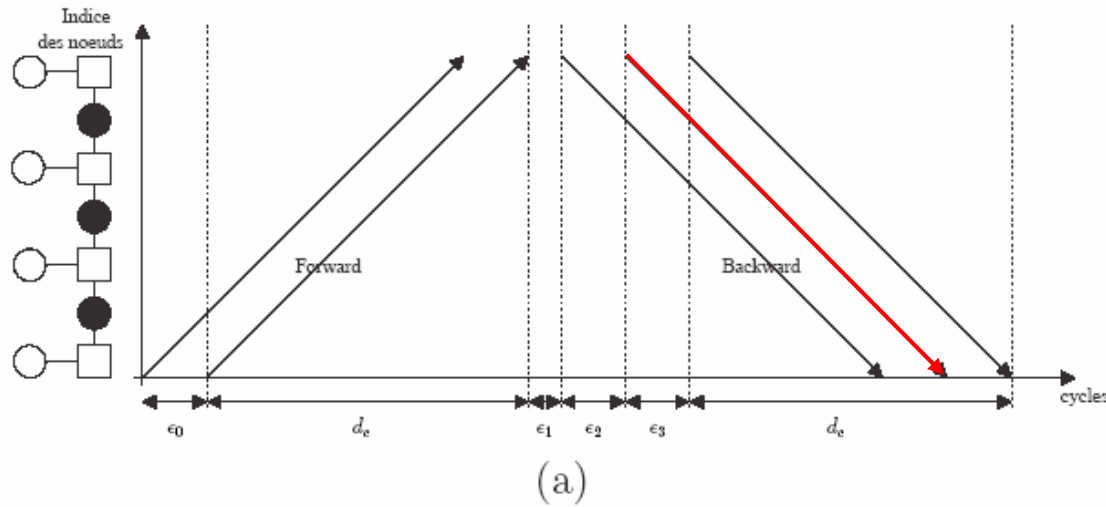
# About CNP processors: Modeling for Min-Sum algorithm (II)



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

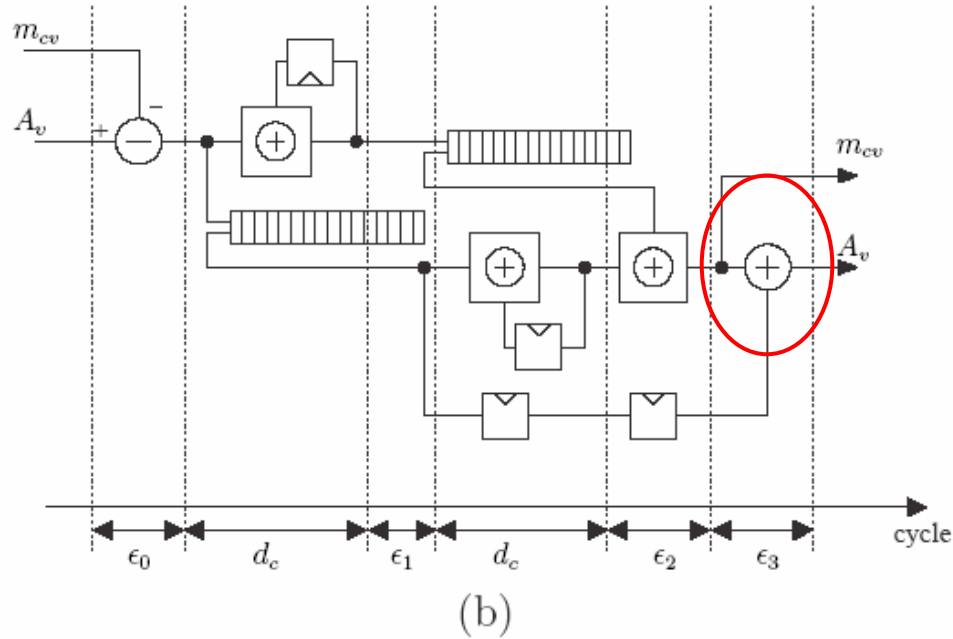
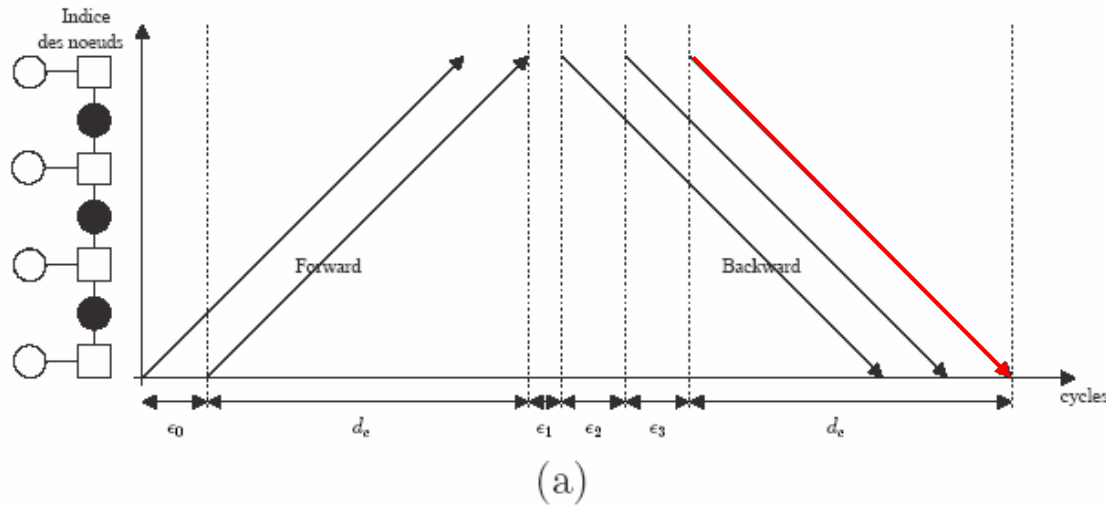


◀ About CNP processors: Modeling for Min-Sum algorithm (II)



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

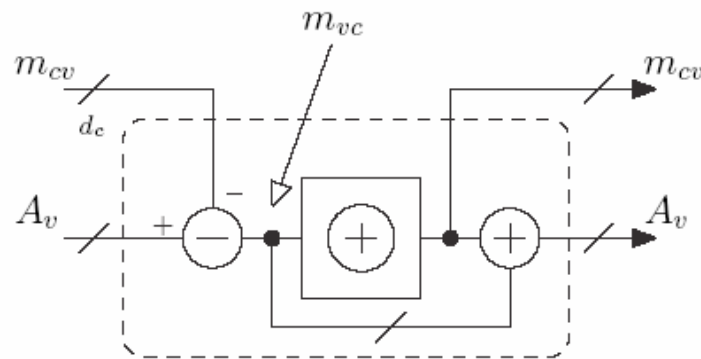
# About CNP processors: Modeling for Min-Sum algorithm (II)



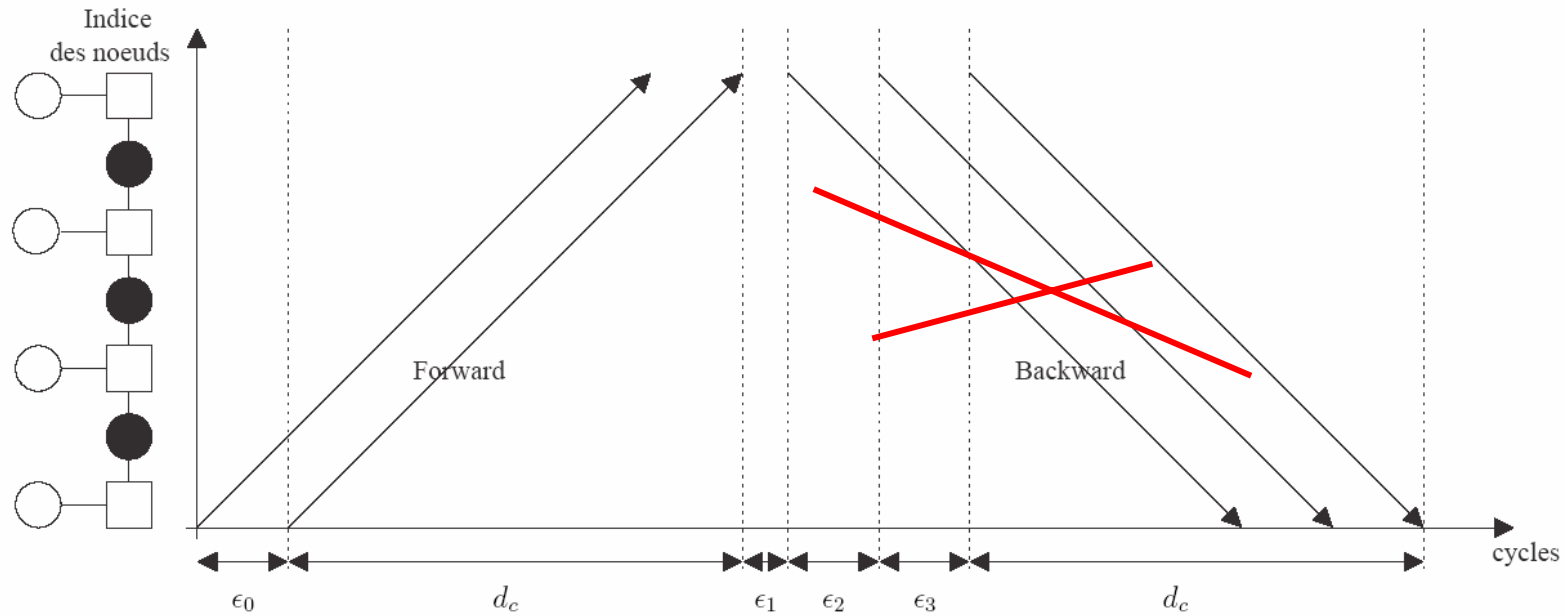
$$D = p \frac{RN}{M(2d_c + \epsilon)it} f_{clk}$$

## ■ 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

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

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

