


Forschungszentrum Telekommunikation Wien


Theory and Design of  
Turbo and Related Codes  
*Lecture 9*

*Jossy Sayir & Gottfried Lechner*

<http://userver.ftw.at/~jossy/turbo/index.html>



Kplus  
Kompetenzzentren-Programm



Contents

- Comparison LDPC / Turbo Codes
- Efficient LDPC Encoding
- Repeat-Accumulate Codes

© ftw. 2005

## Encoder/Decoder Comparison



- We have learned about:
  - LDPC codes
  - Turbo Codes
- How do the 2 methods **compare** to each other?
- What are the advantages / disadvantages of each method?
- Can we devise a coding method that combines the advantages of LDPC and Turbo codes?

© ftw. 2005

## Common Advantage



What is the fundamental difference between the **new** (post-1993) and the **old** (pre-1993) coding techniques?

Decoding complexity is now **linear** in the code block length

This is true of Turbo & LDPC codes...

© ftw. 2005

## Linear Decoding Complexity



What does this mean?

For regular LDPC codes:

$$\begin{aligned} \text{Complexity per bit} &= \text{No of iterations} \times \text{Operations per bit per iteration} \\ &= N_{it} \times d_v \times \left( \text{Variable node decoder} + d_c \times \text{Check node decoder} \right) \end{aligned}$$

$$\text{Total Complexity} = \text{Block Length } N \times \text{Complexity per bit}$$

© ftw. 2005

## Linear Decoding Complexity



- The complexity per bit is **independent** of the block length.
- Therefore, the total complexity is **linear** in the block length.
- In general, the **complexity per bit** will be a **function** of the number of iterations  $N_{it}$  and of the degree polynomials  $\lambda(x)$  and  $\rho(x)$
- For **Turbo codes**, the total complexity is  
Total Complexity =  $N_{it} \times 2 \times \text{BCJR Complexity}$   
BCJR Complexity =  $N \times N_{States} \times \text{Ops per state}$
- The result is again **linear** in the block length  $N$

© ftw. 2005

## Other Criteria

Decoding  
Complexity

Encoding  
Complexity

Can the code  
be optimized?

Is the code  
easily described?

## Other Criteria



- Encoding complexity?
  - linear?
  - quadratic?
- Can the code be optimized?
  - determine rate threshold?
  - maximize threshold?
- Is the code easily described?
  - how can the code be specified in a standard?
  - how much storage will the code description require?

## Comparison



	Turbo codes	LDPC code	
Decoder complexity	linear	linear	
Encoder complexity	linear	quadratic	
Threshold?	determined by simulation, only trial and error optimization	determined analytically, optimizable	
Code description	conv. encoders and interleaver	sparse parity-check matrix, encoding matrix	

© ftw. 2005

## How do we encode LDPC codes?



From Lecture 3:

Systematic Encoder:

$$(x_1 \dots x_{16}) = (u_1 \dots u_6) G = \underbrace{(u_1 \dots u_6)}_{\text{Systematic part}} \underbrace{x_7 \dots x_{16}}_{\text{Parity-check part}}$$

The encoder matrix must be of the form:  $G = [I \ P]$

Systematic Parity-Check Matrix:  $H = [P^T_{M \times K} \ I_{M \times M}]$

$$\begin{aligned} G H^T &= [I_{K \times K} \ P_{K \times M}] [P^T_{M \times K} \ I_{M \times M}]^T \\ &= I_{K \times K} P_{K \times M} + P_{K \times M} I_{M \times M} \\ &= P_{K \times M} + P_{K \times M} \\ &= 0 \end{aligned}$$

© ftw. 2005

## How do we encode LDPC codes?



- Starting from the low-density parity-check matrix  $H$ , use row-column operations to obtain a new parity-check matrix of the form  $H' = [P_{M \times K}^T \ I_{M \times M}]$ ,

where  $M = N - K$

- The systematic generator matrix is  $G = [I_{K \times K} \ P_{K \times M}]$ , i.e., we can compute the parity bits as

$$x_p = u P_{K \times M}$$

- The **pre-processing** (determining  $P$ ) is done **offline** and doesn't count towards encoder complexity
- The resulting encoder complexity (since  $P$  isn't sparse) is  $K \times M = R N^2$  for  $R = K/N$ , i.e., **quadratic** in the block length

Can we do any better?

© ftw. 2005

## Efficient encoding of LDPC codes



Idea: use **erasure decoding** to encode LDPC codes!

For systematic encoders,  $(x_1 \dots x_N) = (u_1 \dots u_k \ x_{k+1} \dots x_N)$

We set all **parity-bits** to **erasures** and use the sum-product algorithm to determine their values...

Will this work?

© ftw. 2005

## Efficient encoding of LDPC codes



- The probability of success with the sum-product algorithm can only approach 1 for rates below the threshold  $\theta$  (and approaches 0 above this rate)
- For the Binary Erasure Channel (BEC) with erasure probability  $\delta$ ,  $\theta < 1-\delta$
- We are setting  $N-K = (1-R)N$  positions in our codeword to erasures, i.e., we are operating our encoder with the same number of erasures as the average for a BEC with  $\delta = 1-R$
- $R = 1-\delta > \theta$ . We are operating above the threshold!
- Our probability of success approaches 0 for  $N \rightarrow \infty$

© ftw. 2005

## Efficient encoding of LDPC codes



(Richardson & Urbanke, 2001)

- Run sum-product decoder until no more erasures can be resolved, i.e., no more rows with only 1 erasure are available in the matrix (this is equivalent to a partial triangulation of the matrix by column permutations)
- Use Gauss elimination to "invert" the remaining  $g$  rows
- This pre-processing can be done offline
- The resulting encoding operation consists of
  - sum-product algorithm, at most  $N-K-g$  iterations
  - matrix multiplication with a  $g \times g$  matrix
- The resulting complexity is of the order  $N+g^2$

© ftw. 2005

## Comparison



	Turbo codes	LDPC code	
Decoder complexity	linear	linear	
Encoder complexity	linear (2 conv. enc)	quadratic ("almost" linear after pre-proc.)	
Threshold?	determined by simulation, only trial and error optimization	determined analytically, optimizable	
Code description	conv. encoders and interleaver	sparse parity- check matrix, encoding matrix	

© ftw. 2005

## Low-Density Encoder Matrix



Idea: why don't we use a low-density matrix as an encoding matrix instead of as a parity-check matrix?

- Multiplication with a sparse matrix has order  $N$  ( $\rightarrow$  low-complexity encoder!), e.g.,  $d_v \times N$  for a regular matrix
- We learned in Lecture 1 how to decode based on the encoder matrix

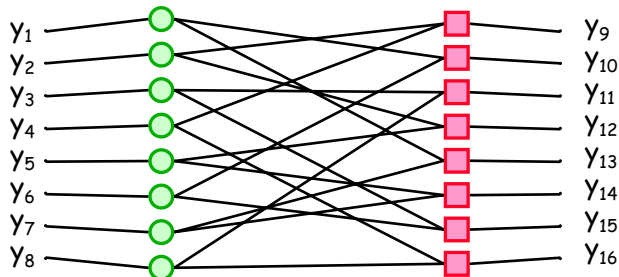
Can we decode using the sum-product algorithm?  
( $\rightarrow$  low-complexity decoding)

Why is this still a bad idea?

© ftw. 2005

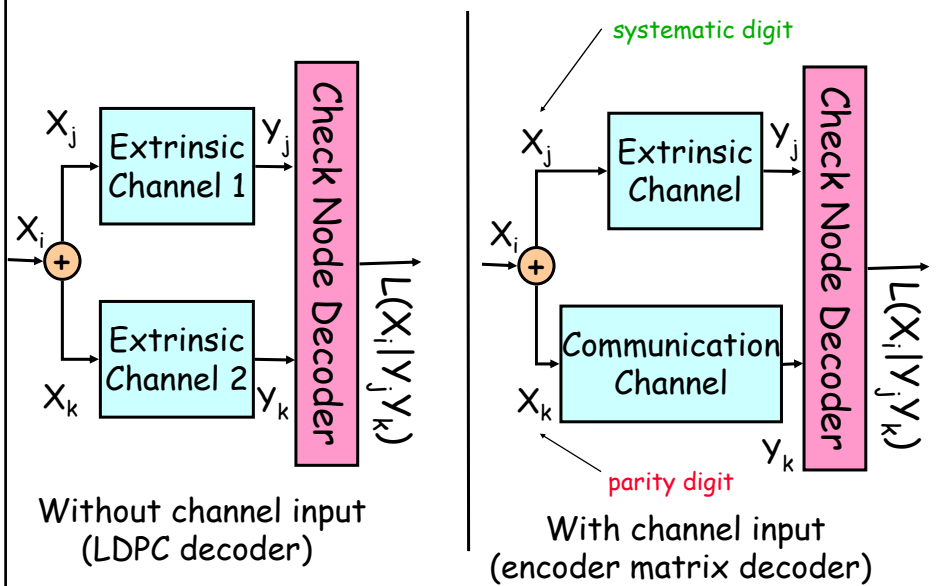
## Sum-Product Algorithm for the Encoder Matrix

$$\begin{array}{l}
 \text{systematic part} \\
 (u_1 \dots u_k)
 \end{array}
 \begin{pmatrix}
 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
 \end{pmatrix}
 \begin{array}{l}
 \text{sparse} \\
 \text{parity-checks}
 \end{array}
 = (x_1 \dots x_N)$$



**New:** parity-check nodes receive messages from the communication channel!

## Check node decoder

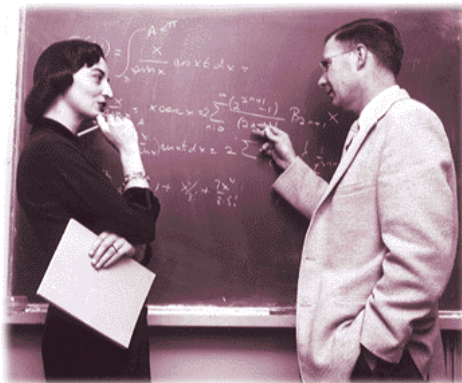


Why is this **still** a **bad** idea?

Distance and Weight  
(from Lecture 1)

The Hamming Distance between two N-tuples is the number of positions in which they differ

The Hamming Weight of an N-tuple is the number of positions in which it is non-zero



From the Homepage <http://www.cs.mcgill.ca/~sarnovs/hamming.html>

R. Hamming explaining distance? Around 1948...

## Examples



$$d(01001011, 11000011) = 2$$

$$w(00101001) = 3$$

$$d(x,y) = w(x-y) = w(x+y)$$

For a linear code, the number of codewords at Hamming distance  $i$  from any codeword  $v$  is equal to the number  $A_i$  of codewords of Hamming weight  $i$ .

Proof: for every codeword  $v'$  at distance  $i$  from  $v$ ,  $v+v'$  has weight  $i$ . For every codeword  $v''$  of weight  $i$ ,  $d(v,v+v'') = i$ .

© ftw. 2005

## Minimum Distance



The minimum distance  $d_{\min}$  of a code is the smallest distance between any two codewords

For a linear code:

$$\text{minimum distance } d_{\min} = \text{minimum weight } w_{\min}$$

Minimum weight and weight profile are important characteristics to determine the performance of a linear code

© ftw. 2005

## Low-Density Encoder Matrix



A Low-Density Encoder Matrix will produce  
a **bad code** with **bad error performance**  
(even for optimal sequence or symbol decoders)

But all is not lost! There is a simple  
fix for this...

© ftw. 2005

## Repeat Accumulate Codes



Invented by Hui Jin and Robert J. McEliece  
from the California Institute of Technology  
(CalTech) in 1998 / 2000

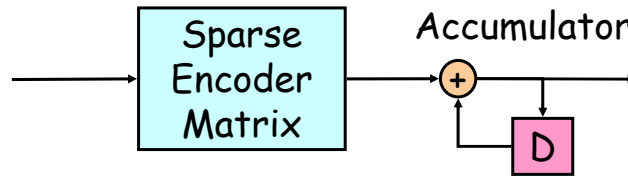


From the Website of CalTech

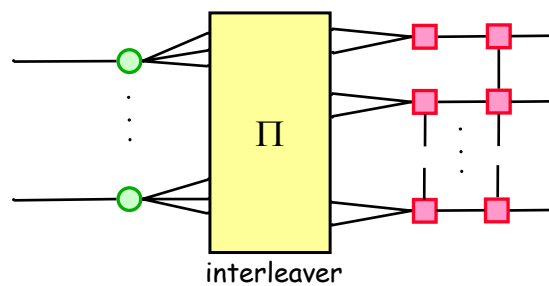
Bob McEliece

© ftw. 2005

## Repeat Accumulate Codes



Decoder:



© ftw. 2005

## Repeat Accumulate Codes



- Encoding complexity: multiplication with a sparse matrix,  $d_v \times N \rightarrow \text{linear!}$
- Accumulator complexity negligible
- Good code, good error performance
- Decoding with sum-product algorithm  $\rightarrow$  linear complexity
- Description: degree polynomials + interleaver

© ftw. 2005

## Comparison



	Turbo codes	LDPC code	RA Codes
Decoder complexity	linear	linear	linear
Encoder complexity	linear (2 conv. enc)	quadratic ("almost" linear after pre-proc.)	linear (sparse matrix mult. & accumulate)
Threshold?	determined by simulation, only trial and error optimization	determined analytically, optimizable	determined analytically, optimizable
Code description	conv. encoders and interleaver	sparse parity-check matrix, encoding matrix	sparse encoding matrix

© ftw. 2005

## Asymptotic Decoder Complexity for Optimized Codes



- The complexity depends on  $N_{it}$ ,  $\lambda(x)$  and  $\rho(x)$
- When we optimize code, (maximize threshold) how does the number of operations per variable behave?



Igal Sasson & Rüdiger Urbanke

- For LDPC, density per variable and number of iterations grow, i.e. complexity per variable goes to infinity!
- For RA codes, complexity remains constant!!!

(Result yet unpublished, paper currently being written)

© ftw. 2005

## Comparison



	Turbo codes	LDPC code	RA Codes
Decoder complexity	linear <b>(non-optim.)</b>	linear <b>(<math>\rightarrow \infty</math>)</b>	linear <b>(<math>&lt; \infty</math> ?)</b>
Encoder complexity	linear (2 conv. enc)	<b>quadratic</b> ("almost" linear after pre-proc.)	linear (sparse matrix mult. & accumulate)
Threshold?	determined by <b>simulation</b> , only <b>trial and error</b> optimization	determined analytically, <b>optimizable</b>	determined analytically, <b>optimizable</b>
Code description	<b>conv. encoders</b> and <b>interleaver</b>	<b>sparse parity-</b> <b>check matrix,</b> <b>encoding matrix</b>	<b>sparse</b> <b>encoding</b> <b>matrix</b>

© ftw. 2005