Multiple Slice Turbo Codes

David Gnaedig

Supervisors:

Emmanuel Boutillon

Michel Jezequel

Outline

- General description
- Tail-biting code
- Construction of a Slice Turbo Code
- Hardware implementation
- Performance results
- Conclusion

Introduction

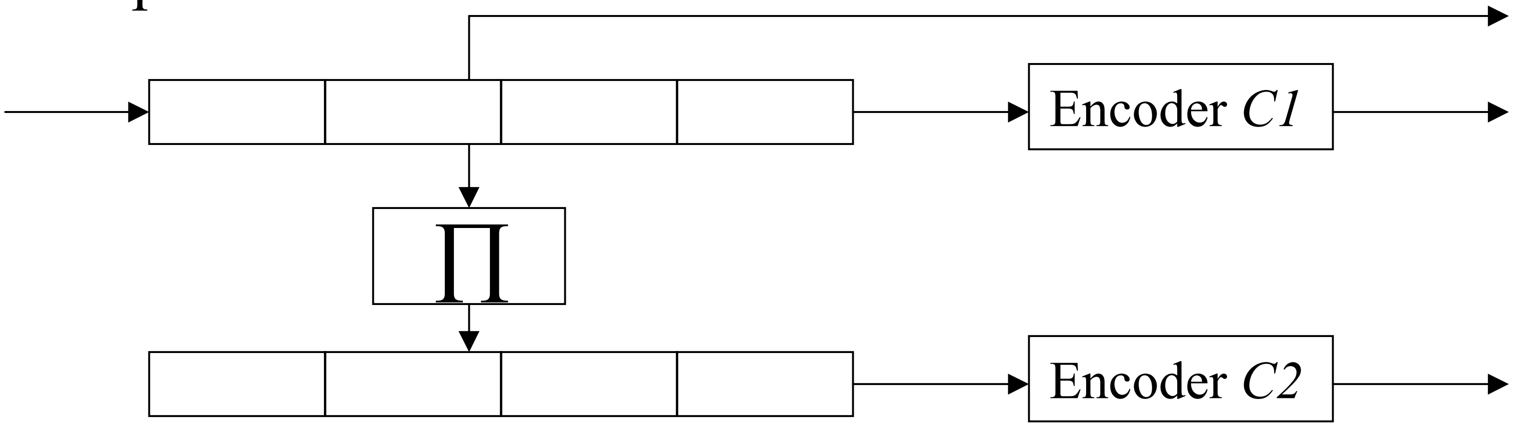
- Convolutional turbo-codes advantages:
  - simple encoding
  - scalability : rate, frame size
  - very good performance close to Shannon limit
- Drawbacks:
  - Lack of parallelism in the decoding algorithm: MAP-based algorithms are sequential
  - high latency
  - low throughput

Introduction

- Original idea from V. Gaudet and G. Gulak
  - Analog implementation requires trellis of small length
  - Frame is cut into sub-blocks: small length trellis
  - Several trellises decoded in parallel
- High parallelism but worse performance
- But applicable to digital decoder
- Improvements :
  - trellis termination with tail-biting codes
  - Interleaver structure to allow parallelism in both dimensions
  - Equations and parameters to have good performance
- Good performance, no degradation under the parallelism constraint

General description

- Parallel concatenation in 2 dimensions:
  - In each direction, the frame is split into $P \geq 2$ sub-blocs of M symboles ($N = M \,.\, P$)
  - Interleaver of size N
  - Constituent code: duo-binary convolutional systematic recursive code (8 state or 16 states)
  - Interleaver and memory organization allow decoding parallelism



Tail-biting code

C. Berrou, C. Douillard, M. Jézéquel, " *Multiple parallel concatenation of circular recursive systematic codes* ", Annales des Télécommunications, tome 54, n°3-4, pp 166-172, 1999.

- Circular Recursive Systematic Convolutional code (CRSC): resolve trellis termination
  - starting state = ending state : $S_0 = S_{N-1}$
  - no loss with trellis termination
- Encoding process :
  - from initial state S_i=0, first encoding of the whole frame
  - from the final state S_f, determine circular state S_c
  - from state S_c, second encoding of the whole frame
  - generate coded bits while second encoding process

Construction of a Slice Turbo Code

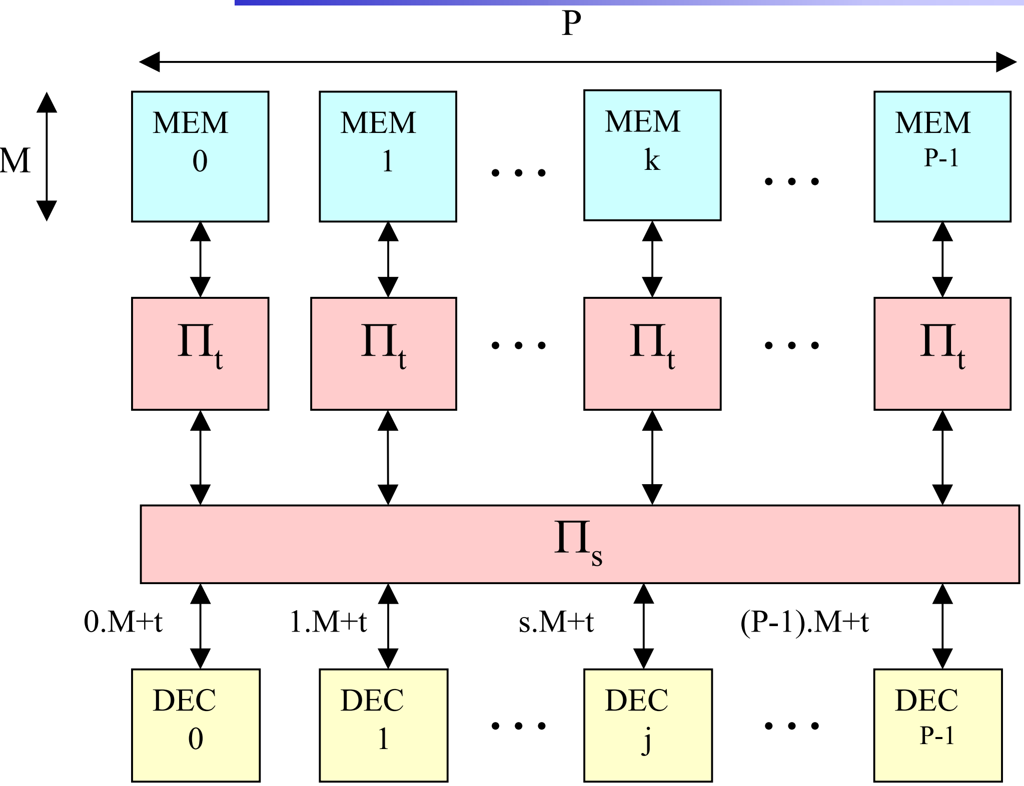
- A basic example
- Interleaver structure
- Evaluation of the interleaver properties
- Interleaver construction

A basic example

- 4 slices in each dimension in 4 memory banks: clubs, spades, hearts, diamonds
- First dimension: memory bank = slice
- Interleaver:
  - address 1 - rotation 1
  - address 3 - rotation 2
  - address 5 - rotation 3
  - address 2 - rotation 4
  - address 4 - rotation 5

  =>No memory conflict
- Second dimension: encode 4 sets separately

Interleaver structure



- Frame of size *N*
- *P* slices of size *M* stored in *P* memory banks
- *P* SISO decoders
- *t*: temporal index
- *s*: spatial index

❖ Natural order:

- temporal: identity
- spatial: identity

❖ Permuted order:

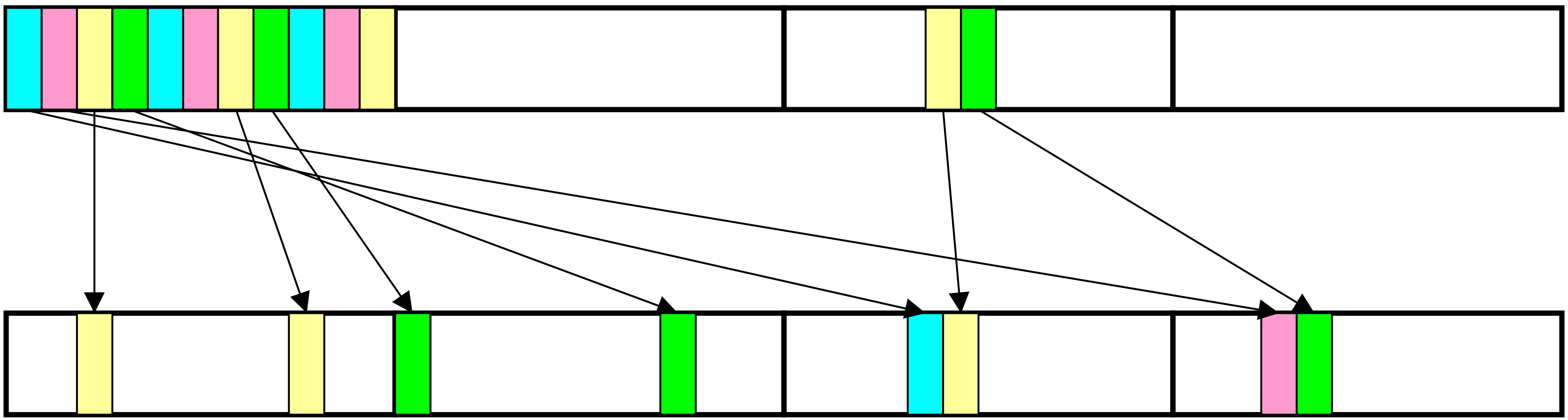
- temporal: function of *t*
- spatial: function of *t* and *s*

Evaluation of the interleaver properties

- Objective of the interleaver:
  - Irregular dispersion: randomness of the code
  - Low weight sequence / high weight sequence
- Construction criteria:
  - Correlation properties: influences the convergence of the iterative decoding
    - Avoid short cycles (primary and secondary cycles)
  - Minimum distance: asymptotic performance
    - Return To Zero sequences
    - (locked) error patterns
    - error impulse method
- Joint optimization of the temporal and spatial permutation

Correlation property

- Correlation of the extrinsic information influences performance of iterative decoding
- Objective : maximize cycle length
  - Primary cycle : one interleaving
  - Secondary cycle : one interleaving / deinterleaving

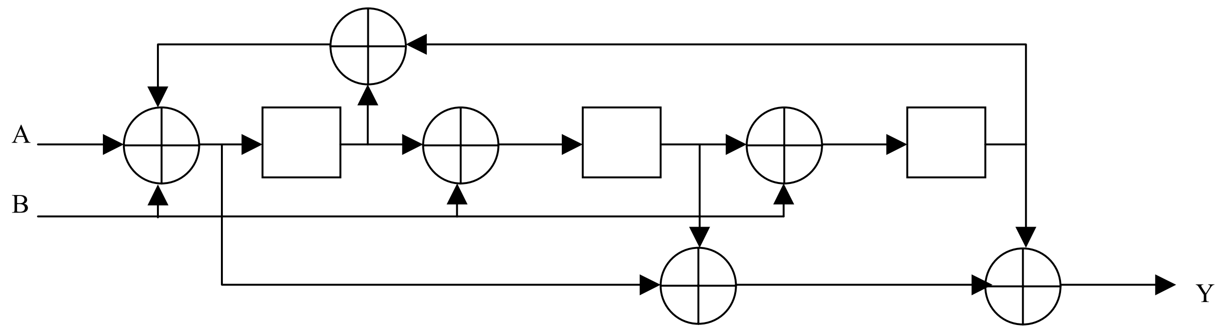


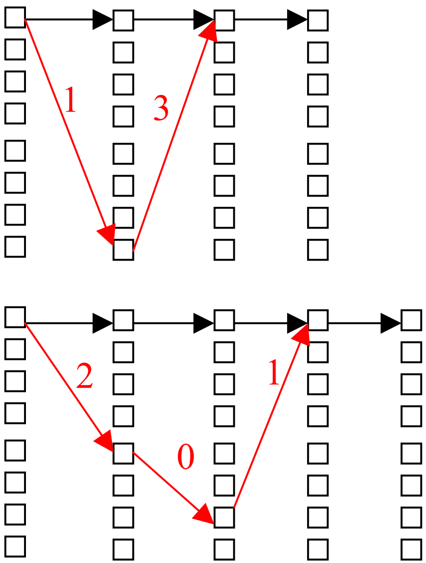
$d = 4+5 = 9$ $d = 4+6 = 10$ $d = 1+1+1+1 = 4$

RTZ sequences

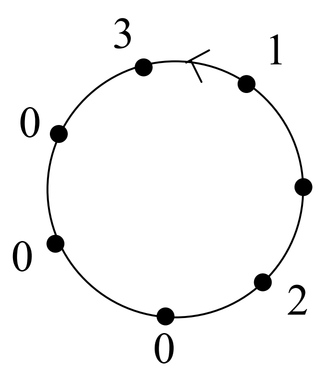
❖ Return to Zero sequence (RTZ) :

$$(A,B) = 2.A + B$$



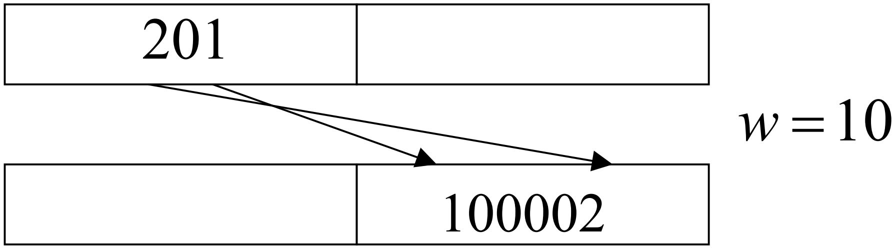


13	3 + 1 = 4
201	2 + 3 = 5
2003	3 + 2 = 5
30002	3 + 4 = 7
100002	2 + 5 = 7
3000001	3 + 5 = 8
10000001	2 + 6 = 8
20000002	2 + 6 = 8
30000003	4 + 4 = 8

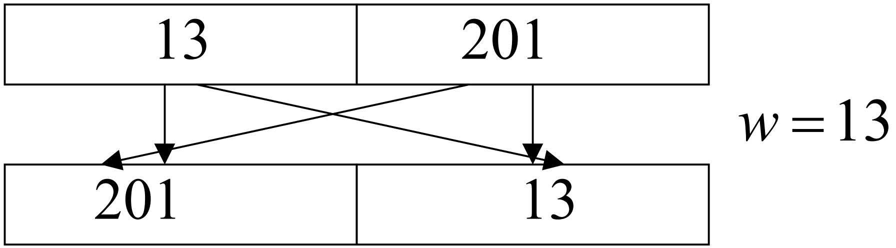


Errors patterns

- Primary error pattern:



- Secondary error pattern:



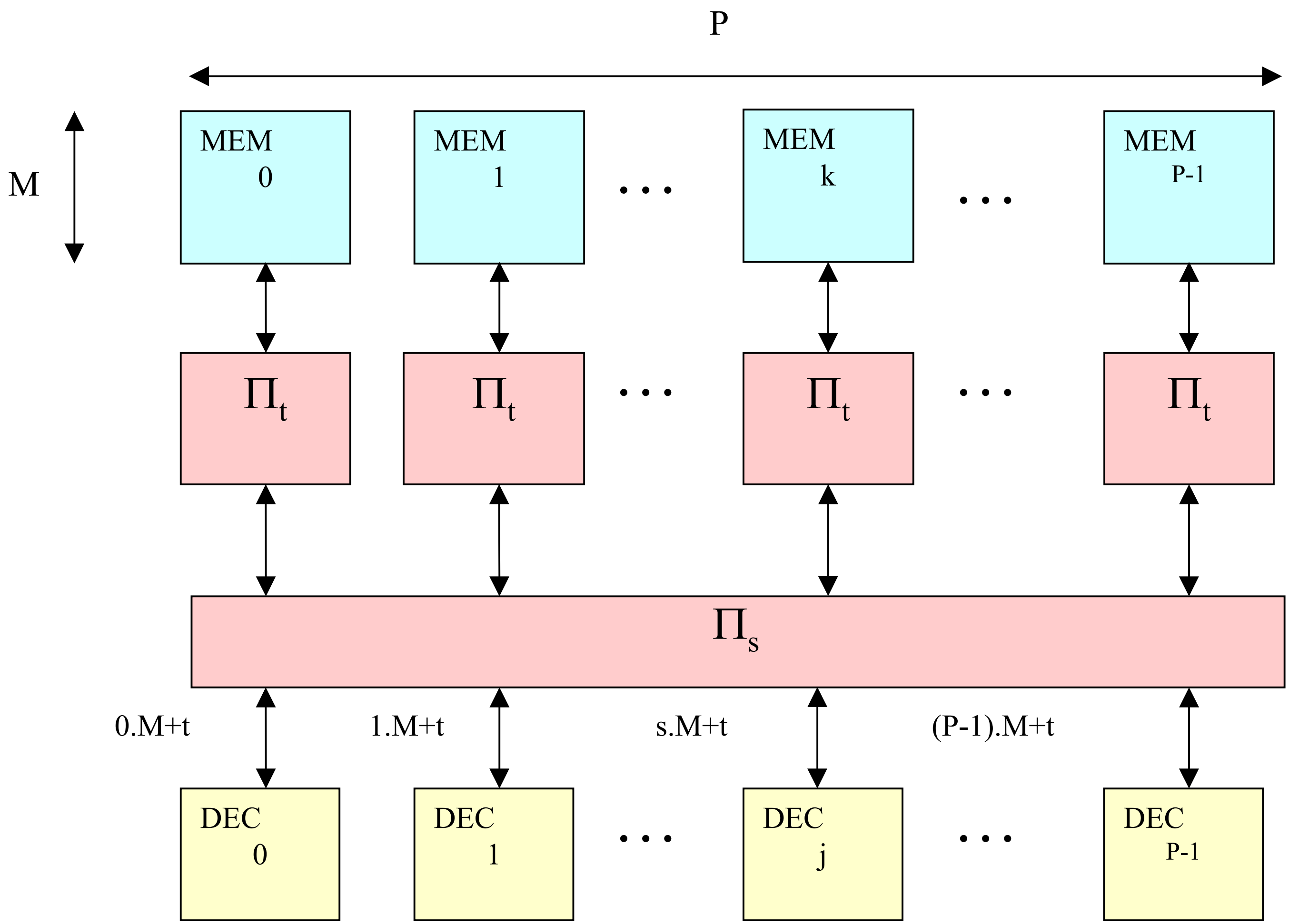
Error impulse method

C. Berrou, S. Vaton, *" Computing the Minimum Distances of Linear Codes by the Error Impulse Method"*, ISIT 2002, Lausanne, Switzerland, July 2002.

- Error impulse of amplitude A_i on the all-zero codeword x_0 for each information bit.

- Impulse distance : $d_{imp} = \max(A_i, \hat{x} = x_0)$
  - Maximal error impulse that the decoder can correct
- Minimal distance:minimum of the impulse distance $d_{\min} = \min_k(d_{imp}(k))$
- Periodicity of the code: puncturing, interleaver

Interleaver construction



Interleaver construction

- **Temporal permutation:** $\Pi_t(t) = \alpha.t + \beta(t \bmod 4) \bmod M$
  - α et M relatively primes
  - α chosen to maximize length of primary error pattern
  - β chosen to maximize minimum distance

- **Spatial permutation: rotation**

$$\Pi_s(t,s) = S(\xi) + s \bmod$$
$$\xi = t \bmod P$$

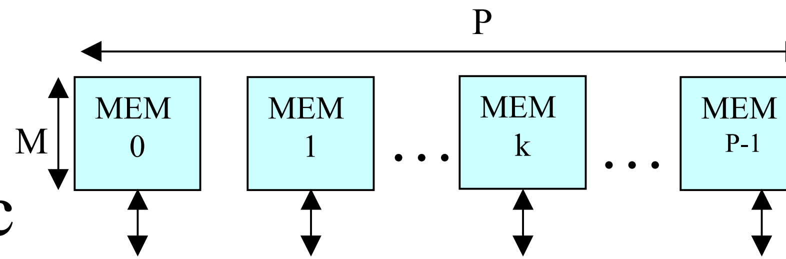
  - P-periodic function
  - S is a bijection chosen with the maximum of irregularity to improve convergence of the code

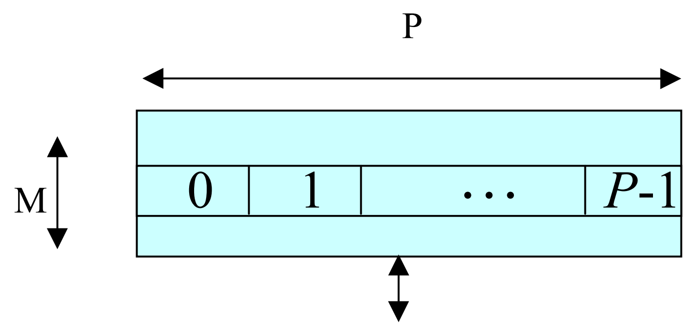
Hardware implementation

- Memory organization
- Decoder implementation
- Complexity evaluation

Memory organization

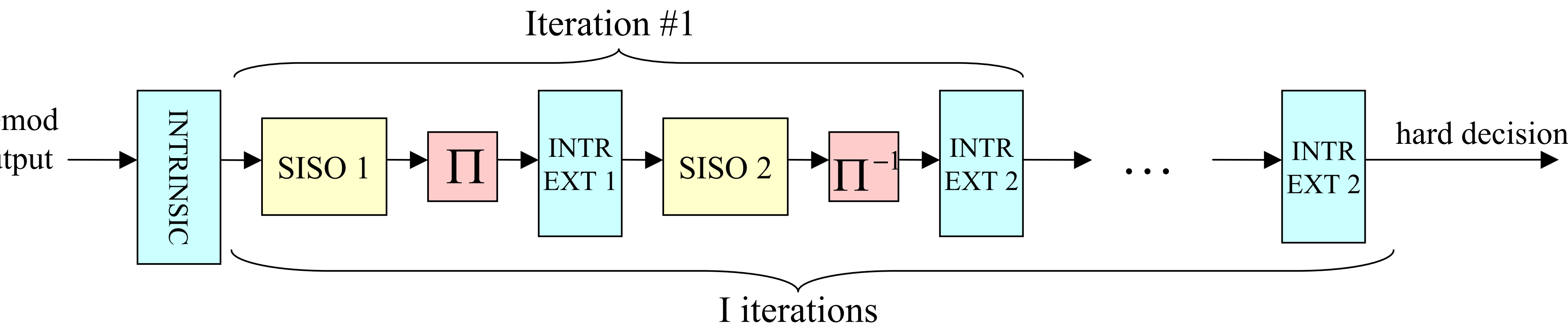
- **Permutation equations:**
  - Temporal permutation : addition
  - Spatial permutation : circular shift
- ➤ Easy to implement
- **Memory organization:**
  - *P* memory banks : intrinsic and extrinsic information for each symbol
  - Concatenation into a single memory



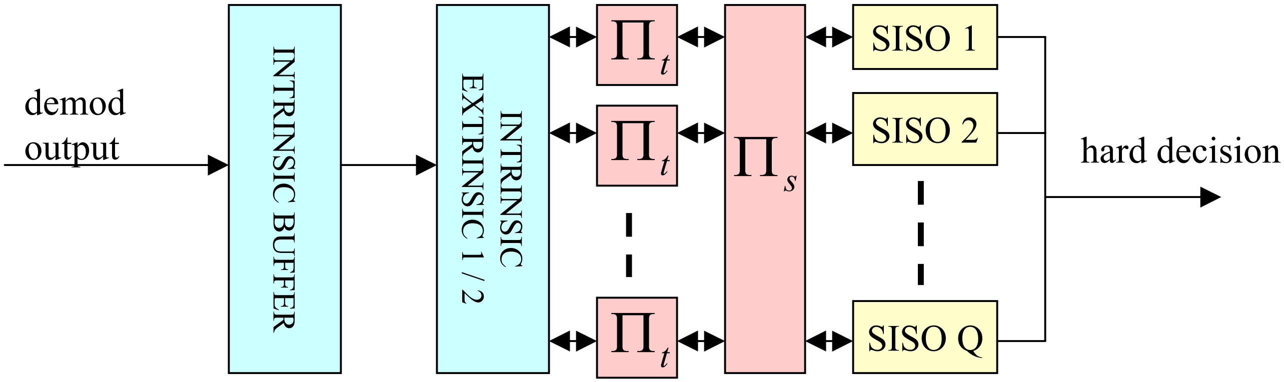


Decoder implementation

- Pipeline implementation:



- Parallel implementation:



Complexity evaluation

❖Complexity formula :

$$D = (1 + \eta_c \cdot I \cdot \eta_M) \cdot Mem(S \cdot 2 \cdot B_{IQ}) + \eta_c \cdot I \cdot \left[2 \cdot \eta_S \cdot SISO + 2 \cdot \eta_E \cdot Mem\left(4 \cdot \frac{N}{2} \cdot \mathrm{B}_e\right)\right]$$

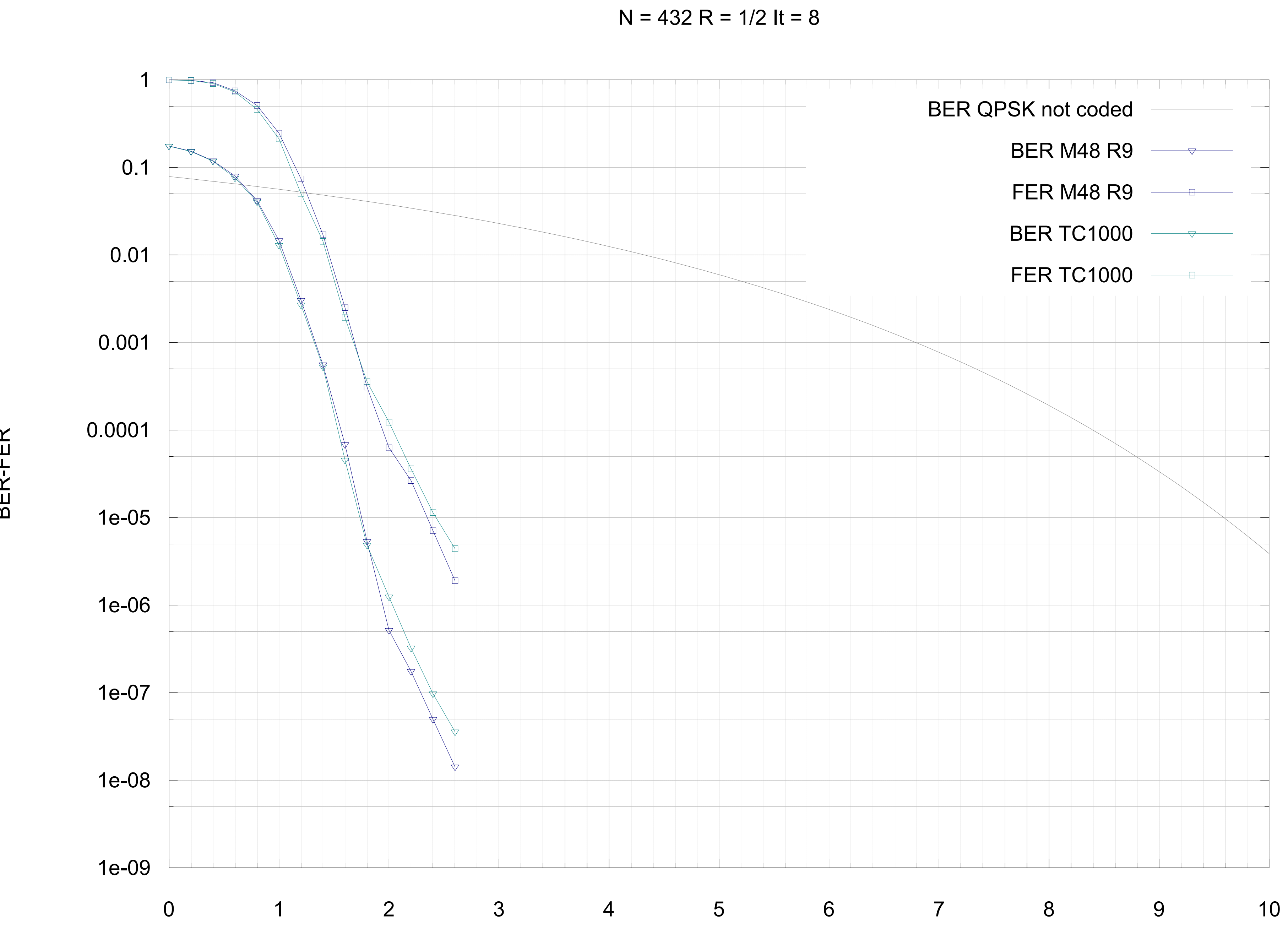
N : number of information symbols
S : number of channel symbols
I: number of iterations

Quantification:
B_{IQ} : number of bits for channel output
B_e : number of bits for extrinsic info.

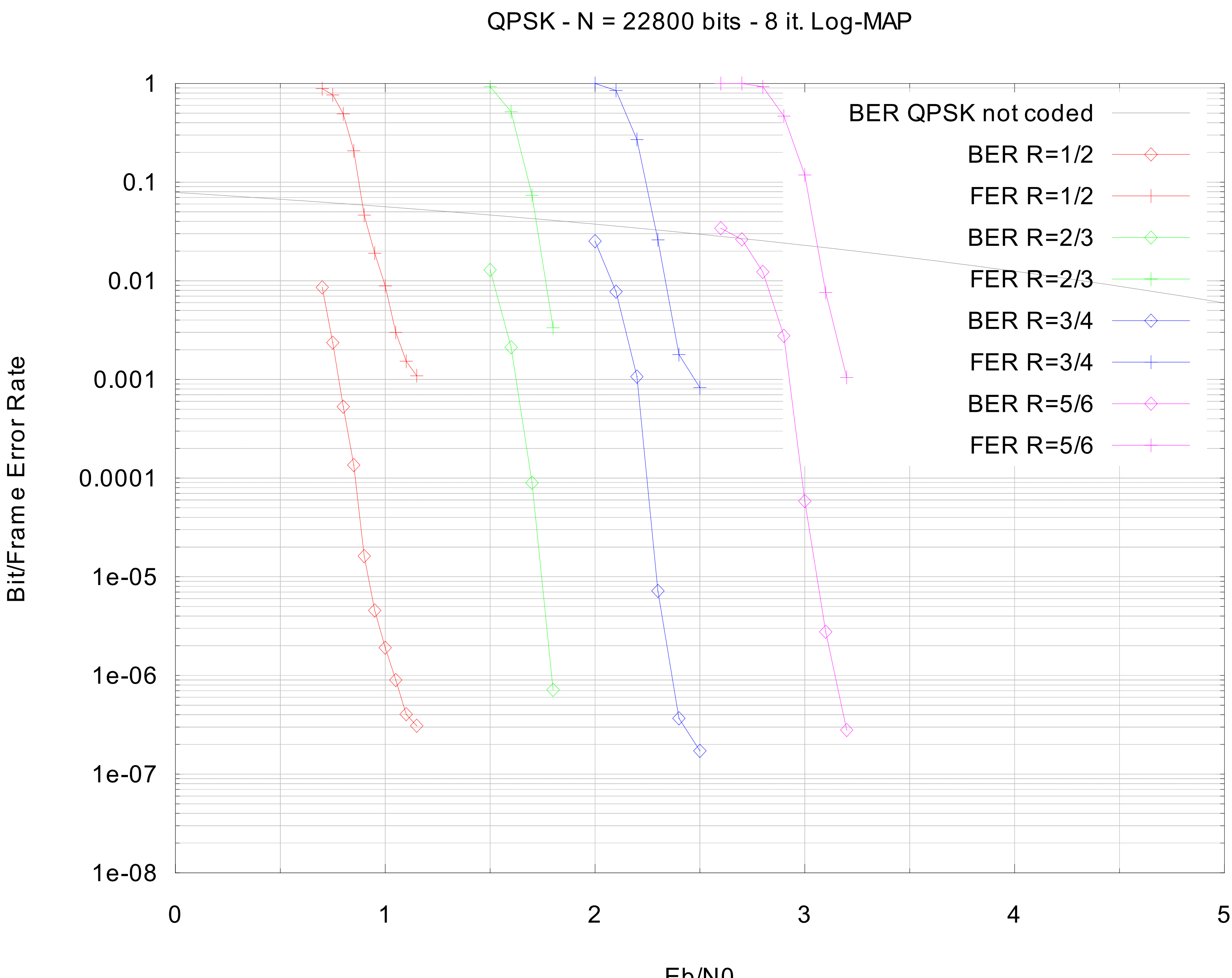
η_c: clock reuse factor
η_S: symbol reuse factor
η_M: input memory reuse factor
η_E: extrinsic memory reuse factor

Complexity comparison

Performance comparison



N = 432 R = 1/2 It = 8
BER-FER
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
1e-09
0
1
2
3
4
5
6
7
8
9
10
BER QPSK not coded
BER M48 R9
FER M48 R9
BER TC1000
FER TC1000



QPSK - N = 22800 bits - 8 it. Log-MAP
Bit/Frame Error Rate
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
1e-07
1e-08
0
1
2
3
4
5
Eb/N0
BER QPSK not coded
BER R=1/2
FER R=1/2
BER R=2/3
FER R=2/3
BER R=3/4
FER R=3/4
BER R=5/6
FER R=5/6

Conclusion

- Main results :
  - High degree of parallelism
  - No performance degradation
- Properties of the coding scheme
  - Interleaver is easy to implement in hardware thanks to closed-form equations
  - Interleaver is split into two levels : spatial and temporal permutation
  - Good convergence (spatial permutation), good minimum distance (temporal permutation).