

Contribution à l'amélioration des performances de décodage des turbo codes : algorithmes et architecture

Thibaud Tonnellier

5 Juillet 2017





Problématique industrielle

- Un des coeurs de métier de TAS : satellites de télécommunications
- Un standard de communication fige un protocole pour la transmission de l'information
- Or, les besoins applicatifs peuvent évoluer et ne plus correspondre à ceux définis
- Objectif : améliorer les performances des codes correcteurs d'erreurs sans modifier le standard

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

Plan

① Contexte des travaux de thèse

De la communication au codage de canal

Les turbo codes

② Abaissement du plancher d'erreurs des turbo codes

③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

Plan

① Contexte des travaux de thèse

De la communication au codage de canal

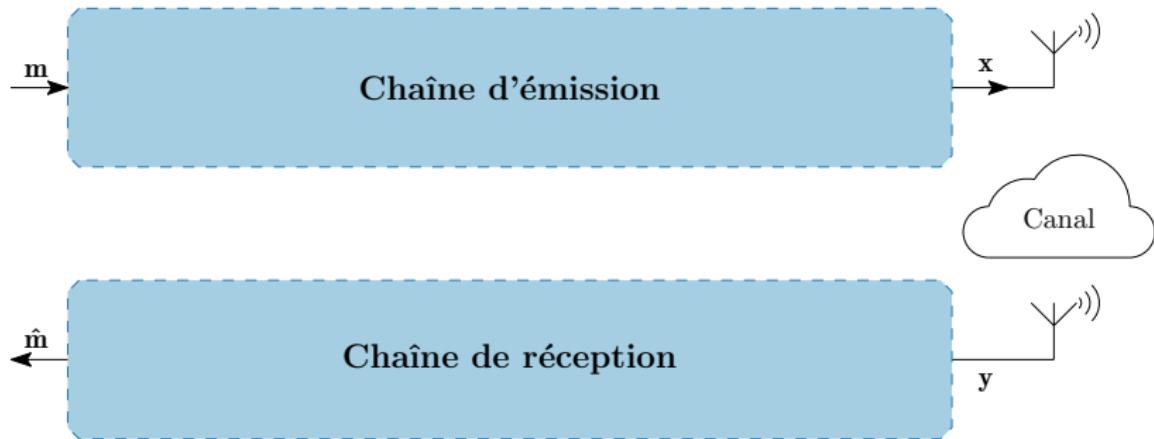
Les turbo codes

② Abaissement du plancher d'erreurs des turbo codes

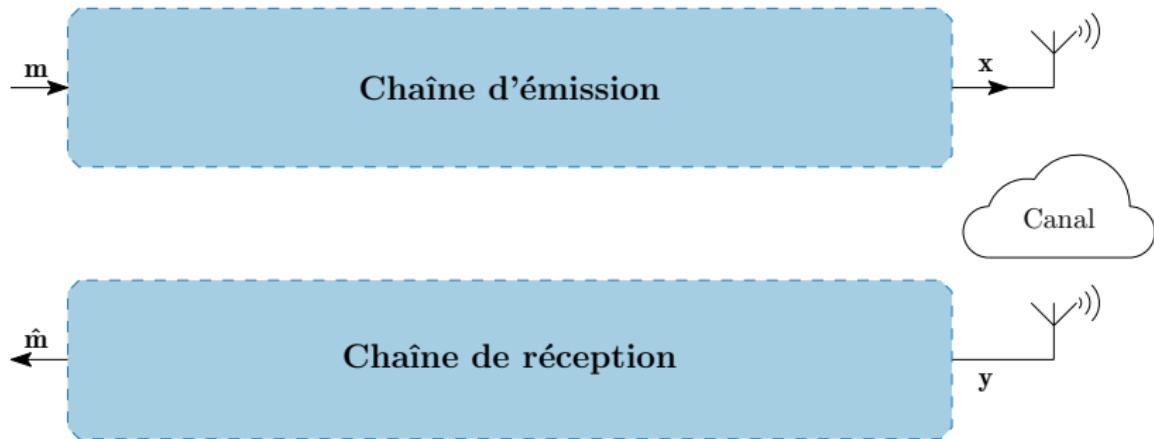
③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

Chaîne de communications numériques simplifiée

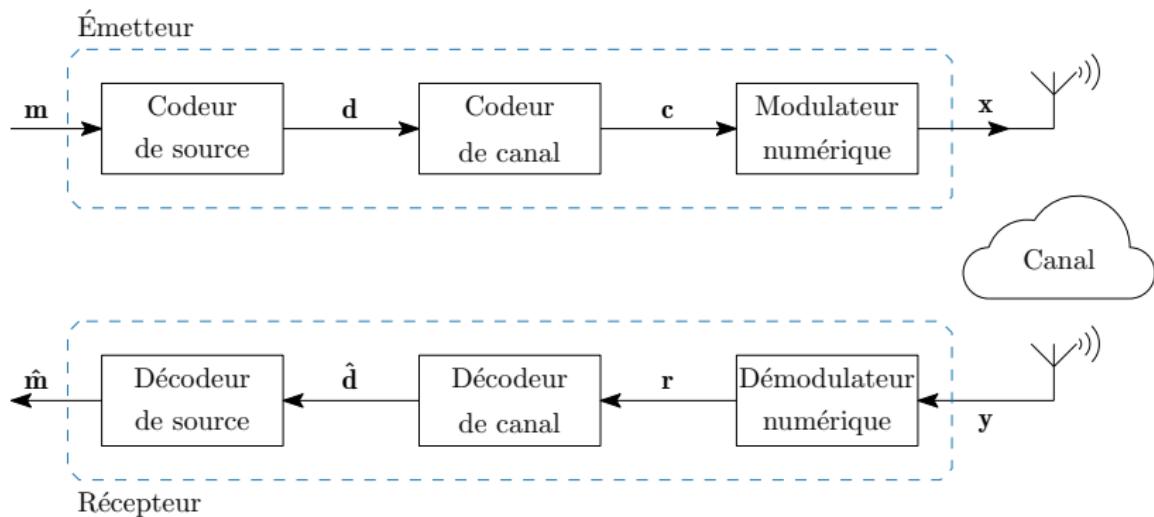


Chaîne de communications numériques simplifiée



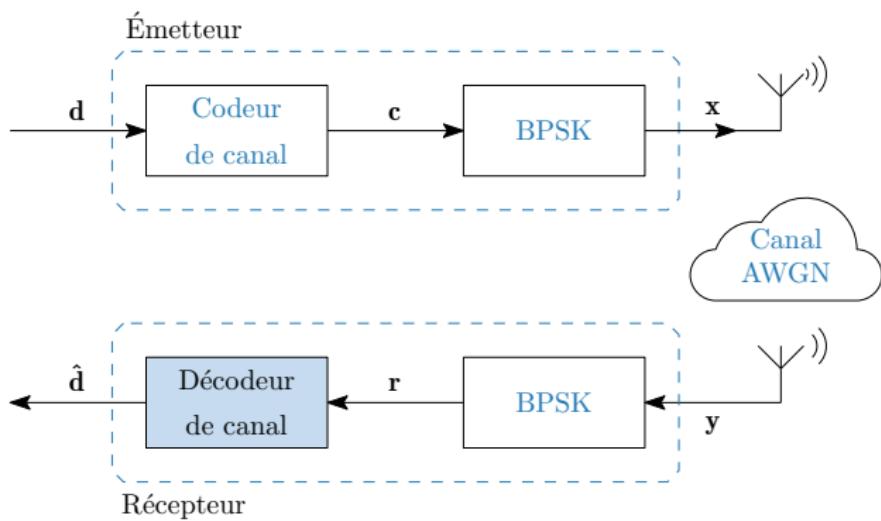
- Canal : perturbations (ex : bruit, interférences...)

Chaîne de communications numériques simplifiée



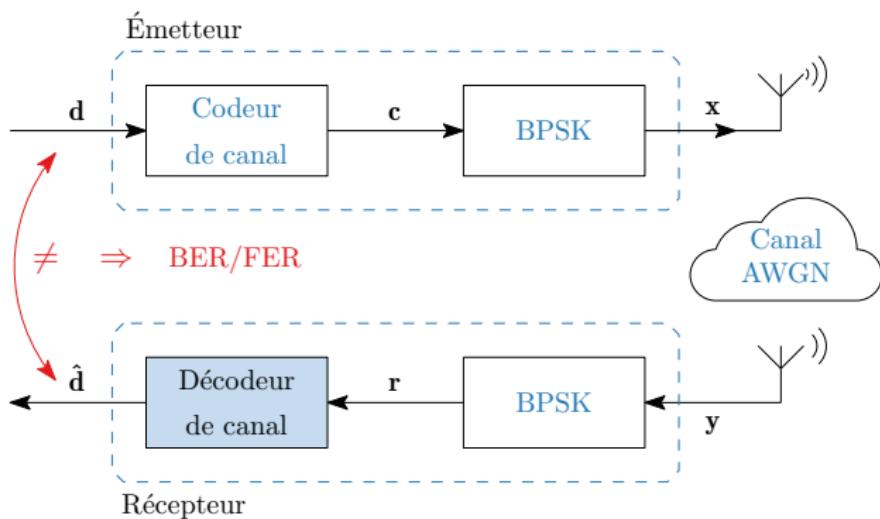
- **Canal** : perturbations (ex : bruit, interférences...)
- **Codage de source** : obtenir le maximum de concision dans l'expression d'un message
- **Codage de canal** : rendre robuste un message face aux perturbations du canal

Chaîne de communications numériques simplifiée



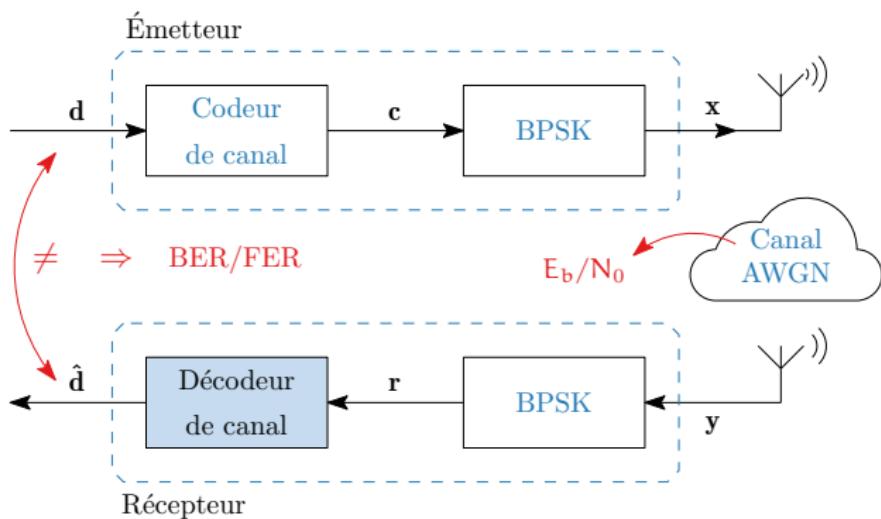
- BPSK : modulation de phase à deux valeurs
 - AWGN : bruit additif blanc Gaussien
- couramment utilisés pour évaluer les codes correcteurs d'erreurs

Chaîne de communications numériques simplifiée



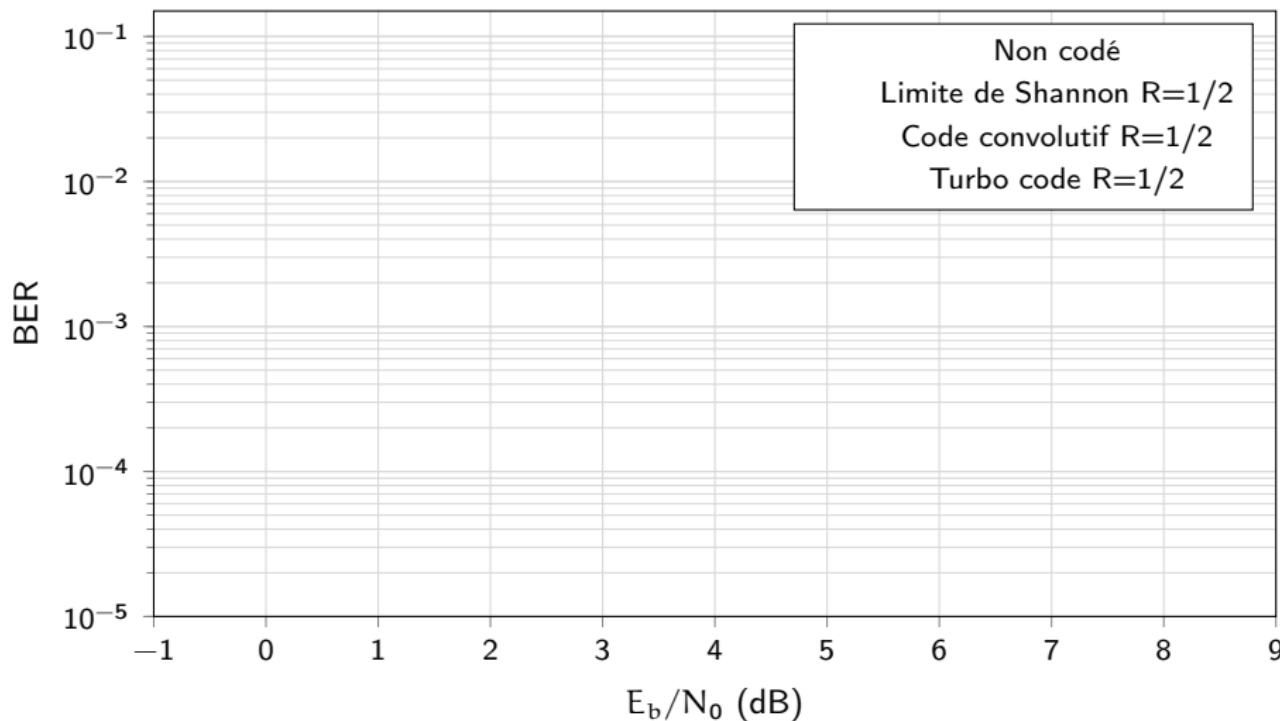
- BPSK : modulation de phase à deux valeurs
- AWGN : bruit additif blanc Gaussien
- couramment utilisés pour évaluer les codes correcteurs d'erreurs

Chaîne de communications numériques simplifiée

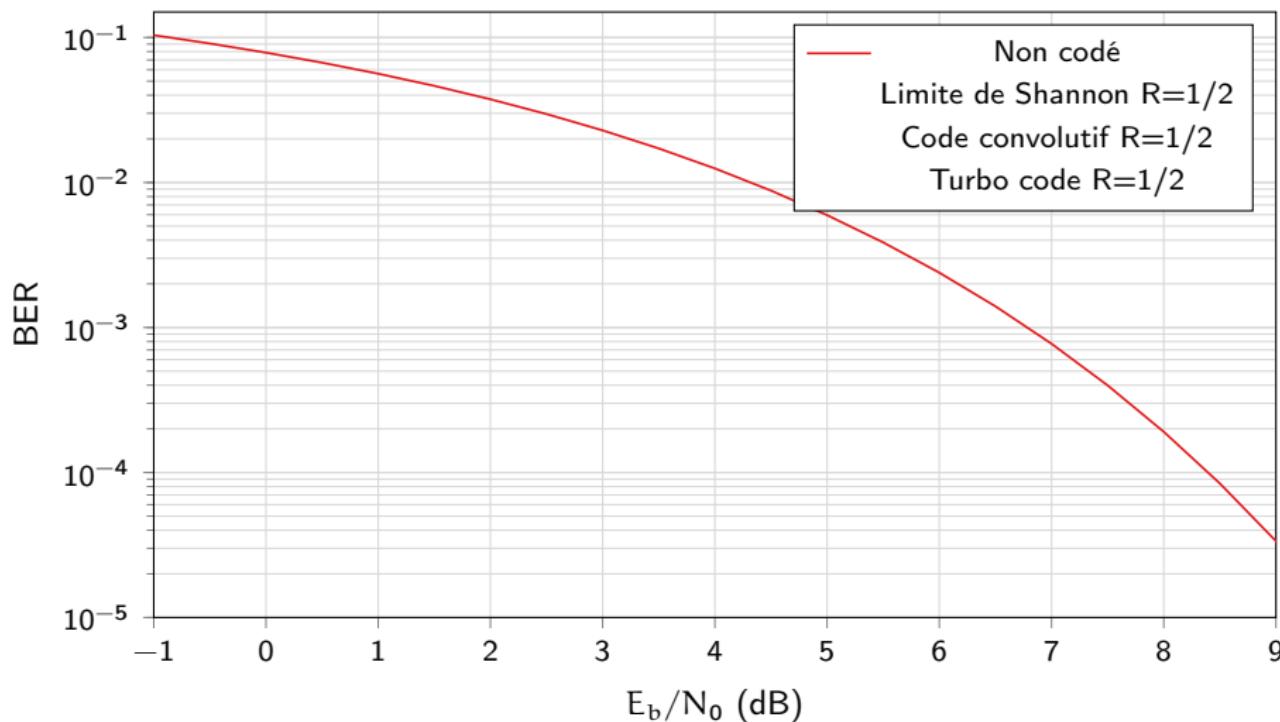


- BPSK : modulation de phase à deux valeurs
- AWGN : bruit additif blanc Gaussien
- couramment utilisés pour évaluer les codes correcteurs d'erreurs

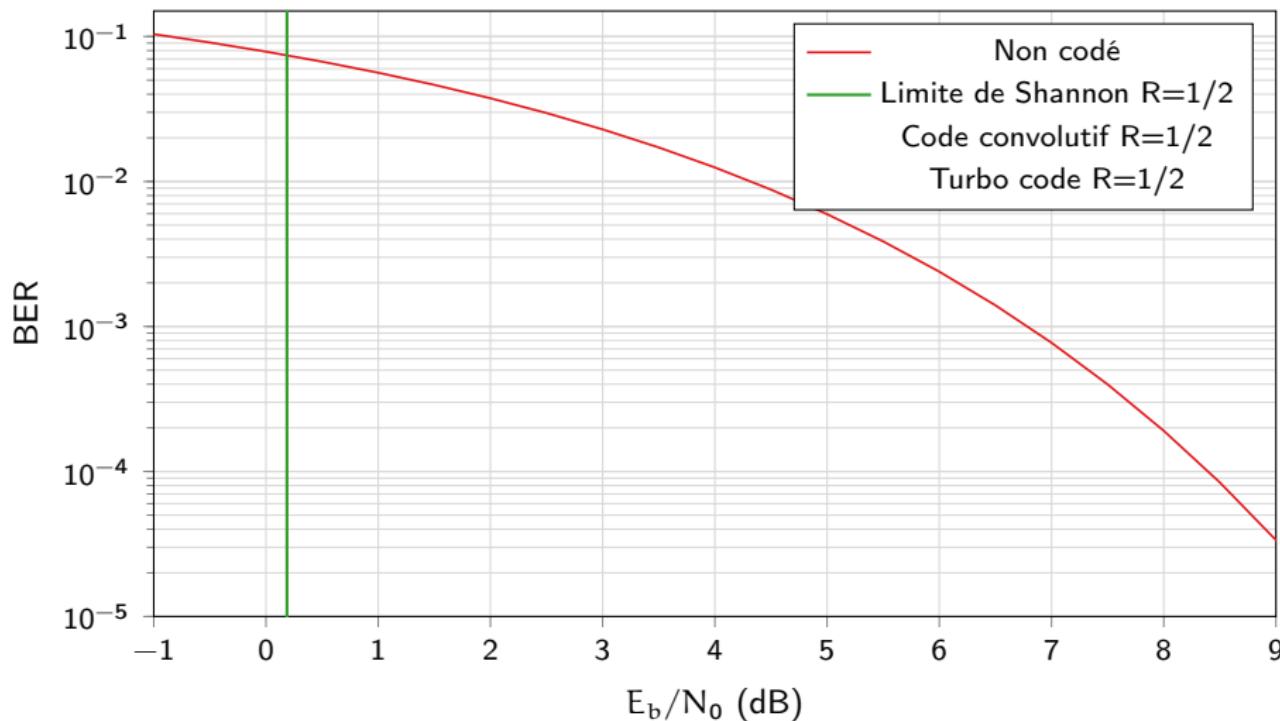
Courbes de performances et limites



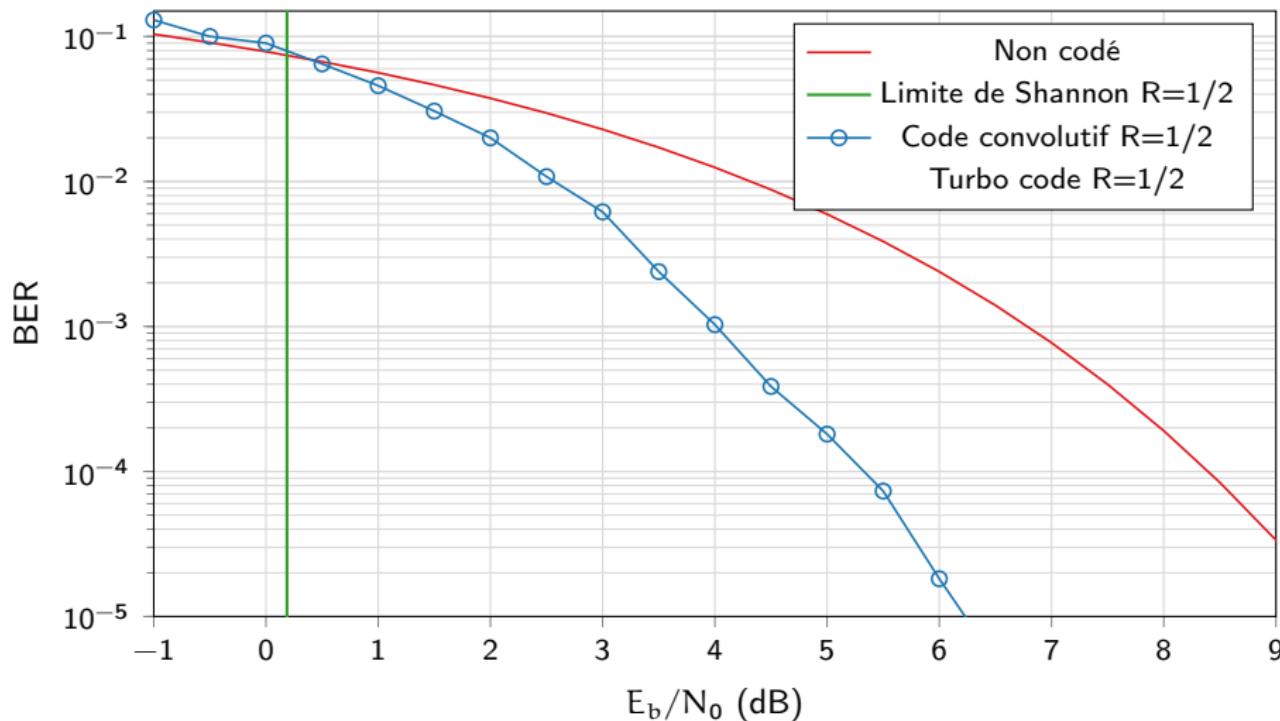
Courbes de performances et limites



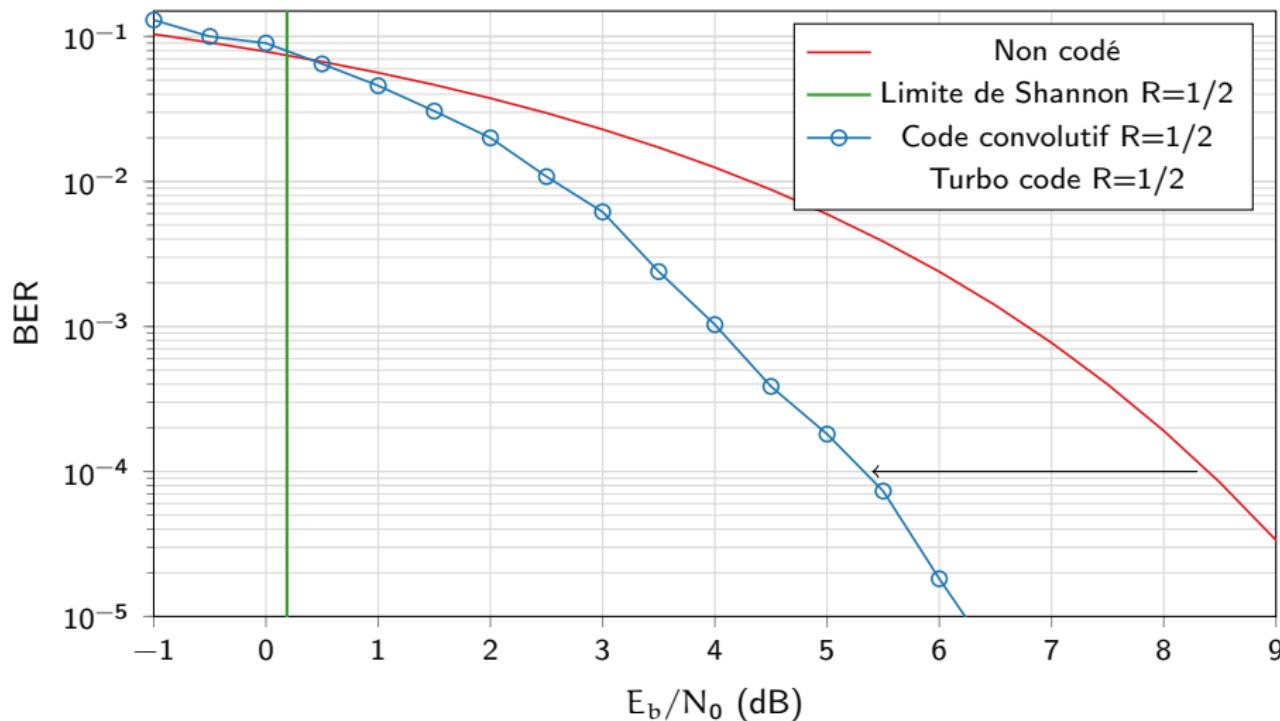
Courbes de performances et limites



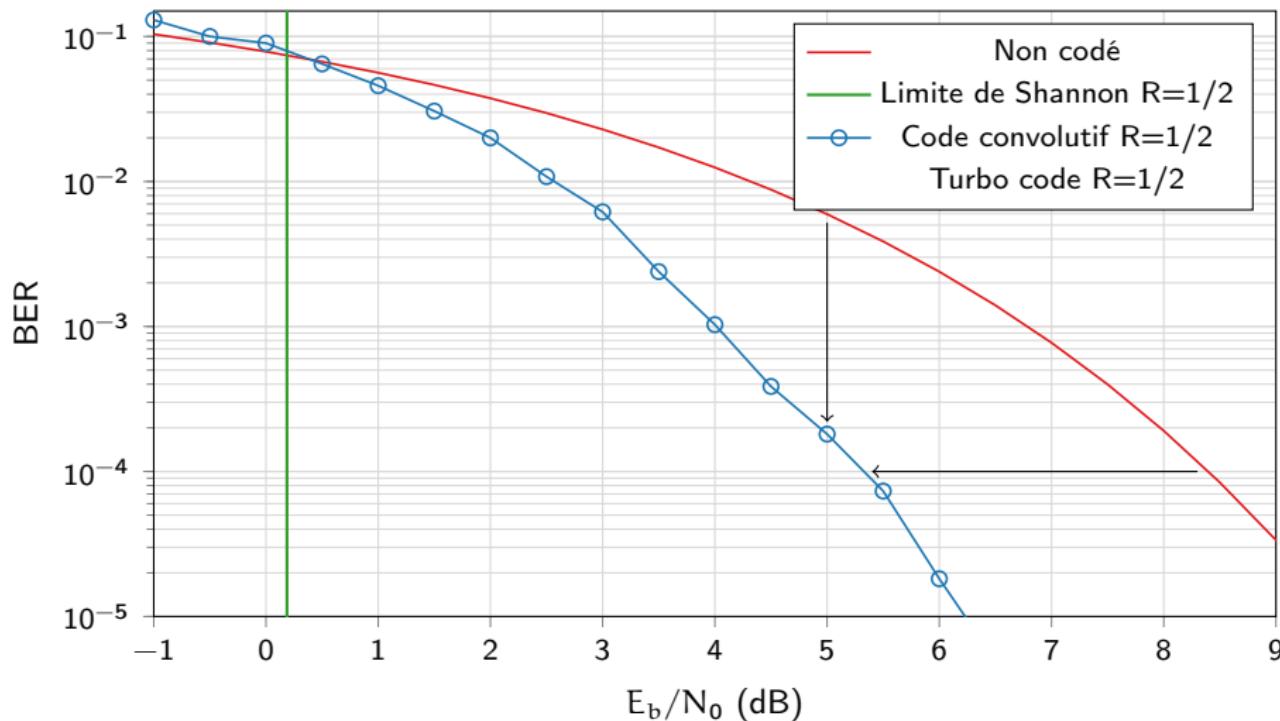
Courbes de performances et limites



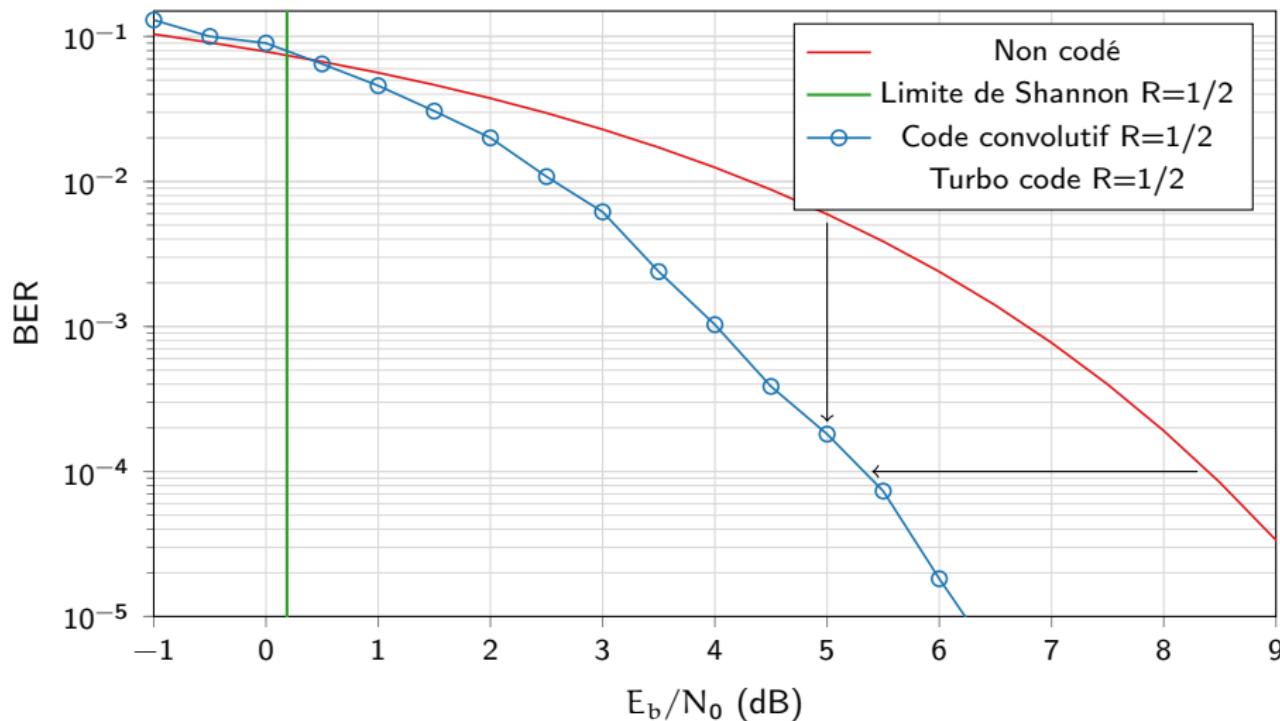
Courbes de performances et limites



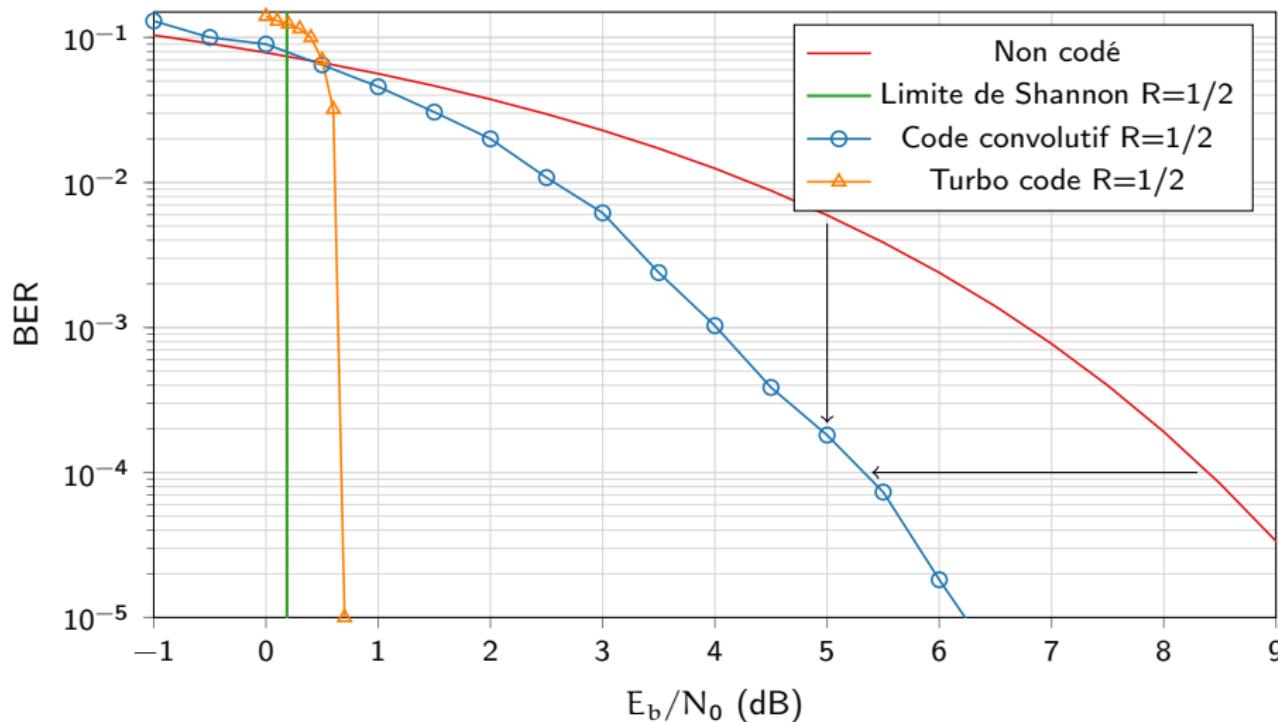
Courbes de performances et limites



Courbes de performances et limites



Courbes de performances et limites



Plan

① Contexte des travaux de thèse

De la communication au codage de canal

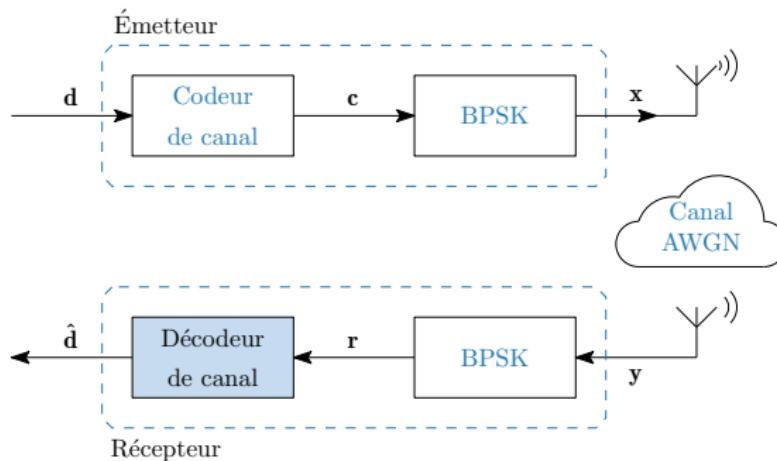
Les turbo codes

② Abaissement du plancher d'erreurs des turbo codes

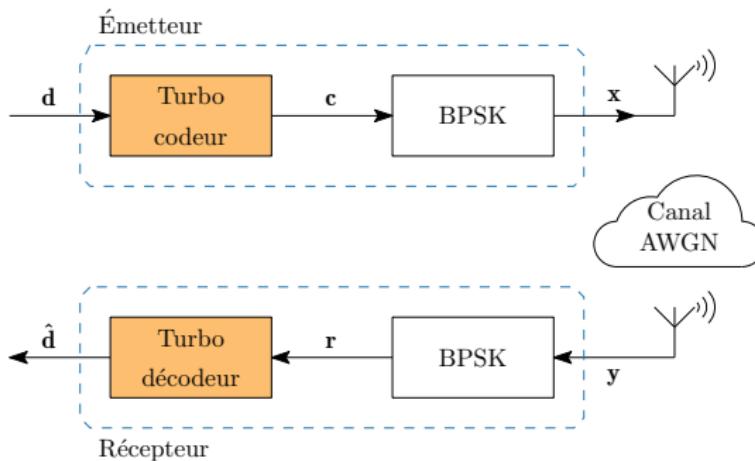
③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

Sa place dans la chaîne de communications numériques

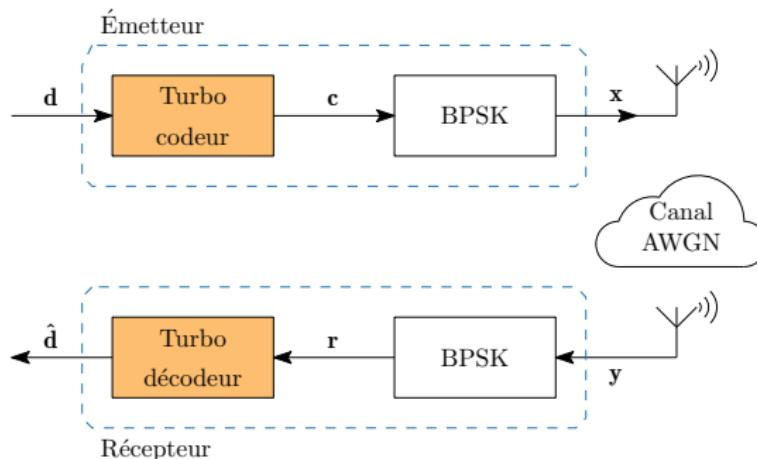


Sa place dans la chaîne de communications numériques



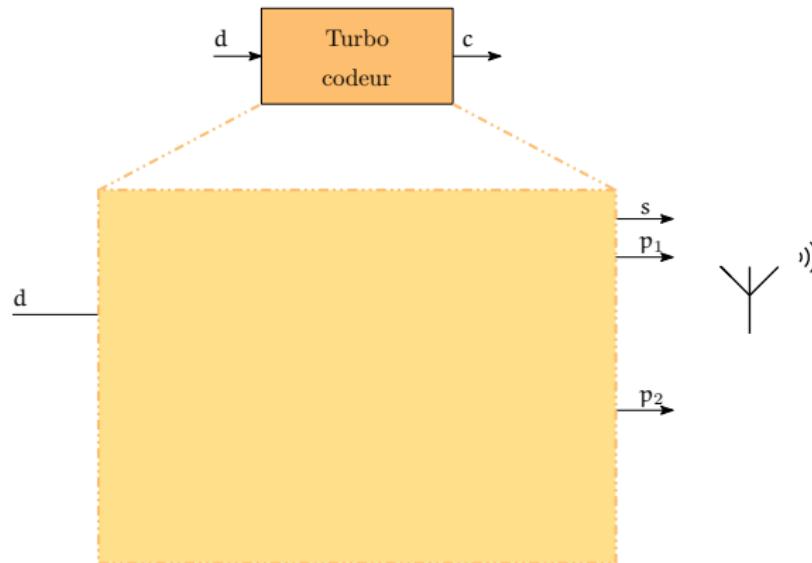
- L'un des codes correcteurs d'erreurs les plus efficaces

Sa place dans la chaîne de communications numériques



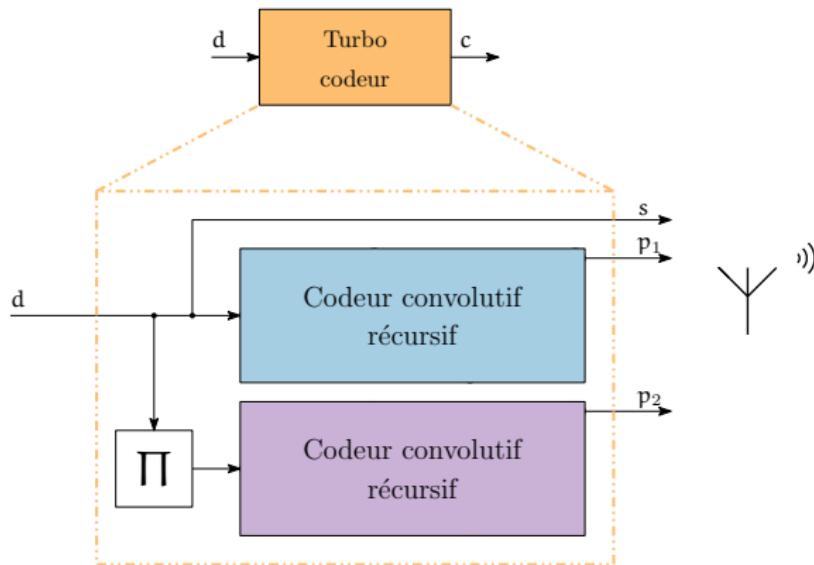
- L'un des codes correcteurs d'erreurs les plus efficaces
- Excellent compromis entre la complexité calculatoire et les performances de décodage
- Choisis dans plusieurs standards de communications numériques sans-fil :
 - Contexte satellitaire (CCSDS, DVB-RCS)
 - Contexte mobile (LTE)

La fonction de turbo codage



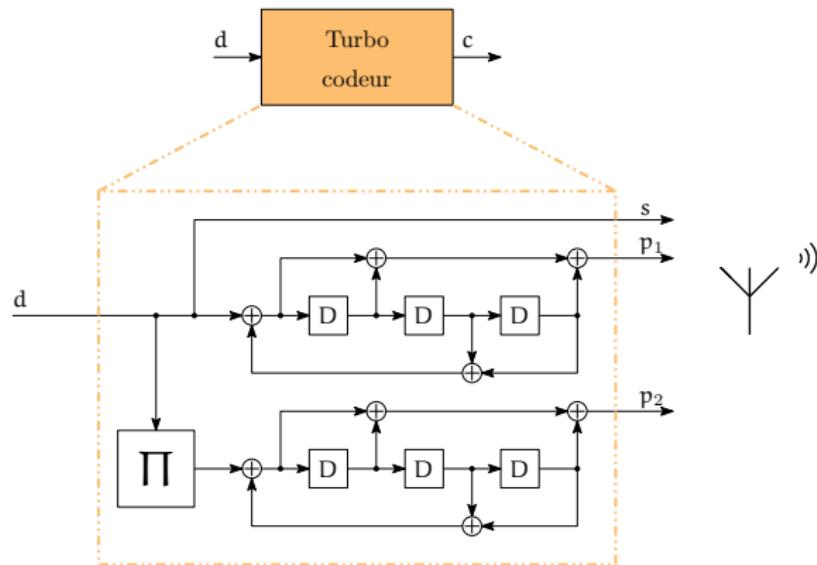
- Rendement natif : $1/3$ $c = [s, p_1, p_2]$
- d : binaire ou double binaire

La fonction de turbo codage



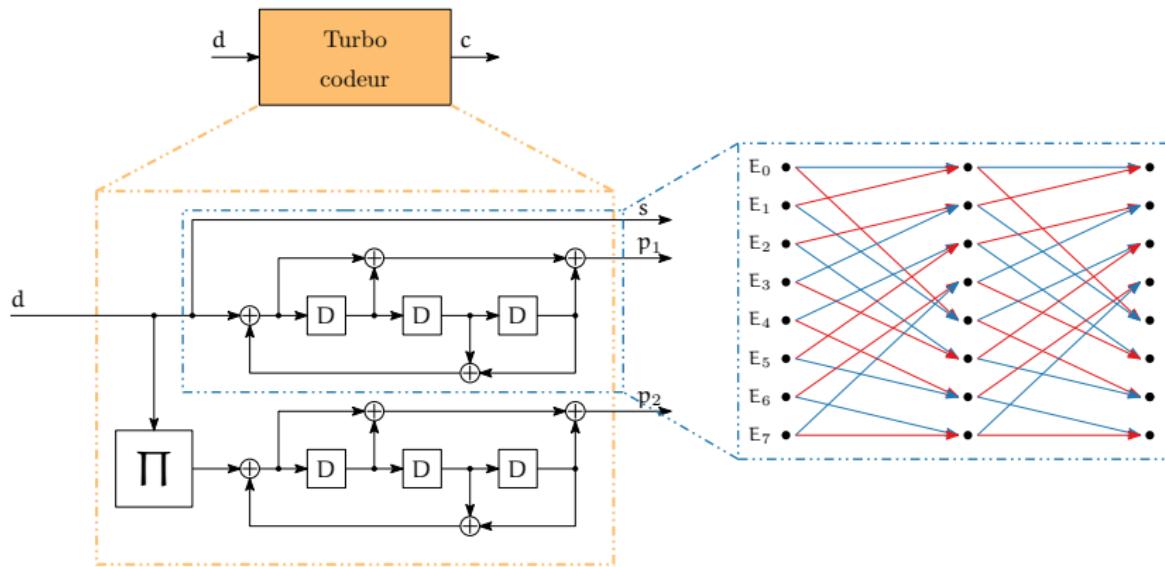
- Trois blocs élémentaires : 2 codeurs convolutifs + entrelaceur (Π)
- Rendement natif : $1/3$ $c = [s, p_1, p_2]$

La fonction de turbo codage



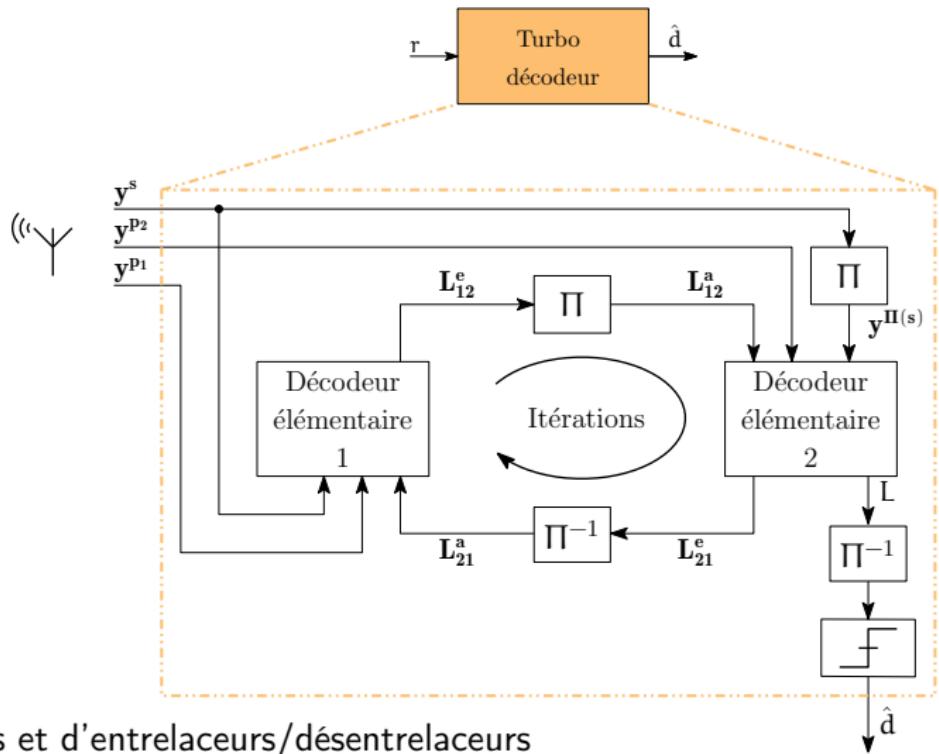
- Trois blocs élémentaires : 2 codeurs convolutifs + entrelaceur (Π)
- Rendement natif : $1/3 \ c = [s, p_1, p_2]$

La fonction de turbo codage



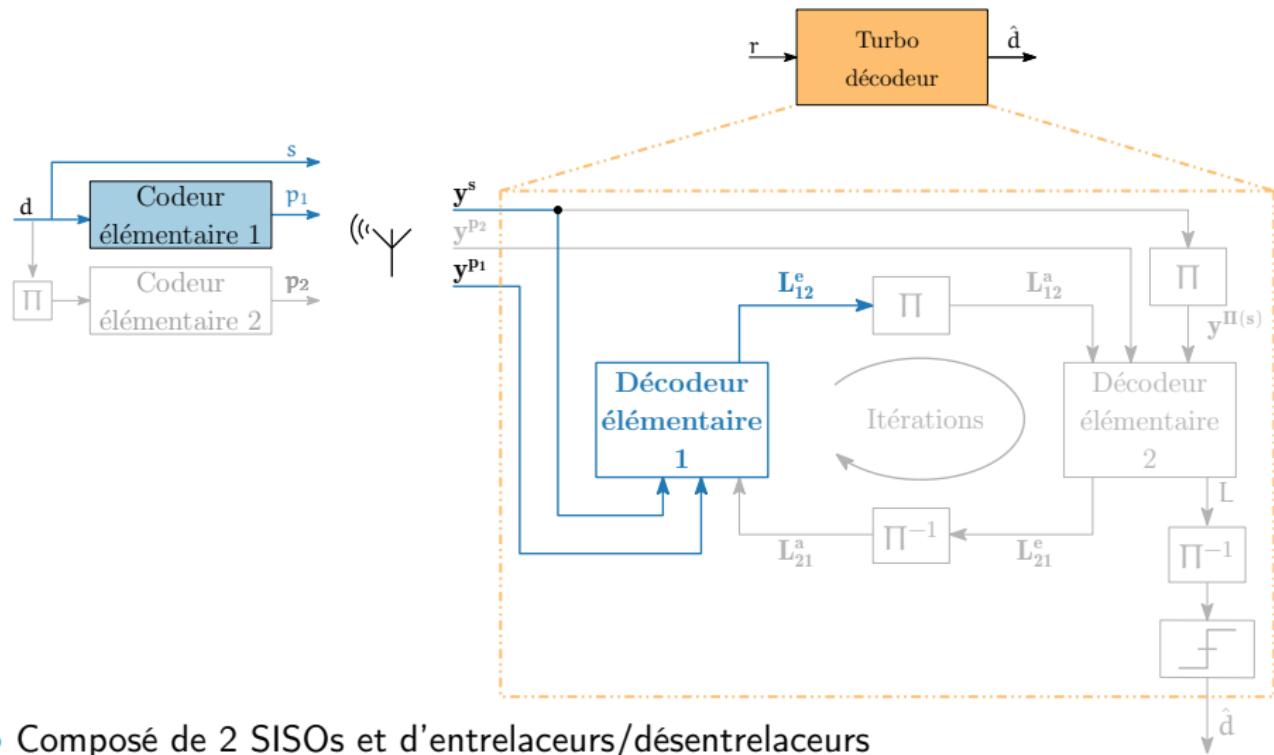
- Trois blocs élémentaires : 2 codeurs convolutifs + entrelaceur (Π)
- Rendement natif : $1/3 \ c = [s, p_1, p_2]$
- Représentation en treillis

La fonction de turbo décodage



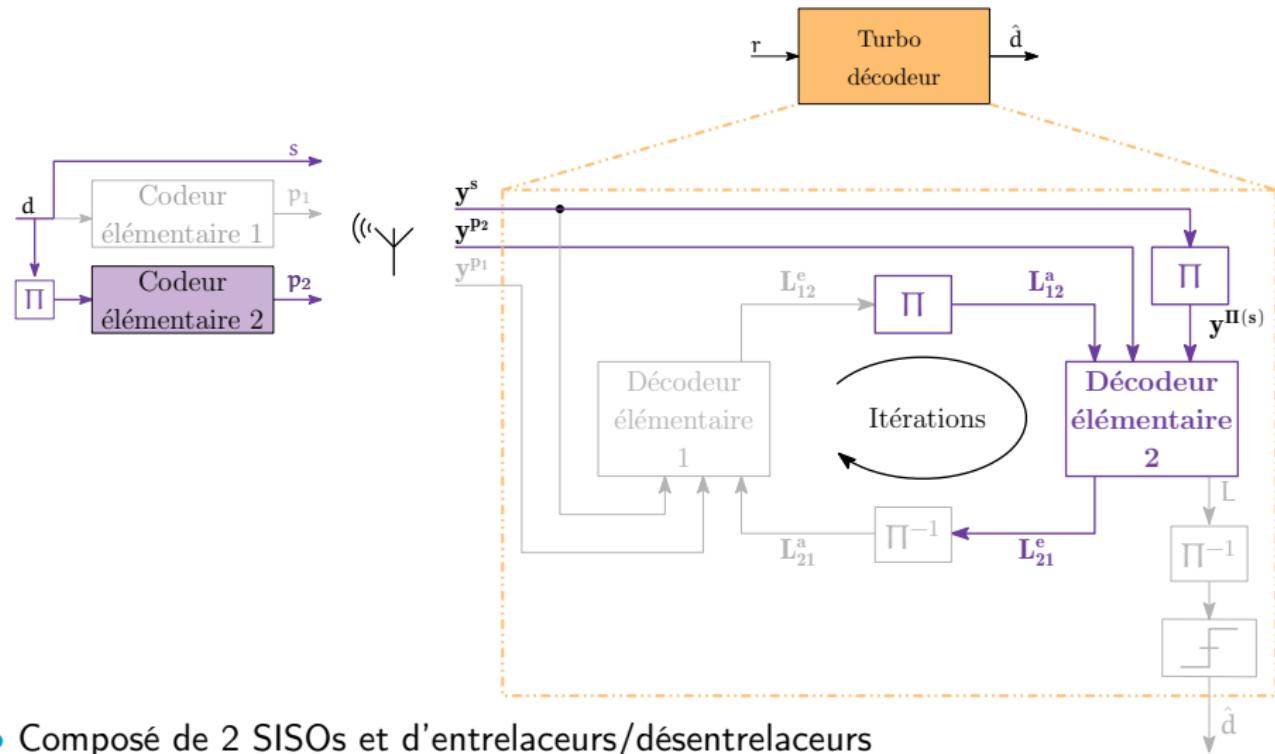
- Composé de 2 SISOs et d'entrelaceurs/désentrelaceurs

La fonction de turbo décodage



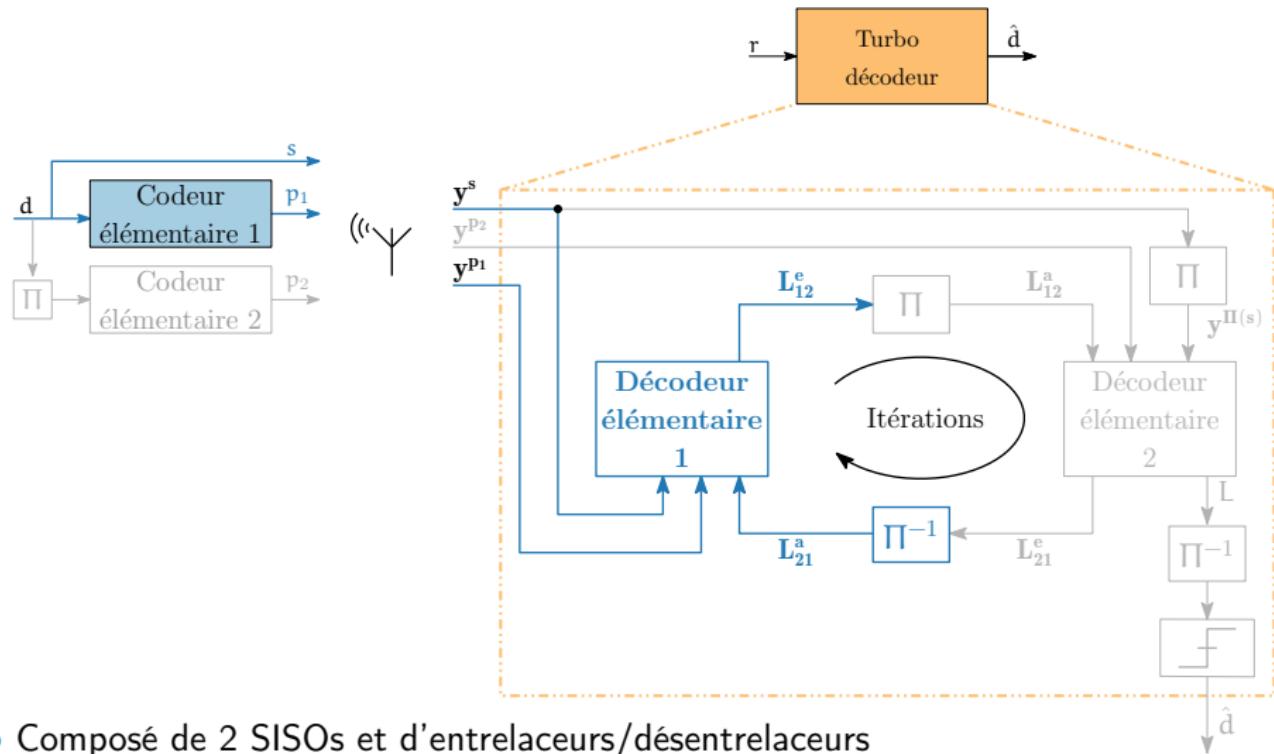
- Composé de 2 SISOs et d'entrelaceurs/désentrelaceurs
- Processus itératif : SISOs échangent leurs avis via l'information extrinsèque (L^e)

La fonction de turbo décodage



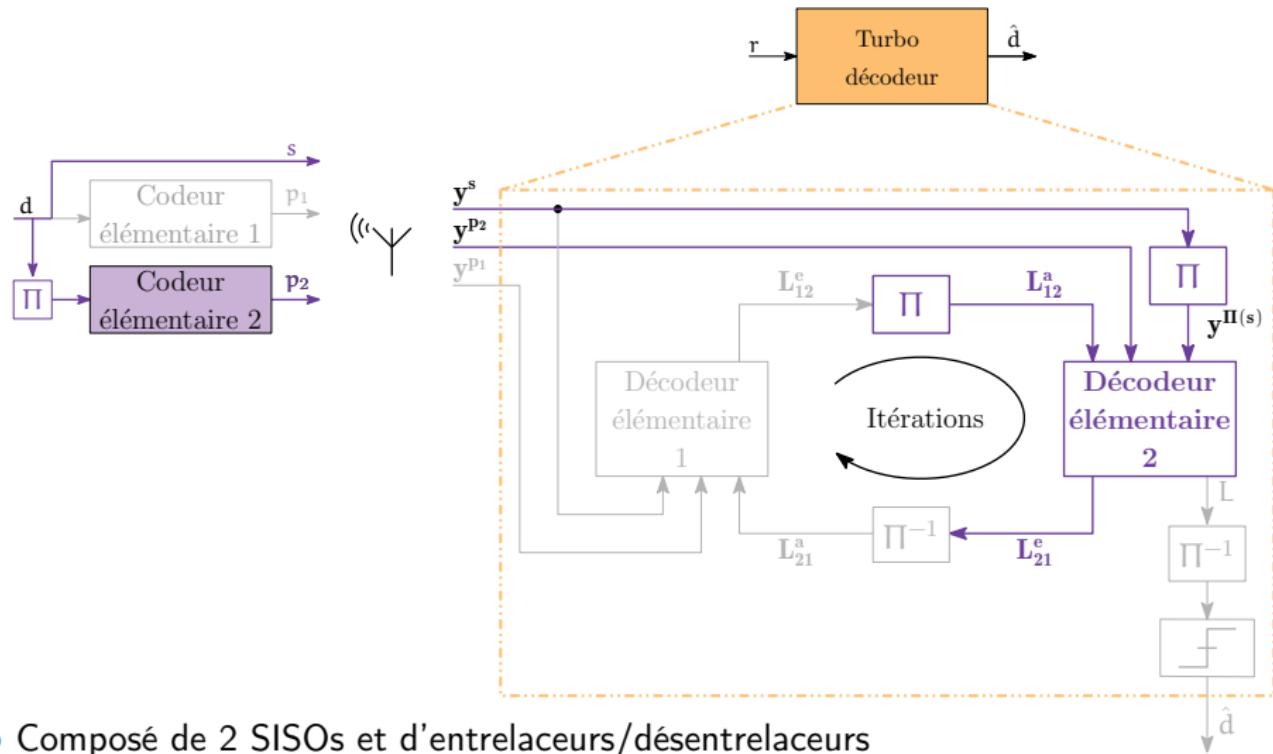
- Composé de 2 SISOs et d'entrelaceurs/désentrelaceurs
- Processus itératif : SISOs échangent leurs avis via l'information extrinsèque (L^e)

La fonction de turbo décodage



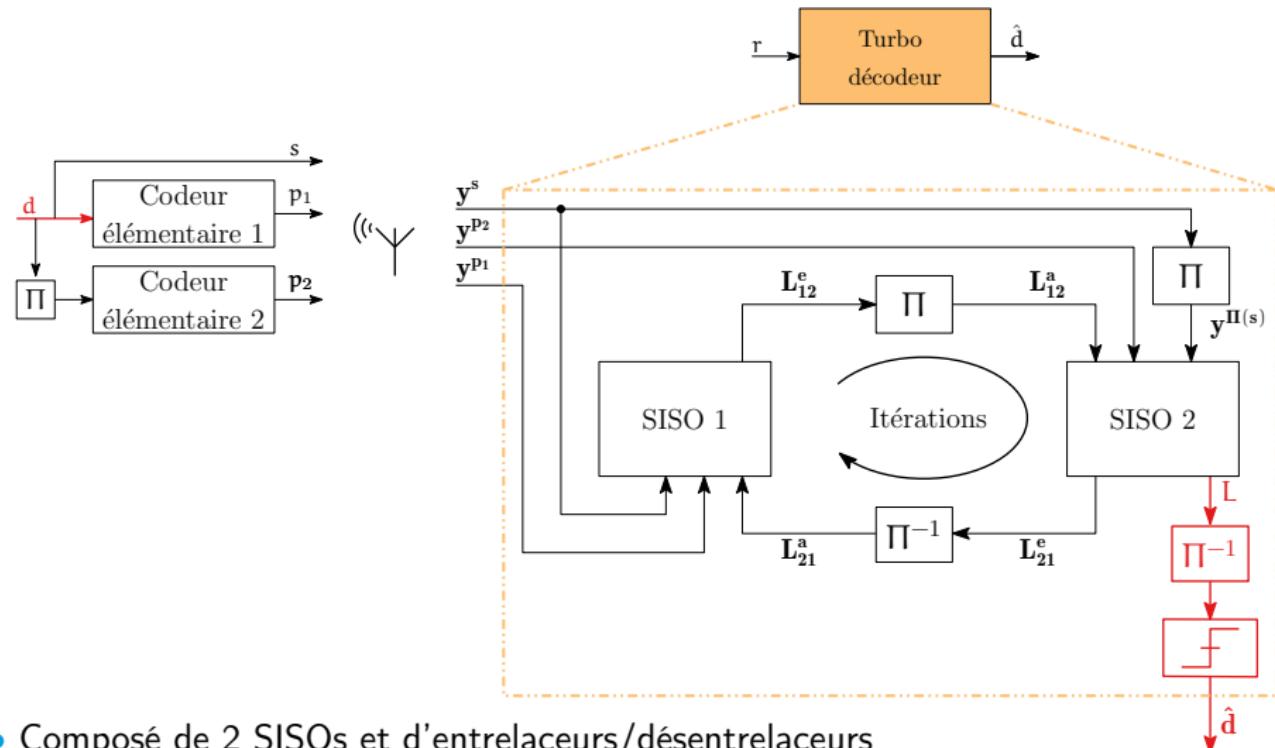
- Composé de 2 SISOs et d'entrelaceurs/désentrelaceurs
- Processus itératif : SISOs échangent leurs avis via l'information extrinsèque (L^e)

La fonction de turbo décodage



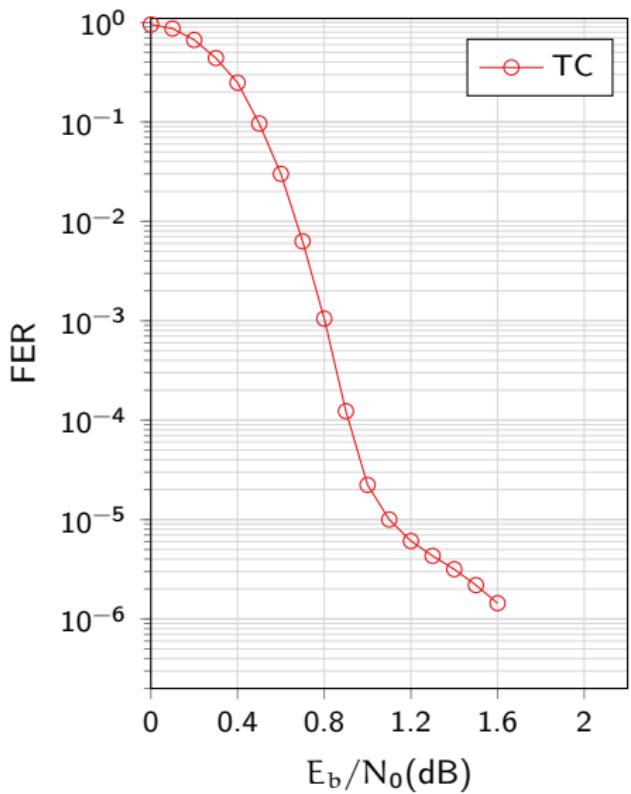
- Composé de 2 SISOs et d'entrelaceurs/désentrelaceurs
- Processus itératif : SISOs échangent leurs avis via l'information extrinsèque (L^e)

La fonction de turbo décodage



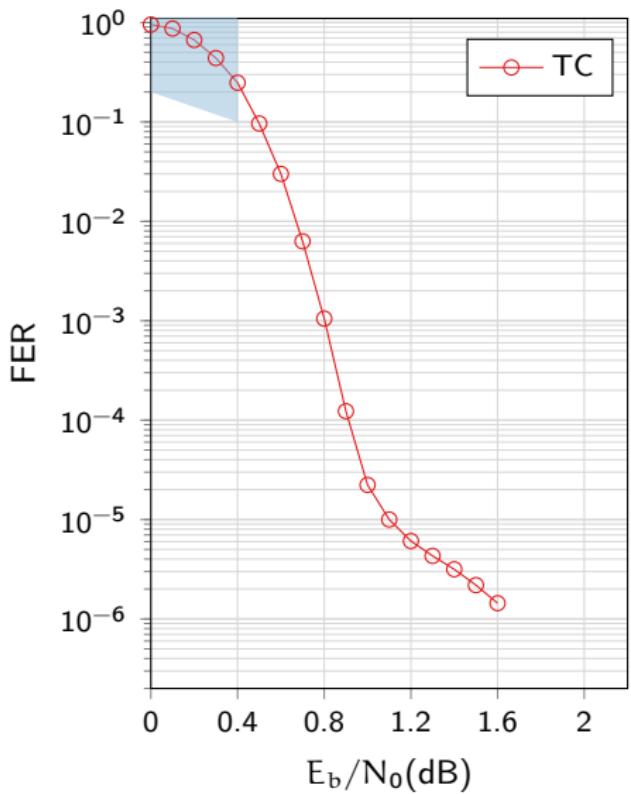
- Composé de 2 SISOs et d'entrelaceurs/désentrelaceurs
- Processus itératif : SISOs échangent leurs avis via l'information extrinsèque (L^e)

Performances - LTE, K = 2048, R=1/3, 8 its, virgule flottante, AWGN, BPSK



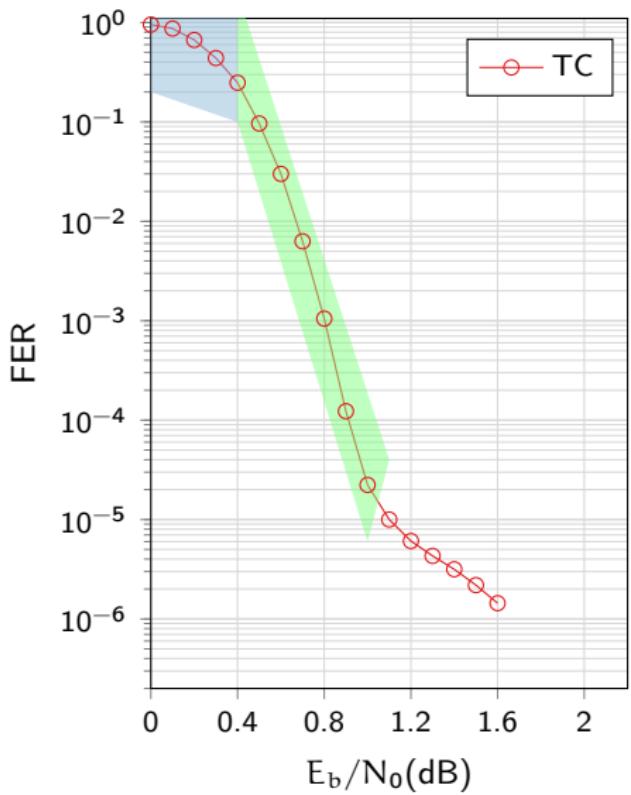
- Cas d'étude : standard LTE
- Trois zones :

Performances - LTE, K = 2048, R=1/3, 8 its, virgule flottante, AWGN, BPSK



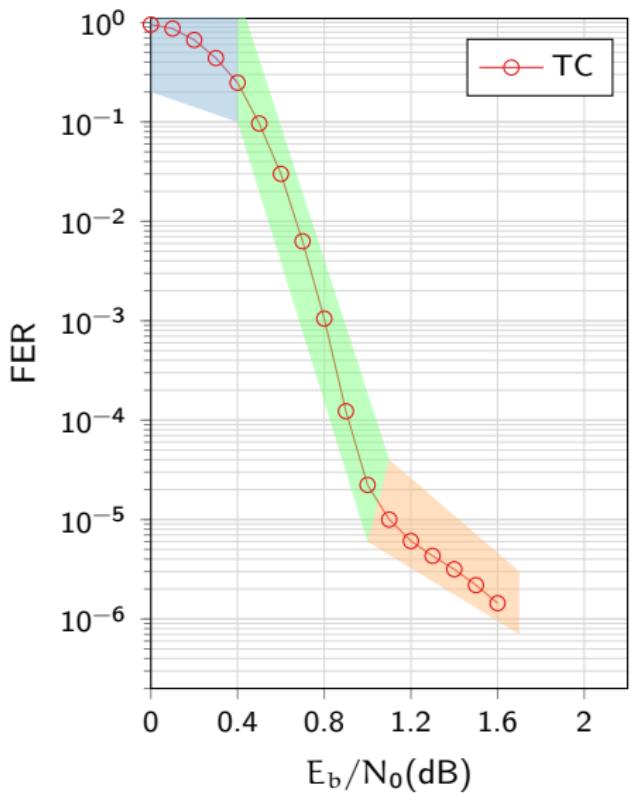
- Cas d'étude : standard LTE
- Trois zones :
 - Non convergence

Performances - LTE, K = 2048, R=1/3, 8 its, virgule flottante, AWGN, BPSK



- Cas d'étude : standard LTE
- Trois zones :
 - Non convergence
 - Convergence (waterfall)

Performances - LTE, K = 2048, R=1/3, 8 its, virgule flottante, AWGN, BPSK



- Cas d'étude : standard LTE
- Trois zones :
 - Non convergence
 - Convergence (waterfall)
 - Plancher d'erreurs (error floor)
 - La courbe de performances s'aplatit à fort SNR
 - Empêche l'obtention de très faibles taux d'erreurs pour de faibles SNR.

Quelles solutions pour l'amélioration des performances de décodage ?

① Modifier le code

- Entrelaceur
- Nombre d'états du codeur
- Concaténation (BCH, RSC-1 par exemple)

Quelles solutions pour l'amélioration des performances de décodage ?

① ~~Modifier le code~~

Quelles solutions pour l'amélioration des performances de décodage ?

① ~~Modifier le code~~

② Modifier l'algorithme de décodage

Quelles solutions pour l'amélioration des performances de décodage ?

① ~~Modifier le code~~

② Modifier l'algorithme de décodage

- Modifier l'échange de l'information extrinsèque

Quelles solutions pour l'amélioration des performances de décodage ?

① Modifier le code

② Modifier l'algorithme de décodage

- Modifier l'échange de l'information extrinsèque
- Facteur de remise à l'échelle (scaling factor) [1]

[1] Vogt et al., "Improving the Max-Log-MAP turbo decoder", Electron. Lett., 2000

Quelles solutions pour l'amélioration des performances de décodage ?

① Modifier le code

② Modifier l'algorithme de décodage

- Modifier l'échange de l'information extrinsèque
 - Facteur de remise à l'échelle (scaling factor) [1]
- Exploiter un code détecteur d'erreurs (CRC) présent dans la majorité des standards

[1] Vogt et al., "Improving the Max-Log-MAP turbo decoder", Electron. Lett., 2000

Quelles solutions pour l'amélioration des performances de décodage ?

① Modifier le code

② Modifier l'algorithme de décodage

- Modifier l'échange de l'information extrinsèque
 - Facteur de remise à l'échelle (scaling factor) [1]
- Exploiter un code détecteur d'erreurs (CRC) présent dans la majorité des standards
 - Décodage par liste [2]
 - Décodage par statistiques ordonnées [3]
 - Décodages successifs d'une même trame [4]

[1] Vogt et al., "Improving the Max-Log-MAP turbo decoder", Electron. Lett., 2000

[2] Akmalkhodzhaev et al., "New iterative turbo code list decoder", REDUNDANCY, 2014

[3] Xu et al., "An efficient OSD-aided iterative decoding algorithm for LTE turbo codes", WCSP, 2012

[4] Ould Cheikh Mouhamedou et al., "Improving the Error Rate Performance of Turbo Codes using the Forced Symbol Method", Commun. Lett., 2007

Quelles solutions pour l'amélioration des performances de décodage ?

① Modifier le code

② Modifier l'algorithme de décodage

- Modifier l'échange de l'information extrinsèque
 - Facteur de remise à l'échelle (scaling factor) [1]
- Exploiter un code détecteur d'erreurs (CRC) présent dans la majorité des standards
 - Décodage par liste [2]
 - Décodage par statistiques ordonnées [3]
 - Décodages successifs d'une même trame [4]
- Propositions :

[1] Vogt et al., "Improving the Max-Log-MAP turbo decoder", Electron. Lett., 2000

[2] Akmalkhodzhaev et al., "New iterative turbo code list decoder", REDUNDANCY, 2014

[3] Xu et al., "An efficient OSD-aided iterative decoding algorithm for LTE turbo codes", WCSP, 2012

[4] Ould Cheikh Mouhamedou et al., "Improving the Error Rate Performance of Turbo Codes using the Forced Symbol Method", Commun. Lett., 2007

Quelles solutions pour l'amélioration des performances de décodage ?

① Modifier le code

② Modifier l'algorithme de décodage

- Modifier l'échange de l'information extrinsèque
 - Facteur de remise à l'échelle (scaling factor) [1]
- Exploiter un code détecteur d'erreurs (CRC) présent dans la majorité des standards
 - Décodage par liste [2]
 - Décodage par statistiques ordonnées [3]
 - Décodages successifs d'une même trame [4]
- Propositions :
 - Self-Corrected [Chapitre II du manuscrit]

[1] Vogt et al., "Improving the Max-Log-MAP turbo decoder", Electron. Lett., 2000

[2] Akmalkhodzhaev et al., "New iterative turbo code list decoder", REDUNDANCY, 2014

[3] Xu et al., "An efficient OSD-aided iterative decoding algorithm for LTE turbo codes", WCSP, 2012

[4] Ould Cheikh Mouhamedou et al., "Improving the Error Rate Performance of Turbo Codes using the Forced Symbol Method", Commun. Lett., 2007

Quelles solutions pour l'amélioration des performances de décodage ?

① Modifier le code

② Modifier l'algorithme de décodage

- Modifier l'échange de l'information extrinsèque
 - Facteur de remise à l'échelle (scaling factor) [1]
- Exploiter un code détecteur d'erreurs (CRC) présent dans la majorité des standards
 - Décodage par liste [2]
 - Décodage par statistiques ordonnées [3]
 - Décodages successifs d'une même trame [4]
- Propositions :
 - Self-Corrected [Chapitre II du manuscrit]
 - Flip and Check (FNC) [Chapitre III du manuscrit]

[1] Vogt et al., "Improving the Max-Log-MAP turbo decoder", Electron. Lett., 2000

[2] Akmalkhodzhaev et al., "New iterative turbo code list decoder", REDUNDANCY, 2014

[3] Xu et al., "An efficient OSD-aided iterative decoding algorithm for LTE turbo codes", WCSP, 2012

[4] Ould Cheikh Mouhamedou et al., "Improving the Error Rate Performance of Turbo Codes using the Forced Symbol Method", Commun. Lett., 2007

Plan

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

Observations dans la zone du plancher d'erreurs

Identification des positions des bits erronés

L'algorithme Flip and Check

Extension aux turbo codes doubles binaires

③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

Plan

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

Observations dans la zone du plancher d'erreurs

Identification des positions des bits erronés

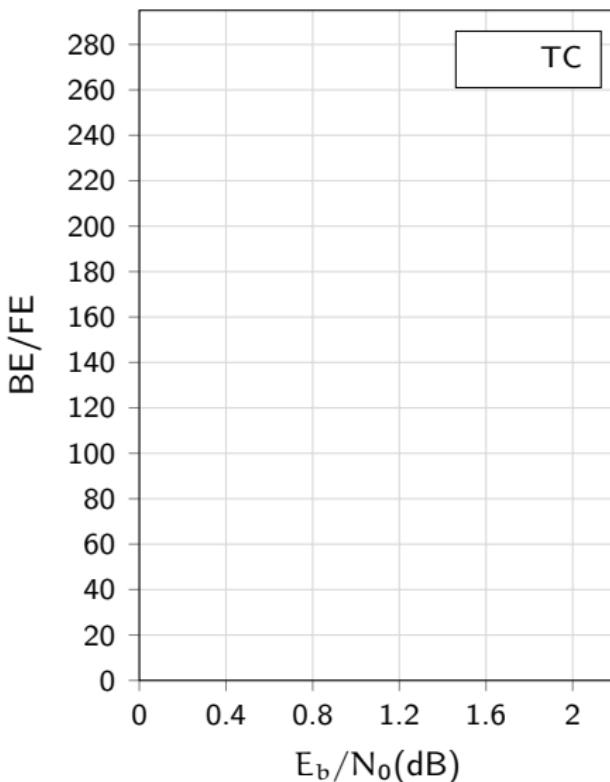
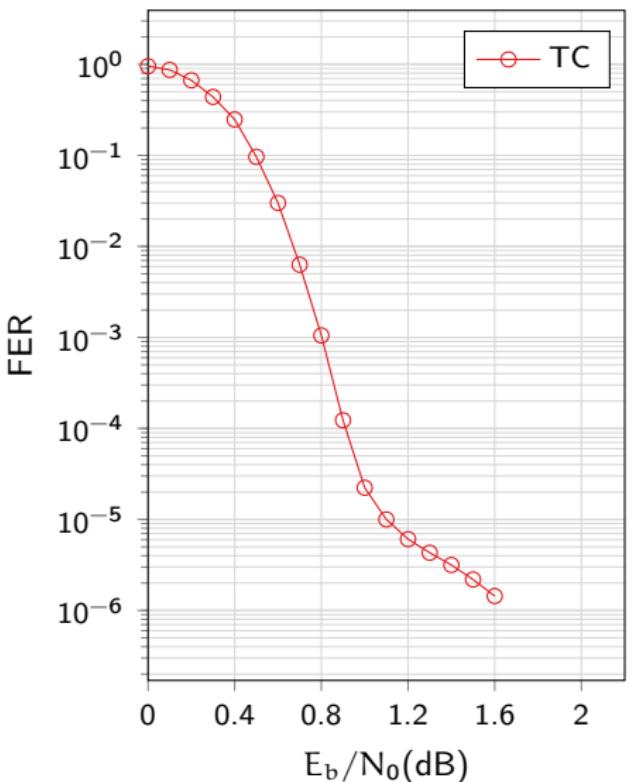
L'algorithme Flip and Check

Extension aux turbo codes doubles binaires

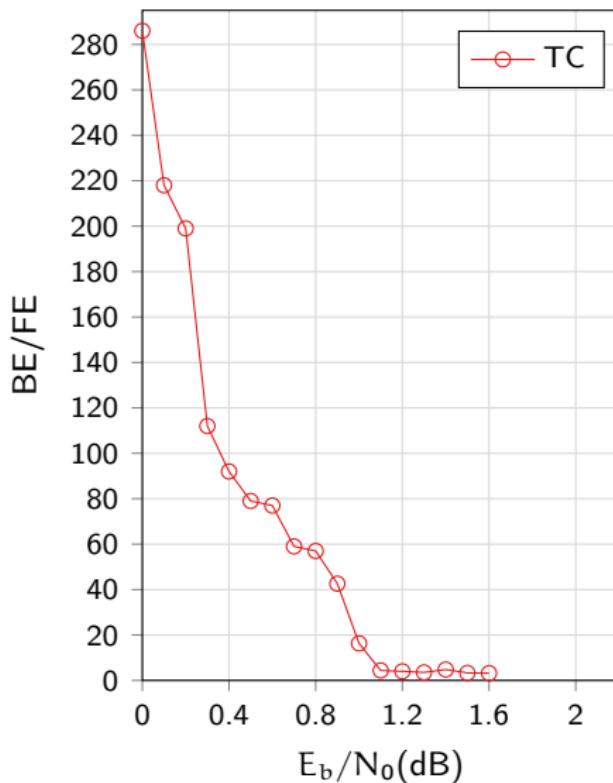
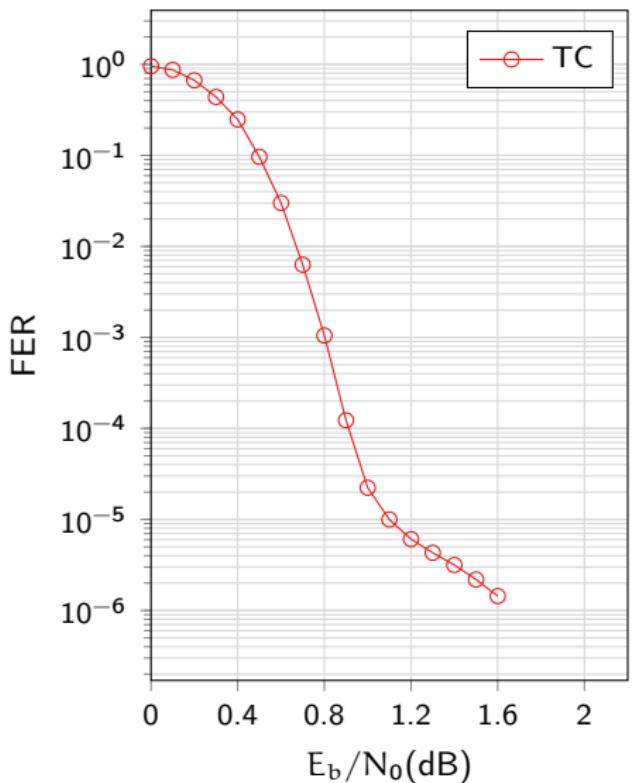
③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

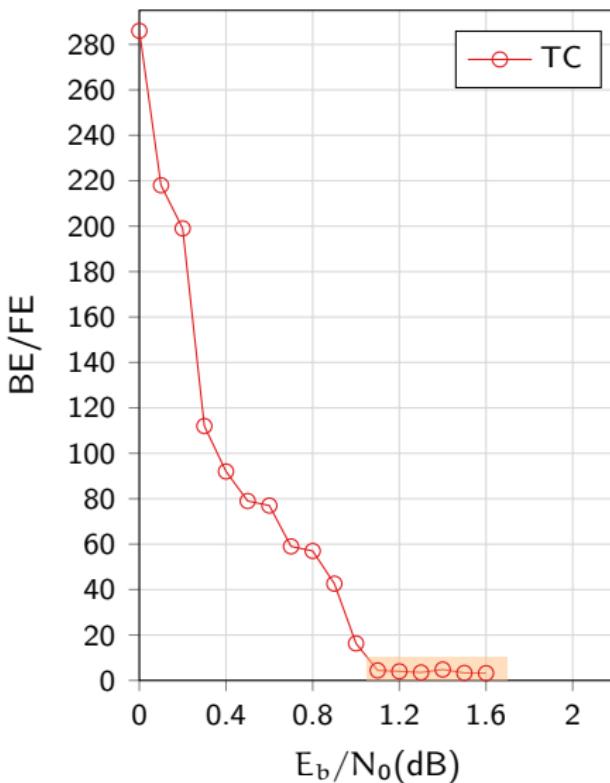
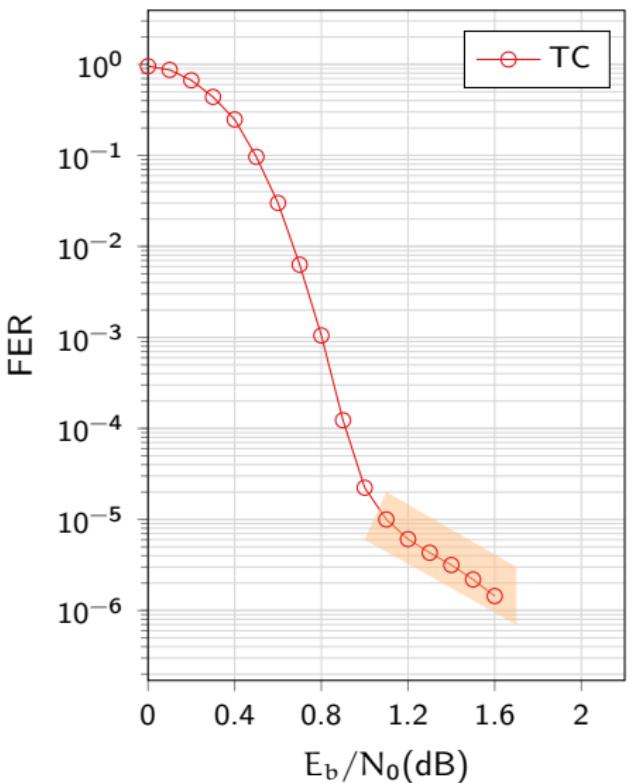
Exemple standard LTE, K=2048, R=1/3 - Nombre moyen de bits erronés par trame



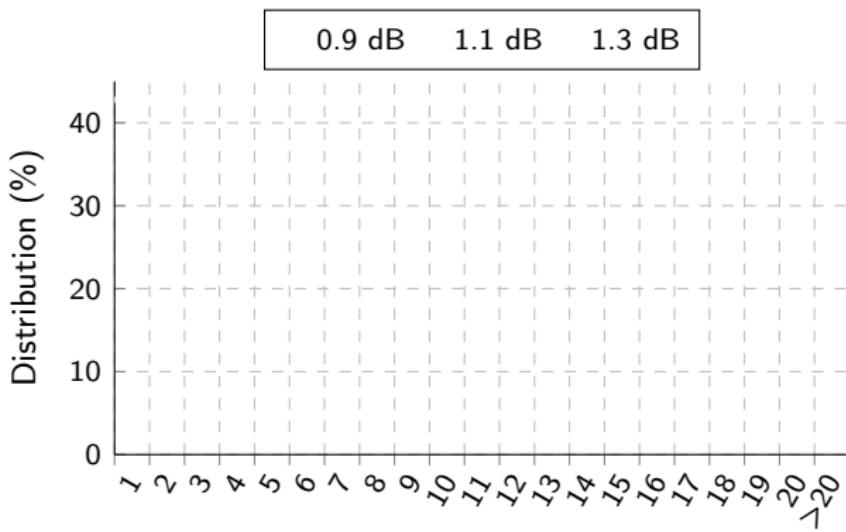
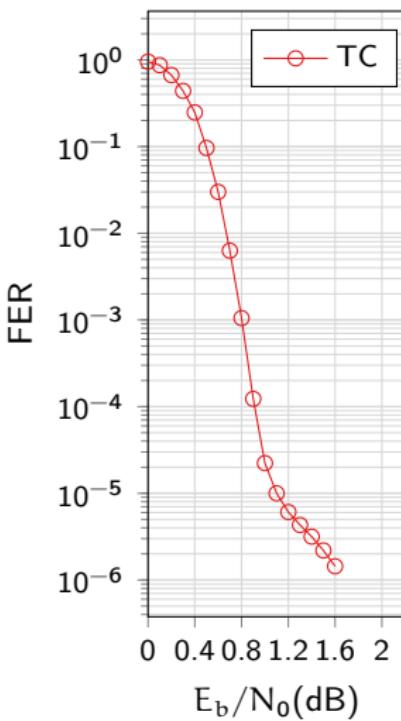
Exemple standard LTE, K=2048, R=1/3 - Nombre moyen de bits erronés par trame



Exemple standard LTE, K=2048, R=1/3 - Nombre moyen de bits erronés par trame

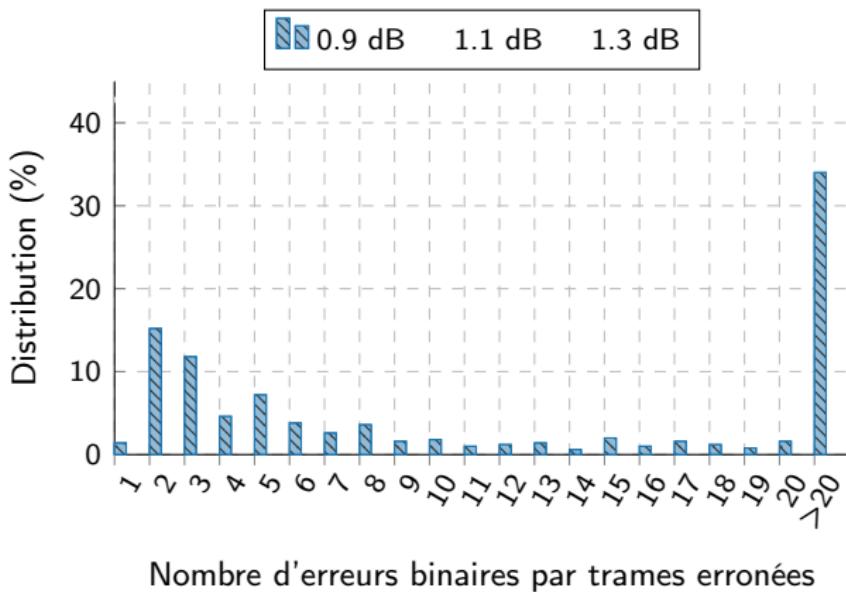
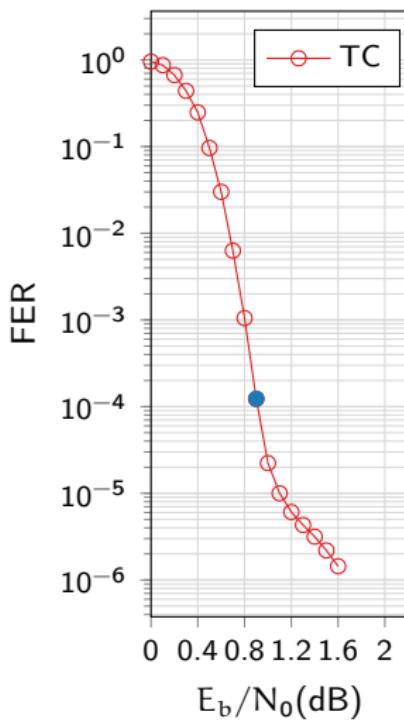


Exemple standard LTE, K=2048, R=1/3 - Distribution des erreurs binaires



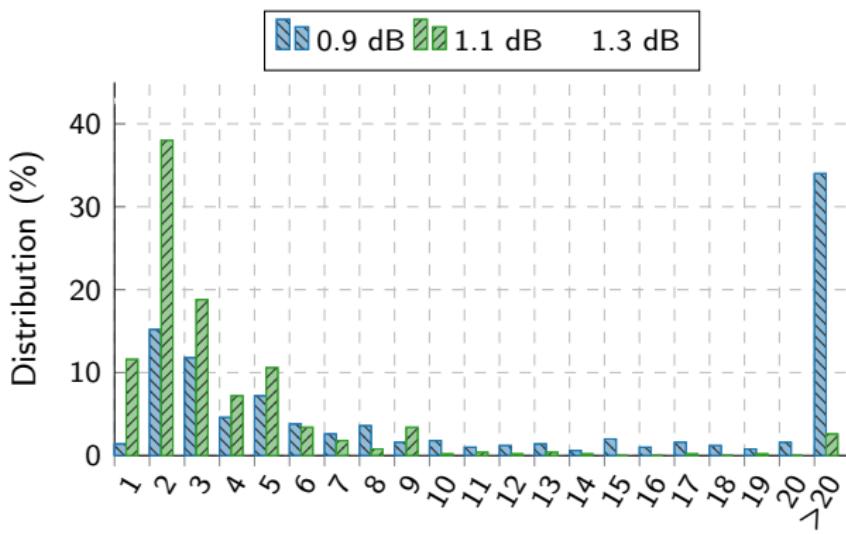
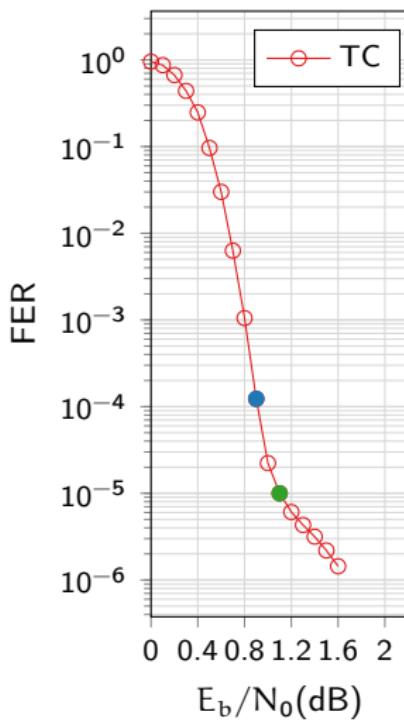
Nombre d'erreurs binaires par trames erronées

Exemple standard LTE, K=2048, R=1/3 - Distribution des erreurs binaires



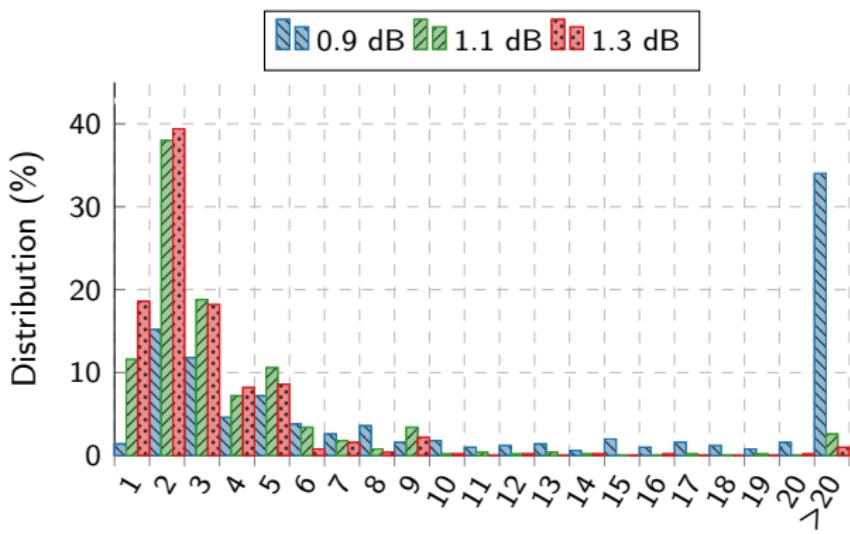
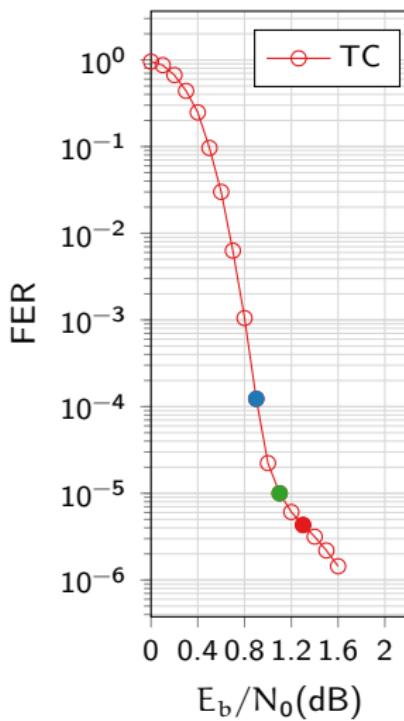
Nombre d'erreurs binaires par trames erronées

Exemple standard LTE, K=2048, R=1/3 - Distribution des erreurs binaires



Nombre d'erreurs binaires par trames erronées

Exemple standard LTE, K=2048, R=1/3 - Distribution des erreurs binaires



Nombre d'erreurs binaires par trames erronées

Plan

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

Observations dans la zone du plancher d'erreurs

Identification des positions des bits erronés

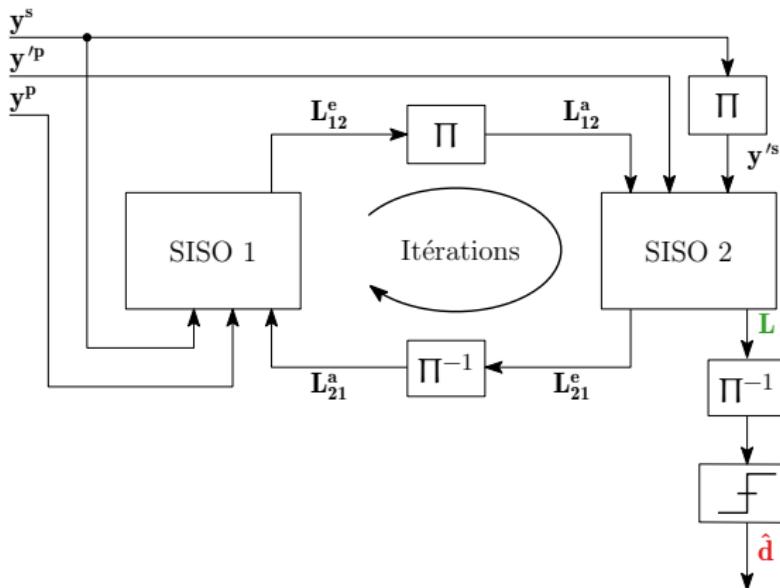
L'algorithme Flip and Check

Extension aux turbo codes doubles binaires

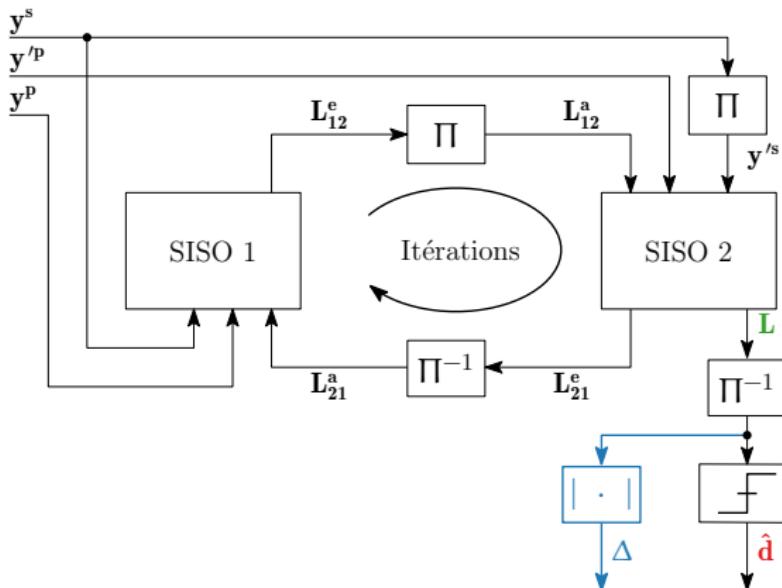
③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

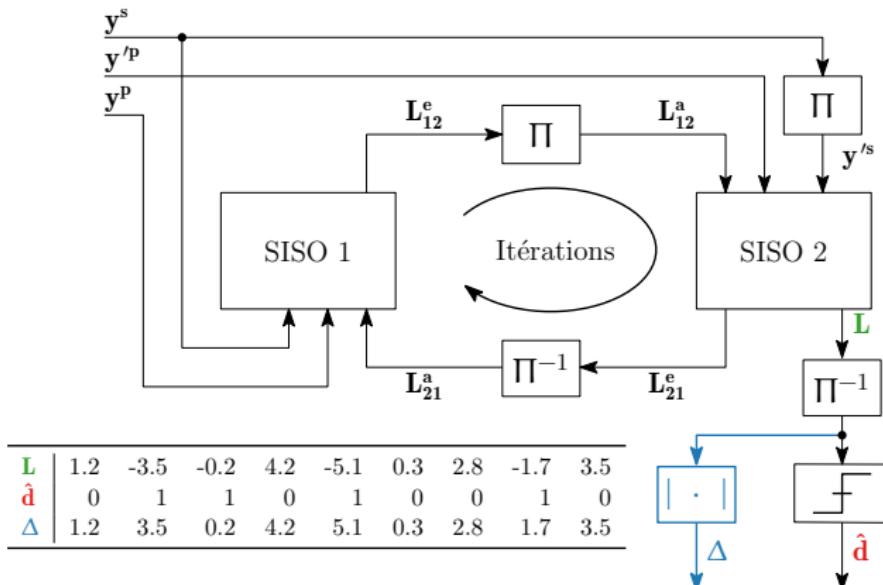
Le critère d'identification retenu : construction et exemple



Le critère d'identification retenu : construction et exemple

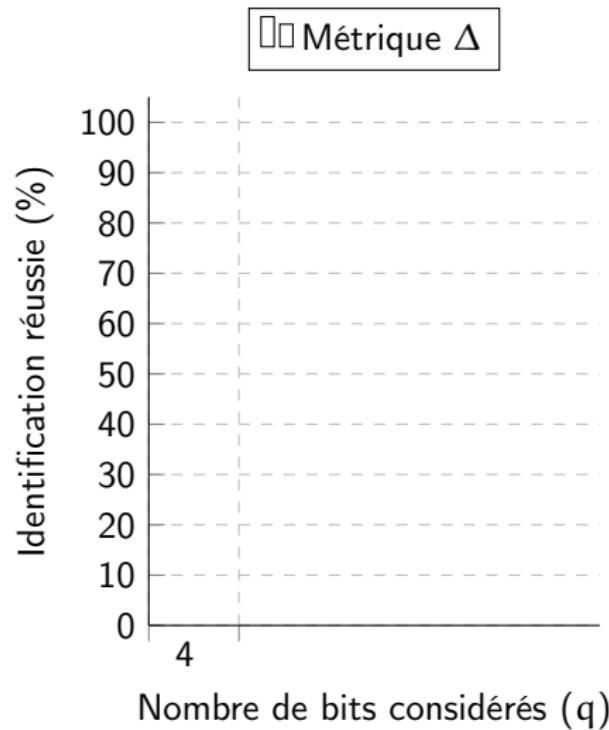
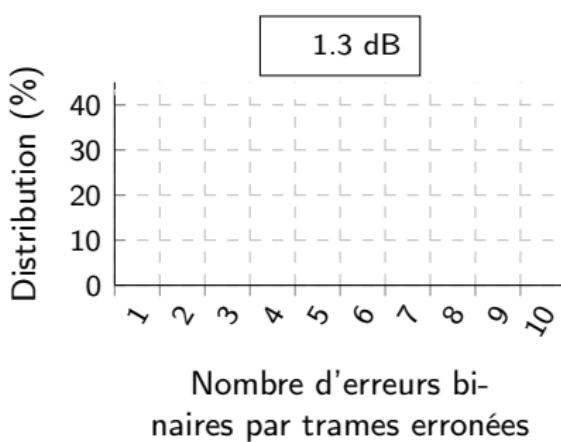


Le critère d'identification retenu : construction et exemple

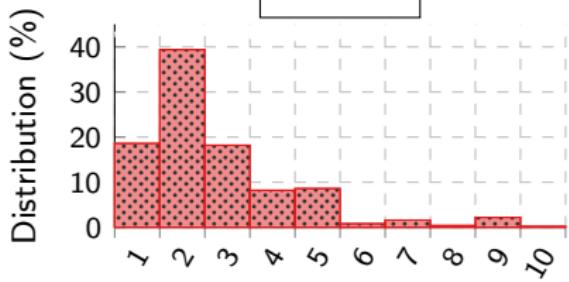


- $\Delta(k) = |L(k)|$, avec $k \in [0; K]$
- Module de l'information *a posteriori* → niveau de confiance associé à chaque bit du mot décodé
- Corrélation forte entre une information *a posteriori* faible et une erreur de décodage

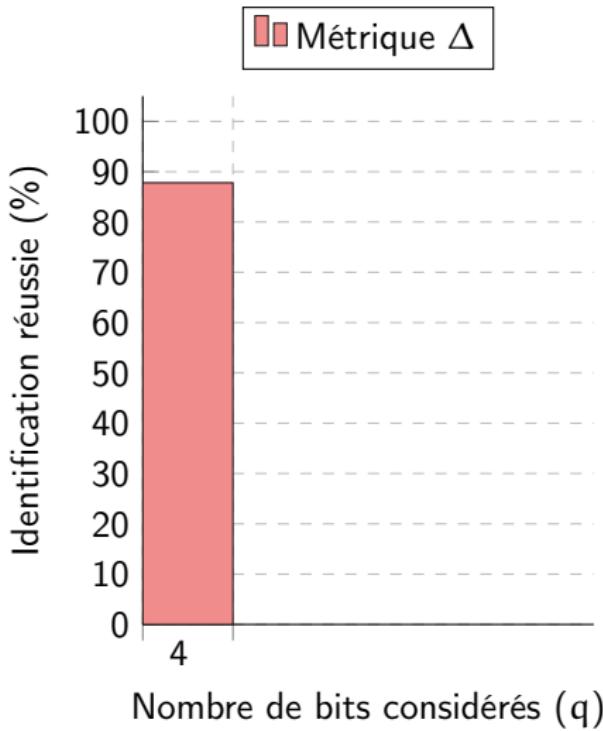
Résultats d'identification - Standard LTE, K=2048, R=1/3, 8its, flottant



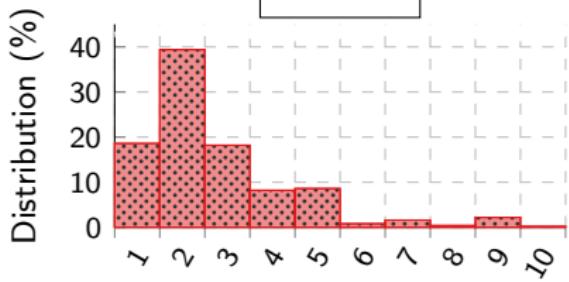
Résultats d'identification - Standard LTE, K=2048, R=1/3, 8its, flottant



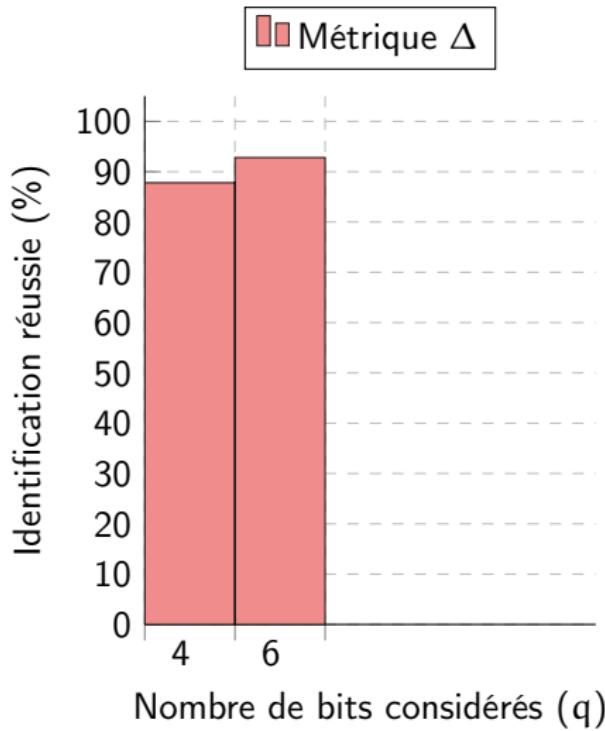
Nombre d'erreurs binaires par trames erronées



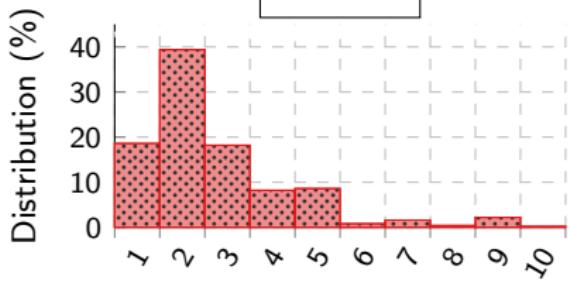
Résultats d'identification - Standard LTE, K=2048, R=1/3, 8its, flottant



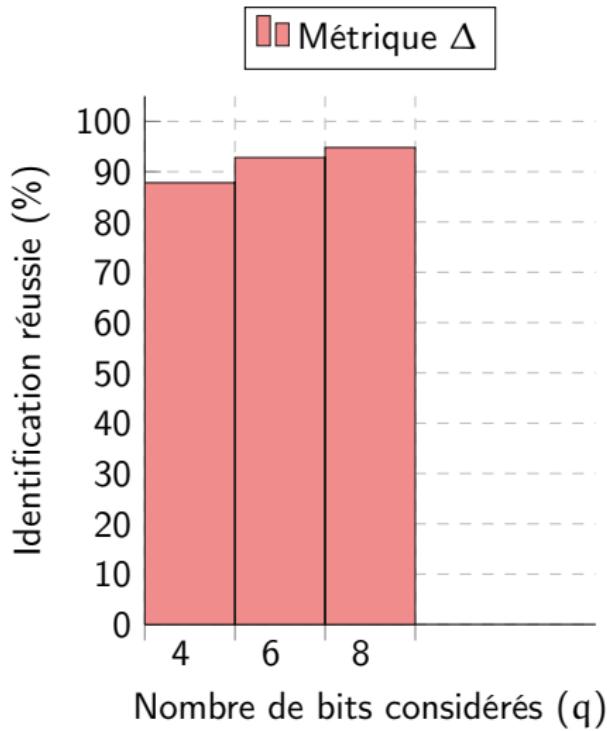
Nombre d'erreurs binaires par trames erronées



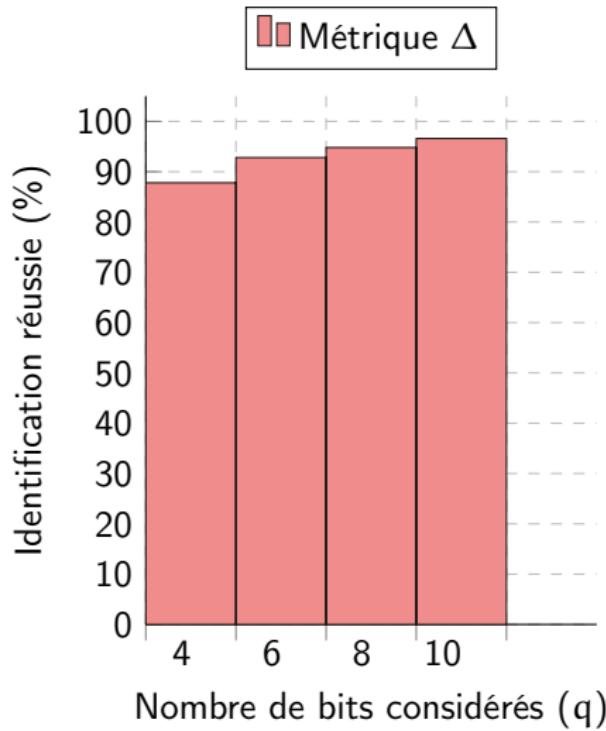
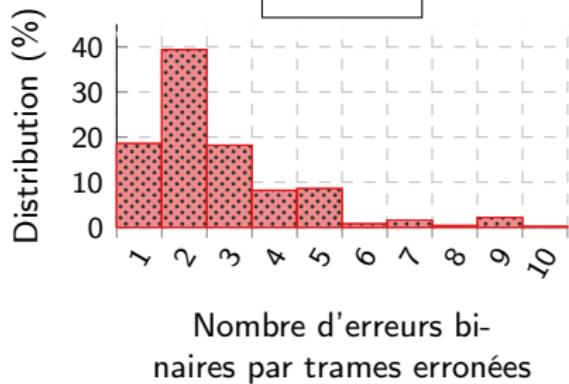
Résultats d'identification - Standard LTE, K=2048, R=1/3, 8its, flottant



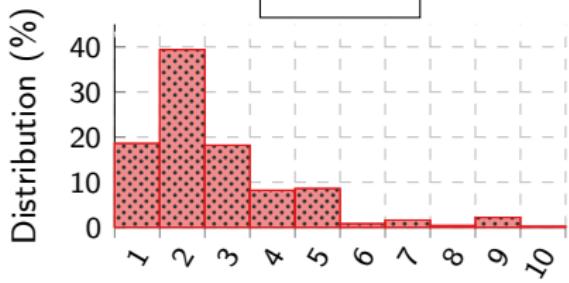
Nombre d'erreurs binaires par trames erronées



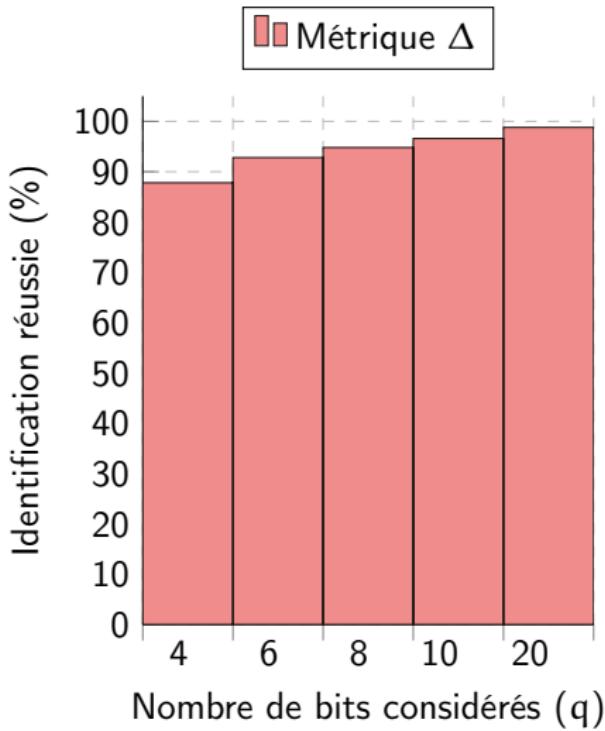
Résultats d'identification - Standard LTE, K=2048, R=1/3, 8its, flottant



Résultats d'identification - Standard LTE, K=2048, R=1/3, 8its, flottant



Nombre d'erreurs binaires par trames erronées



Plan

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

Observations dans la zone du plancher d'erreurs

Identification des positions des bits erronés

L'algorithme Flip and Check

Extension aux turbo codes doubles binaires

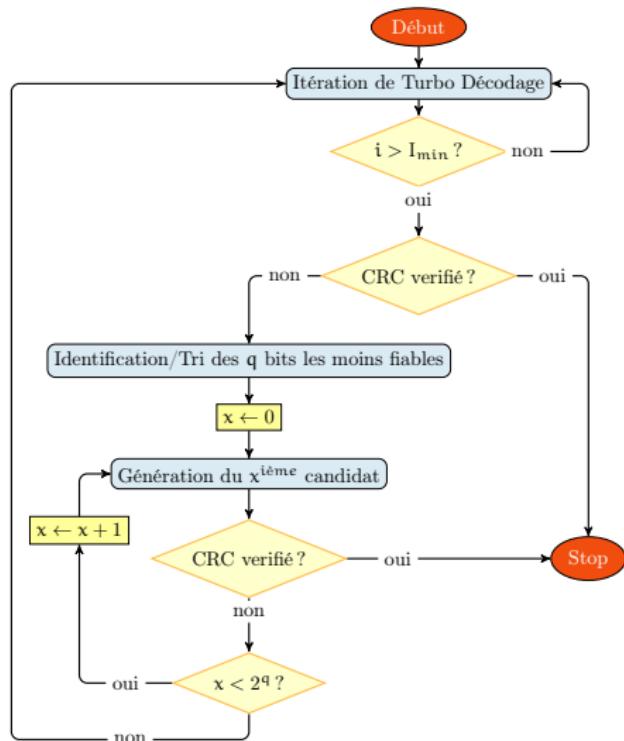
③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

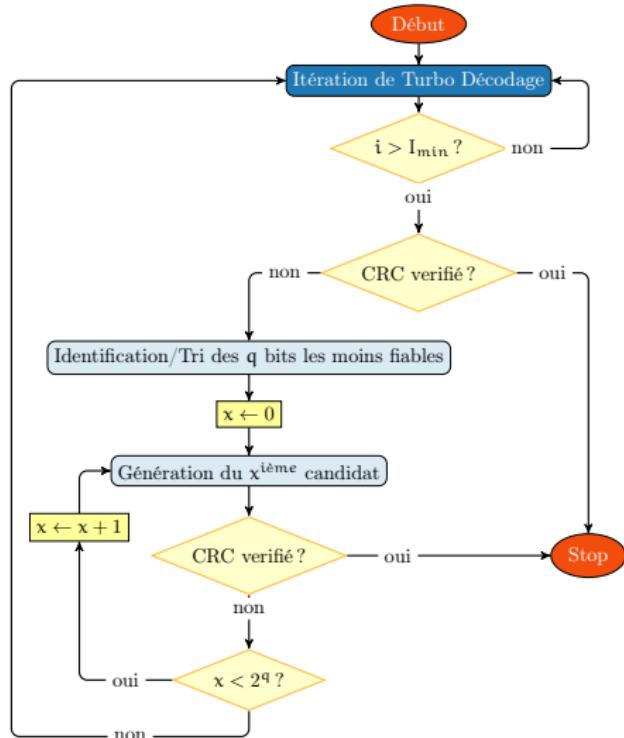
Principe

- ① Calcul de la fiabilité de chacun des bits (Δ)
- ② Identification des q positions des bits les moins fiables
- ③ Génération de $2^q - 1$ mots candidats
- ④ Vérification par le code CRC des mots candidats

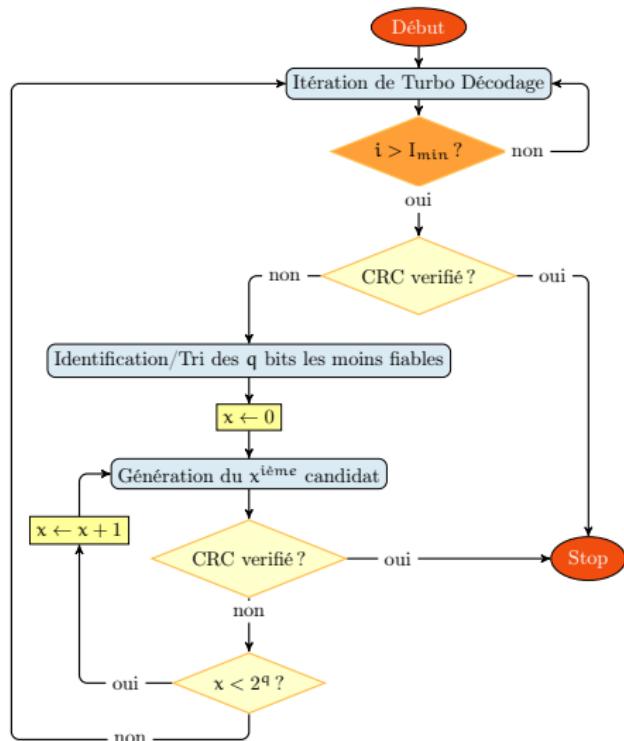
Explication schématique et exemple



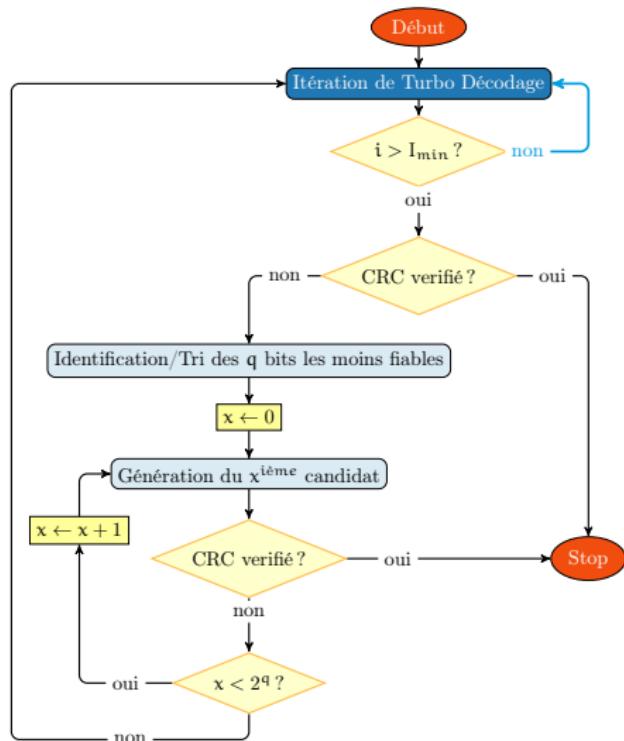
Explication schématique et exemple



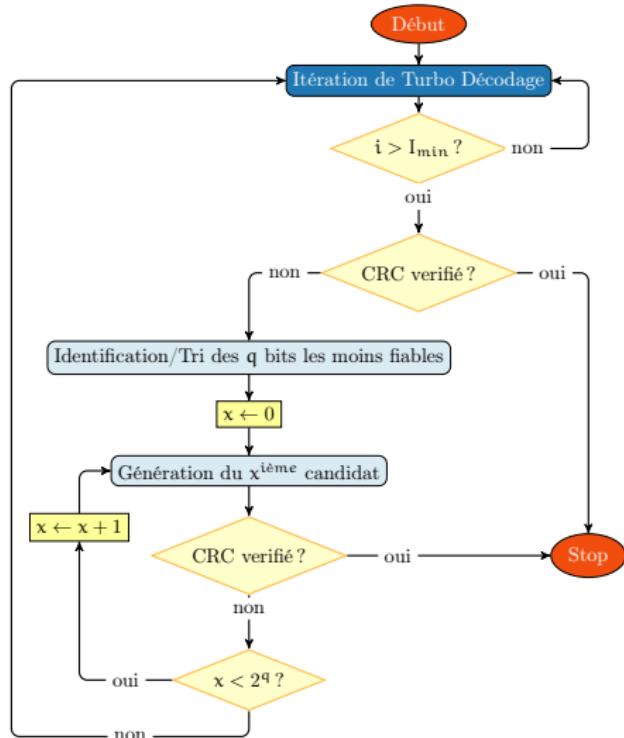
Explication schématique et exemple



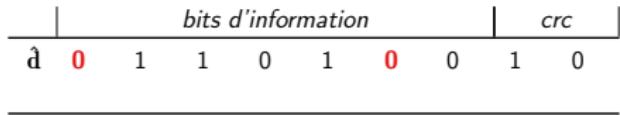
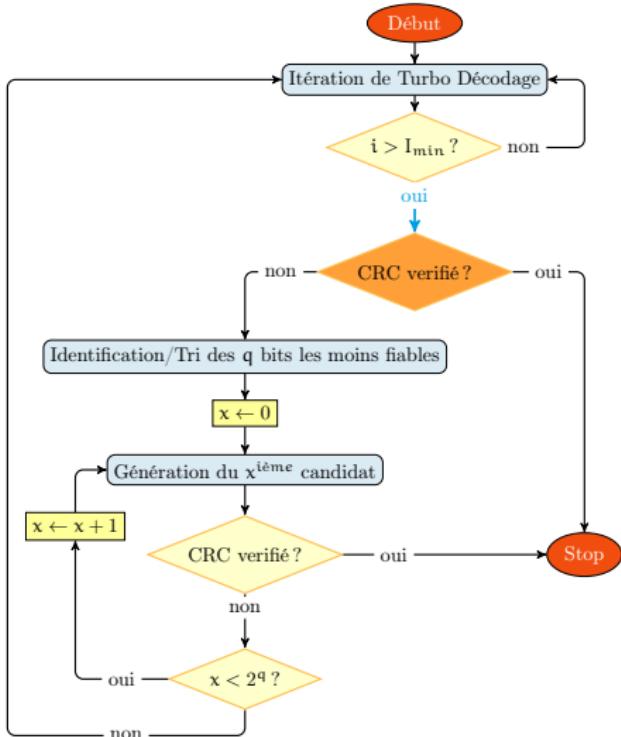
Explication schématique et exemple



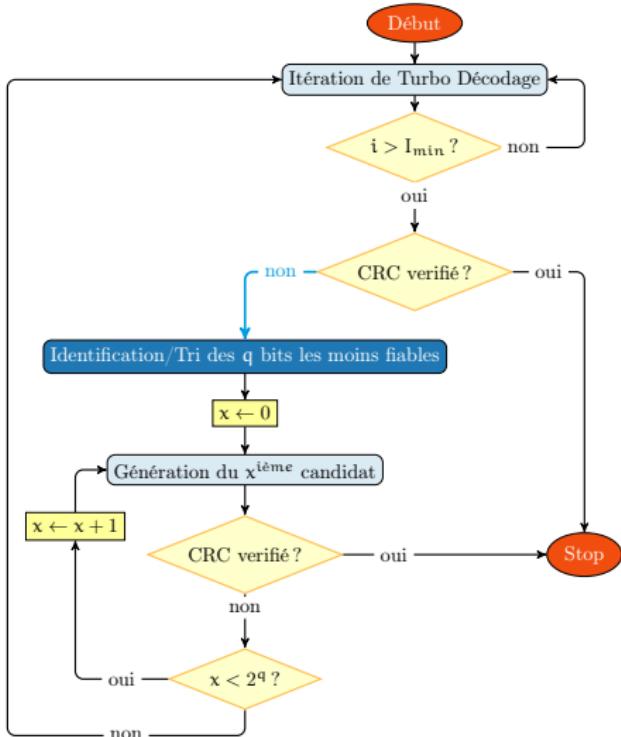
Explication schématique et exemple



Explication schématique et exemple

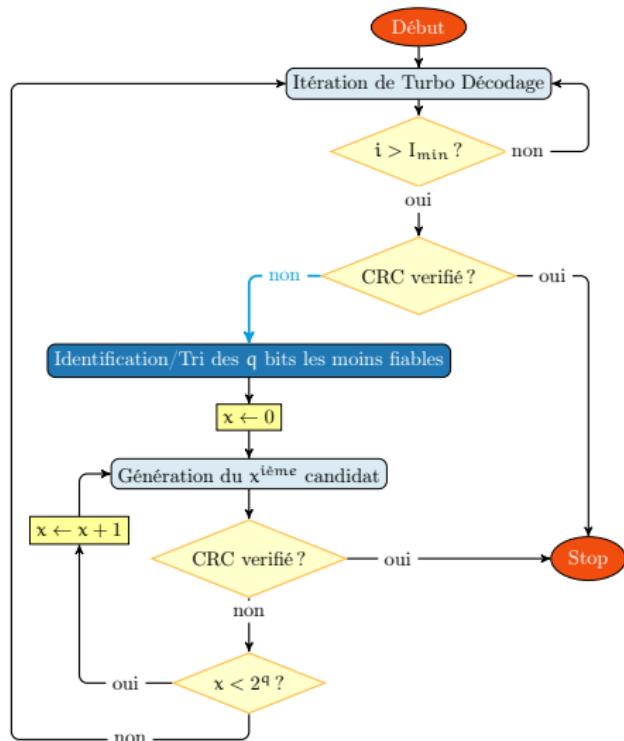


Explication schématique et exemple



	bits d'information								crc
\hat{d}	0	1	1	0	1	0	0	1	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8	1.7	3.5

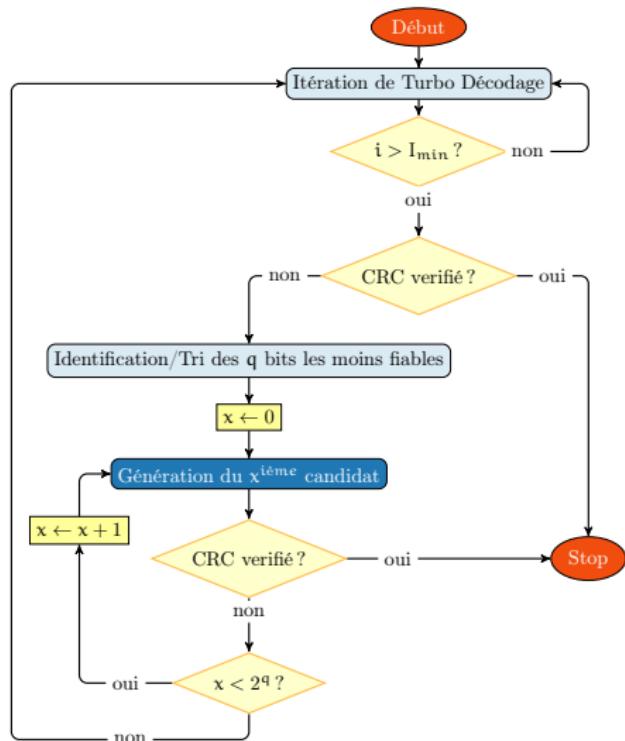
Explication schématique et exemple



bits d'information							crc
\hat{d}	0	1	1	0	1	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

Explication schématique et exemple



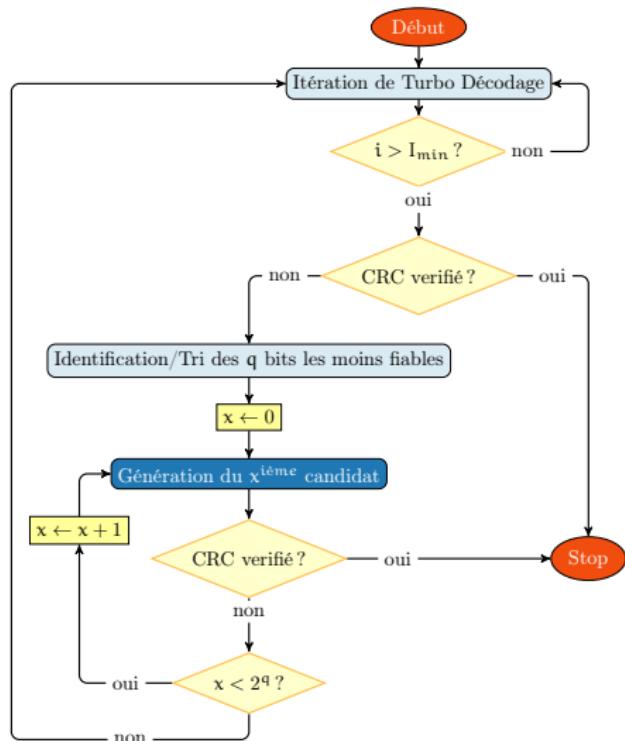
bits d'information							crc
d	0	1	1	0	1	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d ₁	0	1	0	0	1	0	0	1	0
<hr/>									

Explication schématique et exemple



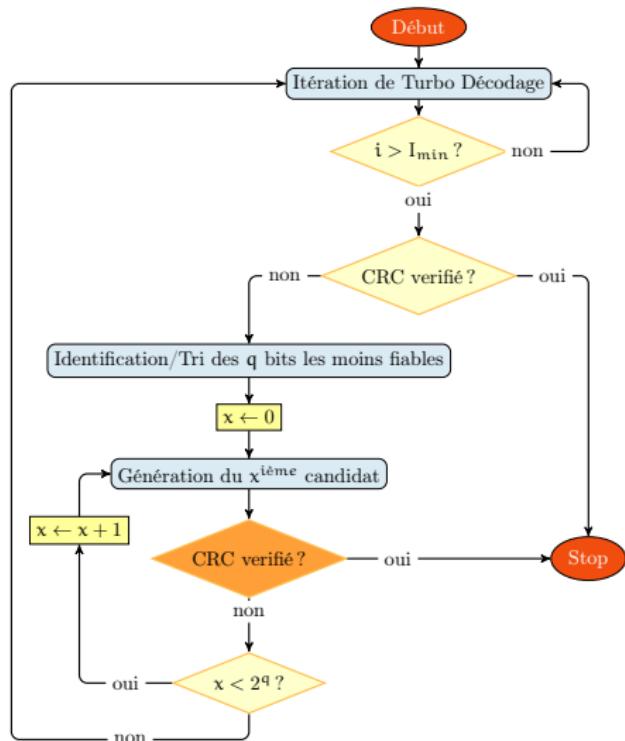
bits d'information							crc
d	0	1	1	0	1	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d ₁	0	1	0	0	1	0	0	1	0
<hr/>									

Explication schématique et exemple



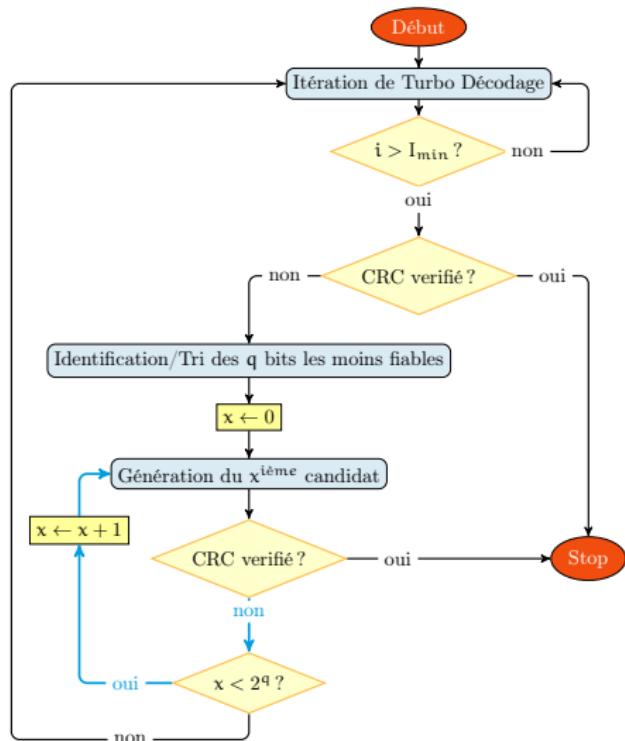
bits d'information							crc
d	0	1	1	0	1	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d_1	0	1	0	0	1	0	0	1	0
-------	---	---	---	---	---	---	---	---	---

Explication schématique et exemple



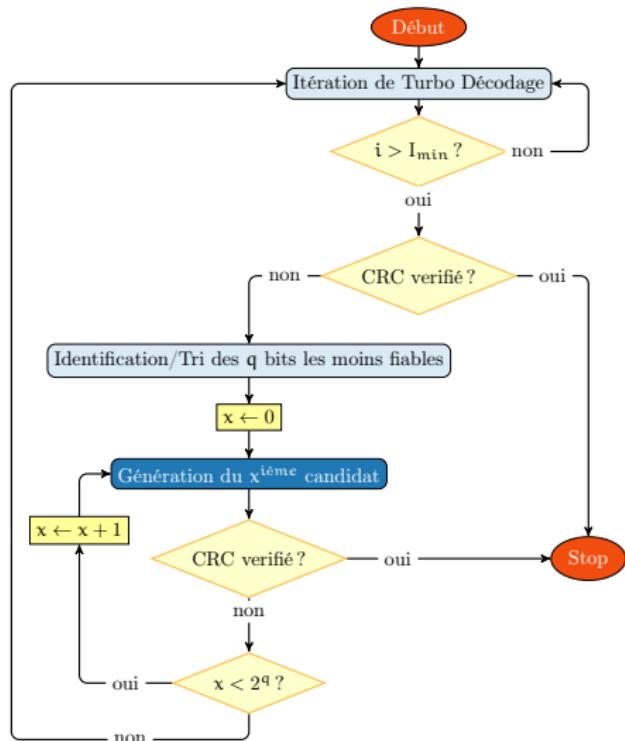
bits d'information							crc
d	0	1	1	0	1	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d ₁	0	1	0	0	1	0	1	0
<hr/>								

Explication schématique et exemple



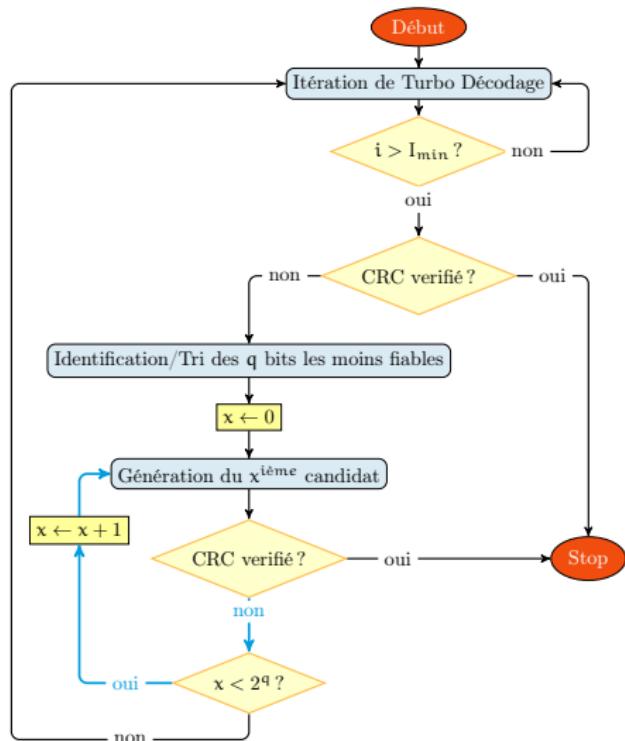
	bits d'information							<i>crc</i>	
\hat{d}	0	1	1	0	1	0	0	1	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8	1.7	3.5

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d_1	0	1	0	0	1	0	0	1	0
d_2	0	1	1	0	1	1	0	1	0

Explication schématique et exemple



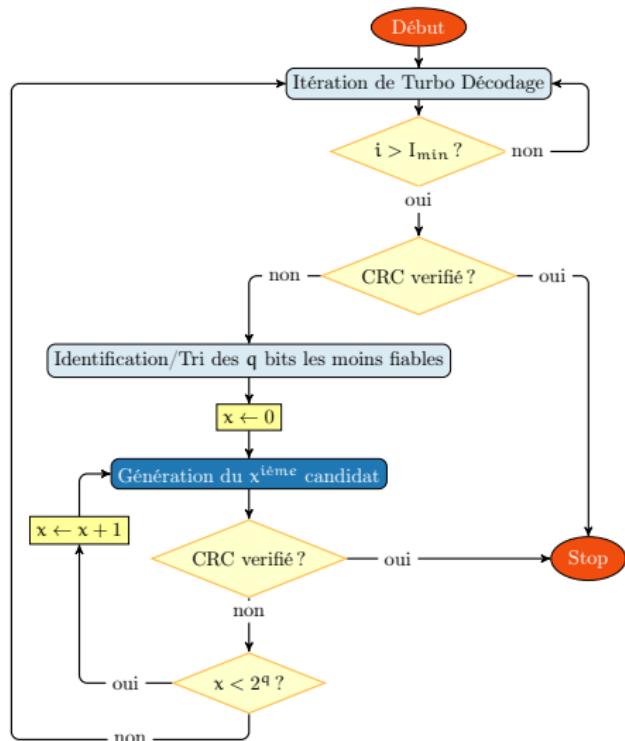
	bits d'information							<i>crc</i>	
\hat{d}	0	1	1	0	1	0	0	1	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8	1.7	3.5

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d_1	0	1	0	0	1	0	0	1	0
d_2	0	1	1	0	1	1	0	1	0

Explication schématique et exemple



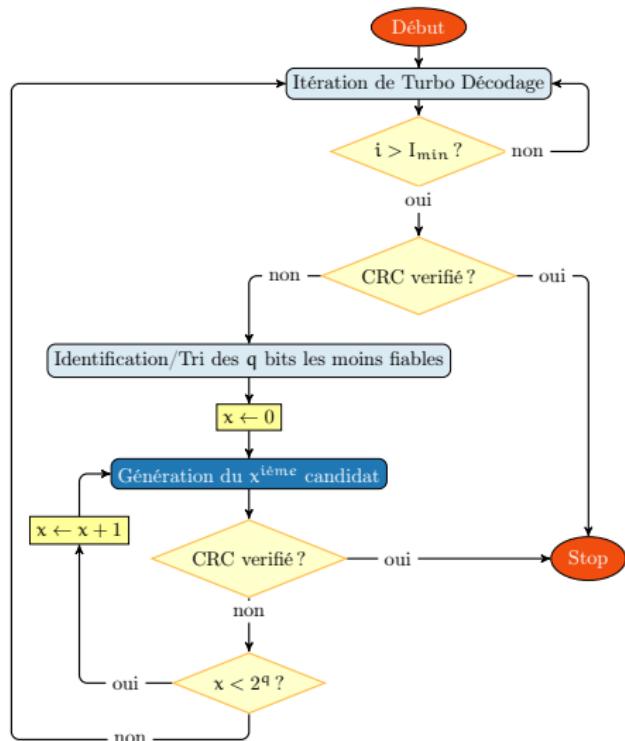
	bits d'information							<i>crc</i>
\hat{d}	0	1	1	0	1	0	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8	1.7

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d_1	0	1	0	0	1	0	0	1	0
d_2	0	1	1	0	1	1	0	1	0
d_3	1	1	1	0	1	0	0	1	0

Explication schématique et exemple



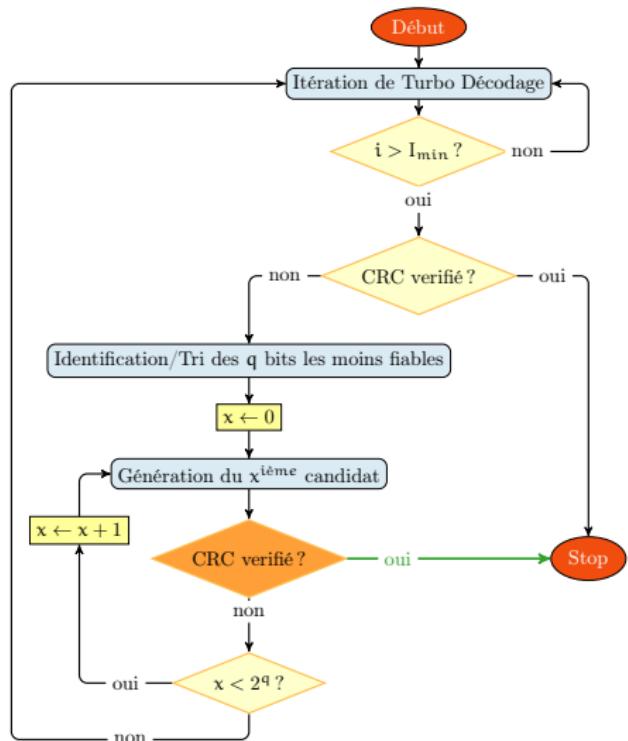
	bits d'information							<i>crc</i>
\hat{d}	0	1	1	0	1	0	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8	1.7

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d_1	0	1	0	0	1	0	0	1	0
d_2	0	1	1	0	1	1	0	1	0
d_3	1	1	1	0	1	0	0	1	0
d_4	0	1	0	0	1	1	0	1	0
d_5	1	1	0	0	1	0	0	1	0
d_6	1	1	1	0	1	1	0	1	0
d_7									

Explication schématique et exemple



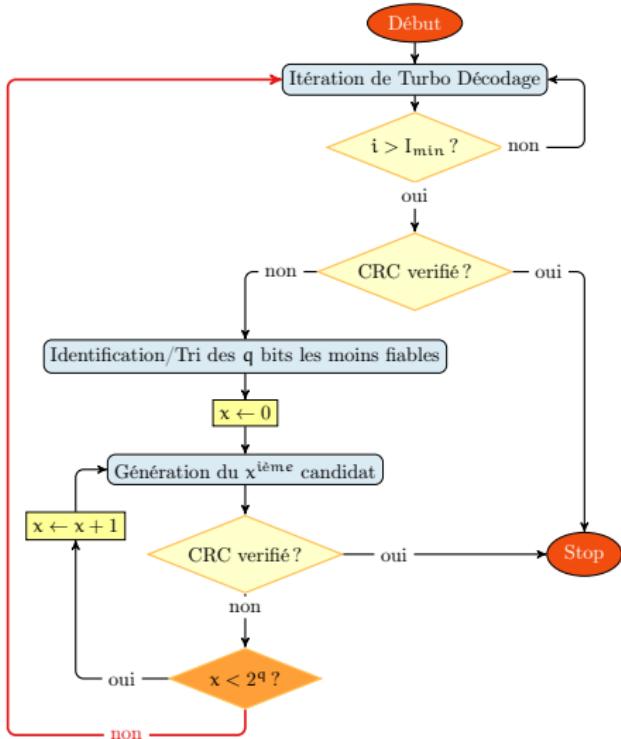
	bits d'information							<i>crc</i>
\hat{d}	0	1	1	0	1	0	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8	1.7

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

$2^q - 1 = 7$ mots candidats générés :

d_1	0	1	0	0	1	0	0	1	0
d_2	0	1	1	0	1	1	0	1	0
d_3	1	1	1	0	1	0	0	1	0
d_4	0	1	0	0	1	1	0	1	0
d_5	1	1	0	0	1	0	0	1	0
d_6	1	1	0	1	1	1	0	1	0
d_7									

Explication schématique et exemple



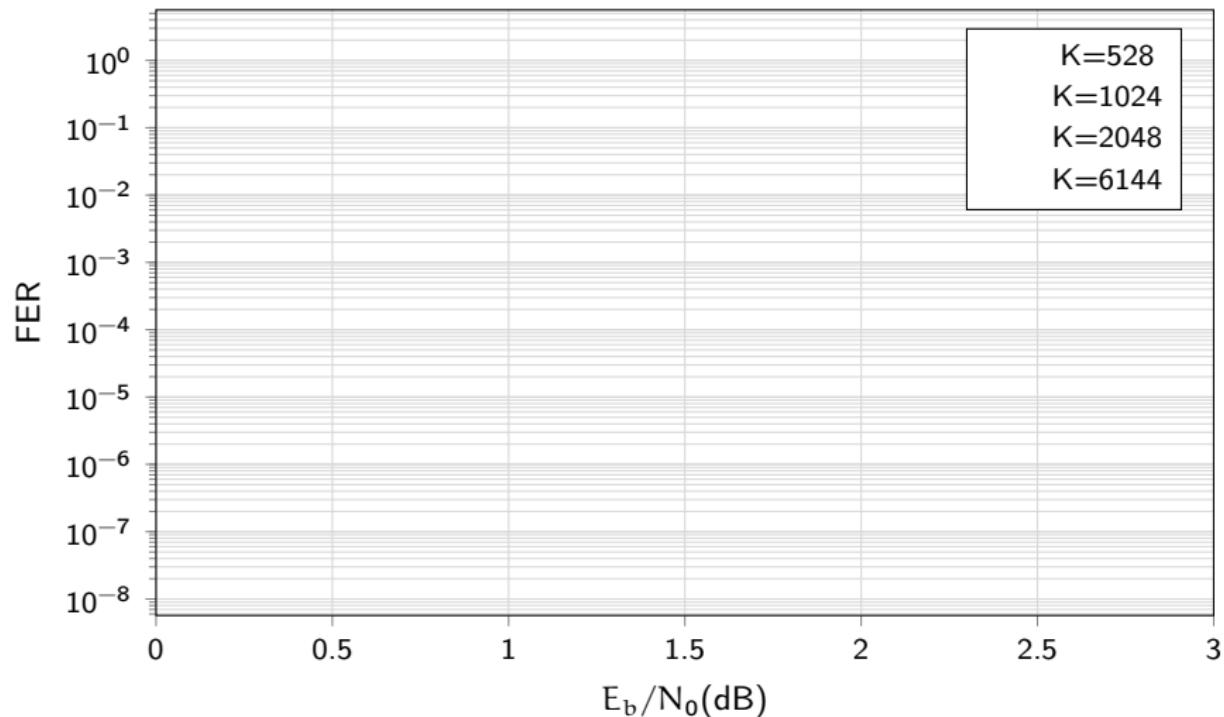
	bits d'information							<i>crc</i>
\hat{d}	0	1	1	0	1	0	0	0
Δ	1.2	3.5	0.2	4.2	5.1	0.3	2.8	1.7

$q = 3$ positions identifiées : $k_3 \quad k_6 \quad k_1$

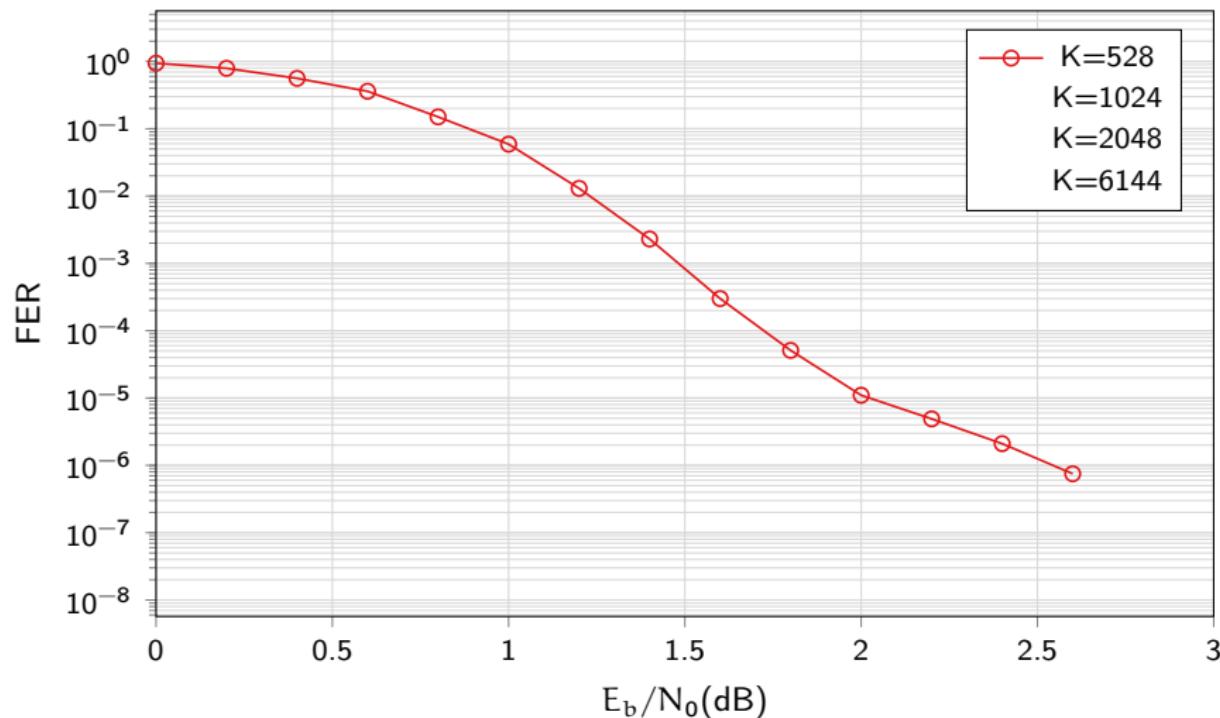
$2^q - 1 = 7$ mots candidats générés :

d_1	0	1	0	0	1	0	0	1	0
d_2	0	1	1	0	1	1	0	1	0
d_3	1	1	1	0	1	0	0	1	0
d_4	0	1	0	0	1	1	0	1	0
d_5	1	1	0	0	1	0	0	1	0
d_6	1	1	0	1	1	0	1	0	0
d_7									

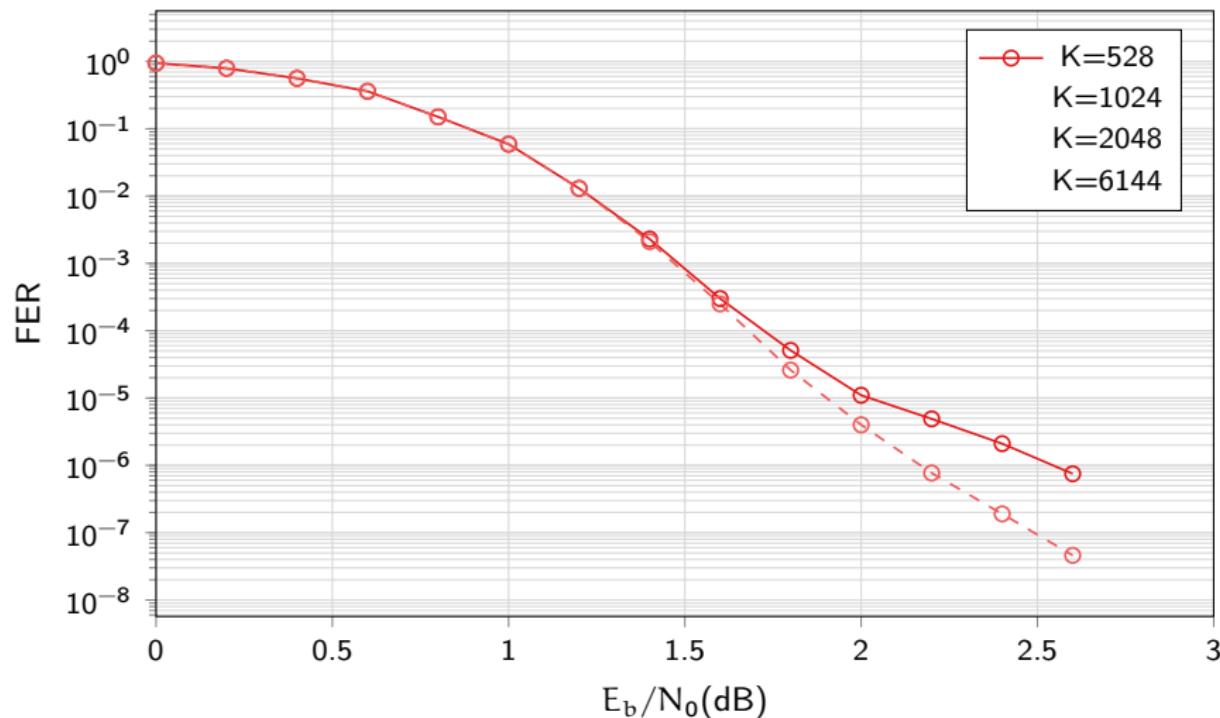
Performances de décodage - LTE, R=1/3, Canal AWGN, flottant, 8 its - q=10



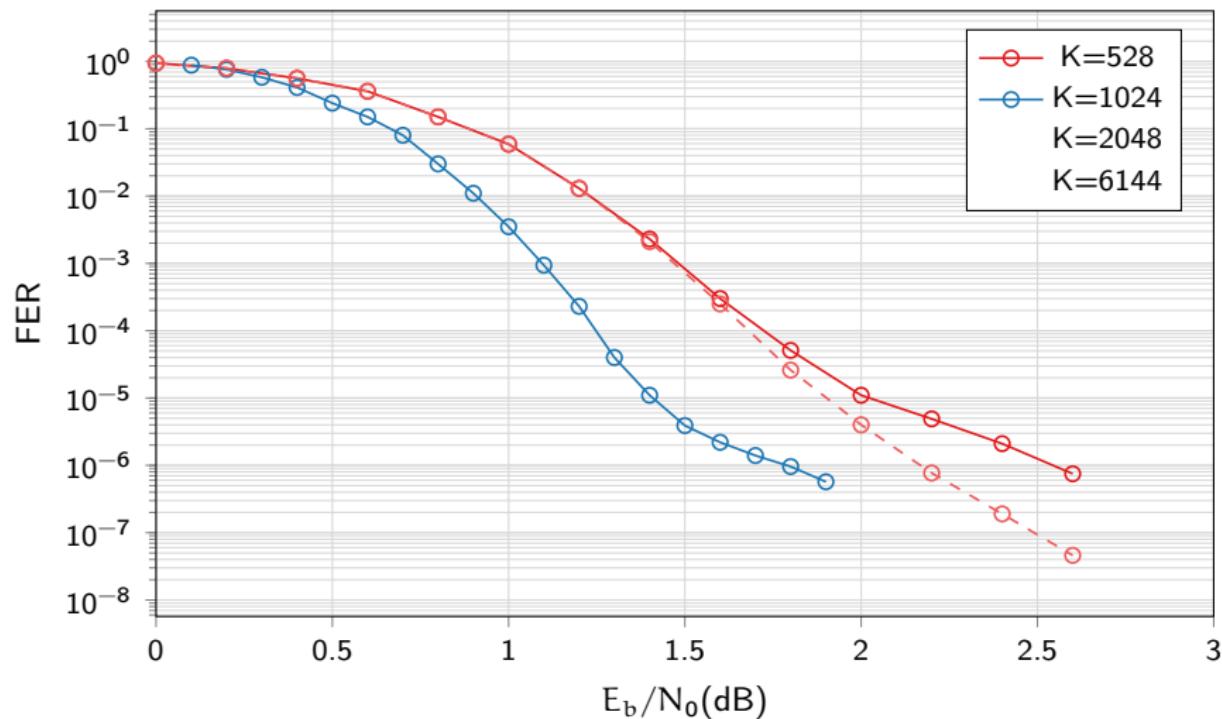
Performances de décodage - LTE, R=1/3, Canal AWGN, flottant, 8 its - q=10



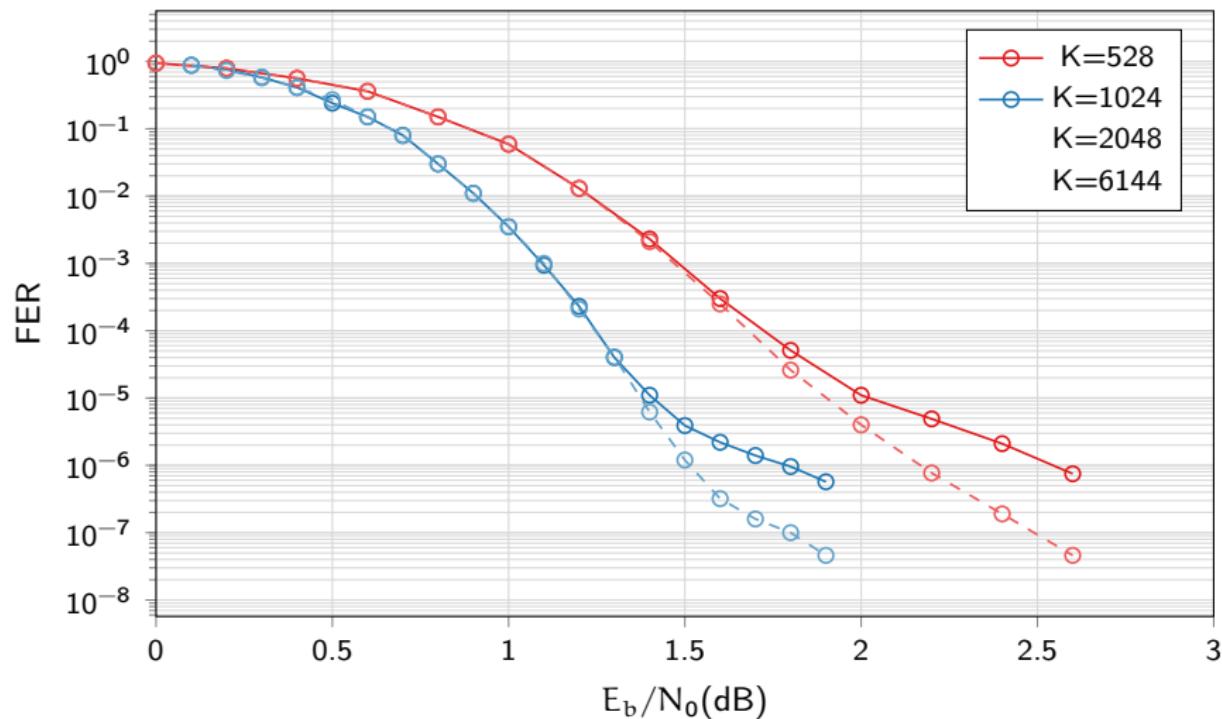
Performances de décodage - LTE, R=1/3, Canal AWGN, flottant, 8 its - q=10



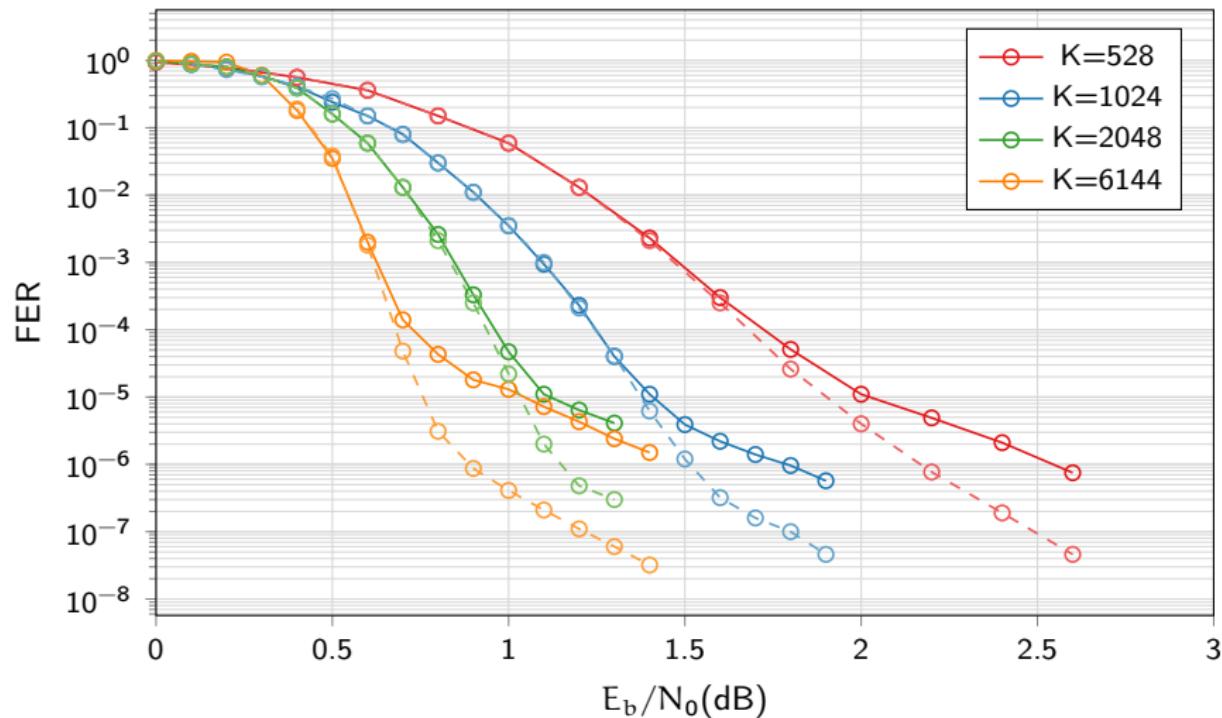
Performances de décodage - LTE, R=1/3, Canal AWGN, flottant, 8 its - q=10



Performances de décodage - LTE, R=1/3, Canal AWGN, flottant, 8 its - q=10



Performances de décodage - LTE, R=1/3, Canal AWGN, flottant, 8 its - q=10



Plan

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

Observations dans la zone du plancher d'erreurs

Identification des positions des bits erronés

L'algorithme Flip and Check

Extension aux turbo codes doubles binaires

③ Architecture matérielle de correction des erreurs résiduelles

④ Conclusion et perspectives

Introduction

Les turbo codes double binaires

- Codage par symbole (couple de 2 bits)
- Décodage par symbole

Introduction

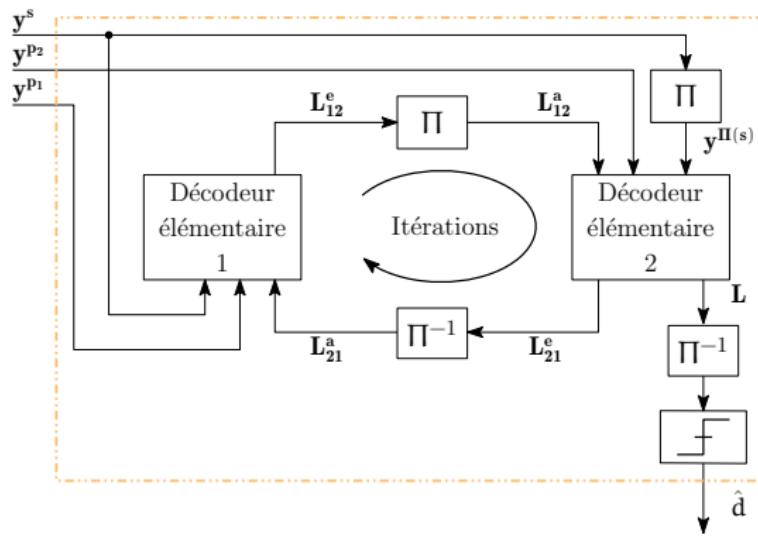
Les turbo codes double binaires

- Codage par symbole (couple de 2 bits)
- Décodage par symbole

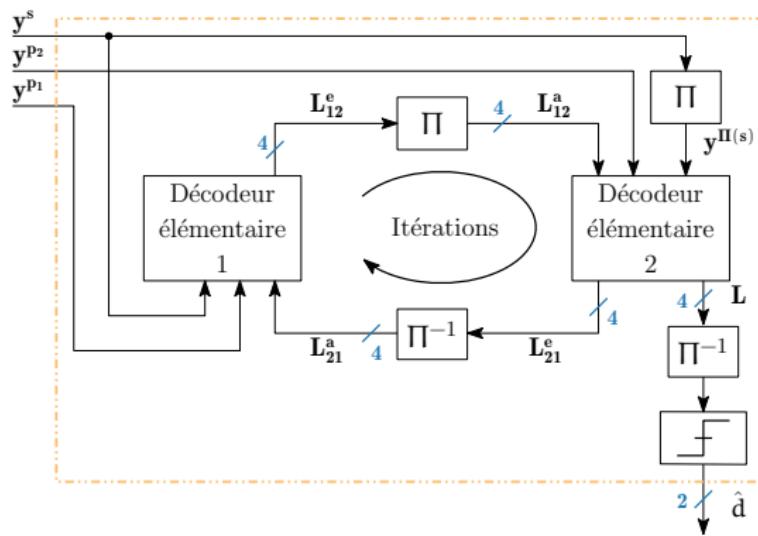
Extensions

- Métrique d'identification de symboles les moins fiables
- Génération des mots candidats

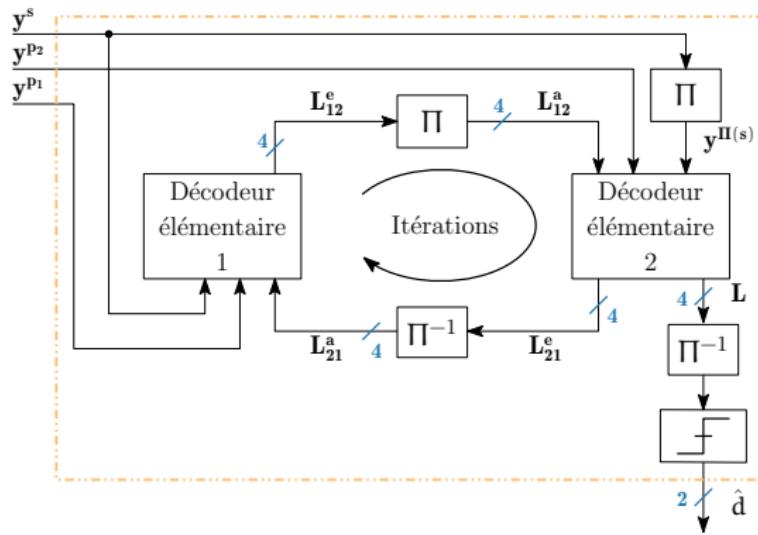
Le critère pour les turbo codes doubles binaires



Le critère pour les turbo codes doubles binaires



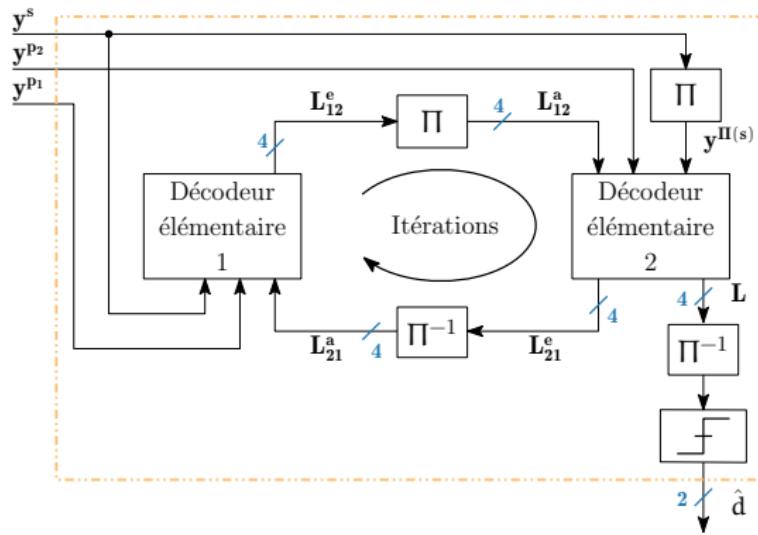
Le critère pour les turbo codes doubles binaires



	k_1	k_2	k_3	k_4
L_0	1.2			
L_1	3.8			
L_2	3.6			
L_3	5.5			

- Différentes manières d'étendre la métrique Δ

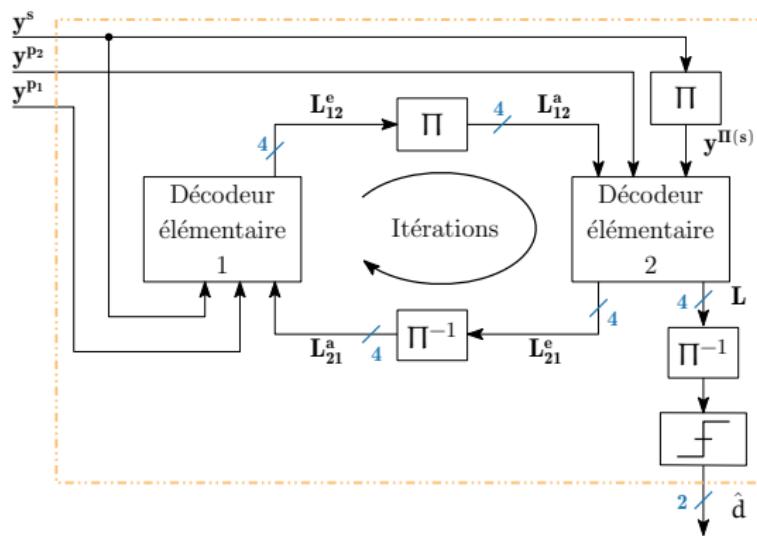
Le critère pour les turbo codes doubles binaires



	k_1	k_2	k_3	k_4
L_0	1.2			
L_1	3.8			
L_2	3.6			
L_3	5.5			
Δ	1.7			

- Différentes manières d'étendre la métrique Δ

Le critère pour les turbo codes doubles binaires



	k_1	k_2	k_3	k_4
L_0	1.2	3.3	5.5	3.1
L_1	3.8	3.5	1.3	4.4
L_2	3.6	0.2	1.1	4.0
L_3	5.5	0.4	0.2	1.4
Δ	1.7	0.2	4.2	0.4

- Différentes manières d'étendre la métrique Δ
- Études d'identification menées afin de sélectionner cette métrique

Génération des mots candidats pour les turbo codes doubles binaires

- Considérer tous les symboles possibles par position est coûteux

Génération des mots candidats pour les turbo codes doubles binaires

- Considérer tous les symboles possibles par position est coûteux
- $4^q = 2^{2q}$

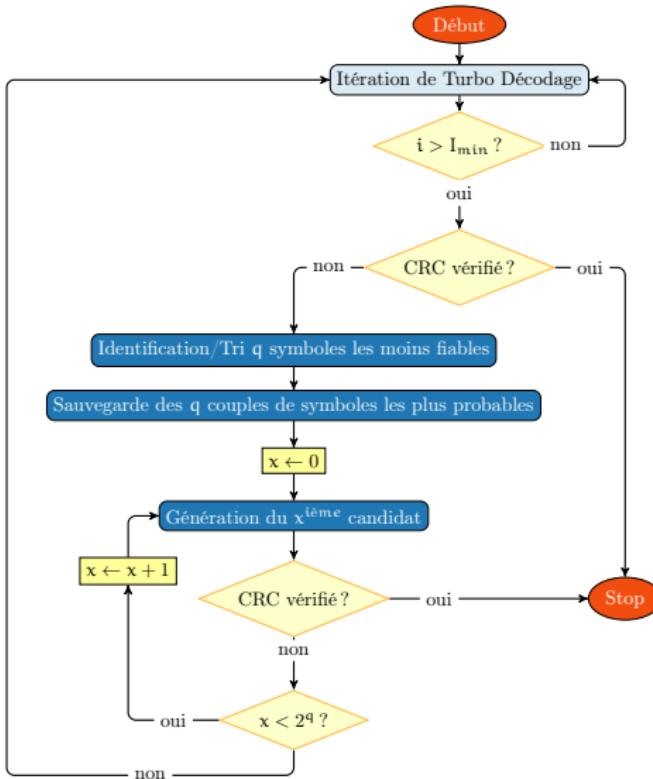
Génération des mots candidats pour les turbo codes doubles binaires

- Considérer tous les symboles possibles par position est coûteux
- $4^q = 2^{2q}$
- D'après les études menées, majoritairement, si un symbole est erroné, alors le symbole émis correspond au deuxième plus probable

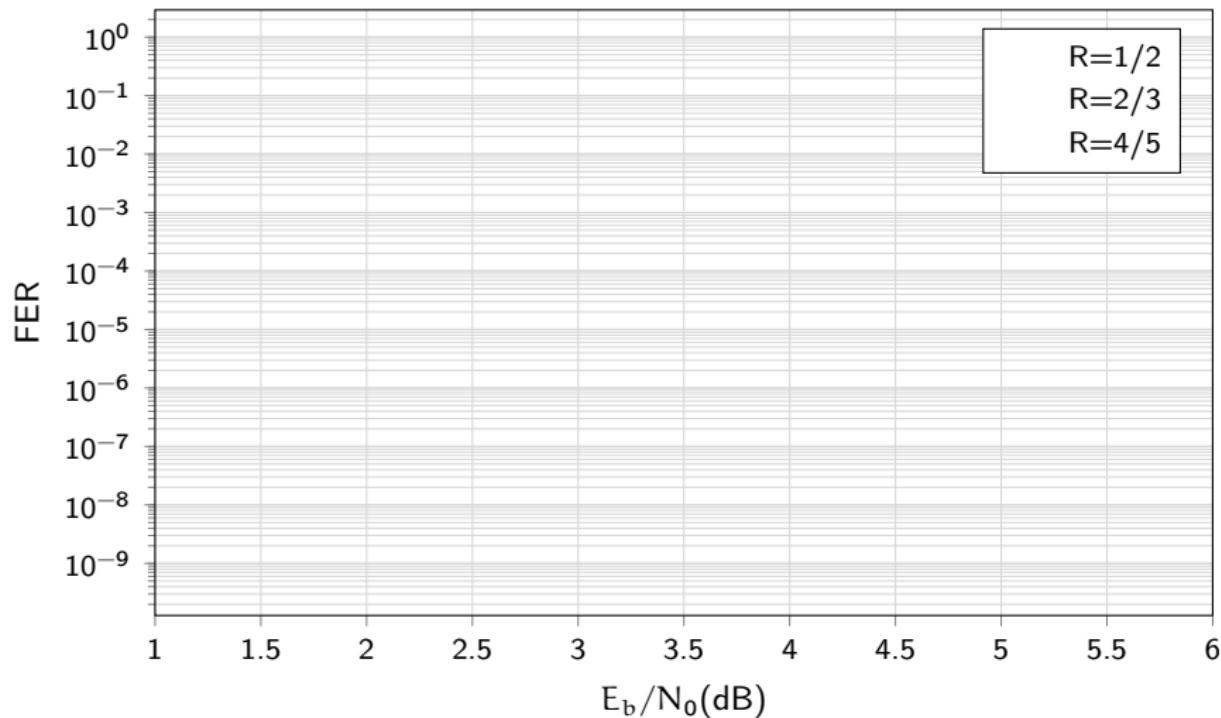
Génération des mots candidats pour les turbo codes doubles binaires

- Considérer tous les symboles possibles par position est coûteux
 - $4^q = 2^{2q}$
 - D'après les études menées, majoritairement, si un symbole est erroné, alors le symbole émis correspond au deuxième plus probable
- Mémorisation des 2 symboles les plus probables et génération de 2^q mots candidats

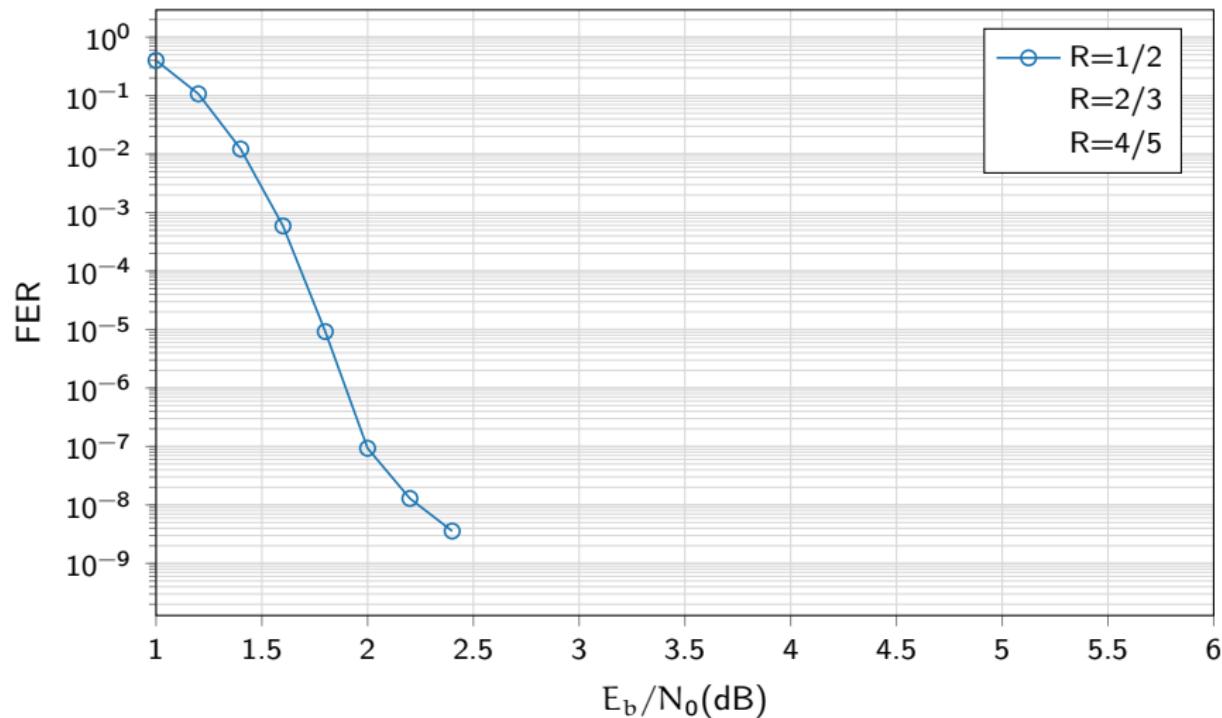
Algorithme FNC pour les turbo codes doubles binaires



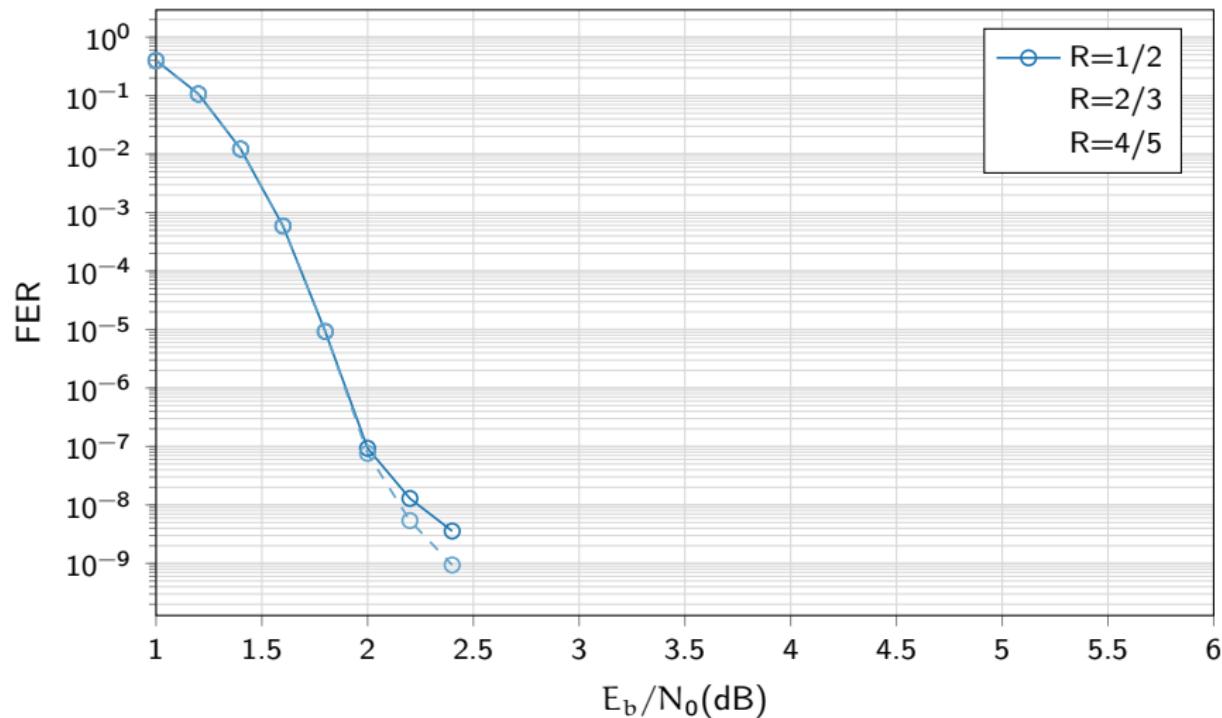
Performances de décodage - DVB-RCS2 - K=752 symb - QPSK - AWGN - 8 its - q=10



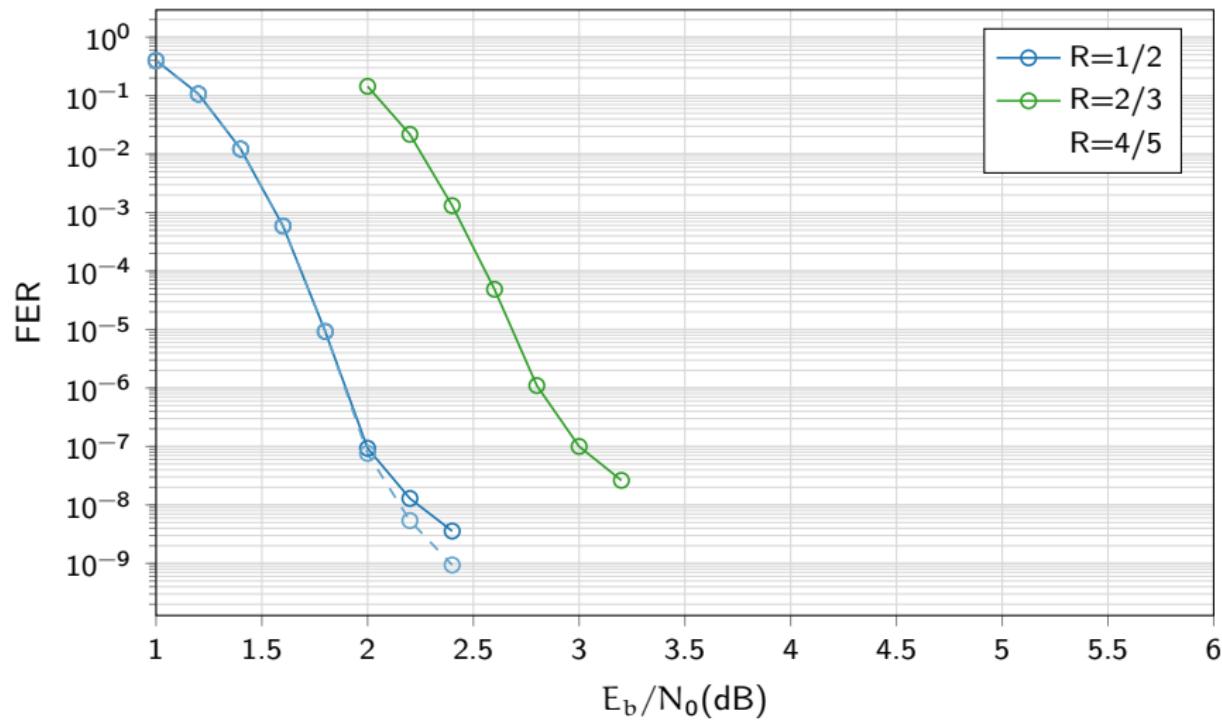
Performances de décodage - DVB-RCS2 - K=752 symb - QPSK - AWGN - 8 its - q=10



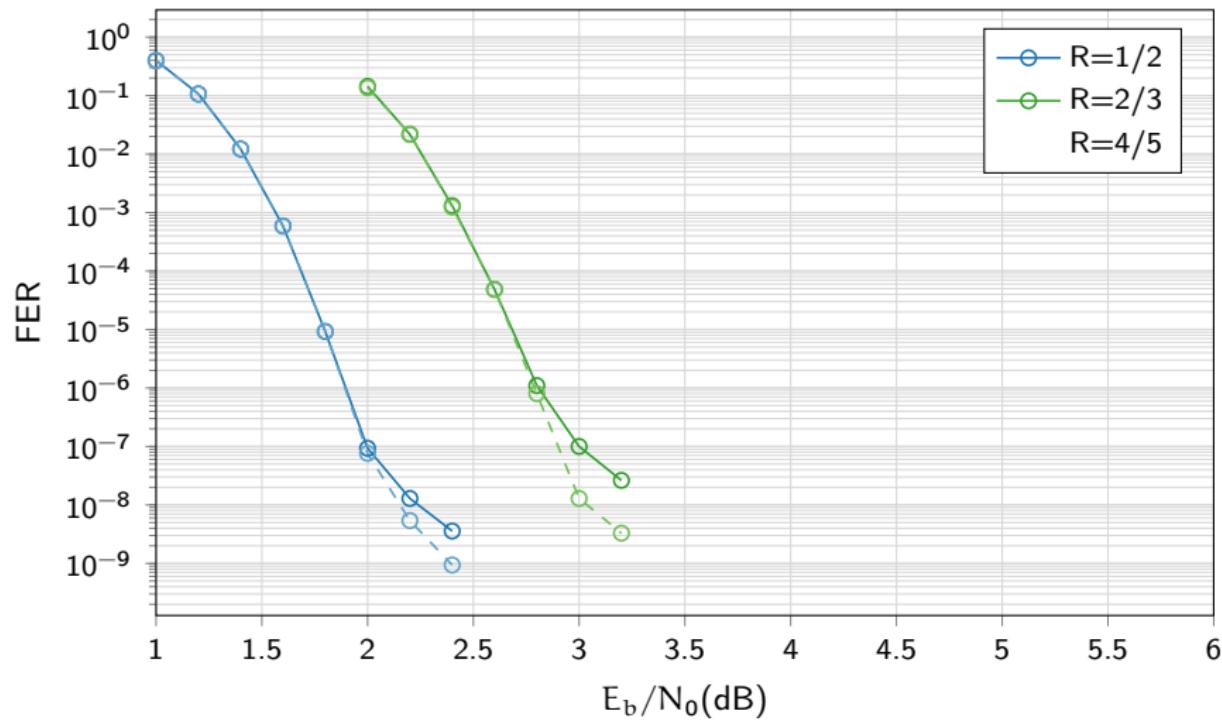
Performances de décodage - DVB-RCS2 - K=752 symb - QPSK - AWGN - 8 its - q=10



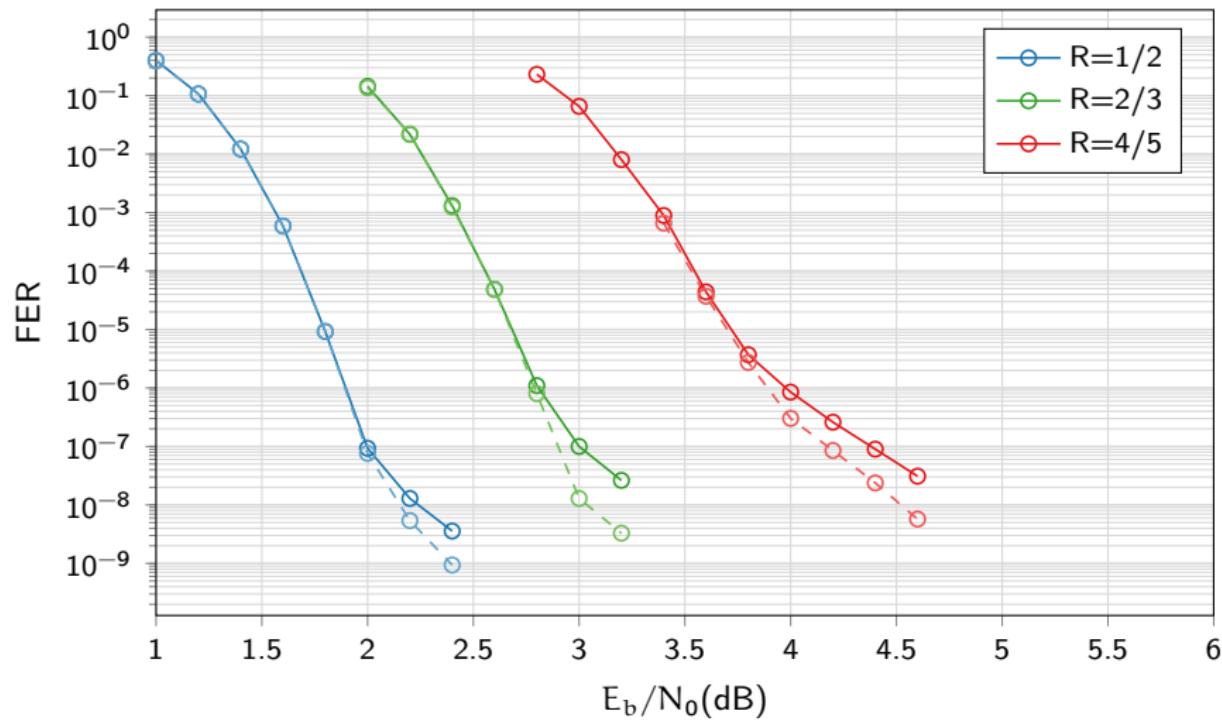
Performances de décodage - DVB-RCS2 - K=752 symb - QPSK - AWGN - 8 its - q=10



Performances de décodage - DVB-RCS2 - K=752 symb - QPSK - AWGN - 8 its - q=10



Performances de décodage - DVB-RCS2 - K=752 symb - QPSK - AWGN - 8 its - q=10



Plan

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

③ Architecture matérielle de correction des erreurs résiduelles

Les paramètres de l'algorithme FNC

Implantation matérielle de l'algorithme FNC

④ Conclusion et perspectives

Plan

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

③ Architecture matérielle de correction des erreurs résiduelles

 Les paramètres de l'algorithme FNC

 Implantation matérielle de l'algorithme FNC

④ Conclusion et perspectives

Adéquation algorithme architecture

- L'algorithme FNC est ajustable en fonction de deux paramètres :

Adéquation algorithme architecture

- L'algorithme FNC est ajustable en fonction de deux paramètres :
 - Nombre de mots candidats considérés

Adéquation algorithme architecture

- L'algorithme FNC est ajustable en fonction de deux paramètres :
 - Nombre de mots candidats considérés
 - Itérations à l'issue desquelles le FNC est appliqué

Adéquation algorithme architecture

- L'algorithme FNC est ajustable en fonction de deux paramètres :
 - Nombre de mots candidats considérés
 - Itérations à l'issue desquelles le FNC est appliqué
- Double impact : complexité calculatoire et performances de décodage

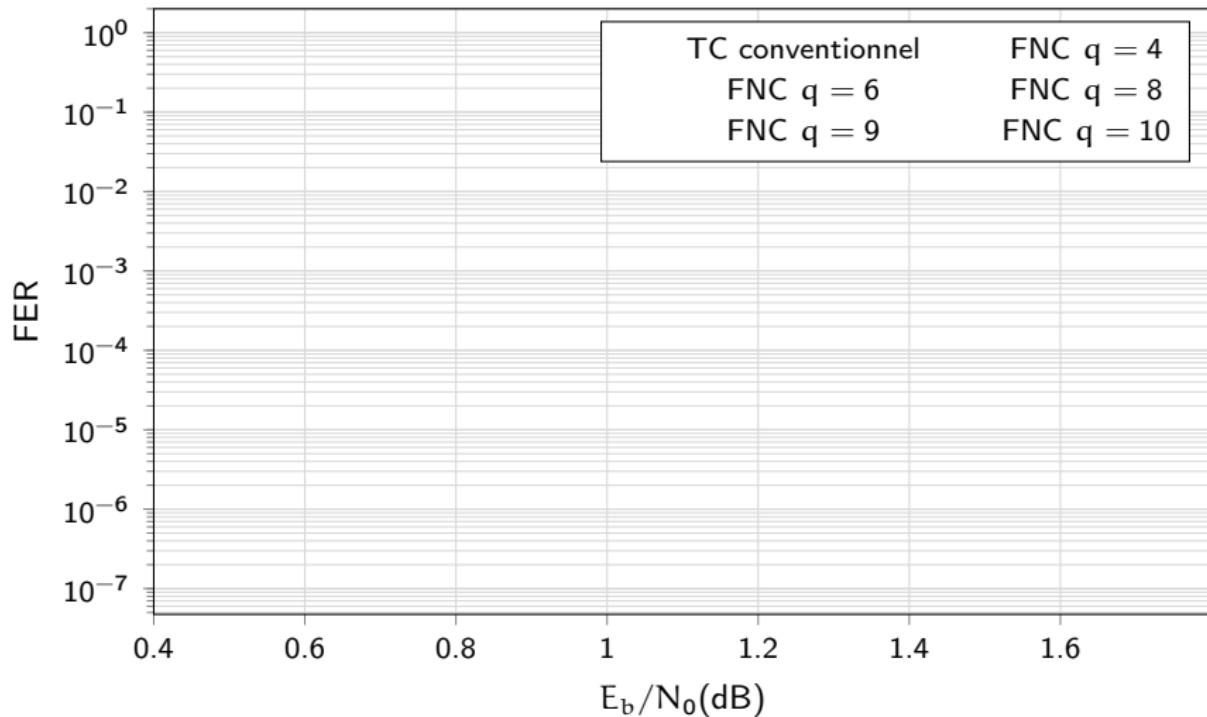
Adéquation algorithme architecture

- L'algorithme FNC est ajustable en fonction de deux paramètres :
 - Nombre de mots candidats considérés
→ Une position supplémentaire ⇒ deux fois plus de mots candidats
 - Itérations à l'issue desquelles le FNC est appliqué
- Double impact : complexité calculatoire et performances de décodage

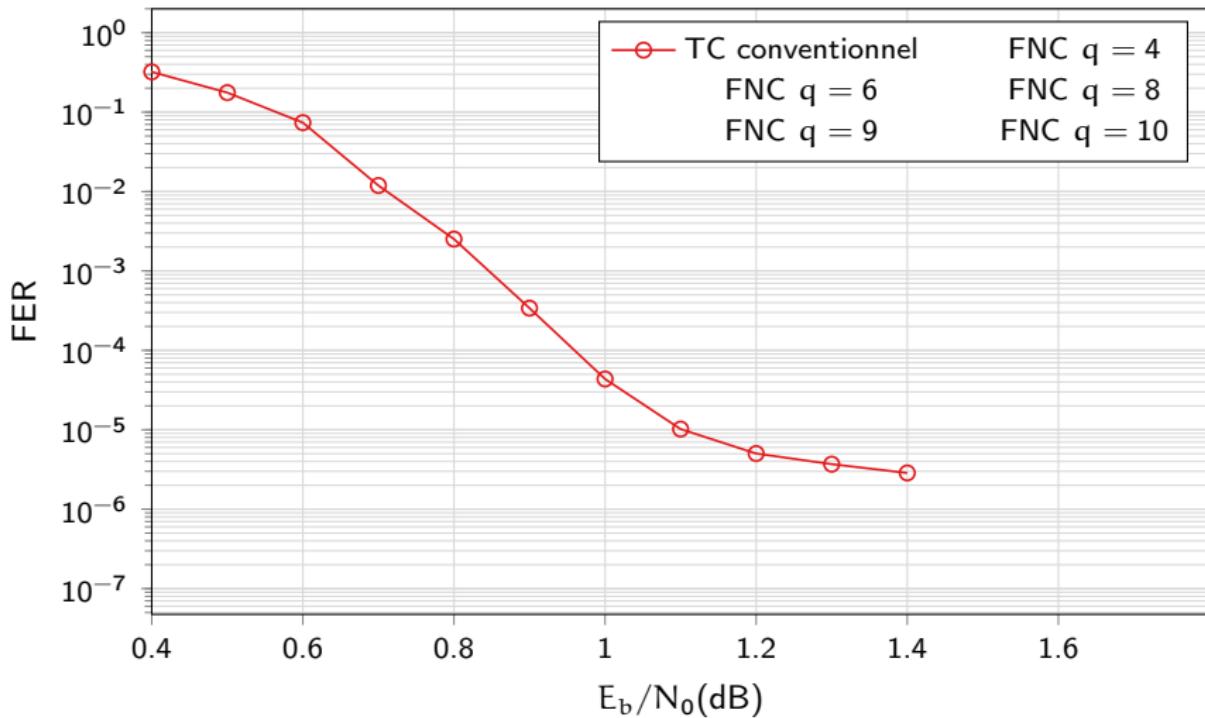
Adéquation algorithme architecture

- L'algorithme FNC est ajustable en fonction de deux paramètres :
 - Nombre de mots candidats considérés
→ Une position supplémentaire ⇒ deux fois plus de mots candidats
 - Itérations à l'issue desquelles le FNC est appliqué
→ Modification du budget temporel alloué
- Double impact : complexité calculatoire et performances de décodage

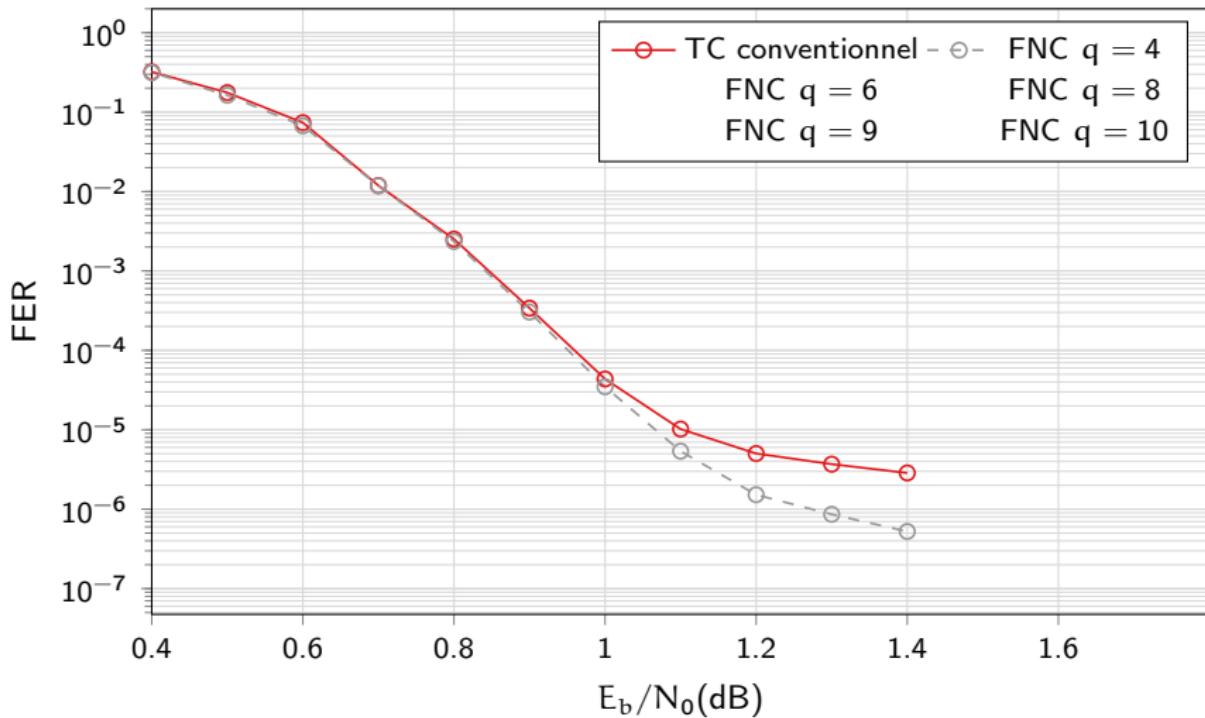
Variation du nombre de mots candidats - LTE, K=2048, R=1/3, 8its



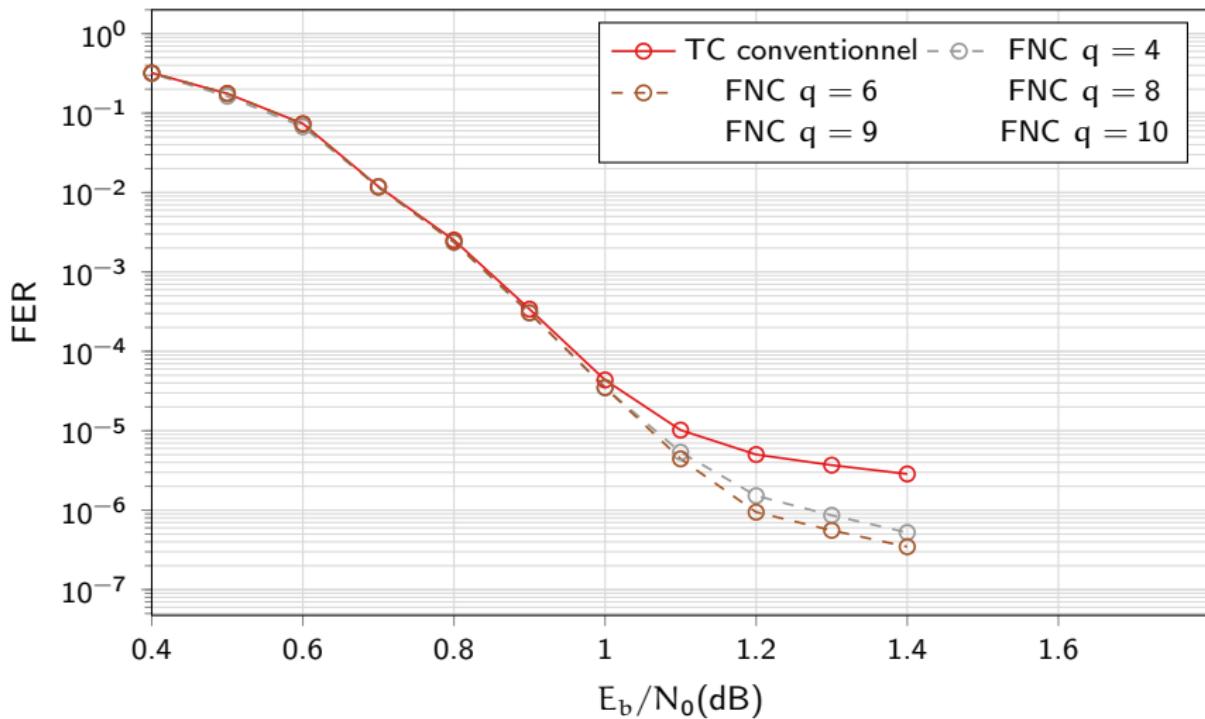
Variation du nombre de mots candidats - LTE, K=2048, R=1/3, 8its



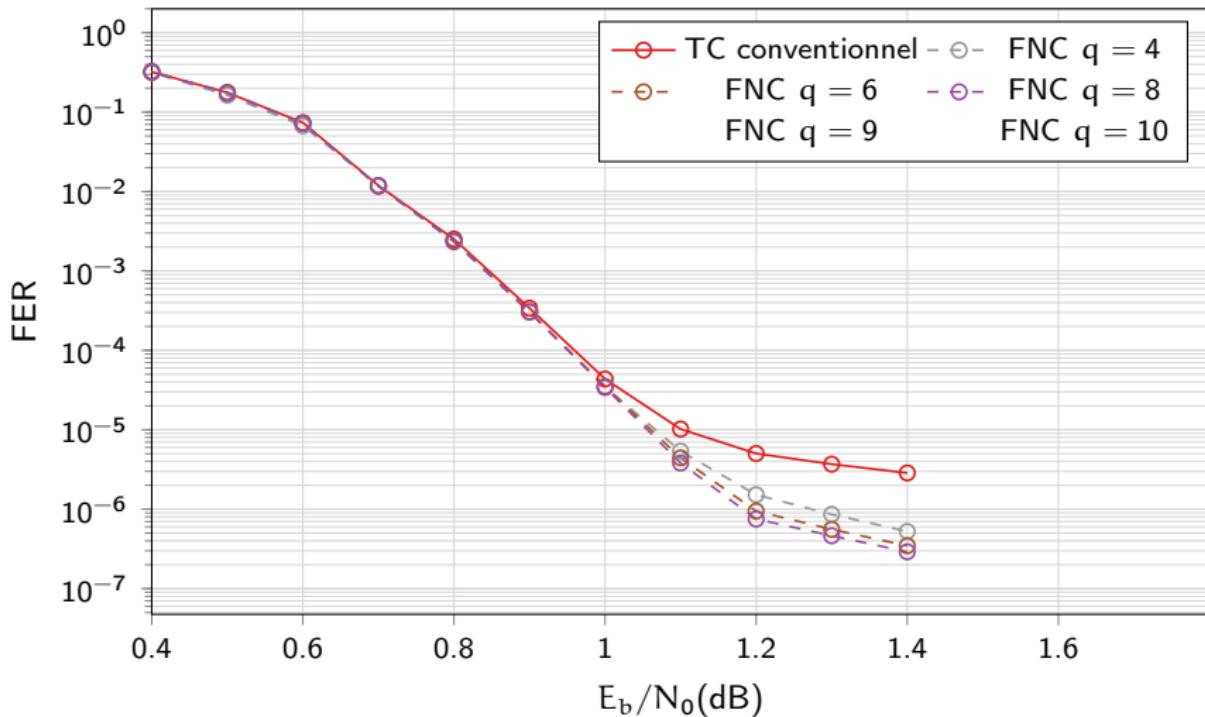
Variation du nombre de mots candidats - LTE, K=2048, R=1/3, 8its



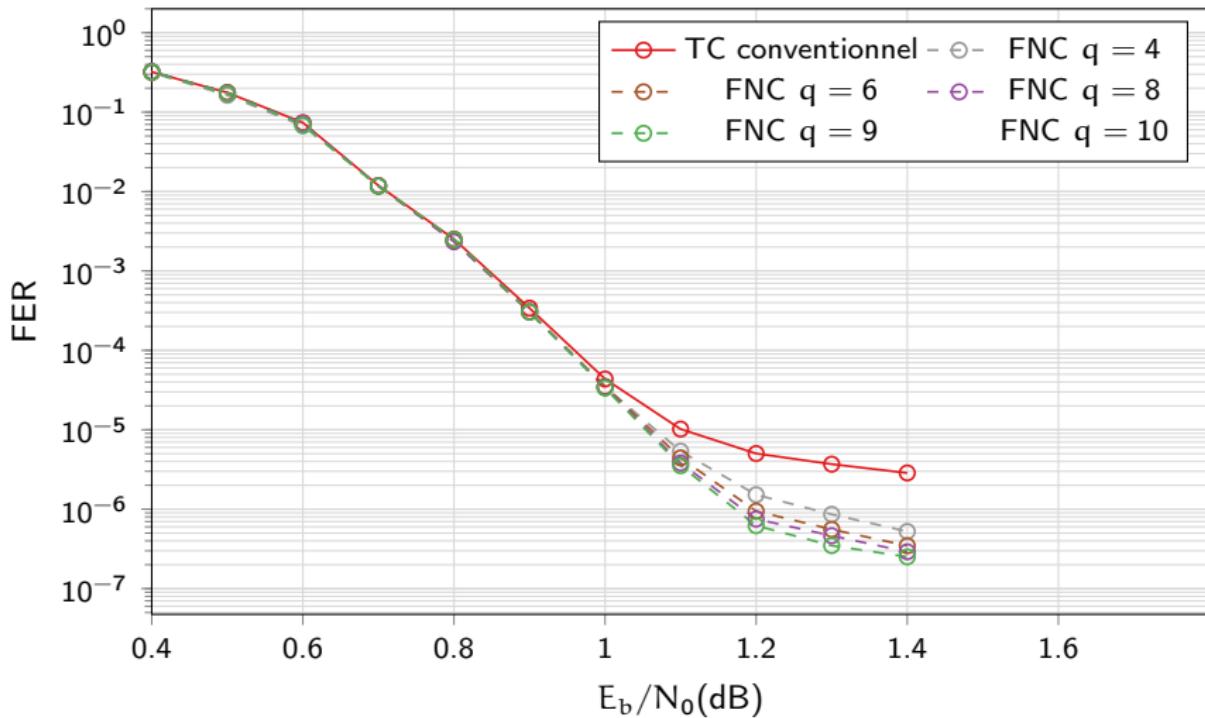
Variation du nombre de mots candidats - LTE, K=2048, R=1/3, 8its



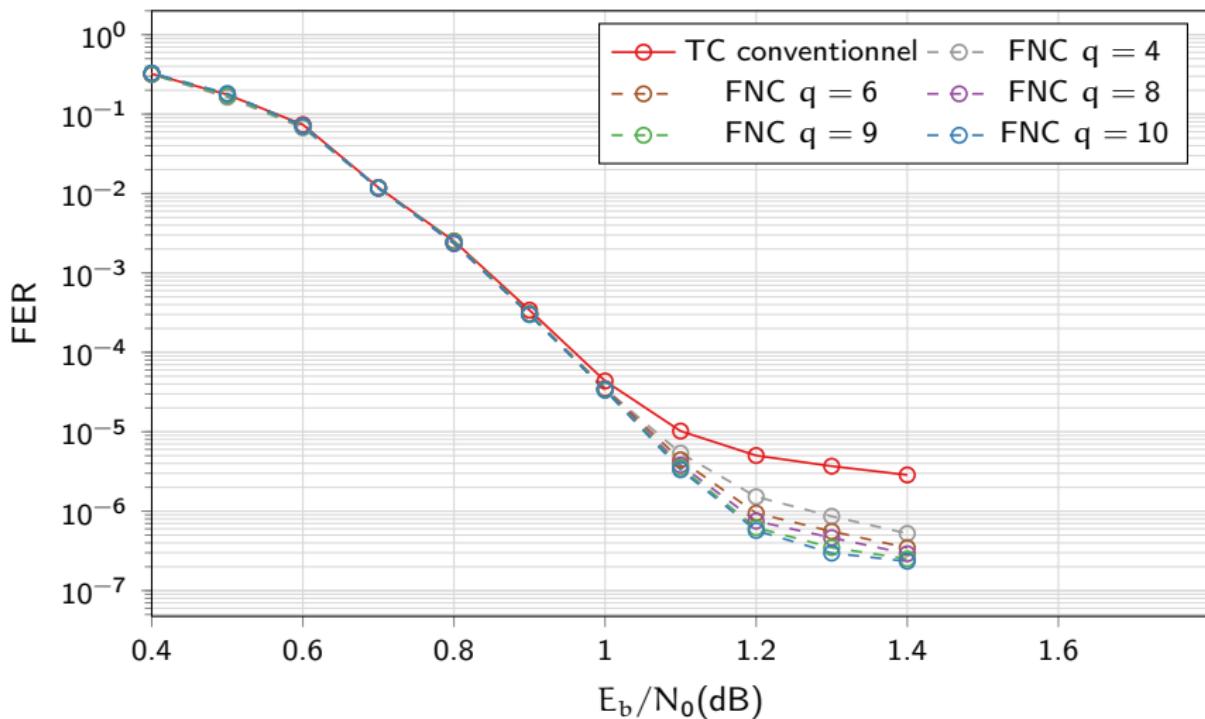
Variation du nombre de mots candidats - LTE, K=2048, R=1/3, 8its

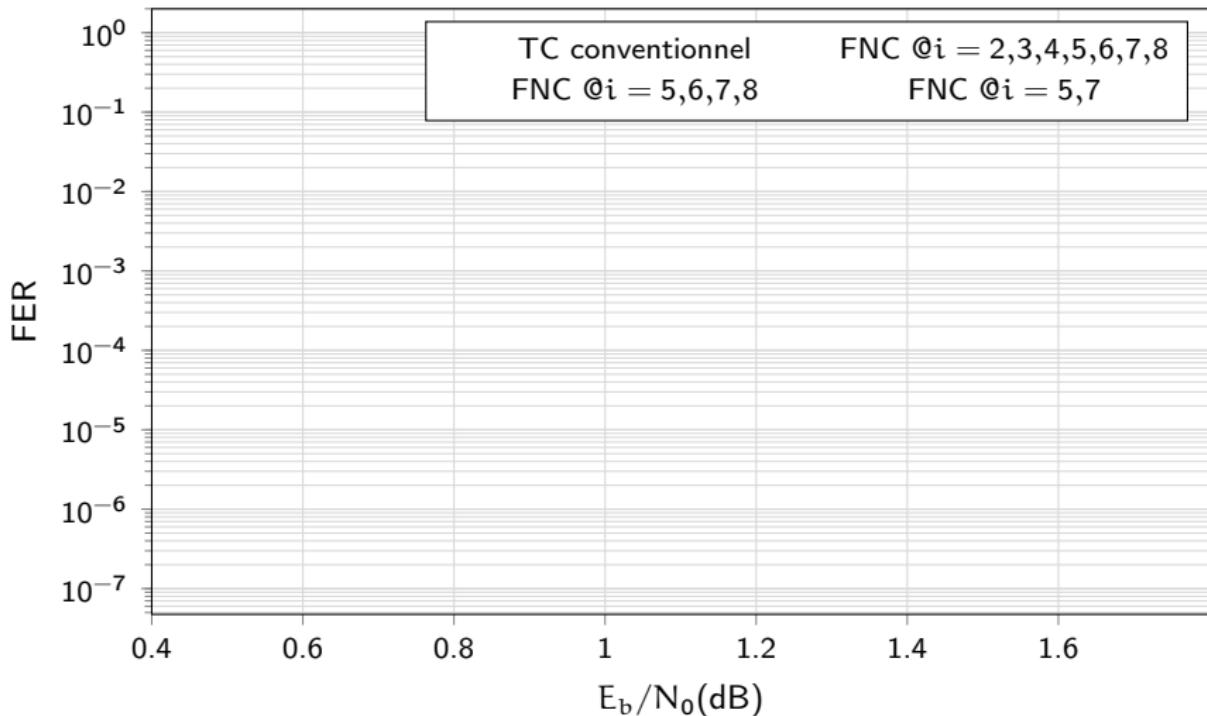


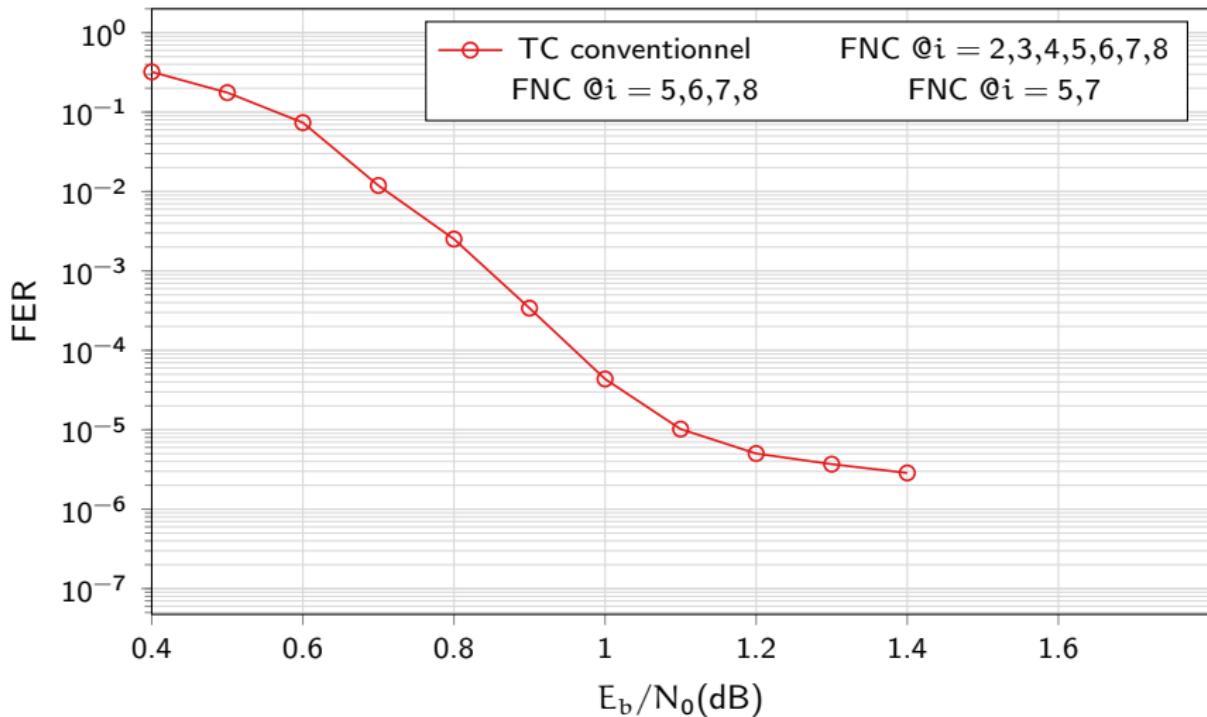
Variation du nombre de mots candidats - LTE, K=2048, R=1/3, 8its

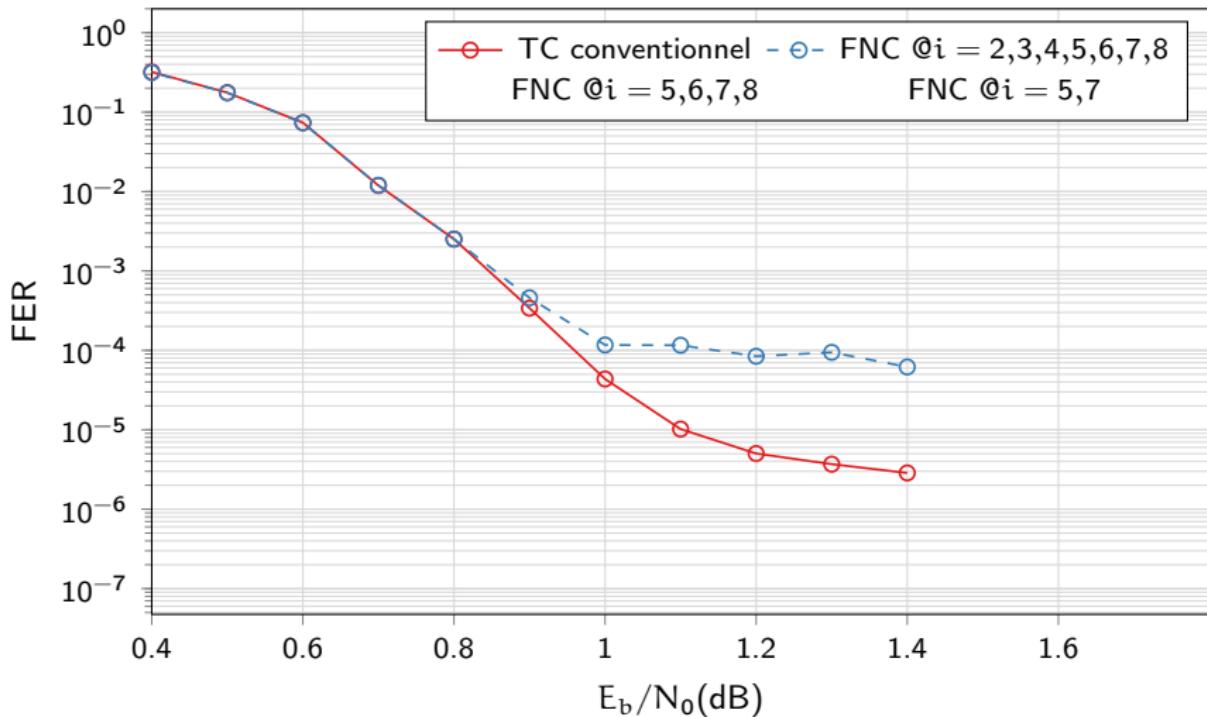


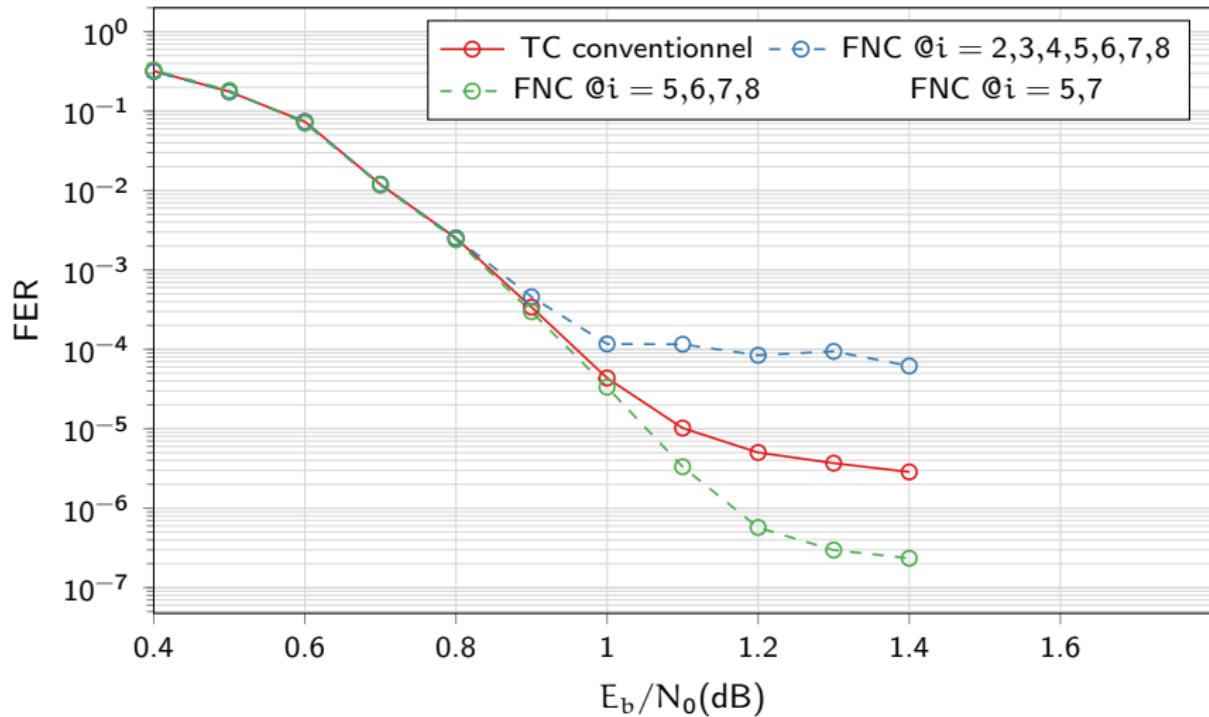
Variation du nombre de mots candidats - LTE, K=2048, R=1/3, 8its

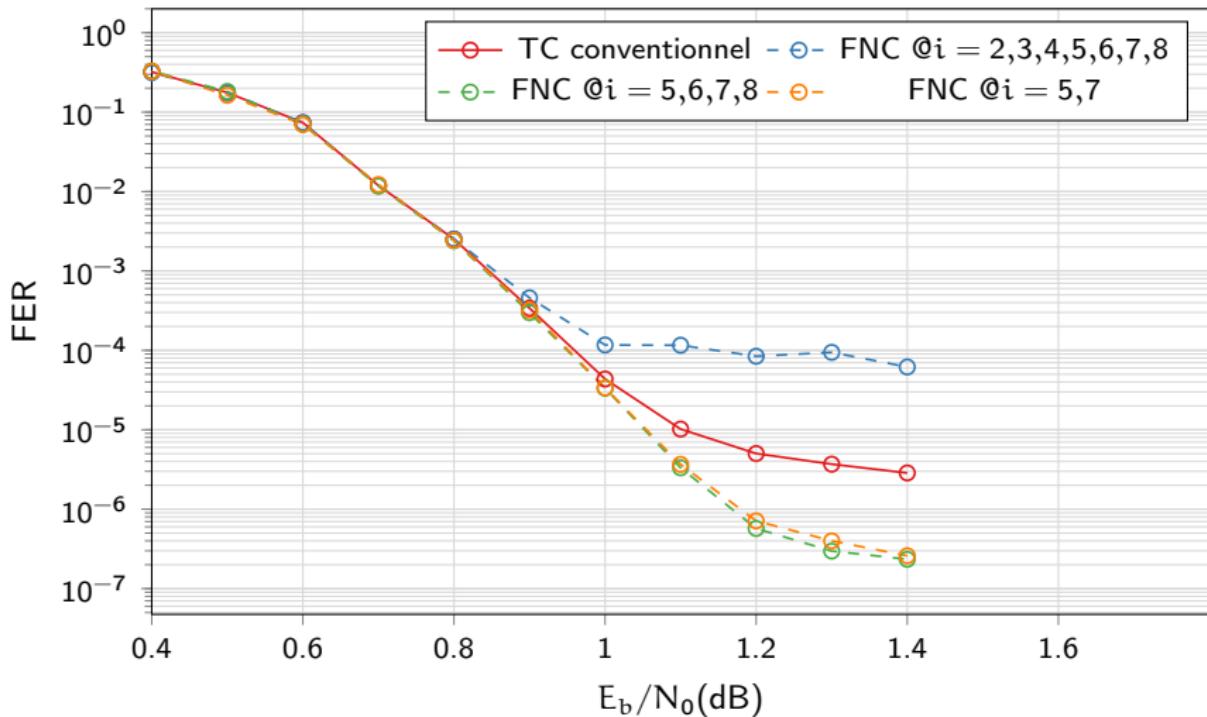


Itérations à l'issue desquelles le processus FNC est appliqué - LTE, K=2048, R=1/3, 8its

Itérations à l'issue desquelles le processus FNC est appliqué - LTE, K=2048, R=1/3, 8its

Itérations à l'issue desquelles le processus FNC est appliqu   - LTE, K=2048, R=1/3, 8its

Itérations à l'issue desquelles le processus FNC est appliqu   - LTE, K=2048, R=1/3, 8its

Itérations à l'issue desquelles le processus FNC est appliqu   - LTE, K=2048, R=1/3, 8its

Conclusion sur l'impact des paramètres de l'algorithme

① Nombre de mots candidats :

- Augmenter le nombre de mots candidats augmente le pouvoir de correction
- Mais la complexité calculatoire est aussi augmentée

Conclusion sur l'impact des paramètres de l'algorithme

① Nombre de mots candidats :

- Augmenter le nombre de mots candidats augmente le pouvoir de correction
- Mais la complexité calculatoire est aussi augmentée

② Itérations :

- Appliquer trop tôt le FNC est problématique : faux positifs du CRC
- Cela dépend du pouvoir de détection du CRC et donc de la taille de la trame
- Il est possible de réduire le nombre d'applications tout en conservant les gains

Conclusion sur l'impact des paramètres de l'algorithme

① Nombre de mots candidats :

- Augmenter le nombre de mots candidats augmente le pouvoir de correction
- Mais la complexité calculatoire est aussi augmentée

② Itérations :

- Appliquer trop tôt le FNC est problématique : faux positifs du CRC
- Cela dépend du pouvoir de détection du CRC et donc de la taille de la trame
- Il est possible de réduire le nombre d'applications tout en conservant les gains

→ Choix des paramètres à définir par le concepteur suivant le compromis entre les gains de décodage et le surcoût calculatoire acceptable

Plan

① Contexte des travaux de thèse

② Abaissement du plancher d'erreurs des turbo codes

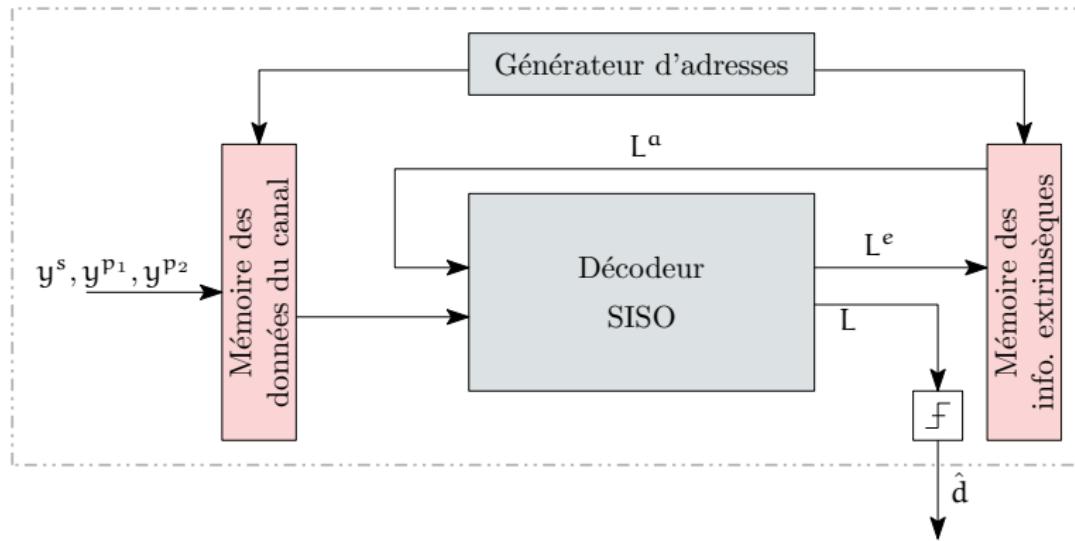
③ Architecture matérielle de correction des erreurs résiduelles

 Les paramètres de l'algorithme FNC

 Implantation matérielle de l'algorithme FNC

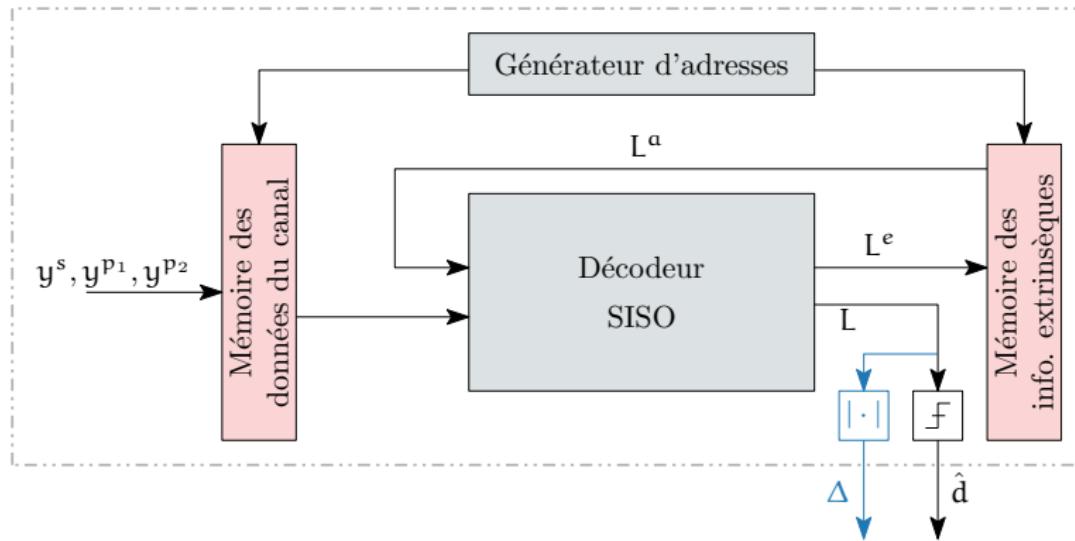
④ Conclusion et perspectives

Interconnexion entre le turbo décodeur et l'architecture FNC



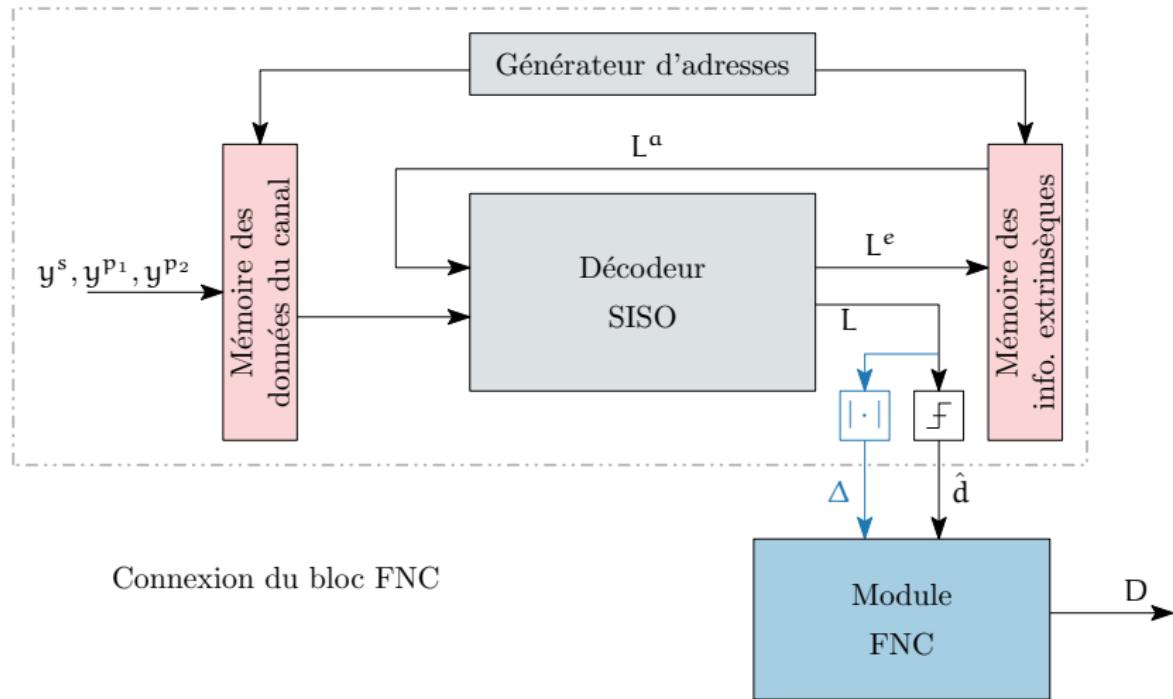
Principe du turbo décodage séquentiel

Interconnexion entre le turbo décodeur et l'architecture FNC

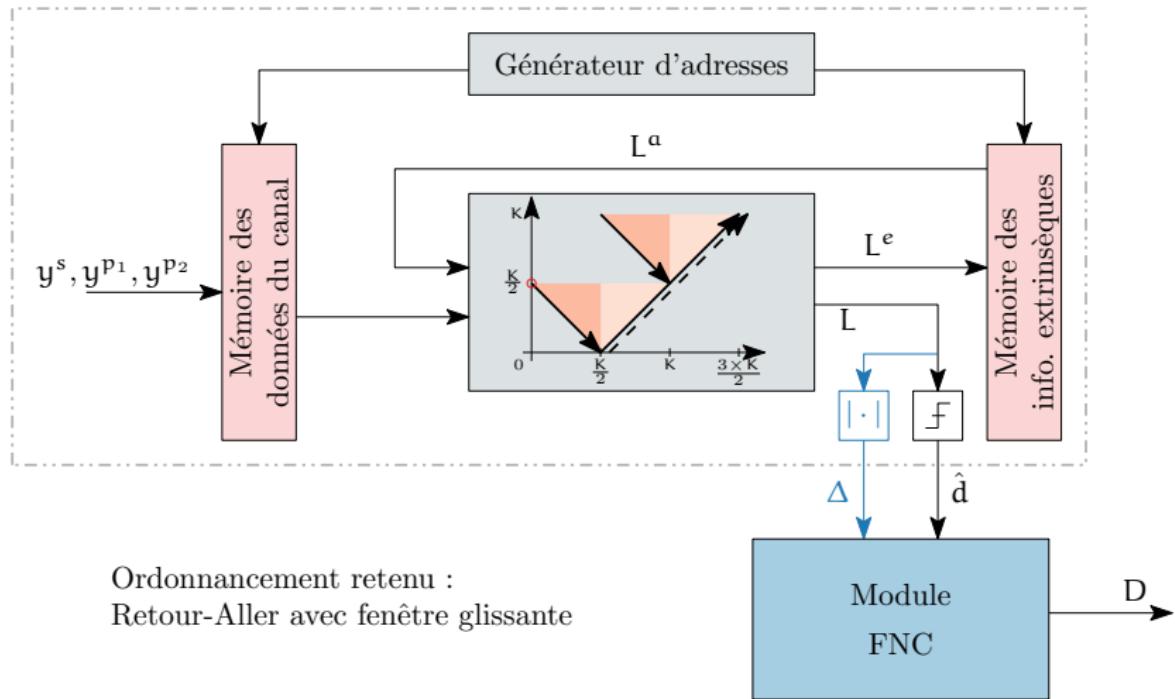


Calcul de Δ

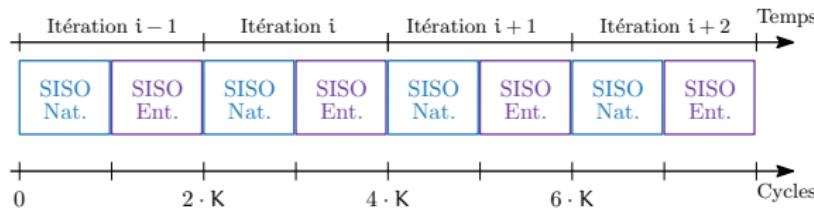
Interconnexion entre le turbo décodeur et l'architecture FNC



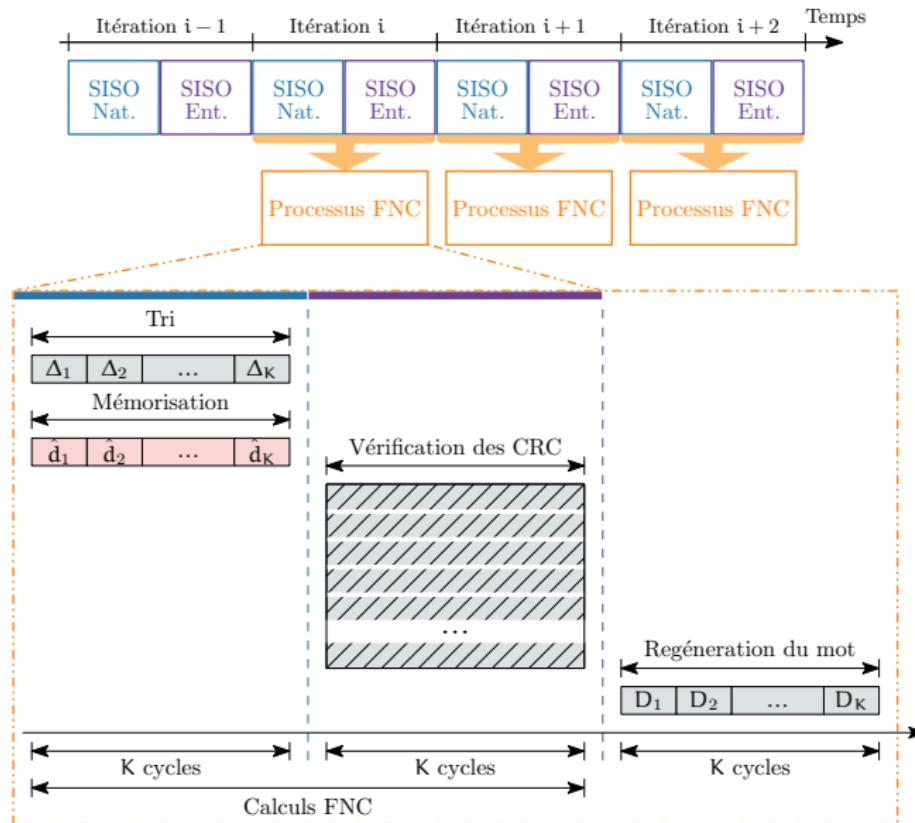
Interconnexion entre le turbo décodeur et l'architecture FNC



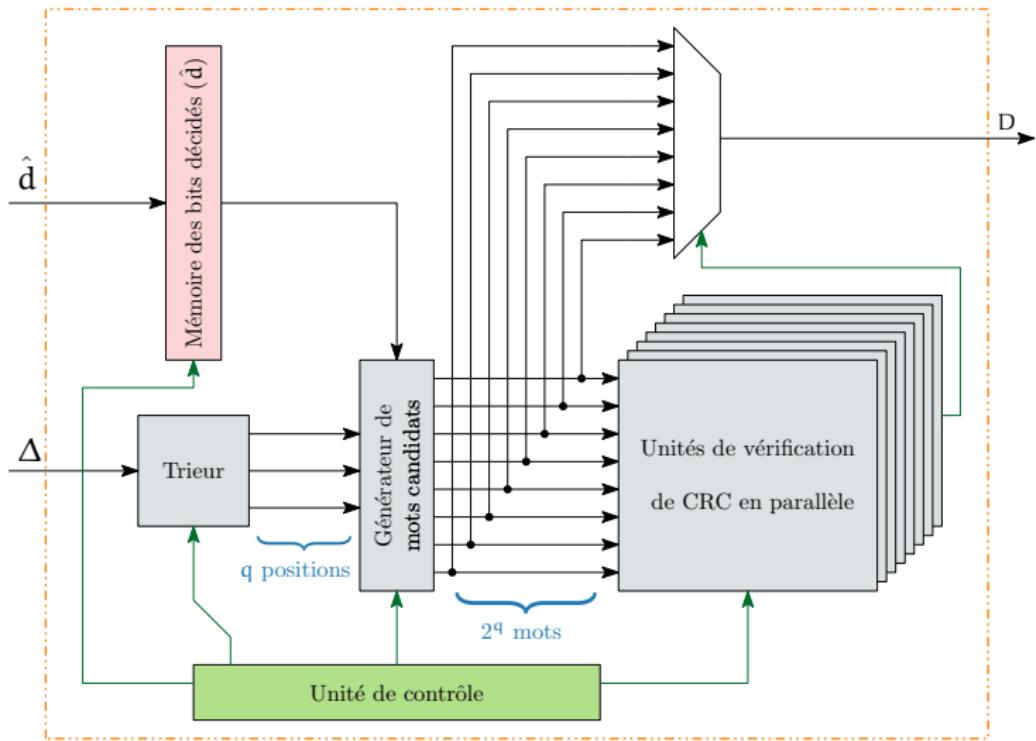
Ordonnancement du système turbo décodeur + module FNC



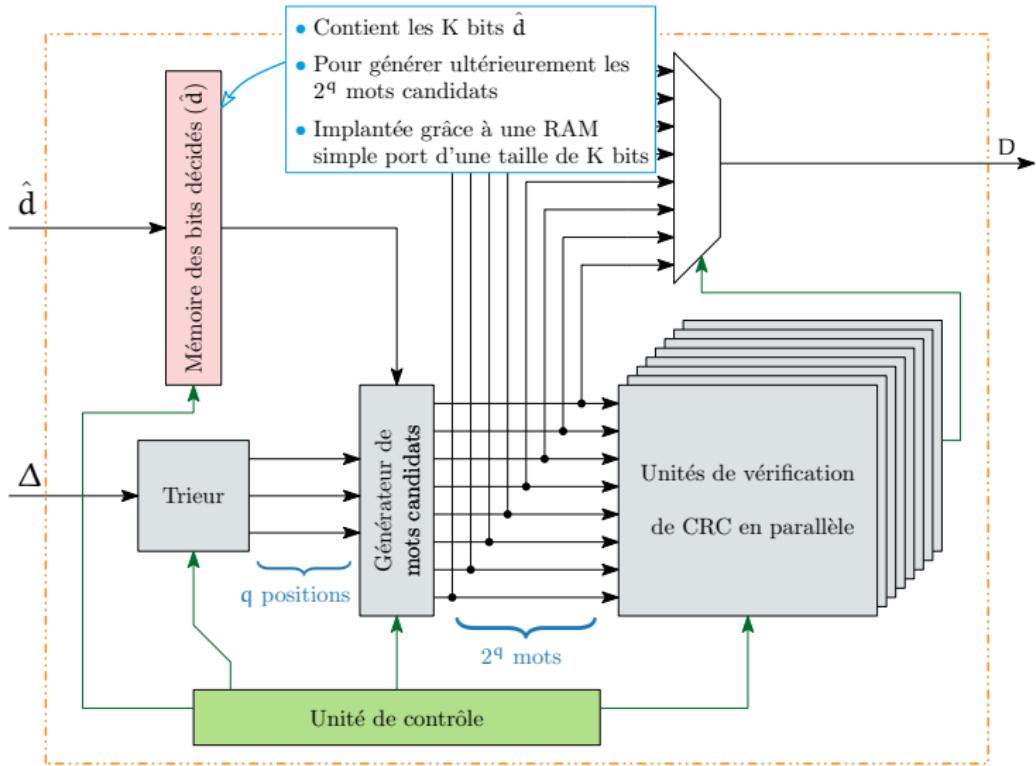
Ordonnancement du système turbo décodeur + module FNC



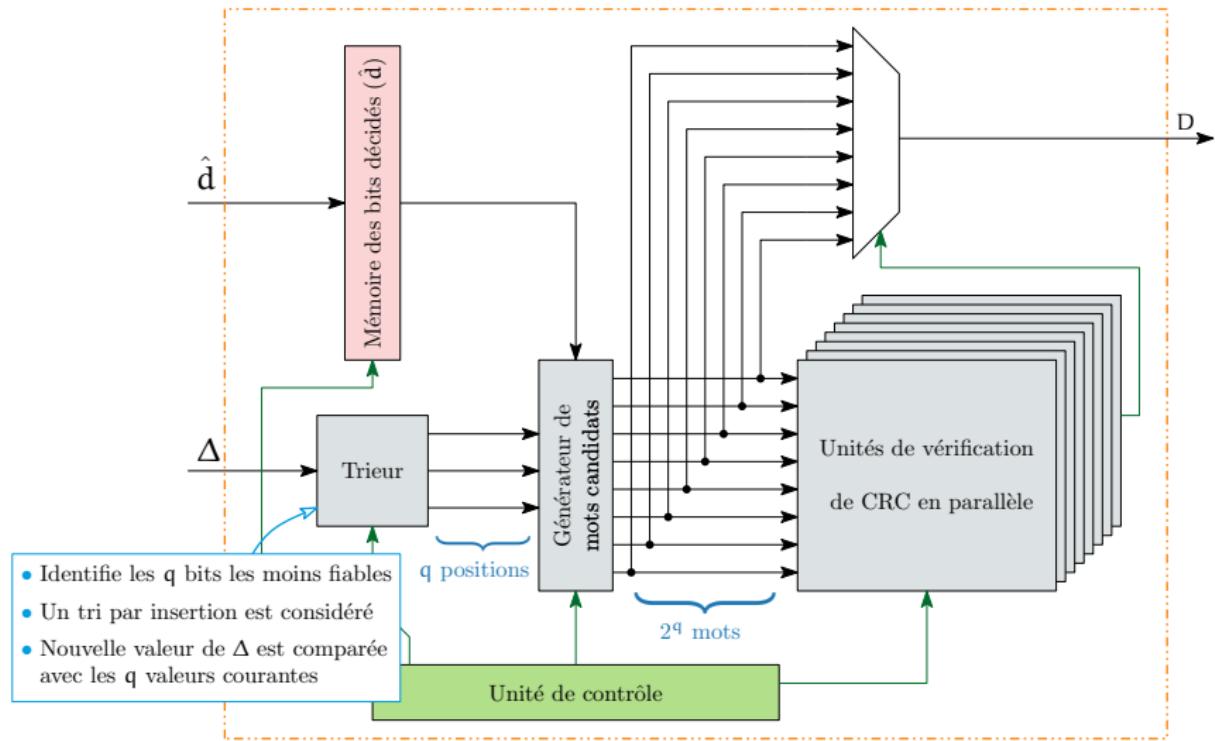
Architecture du module FNC : Vue d'ensemble et détails



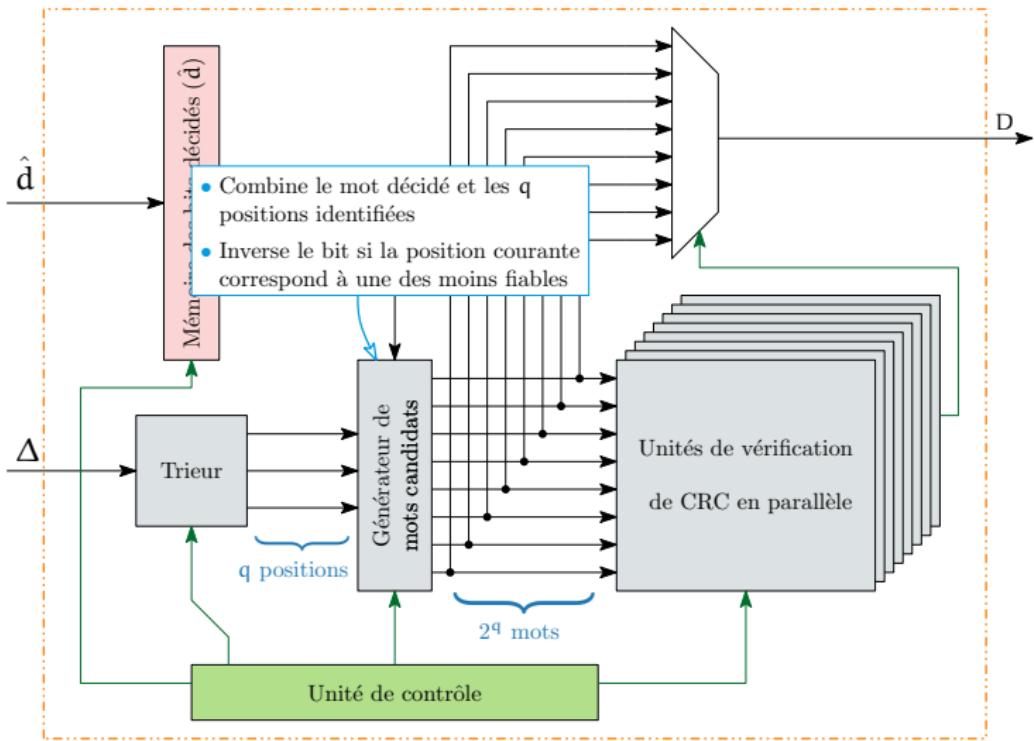
Architecture du module FNC : Vue d'ensemble et détails



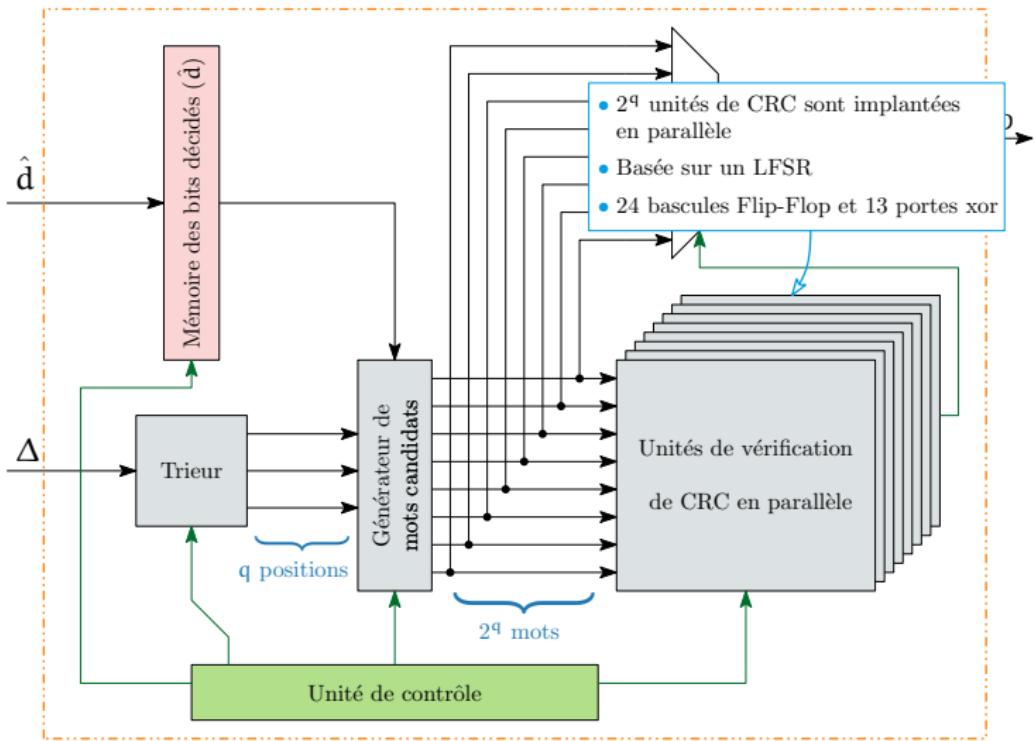
Architecture du module FNC : Vue d'ensemble et détails



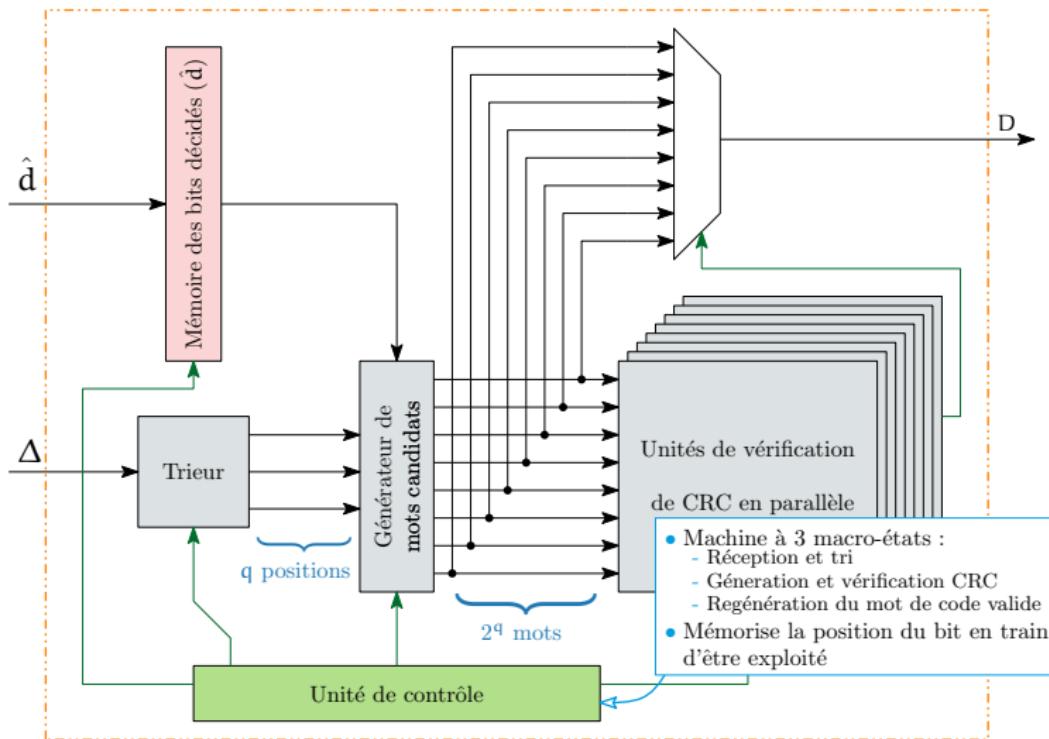
Architecture du module FNC : Vue d'ensemble et détails



Architecture du module FNC : Vue d'ensemble et détails

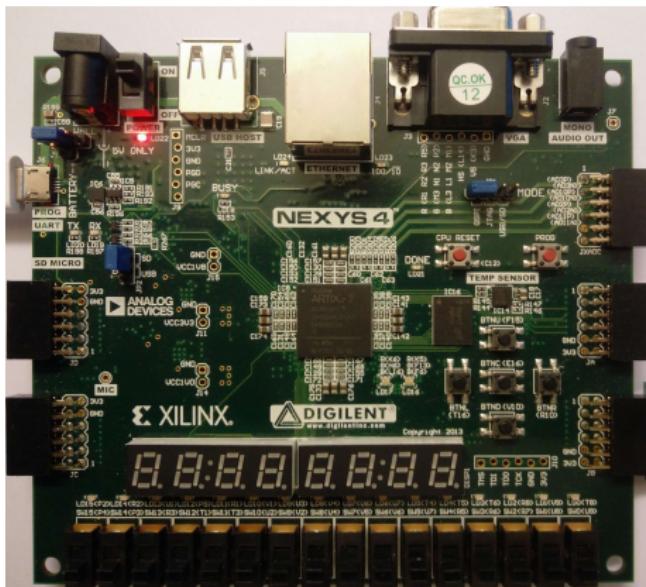


Architecture du module FNC : Vue d'ensemble et détails



Résultats d'implantation - Paramètres

- Architecture décrite en VHDL → FPGA
- Validation par co-simulation
- Description générique avec K et q
- K=6144, pire cas → 1 bloc RAM utilisé (Xilinx, 18 Kbits).

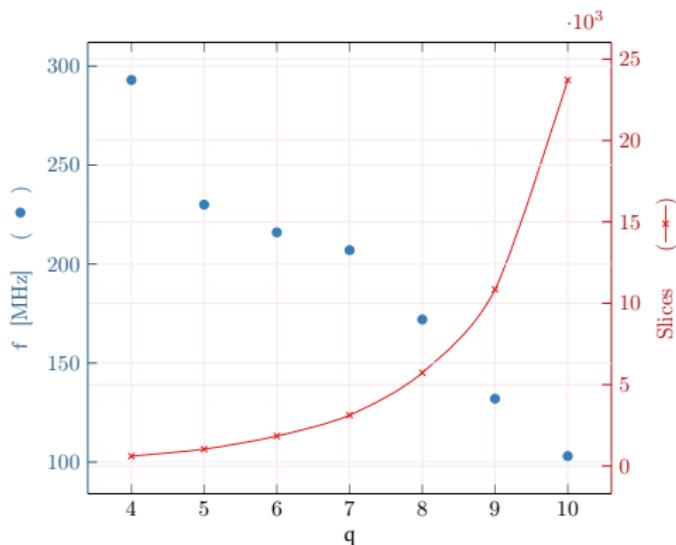


Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103

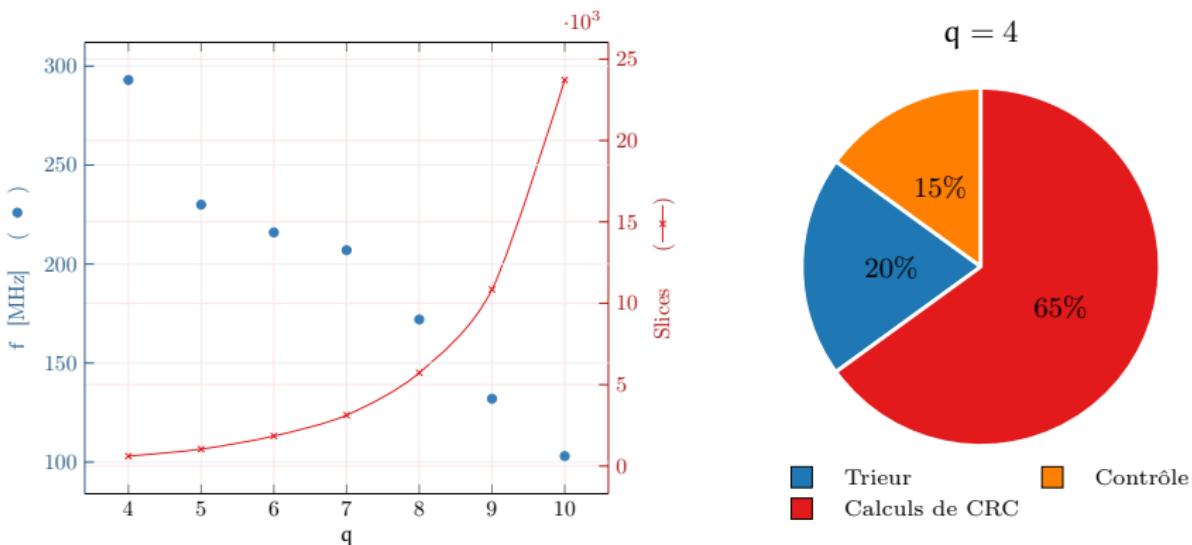
Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103



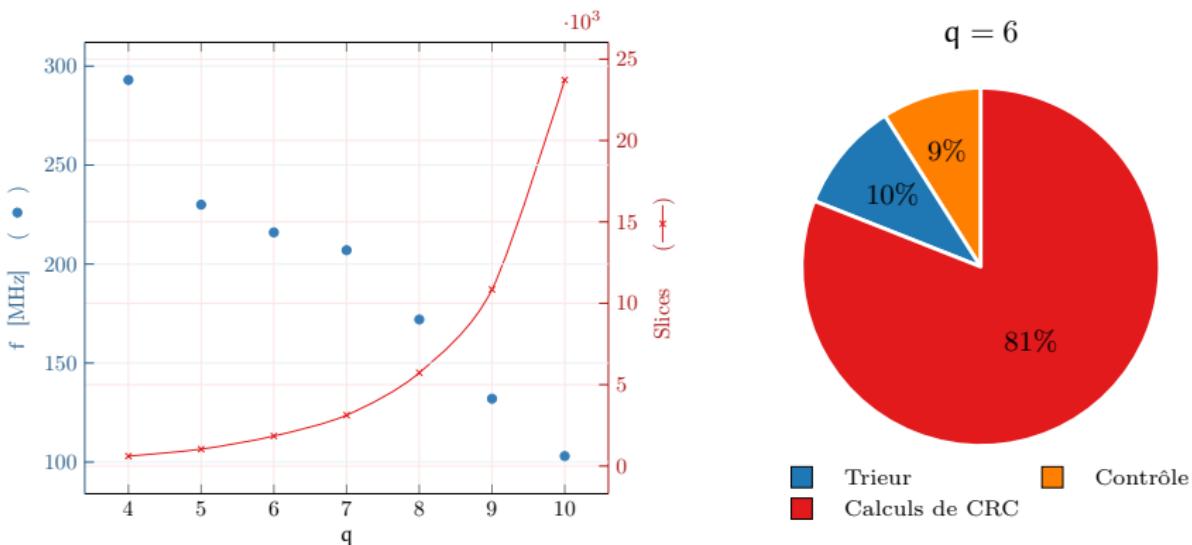
Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103



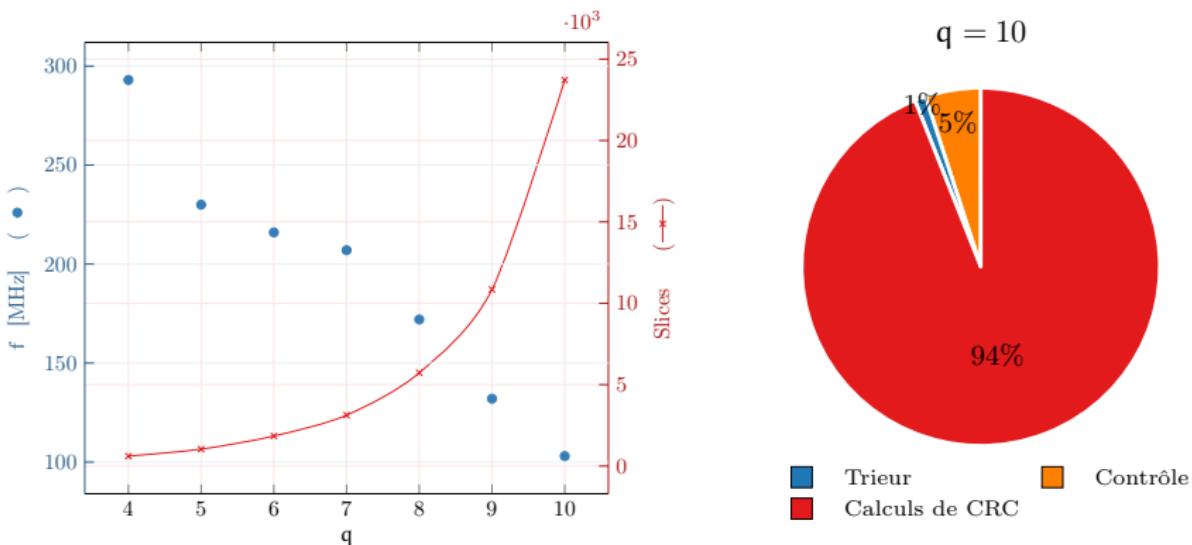
Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103



Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103



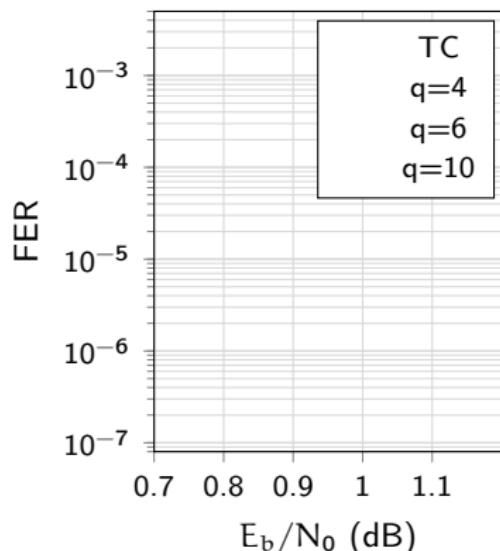
Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103

IP Core Xilinx :

LUT	FF	Slices	BRAM	f (MHz)
3712	4062	3765	13	285

FER @ 1 dB Slices (ratio)



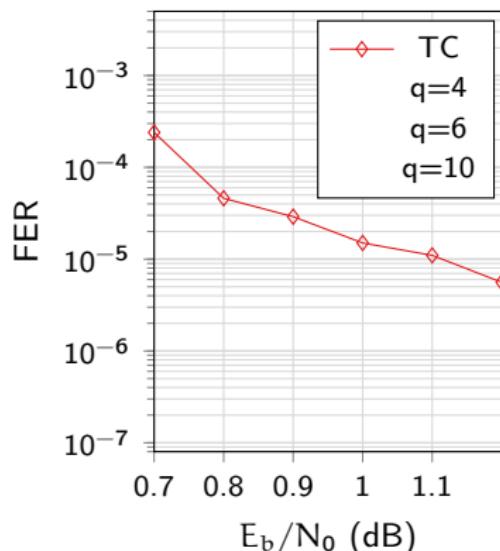
Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103

IP Core Xilinx :

LUT	FF	Slices	BRAM	f (MHz)
3712	4062	3765	13	285

	FER @ 1 dB	Slices (ratio)
TC	$1,5 \times 10^{-5}$	



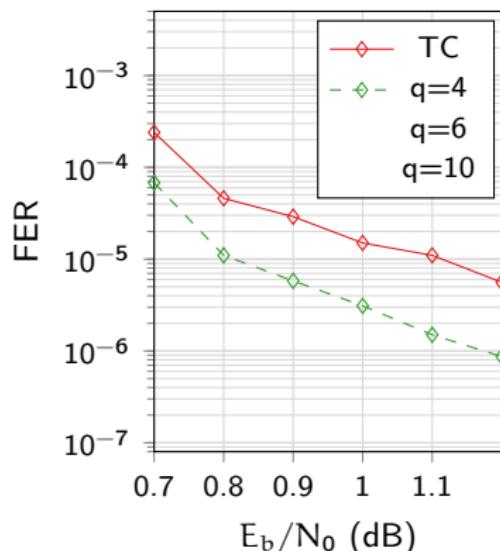
Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103

IP Core Xilinx :

LUT	FF	Slices	BRAM	f (MHz)
3712	4062	3765	13	285

	FER @ 1 dB	Slices (ratio)
TC	$1,5 \times 10^{-5}$	
q = 4	3×10^{-6}	16%



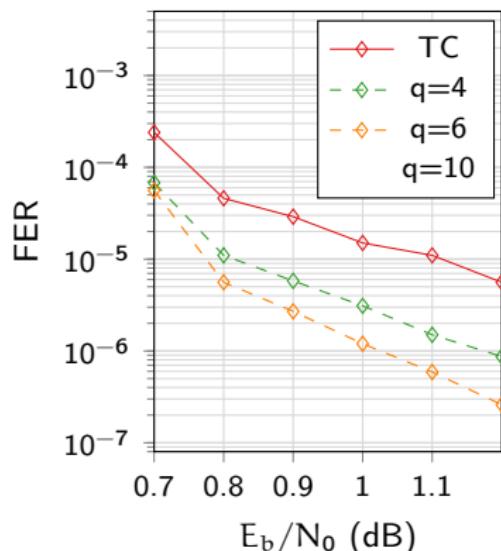
Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103

IP Core Xilinx :

LUT	FF	Slices	BRAM	f (MHz)
3712	4062	3765	13	285

	FER @ 1 dB	Slices (ratio)
TC	$1,5 \times 10^{-5}$	
q = 4	3×10^{-6}	16%
q = 6	$1,2 \times 10^{-6}$	49%



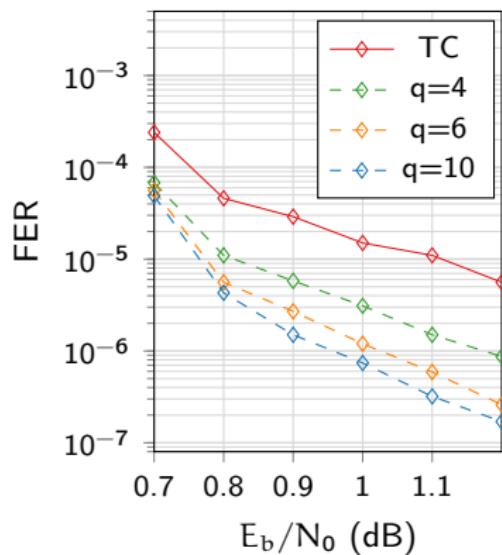
Résultats d'implantation - Circuit FPGA Xilinx Virtex-6 xc6vlx75t-1

q	LUT	FF	Slices	BRAM	f (MHz)
4	533	539	610	1	293
6	1438	1725	1844	1	216
8	4283	6267	5731	1	172
10	15193	24833	23720	1	103

IP Core Xilinx :

LUT	FF	Slices	BRAM	f (MHz)
3712	4062	3765	13	285

	FER @ 1 dB	Slices (ratio)
TC	$1,5 \times 10^{-5}$	
q = 4	3×10^{-6}	16%
q = 6	$1,2 \times 10^{-6}$	49%
q = 10	$7,2 \times 10^{-7}$	630%



Plan

- ① Contexte des travaux de thèse
- ② Abaissement du plancher d'erreurs des turbo codes
- ③ Architecture matérielle de correction des erreurs résiduelles
- ④ Conclusion et perspectives

Conclusion

- ① Propositions de deux algorithmes améliorant les performances de décodage des turbo codes

Conclusion

- ① Propositions de deux algorithmes améliorant les performances de décodage des turbo codes
 - Self-Corrected : Sous certaines conditions, gain d'un ordre de grandeur dans la convergence

Conclusion

① Propositions de deux algorithmes améliorant les performances de décodage des turbo codes

- Self-Corrected : Sous certaines conditions, gain d'un ordre de grandeur dans la convergence

[JNRDM 2015]

[GRETSI 2015]

Conclusion

① Propositions de deux algorithmes améliorant les performances de décodage des turbo codes

- Self-Corrected : Sous certaines conditions, gain d'un ordre de grandeur dans la convergence
[JNRDM 2015]
[GRETSI 2015]
- Algorithme Flip And Check : Gain d'environ un ordre de grandeur dans la zone du plancher d'erreurs

Conclusion

① Propositions de deux algorithmes améliorant les performances de décodage des turbo codes

- Self-Corrected : Sous certaines conditions, gain d'un ordre de grandeur dans la convergence

[JNRDM 2015]

[GRETSI 2015]

- Algorithme Flip And Check : Gain d'environ un ordre de grandeur dans la zone du plancher d'erreurs

[IEEE Wireless Commun. Lett. 2016]

[ISTC 2016]

Conclusion

① Propositions de deux algorithmes améliorant les performances de décodage des turbo codes

- Self-Corrected : Sous certaines conditions, gain d'un ordre de grandeur dans la convergence

[JNRDM 2015]

[GRETSI 2015]

- Algorithme Flip And Check : Gain d'environ un ordre de grandeur dans la zone du plancher d'erreurs

[IEEE Wireless Commun. Lett. 2016]

[ISTC 2016]

② Architecture matérielle pour l'algorithme FNC

Conclusion

① Propositions de deux algorithmes améliorant les performances de décodage des turbo codes

- Self-Corrected : Sous certaines conditions, gain d'un ordre de grandeur dans la convergence

[JNRDM 2015]

[GRETSI 2015]

- Algorithme Flip And Check : Gain d'environ un ordre de grandeur dans la zone du plancher d'erreurs

[IEEE Wireless Commun. Lett. 2016]

[ISTC 2016]

② Architecture matérielle pour l'algorithme FNC

[DASIP 2016]

[GDR SoC-SiP 2017]

Perspectives

① Algorithmiques

② Architecturales

③ Industrielle

Perspectives

① Algorithmiques

- Ordonner la génération des mots candidats de façon plus optimisée
- Adaptation pour les codes LDPC

② Architecturales

③ Industrielle

Perspectives

① Algorithmiques

- Ordonner la génération des mots candidats de façon plus optimisée
- Adaptation pour les codes LDPC

② Architecturales

- Réduire la complexité des calculs de CRC
- Compatibilité avec d'autres ordonnancements de turbo codes binaires
- Support des turbo codes doubles binaires

③ Industrielle

Perspectives

① Algorithmiques

- Ordonner la génération des mots candidats de façon plus optimisée
- Adaptation pour les codes LDPC

② Architecturales

- Réduire la complexité des calculs de CRC
- Compatibilité avec d'autres ordonnancements de turbo codes binaires
- Support des turbo codes doubles binaires

③ Industrielle

- Valider les performances de décodage sur des modèles de canaux aéronautique/spatial

Merci

Merci pour votre attention !