Pre-sorted Forward-Backward NB-LDPC Check Node Architecture

Hassan Harb, Cédric Marchand, Ali Al Ghouwayel *, **Laura Conde-Canencia** and Emmanuel Boutillon

Lab-STICC Laboratory, Université de Bretagne Sud, Lorient, France
* Lebanese International University (LIU), Beirut, Lebanon

Contact : laura.conde-canencia@univ-ubs.fr
emmanuel.boutillon@univ-ubs.fr

IEEE International Workshop on Signal Processing Systems
October 26-28, 2016 - Dallas, Texas

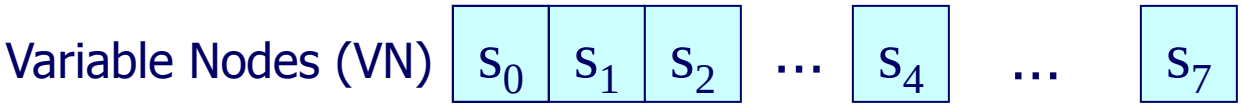
What is a NB–LDPC code?

Error-correcting code based on Galois Field of order q (extension of binary LPDC code).

What is a NB–LDPC code?

Error-correcting code based on Galois Field of order q (extension of binary LPDC code).

Codeword of N symbols of $log\ q$ bits

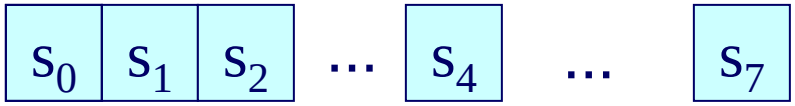


What is a NB–LDPC code?

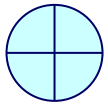
Error-correcting code based on Galois Field of order q (extension of binary LPDC code).

Codeword of N symbols of $log\ q$ bits

Variable Nodes (VN)

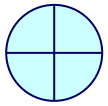


Check Nodes (CN)
Parity-check equations



CN1

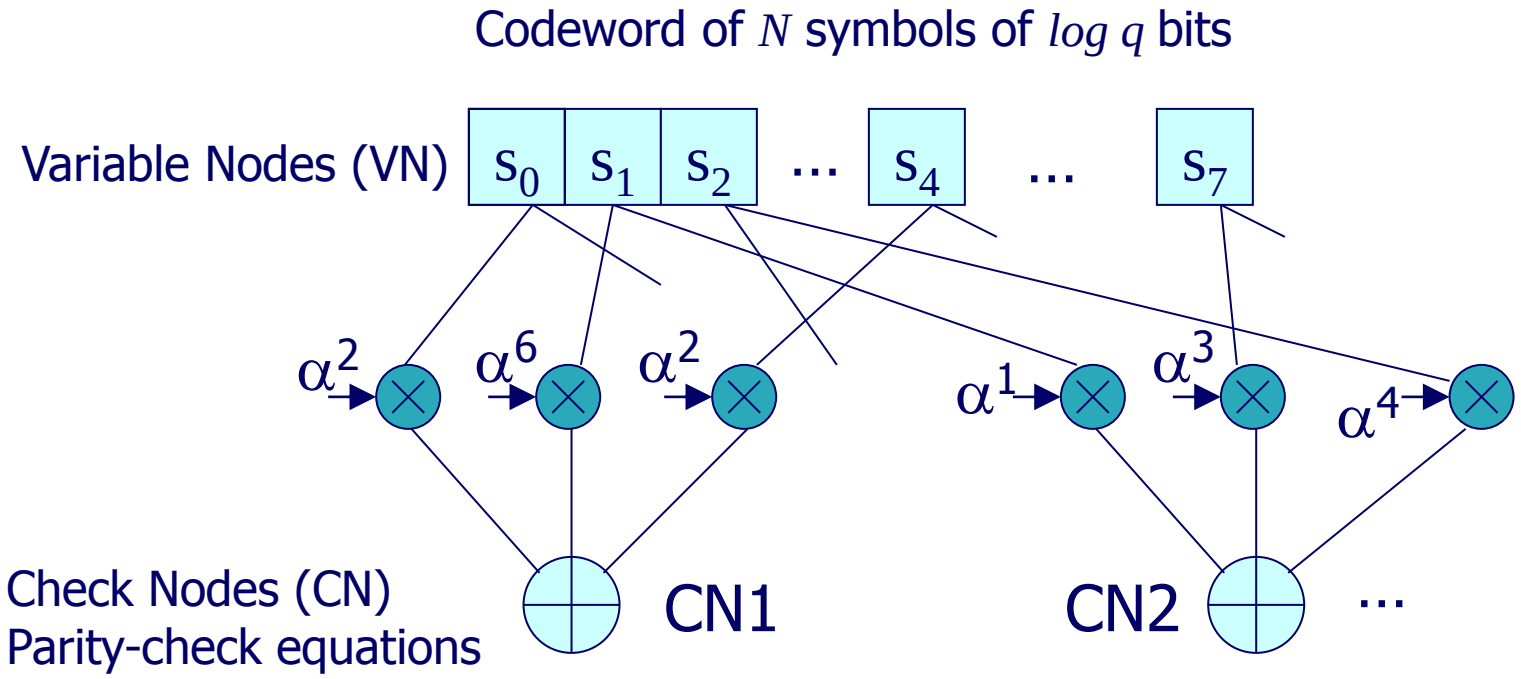
CN2



…

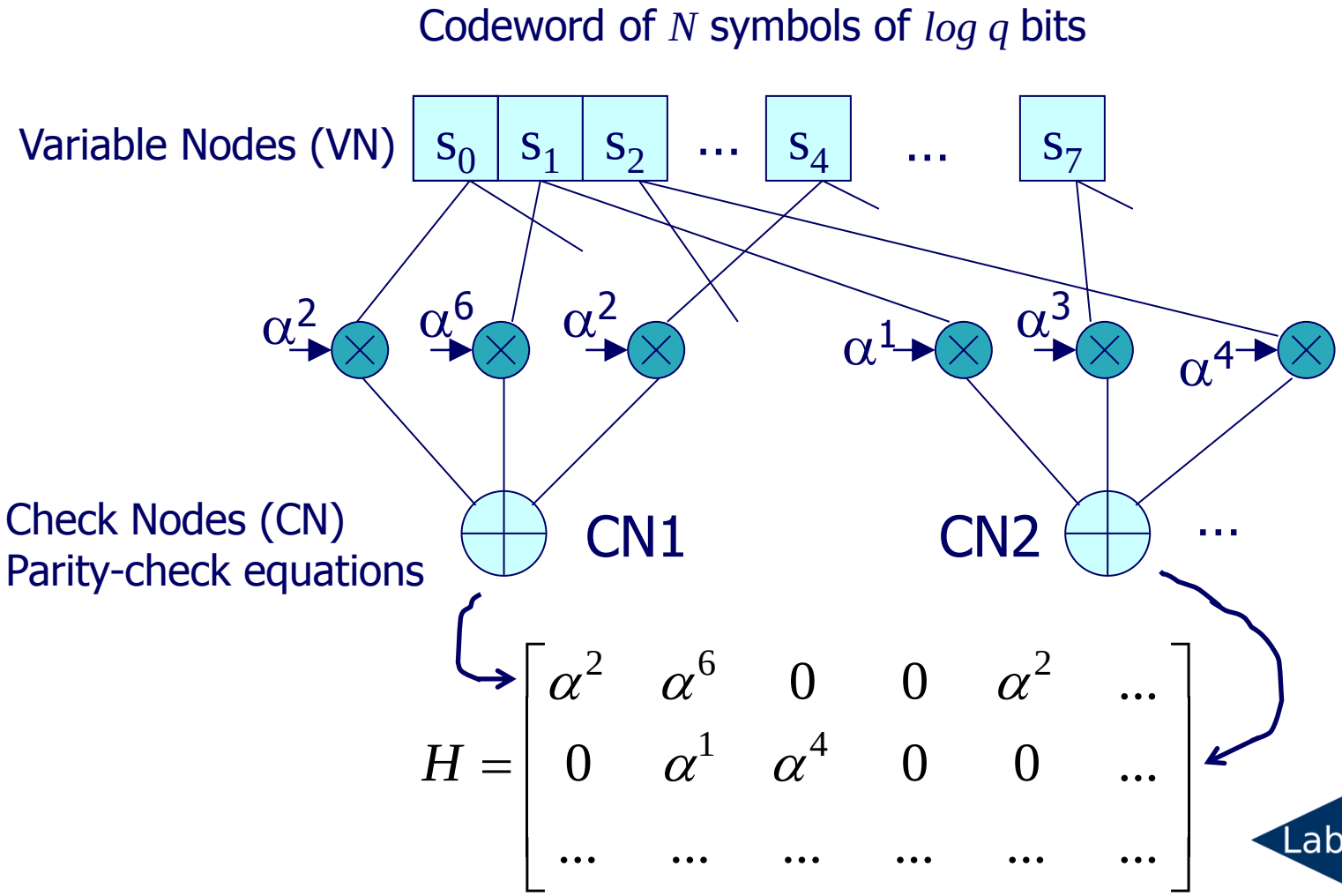
What is a NB-LDPC code?

Error-correcting code based on Galois Field of order q (extension of binary LPDC code).



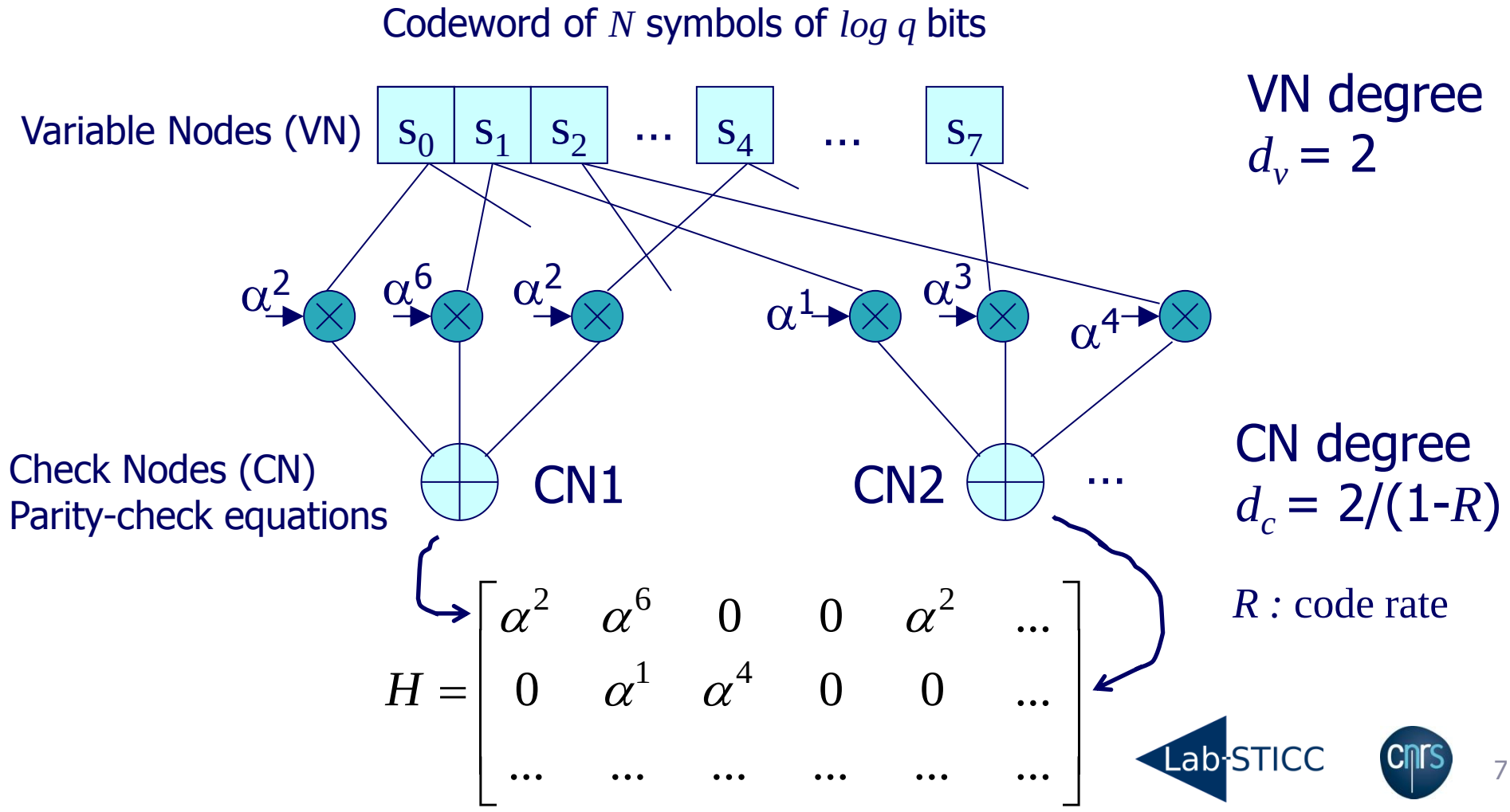
What is a NB-LDPC code?

Error-correcting code based on Galois Field of order q (extension of binary LPDC code).



What is a NB-LDPC code?

Error-correcting code based on Galois Field of order q (extension of binary LPDC code).



Quick review of NB-LDPC

Gallager 1963	LDPC codes in Galois Fields, iterative hard decoding for $d_v = 3$
MacKay 1998	Strong error correction capability for small blocks/[high rates], ultra-sparse $d_v = 2$, NB-LDPC codes defined on high-order fields
Since 2003 …	Development of practical decoders for NB-LDPC codes

ADVANTAGES

NB-LDPC codes retain benefits of:

- Steep waterfall region (as Turbo-codes)
- Low error floor (as binary LDPC)

Sparser matrix than binary LDPC ($d_v = 2$)

➡ better behavior of message-passing algorithms

Suitable for high-spectral-efficiency coded modulations

DRAWBACKS

Highly increased decoding complexity:

- Messages exchanged between processing nodes are arrays of q values …

Main NB–LDPC decoding algorithms

- Direct application of BP decoding is prohibitive in terms of complexity
- Log-BP, FFT-BP still too complex for practical implementation …

- Min-Max [1]
- **Extended Min-Sum (EMS)** [2]**:** messages truncated from size q to size n_m ($n_m < q$)
- Trellis-based EMS [3]
- Stochastic, symbol-flipping…

[1] V. Savin, "Min-max decoding for non binary LDPC codes," in Proc. IEEE Int. Symp. Information Theory, ISIT'2008. Toronto, Canada, July 2008.
[2] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," IEEE Trans. Comm., vol. 55, no. 4, pp. 633– 643, April 2007
[3] E. Li, D. Declercq, K. Gunnam, "Trellis-Based Extended Min-Sum Algorithm for Non-Binary LDPC Codes and its Hardware Structure", IEEE Trans. on Communications 61(7):2600-2611 · July 2013

Outline

I. **Forward-Backward (FB) implementation of EMS CN in the NB-LDPC decoder**
II. Pre-sorting technique and its application to EMS FB implementation
III. Implementation and simulation results
IV. Conclusion

Extended Min-Sum (EMS) Check Node (CN) processing

From the channel information, or the current state of the decoding algorithm, it is possible to express knowledge about the value of a symbol x.

Symbol $x \in GF(q)$	$P(x)$	$\log P(x)$	$-\log \frac{P(x)}{P(\bar{x})}$
α_0	0.042	-3.17	**3**
$\bar{x} = \alpha_1$	0.879	-0.13	**0**
α_2	0.0001	-9.86	10
α_3	0.0003	-8.10	8
α_4	0.0578	-2.85	**3**
α_5	0.0048	-5.35	5
α_6	0.0156	-4.16	**4**
α_7	0.00005	-10.27	10

$$\bar{x} = \arg\max_{x \in GF(q)} P(x) = \alpha_1$$

It is more convenient to use log value of $P(x)$

and to normalize and quantize to obtain Log Likelihood Ratio (LLR)

$$\mathrm{LLR}(x) = -\log \frac{\boldsymbol{P(x)}}{\boldsymbol{P(\bar{x})}}$$

Example for GF(8)

Highest values of LLR correspond to least probable symbols

Extended Min-Sum (EMS) Check Node (CN) processing

- Among the q LLR, select only the first n_m most reliable couples (in order).

Symbol $x \in GF(q)$	$P(x)$	$\log P(x)$	LLR(x)
α_0	0.042	-3.17	**3**
$\bar{x} = \alpha_1$	0.879	-0.13	**0**
α_2	0.0001	-9.86	10
α_3	0.0003	-8.10	8
α_4	0.0578	-2.85	**3**
α_5	0.0048	-5.35	5
α_6	0.0156	-4.16	**4**
α_7	0.00005	-10.27	10

- EMS representation

$$\boldsymbol{U(i) = (U^{+}(i), U^{\oplus}(i)), i = 0, \ldots, n_m - 1}$$

U^{+}	$U^{\oplus}$
0	α_1
3	α_0
3	α_4
4	α_6

$n_m = 4$, sorted in increasing order of LLR values

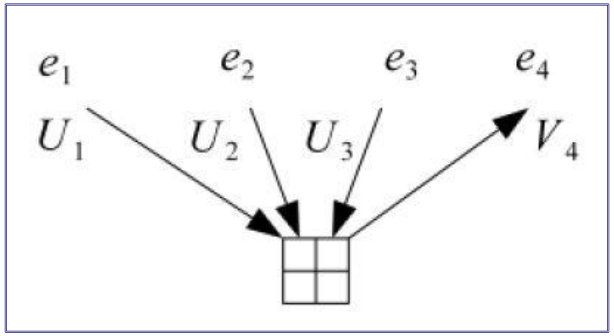
Example for GF(8)

Highest values of LLR correspond to least probable symbols

Extended Min–Sum (EMS) Check Node (CN) processing

CN equation $d_c = 4$

$$U_1 + U_2 + U_3 + U_4 = 0$$

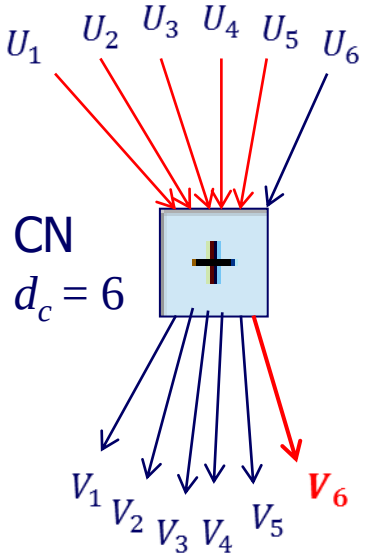


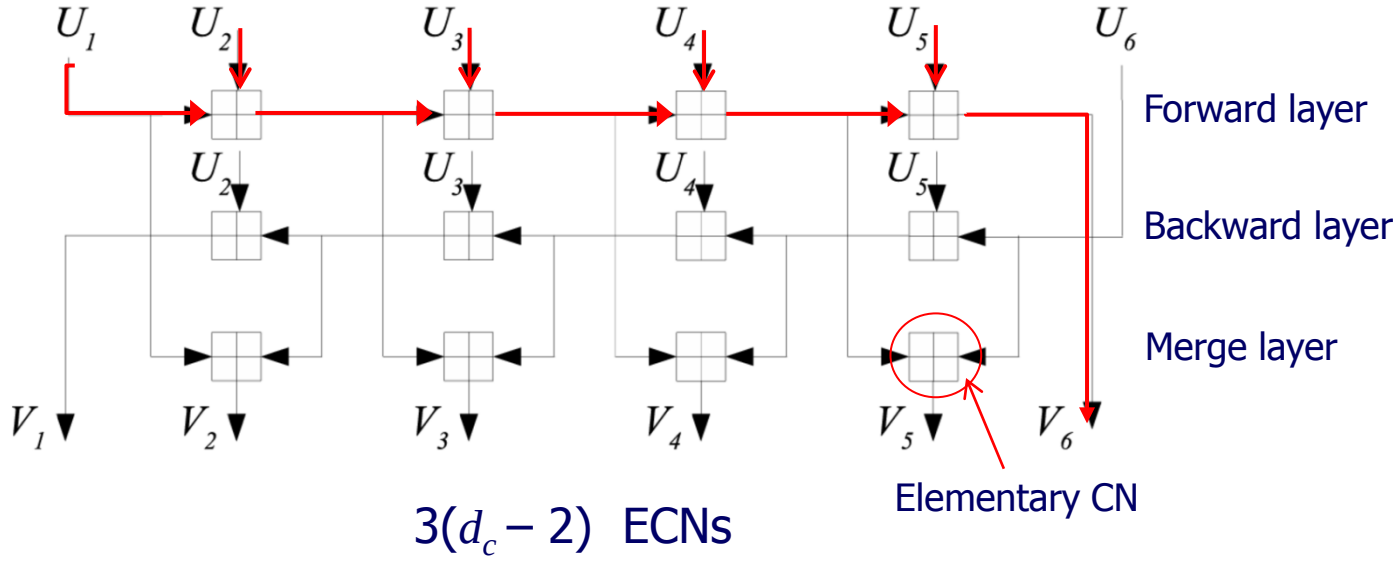
NB parity-check equation with d_c messages of n_m couples each

How to process all these to generate d_c output messages ?

➡ Forward-backward implementation with Elementary Check Nodes (ECN)

Forward–Backward (FB) implementation of EMS CN decoder

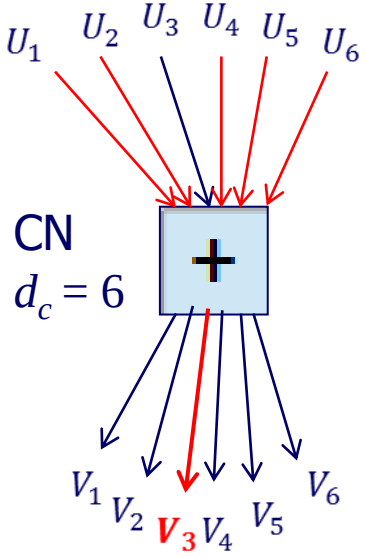


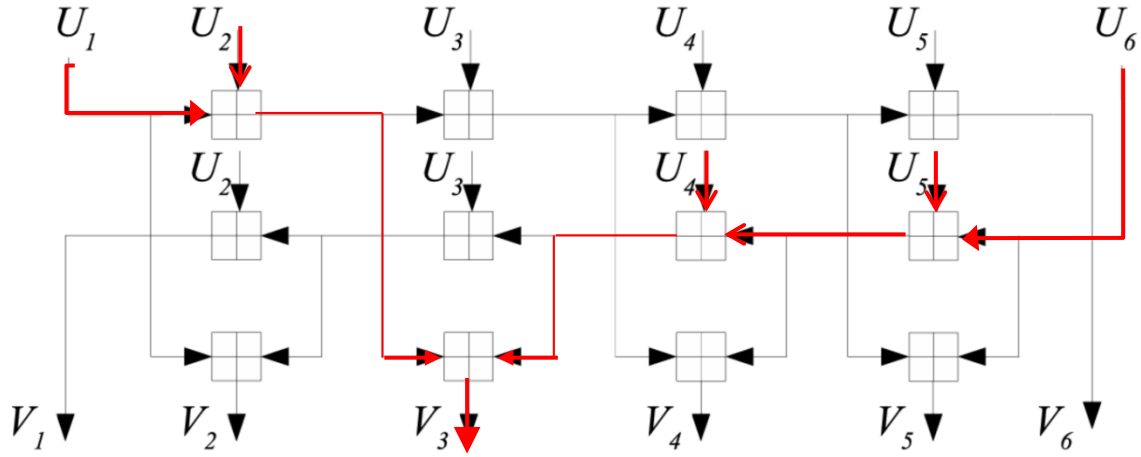


$$v_i^+(x) = \min \left\{ \sum_{i'=1, i' \neq i}^{d_c} U_{i'}^+[j_{i'}] \;\middle|\; \bigoplus_{i'=1, i' \neq i}^{d_c} U_{i'}^{\oplus}[j_{i'}] = x \right\}$$

➡ Reducing complexity at the ECN level to reduce the overall CN complexity

Forward–Backward (FB) implementation of EMS CN decoder





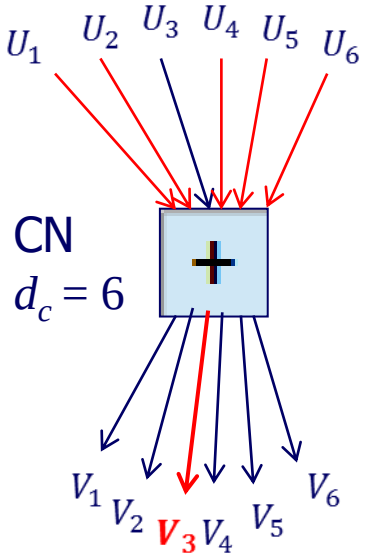
$3(d_c - 2)$ ECNs

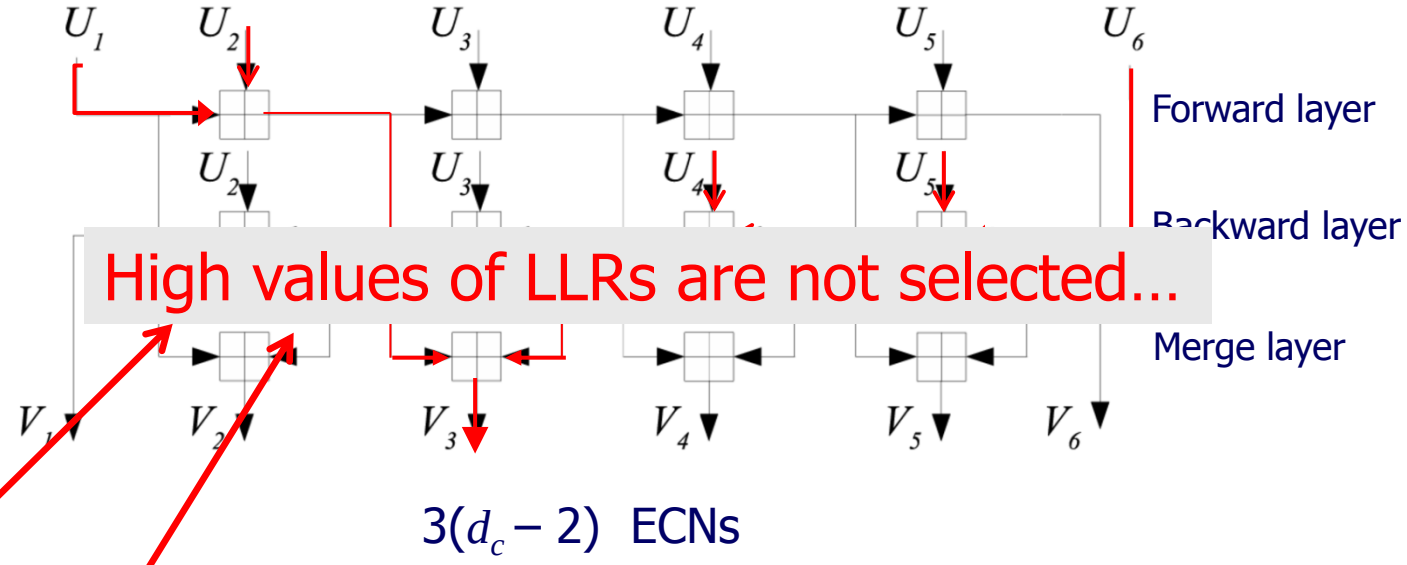
$$v_i^+(x) = \min \left\{ \sum_{i'=1, i' \neq i}^{d_c} U_{i'}^+[j_{i'}] \mid \bigoplus_{i'=1, i' \neq i}^{d_c} U_{i'}^{\oplus}[j_{i'}] = x \right\}$$

Reducing complexity at the ECN level to reduce the overall CN complexity

Lab-STICC cnrs

Forward–Backward (FB) implementation of EMS CN decoder





High values of LLRs are not selected…

$3(d_c - 2)$ ECNs

$$v_i^+(x) = \min\left\{ \sum_{i'=1, i' \neq i}^{d_c} U_{i'}^+[j_{i'}] \;\middle|\; \bigoplus_{i'=1, i' \neq i}^{d_c} U_{i'}^{\oplus}[j_{i'}] = x \right\}$$

Reducing complexity at the ECN level to reduce the overall CN complexity

Lab-STICC cnrs

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

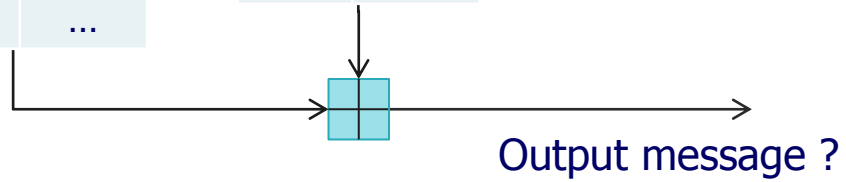
$$T(i,j) = U_1^+[i] + U_2^+[j]$$

U_1

LLR	GF
0	α_0
2	α_{57}
11	α_4
12	α_{22}
16	α_5
…	…

U_2

LLR	GF
0	α_{16}
7	α_4
10	α_{23}
11	α_3
19	α_{51}
…	…

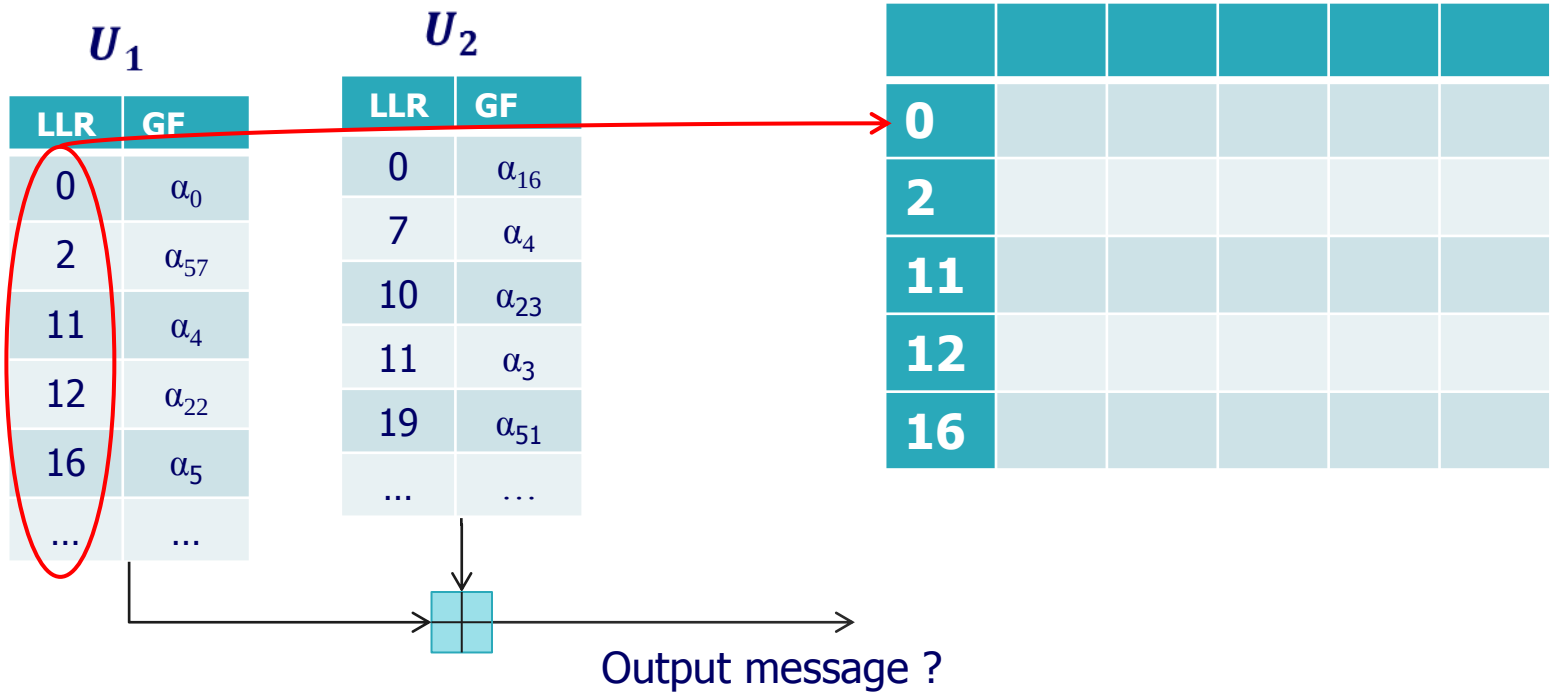


GF(64), n_m = 14

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

$$T^{+}(i,j) = U_1^{+}[i] + U_2^{+}[j]$$

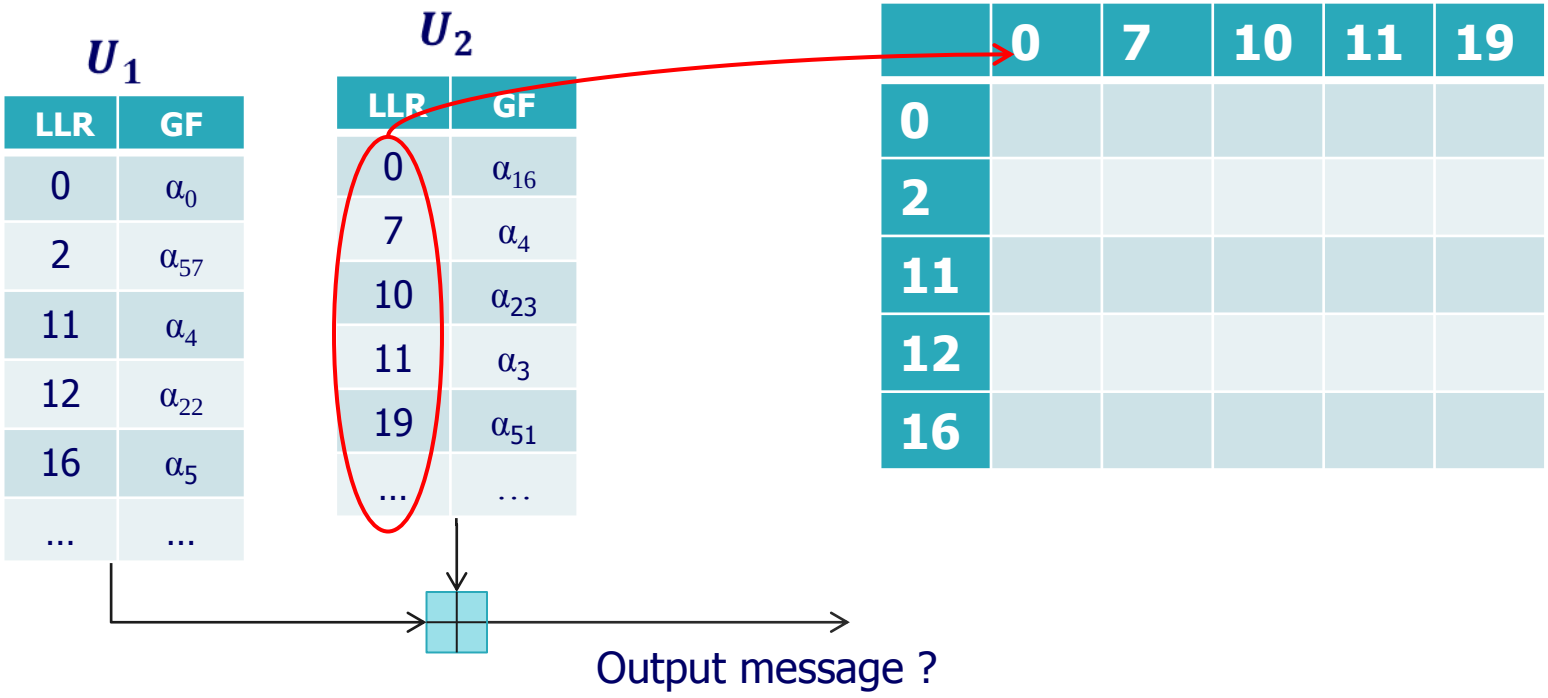


GF(64), n_m = 14

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

$$T(i,j) = U_1^+[i] + U_2^+[j]$$



GF(64), n_m = 14

Lab-STICC cnrs

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

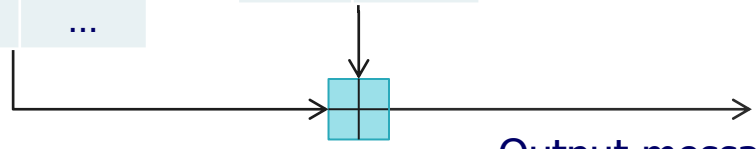
$$T(i,j) = U_1^+[i] + U_2^+[j]$$

U_1

LLR	GF
0	α_0
2	α_{57}
11	α_4
12	α_{22}
16	α_5
...	...

U_2

LLR	GF
0	α_{16}
7	α_4
10	α_{23}
11	α_3
19	α_{51}
...	...



Output message ?

	0	7	10	11	19
0	0	7	10	11	19
2	2	9	12	13	21
11	11	18	21	22	30
12	12	19	22	23	31
16	16	23	26	27	35

GF(64), n_m = 14

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

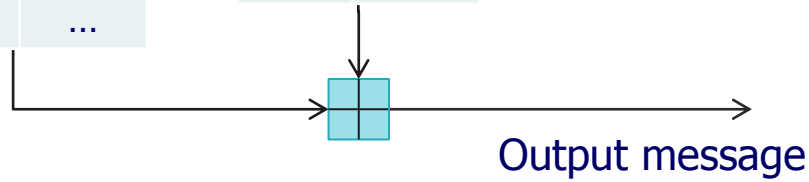
$$T(i,j) = U_1^+[i] + U_2^+[j]$$

U_1

LLR	GF
0	α_0
2	α_{57}
11	α_4
12	α_{22}
16	α_5
…	…

U_2

LLR	GF
0	α_{16}
7	α_4
10	α_{23}
11	α_3
19	α_{51}
…	…



	0	7	10	11	19
0	0	7	10	11	19
2	2	9	12	13	21
11	11	18	21	22	30
12	12	19	22	23	31
16	16	23	26	27	35

$n_m = 14$

GF(64), n_m = 14

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

$$T(i,j) = U_1^+[i] + U_2^+[j]$$

U_1

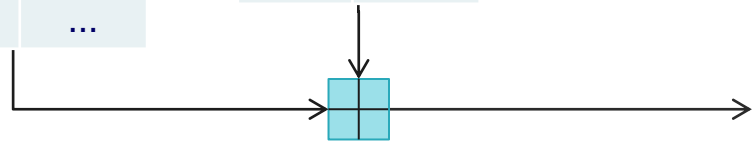
LLR	GF
0	α_0
2	α_{57}
11	α_4
12	α_{22}
16	α_5
…	…

U_2

LLR	GF
0	α_{16}
7	α_4
10	α_{23}
11	α_3
19	α_{51}
…	…

	0	7	10	11	19
0	0	7	10	11	19
2	2	9	12	13	21
11	11	18	21	22	30
12	12	19	22	23	31
16	16	23	26	27	35

$n_m = 14$



$(0, \alpha_0 \oplus \alpha_{16})$

GF(64), $n_m = 14$

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

$$T(i,j) = U_1^+[i] + U_2^+[j]$$

U_1

LLR	GF
0	α_0
2	α_{57}
11	α_4
12	α_{22}
16	α_5
…	…

U_2

LLR	GF
0	α_{16}
7	α_4
10	α_{23}
11	α_3
19	α_{51}
…	…

	0	7	10	11	19
0	0	7	10	11	19
2	2	9	12	13	21
11	11	18	21	22	30
12	12	19	22	23	31
16	16	23	26	27	35

$n_m = 14$

$(2, \alpha_{57} \oplus \alpha_{16})$ $(0, \alpha_0 \oplus \alpha_{16})$

GF(64), $n_m = 14$

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

$$T(i,j) = U_1^+[i] + U_2^+[j]$$

U_1

LLR	GF
0	α_0
2	α_{57}
11	α_4
12	α_{22}
16	α_5
…	…

U_2

LLR	GF
0	α_{16}
7	α_4
10	α_{23}
11	α_3
19	α_{51}
…	…

	0	7	10	11	19
0	0	7	10	11	19
2	2	9	12	13	21
11	11	18	21	22	30
12	12	19	22	23	31
16	16	23	26	27	35

$n_m = 14$

$(7, \alpha_0 \oplus \alpha_4)$ $(2, \alpha_{57} \oplus \alpha_{16})$ $(0, \alpha_0 \oplus \alpha_{16})$

GF(64), $n_m = 14$

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

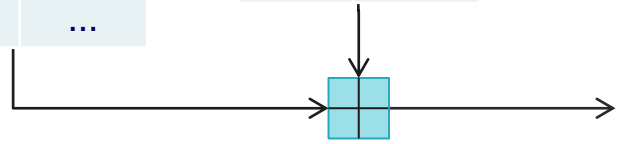
$$T(i,j) = U_1^+[i] + U_2^+[j]$$

U_1

LLR	GF
0	α_0
2	α_{57}
11	α_4
12	α_{22}
16	α_5
…	…

U_2

LLR	GF
0	α_{16}
7	α_4
10	α_{23}
11	α_3
19	α_{51}
…	…



	0	7	10	11	19
0	0	7	10	11	19
2	2	9	12	13	21
11	11	18	21	22	30
12	12	19	22	23	31
16	16	23	26	27	35

$n_m = 14$

$(9, \alpha_{57} \oplus \alpha_4)$ $(7, \alpha_0 \oplus \alpha_4)$ $(2, \alpha_{57} \oplus \alpha_{16})$ $(0, \alpha_0 \oplus \alpha_{16})$

GF(64), $n_m = 14$

Elementary CN: how it works ?

- Output message is generated from the n_m smallest LLR values in matrix

$$T(i,j) = U_1^+[i] + U_2^+[j]$$

U_1

LLR	GF
0	α_0
2	α_{57}
11	α_4
12	α_{22}
16	α_5
…	…

U_2

LLR	GF
0	α_{16}
7	α_4
10	α_{23}
11	α_3
19	α_{51}
…	…

U_2

U_1	0	7	10	11	19
0	0	7	10	11	19
2	2	9	12	13	21
11	11	18	21	22	30
12	12	19	22	23	31
16	16	23	26	27	35

$n_m = 14$

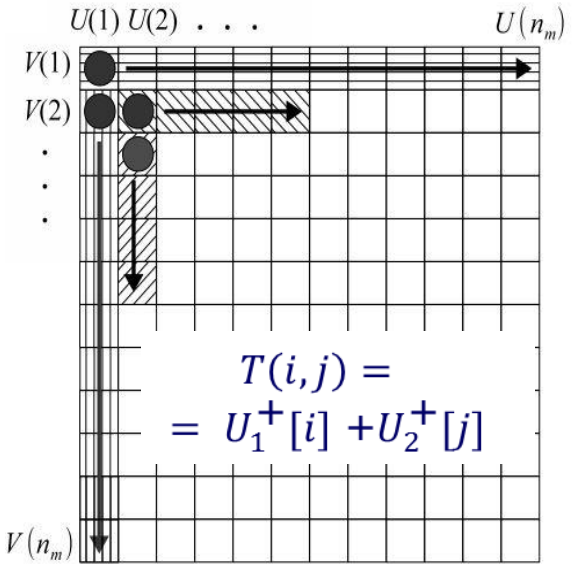
$(18, \alpha_4 \oplus \alpha_4)$ … $(7, \alpha_0 \oplus \alpha_4)$ $(2, \alpha_{57} \oplus \alpha_{16})$ $(0, \alpha_0 \oplus \alpha_{16})$

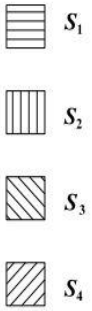
$n_m = 14$

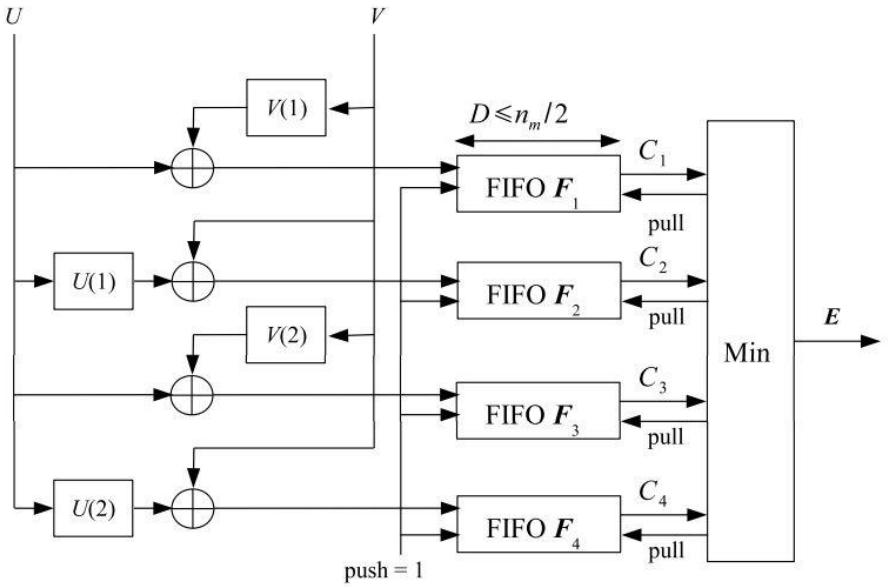
GF(64), $n_m = 14$

ECN: reduced-complexity implementation

- Bubble-Check [1]
- L-Bubble Check [2]
- **S-Bubble Check** [3]: only ~ $3.n_m$ positions (among ${n_m}^2$) are considered







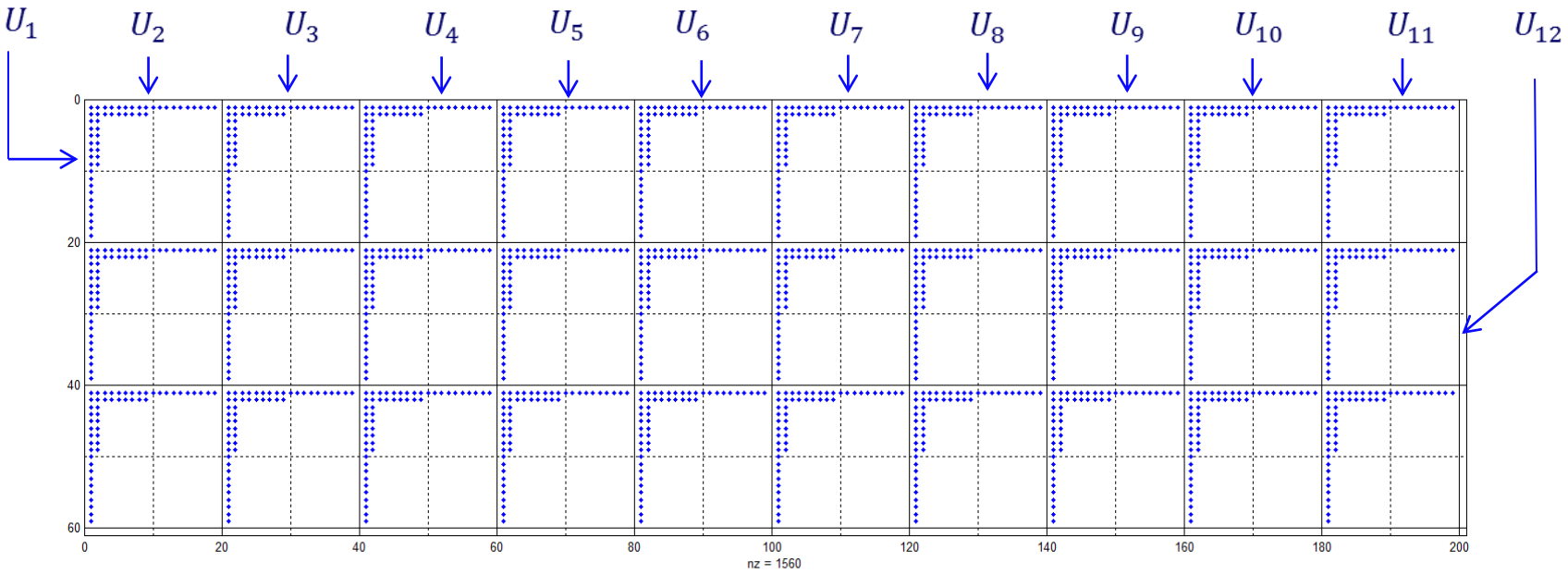
[1] Boutillon, E., Conde-Canencia, L., "Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders" Electronics Letters, Vol. 46, no.9, pp.633-634, April 29 2010.

[2] Boutillon, E., Conde-Canencia, L., Al Ghouwayel, A." Design of a GF(64)-LDPC Decoder Based on the EMS Algorithm" IEEE Transactions on Circuits and Systems I, vol.60, no.10, pp.2644-2656, Oct. 2013

[3] O. Abassi, L. Conde-Canencia, A. Al Ghouwayel, E. Boutillon "A Novel Architecture For Elementary Check Node Processing In Non-Binary LDPC Decoders" in IEEE Transactions on Circuits and Systems II: Express Briefs , vol.PP, no.99, April 2016

Forward–Backward CN with simplified ECNs

- S-Bubble architecture for $d_c = 12$



Points represent the selected values in each ECN

But we can still further simplify thanks to the **pre-sorting technique** …

Outline

I. Forward-Backward (FB) implementation of EMS CN in the NB-LDPC decoder

II. Pre-sorting technique and its application to EMS FB implementation

III. Implementation and simulation results

IV. Conclusion

Pre-sorting technique and its application to EMS FB implementation

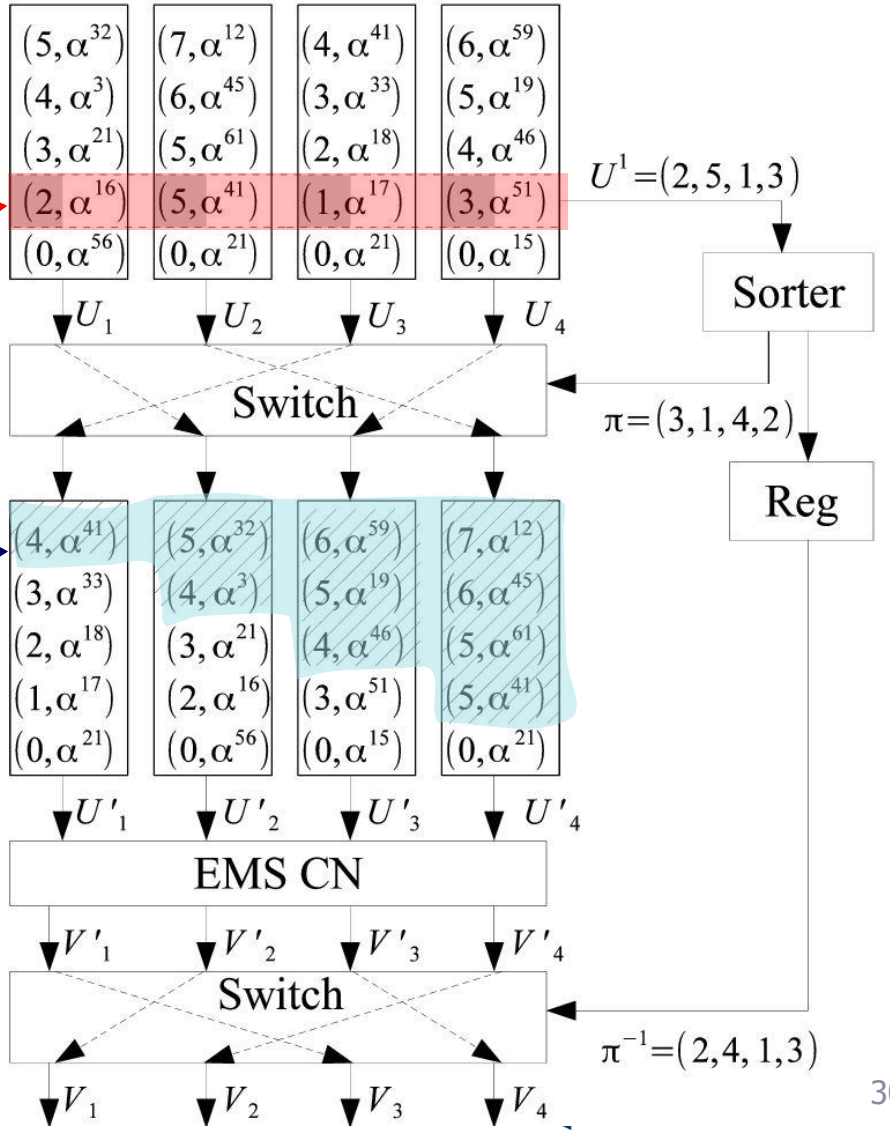
Pre-sorting: sort input messages to further favorise ECN simplification…

as a function of the second LLR values, because they mainly determine the reliability of the first symbol

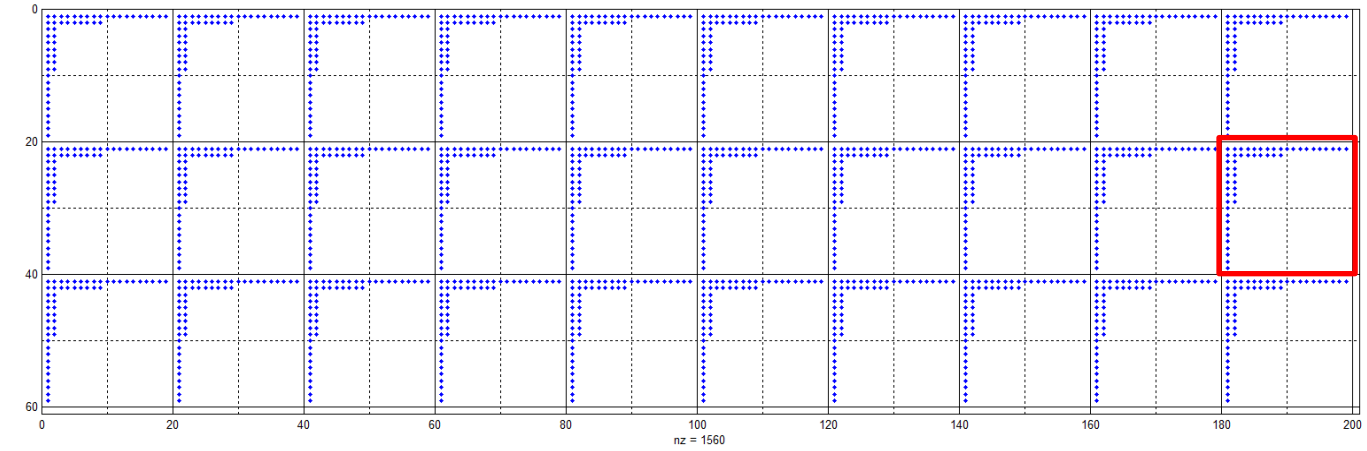
Blue values not considered in CN processing as they will not contribute to output messages

➡ right part of the CN FB implementation can be simplified

New blocks in architecture:
- 2 switches
- 1 sorter



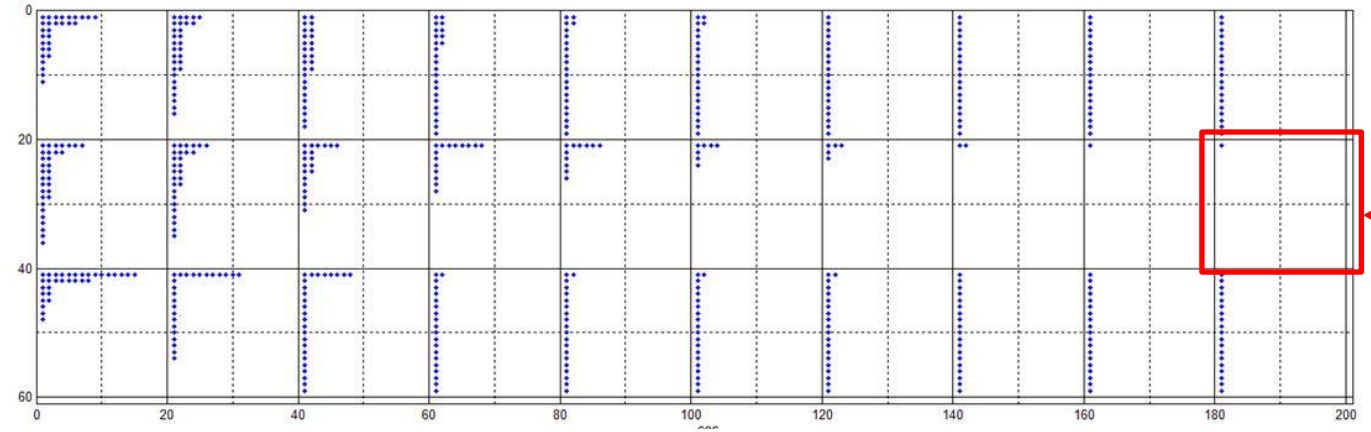
How pre-sorting can simplify the architecture ?



Without pre-sorting

S-Bubble architecture
(4 FIFOs + control)

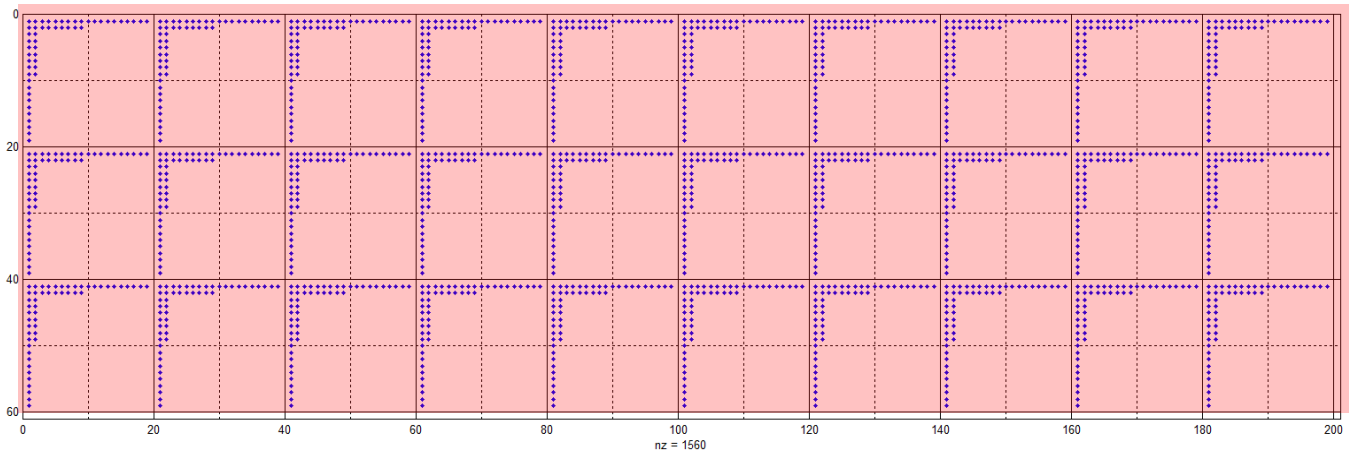
Example of ECN simplification



1 XOR architecture

With pre-sorting

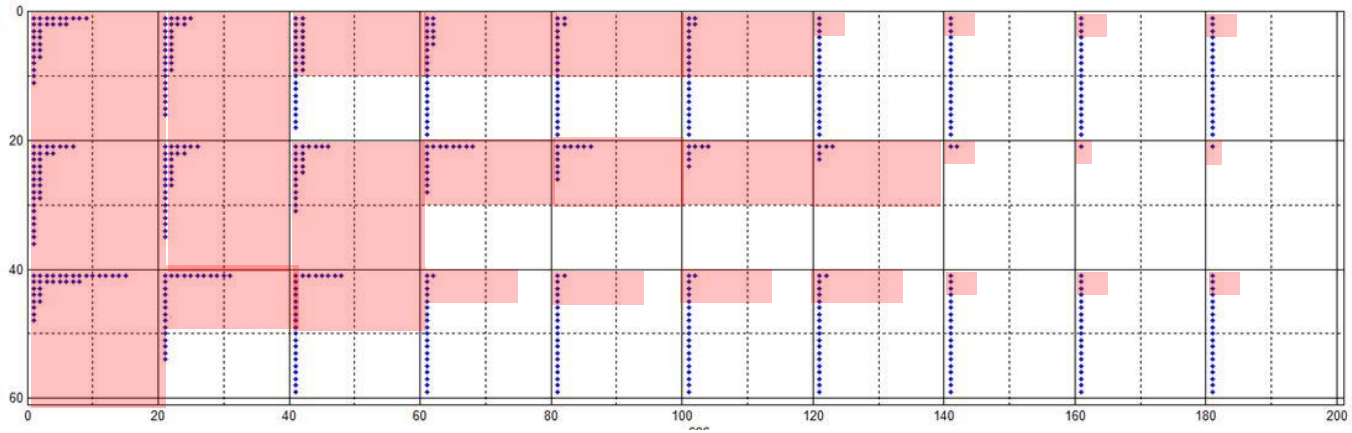
How pre-sorting can simplify the architecture ?



Without pre-sorting

~ 4100 slices
FPGA Xilinx Virtex 6

$\mathrm{GF}(64),\ n_m = 20$



With pre-sorting

~ 1900 slices
FPGA Xilinx Virtex 6

Outline

I. Forward-Backward (FB) implementation of EMS CN in the NB-LDPC decoder
II. Pre-sorting technique and its application to EMS FB implementation
III. **Implementation and simulation results**
IV. Conclusion

Application to serial architecture: implementation and simulation results

Number of occupied slices in Xilinx Virtex 6 FPGA device

d_c		Sorter	Switch	CN	Total	Gain
8	FB	-	-	2481	2481	17%
	P-FB	77	142	1701	2061	
12	FB	-	-	4666	4666	43%
	P-FB	160	283	1858	2653	
20	FB	-	-	6519	6519	54%
	P-FB	386	495	1232	2955	

FB:
Forward Backward with S-bubble ECN

P-FB:
Pre-sorting + Forward Backward + Simplified ECN

- Cons: Additionnal sorter and switch blocks
- Pros: Area reduction of the core Check Node
- Conclusion: significant complexity reduction without performance loss.

Outline

I. Forward-Backward (FB) implementation of EMS CN in the NB-LDPC decoder
II. Pre-sorting technique and its application to EMS FB implementation
III. Implementation and simulation results
IV. Conclusion

Conclusion

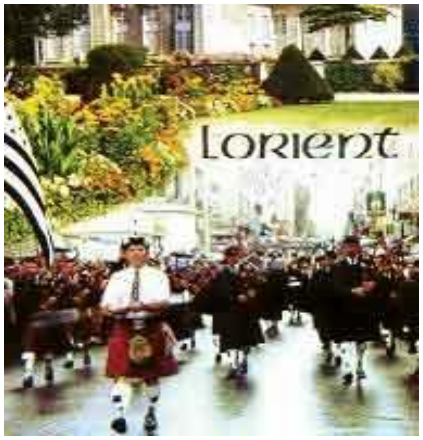
- Reduced-complexity Check Node implementation of NB-LDPC decoders based on the Extended Min-Sum

- Application of the **pre-sorting** technique to CN EMS Forward-Backward implementation:
  - Leads to significant complexity reduction
  - In serial FPGA implementation of EMS CN: significant area reduction (up to 54% for $d_c = 20$) without performance loss

- **Future work:** global decoder implementation (new version)

- Visit "NB-LDPC codes" webpage: http://www-labsticc.univ-ubs.fr/nb_ldpc/
  Source code available under request

Thank you. Questions ?



Contact: Prof. Laura Conde-Canencia
Email: **condecan@univ-ubs.fr**
Currently looking for a visiting professor position in the **USA** (1 year from Sept-Oct 2017)

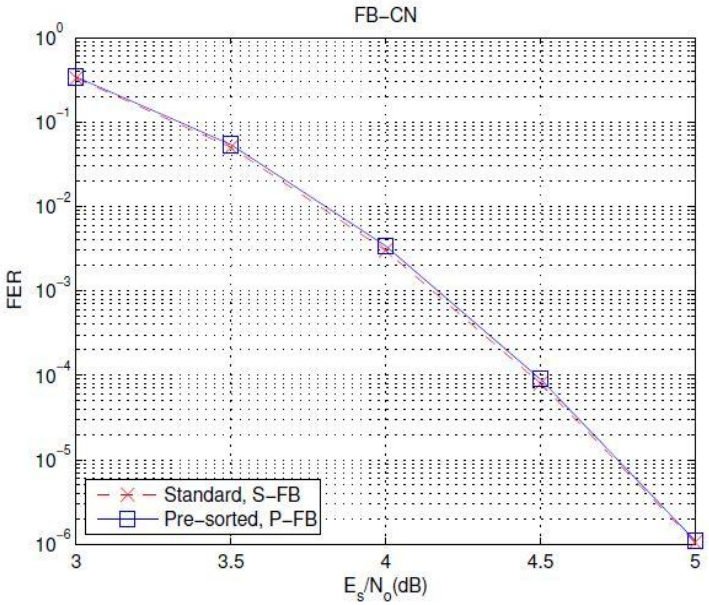
Welcome to SIPS'2017 in **Lorient, France**

Application to serial architecture: implementation and simulation results

S-FB: S-bubble architecture

P-FB: with pre-sorting, simplified architecture

FB-CN		Nb. of occupied slices				
d_c	Case	Sorter	Switch	CN	Total	Gain(%)
6	S-FB	0	0	1,617	1,617	5 %
	P-FB	50	93	1,268	1,532	
8	S-FB	0	0	2,481	2,481	17 %
	P-FB	77	142	1,701	2,061	
12	S-FB	0	0	4,666	4,666	43 %
	P-FB	160	283	1,858	2,653	
20	S-FB	0	0	6,519	6,519	54 %
	P-FB	386	495	1,232	2,955	



Simulation results GF-(64)-LDPC
N=576 bits, d_v = 2, d_c = 12, R = 5/6

- Additionnal sorter and switch blocks in P-FB architecture
- Significant area reduction at the CN level in P-FB architecture
- Global area gain, especially for high d_c orders
- No performance loss

Extended Min-Sum (EMS) Check Node (CN) processing

The reliability of the first value is strongly determined by the second LLR value $U[1]$:

$$P(U[0]^{\oplus}) \leq \frac{1}{1+e^{-U[1]}}$$

$$P(U[i]^{\oplus}) \leq \frac{e^{-U[i]}}{1+e^{-U[1]}}$$

$$0,7416 \leq 0,8791 \leq 0,9526$$

$$\mathrm{LLR}(x) = -\log \frac{P(x)}{P(\bar{x})}$$

- EMS: only the n_m most reliable couples

LLR value	GF(8) symbol
$U[0] = 0$	α_7
$U[1] = 3$	α_6
$U[2] = 3$	α_4
$U[3] = 4$	α_2

$n_m = 4$, sorted in increasing order of LLR values