Good LDPC Decoders with Exchanged Messages of size 1, 2 or 3 bits

Emmanuel Boutillon*, Fakhreddine Ghaffari†, Khoa Le† and Franklin Cochachin*†

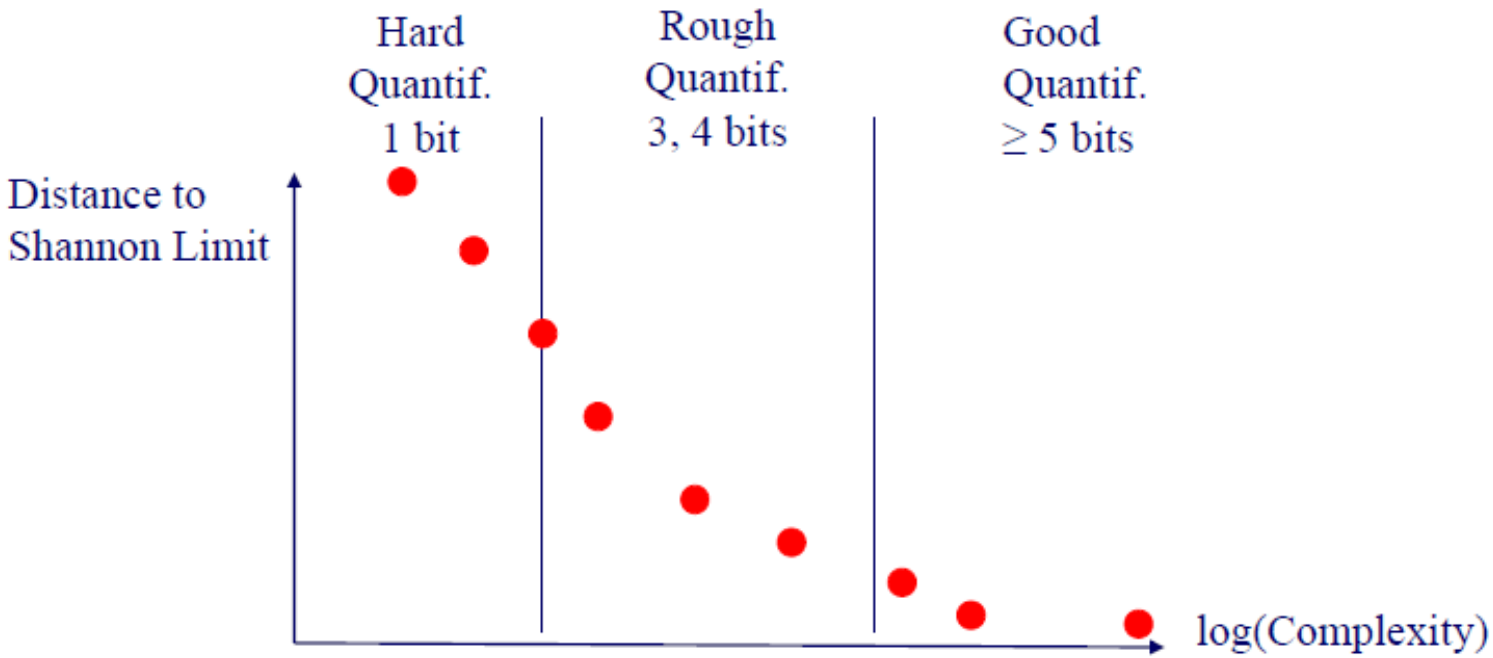
in collaboration with Chris Winstead

*Lab-STICC, UMR 6285, Université de Bretagne Sud, Centre de Recherche BP 92116, Lorient 56321, France
†ETIS, ENSEA/University of Cergy-Pontoise/CNRS, 95014 Cergy-Pontoise, France

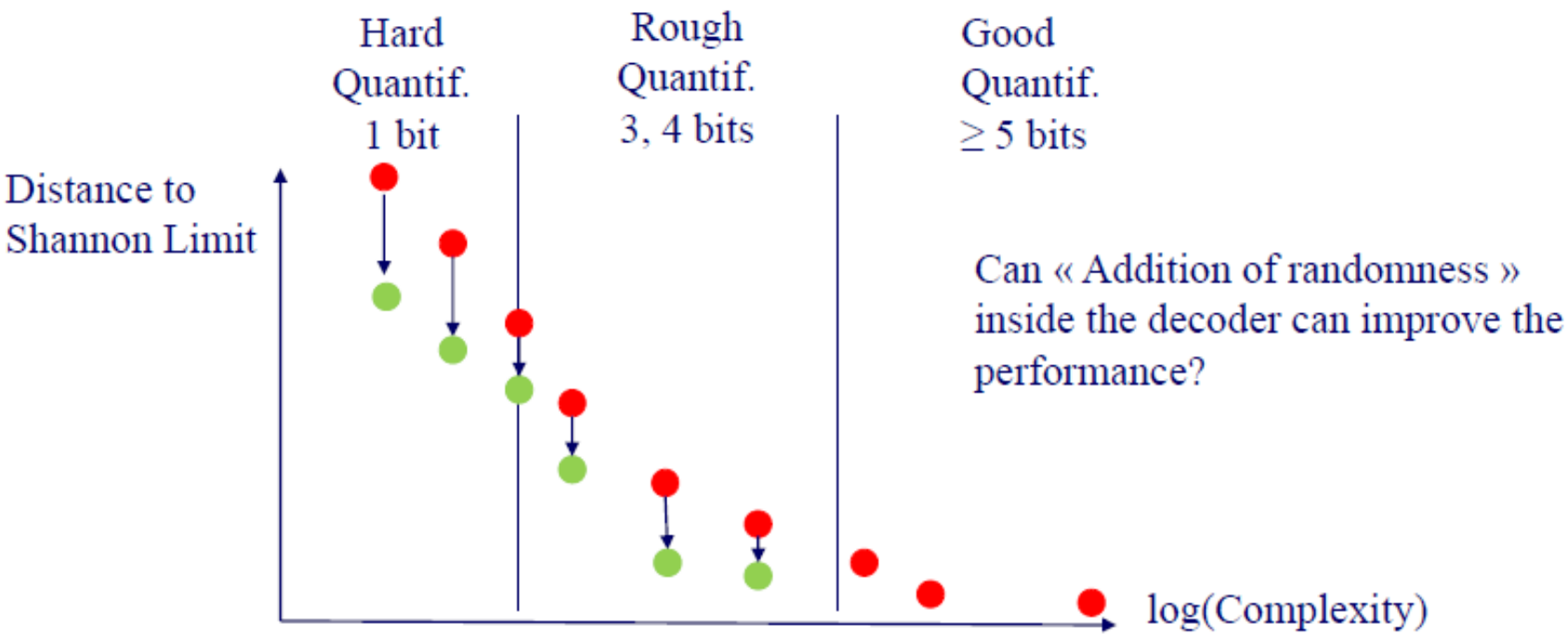
NAND Project: Funded by ANR, grant number ANR-15-CE25-0006-01

September 19, 2019

Objective: toward 1 Tbit/s decoders



Objective: toward 1 Tbit/s decoders



Noise Against Noise Decoder

$01/2015 \longrightarrow 09/2019$

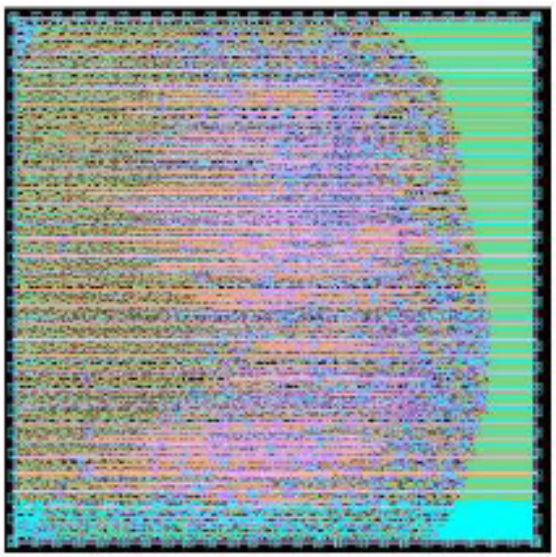
Lab-STICC, IMS, ETIS, CEA, Turbo-Concept, Thales Communications

- New algorithms:
  PPBF : Probabilistic Parallel Bit Flipping.
  SBF: Syndrome Bit Flipping.
  SP-MS: Sign-Preserving Min-Sum.
- Contribution to AFF3CT (https://github.com/aff3ct/aff3ct)
- Importance Sampling with hardware emulation (FER $< 8 \times 10^{-11}$ in 10 hours)

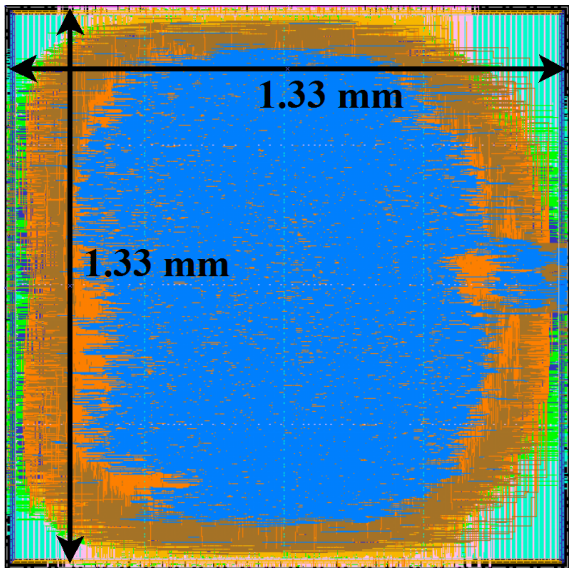
http://www-labsticc.univ-ubs.fr/ boutillon/NAND/index.html

Outline of the Presentation

	Input Quantization	
Exchanged Message	$q_{ch} = 1$ bit	$q_{ch} \geq 3$ bits
$q = 1$ bit	PPGDBF	
$q \geq 2$ bits		SP-MS



(a) PPBF, Fakhreddine Ghaffari



(b) SP-MS, Franklin Cochachin

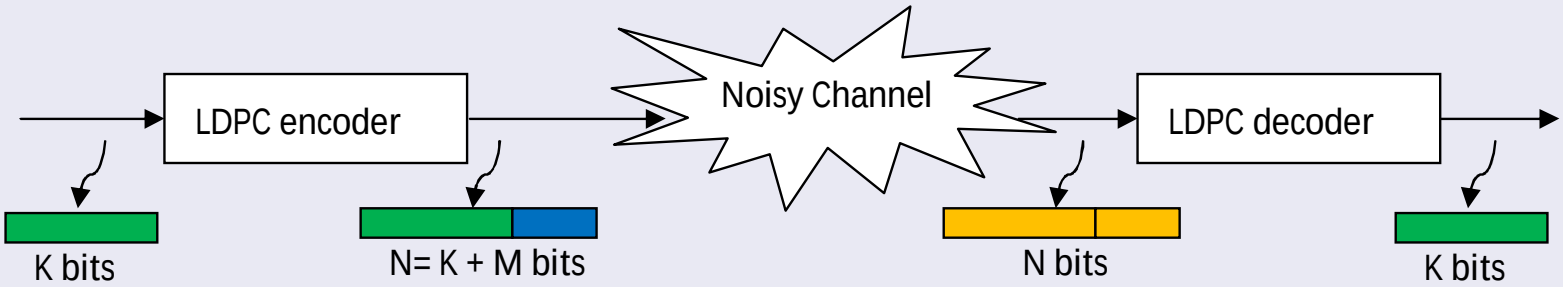
Outline

Outline

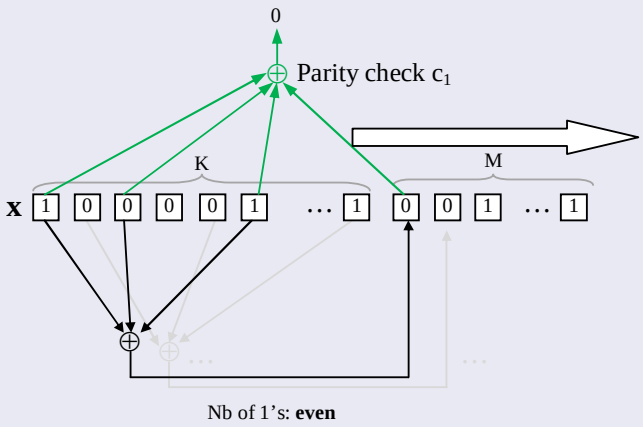
Context

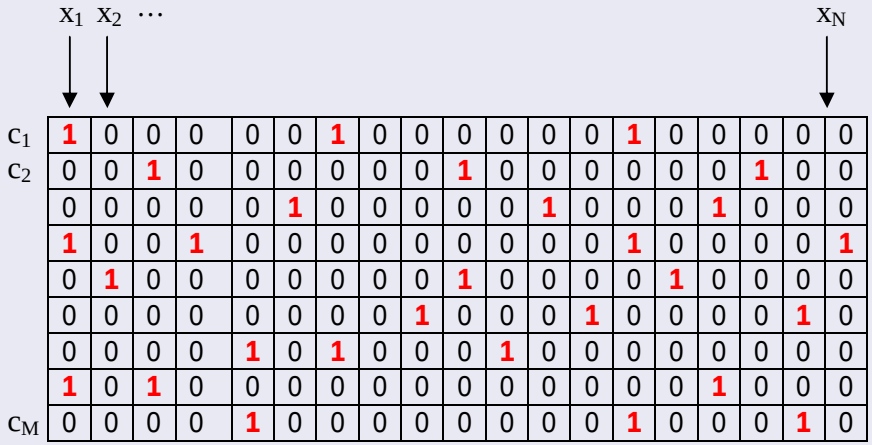
Low Density Parity Check (LDPC) Decoders

- Applications: Wired, Wireless communication, Storage...



2 types of LDPC decoding algorithms:

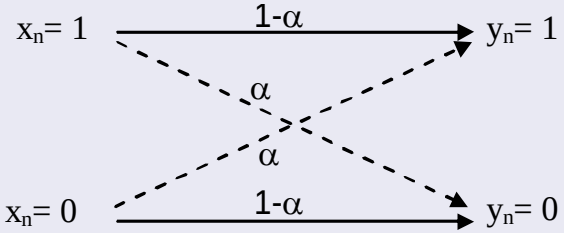




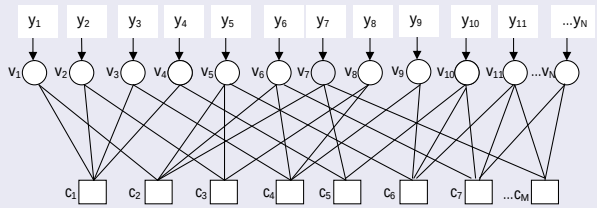
$\mathbf{c} = (c_1, c_2, c_3, \ldots, c_M) = \mathbf{0}$

Context

- $\mathbf{x} = \{x_1, x_2, .., x_N\} = \{0, 1\}^N$ is a codeword if $\mathbf{c} = Hx^T = \mathbf{0}$
- $\mathbf{y} = \{y_1, y_2, .., y_N\} = \{0, 1\}^N$ is the received codeword after $\mathbf{x}$ is transmitted through a BSC channel.



- A LDPC is represented by bipartite graph called Tanner graph with 2 groups of nodes
  - Variable Node (v_n) $1 \leq n \leq N$
  - Check Node (c_m) $1 \leq m \leq M$



2 types of LDPC decoding algorithms:

- Soft information algorithms: Sum-Product, Min-Sum..., powerful error correction capability but high complexity.
- Hard-decision algorithms: Gallager Bit Flipping (BF), Gradient Descent Bit Flipping (GDBF), Probabilistic GDBF..., low complexity, usually weak in error correction.

Context

- BF evolution:

BF 1963

WBF 2001

MWBF 2002

PBF, IMWBF 2005

PWBF 2007

IPBF 2008

GDBF 2010

Multi-bit-GDBF, RRWGDBF 2012

AT-GDBF, WCBBF 2013

TB-BF, NGDBF, **PGDBF**, MM-WBF 2014

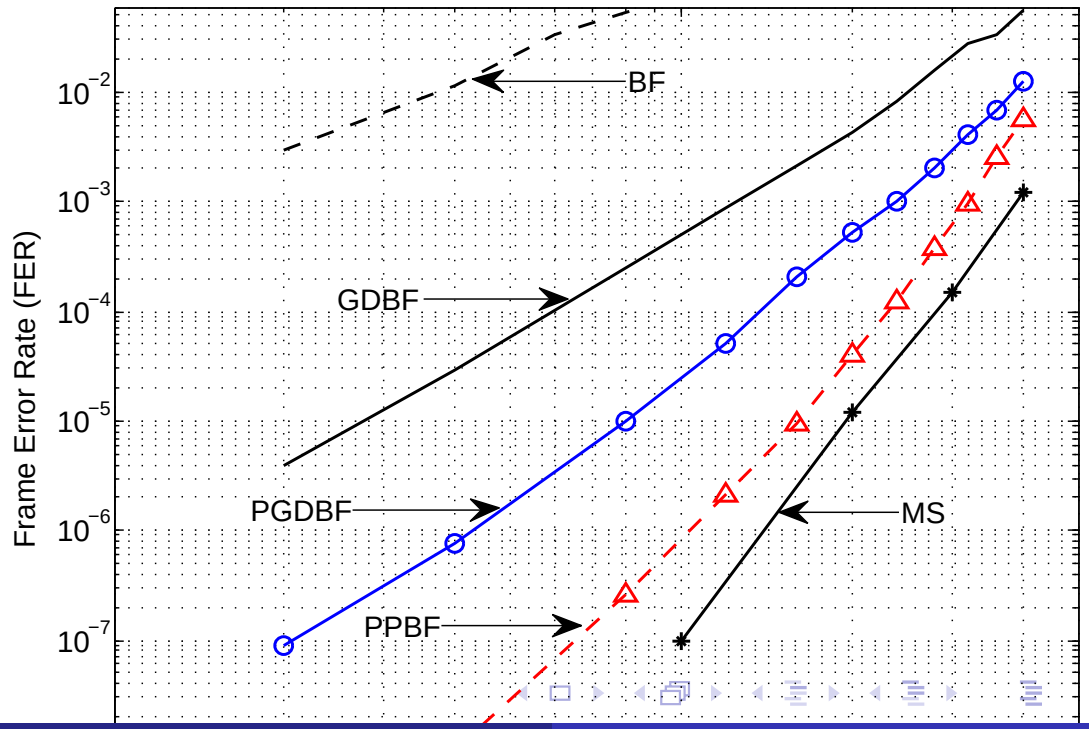
MTBF, DWBF, RECWBF, TSWBF 2015

HGDBF 2016

MBF, **PPBF** 2017

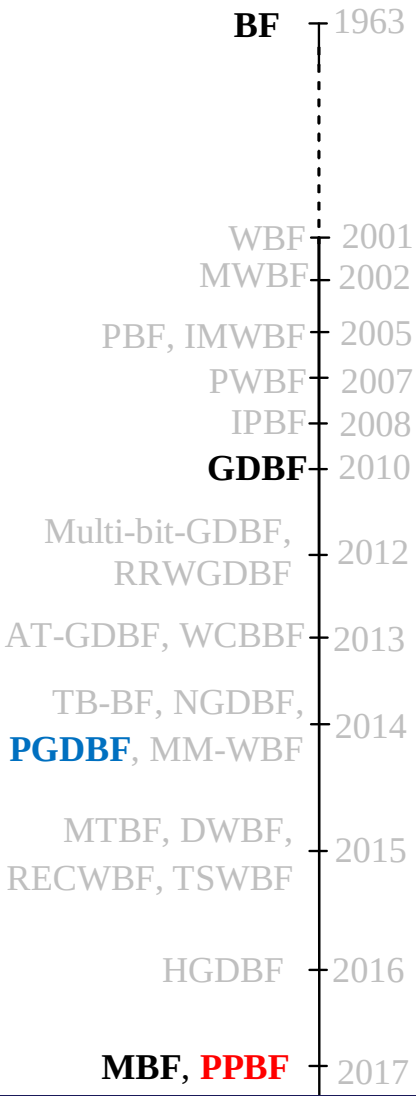
$$v_n^{(k+1)} = \text{flip}\left(v_n^{(k)}\right) \text{ if:}$$

- BF: $E_n^{(k)} = \sum_{m \in \mathcal{N}(n)} c_m^{(k)} \geq \tau$; τ: predefined threshold.



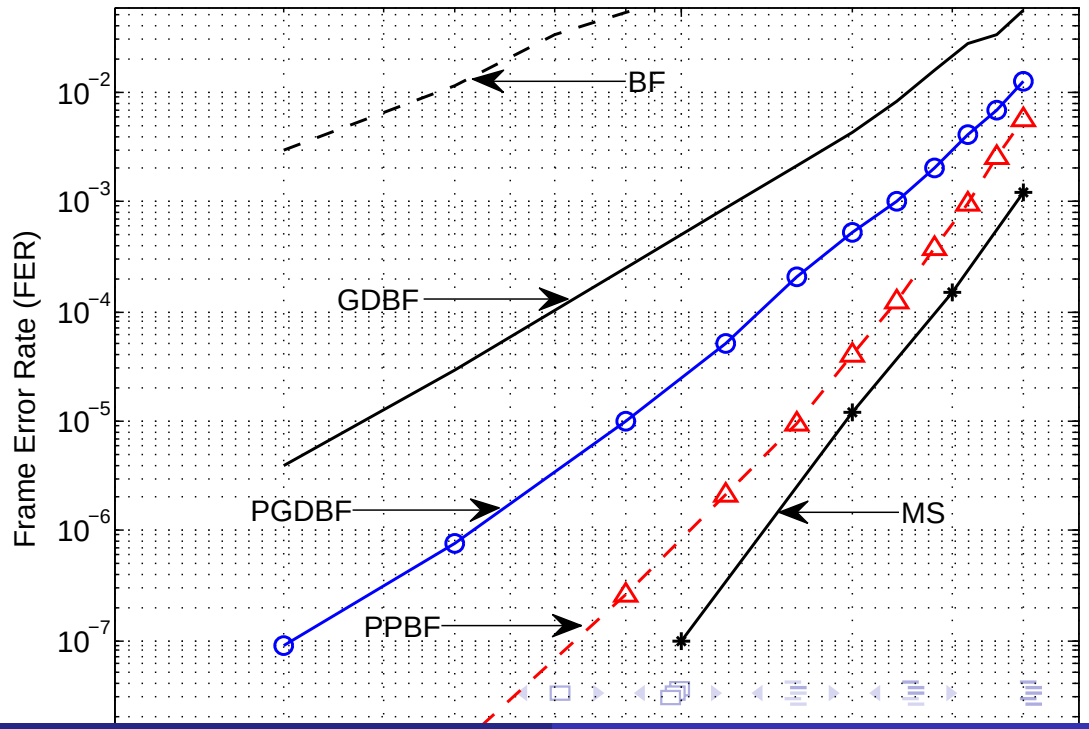
Context

- BF evolution:



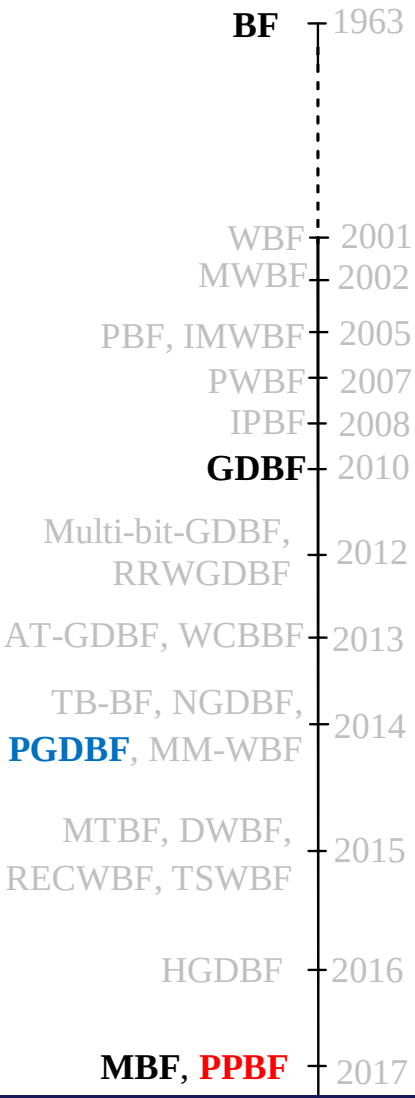
$$v_n^{(k+1)} = \text{flip}\left(v_n^{(k)}\right) \text{ if:}$$

- BF: $E_n^{(k)} = \sum_{m \in \mathcal{N}(n)} c_m^{(k)} \geq \tau$; τ: predefined threshold.
- GDBF: $E_n^{(k)} = E_{max}^{(k)}$ where $E_n^{(k)} = v_n^{(k)} \oplus y_n + \sum_{m \in \mathcal{N}(n)} c_m^{(k)}$



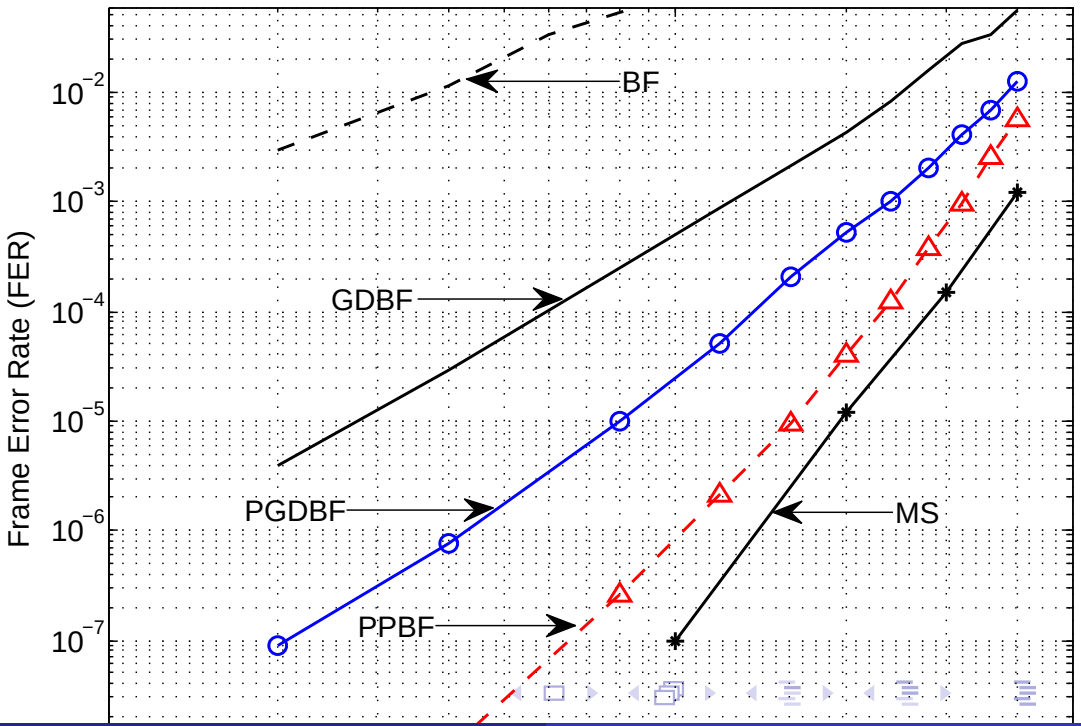
Context

- BF evolution:



$$v_n^{(k+1)} = \text{flip}\left(v_n^{(k)}\right) \text{ if:}$$

- BF: $E_n^{(k)} = \sum_{m \in \mathcal{N}(n)} c_m^{(k)} \geq \tau$; τ: predefined threshold.
- GDBF: $E_n^{(k)} = E_{max}^{(k)}$ where $E_n^{(k)} = v_n^{(k)} \oplus y_n + \sum_{m \in \mathcal{N}(n)} c_m^{(k)}$
- PGDBF: $E_n^{(k)} = E_{max}^{(k)}$ and $R_n^{(k)} = 1$ $\Pr(R_n^{(k)} = 1) = p_0, p_0 < 1$



Probabilistic Parallel Bit Flipping

PPBF

- Energy function: $E_n^{(k)} = v_n^{(k)}$ **xor** $y_n + \sum_{m \in \mathcal{N}(n)} c_m^{(k)}$,
- -
- sample N random bits $R_n^{(k)}$ from a Bernoulli distribution with probability $p_{E_n^{(k)}}$,
- $v_n^{(k+1)} = \text{NOT}\left(v_n^{(k)}\right)$ **if** $R_n^{(k)} = 1$

PGDBF

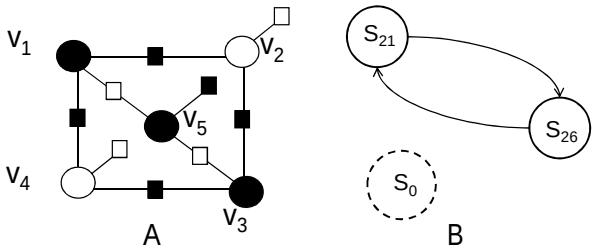
- Energy function: $E_n^{(k)} = v_n^{(k)} \oplus y_n + \sum_{m \in \mathcal{N}(n)} c_m^{(k)}$,
- $E_{max}^{(k)} = \text{Max}\left(E_n^{(k)}\right)$,
- sample N random bits $R_n^{(k)}$ from a Bernoulli distribution with probability p_0,
- $x_n^{(k+1)} = \text{NOT}\left(x_n^{(k)}\right)$ **if** $E_n^{(k)} = E_{max}^{(k)}$ **and** $R_n^{(k)} = 1$.

- Energy values $E_n^{(k)}$ is the integer, and $0 \leq E_n^{(k)} \leq d_v + 1$
- In PPBF, depend on the value of $E_n^{(k)}$, VN n is flipped with $p_{E_n^{(k)}}$
  - define $\mathbf{p} = \{p_0, p_1, \ldots p_{d_v+1}\}$
  - $E_n^{(k)} = q$ then flip with $p_q \in \mathbf{p}$
- In PPBF, all the VNs are the flipping candidates

Probabilistic flip helps improve error correction

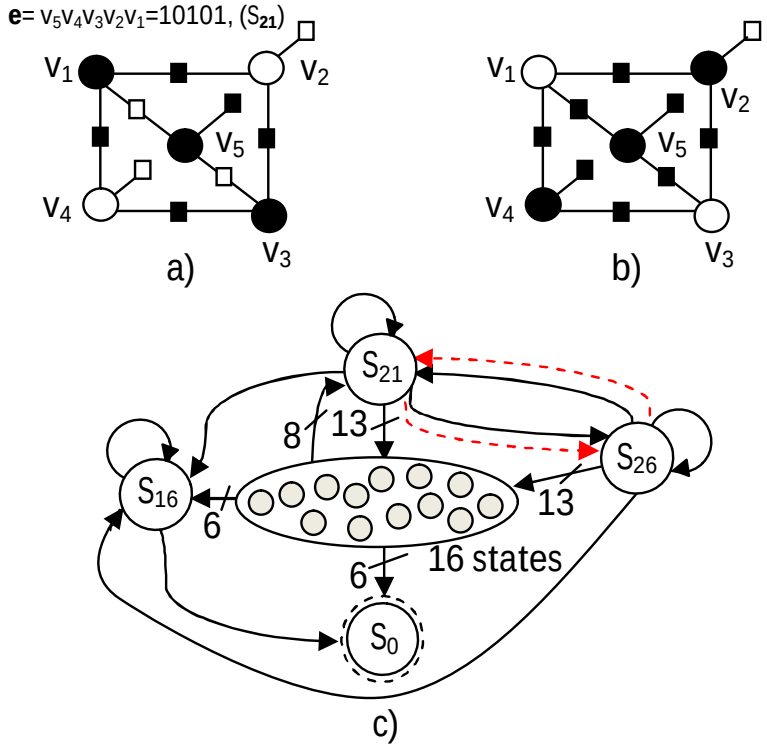
- white circles (squares) are correct VNs (satisfied CNs)
- white circles (squares) are incorrect VNs (unsatisfied CNs)
- Consider: $\mathbf{v}^{(k)}$ is a states of a state machine

GDBF



- Deterministic decoder oscillates between states and fails to converge

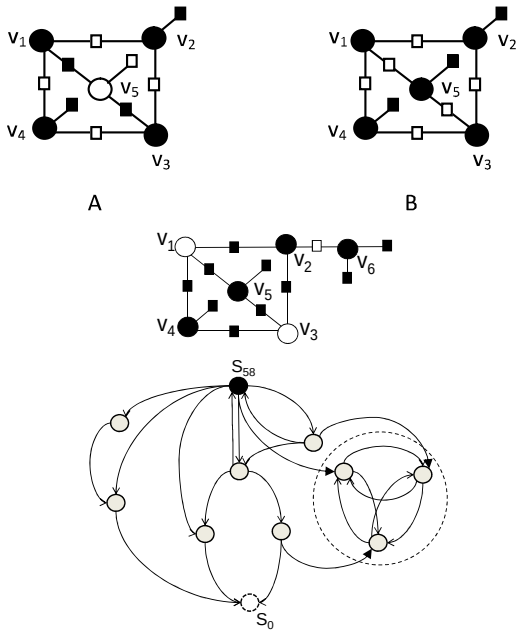
PGDBF



- The probabilistic flip creates new paths directing to the converging state

Probabilistic flip helps improve error correction

PGDBF

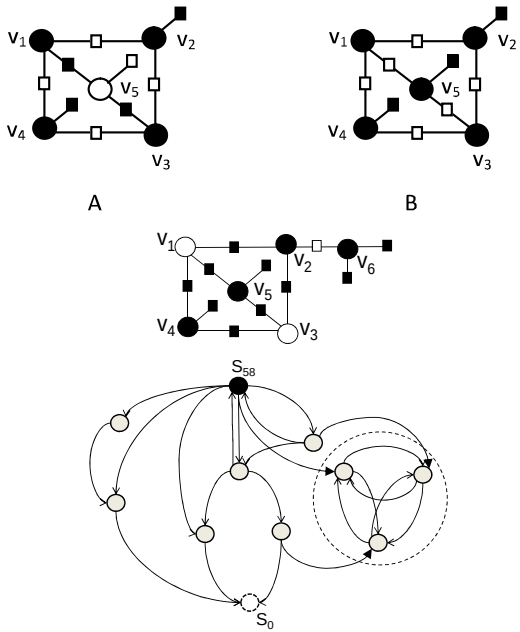


- PGDBF oscillates between states and fails to converge because only the maximum energy VNs are flipped.
- Experiment shows that, PPBF can correct these EP.

- Statistical analysis:

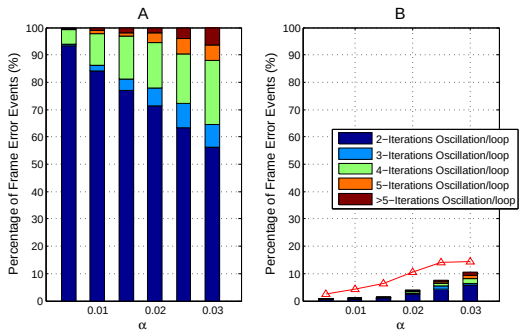
Probabilistic flip helps improve error correction

PGDBF



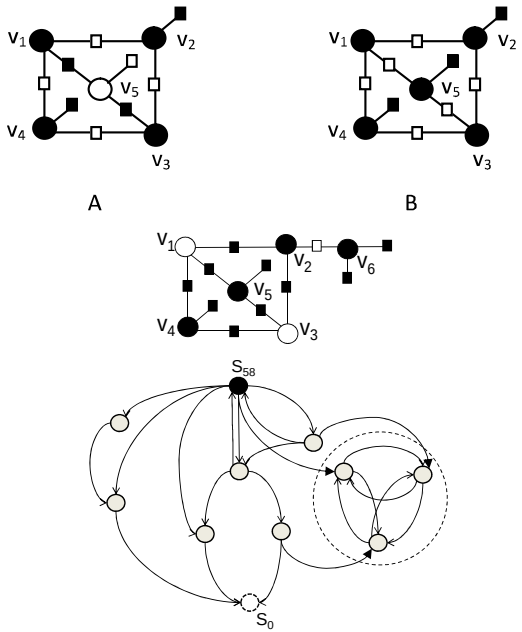
- PGDBF oscillates between states and fails to converge because only the maximum energy VNs are flipped.
- Experiment shows that, PPBF can correct these EP.

- Statistical analysis:
- Classify the failures of GDBF (Figure A)
- Re-decode the GDBF's failure EP by PGDBF and PPBF (Figure B) (PGDBF: triangle-marked red line)



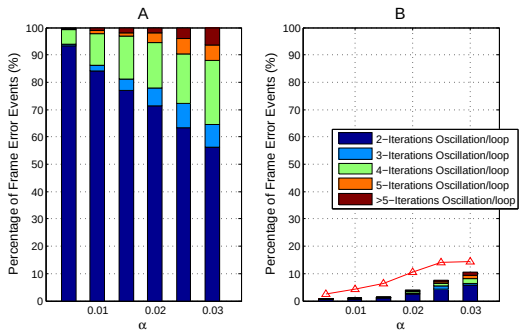
Probabilistic flip helps improve error correction

PGDBF



- PGDBF oscillates between states and fails to converge because only the maximum energy VNs are flipped.
- Experiment shows that, PPBF can correct these EP.

- Statistical analysis:
- Classify the failures of GDBF (Figure A)
- Re-decode the GDBF's failure EP by PGDBF and PPBF (Figure B) (PGDBF: triangle-marked red line)

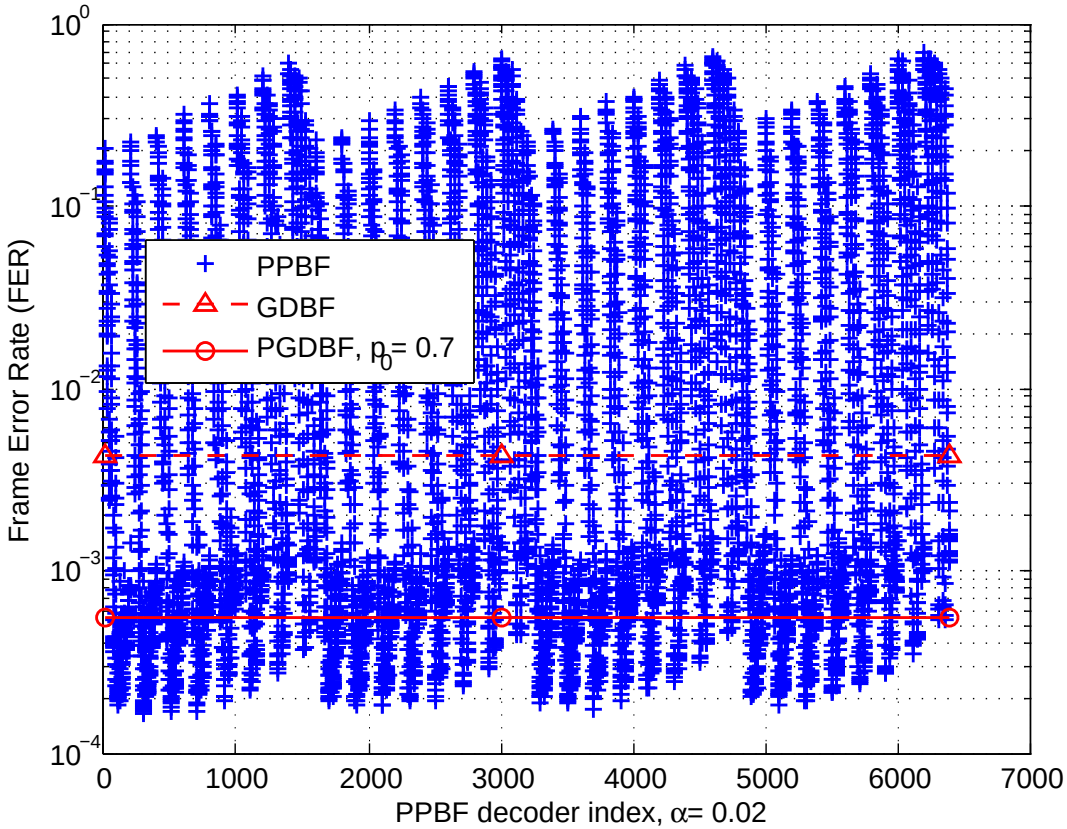


- All failure cases of GDBF are oscillations
- PPBF corrects more errors patterns than PGDBF

Probabilistic Parallel Bit Flipping: choosing $\mathbf{p}$

- For $d_v = 3$ then $\mathbf{p} = \{p_0, p_1, p_2, p_3, p_4\}$
- Test for all values of $\mathbf{p}$ at $\alpha = 0.02$ and $It_{max} = 300$.

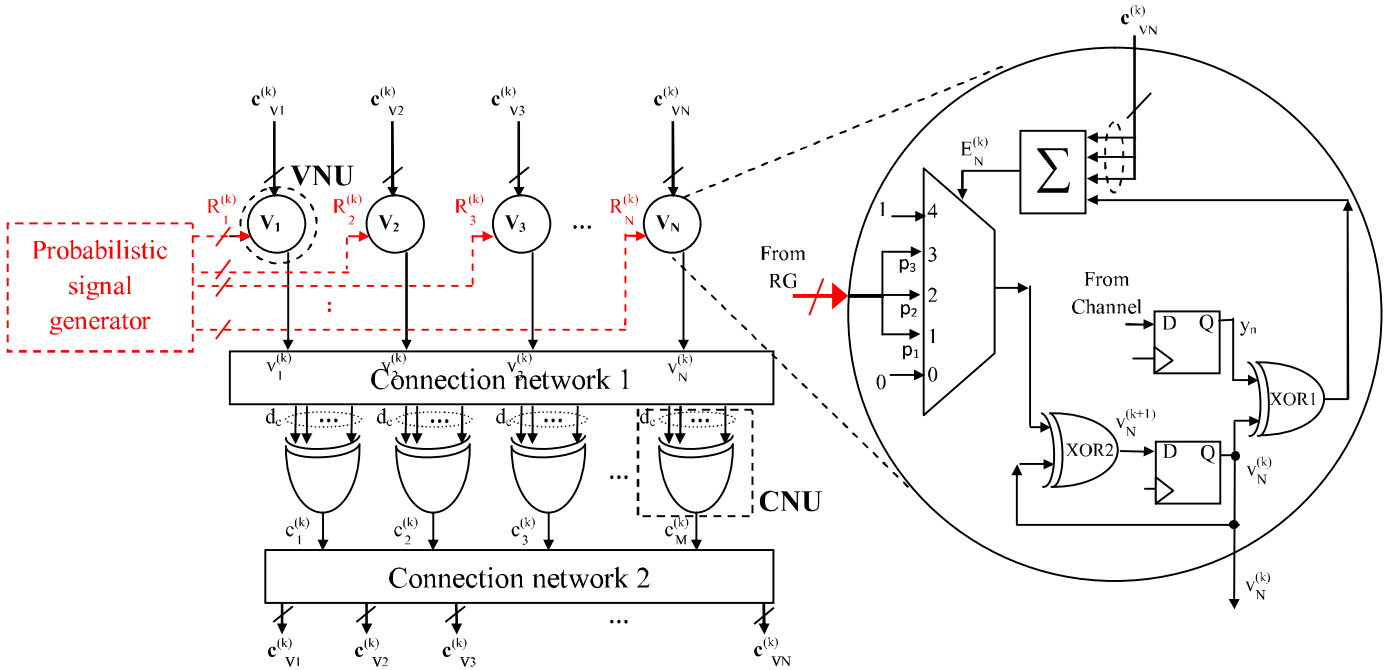
	min	max	step
p_0	0	0	0
p_1	0	0.04	0.001
p_2	0.1	0.5	0.1
p_3	0.3	1.0	0.1
p_4	0.7	1.0	0.1



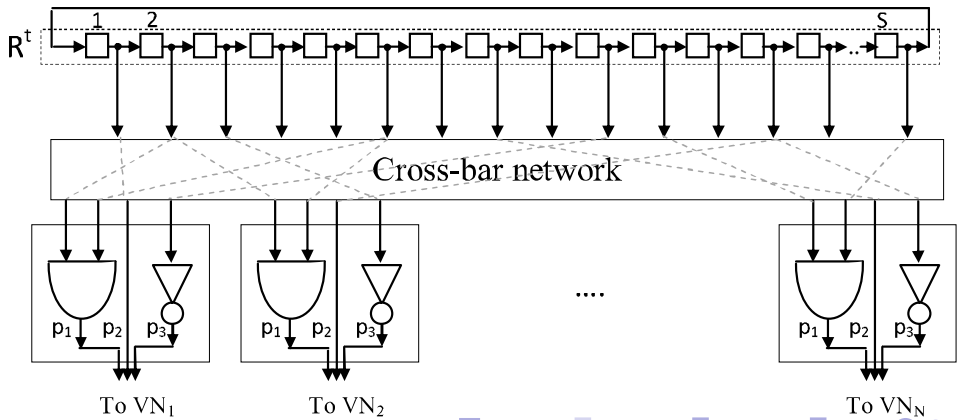
- There are 6400 values of $\mathbf{p}$ indexed from 0 to 6399
- Compare to GDBF and PGDBF with $p_0 = 0.7$

- Can choose $\mathbf{p}$ so that PPBF is better than GDBF and PGDBF and,
- this facilitates the optimization of decoder complexity and decoding performance.

Probabilistic Parallel Bit Flipping: implementation



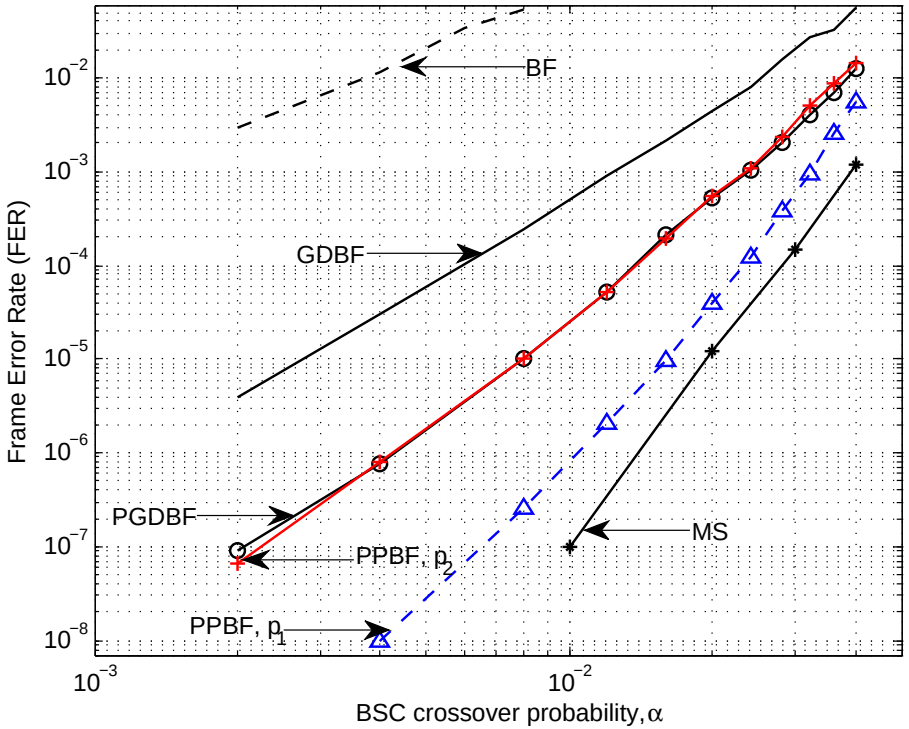
- $\mathbf{p} = \{0, 0.1, 0.2, 0.9, 1\}$
- $\Pr(p_{R_i^t} = 1) = p_1 = 0.1,$
- $\Pr(\text{NOT}(R_i^t)) = 1 - p_0 = 0.9$
- $\Pr((R_i^t \text{ AND } R_j^t) = 1) = p_0 * p_0 = 0.2$



Outline

Decoding performance

- Test code: $d_v = 3, d_c = 6, N = 155, M = 93$ (Tanner code)
- PPBF $\mathbf{p_1} = \{0.0, 0.019, 0.3, 0.9, 1\}$, $\mathbf{p_2} = \{0, 0.1, 0.2, 0.9, 1.0\}$
- BF decoders are with $It_{max} = 1000$, MS is with $It_{max} = 100$.



- PPBF with $\mathbf{p_1}$ is approaching the MS's performance
- PPBF with $\mathbf{p_2}$ performs equivalently to PGDBF

Synthesis results

- Throughput θ computation: $\theta = \frac{f_{max} * N}{It_{ave} * N_c}$
  - f_{max}: maximum frequency,
  - N_c: nb of clock cycles for each iteration,
  - It_{ave}: target number of average iterations, $(It_{ave} = 10)$

	1-bit Register	Slice LUTs	f_{max} (MHz)	θ (Gbps)
Non-Probabilistic GDBF **Khoa˙IEEEISCAS˙2015**	946 (+0%)	2151 (+0%)	132 (+0%)	2.04 (+0%)
PGDBF (with IVRG) **Khoa˙IEEEISCAS˙2015**	1038(+9.7%)	2412(+12.1%)	132 (+0%)	2.04 (+0%)
PPBF (S= 155)	476 (-50%)	991 (-54%)	244 (+85%)	3.78 (+85%)

- PPBF is lower complexity than other BF decoders
- PPBF is higher in decoding throughput

Synthesis results

Comparison between the proposed BF decoders and the state-of-the-art LDPC decoders.

	This work		**K.Le'17** [26]		**M.Ismail'13** [20]	**J.Cho'10** [28]	**Nguyen-Ly'16** [21]
(N, M)	$(1296, 648)$		$(1296, 648)$		$(1008, 504)$	$(1057, 813)$	$(1296, 648)$
Decoder	PPBF	NS-PPBF	GDBF	PGDBF	ATBF	SBF	MS$(4, 6)$
Technology	90nm		90nm		90nm	180nm	65nm
Frequency (MHz)	476	556	385	357	250	345	250
Area (mm^2)	0.367	0.349	0.360	0.367	0.729	7.4	0.72
Area scaled to 90nm (mm^2)	0.367	0.349	0.360	0.367	0.729	1.85	1.38
It_{max}	300	300	300	300	–	40	20
Throughput @ It_{max} **(Gbps)**	2.06	2.45	1.44	1.35	–	$\theta_{min} = 0.253$	2.7
Tput scaled to 90nm (Gbps)	2.06	2.45	1.44	1.35	–	0.506	1.95
It_{ave} **@ FER=** 10^{-5}	7.71 $@\alpha \approx 0.012$	7.58 $@\alpha \approx 0.012$	2.24 $@\alpha \approx 0.005$	5.48 $@\alpha \approx 0.011$	–	–	2.34 [26] $@\alpha \approx 0.025$
Average throughput @ FER= 10^{-5} **(Gbps)**	80	95.1 (*)	222.8	84.4	25.2 @ $It = 10$	$\theta_{max} = 17.37$	-
Average throughput scaled to 90nm (Gbps)	80	95.1 (*)	222.8	84.4	25.2	34.74	23.1 [26]
Average TAR $(Gbps/mm^2)$ (90nm)	218	272.4 (*)	618.9	230	34.6	18.8	16.7

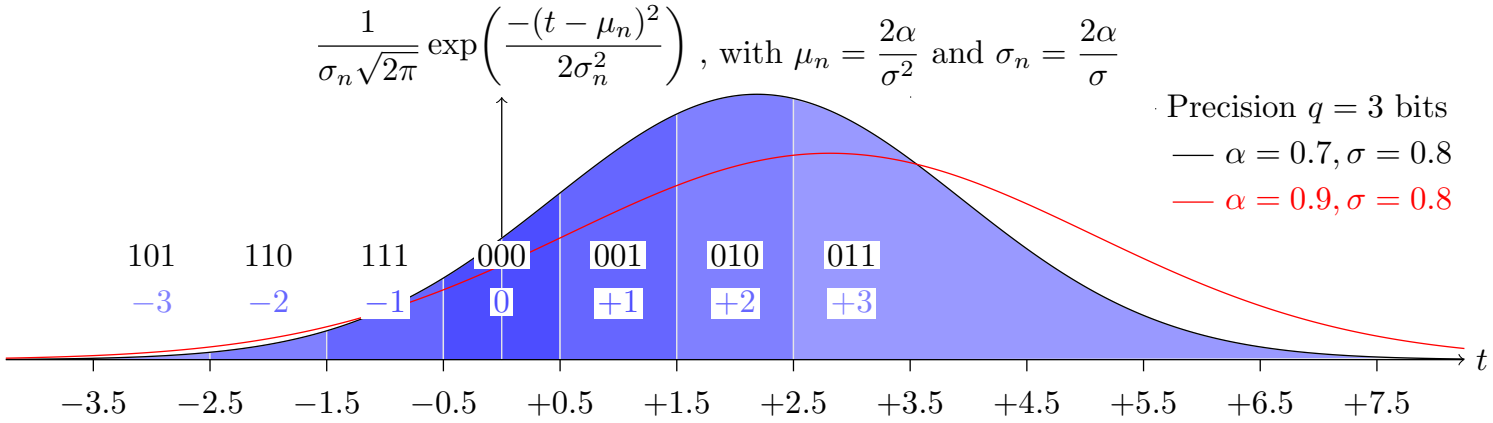
(*) Since the SC module is not implemented in this NS-PPBF implementation, these average values of NS-PPBF are bounds on the achievable results and they are not really practical relevance.

Outline

Outline

Classical OMS decoder: quantification of channel LLR

Quantified intrinsic information I_n is obtained from channel LLR y_n as $I_n = \mathcal{Q}(\alpha \times y_n)$, with $y_n \in \mathbb{R}$, α a scaling factor and $\mathcal{Q}$ a quantification rules from $\mathbb{R}$ to $\mathcal{A}_C = \{-3, -2, -1, 0, +1, +2, +3\}$ defined as $\mathcal{Q}(a) = \mathcal{S}_3(\lfloor a + 0.5 \rfloor)$, with $\mathcal{S}_3$ the saturation function toward [-3, 3].

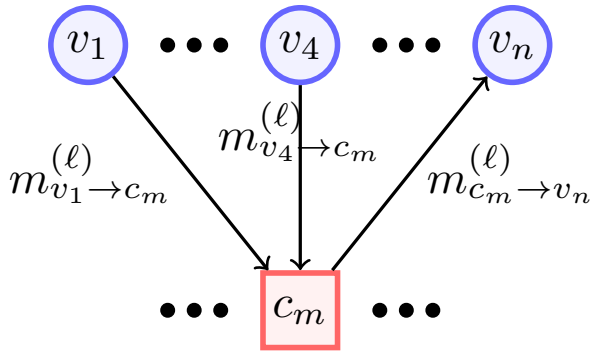


Among the 8 levels of a 3 bits representation, only 7 are used.

Low precision Offset Min-Sum Based Decoders

- **Update rule at a CNU**

$$m_{c_m \to v_n}^{(\ell)} = \left(\prod_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \text{sign}\left(m_{v \to c_m}^{(\ell)} \right) \right) \times \max\left\{ \min_{v \in \mathcal{V}(c_m) \setminus \{v_n\}} \left| m_{v \to c_m}^{(\ell)} \right| - 1, 0 \right\}.$$



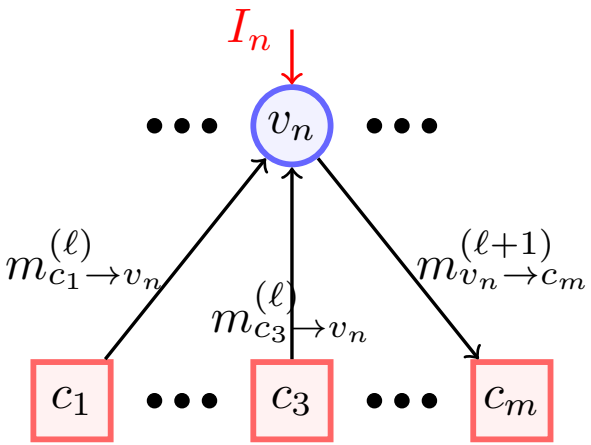
- Note that for low precision $q = 3$ (same for $q = 4$), the offset applied in CNUs only gives us the possibility to use 5 values ($m_{c_m \to v_n}^{(\ell)} \in \{-2, -1, 0, +1, +2\}$) instead of the 7 values of $\mathcal{A}_C$.

Low precision Offset Min-Sum Based Decoders

- **Update rule at a VNU**

$$m_{v_n \to c_m}^{(\ell+1),U} = I_n + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} m_{c \to v_n}^{(\ell)}.$$

exponent U indicates unsaturated message.



$$m_{v_n \to c_m}^{(\ell+1)} = \mathcal{S}_3 \left(m_{v_n \to c_m}^{(\ell+1),U} \right).$$

$$m_{v_n \to c_m}^{(\ell+1)} \in \{-3, -2, -1, 0, +1, +2, +3\}.$$

Low precision Offset Min-Sum Based Decoders

From the analysis of OMS-based decoders with $q = 3$

(i) the quantized LLRs $I_n \in \mathcal{A}_C = \{-3, -2, -1, 0, +1, +2, +3\}$,
(ii) v-to-c messages $m_{v_n \to c_m}^{(\ell+1)} \in \mathcal{A}_C = \{-3, -2, -1, 0, +1, +2, +3\}$, and
(iii) c-to-v messages $m_{c_m \to v_n}^{(\ell)} \in \mathcal{A}_C \backslash \{-3, +3\} = \{-2, -1, 0, +1, +2\}$.

The OMS-based decoders are suboptimal

It can be clearly noted that all combinations that can be obtained from $q = 3$ bits is not used.

How to use the 8 quantization levels for precision $q = 3$

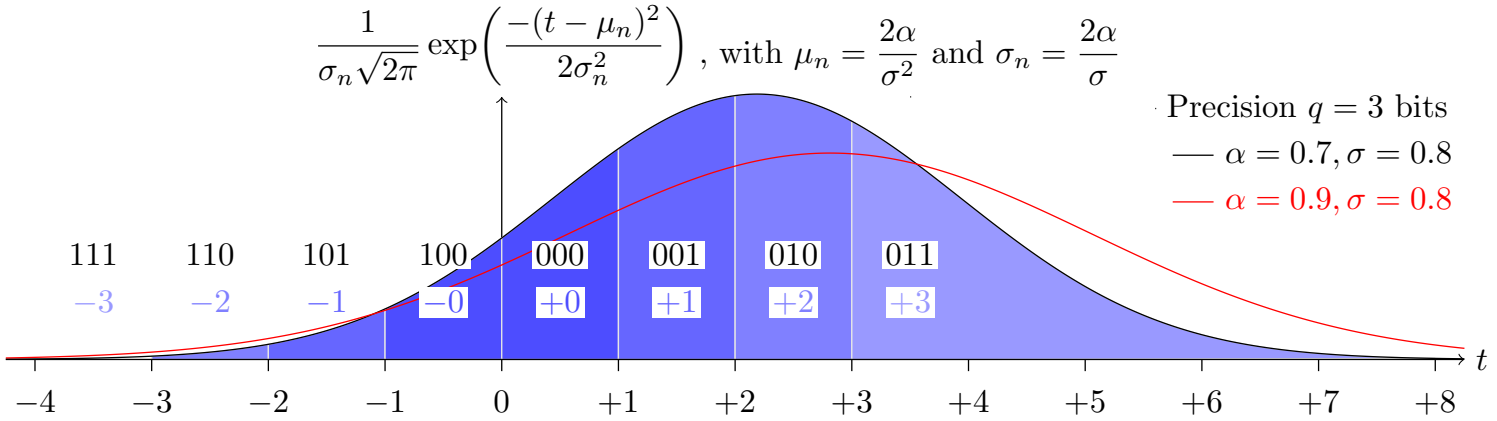
We define a new decoder called Sign-Preserving decoder where:

- The quantization of LLRs has to be changed.
- CNU has to be changed.
- VNU has to be changed.

Outline

Quantization used for Sign-Preserving Min-Sum Decoders

- In order to use the 8 levels of $q = 3$ bits, we define the message alphabet as $\mathcal{A}_S = \{-3, -2, -1, -0, +0, +1, +2, +3\}$. Using the sign-and-magnitude representation we have $100_2 = -0$, $000_2 = +0$, etc.
- A new quantification process $\mathcal{Q}^*$ is used: $I_n = \mathcal{Q}^*(\alpha \times y_n) \in \mathcal{A}_S$ where $\mathcal{Q}^*(a) = (\text{sign}(a), \mathcal{S}_3(\lceil \alpha \times |a| \rceil - 1))$



The *channel gain factor* α represents a degree of freedom in the decoder definition.

Update rules for SP-MS Decoders

We define the Sign-Preserving Min-Sum decoders as decoders that always preserve the sign of the messages **Franklin˙IEEE10thISTC˙2018**.

- **Update rule at a CNU**
  We move the offset from CNs to VNs

- **Update rule at a VNU**
  The VN of the classical OMS can erase the bit value.
  The VN of the SP-MS processes the c-to-v messages and the LLR to always make a decision, *i.e.* the v-to-c messages always have a sign.

- Density Evolution (DE) technique is used to optimize the parameters of the SP-MS decoder.
- SNR gains in the waterfall correspond to what was predicted with DE.

SP-MS decoders for (5,20)-regular LDPC codes

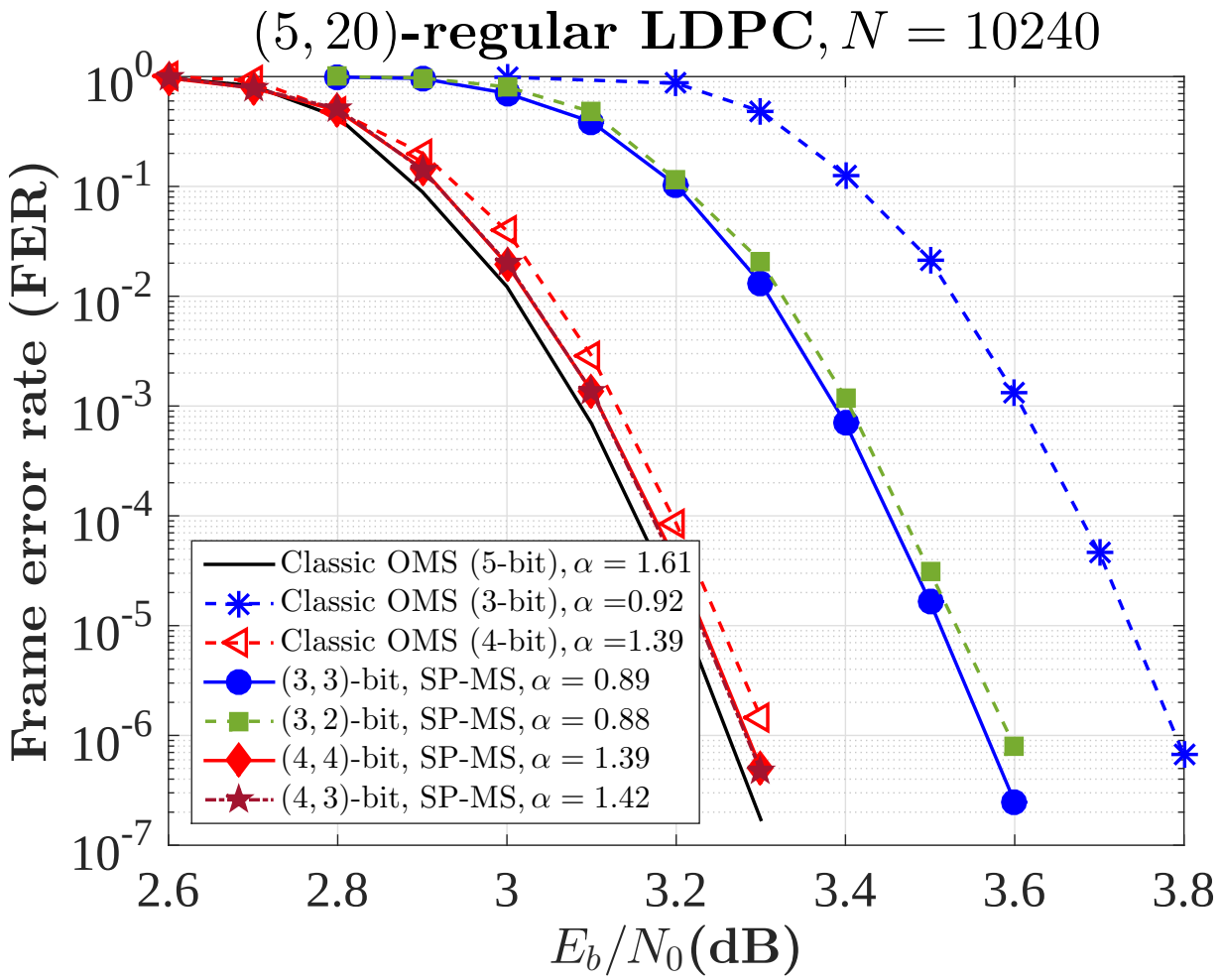


Figure: FER performance of SP-MS decoders for precision $q \in \{2, 3, 4\}$.

SP-MS decoders for (3,18)-regular LDPC codes

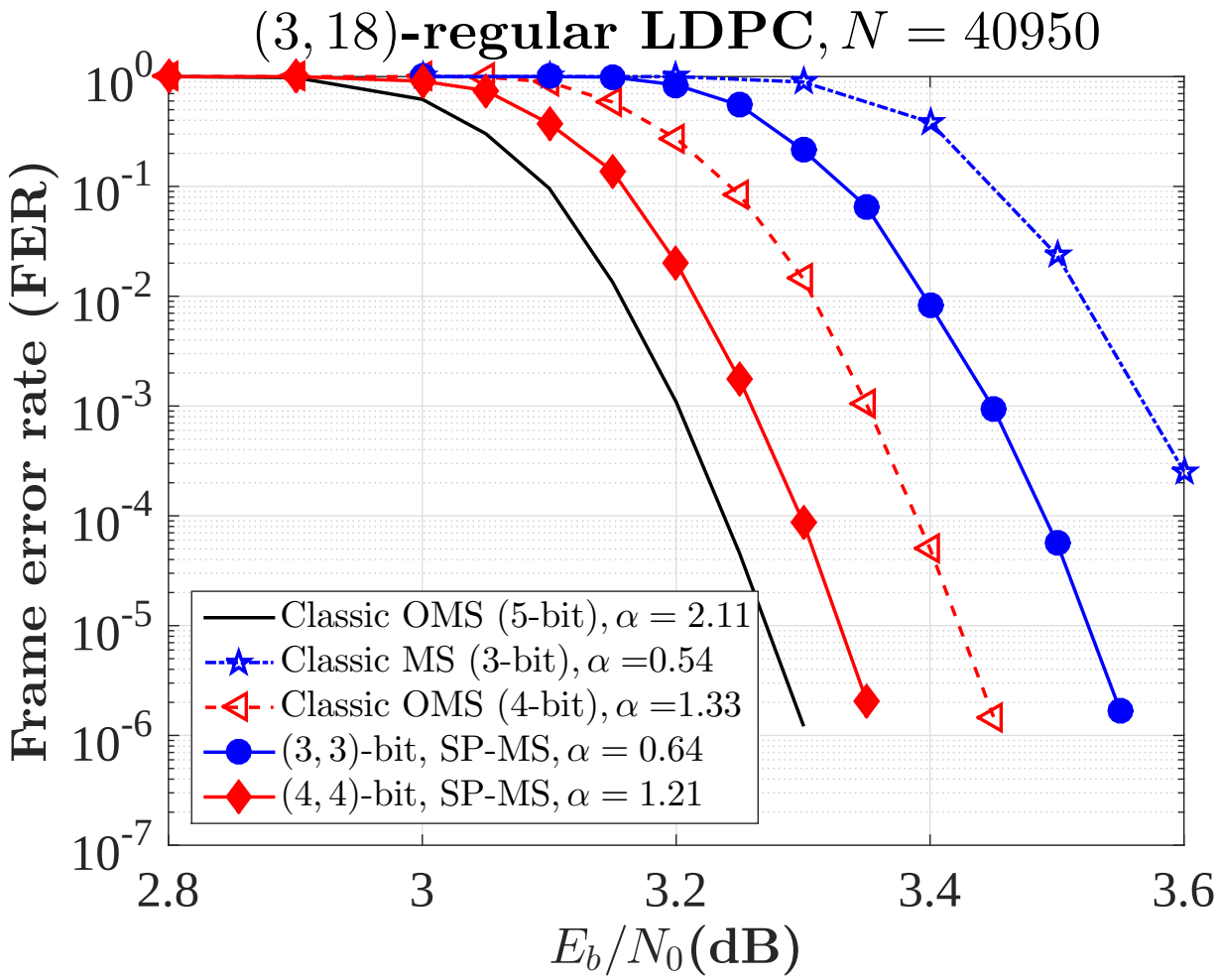


Figure: FER performance of SP-MS decoders for precision $q \in \{3, 4\}$.

Qualitative result obtained by SP-MS algorithm

For $(q_{ch}, q) = (4, 4)$, SP-MS gives small gain compared to MS or OMS

For $(q_{ch}, q) = (4, 3)$, SP-MS is almost equivalent to SP-MS with $(q_{ch}, q) = (4, 4)$!

For $(q_{ch}, q) = (3, 3)$, SP-MS gives 0.2 up to 0.4 dB of gain compared to MS or OMS.

For $(q_{ch}, q) = (3, 2)$, SP-MS is almost equivalent to SP-MS with $(q_{ch}, q) = (3, 3)$!.

Outline

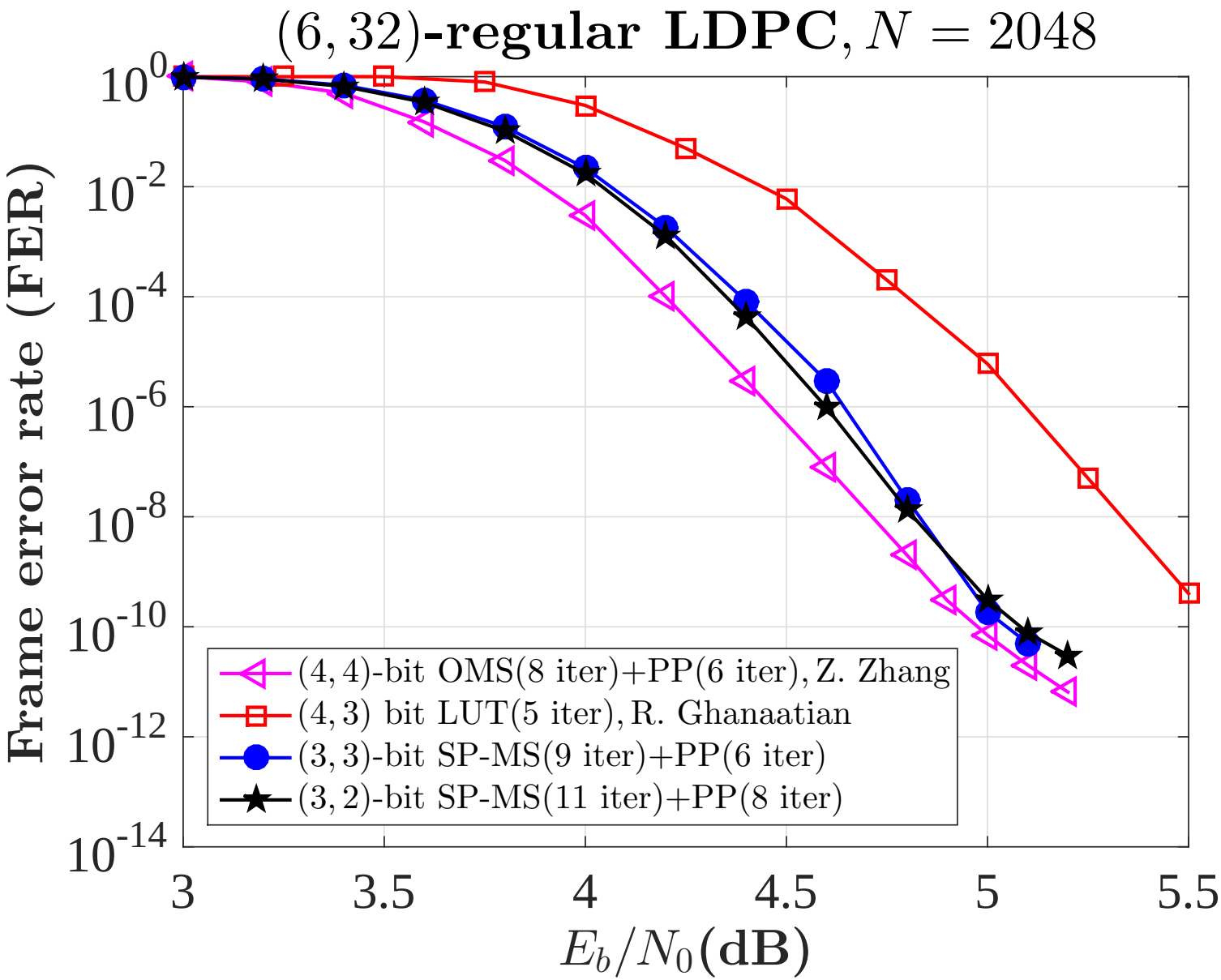
Synthesis results for regular LDPC Codes

IEEE 802.3 LDPC code: Rate 0.8143 $(2048, 1723)$ with regular $(6, 32)$ degree distribution, precision $(q_{ch} = 3, q = 3)$ and $(q_{ch} = 3, q = 2)$.

	This work[†]		Ghanaatian,2018,**Reza˙IEEETransVLSI˙2018**	Zhang,2010,**Zhengya˙IEEEJournalSolidStateCirc˙2010**
Technology	28nm FDSOI		28nm FDSOI	65nm CMOS
Decoder	SP-MS		finite-alphabet	OMS
Precision (q_{ch}, q)	(3,2) bits	(3,3) bits	(4,3) bits	(4,4) bits
Iterations	11 + 8 PP	9 + 6 PP	5	8 + 6 PP
Architecture	full-parallel		unrolled full-parallel	partial-parallel
E_b/N_0 @ FER= 10^{-10}	5.07 dB	5.03 dB	5.51 dB	4.98 dB
Frequency	421 MHz[†]	500 MHz[†]	862 MHz	700 $\xrightarrow{28nm}$ 1000 MHz
Core area	1.76 mm^{2†}	2.56 mm^{2†}	16.2 mm^2	5.05 $\xrightarrow{28nm}$ 1.77 mm^2
Throughput	45.4 Gbit/s[*]	68.3 Gbit/s[*]	588 Gbit/s	13.3 $\xrightarrow{28nm}$ 19 Gbit/s
Hardware efficiency	25.8 Gbit/s/mm^{2*}	26.7 Gbit/s/mm^{2*}	36.3 Gbit/s/mm^2	2.63 $\xrightarrow{28nm}$ 10.7 Gbit/s/mm^2
Throughput (4.5 dB)	215.6 Gbit/s[**]	256 Gbit/s[**]	588 Gbit/s	33.3 $\xrightarrow{28nm}$ 48.2 Gbit/s
Hardware efficiency (4.5 dB)	122.5 Gbit/s/mm^{2**}	100 Gbit/s/mm^{2**}	36.3 Gbit/s/mm^2	6.59 $\xrightarrow{28nm}$ 26.91 Gbit/s/mm^2

[†] Preliminary results.
[*] The worst case.
[**] The number of average decoding iteration is approximately 4 iterations.

Emulation on FPGA for the IEEE 802.3 ETHERNET code



Outline

Conclusions

We believe that PPGDBF and SP-MS can reach few 100 Gbit/s/mm2 on 28 nm technology with very good FER.

In parallel, ondoing work on high girth (10, 12) small size regular QC-LDPC matrix that can be used for high performance, low complexity convolutional codes for optical fiber.

Proposed methods can be used for several trade-off performance complexity, either for short or long frames.

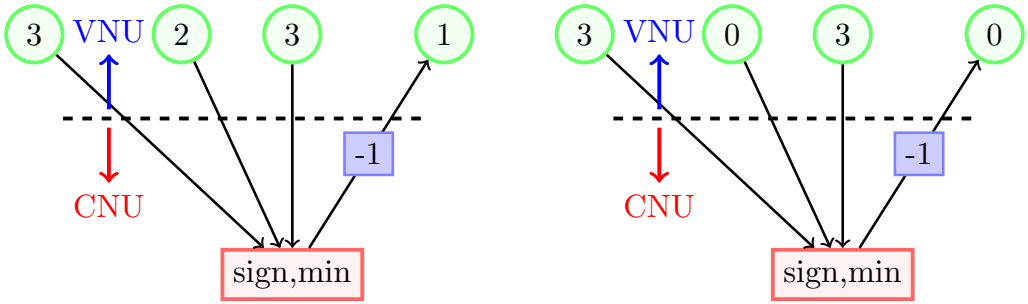
Thank you for listening!

Questions?

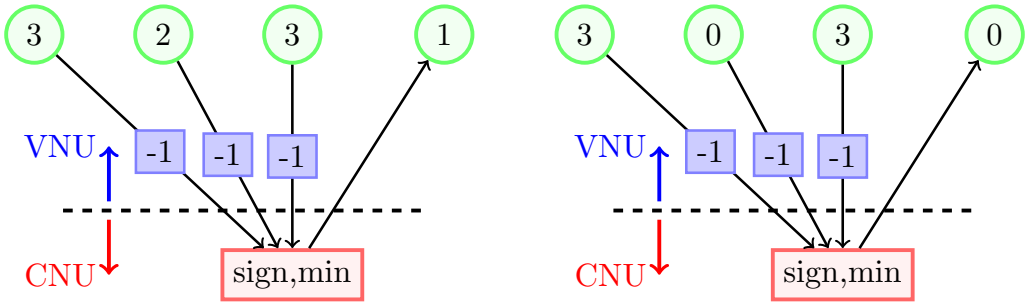
CNU of Sign-Preserving Min-Sum Decoders

The update rule at a CNU can be written in two equivalent ways.

Classical method: $\left|m_{c_m \to v_n}^{(\ell)}\right| = \max\left\{\min_{v \in \mathcal{V}(c_m)\setminus\{v_n\}} \left|m_{v \to c_m}^{(\ell)}\right| - 1, 0\right\}.$



Equivalent method: $\left|m_{c_m \to v_n}^{(\ell)}\right| = \min_{v \in \mathcal{V}(c_m)\setminus\{v_n\}} \left(\max\left(\left|m_{v \to c_m}^{(\ell)}\right| - 1, 0\right)\right).$



Therefore, we can move the offset from CNs to VNs.

VNU of Sign-Preserving Min-Sum Decoders

Moving the offset from CNs to VNs, the update rule at a VNU of OMS-based decoders can be rewritten as

$$m_{v_n \to c_m}^{(\ell+1),U} = I_n + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} m_{c \to v_n}^{(\ell)}.$$

The offset is subtracted before saturation to allow -3 and 3 values in the variable to check message.

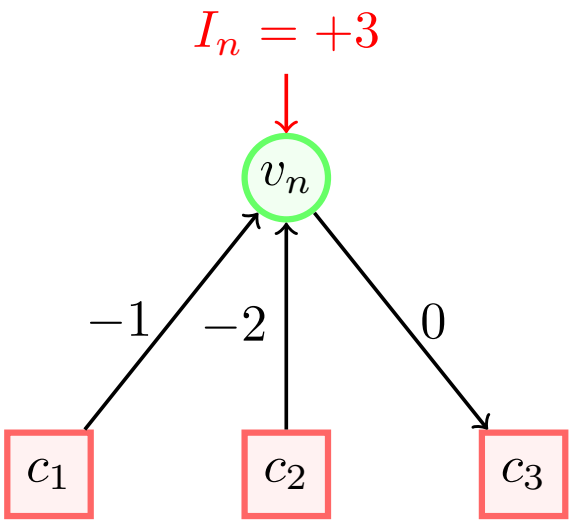
$$m_{v_n \to c_m}^{(\ell+1)} = \text{sign}\left(m_{v_n \to c_m}^{(\ell+1),U}\right) \times \mathcal{S}_3\left(\max\left(\left|m_{v_n \to c_m}^{(\ell+1),U}\right| - 1, 0\right)\right)$$

Problem of the update rule at VNUs

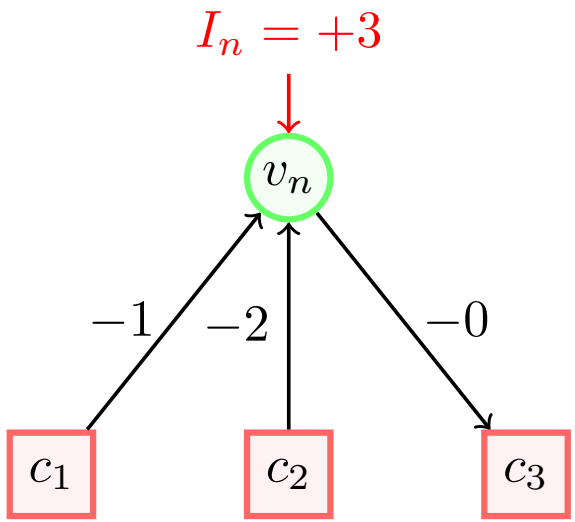
The v-to-c message $m_{v_n \to c_m}^{(\ell+1)}$ can be zero, zero does not have any information about the bit value, this means that the VNU can erase the bit value.

VNU of Sign-Preserving Min-Sum Decoders

Classical VNU:



Modified VNU:



Idea to avoid the zero value.

- **Modified VNU:** we propose to use half the sum of the signs of the c-to-v messages and the sign of the quantized LLR
  Example: $a = (+\mathbf{3}-\mathbf{1}-\mathbf{2}) + (+\mathbf{1}-\mathbf{1}-\mathbf{1})/2 = -\mathbf{0.5}$, hence $m^{(\ell),U}_{v_n \to c_3} = (\text{sign}(a), \lfloor |a| \rfloor) = -\mathbf{0}$ and $m^{(\ell)}_{v_n \to c_3} = -\mathbf{0}$.

We define the Sign-Preserving Min-Sum decoders as decoders that always preserve the sign of the messages **Franklin˙IEEE10thISTC˙2018**.

Update rules for SP-MS Decoders

Let us denote by $\mu_{v_n \to c_m}^{(\ell)}$ the *sign-preserving factor* of the message $m_{v_n \to c_m}^{(\ell+1)}$, defined as

$$\mu_{v_n \to c_m}^{(\ell)} = \xi \times \text{sign}(I_n) + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \text{sign}\left(m_{c \to v_n}^{(\ell)}\right),$$

where $\xi = \begin{cases} 0, & d_v = 2, \\ 1, & d_v \in \{3, 5, 7, ...\}, \\ 2, & d_v \in \{4, 6, 8, ...\}. \end{cases}$

By construction $\mu_{v_n \to c_m}^{(\ell)}$ take its value between $\{-1, 1\}$ for $d_v = 2$, $\{-d_v, -d_v + 2, \ldots, -1, 1, \ldots, d_v\}$ for d_v odd and $\{-d_v - 1, -d_v + 1, \ldots, -1, 1, \ldots, d_v + 1\}$ for d_v even.

Update rules for SP-MS Decoders

- Let us redefined $m_{v_n \to c_m}^{(\ell+1),U}$ as

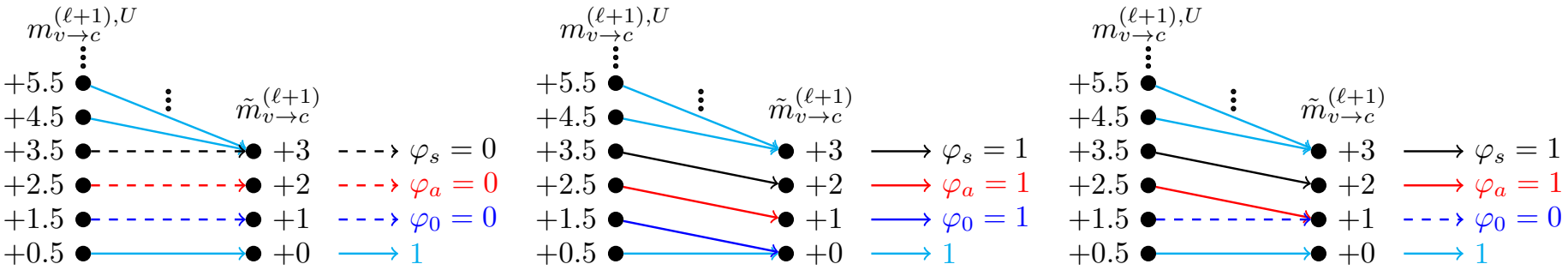
$$m_{v_n \to c_m}^{(\ell+1),U} = \frac{\mu_{v_n \to c_m}^{(\ell)}}{2} + I_n + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} m_{c \to v_n}^{(\ell)}$$

- $m_{v_n \to c_m}^{(\ell+1),U}$ takes its value in the set $\mathcal{A}_U = \{\ldots, -1.5, -0.5, +0.5, +1.5, \ldots\}$.
- and we obtain a message $m_{v_n \to c_m}^{(\ell+1)}$ that has always a defined sign as

$$m_{v_n \to c_m}^{(\ell+1)} = \left(\text{sign}\left(m_{v_n \to c_m}^{(\ell+1),U}\right), \mathcal{S}_3\left(\max\left(\left\lfloor\left|m_{v_n \to c_m}^{(\ell+1),U}\right|\right\rfloor - 1, 0\right)\right)\right).$$

Optimization of Sign-Preserving Min-Sum Decoders

Thanks to density evolution, we can assess the optimal saturation rules at variable node level. For example, for $q_{ch} = 3$ (channel quantization), $q = 3$ (message quantization), $(d_v, d_c) = (4, 8)$ we obtained:



(a) SP-MS. (b) SP-OMS. (c) Optimized SP-MS.

MS Decoder	OMS decoders	Optimized SP-MS	DE gain	SNR gain
$\delta_{db} = 2.736$	$\delta_{db} = 2.322$	$\delta_{db} = 1.982$	**0.3399**	**0.32**

SNR gains in the waterfall correspond to what was predicted with DE.

SP-MS decoders for (6,32)-regular LDPC codes

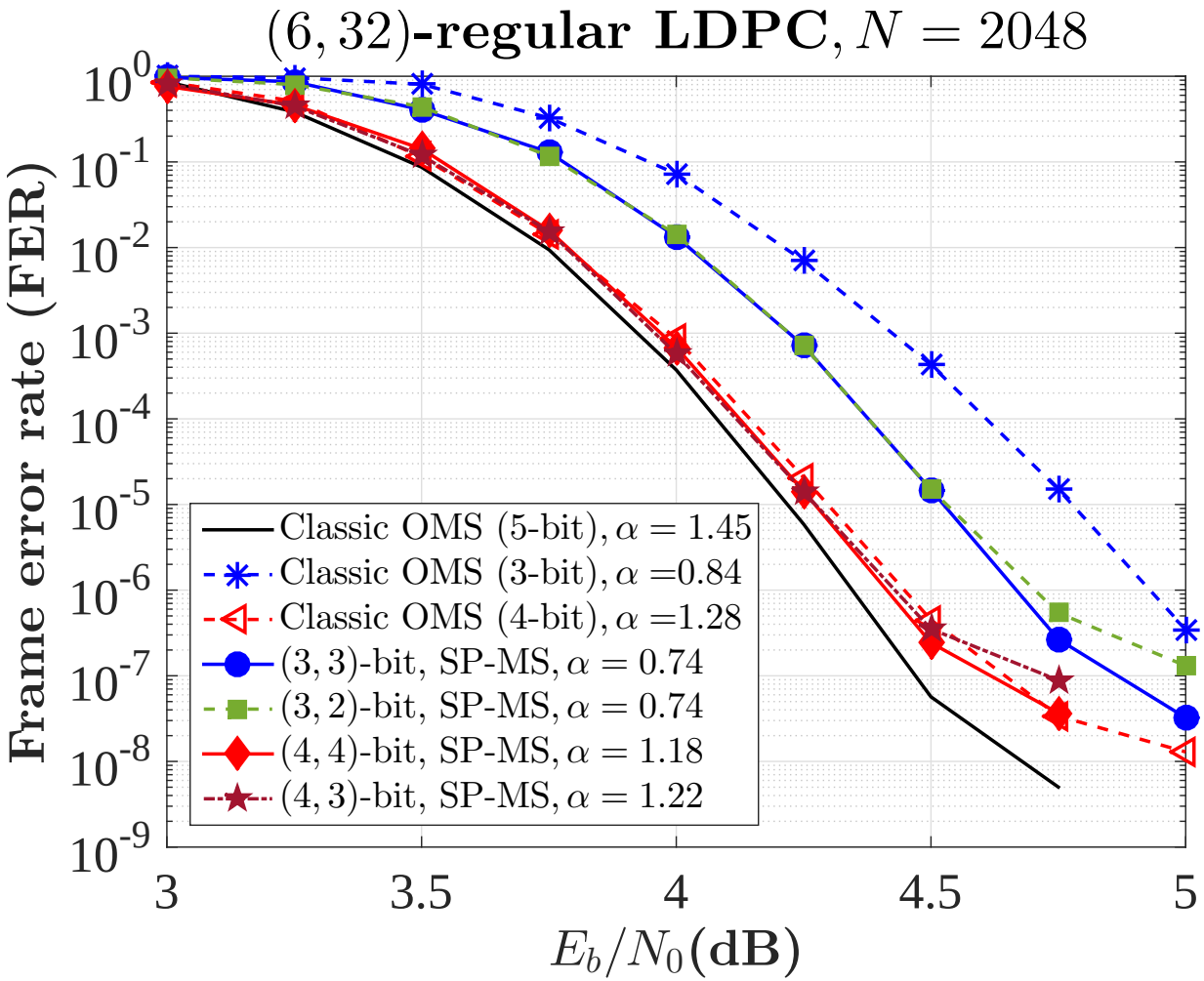


Figure: FER performance of SP-MS decoders for precision $q \in \{2, 3, 4\}$.

SP-MS decoders for (4,64)-regular LDPC codes

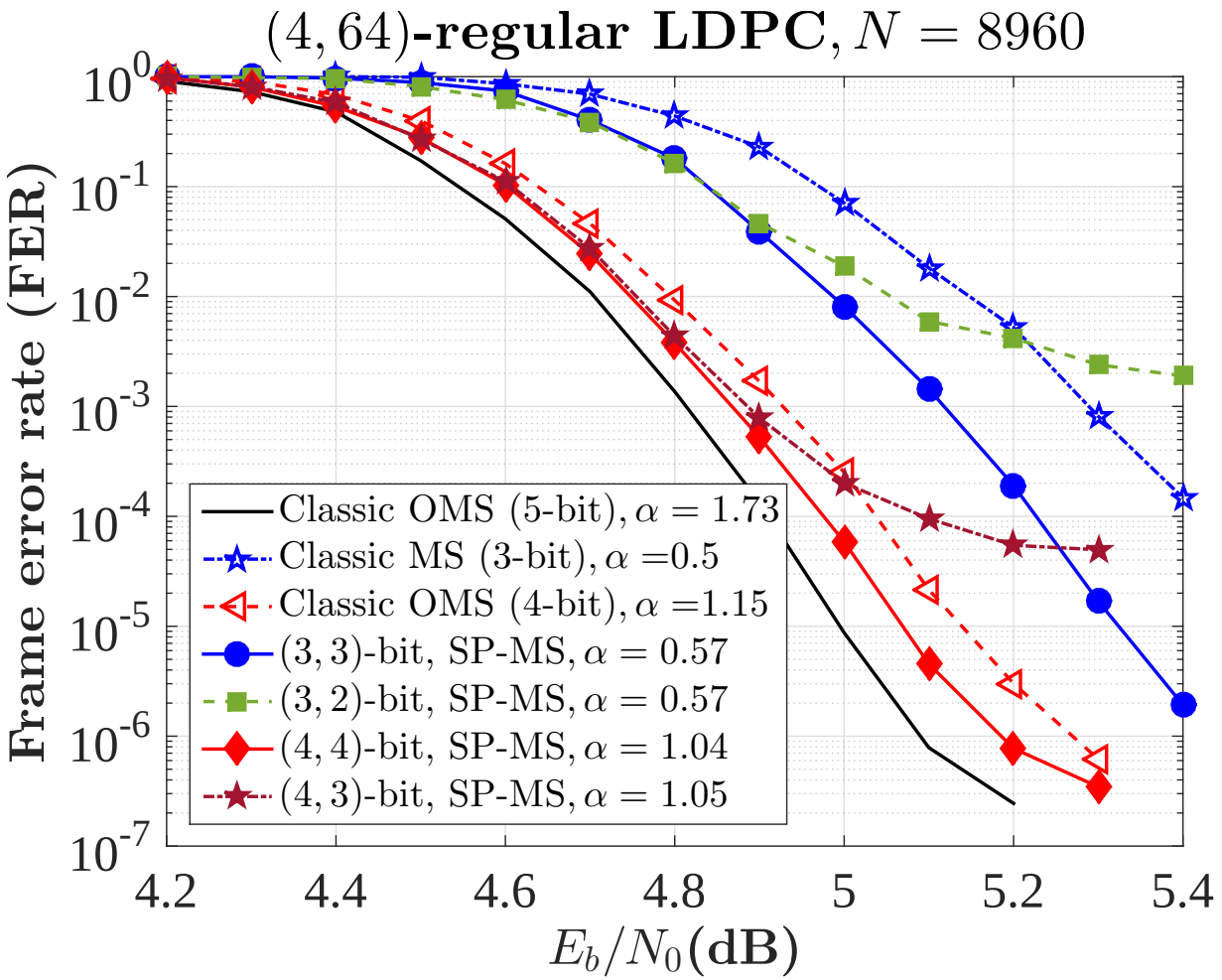


Figure: FER performance of SP-MS decoders for precision $q \in \{2, 3, 4\}$.

SP-MS decoders for (3,15)-regular LDPC codes

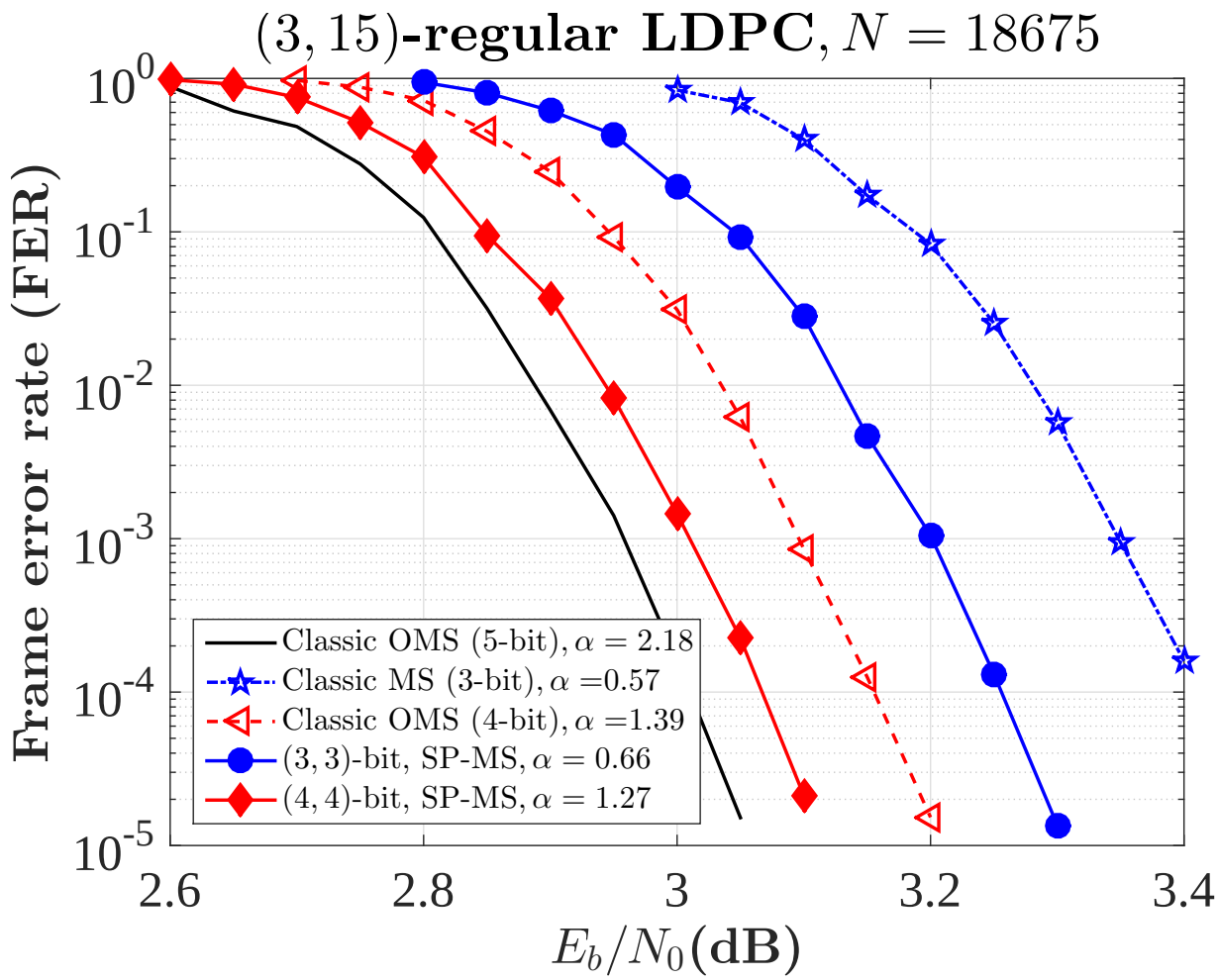


Figure: FER performance of SP-MS decoders for precision $q \in \{3, 4\}$.

References I