Efficient Implementation of Probabilistic Gradient Descent Bit Flipping Decoders

K. Le, F. Ghaffari and D. Declercq

ETIS, ENSEA/UCP/CNRS France

B. Vasic,

Department of Elec.and Comp. Eng. University of Arizona Tucson, USA

September 2015

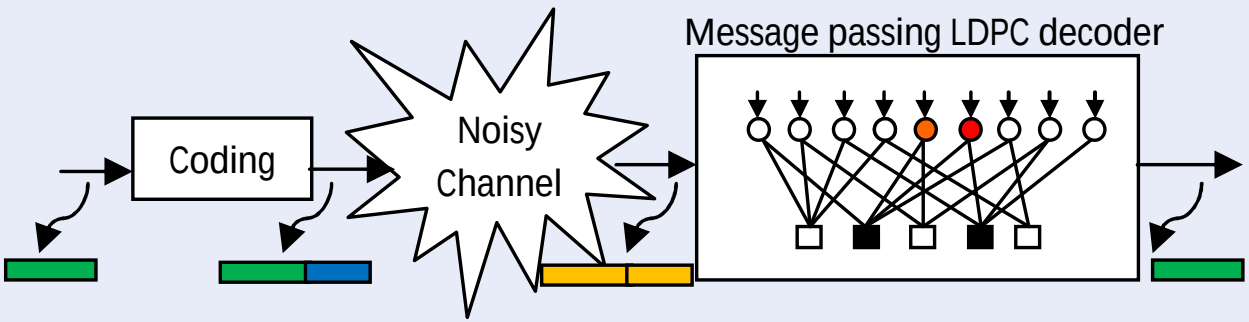
Outline

Outline

Context & Objectives

Low Density Parity Check (LDPC) Decoders

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



2 types of LDPC decoding algorithms :

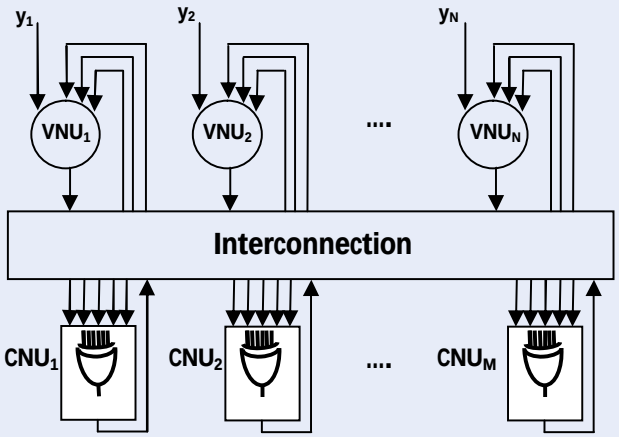
- Soft information algorithms : Sum-Product, Min-Sum..., powerful error correction capacity but high complexity.
- Hard-decision algorithms : Gallager Bit Flipping (BF), Gradient Descent Bit Flipping (GDBF), Probabilistic GDBF..., low complexity, usually weak in error correction.
- We work on hard-bit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

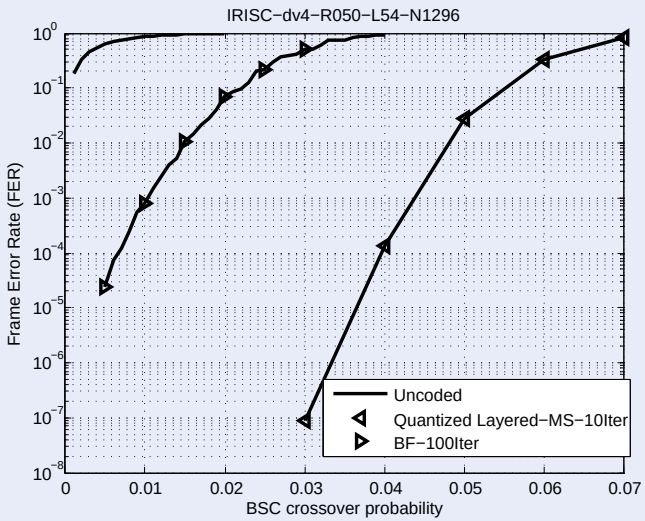
Context & Objectives

Concept of Bit Flipping (BF) Algorithm :

- Iteratively passing binary information between 2 groups of processing units :
  1. **Check Nodes Units** (CNU) : compute parity check equations (XOR operations),
  2. **Variable Nodes Units** (VNU) : the VN value is flipped if the number of violated CN neighbors is too large.
- BF : $x_n^{(k+1)} = \text{flip}\left(x_n^{(k)}\right)$ **if** $E_n^{(k)} = \sum_{m \in \mathcal{N}(\backslash)} s_m^{(k)} \geq \tau$, τ : predefined threshold.
  - $x_n^{(k)}$: current value of VN n at iteration k
  - $s_m^{(k)}$: current value of CN m at iteration k

Hardware architecture and performance comparison



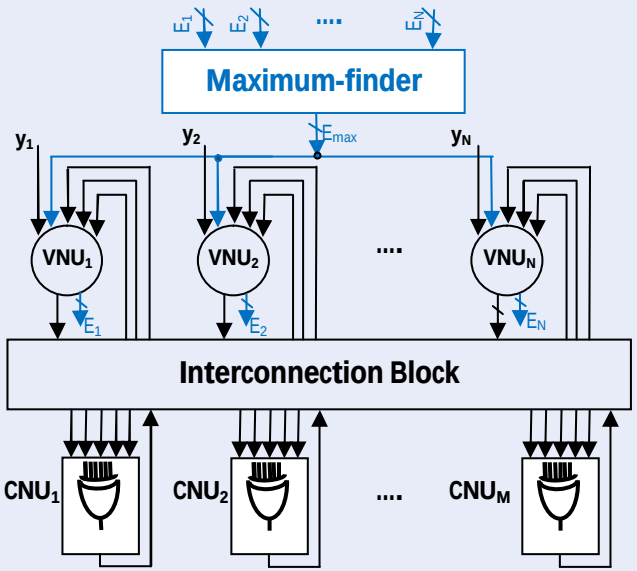


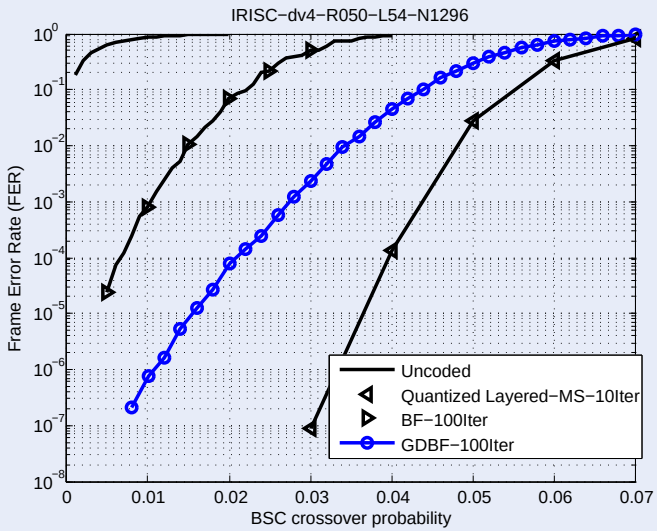
Context & Objectives

Gradient Descent Bit Flipping (GDBF) algorithms : BF with following modifications

- Energy function : $E_n^{(k)} = x_n^{(k)}$ **xor** $y_n + \sum_{m \in \mathcal{N}(\setminus)} s_m^{(k)}$, with y_n the noisy version of VN n.
- $E_{max}^{(k)} = \text{Max}\left(E_n^{(k)}\right)$,
- $x_n^{(k+1)} = \text{flip}\left(x_n^{(k)}\right)$ **if** $E_n^{(k)} = E_{max}^{(k)}$

Hardware architecture and performance comparison





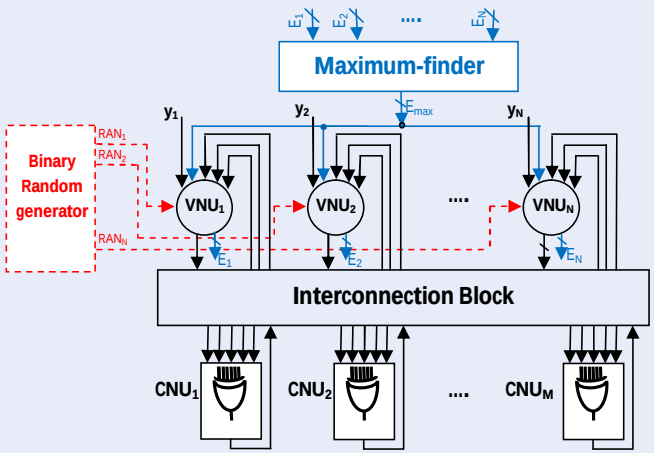
Context & Objectives

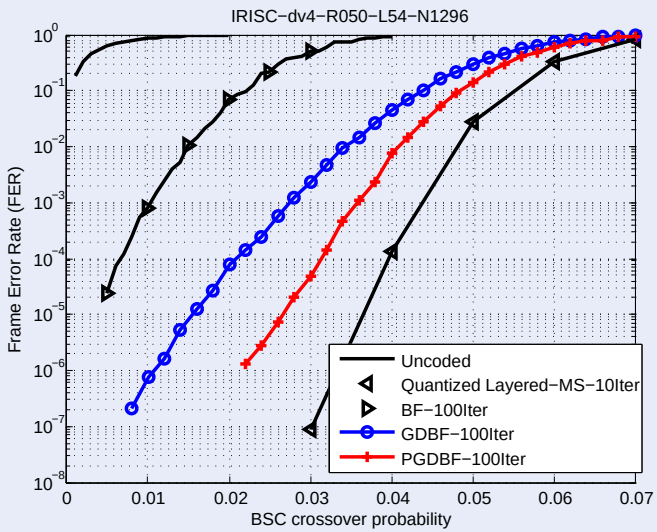
Probabilistic Gradient Descent Bit Flipping (PGDBF) algorithms : GDBF with following modifications

- Energy function : $E_n^{(k)} = x_n^{(k)} \oplus y_n + \sum_{m \in \mathcal{N}(\backslash)} s_m^{(k)}$, with y_n the noisy version of VN n.
- $E_{max}^{(k)} = \text{Max}\left(E_n^{(k)}\right)$,
- sample N random bits $R_n^{(k)}$ from a Bernoulli distribution with probability p_0,
- $x_n^{(k+1)} = \text{flip}\left(x_n^{(k)}\right)$ **if** $E_n^{(k)} = E_{max}^{(k)}$ **and** $R_n^{(k)} = 1$.

[Rasheed2014] O.A. Rasheed, P. Ivanis and B. Vasic, "Fault-Tolerant Probabilistic Gradient-Descent Bit Flipping Decoder ", *Communications Letters, IEEE* , vol. 18, no. 9, pp. 1487–1490, September 2014.

Hardware architecture and performance comparison

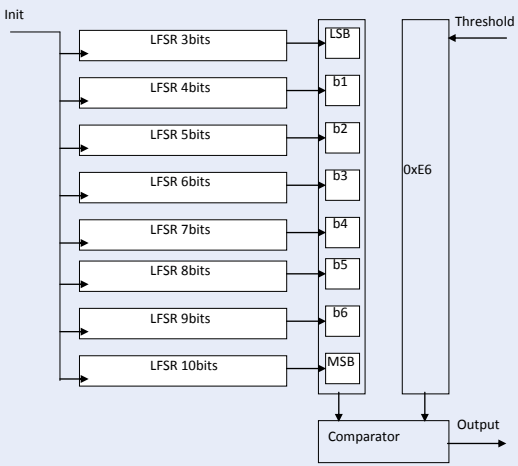




Context & Objectives

Hardware overhead for Random Generator implementation could be prohibitive

Random Generator (RG) : Linear Feedback Shift Register (LFSR)



A Naive implementation of PGDBF would require duplicating N LSFR-RG to get N random bits at each iteration :

	1-bit Register	Slice LUTs
Non-Probabilistic GDBF	946	2151
PGDBF with LFSR	9161	3545
offset min-sum (6 bits)	13694	15350

Our approach

- Statistical study of the PGDBF to identify the critical features of the probabilistic blocks,
- Replace the LSFR-RG by an "approximate" RG with low hardware complexity.

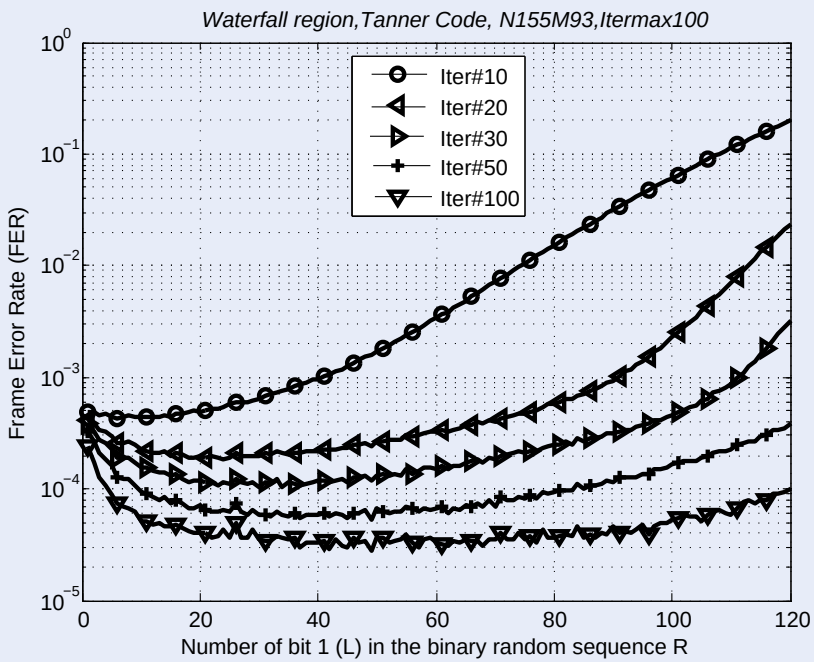
Outline

Statistical Analysis in the Waterfall

Analysis Setting

- Random binary sequence at iteration k : $R^{(k)} = \left\{ R_i^{(k)}, i = 1 \ldots N \right\}$
- Focus on the number L of $1's$ in the sequence $R^{(k)}$, instead of the Bernoulli probability p_0
- Analyse the Frame Error Rate (FER) as a function of L

Binary Symmetric Channel - Waterfall Region - Tanner Code ($M = 93$, $N = 155$)



- **Conclusion 1 :**
  for the first decoding iterations random part does not help,
- **Conclusion 2 :**
  choosing an optimized p_0 is not that important,
  $R^{(k)}$ only need to have a "cut the tail" property ($L_{min} \leq L \leq L_{max}$)

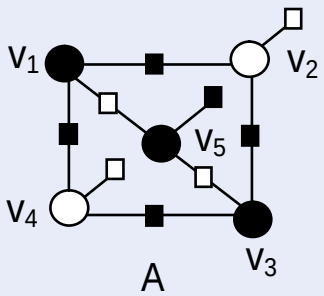
Statistical Analysis in the Error Floor

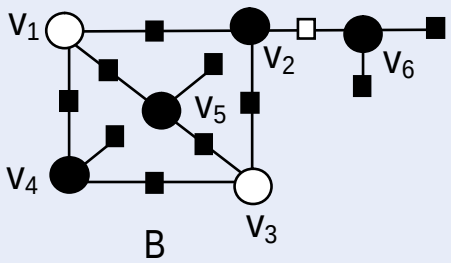
Errors located on Trapping Sets

- In the error floor region, the dominant uncorrectable error configurations are concentrated on Trapping Sets
- Trapping Sets $TS(a, b)$ are defined as a small set of a VNs for which the neighboring CNs contains exactly b odd degree CNs
- $TS(5, 3)$ is the smallest trapping set for regular $d_V = 3$ LDPC codes with girth $g = 8$.

Smallest Error Events **not correctable** by the GDBF

- weight-3 error patterns which does not satisfy 5 parity-checks,
- weight-4 error patterns which does not satisfy 10 parity-checks,

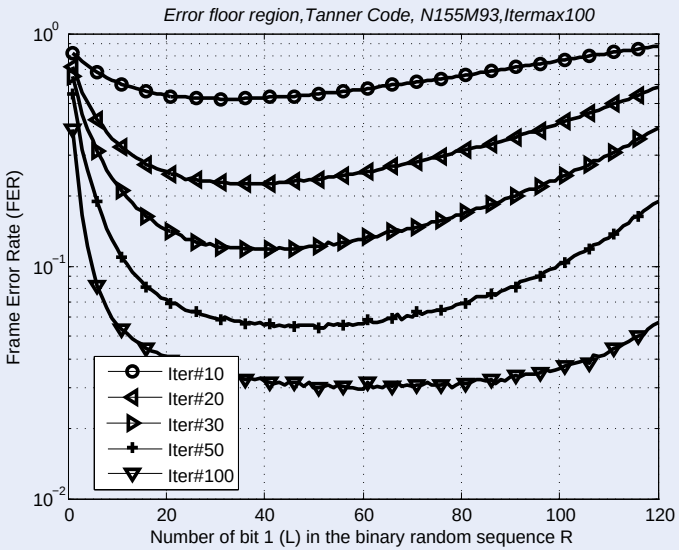




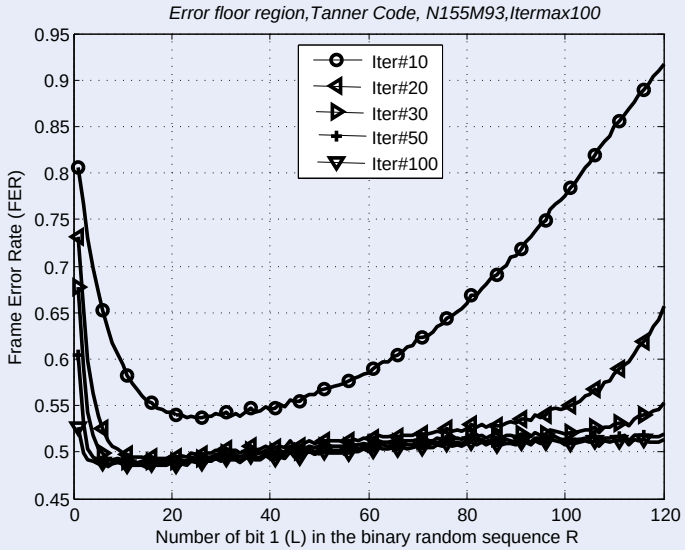
Statistical Analysis in the Error Floor

Frame error rate with fixed input errors

- for each Monte-Carlo round, only the random sequences $R^{(k)}$ differ.



3 bits error pattern



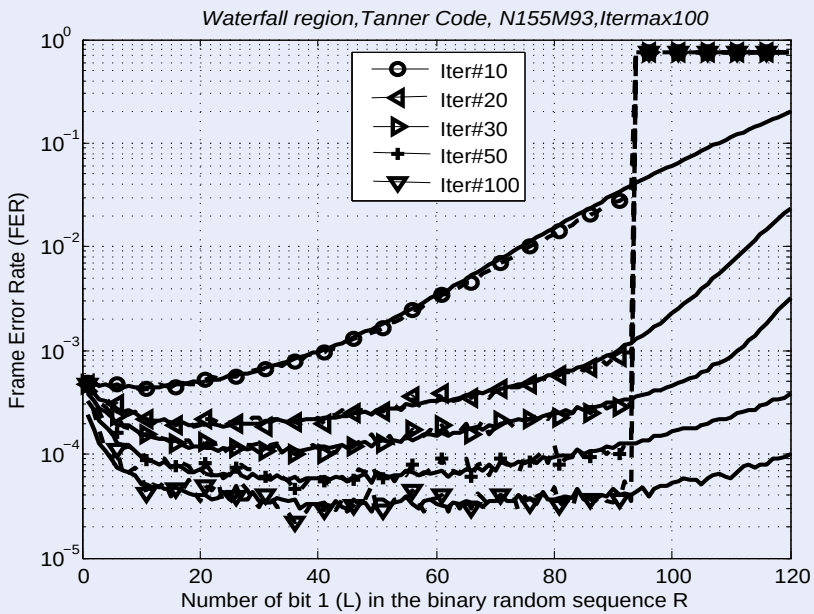
4 bits error pattern

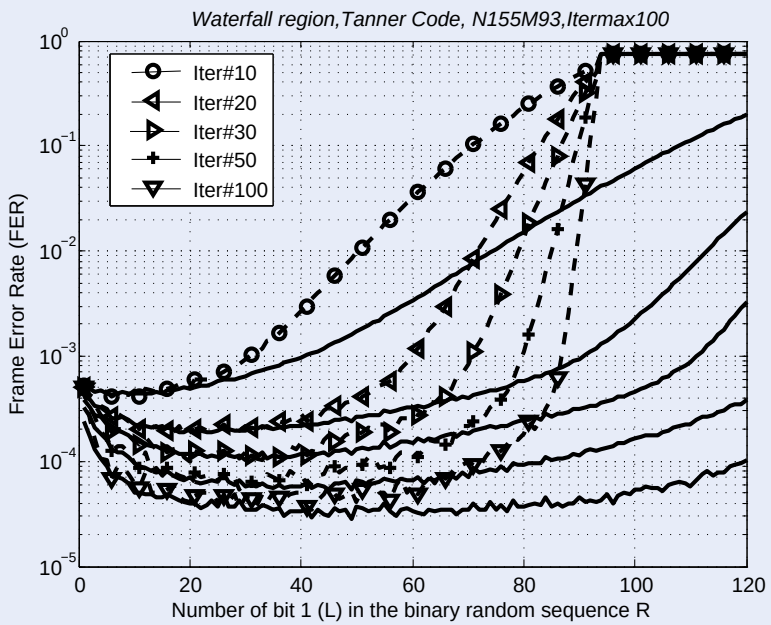
- **Conclusion 1 :** the random part is useful in the first iterations,
- **Conclusion 2 :** $R^{(k)}$ needs to have the "cut the tail" property ($L_{min} \leq L \leq L_{max}$),
- **Concept of decoder rewinding :** replace a single decoder with K iterations with P decoders with K/P iterations, **using same input, but different random seeds** : it is expected to get $FER = (\frac{1}{2})^P$ for the low-weight error events.

Statistical Analysis

Partial PGDBF for QC-LDPC

- Randomness is applied on M/Z random blocks or M/Z fixed blocks out of N/Z (Z : Circulant size).
- Conclusion 1 and 2 are still hold.

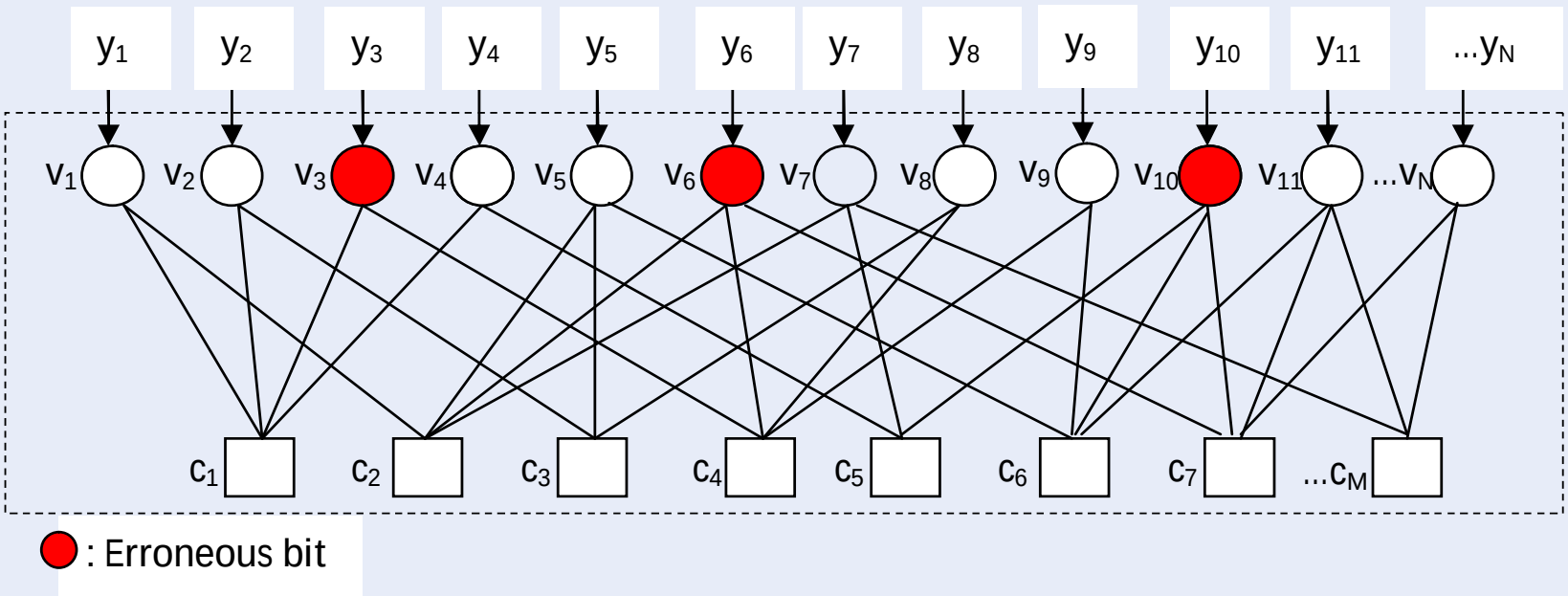




Outline

Intrinsic-Valued Random Generator

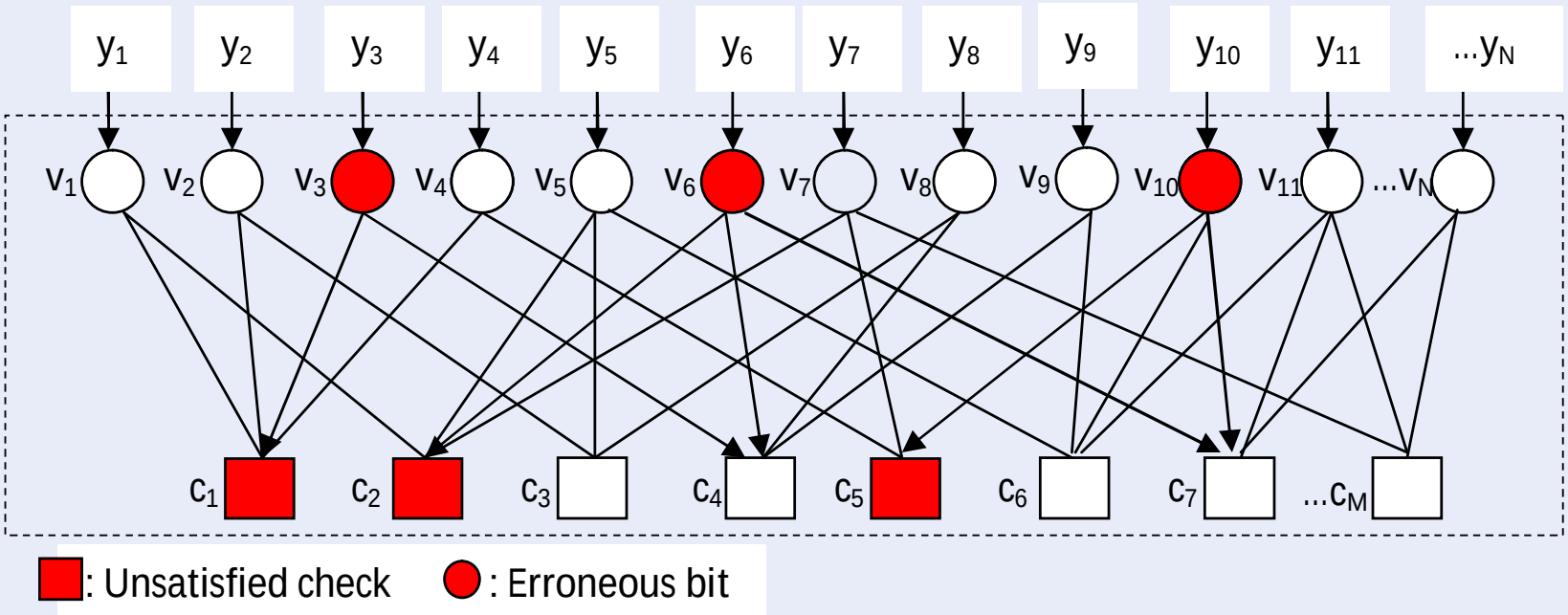
Original decoder :



- All zero codeword sent assumption : $prob(v_i = 1) = \alpha$ and $prob(v_i = 0) = 1 - \alpha$, $i = 1 \ldots N$, α : channel crossover probability

Intrinsic-Valued Random Generator

CNs values sequence at the first iteration :



- $prob(c_j = 1) = 0.5 - (0.5 - 2\alpha)^{dc_j}, j = 1 \ldots M.$
- CNs values sequence $\bar{c} = \left\{ c_j, j = 1 \ldots M \right\}$ is :
  - a sequence of random binary variables
  - generated inside (intrinsic) of the decoder and $L_{\bar{c}}$ has "cut the tail" property as in the following theorem.

Intrinsic-Valued Random Generator

Theorem

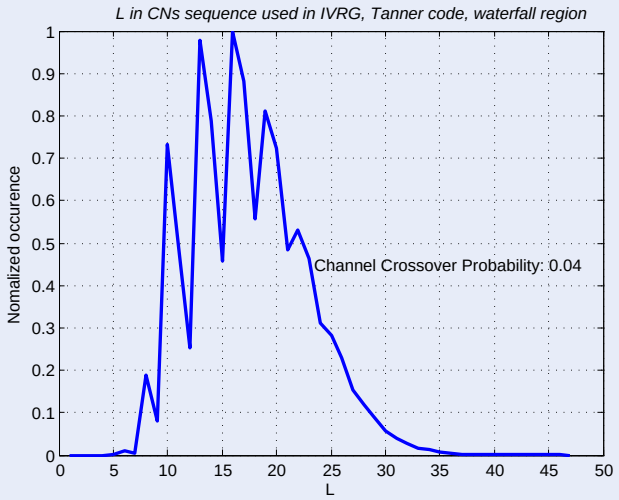
Given an LDPC code having only Trapping Set TS(a, b) in its Tanner graph. If there are a erroneous bits in the whole codeword then L′ is bounded as

$$b \leq L' \leq \sum_{i}^{a} (d_{v(i)})$$

"Cut the tail property of Tanner code in the first iteration CNs sequence"

TABLE: Trapping Sets of Tanner Code (155,93)

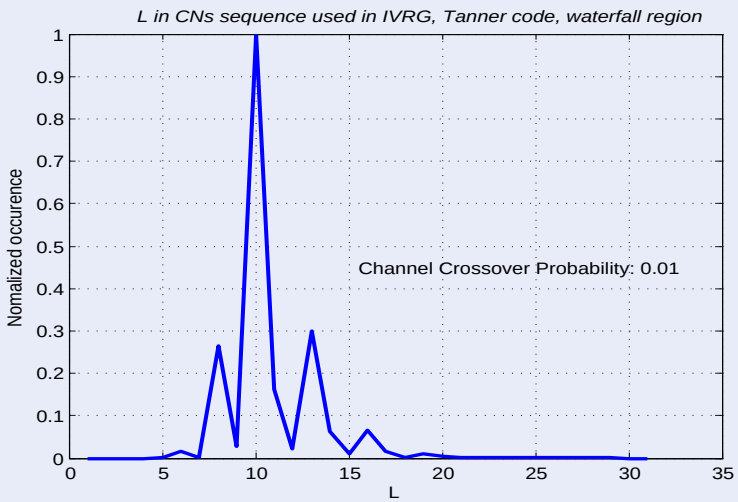
Type of TS	Number of TS
TS(5,3)	155
TS(6,4)	930
TS(7,3)	930
TS(7,5)	13950

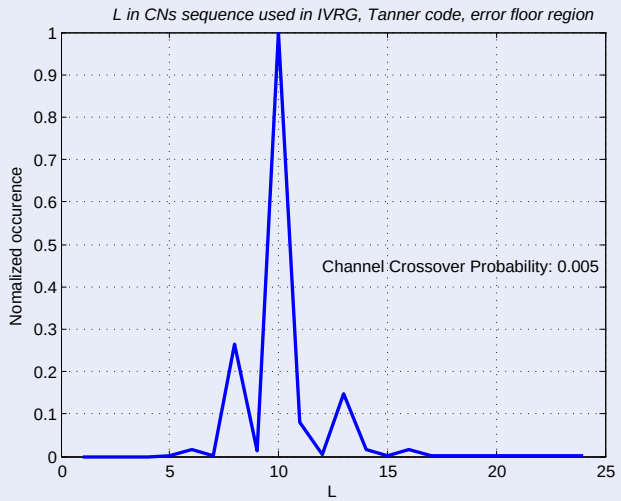


- statistic with number of bits in error $e \geq 3$. (We prove the guaranty error correction $e_g = 2$ for GDBF and PGDBF in another work).

Intrinsic-Valued Random Generator

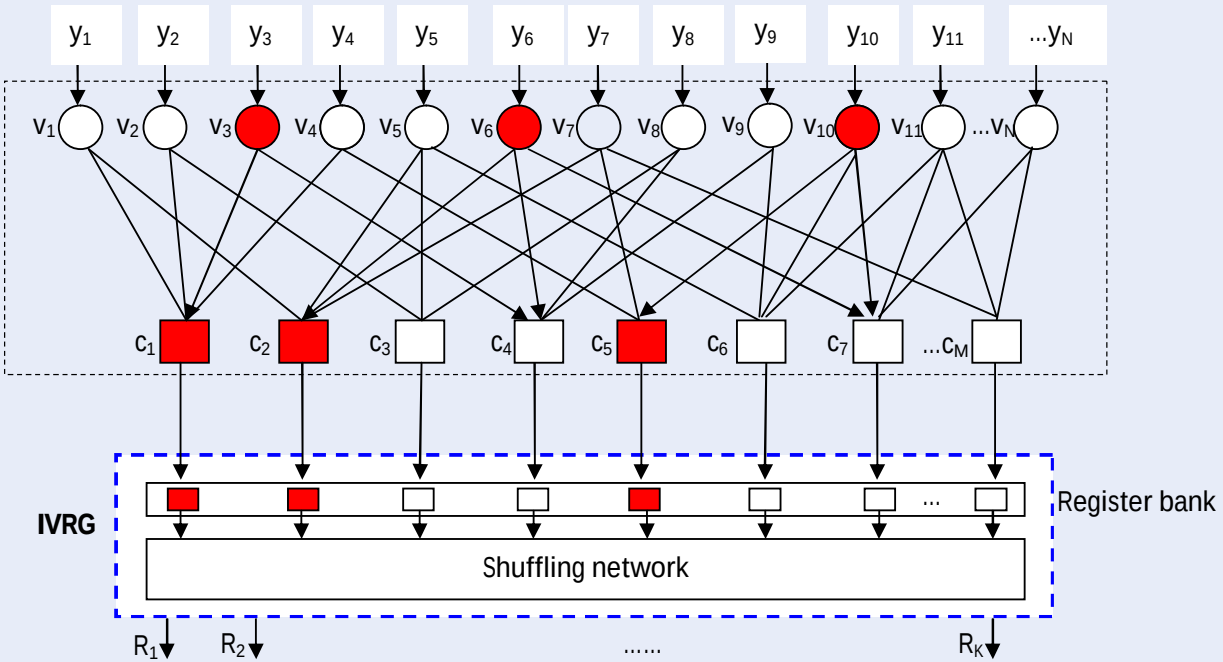
"Cut the tail property of Tanner code in the first iteration CNs sequence"





Intrinsic-Valued Random Generator

IVRG hardware architectures

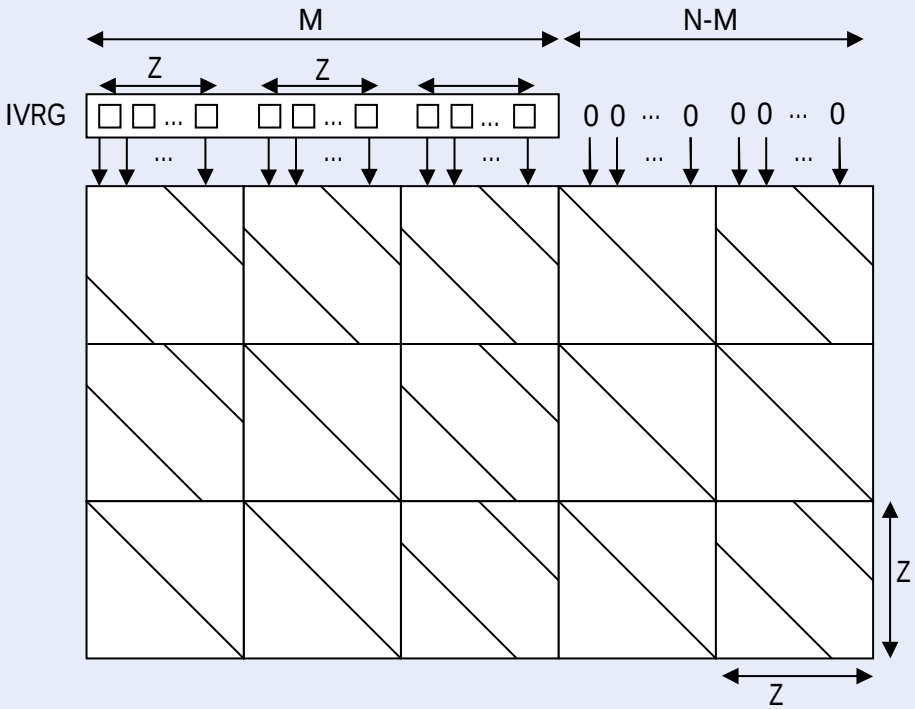


- Store the *M* CNs values in a register bank.
- Shuffle at each iteration these *M* values to generate K outputs :
  - *K* = *N* outputs (Full-IVRG PGDBF decoder)
  - *K* = *M* outputs (Partial IVRG PGDBF decoder)

Intrinsic-Valued Random Generator

Partial IVRG PGDBF

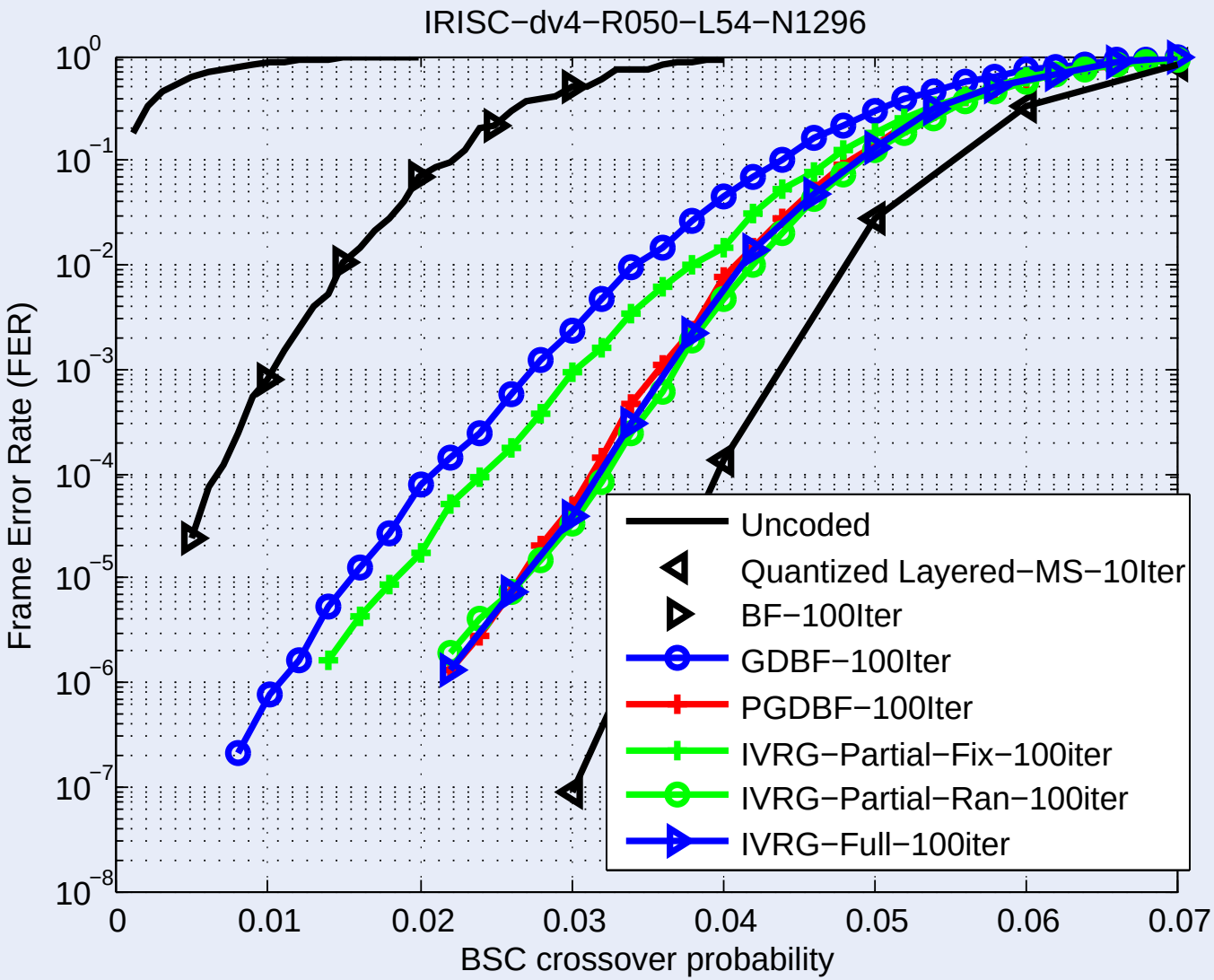
- Further hardware reduction.
- M/Z blocks of IVRG outputs are triggered to :
  - M/Z fixed blocks of QC-LDPC VNs $\rightarrow$ IVRG-Partial-Fix PGDBF decoders.
  - M/Z random blocks of QC-LDPC VNs $\rightarrow$ IVRG-Partial-Ran PGDBF decoders.



Outline

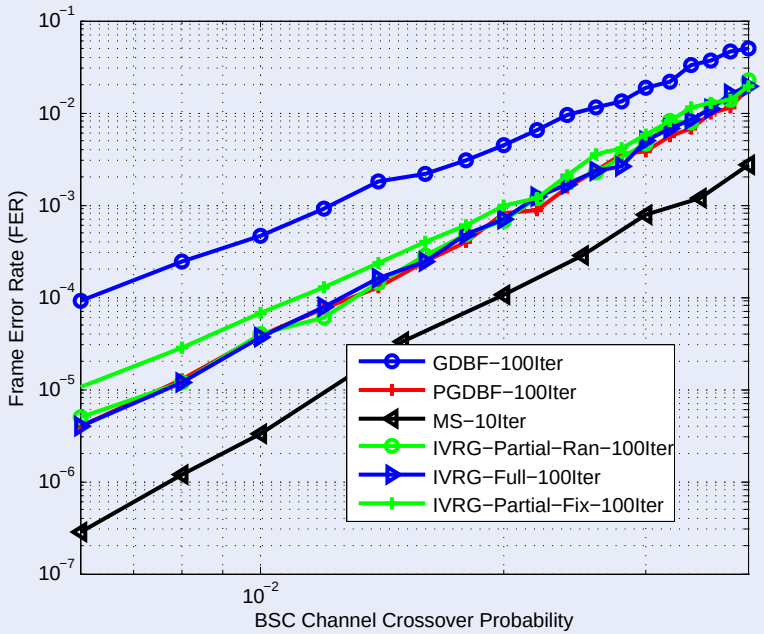
Decoding Performance and Hardware cost

IRISC — dv4 — R050 — L54 — N1296 matrix



Decoding Performance

Tanner – *dv*3 – *L*31 – *N*155 matrix



Harware cost

	1-bit Register	Slice LUTs	F_{max} (MHz)	Throughput (Mbps)
Non-Probabilistic GDBF	946	2151	132.721	4114.3
PGDBF with IVRG	1038	2412	132.721	4114.3
PGDBF with LFSR	9161	3545	135.56	4202.36
offset min-sum (6 bits)	13694	15350	237.185	197.5

Outline

Conclusion

Conclusion

- We introduce a new and original method of random generation called IVRG.
  - Avoid generate but use from the existing decoder memory
  - Highly efficient in hardware cost compared to conventional method (LFSR)
  - Preserving the out standing performance of PGDBF

THANK YOU