Hardware Efficient Implementation of Probabilistic Gradient Descent Bit-Flipping

Fakhreddine Ghaffari, Khoa Le, David Declercq

ETIS, ENSEA/UCP/CNRS France

B. Vasic,

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

June 8, 2016

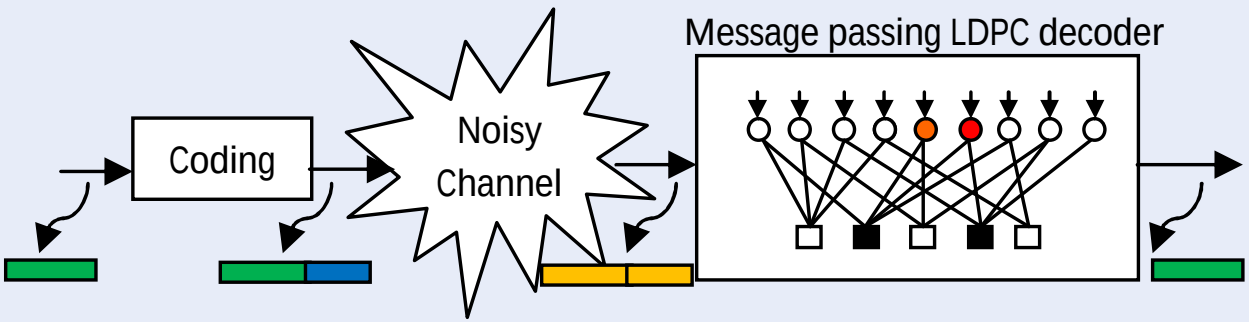
Outline

Outline

Context & Objectives

Low Density Parity Check (LDPC) Decoders

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



2 types of LDPC decoding algorithms :

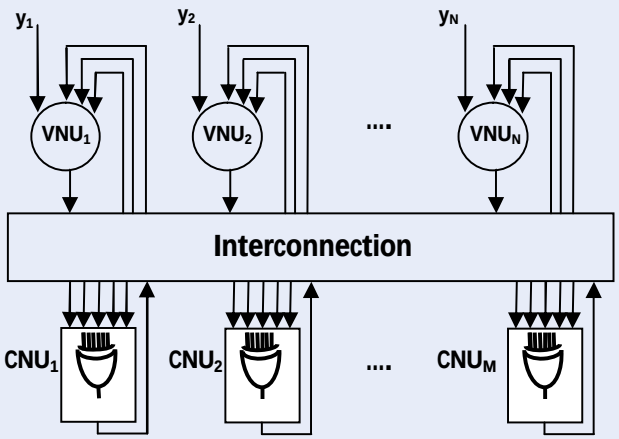
- Soft information algorithms : Sum-Product, Min-Sum..., powerful error correction capacity but high complexity.
- Hard-decision algorithms : Gallager Bit Flipping (BF), Gradient Descent Bit Flipping (GDBF), Probabilistic GDBF..., low complexity, usually weak in error correction.
- We work on hard-bit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

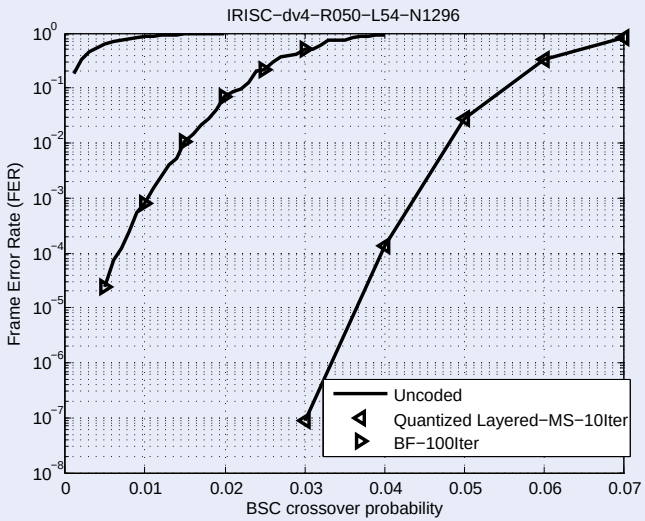
Context & Objectives

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** $E_n^{(k)} = \sum_{m \in \mathcal{N}(\setminus)} s_m^{(k)} \geq \tau$, 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



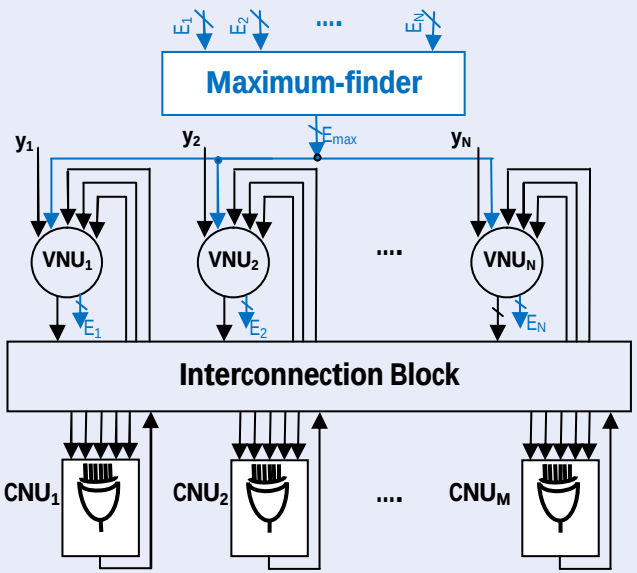


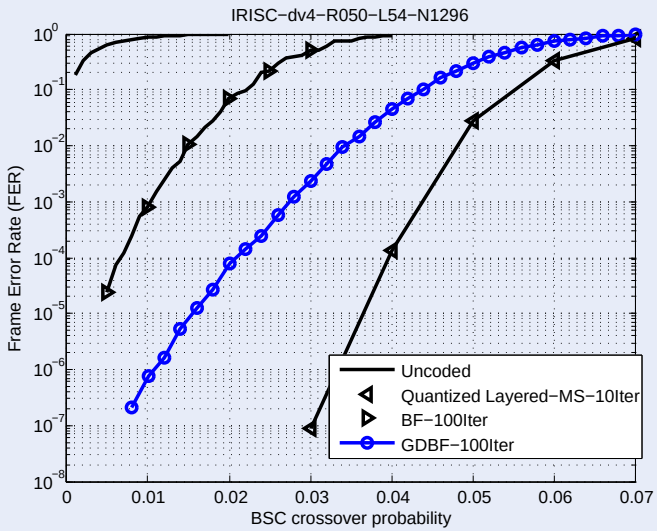
Context & Objectives

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





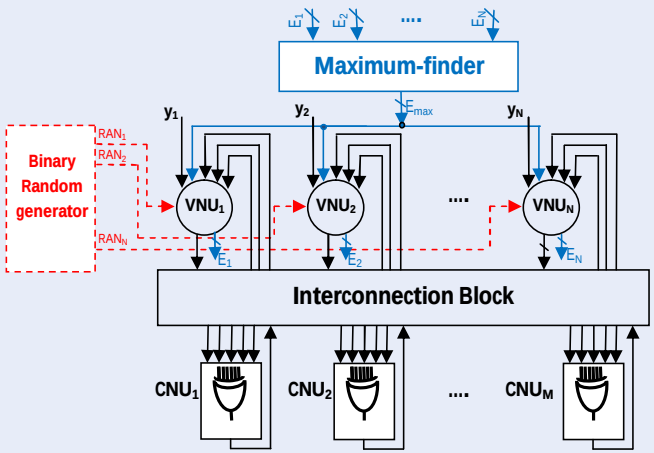
Context & Objectives

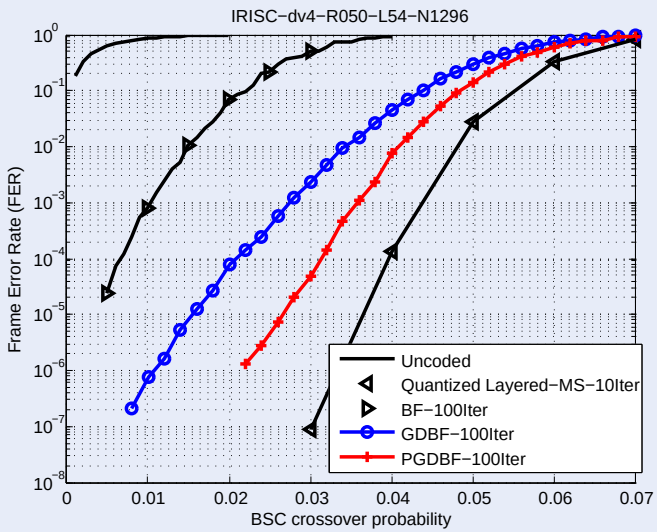
Probabilistic Gradient Descent Bit Flipping (PGDBF) algorithms : GDBF with following modifications

- 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

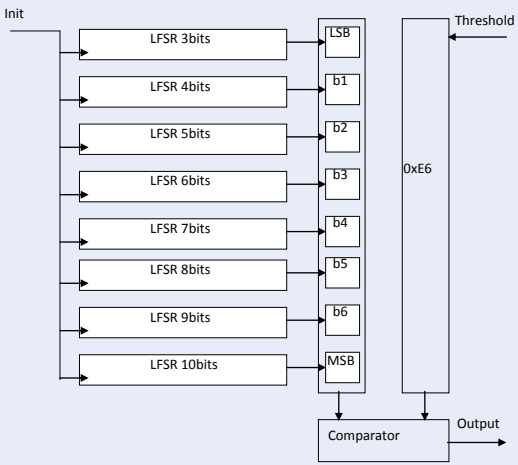




Context & Objectives

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 LFSR-RG to get N random bits at each iteration :

Tanner code (155,93, dv = 3, dc = 5)		
	1-bit Register	Slice LUT
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 LFSR-RG by an "approximate" RG with low hardware complexity.

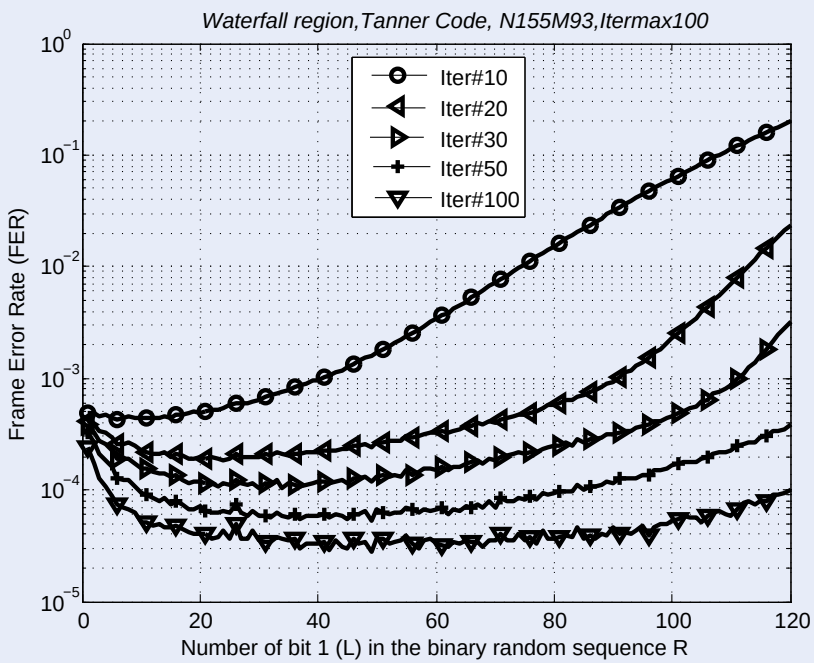
Outline

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 ($L_{min} \leq L \leq L_{max}$)

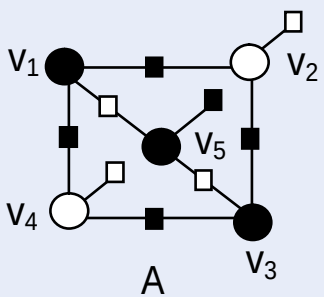
Statistical Analysis in the Error Floor

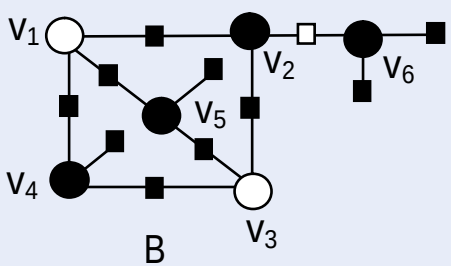
Errors located on Trapping Sets

- 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,

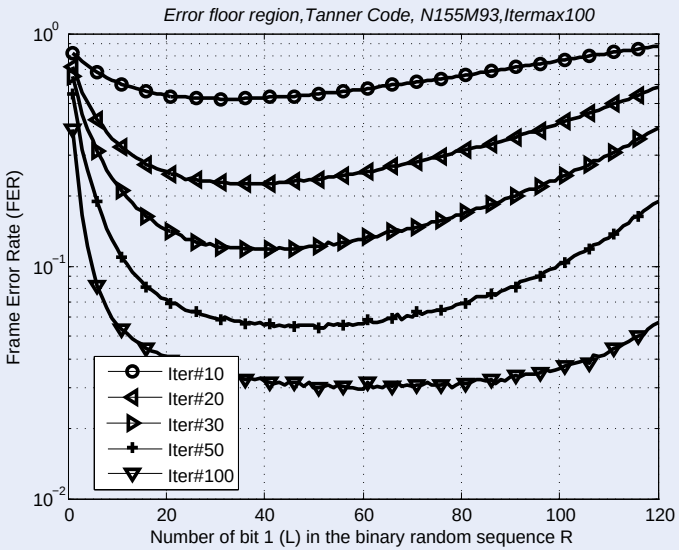




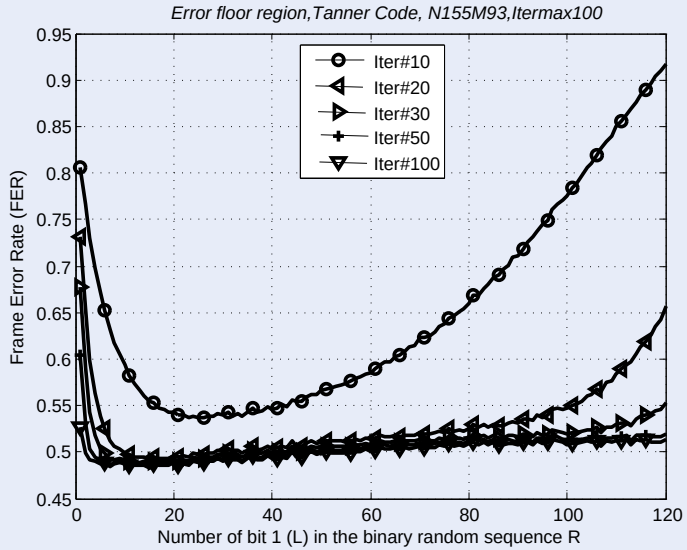
Statistical Analysis in the Error Floor

Frame error rate with fixed input errors

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



3 bits error pattern

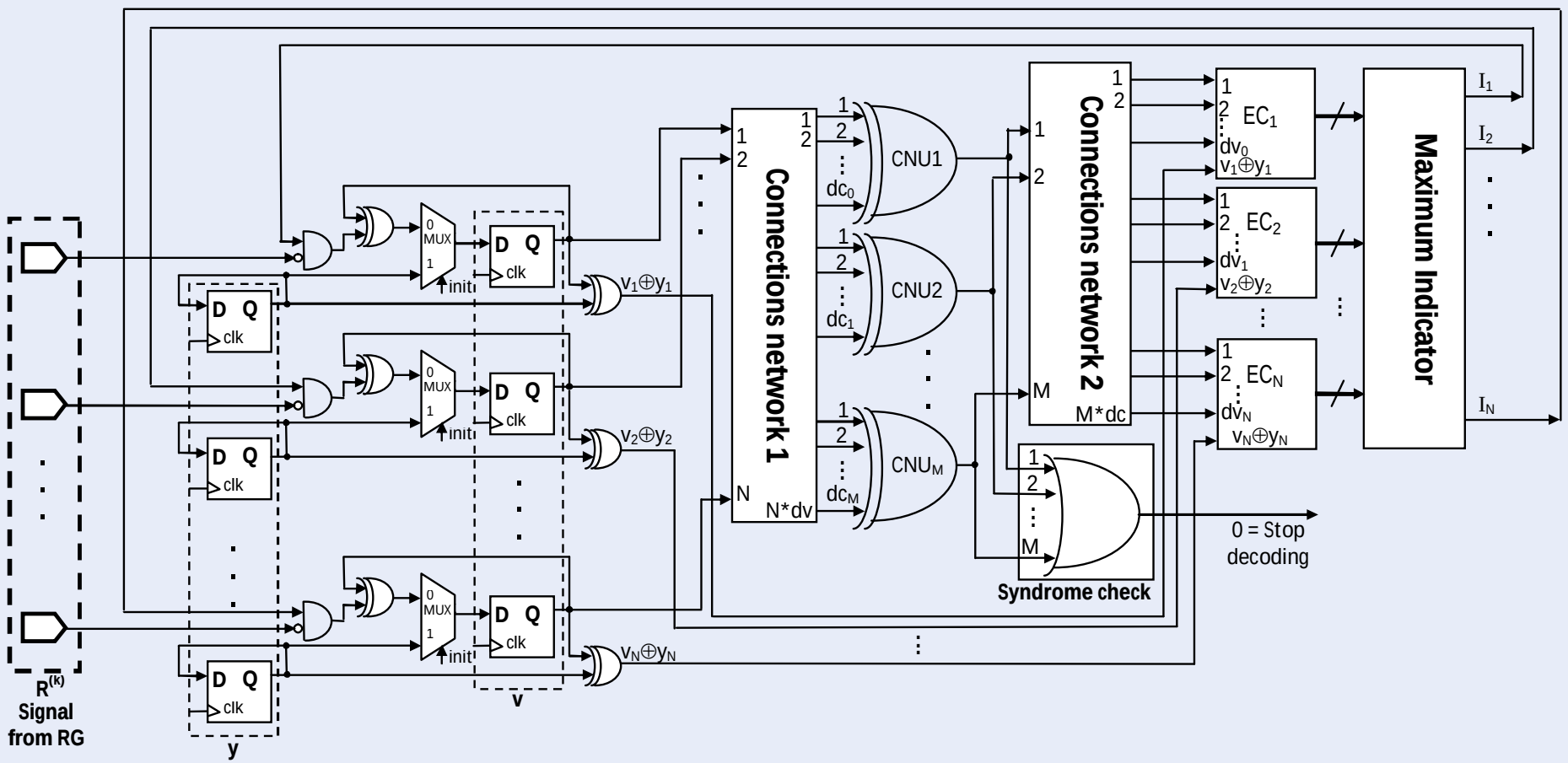


4 bits error pattern

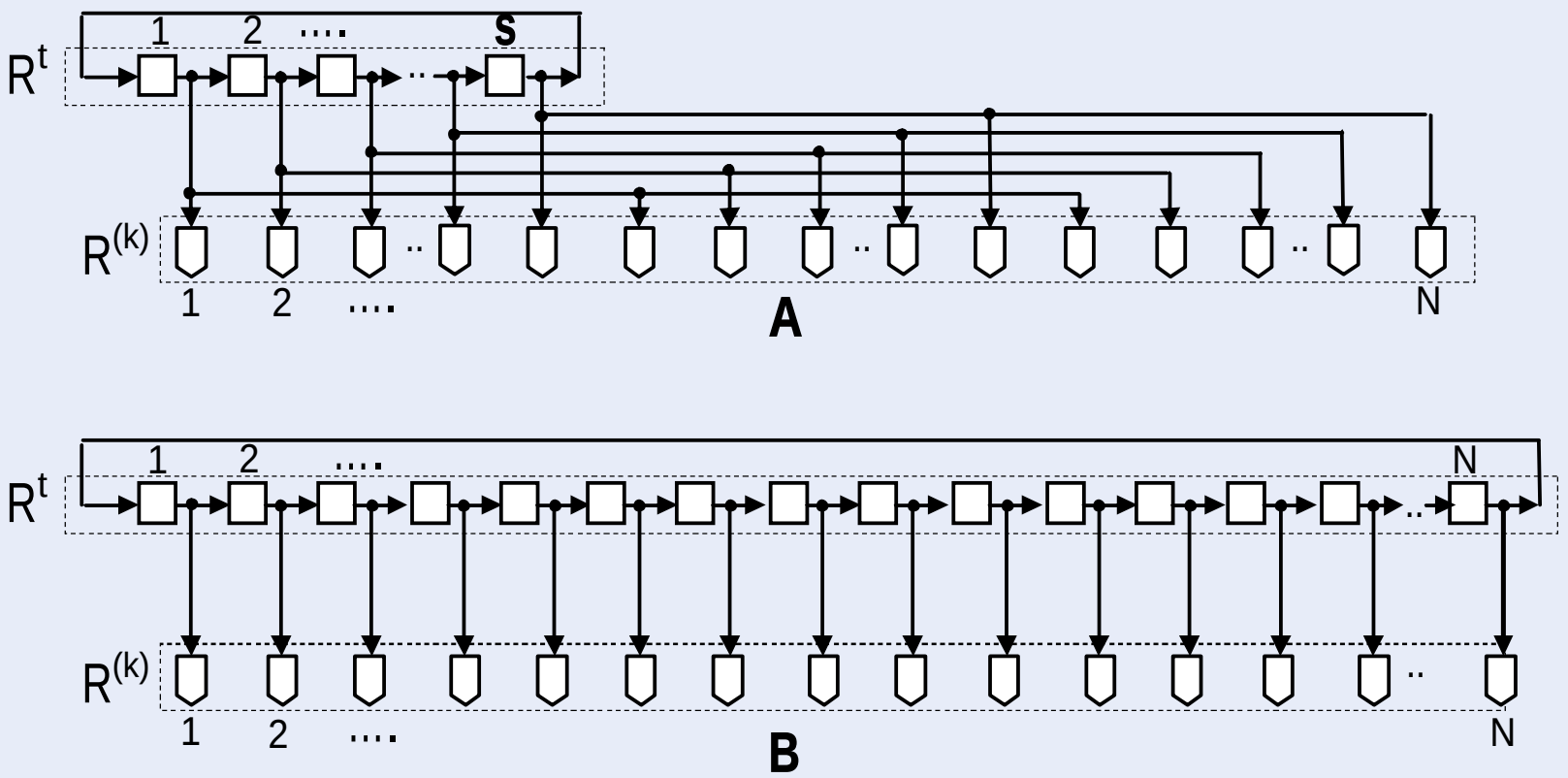
- **Conclusion 1 :** the random part is useful in the first iterations,
- **Conclusion 2 :** $R^{(k)}$ needs to have the "cut the tail" property ($L_{min} \leq L \leq L_{max}$),
- **Concept of decoder rewinding :** replace a single decoder with K iterations with P decoders with K/P iterations, **using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

Outline

PGDBF Implementation Architecture



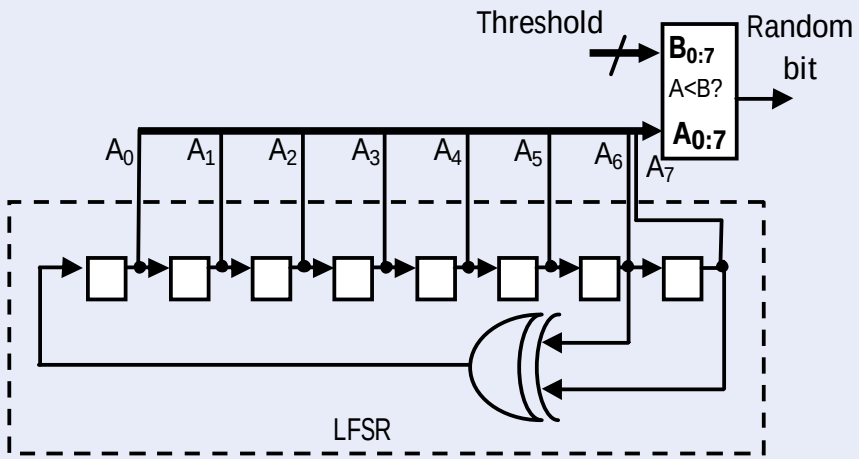
Simplified Random Generator - Duplication approach

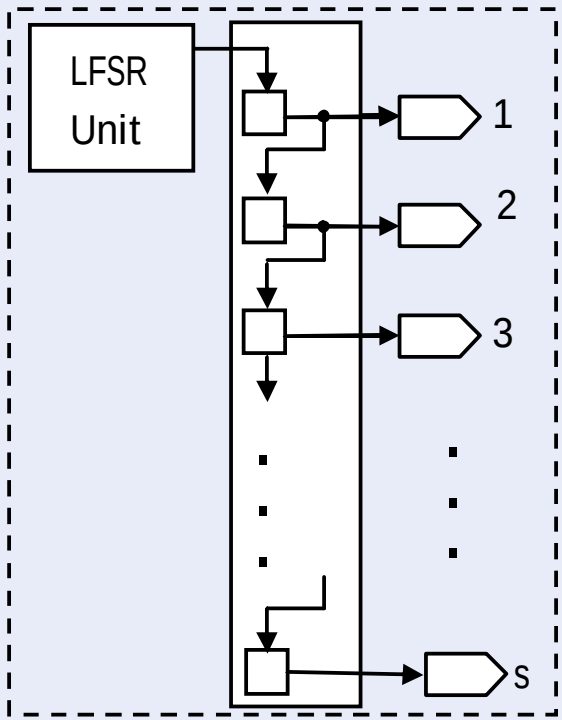


- Initialize the R^t by :
  - Linear Feedback Shift Register (LFSR)
  - Intrinsic-Valued Random Generator (IVRG)

Outline

LFSR



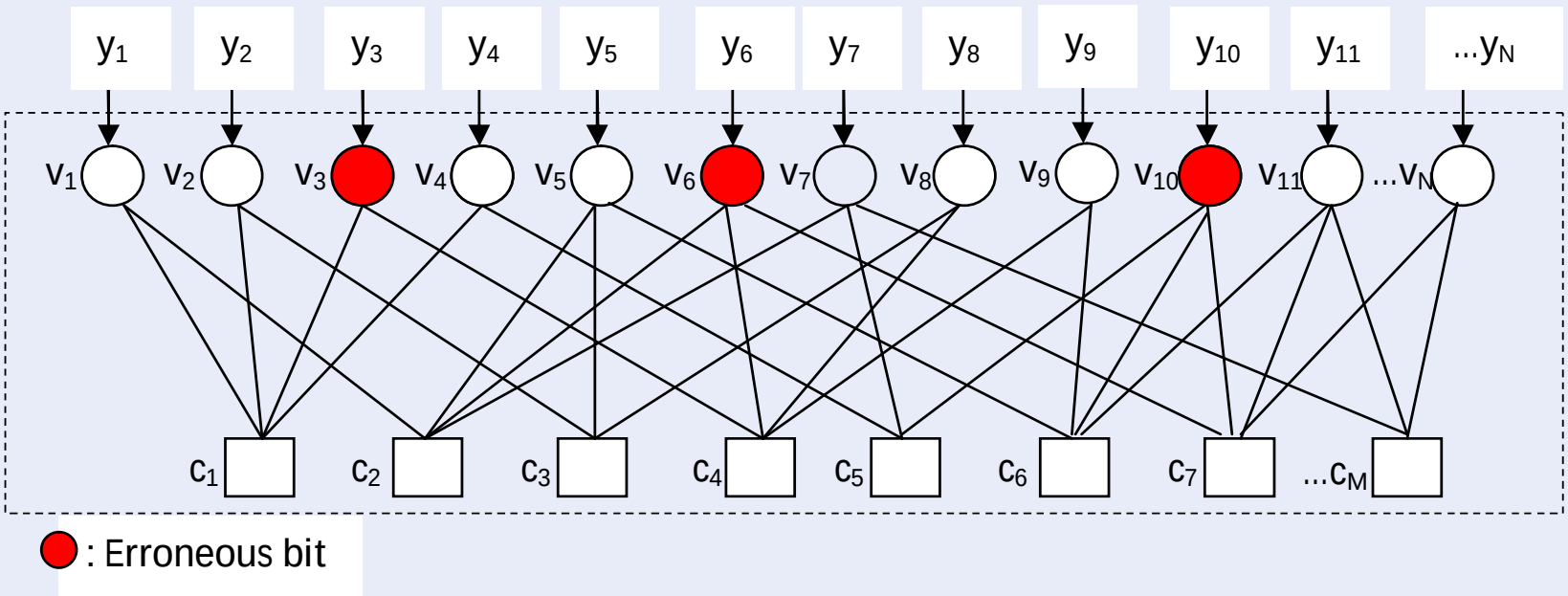


LFSR RG Module

Outline

Intrinsic-Valued Random Generator

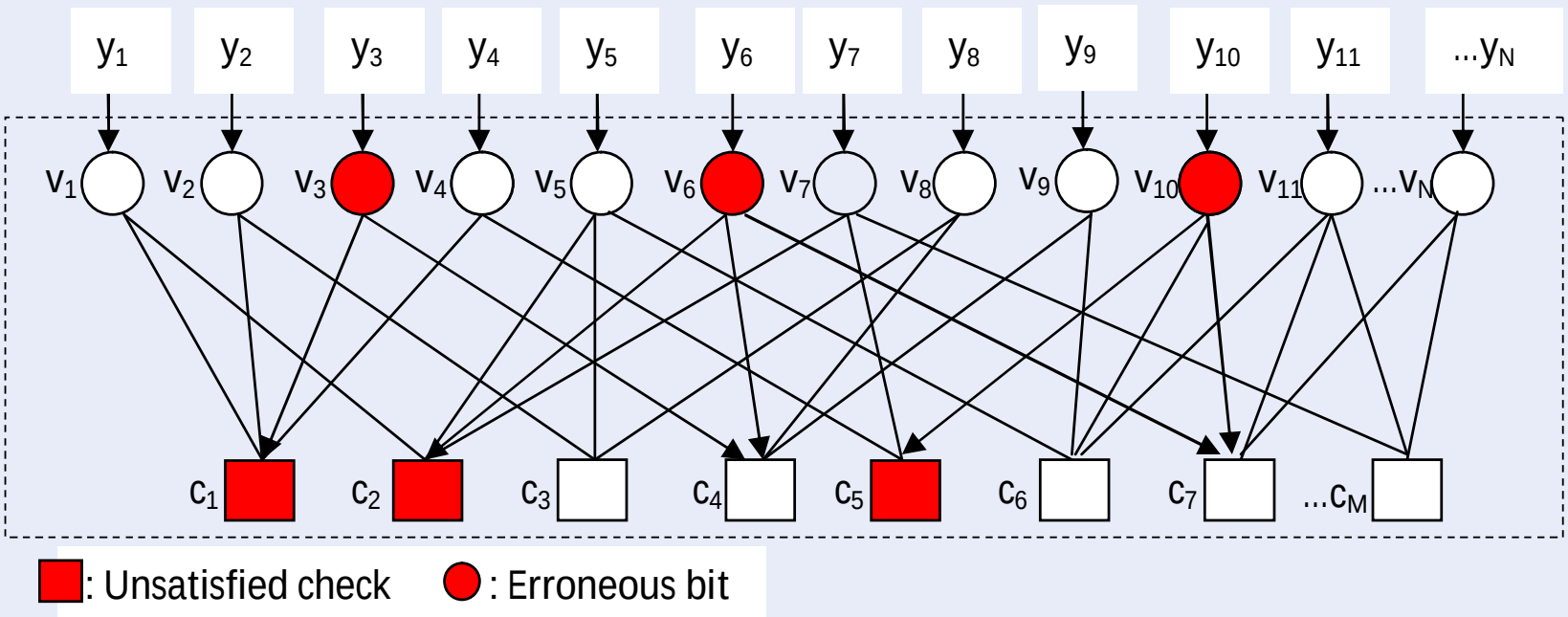
Original decoder :



- 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

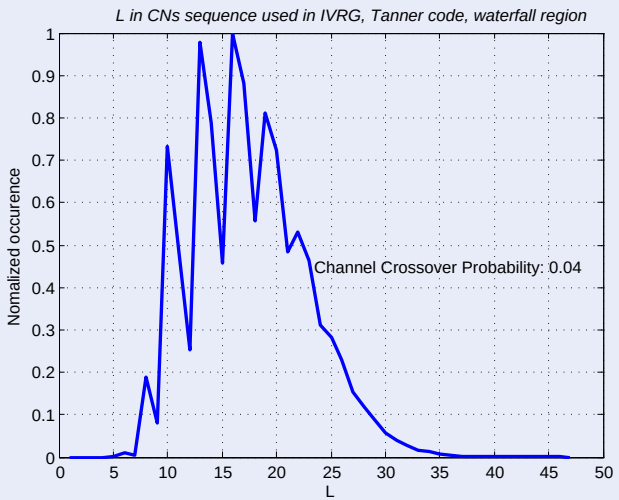
Given an LDPC code having only Trapping Set TS(a, b) in its Tanner graph. If there are a erroneous bits in the whole codeword then L′ is bounded as

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

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

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

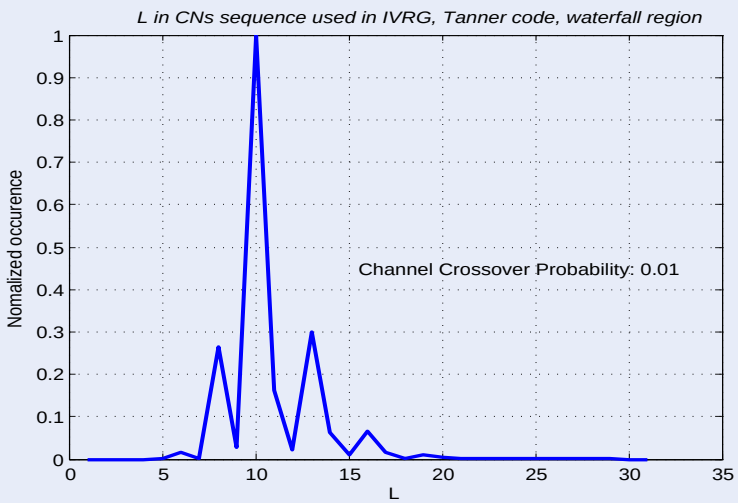
Type of TS	Number of TS
TS(5,3)	155
TS(6,4)	930
TS(7,3)	930
TS(7,5)	13950

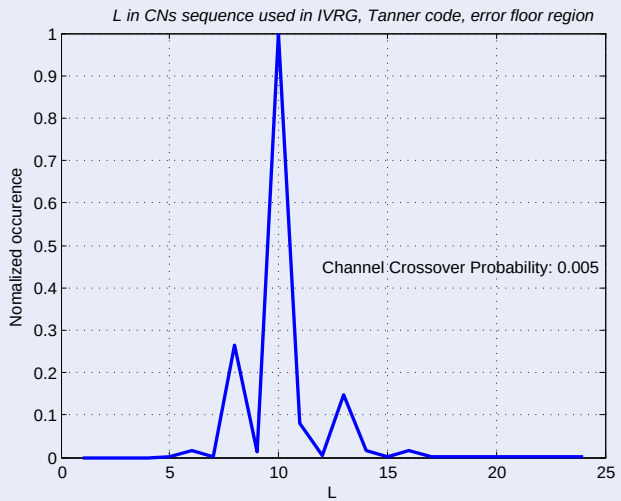


- statistic with number of bits in error $e \geq 3$. (We prove the guaranty error correction $e_g = 2$ for GDBF and PGDBF in another work).

Intrinsic-Valued Random Generator

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





Intrinsic-Valued Random Generator

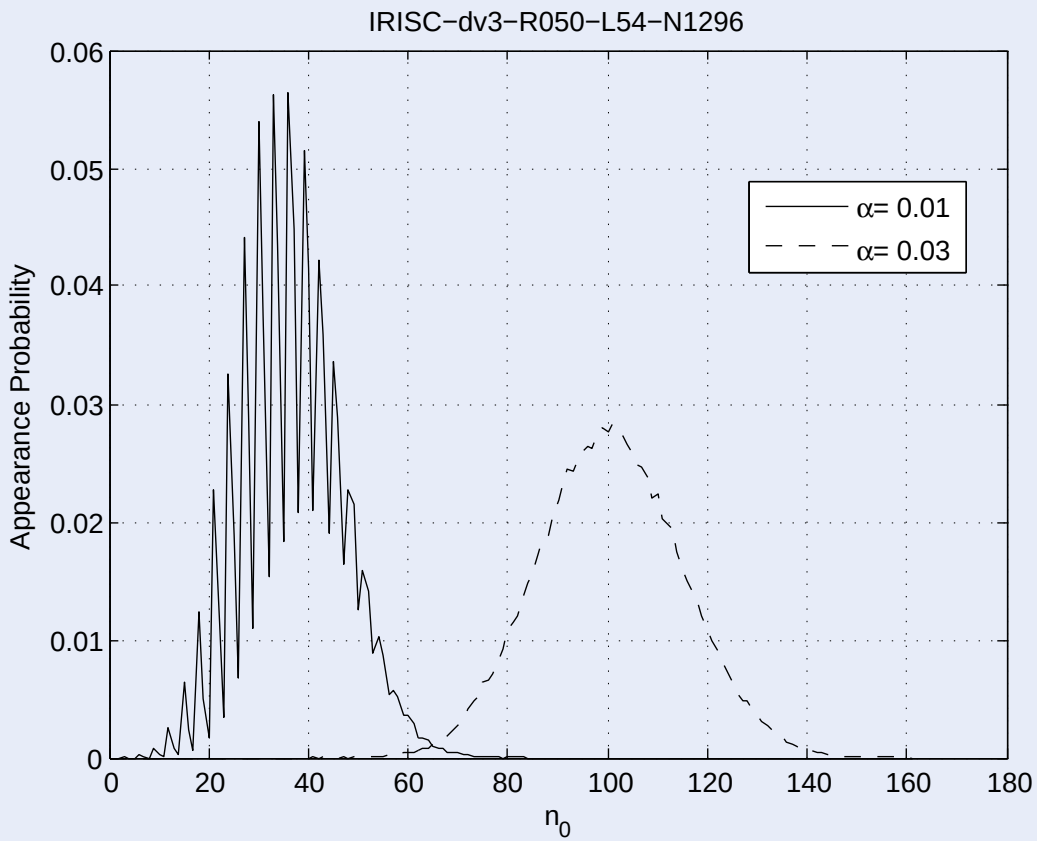
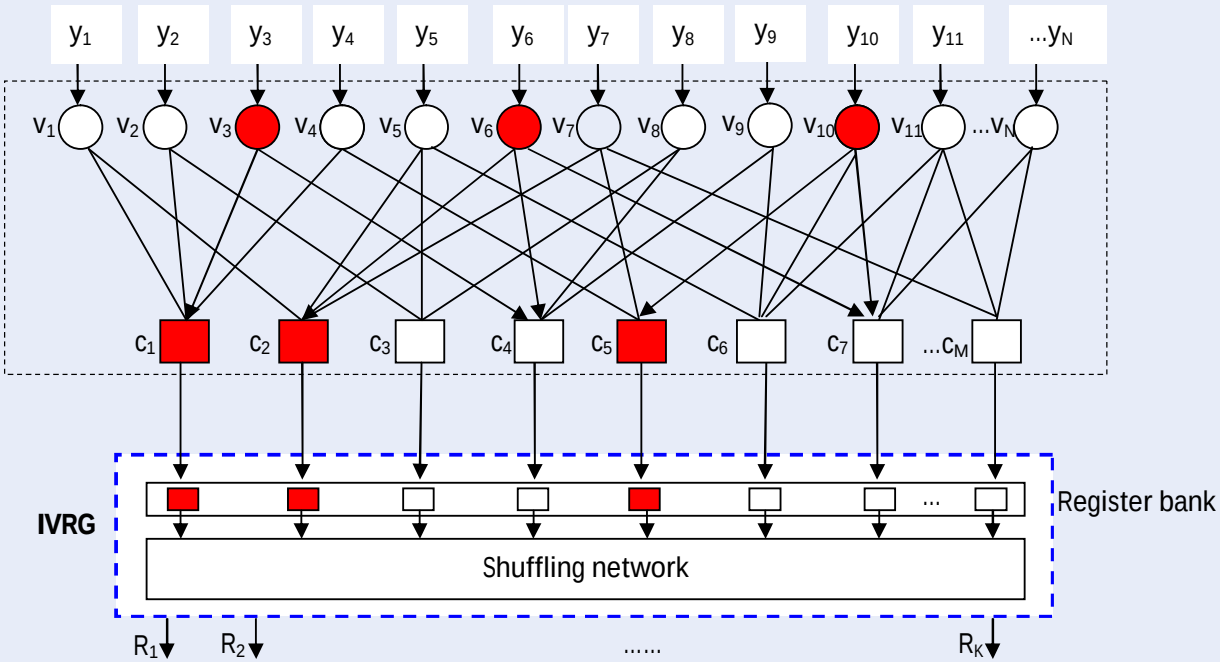


FIGURE: Cut the tail property of CNs sequence in the first iteration $d_v = 3$, $d_c = 6$, code rate $R = 0.5$, $N = 1296$

Intrinsic-Valued Random Generator

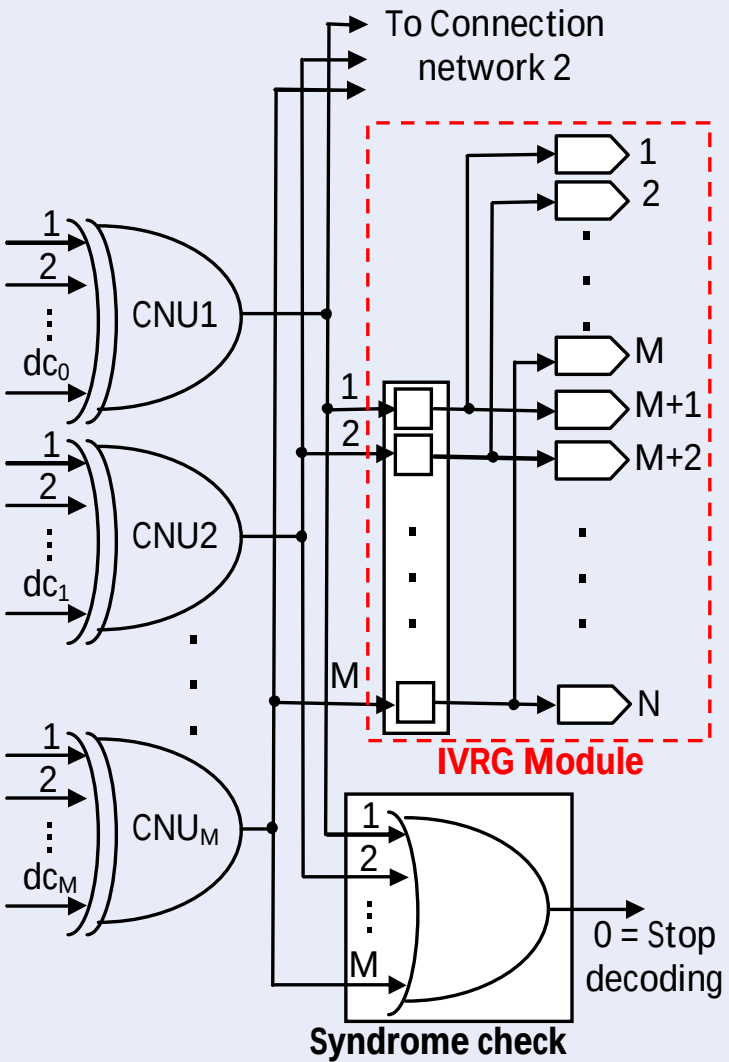
IVRG hardware architectures



- Store the *S* CNs values in a register bank.
- Shuffle at each iteration these *S* values to generate *N* outputs :

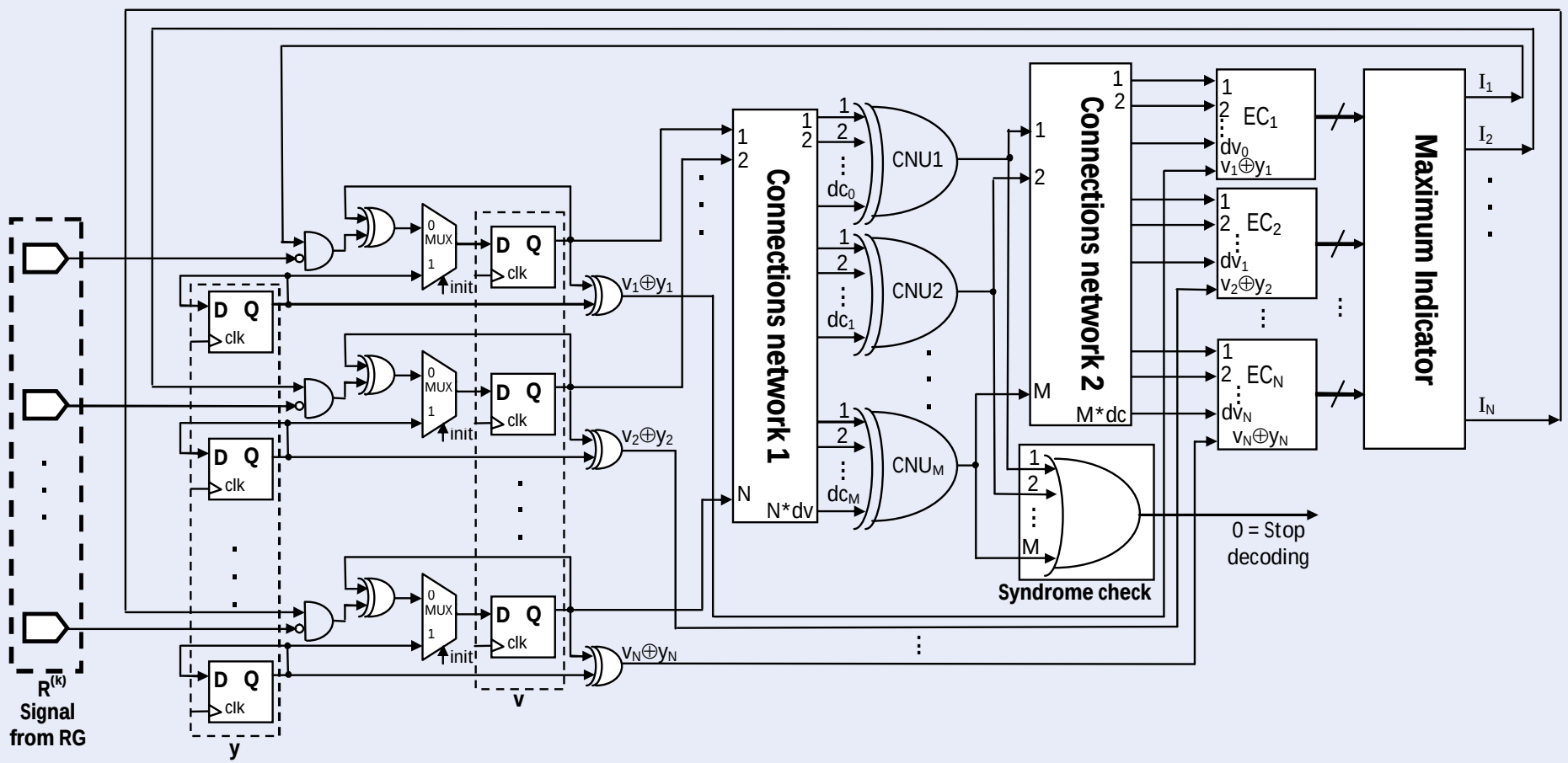
Intrinsic-Valued Random Generator

IVRG hardware architectures



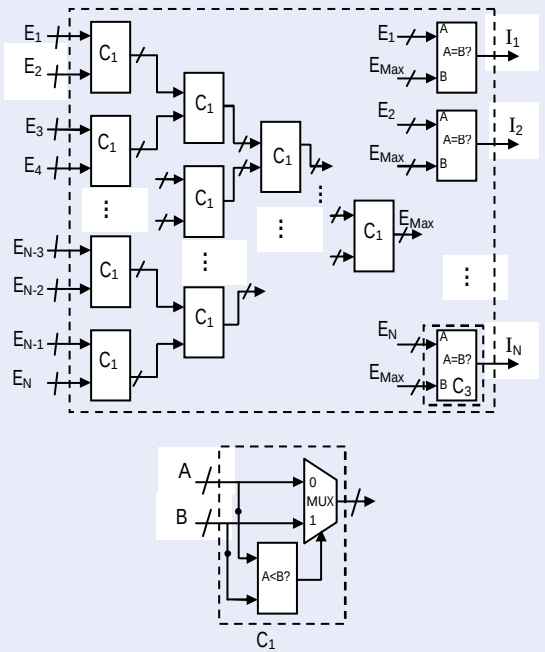
Outline

PGDBF Implementation Architecture



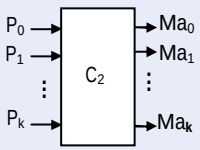
Maximum Indicator (MI)

Traditional MI



Maximum Indicator (MI)

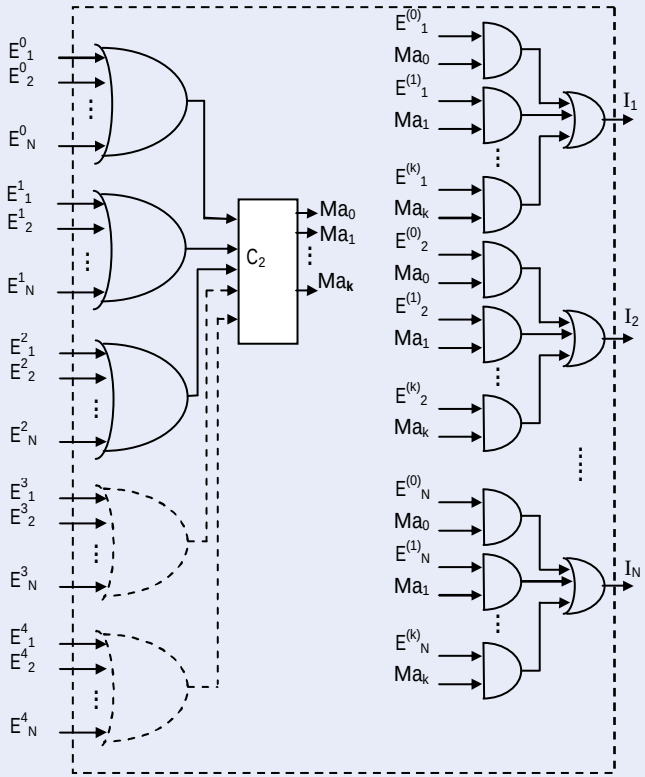
Conventional LCT



C2: "1 in k" code converter

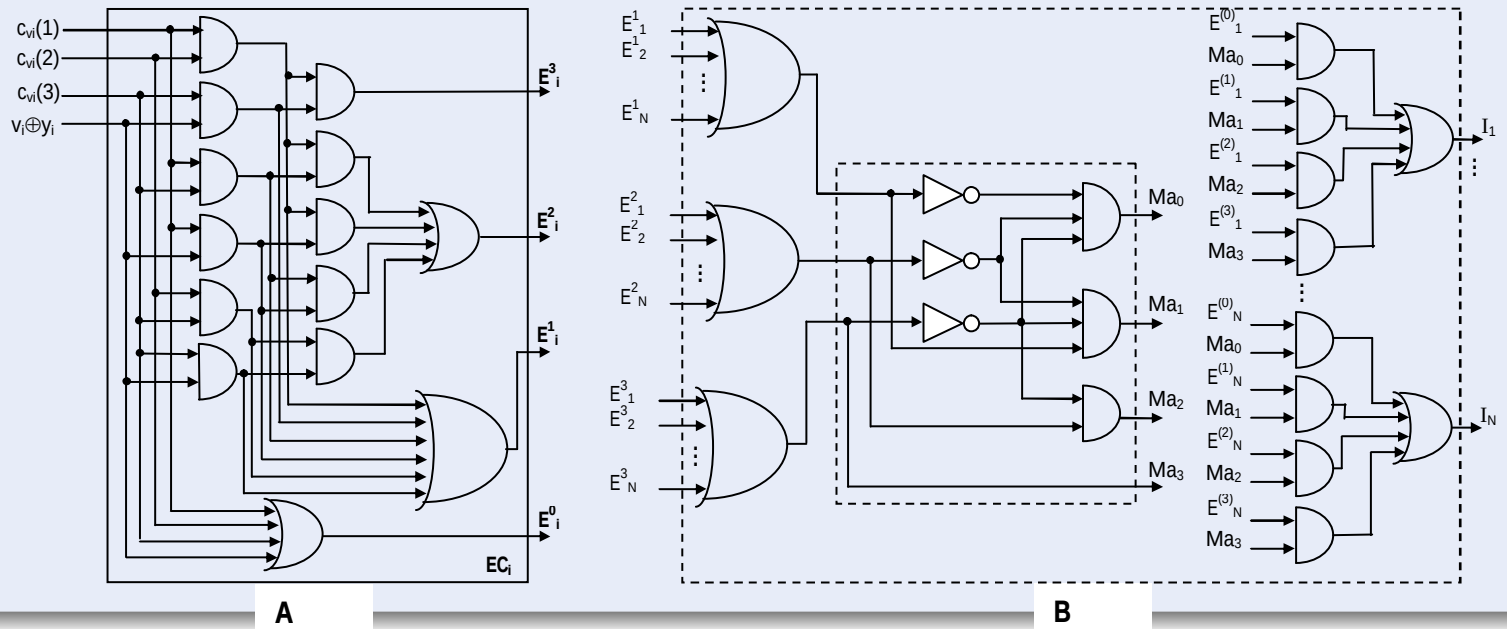
$Ma_i = 1$ **if** ($P_i = 1$ and $P_j = 0$ for all $i < j \leq k$)

Otherwise $Ma_i = 0$



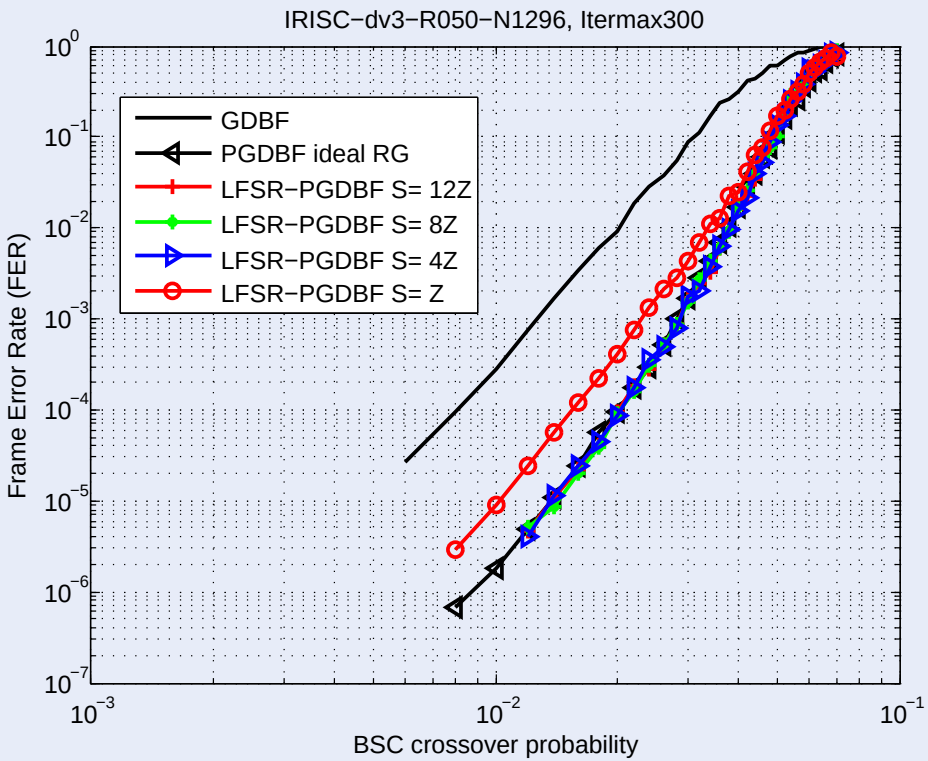
Maximum Indicator (MI)

Modified-LCT



Outline

Decoding Performance



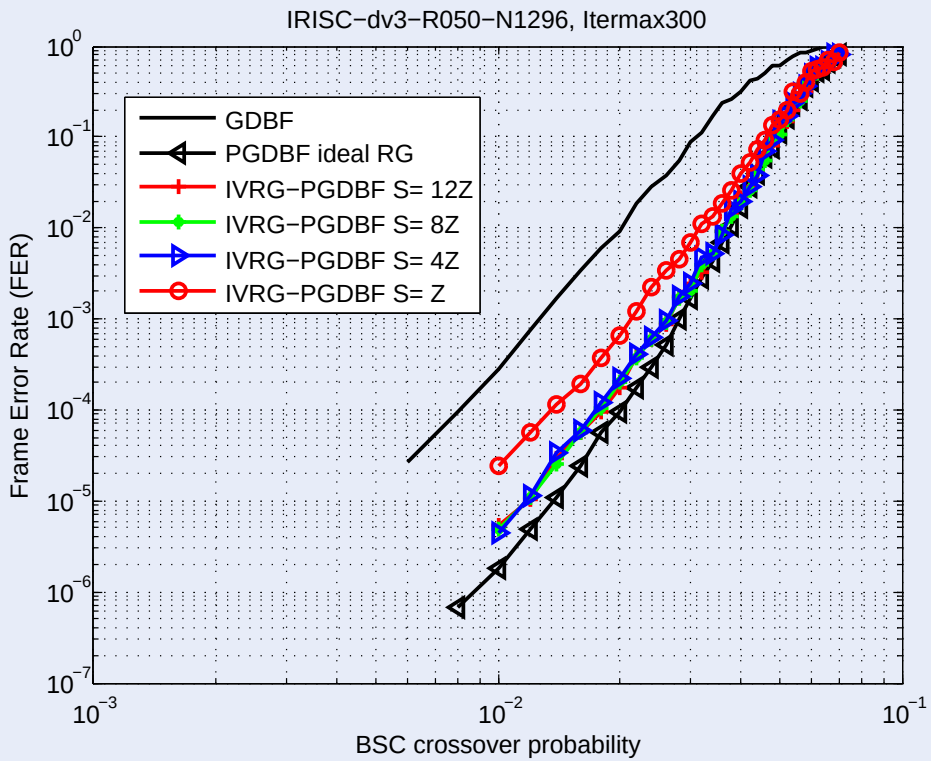


FIGURE: Decoding performance of LFSR-PGDBF, IVRG-PGDBF decoders with different values of S on the LDPC code $d_v = 3$, $d_c = 6$, code rate $R = 0.5$, $N = 1296$

Average number of iteration

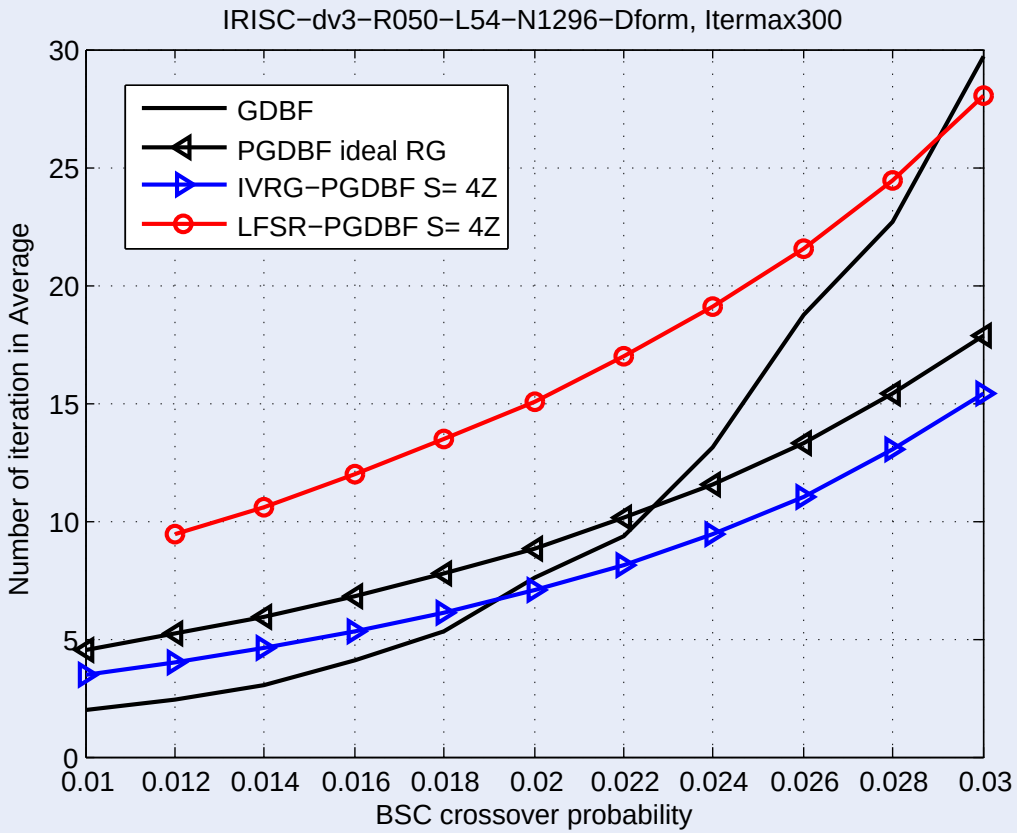
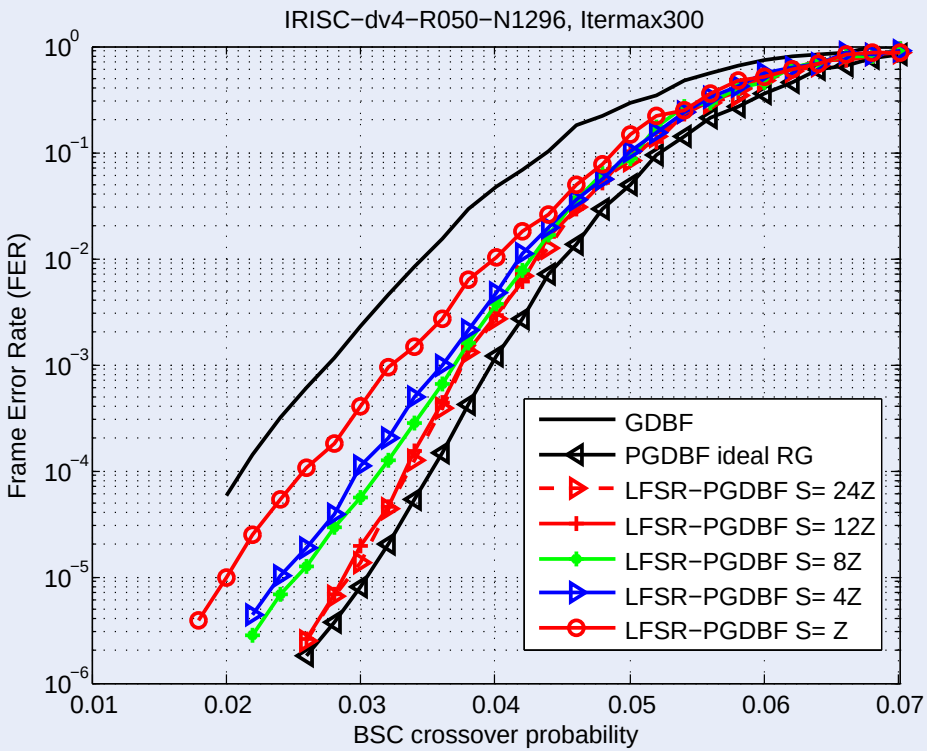


FIGURE: The average number of iteration of GDBF and PGDBF decoders on the LDPC code $d_v = 3$, $d_c = 6$, code rate $R = 0.5$, $N = 1296$

Decoding Performance



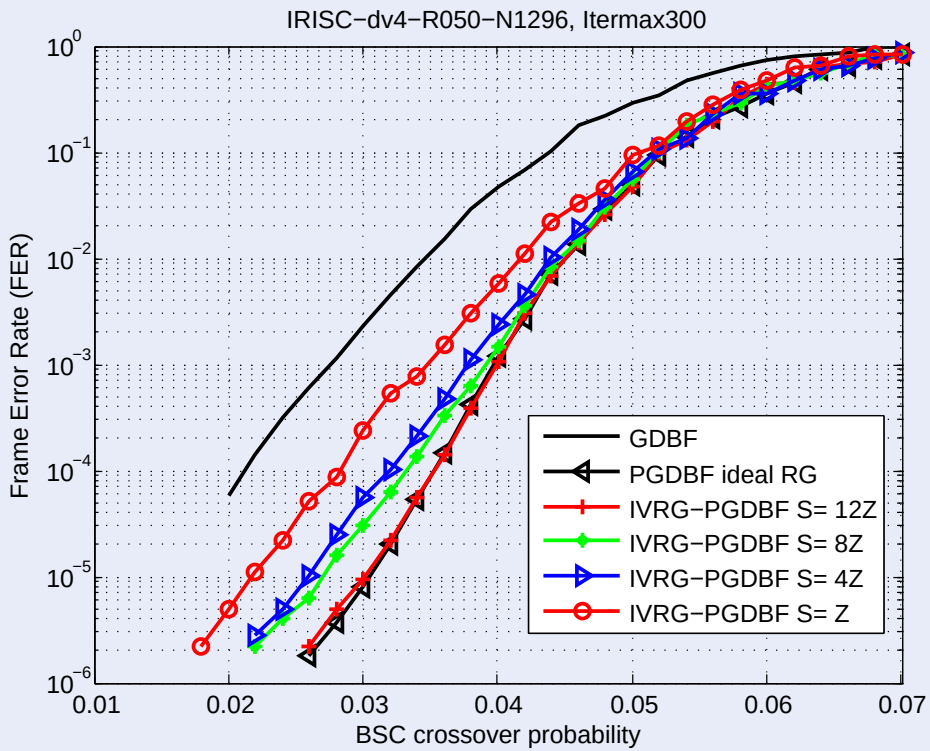


FIGURE: Decoding performance of LFSR-PGDBF, IVRG-PGDBF decoders with different values of S on the LDPC code $d_v = 4$, $d_c = 8$, code rate $R = 0.5$, $N = 1296$

Average number of iteration

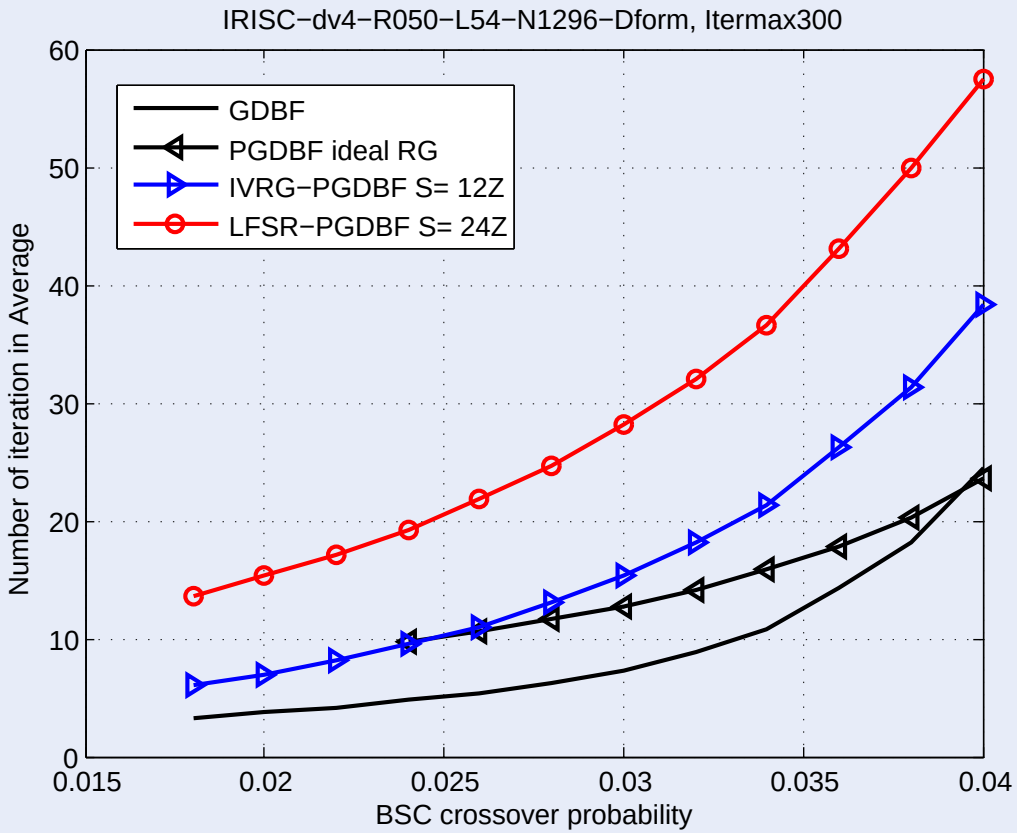
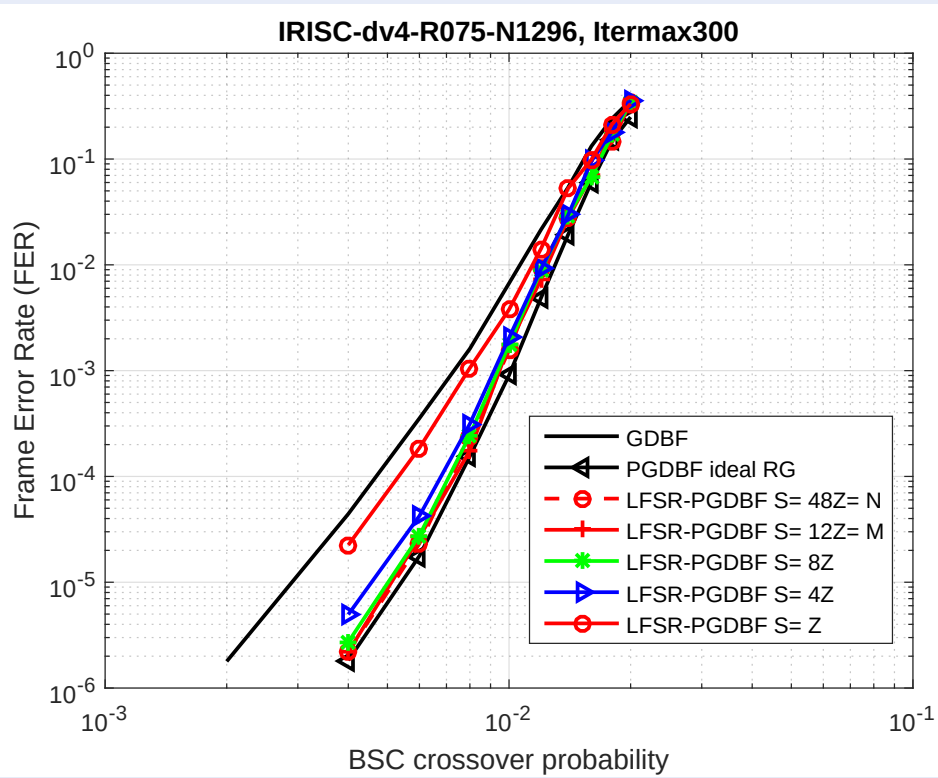


FIGURE: The average number of iteration of GDBF and PGDBF decoders on the LDPC code $d_v = 4$, $d_c = 8$, code rate $R = 0.5$, $N = 1296$

Decoding Performance



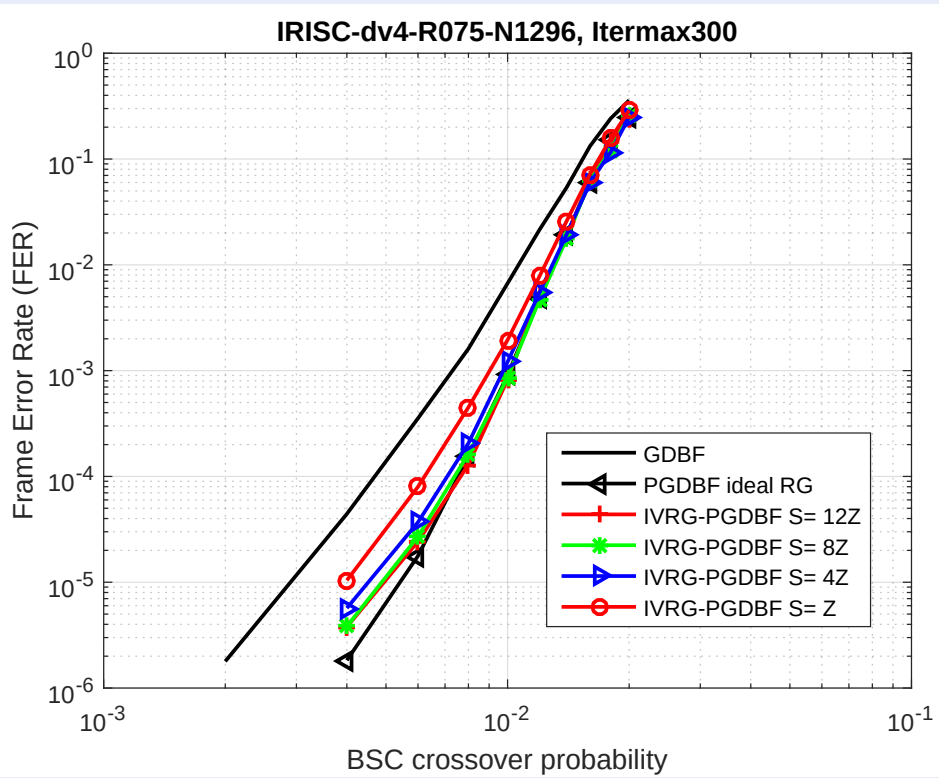


FIGURE: Decoding performance of LFSR-PGDBF, IVRG-PGDBF decoders with different values of S on the LDPC code $d_v = 4$, $d_c = 16$, code rate $R = 0.75$, $N = 1296$

Average number of iteration

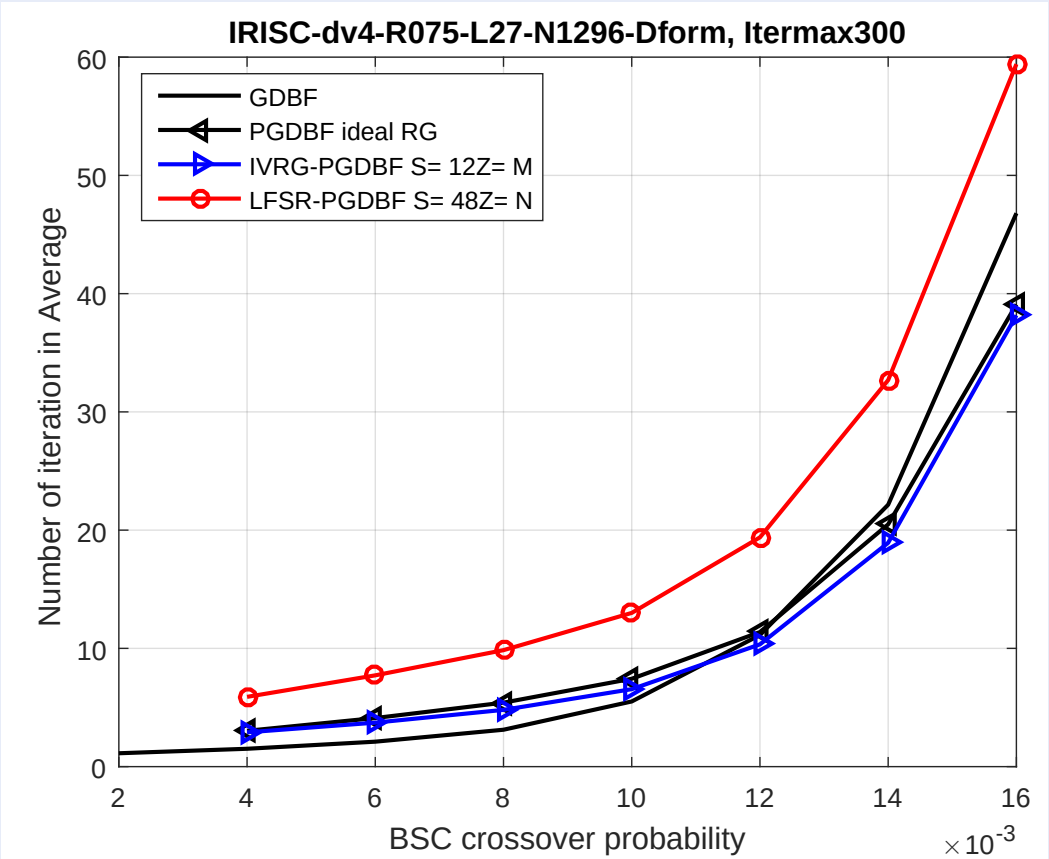


FIGURE: The average number of iteration of GDBF and PGDBF decoders on the LDPC code $d_v = 4$, $d_c = 16$, code rate $R = 0.75$, $N = 1296$

Decoder Implementation Cost

Conventional LCT vs Modified LCT

GDBF		
LUTs	Conventional LCT	Modified LCT
Tanner, dv3-R=0.4 - N=155	1716	1374 (-24.9%)
dv3-R=0.5-N=1296	12554	8959 (-40.1%)
dv3-R=0.75-N=1296	14189	11125 (-27.5%)

Decoder Implementation Cost

Decoder Implementation Cost

dv3R050N1296					
S		**216**	**432**	**648**	**1296**
Registers	**LFSR-PGDBF**	2864 (1.096)	3080 (1.179)	3296 (1.262)	3944 (1.510)
	IVRG-PGDBF	2830 (1.083)	3046 (1.166)	3262 (1.249)	-
LUTs	**LFSR-PGDBF**	10407 (1.162)	10515 (1.174)	10623 (1.186)	10947 (1.222)
	IVRG-PGDBF	10384 (1.159)	10492 (1.171)	10604 (1.184)	-

Decoder Implementation Cost

Decoder Implementation Cost

	Registers					
	dv= 3			dv= 4		
	R040 N155	**R050 N1296**	**R075 N1296**	**R050 N1296**	**R075 N1296**	**R0868 N9520**
GDBF	328 (1.000)	2612 (1.000)	2613 (1.000)	2612 (1.000)	2613 (1.000)	19602 (1.000)
LFSR-PGDBF (S= N)	519 (1.582)	3944 (1.510)	3945 (1.510)	3944 (1.510)	3945 (1.510)	28618 (1.460)
LFSR-PGDBF (S= M)	457 (1.393)	3296 (1.262)	2973 (1.138)	3296 (1.262)	2973 (1.138)	20218 (1.031)
IVRG-PGDBF (S= M)	423 (1.290)	3262 (1.249)	2939 (1.125)	3262 (1.249)	2939 (1.125)	20184 (1.030)
	Look-Up-Tables (LUTs)					
GDBF	1374 (1.000)	8959 (1.000)	11125 (1.000)	13049 (1.000)	12601 (1.000)	92535 (1.000)
LFSR-PGDBF (S= N)	1619 (1.178)	10947 (1.222)	13096 (1.177)	13653 (1.046)	13217 (1.049)	97312 (1.052)
LFSR-PGDBF (S= M)	1589 (1.156)	10623 (1.186)	12608 (1.133)	13333 (1.022)	12731 (1.010)	93111 (1.006)
IVRG-PGDBF (S= M)	1564 (1.138)	10604 (1.184)	12721 (1.143)	13306 (1.020)	12712 (1.009)	93087 (1.006)

Decoder Implementation Cost

Decoder Implementation Cost

dv3R050N1296						
	Registers	**LUTs**	**fmax (MHz)**	**It-Average** α = 0.02	**Ncyc**	**Throughput**
GDBF	2612	8959	96.15	6.74	1.00	18488.93
LFSR-PGDBF (S= N)	3944	10947	100.00	15.31	1.00	8465.06
LFSR-PGDBF (S= 4Z)	2864	10407	102.99	15.05	1.00	8868.48
IVRG-PGDBF (S= 4Z)	2830	10384	101.01	7.02	1.00	18648.02
MinSum	28547	89429	37.04	2.47	3.00	6477.73

Outline

Conclusions

- We introduced a new and original method of randomness generation called IVRG.
  - Instead of generating random bits, we use existing decoder memory
  - Highly efficient in term of hardware cost compared to conventional method (LFSR)
  - Preserving the out standing performance of PGDBF
  - We reduced the average number of iterations (may reduce the dynamic power)
- We Optimized the HW implementation of the PGDBF
  - (modified MI) to reduce the FPGA resources used (to save the static power)

THANK YOU