LDPC Decoding: VLSI Architectures and Implementations

Module 1: LDPC Decoding

Ned Varnica
varnica@gmail.com

Marvell Semiconductor Inc

Overview

- Error Correction Codes (ECC)
- Intro to Low-density parity-check (LDPC) Codes
- ECC Decoders Classification
  - Soft vs Hard Information
- Message Passing Decoding of LDPC Codes
- Iterative Code Performance Characteristics

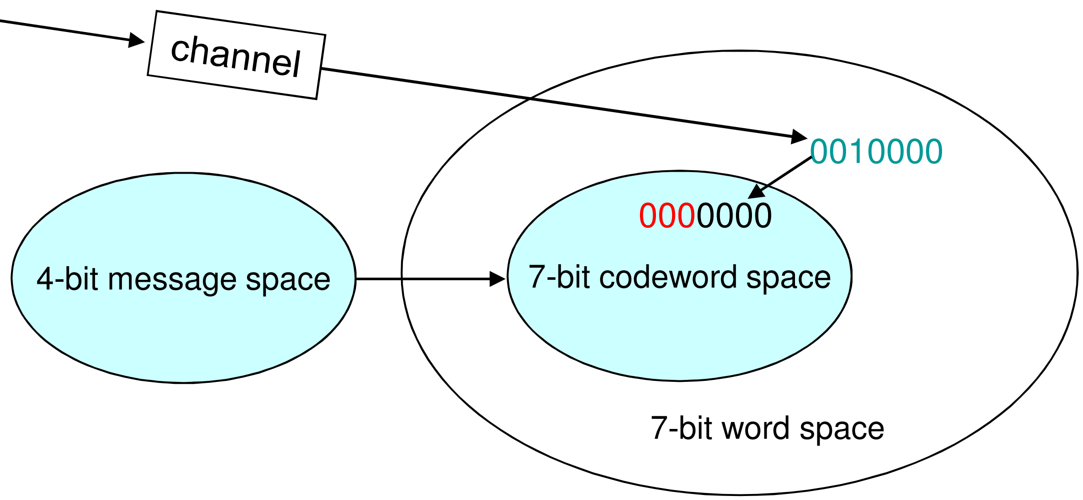
Error Correction Codes (ECC)

Error Correcting Codes (ECC)

User message bits

Parity bits (redundancy)

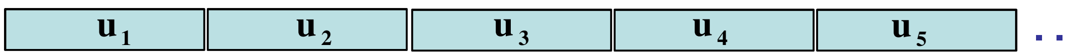
4-bit message	7-bit codeword
0000	0000000
1000	1101000
0100	0110100
1100	1011100
0010	1110010
1010	0011010
0110	1000110
1110	0101110
0001	1010001
1001	0111001
0101	1100101
1101	0001101
0011	0100011
1011	1001011
0111	0010111
1111	1111111



Rate = 4 / 7

Linear Block Codes

- Block Codes
  - User data is divided into blocks (units) of length K bits/symbols



  - Each K bit/symbol user block is mapped (encoded) into an N bit/symbol codeword, where $N > K$

  - Example:
    – in Flash Devices user block length K = 2Kbytes or 4Kbytes is typical
    – code rate $R = K / N$ is usually ~0.9 and higher

- Important Linear Block Codes
    – Reed-Solomon Codes (non-binary)
    – Bose, Chaudhuri, Hocquenghem (BCH) Codes (binary)
    – **Low Density Parity Check (LDPC) Codes**
    – Turbo-Codes

Iterative (ITR) Codes

Generator Matrix and Parity Check Matrix

- A linear block can be defined by a generator matrix

$$\mathbf{G} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0,N-1} \\ g_{10} & g_{11} & \cdots & g_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ g_{K-1,0} & g_{K-1,1} & \cdots & g_{K-1,N-1} \end{bmatrix} \xrightarrow{encoding} \underbrace{\mathbf{v}}_{\text{Codeword}} = \underbrace{\mathbf{u}}_{\text{User message}} \cdot \mathbf{G}$$

- Matrix associated to **G** is parity check matrix **H**, s.t. $\mathbf{G} \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{0}$
  - A vector is a codeword if

$$\mathbf{v} \cdot \mathbf{H}^{\mathrm{T}} = \mathbf{0}$$

- A non-codeword (codeword + noise) will generate a non-zero vector, which is called syndrome

$$\hat{\mathbf{v}} \cdot \mathbf{H}^{\mathrm{T}} = \mathrm{s}$$

- The syndrome can be used in decoding

Example

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Encoding

$$\mathbf{u} = (1 \quad 1 \quad 0 \quad 1)$$

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} = (0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1)$$

$$\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

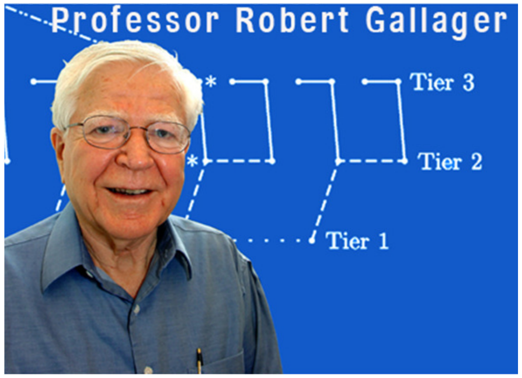
Decoding

$$\hat{\mathbf{v}} = (0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad \underline{0})$$

$$\hat{\mathbf{v}} \cdot \mathbf{H}^T = (1 \quad 0 \quad 1)$$

Flash Memory SUMMIT

Low-Density Parity-Check (LDPC) Codes



David MacKay

LDPC Codes

- LDPC code is often defined by parity check matrix **H**
  - The parity check matrix, **H**, of an LDPC code with practical length has low density (most entries are 0's, and only few are 1's), thus the name *Low-Density* Parity-Check Code
- Each bit of an LDPC codeword corresponds to a column of parity check matrix
- Each rows of **H** corresponds to a single parity check
  - For example, the first row indicates that for any codeword the sum (modulo 2) of bits 0,1, and N-1 must be 0

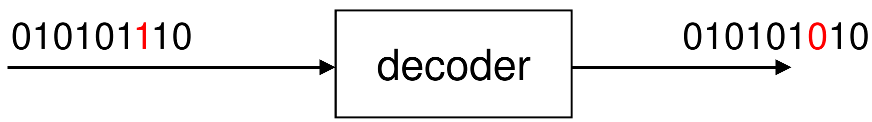
$$\mathbf{H} = \begin{array}{c} \begin{array}{ccccccc} \text{bit 0} & \text{bit 1} & & & & & \text{bit N-1} \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & 0 & 1 & \ldots & 0 & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix} \end{array}$$

Parity check equation

ECC Decoder Classification: Hard vs Soft Decision Decoding

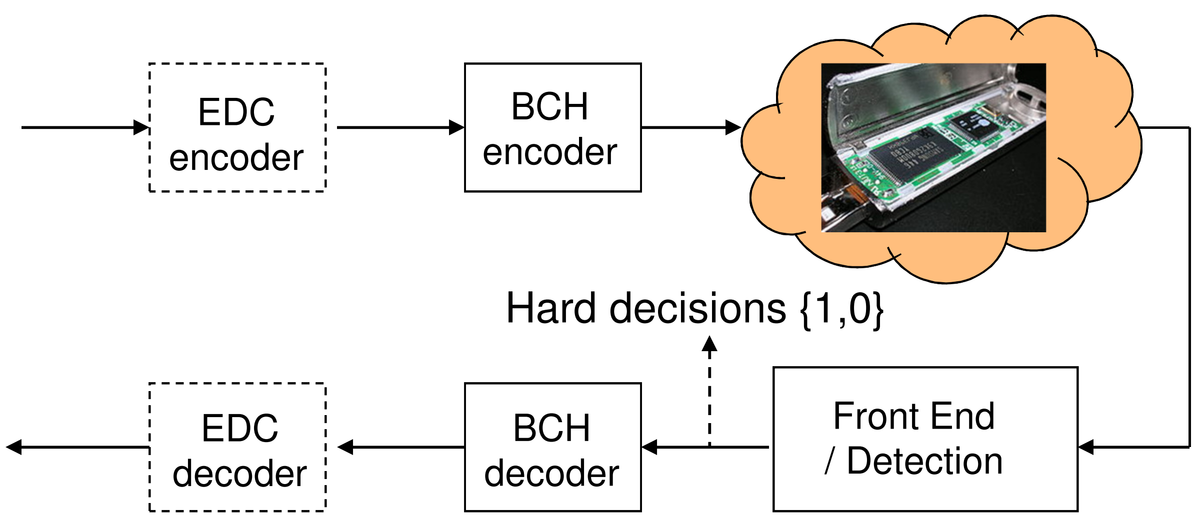
Hard vs. Soft Decoder Classification

- *Hard decoders* only take hard decisions (bits) as the input



  - E.g. Standard BCH and RS ECC decoding algorithm (Berlekamp-Massey algorithm) is a hard decision decoder

- Hard decoder algorithm could be used if one read is available

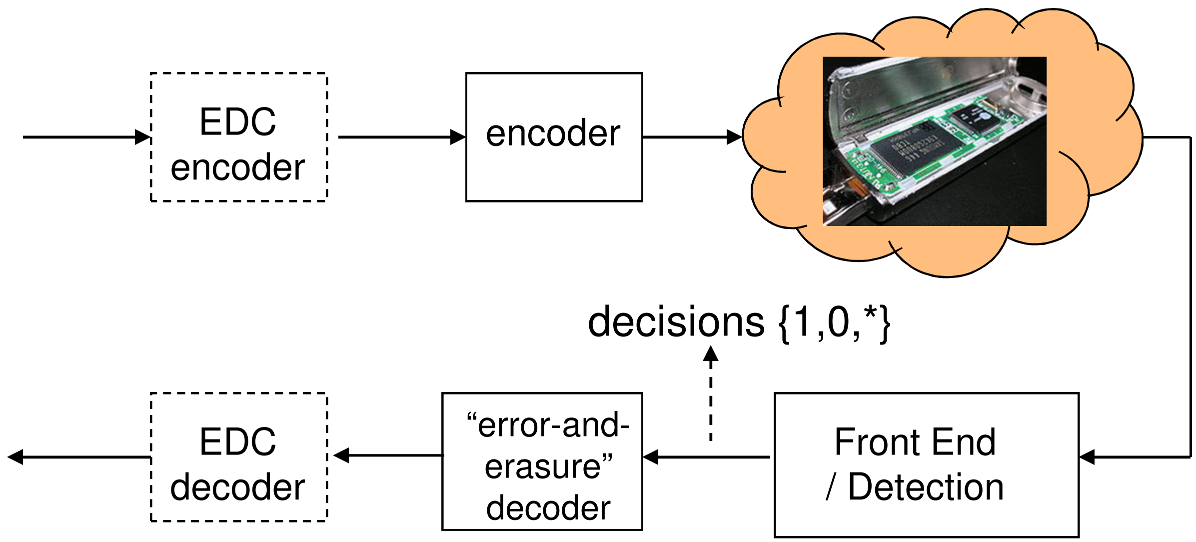


Hard vs. Soft Decoder Classification

- Error-and-Erasure decoder is a variant of soft information decoder: in addition to hard decisions, it takes erasure flag as an input

010101**0 → decoder → 010101010

- *Error-and-Erasure* decoder algorithm could be used if two reads are available



Hard vs. Soft Decoder Classification

- Erasure flag is an example of soft information (though very primitive)
- Erasure flag points to symbol locations that are deemed unreliable by the channel
- Normally, for each erroneous symbol, decoder has to determine that the symbol is in error and find the correct symbol value. However, if erasure flag identifies error location, then only error value is unknown
- Therefore, erasure flag effectively reduces number of unknowns that decoder needs to resolve

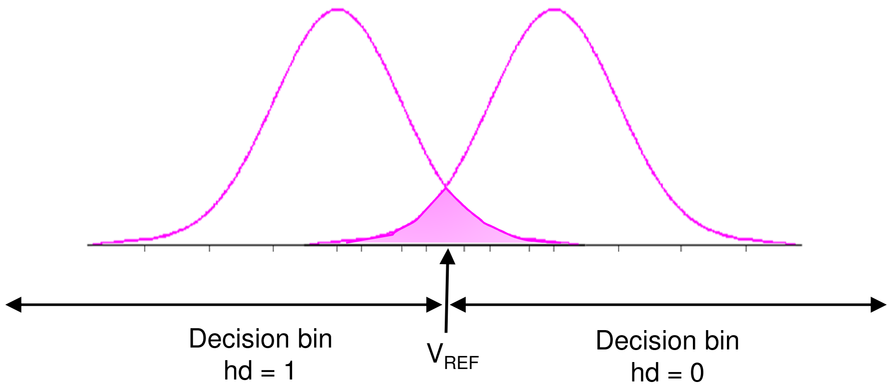
Hard vs. Soft Decoder Classification

- Example. Rate 10/11 Single parity check (SPC) code
  - Each valid 11-bit SPC codeword $c=(c_0,c_1,\ldots c_{10})$ has the sum (mod 2) of all the bits equal to 0
  - Assume that (0,0,0,0,0,0,0,0,0,0,0) is transmitted, and (0,0,0,0,1,0,0,0,0,0,0) is received by decoder
  - The received vector does not satisfy SPC code constraint, indicating to the decoder that there are errors present in the codeword
  - Furthermore, assume that channel detector provides bit level reliability metric in the form of probability (confidence) in the received value being correct
  - Assume that soft information corresponding to the received codeword is given by (0.9,0.8,0.86,0.7,0.55,1,1,0.8,0.98,0.68,0.99)
  - From the soft information it follows that bit c_4 is least reliable and should be flipped to bring the received codeword in compliance with code constraint

Obtaining Hard or Soft Information from Flash Devices

One Read: Hard Information

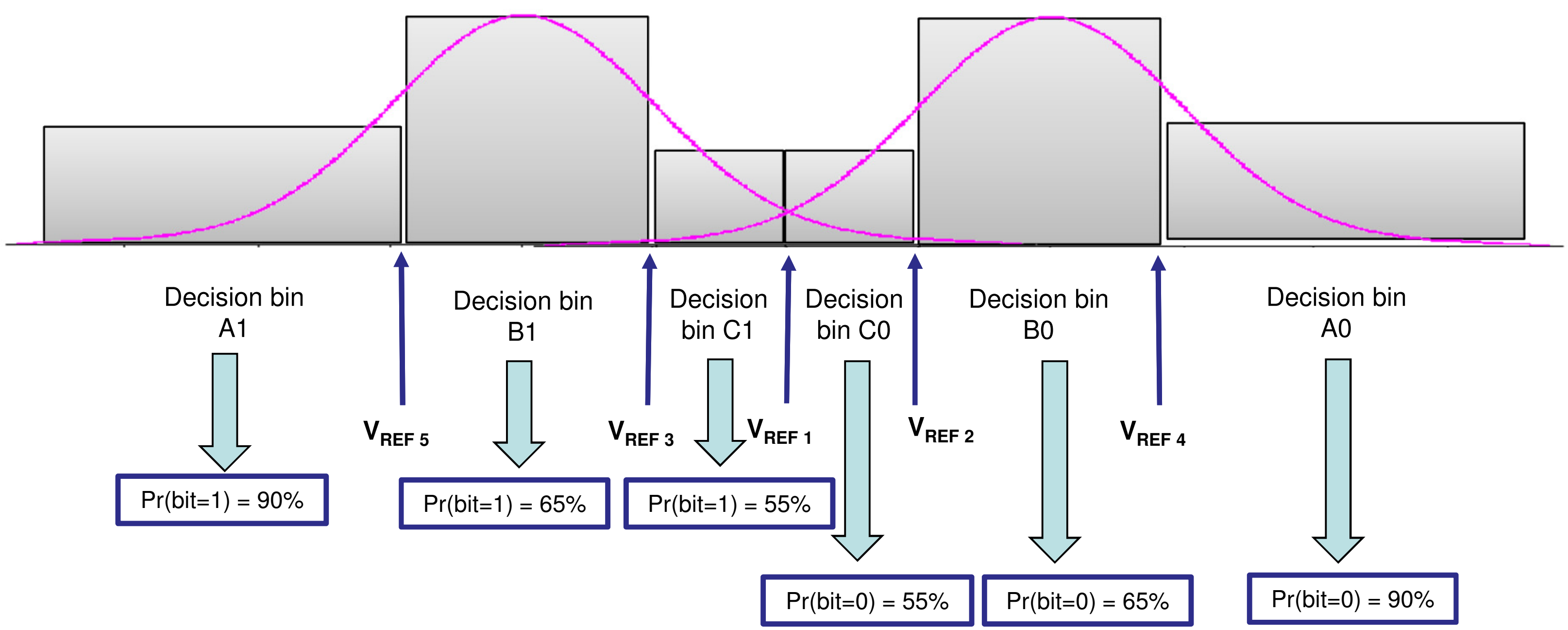
- Obtaining hard information (decision) via one read
- One V_{REF} threshold is available: Threshold value should be selected so that the average raw bit-error-rate (BER) is minimized



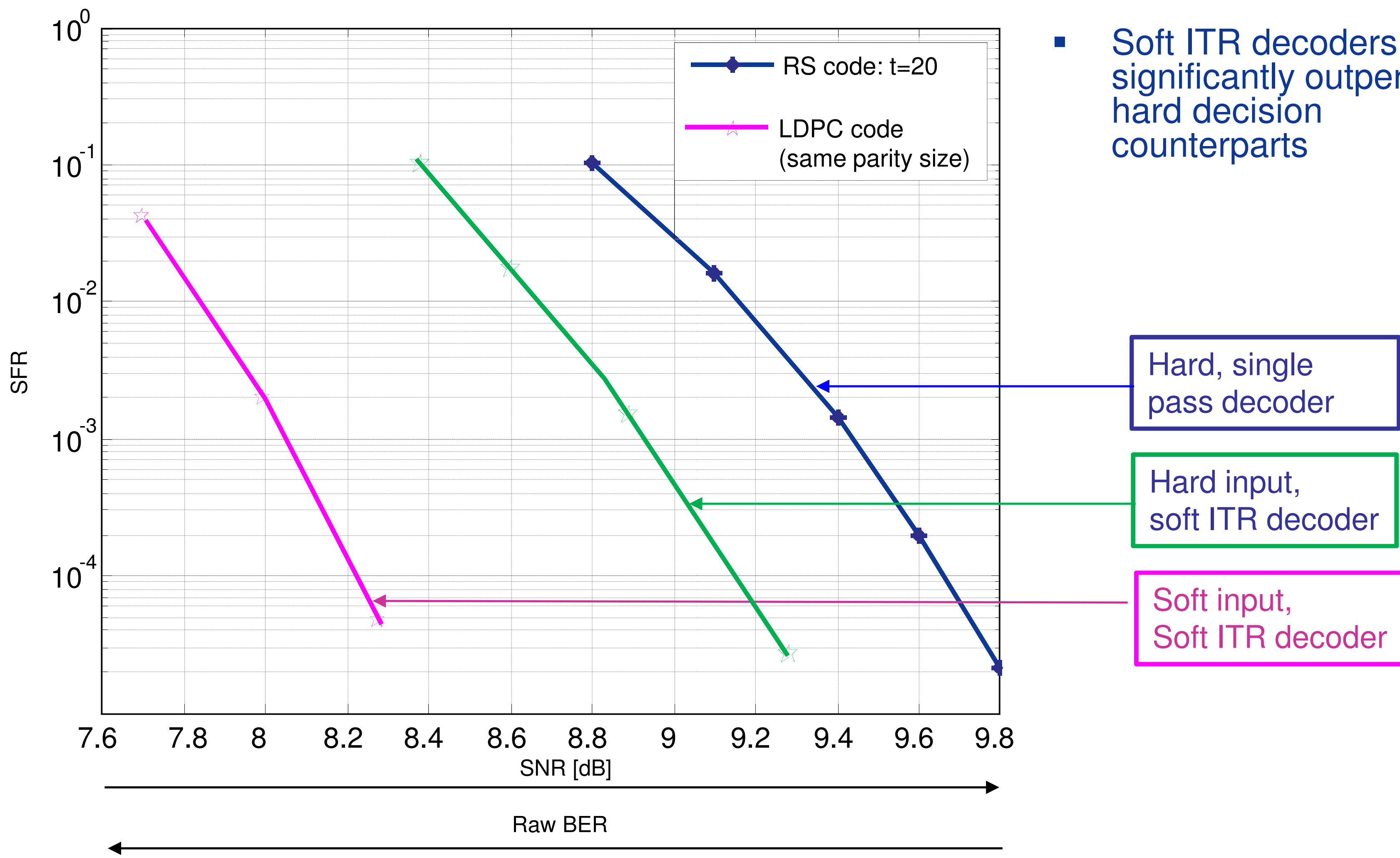
- In each bit-location, the hard-decision hd = 0 or hd = 1 is made
- This information can be used as input into decoder
- Shaded area denotes the probability that a bit error is made

Multiple Reads: Soft Information

- **Obtaining soft information via multiple reads**
  - **Create bins**
  - **Bins can be optimized in terms of their sizes / distribution of V_{REF} values given the number of available reads (e.g. 5 reads)**
  - **These bins can be mapped into probabilities**
  - **Typically, the closer the bin to the middle point, the lower the confidence that the bit value (hard-read value) is actually correct**



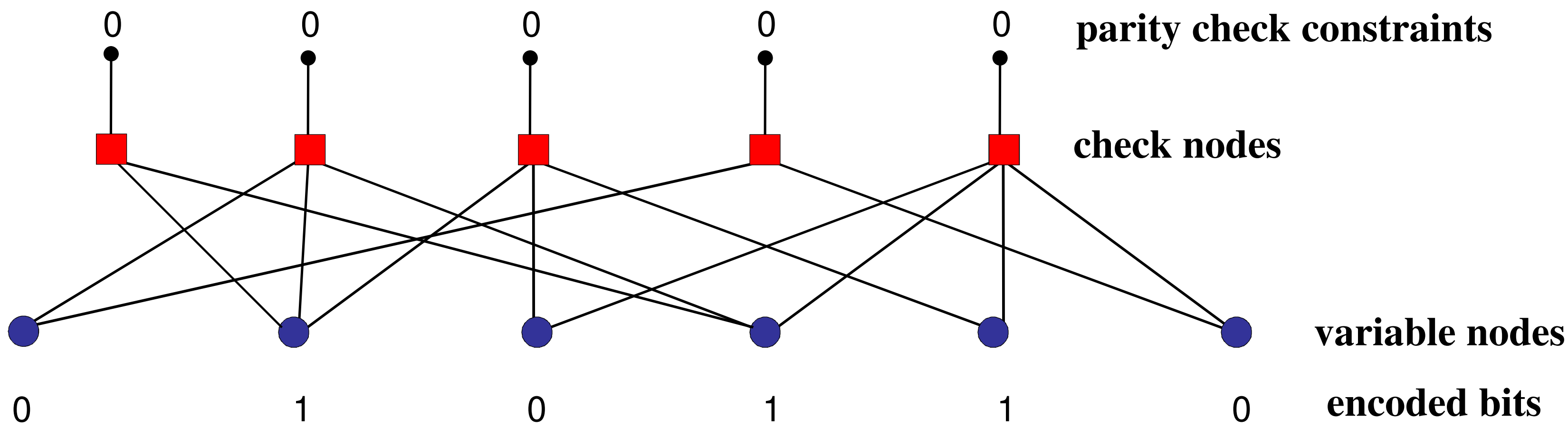
ITR Decoders with Soft Information



- Soft ITR decoders significantly outperform hard decision counterparts

Decoding LDPC Codes

Representation on Bi-Partite (Tanner) Graphs

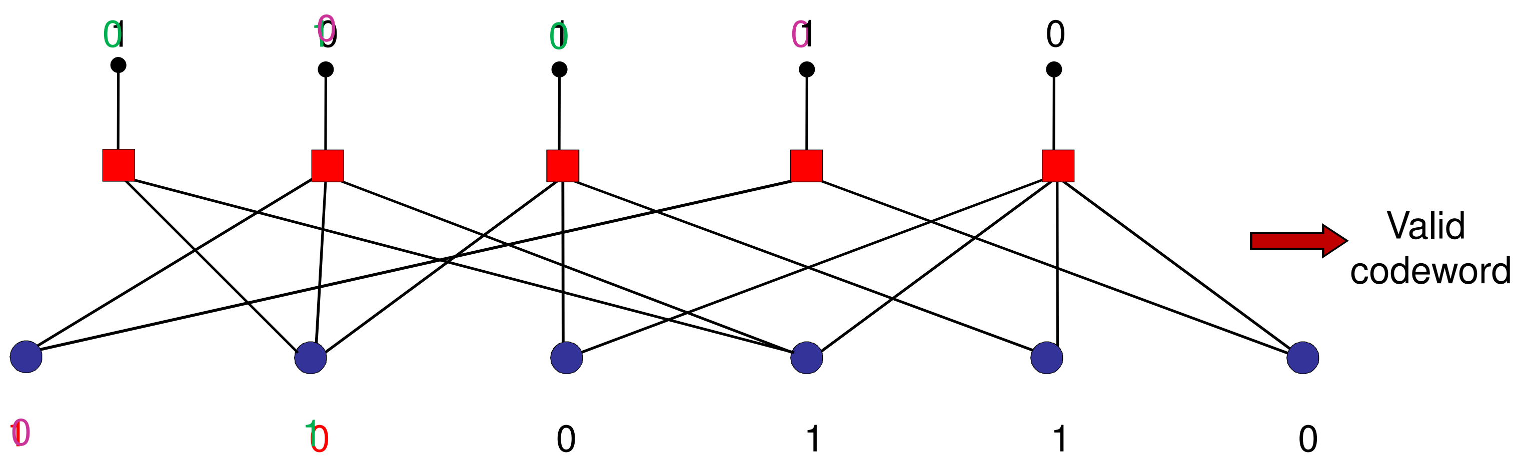


Each bit “1” in the parity check matrix is represented by an edge between corresponding variable node (column) and check node (row)

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

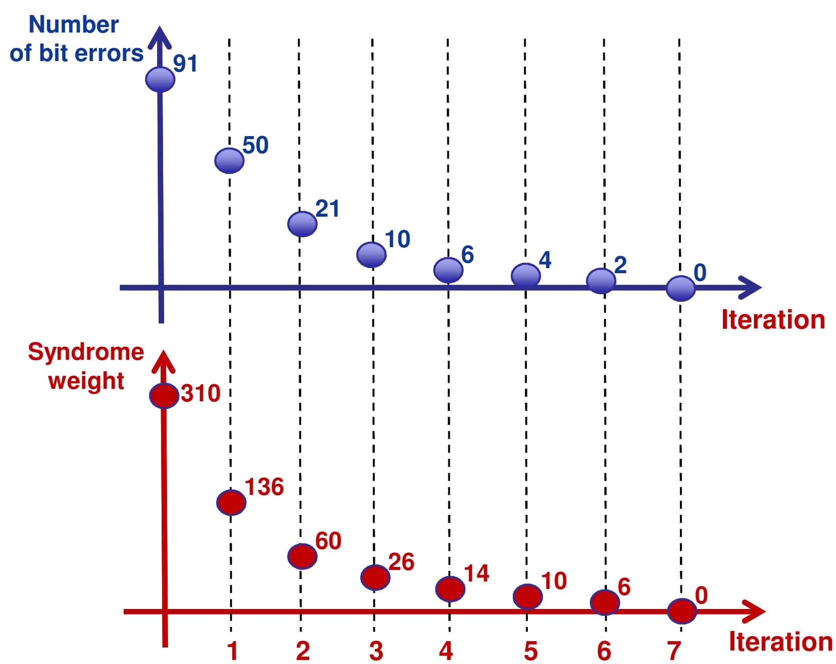
Hard Decision Decoding: Bit-Flipping Decoder

- Decision to flip a bit is made based on the number of unsatisfied checks connected to the bit



Bit-Flipping Decoder Progress on a Large LDPC Code

- Decoder starts with a relatively large number of errors
- As decoder progresses, some bits are flipped to their correct values
- Syndrome weight improves
  - As this happens, it becomes easier to identify the bits that are erroneous and to flip the remaining error bits to actual (i.e. written / transmitted) values



Soft Information Representation

- The information used in soft LDPC decoder represents bit reliability metric, LLR (log-likelihood-ratio)

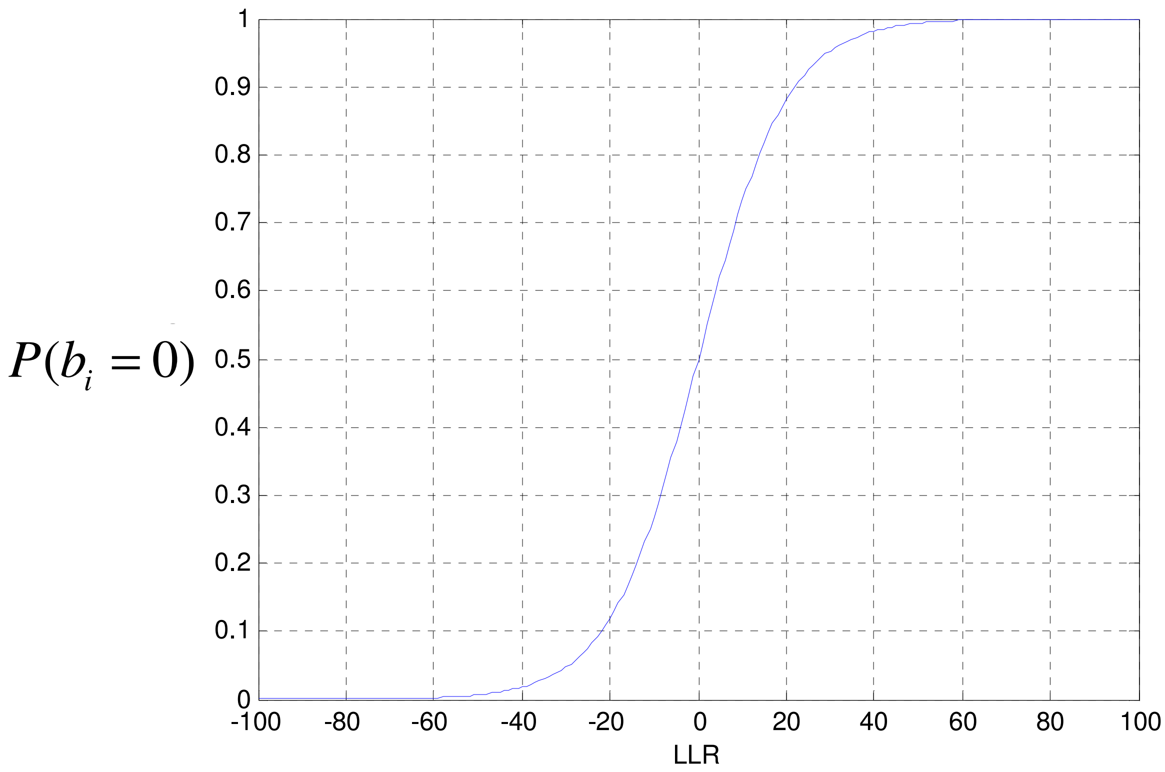
$$LLR(b_i) = \log\left(\frac{P(b_i = 0)}{P(b_i = 1)}\right)$$

- The choice to represent reliability information in terms of LLR as opposed to probability metric is driven by HW implementation consideration

- The following chart shows how to convert LLRs to probabilities (and vice versa)

Soft Information Representation

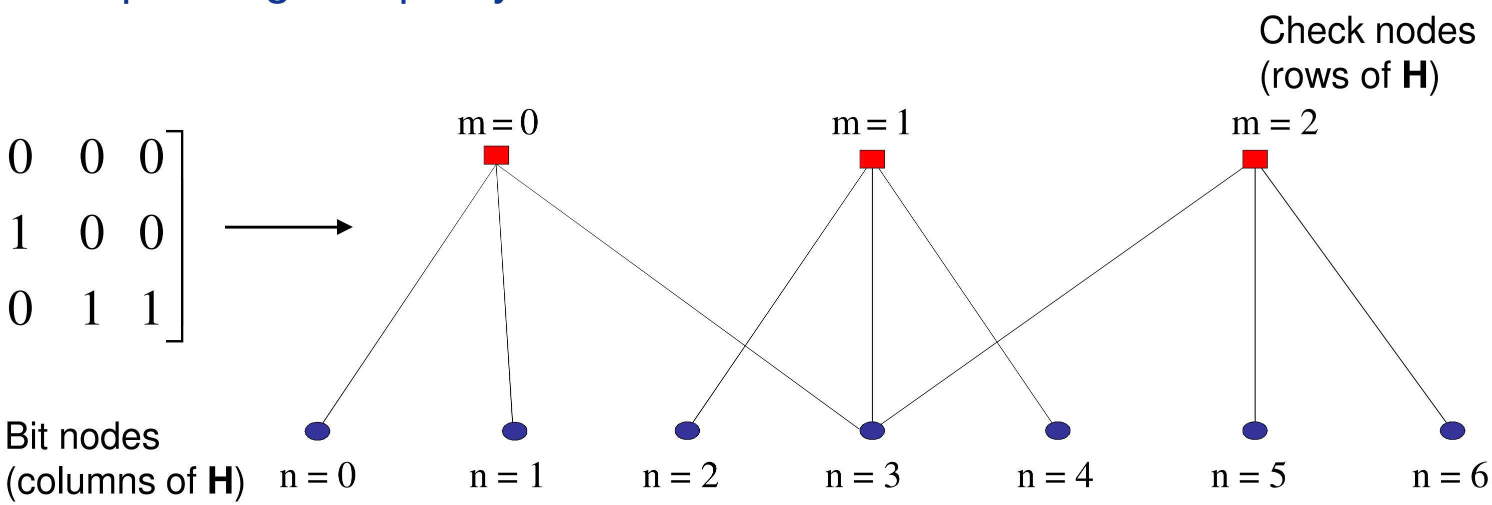
- Bit LLR>0 implies bit=0 is more likely, while LLR<0 implies bit=1 is more likely



Soft Message Passing Decoder

- LDPC decoding is carried out via message passage algorithm on the graph corresponding to a parity check matrix **H**

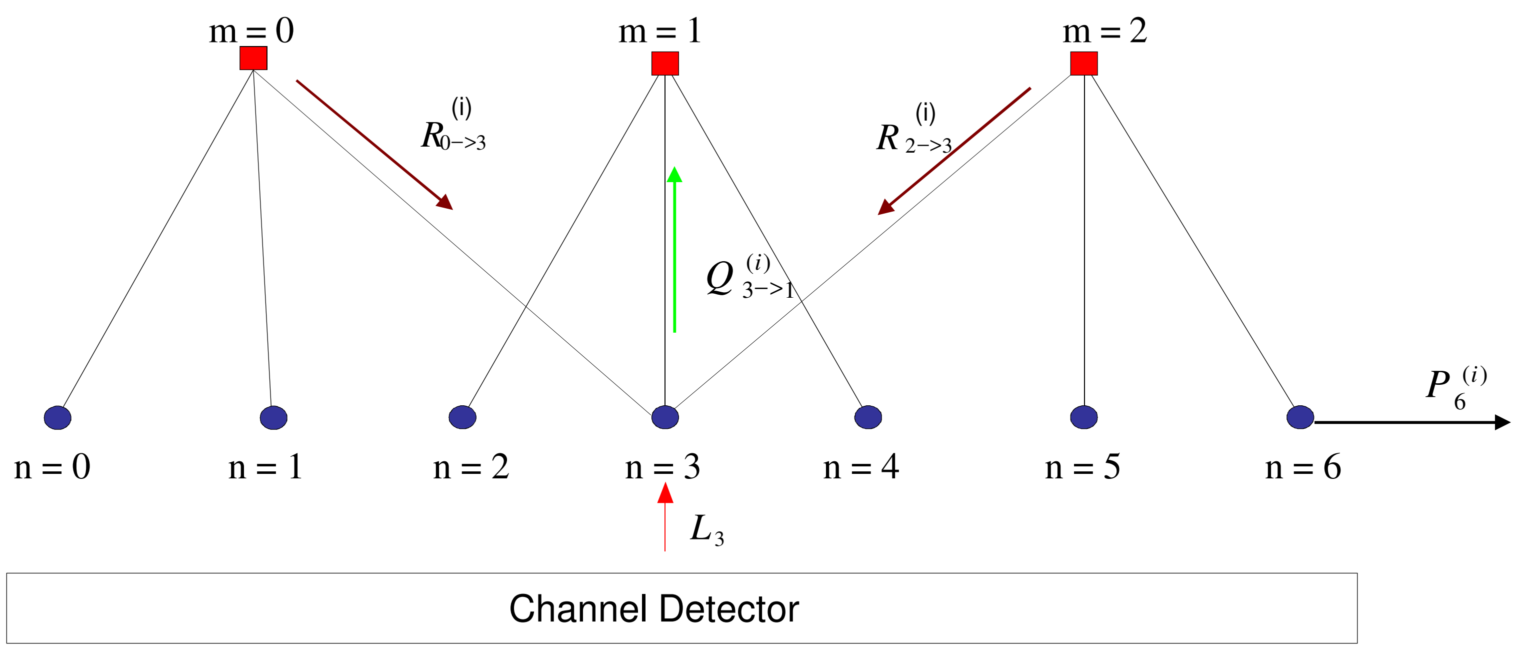
$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \longrightarrow$$



- The messages are passed along the edges of the graph
  - First from the bit nodes to check nodes
  - And then from check nodes back to bit nodes

Soft LDPC Decoder

- There are four types of messages
  - Message from the channel to the n-th bit node L_n
  - Message from n-th bit node to the m-th check node $Q^{(i)}_{n->m}$
  - Message from the m-th check node to the n-th bit node $R^{(i)}_{m->n}$
  - Overall reliability information for n-th bit-node at the end of iteration $P^{(i)}_n$

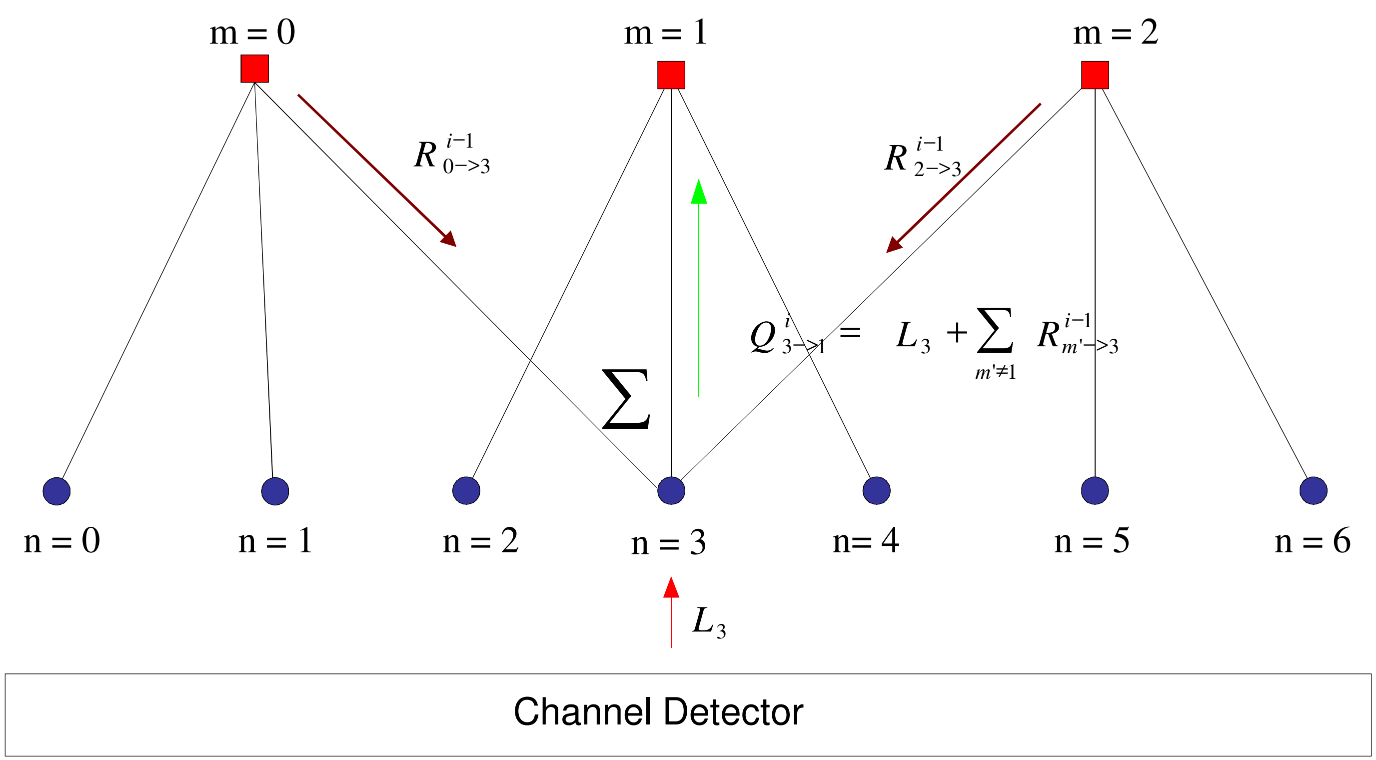


Soft LDPC Decoder (cont.)

- Message passing algorithms are iterative in nature
- One iteration consists of
  - upward pass (bit node processing/variable node processing): bit nodes pass the information to the check nodes
  - downward pass (check node processing): check nodes send the updates back to bit nodes
- The process then repeats itself for several iterations

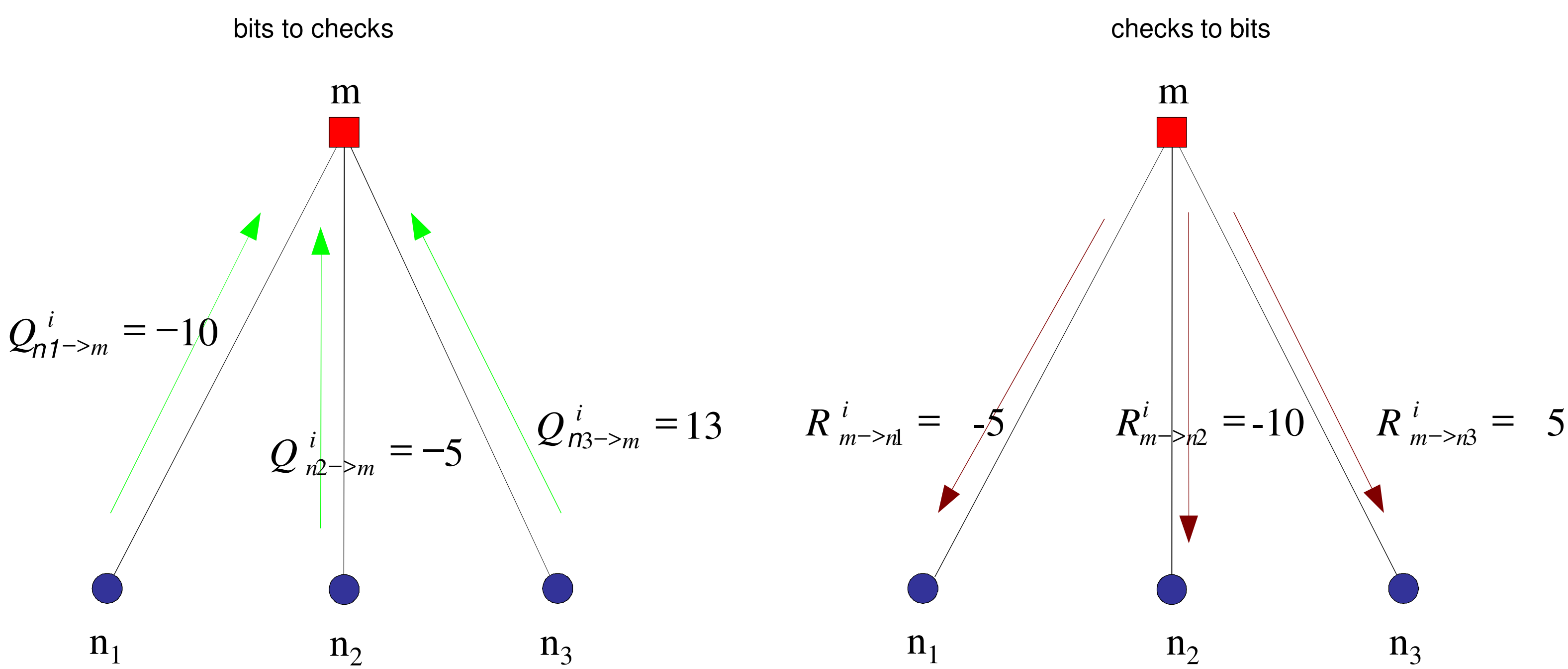
Soft LDPC Decoder (cont.)

- Bits-to-checks pass: $Q^{(i)}_{n->m}$: n-th bit node sums up all the information it has received at the end of last iteration, except the message that came from m-th check node, and sends it to m-th check node
  - At the beginning of iterative decoding all R messages are initialized to zero



Soft LDPC Decoder (cont.)

- Checks-to-bits pass:
  - Check node has to receive the messages from all participating bit nodes before it can start sending messages back
  - Least reliable of the incoming extrinsic messages determines magnitude of check-to-bit message. Sign is determined so that modulo 2 sum is satisfied



Soft LDPC Decoder (cont.)

- At the end of each iteration, the bit node computes overall reliability information by summing up ALL the incoming messages

$$P_n^{(i)} = L_n + \sum_m R_{m->n}^{(i)}$$

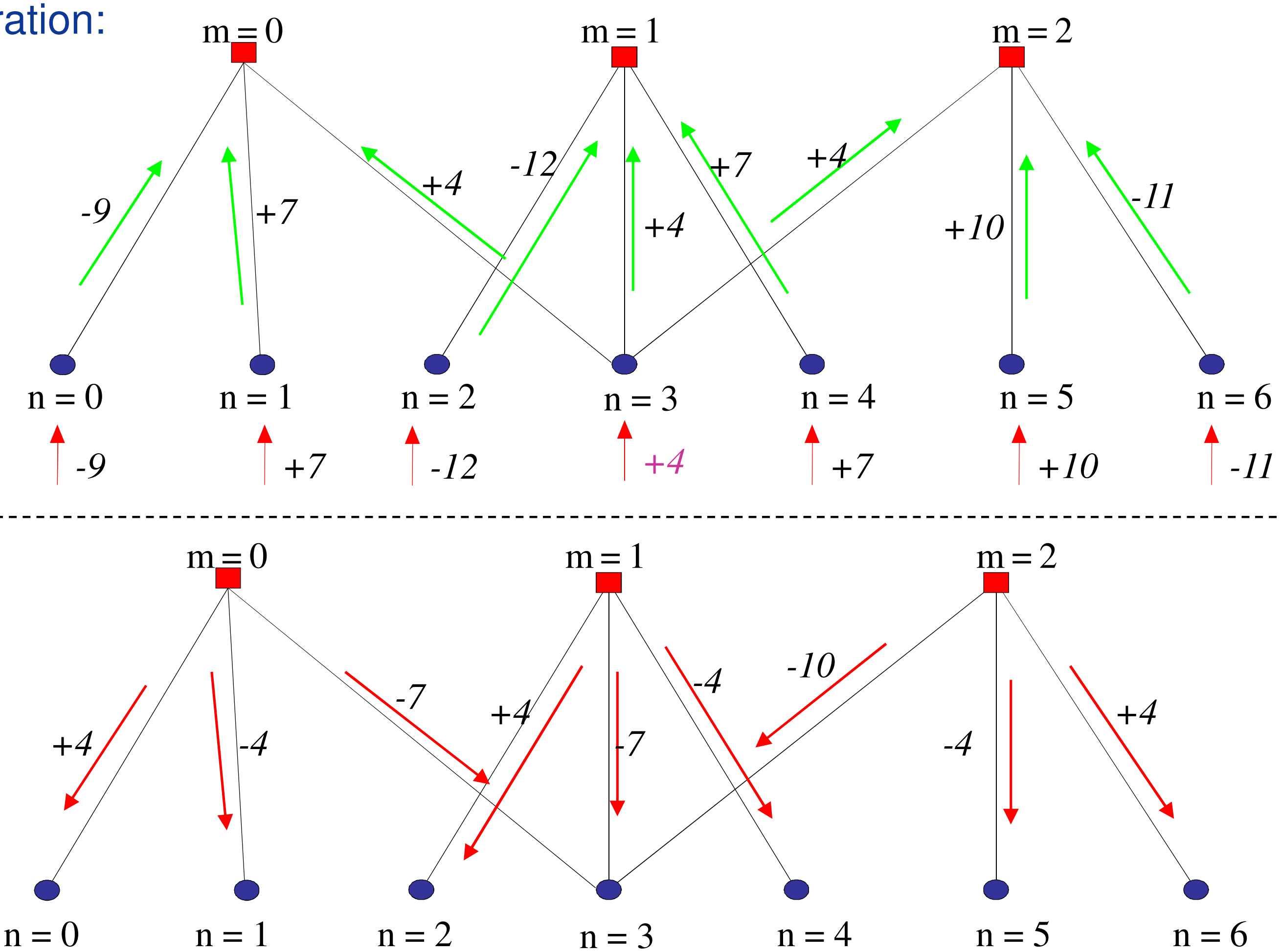
- $P_n^{(i)}$'s are then quantized to obtain hard decision values for each bit

$$\widehat{x}_n = \begin{cases} 1, & \text{if } P_n^{(i)} < 0 \\ 0, & \text{else} \end{cases}$$

- Stopping criterion for an LDPC decoder
  - Maximum number of iterations have been processed OR
  - All parity check equations are satisfied

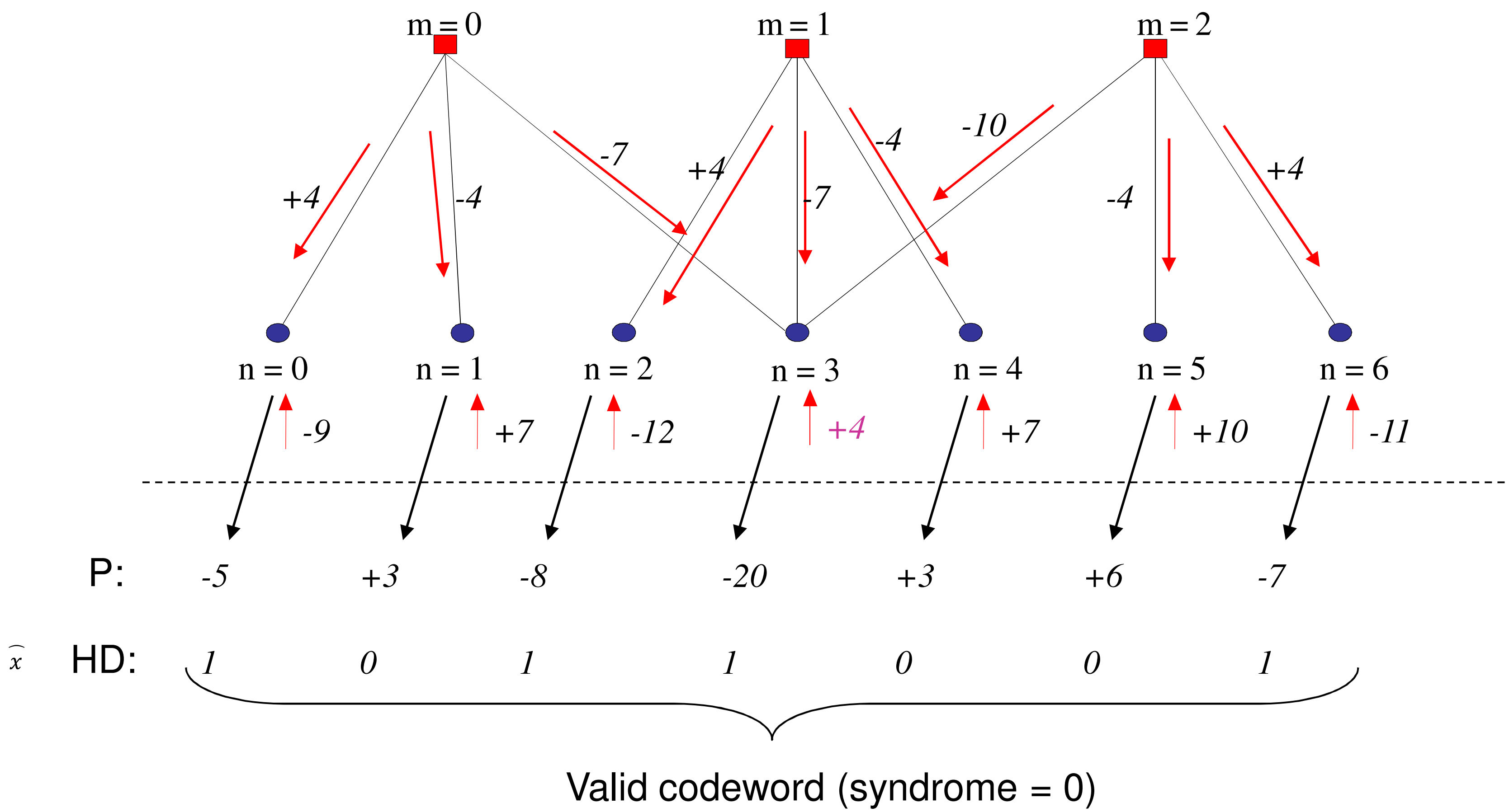
LDPC Decoder Error Correction: Example 1

- 1st iteration:



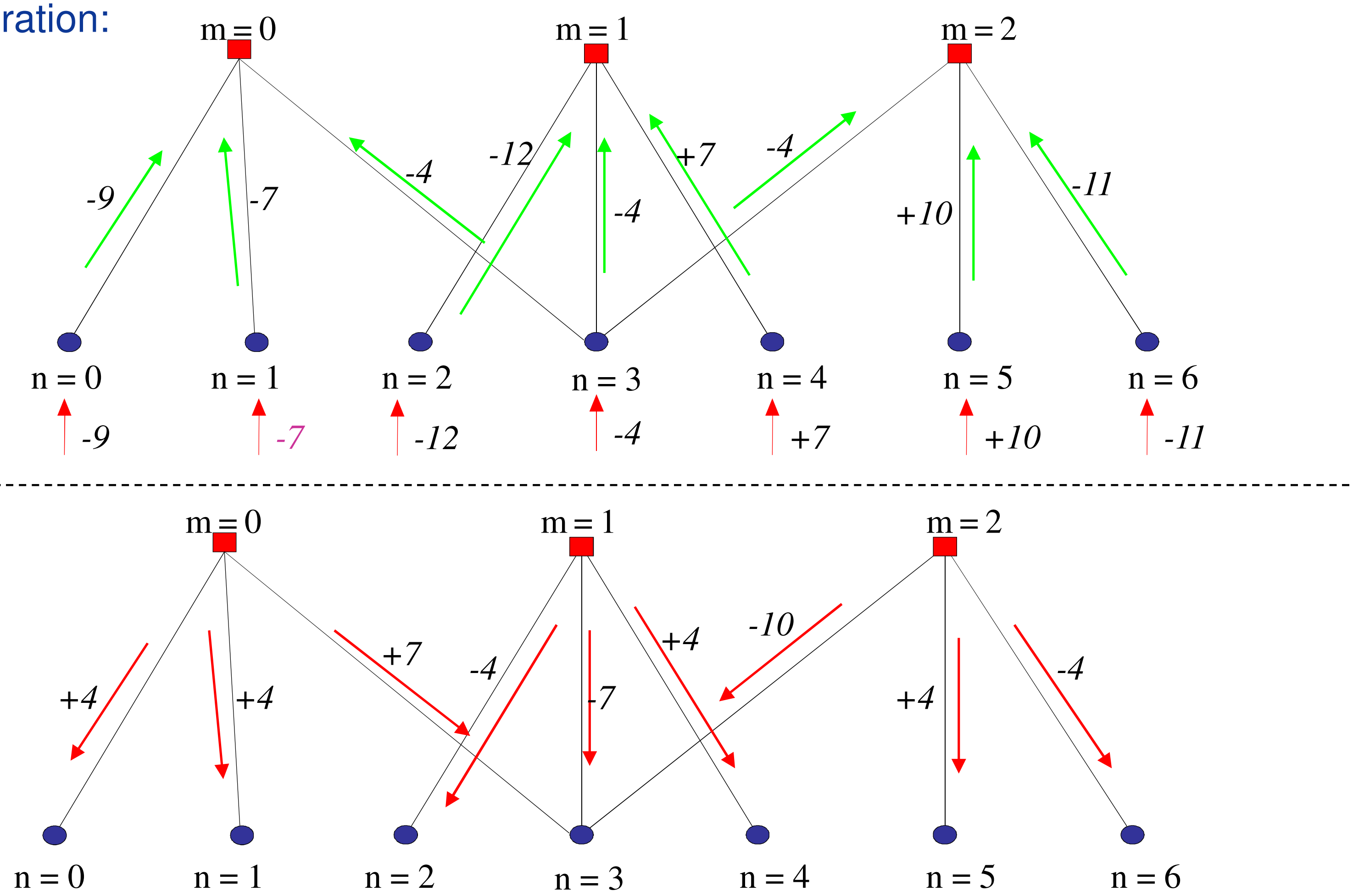
LDPC Decoder Error Correction: Example 1

- APP messages and hard decisions after 1st iteration:



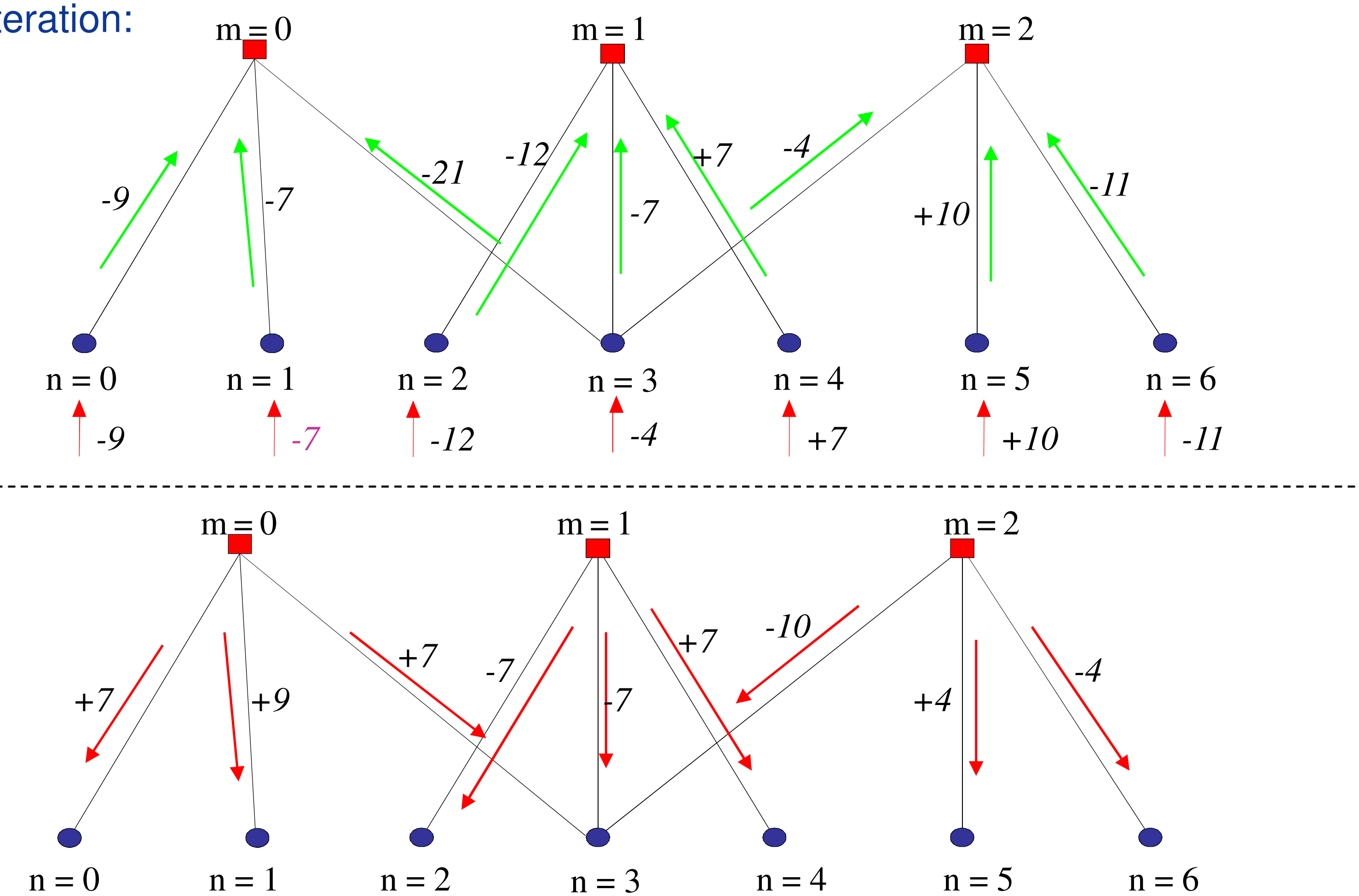
LDPC Decoder Error Correction: Example 2

- 1^{st} iteration:



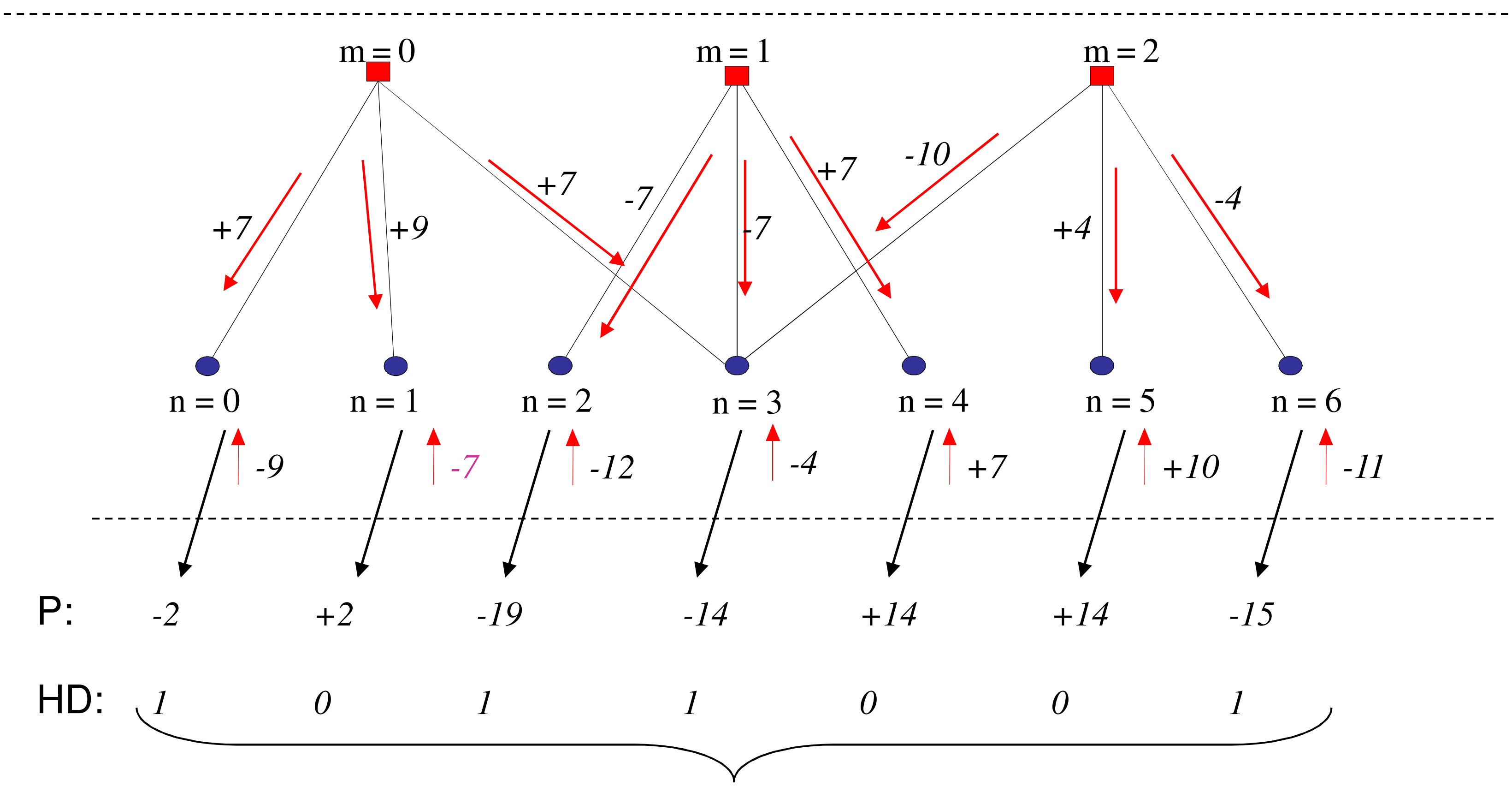
LDPC Decoder Error Correction: Example 2

- 2nd iteration:



LDPC Decoder Error Correction: Example 2

- APP messages and hard decisions after 2nd iteration:



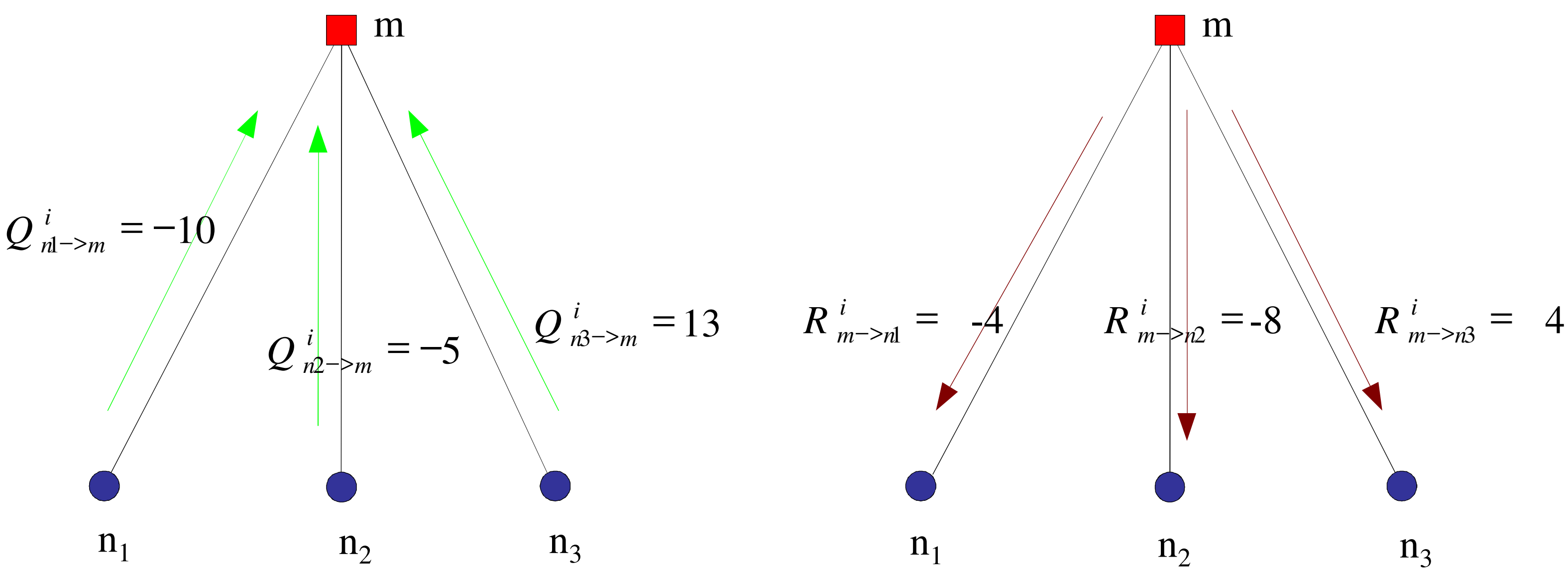
Valid codeword (syndrome = 0)

Sum-Product and Min-Sum Decoders

- Sum-Product: Optimal update rules at the check nodes request implementation of fairly complex *tanh()* function and its inverse

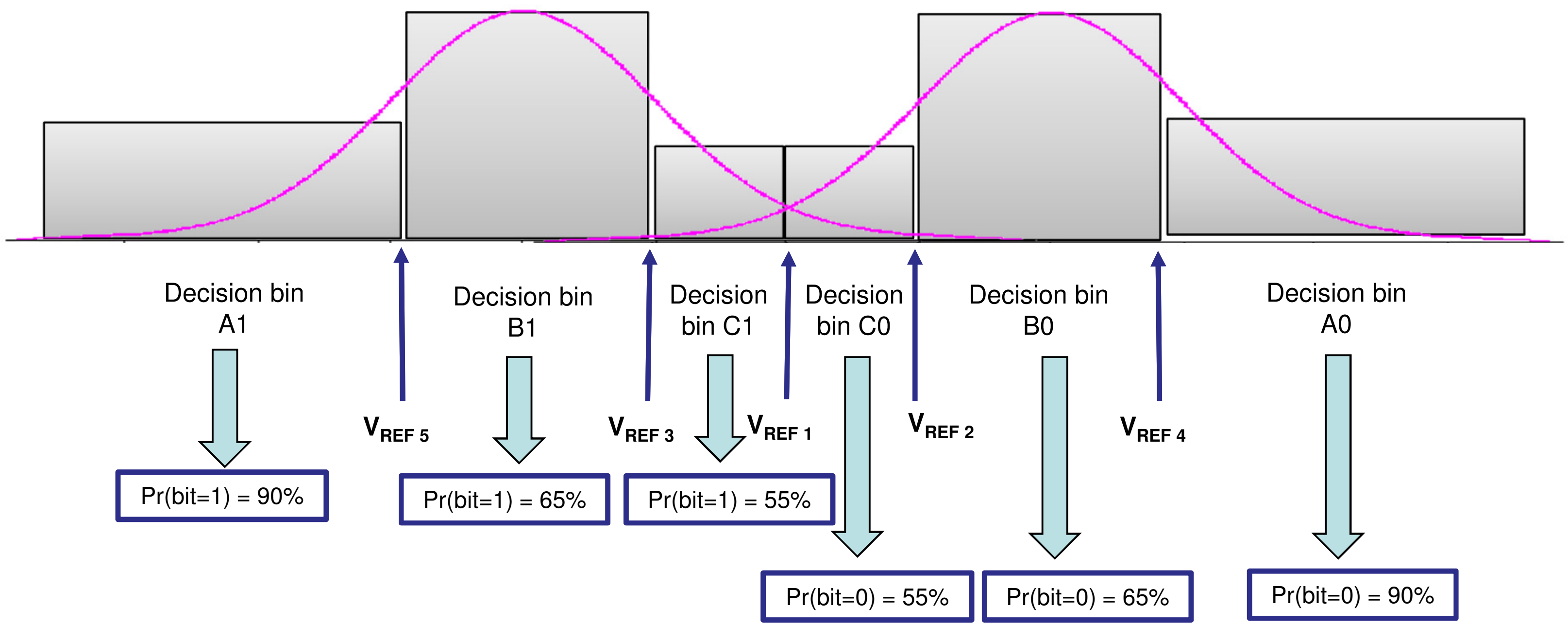
- Instead of these update rules, simple approximate rules have been devised: The rules require only computing minimum messages at each check node
  - In order to make approximation work, it is necessary/critical to utilize scaling/offsetting of messages from check to bit nodes

- This algorithm is widely known as min-sum with scaling/offset and is often choice of implementation in Hardware



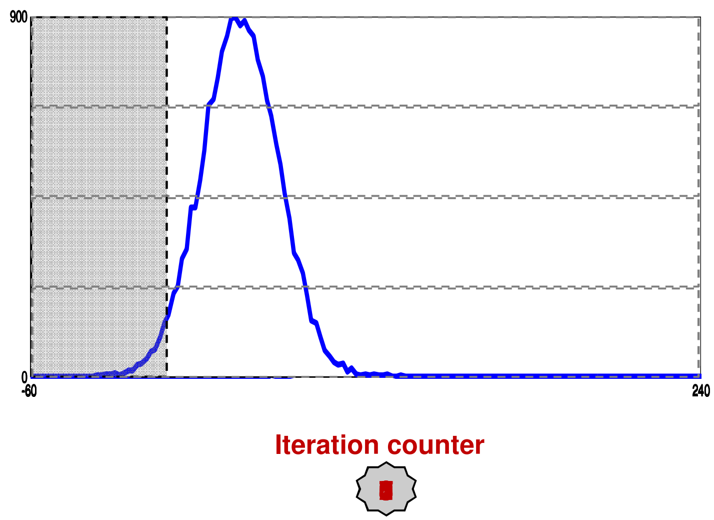
Histogram of LLRs on Large LDPC Codes

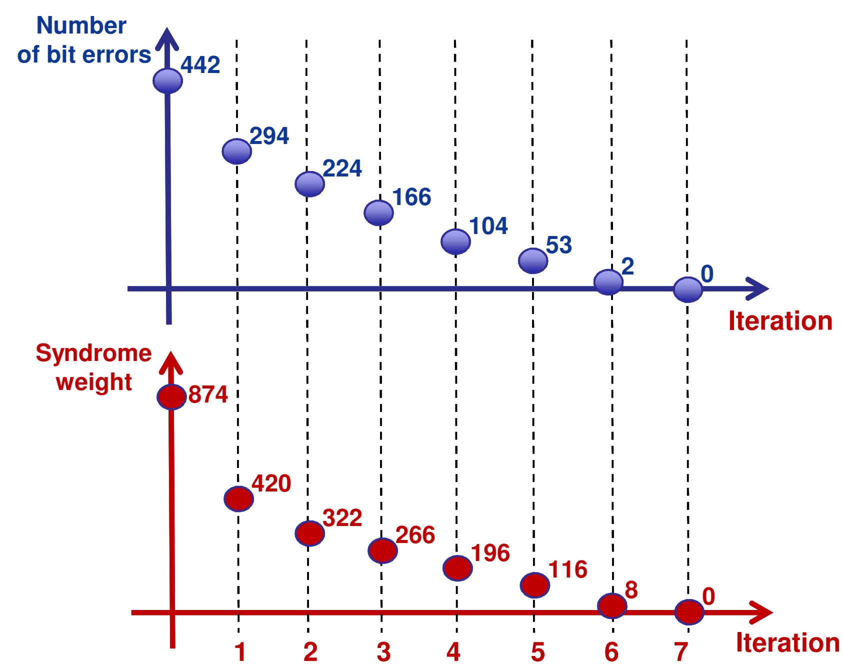
- LDPC min-sum decoder on AWGN channel
- One critical advantage of soft (min-sum) decoder is that it can utilize the information on bits provided by several reads
- Using multiple reads reveals additional information for each individual bit position (bin allocation / LLR mapping)
- Soft decoder could start with a fairly large number of LLRs with incorrect signs



Histogram of LLRs on Large LDPC Codes

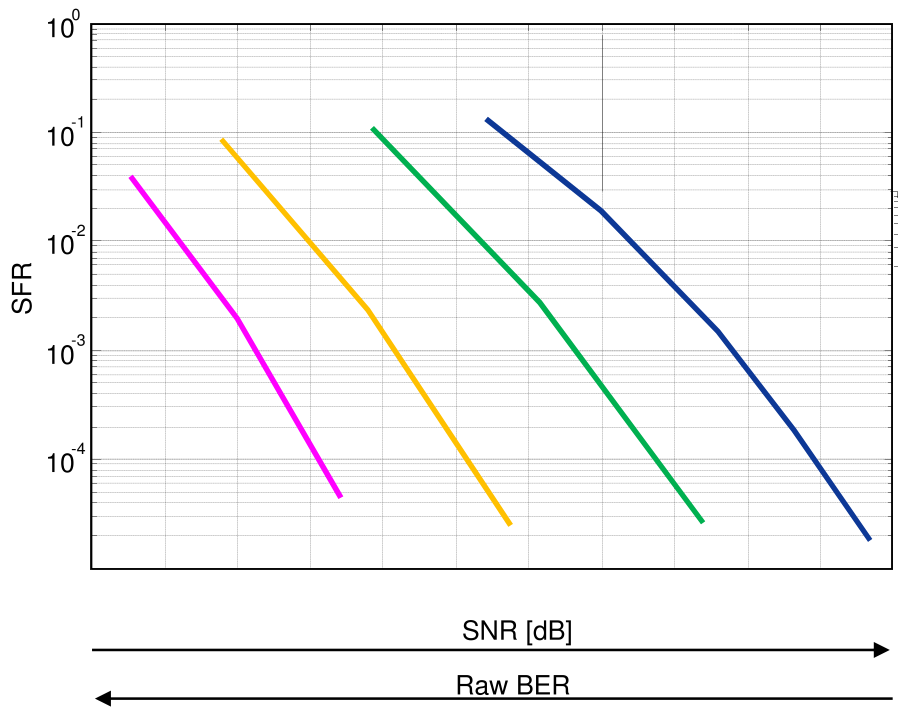
- Soft decoder could start with a fairly large number of LLRs with incorrect signs
- Decoder takes advantage of the original soft information and improves the information on some bits during the initial iteration
- As iterations progress, propagation of improved information continues. This reduces the number of bit positions with incorrect LLR signs (hard-decisions)
- Eventually, all bit positions receive correct sign of LLRs: at this point the syndrome will verify that a valid codeword is found and decoder can terminate

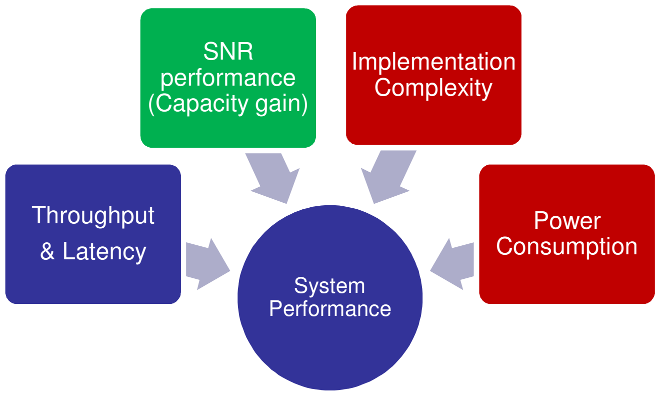




Performance / Complexity Trade-Offs

- **The choice of number of iterations is typically made with consideration of the following parameters:**
  - **Throughput / Latency**
  - **SNR performance (Capacity gain)**
  - **Implementation Complexity**
  - **Power Consumption**



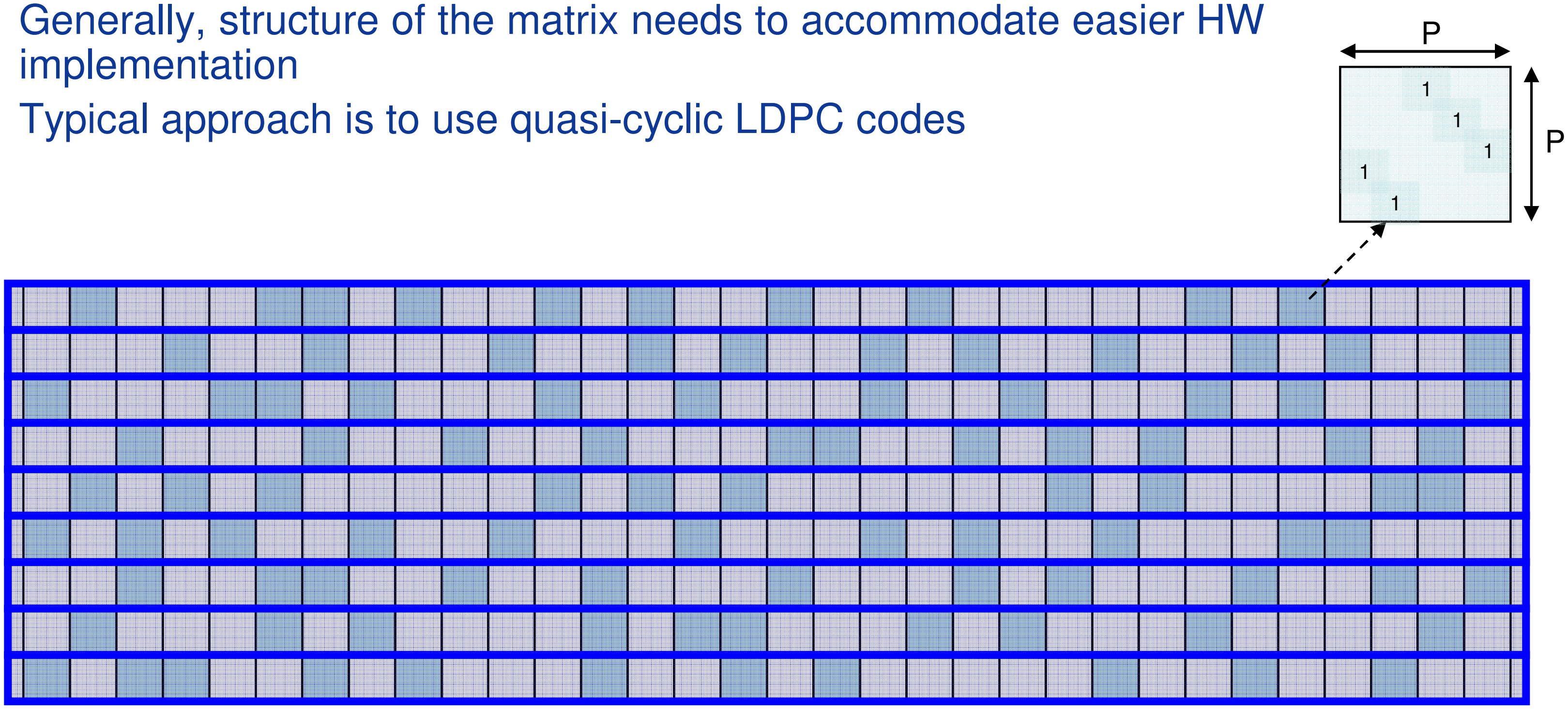


Fixing throughput/latency & increasing parallelism in implementation

Code Design, Code Performance Characteristics and Efficient Hardware

Quasi-Cyclic LDPC Codes

- Generally, structure of the matrix needs to accommodate easier HW implementation
- Typical approach is to use quasi-cyclic LDPC codes



- With such matrix structures, row/column processing in decoding can be parallelized, e.g. process P variable/check nodes in a single clock cycle
- The same processors could be utilized with scheduling and memory addressing handling different portions of the parity check matrix in different clock cycles

Layered / Flooding Decoder

- Updates of messages may be done in a “flooding” fashion or in a layered (serial) fashion

- Both of these decoders benefit from structured matrices that naturally allow for parallel processing of a portion of the matrix, i.e. parallel processing of some number of rows / columns in the matrix

- The main difference in layered decoding approach is that the information is utilized in serial fashion: New messages are utilized during the current iteration, as opposed to the flooding decoder that obtains new information on all nodes exactly once in each iteration

- It has been demonstrated that layered/serial decoder can converge in about ½ of the number of iterations needed by flooding decoder

LDPC Iterative Decoder Performance Characteristics

RS-ECC Performance Characterization

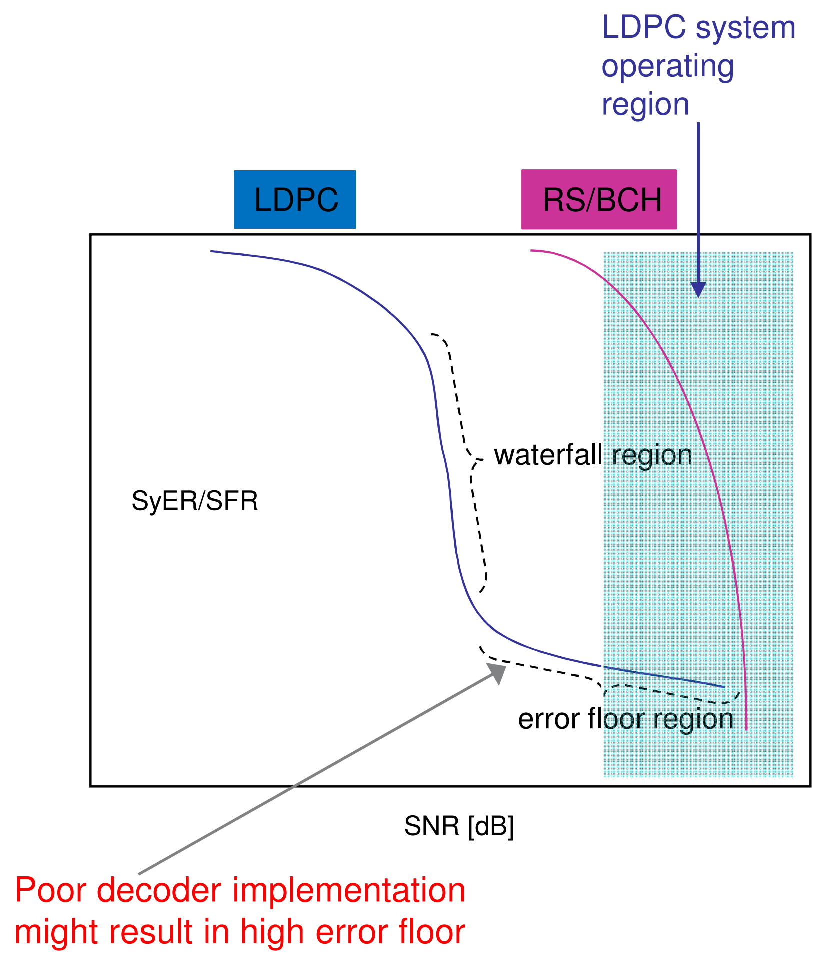
- RS ECC performance is completely determined by its correction power t (in symbols)

- For example, RS ECC with correction power t = 16 symbols.
  - This code is capable of correcting up to 2t = 32 symbols of erasure. There is no restriction on the erasure symbol locations within a sector.
  - The code is capable of correcting t = 16 symbol errors regardless of type and location.

- Sector failure rate of RS ECC keeps falling at exponential rate with SNR increase
  - No flattening of SFR vs. SNR curve is observed at higher SNR’s

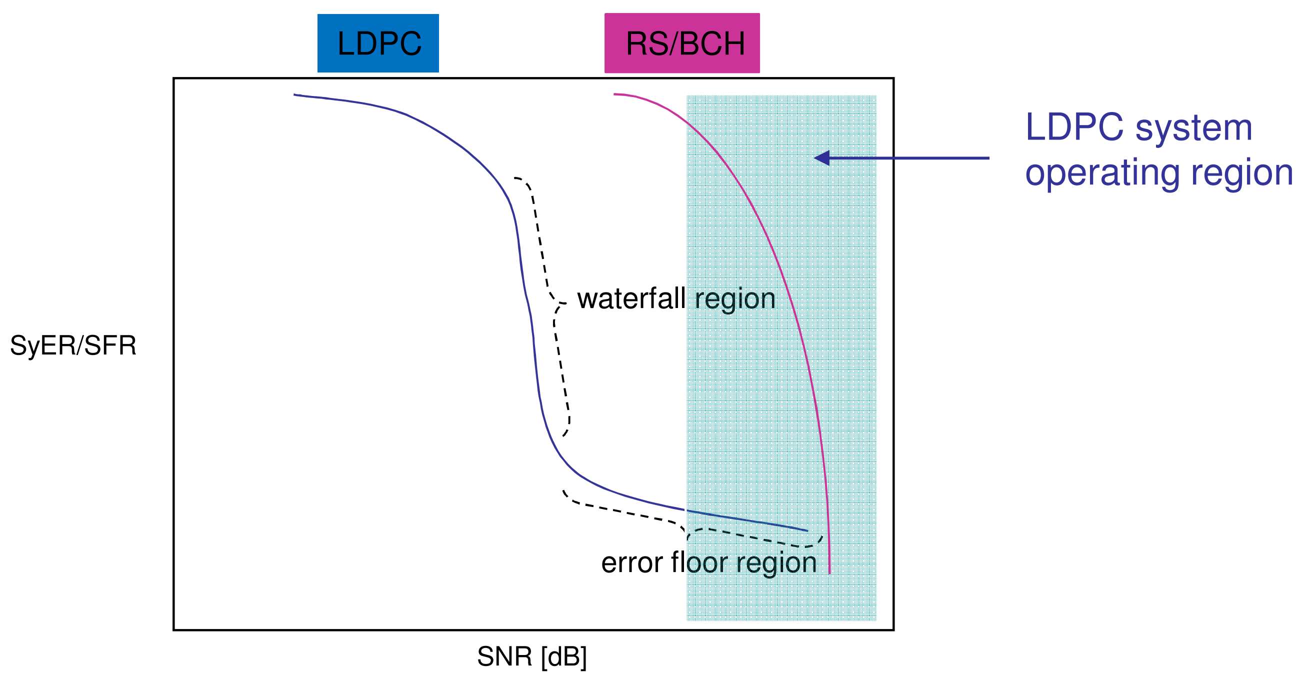
LDPC Decoder Performance Characterization

- LDPC ITR decoder correction guarantees
  - Little to no *deterministic* performance guarantees are provided by the code
  - Error correction is *probabilistic*
    - Code is capable of fixing hundreds of bit errors, but may fail (with small probability) even if there are only few bit errors present
  - Decoder implementation (e.g. quantization of messages) is just as important to the final performance as code design
    - For a fixed ITR code, the differences in decoder implementation can have significant effect on overall performance

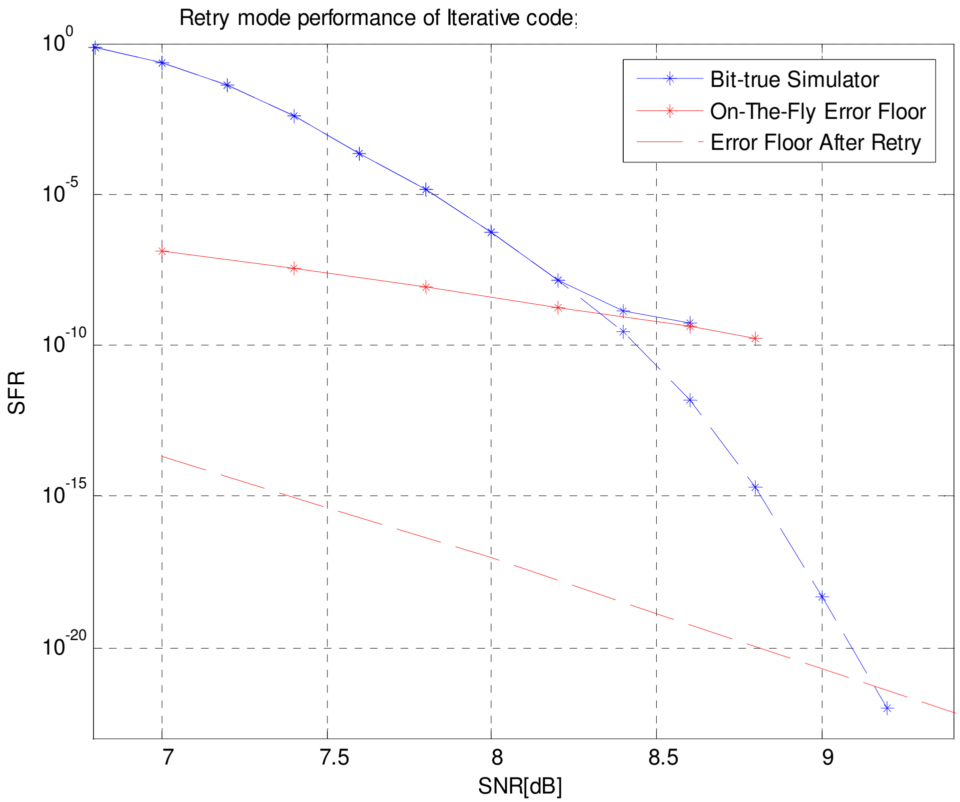


LDPC ITR Decoder Error Rate Characteristics

- Waterfall region
  - BER/SFR drops rapidly with small change in SNR
- Error Floor (EF) region (High SNR region)
  - BER/SFR drop is much slower
- Specific structures in the LDPC code graph lead to decoding errors at high SNRs
  - structures known as *Near-Codewords* (trapping sets) are dominant in the EF region



Mitigating Error Floor of LDPC Code



- Code design can be tailored to achieve the error floor bellow HER requirements

- Another strategy to push error floor to desired levels is via post-processing methodologies

Summary

- Iterative LDPC codes can enable FLASH industry to hit new capacity milestones
- Types of Decoders:
  - Hard: Bit-Flipping Decoders
  - Soft: Sum-Product (Min-Sum) Decoders
- Soft message passing decoders offer large SNR gains – this translates to capacity gains
- Optimized ITR codes/decoders are known to deliver performance near the theoretical limits in the channels dominated by random noise, e.g. AWG noise
- Handling the error floor phenomenon in ITR decoders
  - Code matrix design
  - Decoder design
  - Post-processing

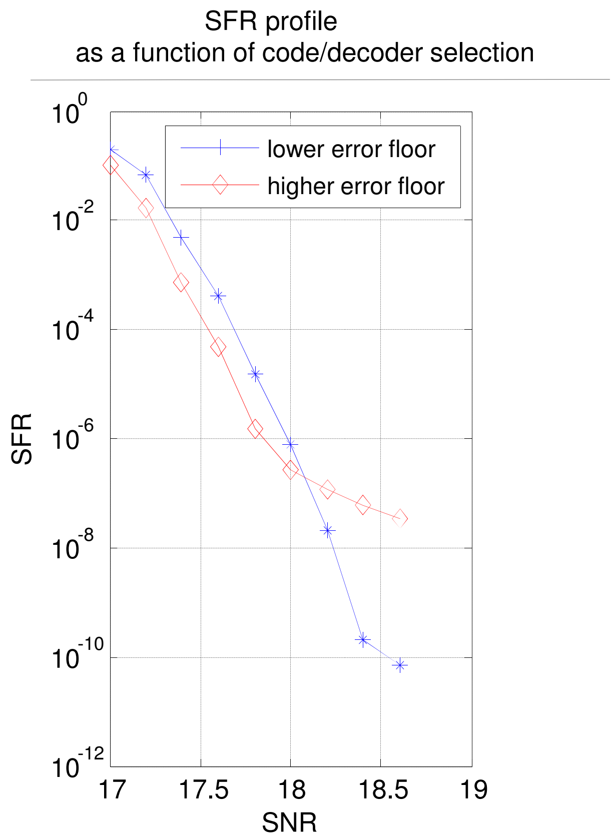
APPENDIX

LDPC Iterative Error Rate Characteristics

- In “high-SNR” region, dominant errors are near-codewords (trapping sets)
- As the name suggests, near-codewords look similar to true codewords.
  - More precisely they have low syndrome weight – violating only few of the parity check equations
  - Recall that a valid codeword has syndrome weight of 0

- Iterative decoder gets trapped into one of NCW’s, and is unable to come out of this steady state (or oscillating state)
  - Even if decoder has time to run 100’s of iterations, it would not be able to come out of the trapping set

Code Selection



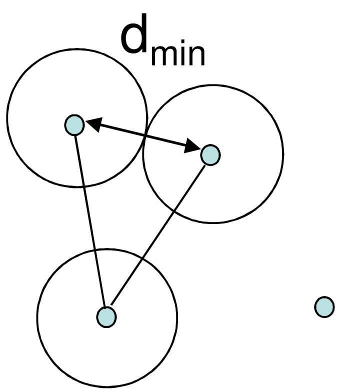
- Optimizing code/decoder selection based on the performance at low SNR's only may lead to impractical code selection.

- There is a trade-off to be made between performance at low SNR's, defect correction, and error floor (performance at high SNR's)

Mis-Correction in LDPC Decoding

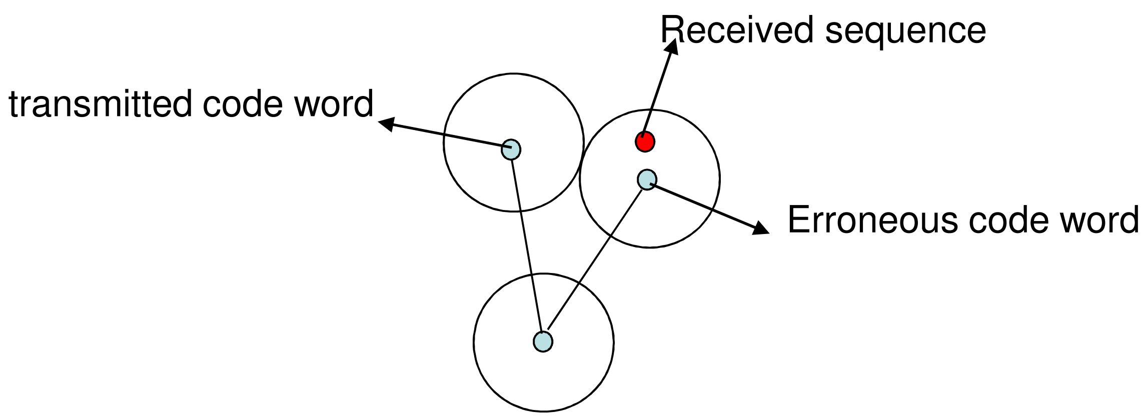
Minimum Distance of a Code

- The minimum of all the distance between any two code words is called the minimum distance of the code, denoted by d_{min}



Decoder Miscorrection

- Miscorrection: For an error correcting code, when the received sequence falls into the decoding area of an erroneous code word, the decoder may deliver the wrong code word as the decoding result.



Decoder Error

Iterative Error Rate Characteristics

- Production grade devices will operate in the Error Floor region (High SNR region)
  - Dominant Error Events in error floor region are near-codewords
  - Mis-correction is much lower probability event than dominant near-codewords