Lower-Complexity Layered Belief-Propagation Decoding of LDPC codes

Yuan-Mao Chang
Andres I. Vila Casado
Advisor: Prof. Wesel
8/11/2008

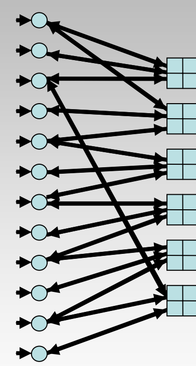
Communication Systems Laboratory
Department of Electrical Engineering
University of California—Los Angeles

Outline

- Introduction
- Efficient computation of the check-to-variable messages
- Variable-node-centric LBP (V-LBP)
- Zigzag LBP (Z-LBP)
- High-Rate LDPC codes
  - Check-node-centric LBP (C-LBP) implementation issues
  - Partially-parallel Z-LBP implementation
- Conclusion
- Further work
- Reference

Introduction

- The original message-passing schedule: Flooding scheduling
- Sequential scheduling:
  - improves the convergence speed in terms of number of iterations
  - outperforms the traditional flooding scheduling for a large number of iterations.
- Different types of sequential schedules:
  - a sequence of check-node updates [3]
  - a sequence of variable-node updates [7] [8]
  - Layered Belief Propagation (LBP) [9]
  - Others [10], [11], and [12]



Quasi-Cyclic LDPC codes

- QC-LDPC codes are represented as $H_{QC} = \begin{bmatrix} A_{1,1} & \cdots & A_{1,t} \\ \vdots & & \vdots \\ A_{s,1} & \cdots & A_{s,t} \end{bmatrix}$

  where each sub-matrix $A_{i,j}$ is a pxp circulant matrix.
- A circulant matrix is a square matrix in which each row is a one-step cyclic shift of the previous row, and the first row is a one-step cyclic shift of the last row.
- The QC-LDPC structure guarantees that at least messages can be computed in a parallel fashion at all times if flooding schedule is used [3].
- QC-LDPC decoders have a significantly higher throughput than the decoders of random sparse matrices [13].
- Well designed QC-LDPC codes perform as well as random sparse matrices [14].

The check-to-variable messages

- The message from check node c_i to variable node v_j

$$m_{c_i \to v_j} = \prod_{v_b \in N(c_i) \backslash v_j} \operatorname{sgn}\left(m_{v_b \to c_i}\right) \times \phi\left(\sum_{v_b \in N(c_i) \backslash v_j} \phi\left(\left| m_{v_b \to c_i} \right|\right) \right)$$

where $N(c_i) \backslash v_j$ denotes the neighbors c_i of excluding v_j

$$\phi(x) = -\log\left(\tanh\left(\frac{x}{2}\right)\right)$$

Soft-XOR

- A good property of the $\phi(x)$ function [1]

$$\phi(x) = \phi^{-1}(x)$$

- Consider a binary operator: Soft XOR

$$x \boxplus y \equiv \phi\left(\phi(x) + \phi(y)\right)$$

- Apply the Jacobian algorithm [2]

$$\begin{aligned}\log\left(e^x + e^y\right) &= \max(x, y) + \log\left(1 + e^{-|x-y|}\right) \\ &= \max(x, y) + \delta(|x - y|)\end{aligned}$$

$$\delta(|x - y|) \approx \max\left(\frac{5 - 2|x - y|}{8}, 0\right) \quad [3]$$

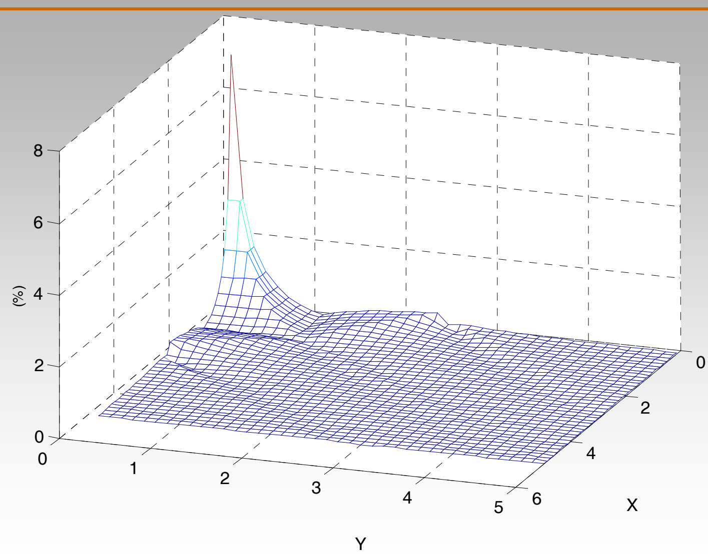
Implementation of Soft-XOR

❑ Soft-XOR can be implemented in practice [4]

$$
\begin{aligned}
x \boxplus y &= \phi(\phi(x)+\phi(y)) \\
&= \log\left(1+e^{-(x+y)}\right)-\log\left(e^{-x}+e^{-y}\right) \\
&= [\max((x+y),0)+\delta(x+y)]-[\max(x,y)+\delta(|x-y|)] \\
&= \min(x,y)+\delta(x+y)-\delta(|x-y|) \\
&\approx \min(x,y)+\max\left(\frac{5-2|x+y|}{8},0\right)-\max\left(\frac{5-2|x-y|}{8},0\right)
\end{aligned}
$$

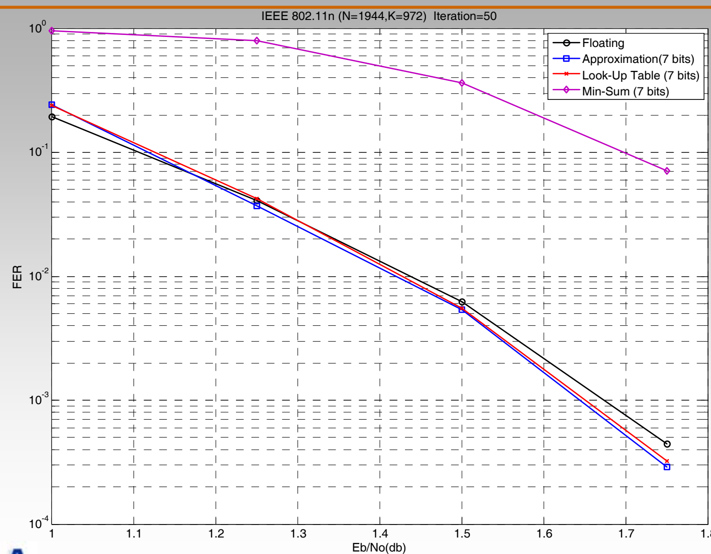
UCLA Communication Systems Laboratory

Simulation result



UCLA Communication Systems Laboratory

Simulation result



UCLA Communication Systems Laboratory

The check-to-variable messages

- By applying Soft-XOR, the message from c_i to v_j becomes

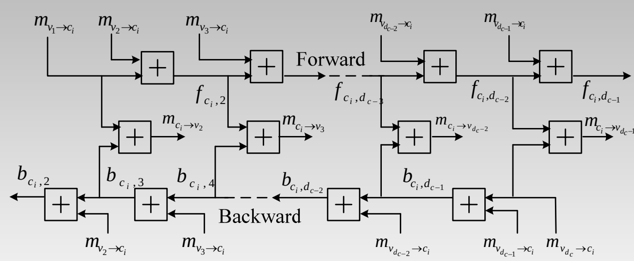
$$m_{c_i \to v_j} = \prod_{v_b \in N(c_i) \backslash v_j} \text{sgn}\left(m_{v_b \to c_i}\right) \times \boxplus_{v_b \in N(c_i) \backslash v_j} m_{v_b \to c_i}$$

- $d_c - 2$ Soft-XORs are required to compute each $m_{c_i \to v_j}$
- $d_c(d_c - 2)$ Soft-XORs are required to separately compute all the $m_{c_i \to v_j}$ from the same check node c_i

UCLA Communication Systems Laboratory

Efficient implementation

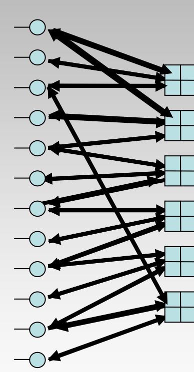
- If a message-passing schedule requires the decoder to compute all the $m_{c_i \to v_j}$ from the same c_i simultaneously, there is an efficient implementation [5].



- This method uses $3(d_c - 2)$ Soft-XORs to correctly compute all the $m_{c_i \to v_j}$ from the same c_i at the same time.
- This check-node update is equivalent to the BCJR algorithm [6] over the trellis representation of the check-node equation in the log-likelihood domain.

Variable-node-centric LBP

- The V-LBP solutions proposed in [7] and [8] have a higher complexity per iteration than flooding and C-LBP.
- The V-LBP algorithm sequentially updates variable nodes, it does not allow computing all the $m_{c_i \to v_j}$ from the same c_i at the same time.
- Hence, the efficient implementation technique can not be applied here.
- V-LBP requires $d_c(d_c - 2)$ Soft-XORs to compute all the $m_{c_i \to v_j}$ from the same c_i .



Zigzag LBP

- Zigzag LBP (Z-LBP) is a V-LBP strategy that performs variable-node updates in a zigzag pattern over the parity-check matrix.

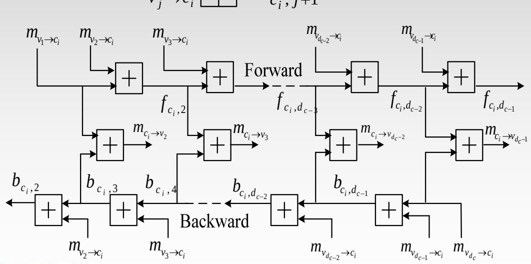
- Z-LBP schedule that requires fewer number of operations per iteration than flooding, C-LBP, and V-LBP to compute all the $m_{c_i \to v_j}$.

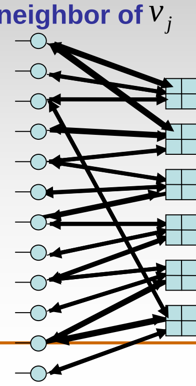
- Zigzag updating guarantees that all the $m_{c_i \to v_j}$ can be generated by using the technique presented before.

Communication Systems Laboratory

Z-LBP

- 1. Initialize all $f_{c,j}$ of the every check node.
- 2. All the odd iterations, consists of the sequential update of variable nodes, in a backward fashion. All the $m_{c_i \to v_j}$ are generated using $f_{c_i,j-1} \boxplus b_{c_i,j+1}$.
- 3. Generate all the $m_{v_j \to c_i}$ from the same v_j .
- 4. Calculate all the $b_{c,j}$ for every c_i that is a neighbor of v_j using $m_{v_j \to c_i} \boxplus b_{c_i,j+1}$.





Communication Systems Laboratory

Z-LBP

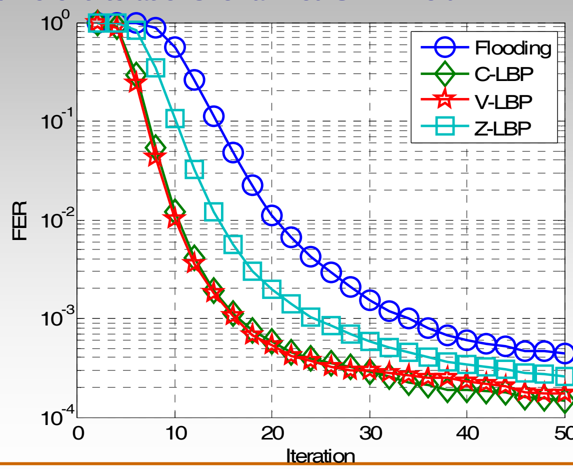
- Z-LBP requires $2(d_c - 2)$ Soft-XORs in order to update a check node.
- Flooding and C-LBP require $3(d_c - 2)$ Soft-XORs to update a check node
- V-LBP needs $d_c(d_c - 2)$ Soft-XORs to update a check node.
- Assume the complexity of computing check-to-variable messages is much higher than the complexity of computing variable-to-check messages [3] [5].
- Z-LBP is 1.5 times simpler than flooding and C-LBP and $d_c/2$ times simpler than V-LBP per iteration.

Memory Size

- Denote the number of the edges of the bi-partite graph as N_E .
- There are N_E $f_{c,j}$ values and N_E $b_{c,j}$ values. Hence, this suggests that the Z-LBP decoder needs a memory of size $2N_E$.
- However, in the case of an odd iteration, the decoder computes a new $b_{c,j}$ after updating $m_{v_j \to c_i}$. Thus, the new $b_{c,j}$ can be written in the same memory address of $f_{c,j}$
- Therefore, the required memory size is only N_E .
- This is the same memory size required for a C-LBP decoder which is half the memory required for a flooding decoder.

Simulation result

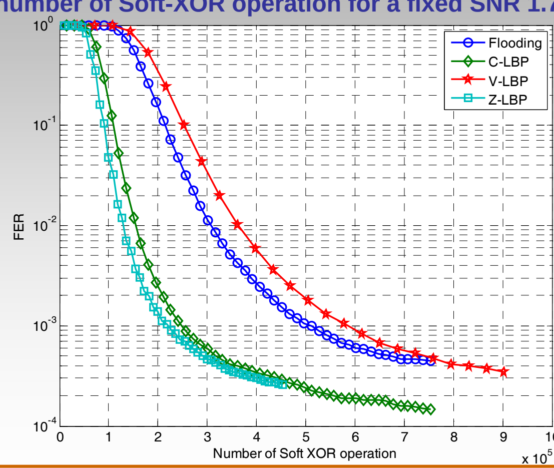
- IEEE 802.11n, Rate-1/2, Blocklength-1944, QC-LDPC code
- FER at different iterations for a fixed SNR 1.75 dB



UCLA *Communication Systems Laboratory*

Simulation result

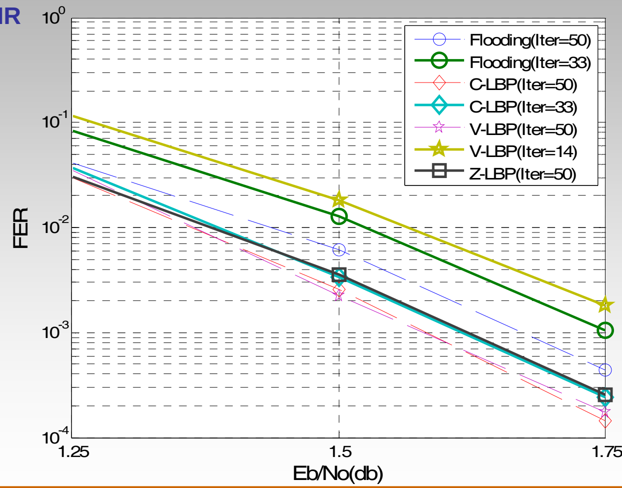
- IEEE 802.11n, Rate-1/2, Blocklength-1944, QC-LDPC code
- FER v.s. number of Soft-XOR operation for a fixed SNR 1.75 dB



UCLA *Communication Systems Laboratory*

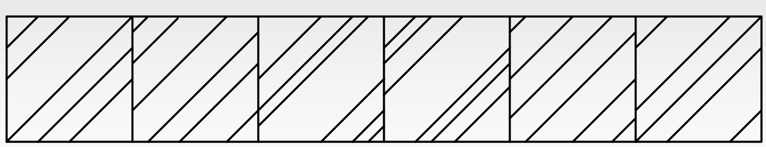
Simulation result

- IEEE 802.11n, Rate-1/2, Blocklength-1944, QC-LDPC code
- FER v.s. SNR



High-Rate LDPC codes

- Small-to-medium blocklength high-rate QC-LDPC codes generally need more than one diagonal per sub-matrix and only allow one row of sub-matrices.
- The single row of sub-matrices is necessary because multiple rows would require the sub-matrix size to be too small to provide the necessary throughput.



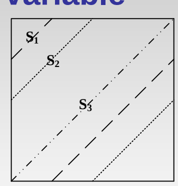
Check-node-centric LBP implementation issues

- Partially-parallel C-LBP implementation
- Step 3 and 4 become the variable-node update and check-node update of the flooding scheduling respectively.
- Therefore, partially-parallel C-LBP becomes exactly the same as flooding in complexity, convergence speed, and decoding capability.

```
1: Initialize all m_{c_i→v_j} = 0
2: for every row of sub-matrix l do
3:    Generate and propagate m_{V→C_l}
4:    Generate and propagate m_{C_l→V}
5: end for
6: if Stopping rule is not satisfied then
7:    Position = 2;
8: end if
```


Partially-parallel Z-LBP implementation

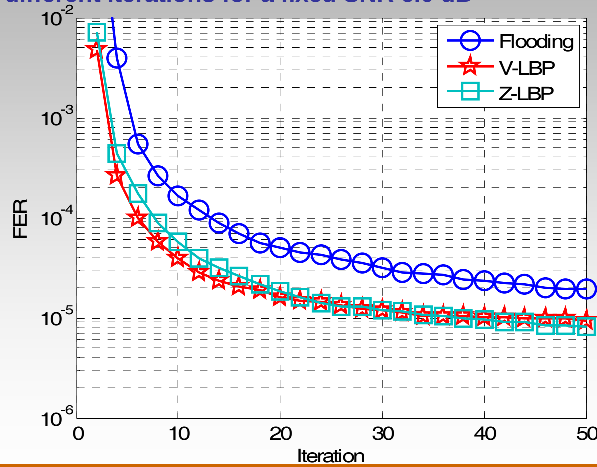
- Z-LBP can perform in a partially-parallel fashion by updating a column of sub-matrices.
- First, label the cyclic-shift diagonals in each sub-matrix. This labeling prevents memory access conflicts when all processors process *p* variable nodes at the same time.



- The other steps are still the same.
- However, the decoder requires extra $d_c - N_{mat}$ Soft-XORs in order to compute $f_{c,j}$ or $b_{c,j}$ in advance.

Simulation result

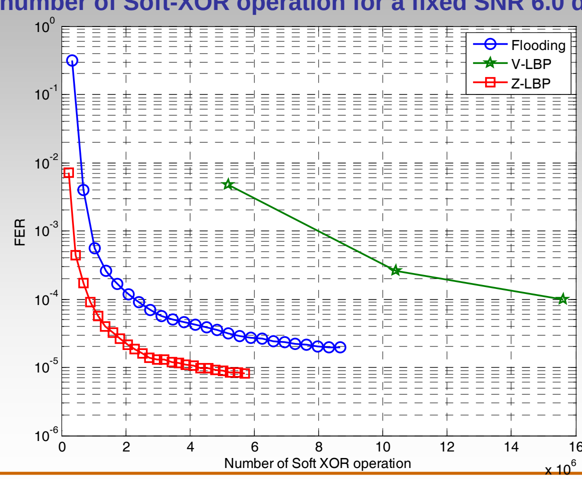
- IEEE 802.15.3c, Rate-14/15, Blocklength-1440, QC-LDPC code
- FER at different iterations for a fixed SNR 6.0 dB



UCLA *Communication Systems Laboratory*

Simulation result

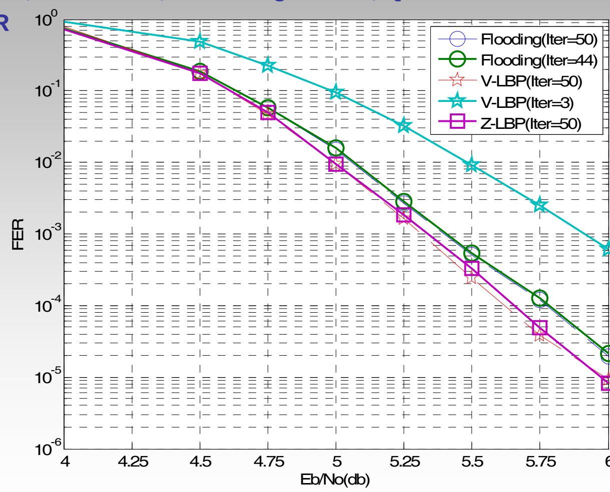
- IEEE 802.15.3c, Rate-14/15, Blocklength-1440, QC-LDPC code
- FER v.s. number of Soft-XOR operation for a fixed SNR 6.0 dB



UCLA *Communication Systems Laboratory*

Simulation result

- IEEE 802.15.3c, Rate-14/15, Blocklength-1440, QC-LDPC code
- FER v.s. SNR



Conclusion

- Z-LBP is a low-complexity sequential schedule of variable node updates.
- Z-LBP is $d_c/2$ times simpler than that of V-LBP and also 1.5 times simpler than flooding and C-LBP.
- Z-LBP outperforms flooding with a faster convergence speed and better decoding capability.
- For QC-LDPC codes where the sub-matrices can have at most one "1" per column and one "1" per row, Z-LBP can perform partially-parallel decoding. It provides the same performance as C-LBP.
- For small-to-medium blocklength high-rate QC-LDPC codes, Z-LBP can still perform partially-parallel decoding and maintains a sequential schedule.

Further Work

- FPGA implementation of C-LBP and Z-LBP
- Target code:
  - IEEE 802.11n, Rate-1/2, Blocklength-1944, LDPC code
  - IEEE 802.15.3c, Rate-14/15, Blocklength-1440, LDPC code
- Challenges
  - Place and Route
  - High throughput
  - Memory access
  - Power
- Building universal QC-LDPC decoders

Reference

[1] R. G. Gallager, "Low density parity check codes," *IRE Trans. Inform Theroy*, vol. IT-8, pp.21-28, Jan. 1962.

[2] P. Roberston, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithm operating in the log domain," in *Proc. IEEE Int. Conf. Communications,* pp. 1009-1013, 1995.

[3] M.M. Mansour and N.R. Shanbhag, "High-throughput LDPC decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, Issue 6, Dec. 2003 pp. 976 – 996

[4] C Jones, S. Dolinar, K. Andrews, D. Divsalar, Y. Zhang, and W. Ryan,. Functions and Architectures for LDPC Decoding. In *IEEE ITW 2007*, pages 577--583, Lake Tahoe, California, September 2007

[5] X.Y. Hu, E. Eleftheriou, D.M. Arnold, and A. Dholakia. Efficient Implementations of the Sum-Product Algorithm for Decoding LDPC Codes. In *Proc. IEEE Global Conference on Communications*, pages 1036--1036E, San Antonio, TX, November 2001.

[6] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate. *IEEE Trans. Info. Theory*, IT-20:284--287, March 1974.

[7] H. Kfir and I. Kanter. Parallel versus sequential updating for belief propagation decoding. *Physica A*, 330:259--270, 2003.

Reference

[8] J. Zhang and M. Fossorier. Shuffled belief propagation decoding. *IEEE Trans. on Commun.*, 53:209--213, February 2005.

[9] D. Hocevar. A reduced complexity decoder architecture via layered decoding of LDPC codes. In *Proc. Signal Processing Systems SIPS 2004*, pages 107--112, October 2004.

[10] E. Sharon and S. Litsyn and J. Goldberger. An efficient message-passing schedule for LDPC decoding. In *Proc. 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, pages 223--226, September 2004.

[11] P. Radosavljevic, A. de Baynast, and J. R. Cavallaro. Optimized message-passing schedules for LDPC decoding. In *Proc. Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*, pages 591--595, 2005.

[12] A. I. Vila Casado, M. Griot, and R. D. Wesel. Informed dynamic scheduling for belief-propagation decoding of LDPC codes. In *IEEE ICC 2007*, pages 932--937, Glasgow, Scotland, July 2007.

[13] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong. Efficient encoding of low-density parity-check codes. *IEEE Trans. on Commun.*, 54:71--81, January 2006.

[14] L. Chen, J. Xu, I. Djurdjevic, and S. Lin. Near Shannon limit quasi-cyclic low-density parity-check codes. *IEEE Trans. On Commun.*, 52(7):1038--1042, July 2004.

Thank you for your attention.